การออกแบบวงจรสื่อสารอนุกรมด้วยภาษา VHDL

SERIAL COMMUNINCATION COMPONENT DESIGN USING VHDL

โดย

นางสาว ทิพวรรณ อรัญวัฒนานนท์

นางสาว สมพิศ อภิวัฒน์อุดมคุณ

อาจารย์ที่ปรึกษา

อาจารย์ อภิเนตร อูนากูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ปีการศึกษา 2541

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การออกแบบวงจรสื่อสารอนุกรมด้วยภาษา VHDL

SERIAL COMMUNINCATION COMPONENT DESIGN USING VHDL

คณะผู้จัดทำ

นางสาว ทิพวรรณ อรัญวัฒนานนท์ รหัสประจำตัว 38014180

นางสาว สมพิศ อภิวัฒน์อุดมคุณ รหัสประจำตัว 38014533

..... อาจารย์ที่ปรึกษา

(อาจารย์ อภิเนตร อูนากูล)

การออกแบบวงจรสื่อสารอนุกรมด้วยภาษา VHDL

นางสาว ทิพวรรณ อรัญวัฒนานนท์ 38014180
นางสาว สมพิศ อภิวัฒน์อุดมคุณ 38014533
อาจารย์ อภิเนตร อูนากูล อาจารย์ที่ปรึกษา
ปีการศึกษา 2541

บทคัดย่อ

ในปัจจุบันระบบดิจิตอลมีความซับซ้อนมากขึ้น ระบบต่างๆจำเป็นต้องมีการสื่อสารระหว่างอุปกรณ์หลายชนิด และการเชื่อมต่ออุปกรณ์เหล่านี้มักจะใช้การสื่อสารแบบอนุกรม เพราะใช้สายในการสื่อสารน้อยกว่าแบบขนานทำให้ไม่เปลืองสายสื่อสารและเนื้อที่ในการออกแบบ ดังนั้นในการทำโครงงานนี้คณะผู้จัดทำจึงเลือกที่จะออกแบบวงจรดิจิตอลที่ใช้ในการสื่อสารแบบอนุกรม โดยเลือกทำการออกแบบการสื่อสารแบบ UART (Universal Asynchronous Receiver/Transmitter) ซึ่งอุปกรณ์ UART เป็นชิพที่ใช้ในการสื่อสารอนุกรมแบบอะซิงโครนัสในระบบคอมพิวเตอร์ทั่วไป ที่สามารถทำการโปรแกรมได้และมีโปรโตคอลในการสื่อสารแบบอนุกรมที่ง่ายไม่ซับซ้อนมาก แต่มีประสิทธิภาพสูง และเป็นรูปแบบการสื่อสารที่ได้รับความนิยมในปัจจุบัน ปริญญานิพนธ์ฉบับนี้เรียบเรียงขึ้นจากวิธีการสร้างระบบดิจิตอลให้มีการทำงานเหมือนกับชิพ UART โดยแสดงขั้นตอนการออกแบบระบบ การทดสอบระบบที่ได้ออกแบบไว้ทั้งในระดับซอฟต์แวร์โดยการจำลองการทำงานและในระดับฮาร์ดแวร์โดยใช้ FPGA

SERIAL COMMUNICATION COMPONENT DESIGN USING VHDL

Tippawan Aranwattananon

Sompit Aphiwatudomkun

Apinetr Unakul (Advisor)

ABSTRACT

Today, Complexity of digital system has increased in a way that the system is composed of a number of peripheral devices. These devices are usually connected via serial communication because it requires less number of wires which results in an optimized design in terms of space required. In this project, we designed and implemented the firm core of serial communication called Universal Asynchronous Receiver/Transmitter (UART). The firm core is reusable component containing more structure, commonly a gate-level netlist that is ready for placement and routing. This thesis illustrates steps in digital system design and test at software level (by simulation) and hardware level (by FPGA).

กิตติกรรมประกาศ

ขอขอบพระคุณบุคคลที่มีรายนามดังต่อไปนี้ ที่ให้ความช่วยเหลือในการทำโครงงานครั้งนี้จนประสบความสำเร็จ

1. อาจารย์ อภิเนตร อูนากูล
   ให้โอกาสในการทำงานที่น่าสนใจ
   ให้คำปรึกษา แนวทางในการทำโครงงานและดูแลการทำงานอย่างต่อเนื่อง
2. คุณธนพร ทองเผือก
   สละเวลาในการทำงานมาให้คำปรึกษาและแนะนำ
3. คุณเกรียงไกร ปทุมพร
   สละเวลาในการทำงานมาให้คำปรึกษาและแนะนำ
   เอื้อเฟื้ออุปกรณ์ในการทดสอบ
4. คุณเจริญ วงษ์ชุ่มเย็น
   ให้คำปรึกษาและแนะนำในการทดสอบ
5. คุณปัญจภา เรืองสินทรัพย์
   ให้คำปรึกษาและแนะนำในการออกแบบ
6. คุณวรรณรัช สันติอมรทัต
   ให้คำปรึกษาและแนะนำในการทำโครงงาน
7. คุณสุรศักดิ์ สุวรรณคร
   ให้คำปรึกษาและเอื้อเฟื้อข้อมูลในการทำโครงงาน
8. คุณฐาปนี ศรีช่วง
   เอื้อเฟื้อข้อมูลในการทำโครงงาน
9. ห้องปฏิบัติการ Embedded System Lab (ESL)
   เอื้อเฟื้อสถานที่และอุปกรณ์ในการทำโครงงาน
10. NECTEC
    เอื้อเฟื้อ FPGA ที่ใช้ในการทดสอบ
11. ภาควิชาวิศวกรรมคอมพิวเตอร์

สุดท้ายขอขอบคุณพี่ๆ และเพื่อนๆ ที่ไม่ได้เอ่ยนามในที่นี้ ที่คอยเป็นกำลังใจให้ทำโครงงานนี้สำเร็จ

ทิพวรรณ อรัญวัฒนานนท์
สมพิศ อภิวัฒน์อุดมคุณ

สารบัญ

สารบัญรูปภาพ

หน้าที่

สารบัญตาราง

หน้าที่

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

ในปัจจุบันเทคโนโลยีได้ก้าวหน้าไปอย่างรวดเร็ว พัฒนาการทางด้านการติดต่อสื่อสารและการใช้งานระบบดิจิตอลมีความซับซ้อนมากขึ้น ในระบบต้องมีการสื่อสารกันระหว่างอุปกรณ์หลายชนิด เช่น การต่อไมโครคอนโทรลเลอร์กับ Serial EEPROM, LCD display, A/D converter หรืออุปกรณ์อื่น ๆ เพื่อใช้งานในแอพพลิเคชั่นต่าง ๆ และในการเชื่อมต่ออุปกรณ์เหล่านี้มักจะใช้การสื่อสารแบบอนุกรม เพราะใช้สายในการสื่อสารน้อยกว่าแบบขนานทำให้ไม่เปลืองสายสื่อสารและเนื้อที่ในการออกแบบ (เช่น การทำ PCB) อีกทั้งยังมีความสะดวกกว่าการสื่อสารแบบขนาน ดังนั้นในการทำโครงงานนี้คณะผู้จัดทำจึงเลือกที่จะออกแบบวงจรดิจิตอลที่ใช้ในการสื่อสารแบบอนุกรม โดยเลือกทำการออกแบบการสื่อสารแบบ UART (Universal Asynchronous Receiver/Transmitter) ซึ่งอุปกรณ์ UART เป็นชิปที่ใช้ในการสื่อสารอนุกรมแบบอะซิงโครนัสในระบบคอมพิวเตอร์ทั่วไปที่สามารถทำการโปรแกรมได้ โดยรูปแบบของข้อมูลจะประกอบด้วยบิทเริ่มต้น (start bit), บิทข้อมูล 8 บิท, บิทพาริตี้ (parity bit) และบิทสิ้นสุด (stop bit) และทำการรับส่งข้อมูลแบบฟูลดูเพล็กซ์ (full duplex)

นอกจากการสื่อสารแบบ UART จะสามารถทำการโปรแกรมได้แล้ว UART ยังเป็นอุปกรณ์ที่ใช้ในการสื่อสารแบบอนุกรมที่ง่ายไม่ซับซ้อนมาก แต่มีประสิทธิภาพสูง และเป็นรูปแบบการสื่อสารที่เป็นที่นิยมในปัจจุบัน

นอกจากนี้ระบบการสื่อสารอนุกรมแบบ UART ยังเป็นระบบที่มีบริษัทผู้ผลิตหลายรายทำการผลิตอุปกรณ์ขึ้นมาเพื่อเลียนแบบการทำงานของ UART บริษัทต่างๆ จึงมีการแข่งขันกันในการปรับปรุงผลิตภัณฑ์ทั้งในด้านคุณภาพ ราคา การส่งเสริมความรู้และการบริการ ดังนั้นอุตสาหกรรมซึ่งมีระบบการทำงานที่ต้องการการสื่อสารอนุกรมแบบ UART จึงมีตัวเลือกมากมายในการเลือกให้เหมาะสมกับอุตสาหกรรมของตนเองมากที่สุด

อย่างไรก็ตาม เทคโนโลยีทางด้านการติดต่อสื่อสารและการใช้งานระบบดิจิตอลยังคงมีพัฒนาการไปอย่างไม่หยุดยั้ง ดังนั้นจึงควรมีการศึกษาและติดตามข่าวสาร เพื่อจะเป็นประโยชน์ต่อการพัฒนาอุตสาหกรรมการผลิตและออกแบบอุปกรณ์อิเล็กทรอนิกส์ของประเทศ

และจากที่ได้กล่าวมาทั้งหมดนั้นจึงเป็นที่มาของการออกแบบงานวิจัยนี้ โดยมีขั้นตอนในการออกแบบเป็นลำดับ ซึ่งจะได้กล่าวต่อไป

1.2 วัตถุประสงค์ของงานวิจัย

1) ศึกษาหลักการและขั้นตอนในการออกแบบวงจรดิจิตอล
2) ศึกษาทฤษฎีเบื้องต้นของการสื่อสารแบบอนุกรมของไมโครคอนโทรลเลอร์ในรูปแบบของ UART (Universal Asynchronous Receiver/Transmitter) และทฤษฎีเบื้องต้นของอุปกรณ์ FPGA (Field Programmable Gate Arrays)
3) ทำการออกแบบวงจรโดยใช้ทฤษฎีเบื้องต้นของ UART เพื่อให้ได้เฟิร์มคอร์ (Firm Core) ที่สามารถทำงานเลียนแบบการทำงานของอุปกรณ์ UART ด้วยการใช้ภาษา VHDL เป็นตัวอธิบายการทำงานและการออกแบบ
4) ทำการทดสอบการทำงานของโปรแกรมที่ออกแบบโดยใช้อุปกรณ์ FPGA เพื่อทดลองรับ-ส่งข้อมูลแบบ UART

1.3 ขอบเขตของงานวิจัย

งานวิจัยนี้จะทำการออกแบบ Universal Asynchronous Receiver/Transmitter (UART) เลียนแบบการทำงานของชิพ MC6850 ACIA ซึ่งเป็น IC ของบริษัทโมโตโรล่า ให้มีความสามารถในการติดต่อสื่อสารแบบอะซิงโครนัสกับอุปกรณ์ภายนอกโดยใช้ภาษา VHDL และทดสอบการทำงานของการออกแบบโดยใช้อุปกรณ์ FPGA

1.4 วิธีการดำเนินงาน

1) งานวิจัยในโครงงานนี้จะเริ่มด้วยการศึกษาทฤษฎีพื้นฐานต่าง ๆ ที่เกี่ยวข้องกับงานวิจัย ซึ่งมีรายละเอียดดังในบทที่ 2 โดยมีเรื่องหลัก ๆ อยู่ 3 เรื่องด้วยกัน คือ
   - การศึกษาการใช้ภาษา VHDL ซึ่งใช้เป็นตัวอธิบายการทำงานของคอร์ (core) ที่ต้องการออกแบบ รวมทั้งการใช้ซอฟต์แวร์ V-System เพื่อทำการคอมไพล์และตรวจสอบสัญญาณ
   - การศึกษาลักษณะและรูปแบบการสื่อสารข้อมูลแบบอนุกรมของ UART โดยทั่วไปและลักษณะของชิพ MC6850 ACIA
   - การศึกษาการทำงานภายในของ FPGA ซึ่งใช้เป็นอุปกรณ์ทดสอบการทำงาน
2) นำความรู้ที่ได้จากการศึกษามาออกแบบวงจรการสื่อสาร โดยเริ่มจากการออกแบบบล็อกไดอะแกรม (Block diagram) แล้วจึงออกแบบการทำงานภายในและการเชื่อมต่อแต่ละบล็อก
3) นำสิ่งที่ได้ออกแบบเสร็จเรียบร้อยแล้วมาสร้างการทำงานของระบบ โดยเริ่มด้วยการเขียนโปรแกรมด้วยภาษา VHDL เป็นขั้นตอนตามที่ได้ออกแบบไว้ หลังจากนั้นก็ทำการทดสอบการทำงานแต่ละขั้นตอน และทำการรวมแต่ละส่วนการทำงานเข้าด้วยกันเป็นระบบใหญ่แล้วทดสอบการทำงานรวมอีกครั้งหนึ่ง ด้วยการใช้โปรแกรม Test Bench ทำการตรวจสอบความถูกต้องในการทำงานของคำสั่งและการรับ-ส่งข้อมูลต่างๆ ของระบบ โดยดูผลการ

ตรวจสอบจากกราฟที่ได้จากการจำลองการทำงานด้วย V-System เทียบกับการทำงานของเทคโนโลยีของ UART จริง

4) เมื่อผ่านการทดสอบและตรวจสอบความถูกต้องเรียบร้อยแล้ว จึงนำมาสร้างลงบน FPGA เพื่อทดสอบการทำงานรับ-ส่งข้อมูลกับอุปกรณ์ภายนอกอีกทีหนึ่ง จึงจะทราบว่าสามารถนำมาใช้งานจริงได้หรือไม่ ซึ่งก็คือการทดสอบในระดับฮาร์ดแวร์นั่นเอง

1.5 ประโยชน์ที่ได้รับ

1) เข้าใจการใช้งานภาษา VHDL ซึ่งเป็นภาษาที่ใช้อธิบายการทำงานของฮาร์ดแวร์ที่ใช้กันแพร่หลาย และการใช้งาน Model Sim ซึ่งเป็นซอฟต์แวร์สำหรับทดสอบการออกแบบด้านฮาร์ดแวร์
2) เข้าใจลักษณะการทำงานของการสื่อสารอนุกรมแบบ UART เพื่อประโยชน์ในด้านการนำไปใช้งานและการพัฒนาระบบต่อไป
3) มีการพัฒนาในด้านกระบวนการคิดและการทำงาน รู้จักการจัดลำดับขั้นตอนการทำงาน การวางแผนงาน การตรวจสอบการทำงานและแนวทางในการแก้ปัญหาที่เกิดขึ้น

บทที่ 2
ทฤษฎีเบื้องต้น

2.1 ทฤษฎีพื้นฐานของ FPGA

2.1.1 Field Programmable Gate Arrays & Gate Arrays

ฟิลโปรแกรมเมเบิลเกตอาเรย์ (Field Programmable Gate Arrays) เป็นชิพชนิด ASIC (Application-Specific IC chips) ที่ใช้ในการสร้างวงจรลอจิกในอุปกรณ์ต่างๆ หลายประเภท ไม่ว่าจะเป็นอุปกรณ์ที่ใหญ่ตั้งแต่ซุปเปอร์คอมพิวเตอร์จนถึงเครื่องมือขนาดเล็กแค่มือ หรือตั้งแต่ระบบมิไซล์นำวิถีที่สำคัญไปจนถึงระบบควบคุมกีตาร์ธรรมดา

FPGA เป็นอุปกรณ์ที่มีสถาปัตยกรรมคล้ายเกตอาเรย์ (Gate Arrays) แต่มีประสิทธิภาพสูงกว่าเกตอาเรย์ธรรมดามาก ซึ่งสามารถเปรียบเทียบข้อแตกต่างได้ดังนี้

- FPGA สามารถออกแบบและทดสอบการทำงานได้อย่างรวดเร็วในเวลาเพียง 2-3 วัน ซึ่งถ้าใช้เกตอาเรย์อาจต้องใช้เวลาหลายสัปดาห์
- FPGA เป็นอุปกรณ์ที่คอนฟิกการทำงานโดยใช้ซอฟต์แวร์ (software-configured) และผู้ใช้สามารถโปรแกรมการทำงานได้เอง (user-programmed) ดังนั้นจึงสามารถเปลี่ยนแปลงโปรแกรมการทำงานได้ทุกเวลา และใช้เวลาไม่นานในการเปลี่ยนแปลงโปรแกรมเมื่อเทียบกับเกตอาเรย์ที่ต้องใช้เวลาเป็นสัปดาห์ ทำให้ลดค่าใช้จ่ายในการออกแบบและการผลิตไปได้มาก
- การออกแบบผลิตภัณฑ์ด้วย FPGA จะช่วยลดเวลาการส่งตัวผลิตภัณฑ์ออกสู่ตลาด ซึ่งเป็นประโยชน์ต่อการผลิตวงจรรวมเป็นอย่างมาก นักออกแบบเพียงแต่ออกแบบฟังก์ชันการทำงานของวงจรในรูปข้อความอธิบายหรือในรูปแผนภาพวงจร (schematic format) โดยไม่ต้องทำปรินท์บอร์ด (printed circuit board)

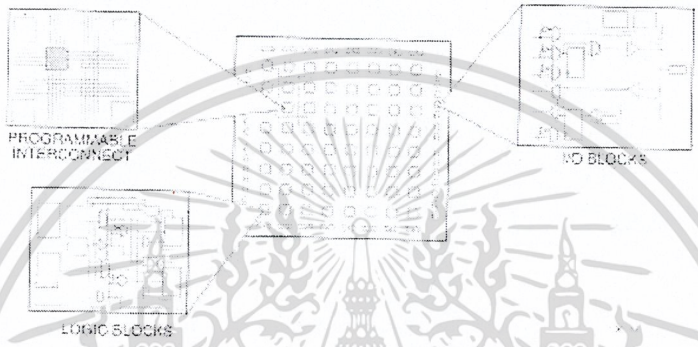
จากที่กล่าวมาจะเห็นได้ว่าการนำ FPGA มาช่วยในการออกแบบจะช่วยในการประหยัดค่าใช้จ่ายเป็นอย่างมาก เพราะจะลดความเสี่ยงในการที่จะต้องแก้ไขตัววงจร และลดการเลื่อนเวลาการส่งผลิตภัณฑ์ออกสู่ตลาดด้วย

2.1.2 Field Programmable Gate Arrays (FPGAs)

FPGA เป็นชิพที่มีปริมาณความหนาแน่นของเกตสูง สามารถกำหนดฟังก์ชันการทำงานได้ตามความต้องการของผู้ใช้ จากรูปที่ 2-1 เป็นสถาปัตยกรรมของ FPGA ประกอบด้วยเมตริกของลอจิก บล็อก (Logic blocks) ล้อมรอบด้วยอินพุท/เอาท์พุทบล็อก (I/O blocks) และมีการเชื่อมต่อลอจิกแต่ละเซลล์ด้วยโปรแกรมเมเบิลสวิตช์ (programmable switches) ทำให้สามารถสร้างเครือข่ายสัญญาณ (signal nets) ติดต่อกันระหว่างเซลล์ได้ตามต้องการ

การเชื่อมต่อระหว่าง CLB และ IOB ทำได้โดยผ่านช่องทาง (Channel) ที่วางพาดผ่านระหว่างแถว (row) และคอลัมน์ (column) ซึ่งมีการทำงานเหมือนกับไมโครโปรเซสเซอร์

การเชื่อมต่อภายใน (Interconnection) จะมีการกำหนดไว้ในโปรแกรมการคอนฟิก (Configuration Program) หรือเก็บไว้ในหน่วยความจำ EPROM ภายใน FPGA และโปรแกรมจะถูกโหลดเข้าสู่ชิพเมื่อมีการจ่ายไฟ (Power-up) โดยทางคำสั่ง (Command) ส่วนความเร็วของสัญญาณนาฬิกาของระบบจะกำหนดด้วยทอกเกิลฟลิปฟลอป (Toggle flip-flop) สำหรับการประยุกต์ใช้โดยทั่วไปจะอยู่ที่ประมาณ 1 3 ถึง ½ ของทอกเกิลสูงสุด (maximum toggle gate)



รูปที่ 2-1 สถาปัตยกรรมของ FPGA

2.1.3 XC4000 Series FPGAs Features

FPGA ที่คณะผู้จัดทำนำมาใช้เป็น XC4000 FPGA ของบริษัท Xilinx ประกอบด้วยลอจิกเกต รีจิสเตอร์และ I O จำนวนมากมาย อีกทั้งยังมีความเร็วของระบบสูง FPGA ของ Xilinx ยังมีอีกหลายตระกูล เช่น static-memory-based FPGAs (SRAM-based FPGAs) รวมทั้งตระกูล XC3000, XC5000 และ XC6000

2.1.3.1 ลักษณะของ XC4000 FPGA

XC4000 FPGA ประกอบด้วยลอจิกบล็อก (Configurable Logic Blocks (CLBs)) ที่โปรแกรมได้และมีความยืดหยุ่นมาเชื่อมต่อกัน และล้อมรอบเป็นวงด้วยอินพุท/เอาท์พุทบล็อก (Input/Output Blocks (IOBs)) ที่โปรแกรมได้

อุปกรณ์ต่างๆ ทำงานโดยโหลดข้อมูลการคอนฟิก (configuration data) ลงในเซลล์หน่วยความจำภายใน แล้ว FPGA จะอ่านข้อมูลนี้ได้จาก PROM ภายนอกแบบอนุกรมหรือขนาน (ในกรณีที่อยู่ในโหมดมาสเตอร์) หรือจะเขียนข้อมูลการคอนฟิกลง FPGA จากอุปกรณ์ภายนอกเลยก็ได้ (ในกรณีที่อยู่ในโหมดสเลฟหรือโหมดเพอริเฟอรัล)

FPGA สามารถโปรแกรมลงไปใหม่ได้เรื่อยๆ ไม่จำกัด ดังนั้น FPGA จึงสามารถใช้ในการออกแบบฮาร์แวร์ที่มีการเปลี่ยนแปลงแบบไดนามิก หรือฮาร์ดแวร์ที่ต้องมีการปรับเปลี่ยนตามแอพพลิเคชัน

ของผู้ใช้ สามารถเปลี่ยนแปลงได้แม้ขณะอยู่ในระบบ นอกจากนี้ฮาร์ดแวร์ยังสามารถเปลี่ยนแปลงและอัพเดตการออกแบบได้ง่ายเหมือนซอฟต์แวร์

2.1.3.2 บล็อกพื้นฐาน

FPGA ประกอบด้วยบล็อกพื้นฐานคือ configuration logic blocks (CLBs) และ input/output blocks (IOBs)

- CLB แต่ละตัวประกอบด้วยส่วนโปรแกรมเมเบิลคอมไบเนชันลอจิก (Programmable Combination Logic Section) ใช้สร้างวงจรทางด้านบูลีนฟังก์ชันของอินพุท
- IOB แต่ละตัวประกอบด้วยอินเทอร์เฟสระหว่างแพคเก็จพินและสายสัญญาณภายใน

นอกจากนี้ยังประกอบด้วยวงจร ต่อไปนี้

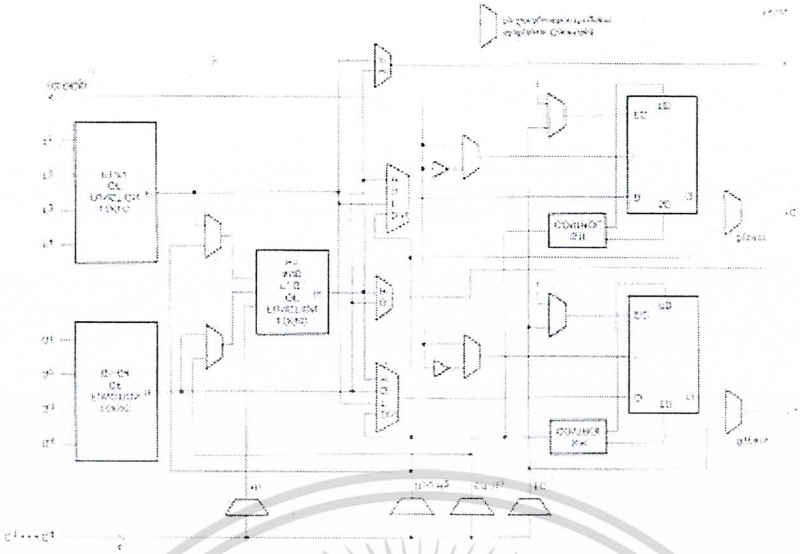
- วงจรไตรสเตทบัฟเฟอร์ (3-state buffers(TBUFs)) ใช้ขับเส้นสัญญาณตามขวางที่เชื่อมระหว่าง CLB แต่ละตัว
- ไวด์เอจดีโคดเดอร์ (Wide edge decoders) ใช้ดีโคดสัญญาณในกรณีที่ฟิลด์ข้อมูลหรือแอดเดรสมากกว่าอินพุทของฟังก์ชันเจเนอเรเตอร์ (เช่น อินพุท 9 ตัว) โดยเฉพาะในระบบไมโครโปรเซสเซอร์ใหญ่ๆ
- ออสซิลเลเตอร์แบบออนชิพ (On-chip oscillator) เป็นออสซิลเลเตอร์ภายใน ใช้เป็นแหล่งกำเนิดสัญญาณนาฬิกา (CCLK) ในโหมดการคอนฟิกแบบมาสเตอร์

❖ **Configurable Logic Blocks (CLBs)**

CLB มีโครงสร้างหลักแสดงดังรูปที่ 2-2 ประกอบด้วยอินพุทฟังก์ชันเจเนอเรเตอร์ 4 ตัว (4-input function generator) อยู่ 2 ชุด คือ ชุด F (มี F1-F4) และ G (มี G1-G4) ซึ่งแต่ละชุดสามารถนำมาใช้งานเป็นอินพุทได้ไม่จำกัดรูปแบบ โดยที่คอมไบเนชันลอจิกฟังก์ชันส่วนใหญ่จะใช้อินพุทครั้งหนึ่งอย่างมาก 4 ตัว แต่ CLB ก็มี H ฟังก์ชันเจเนอเรเตอร์ไว้สำรองด้วย ซึ่งจะมีอินพุทอยู่ 3 ตัว คือ 0, 1 และ 2 โดยอินพุทของ H ตัวที่ 0 และ 1 จะมาจากเอาท์พุทของ F และ G ส่วนตัวที่เหลือจะมาจากภายนอก CLB

CLB แต่ละตัวจะมีตัวเก็บข้อมูลอยู่ 2 ตัว ไว้เก็บเอาท์พุทของฟังก์ชันเจเนอเรเตอร์ ซึ่งจะใช้ฟลิปฟลอป (มักใช้ D-type flip-flops) เป็นตัวเก็บข้อมูล และฟังก์ชันเจเนอเรเตอร์ยังสามารถสร้างสัญญาณเอาท์พุทส่งออกได้โดยไม่ต้องเก็บในฟลิปฟลอปอีก 2 ตัว

โดยรวมแล้ว CLB มีอินพุททั้งหมด 13 ตัว (C1-C4, F1-F4, G1-G4 และ K) และเอาท์พุททั้งหมด 4 ตัว (YQ, Y, XQ และ X) ที่เป็นอินพุทและเอาท์พุทของทั้งฟังก์ชันเจเนอเรเตอร์และตัวเก็บข้อมูล ดังรูปที่ 2-2 ซึ่งอินพุทและเอาท์พุทเหล่านี้จะเชื่อมต่อกับโปรแกรมเมเบิลอินเตอร์คอนเน็ค (Programmable interconnect resource) ที่อยู่ภายนอกบล็อก



รูปที่ 2-2 บล็อกไดอะแกรมของ CLB

การใช้งานฟังก์ชันเจเนอเรเตอร์เป็น RAM

ฟังก์ชันเจเนอเรเตอร์นั้น นอกจากจะรับอินพุทแล้วส่งไปยังตัวเก็บข้อมูลภายใน CLB แล้ว ยังสามารถใช้เป็น RAM ได้ โดยจะโหมด 3 โหมดคือ level-sentitive, edge-triggered และ dual-port edge-triggered ซึ่ง CLB จะได้รับการคอนฟิกบิทอาเรย์เป็นแบบ 16x2, 32x1 หรือ 16x1 นั้นขึ้นอยู่กับโหมดที่เลือก

การที่มี RAM อยู่บนชิพเลยนั้น (on-chip RAM) ทำให้ความเร็วเพิ่มขึ้น โดยเวลาในการเข้าถึงเพื่ออ่านข้อมูลจะเท่ากับค่าดีเลย์ของลอจิก และเวลาในการเข้าถึงเพื่อเขียนข้อมูลจะช้ากว่าเล็กน้อยเท่านั้น ซึ่งเวลาในการเข้าถึงข้อมูลทั้งสองแบบจะเร็วกว่าแบบที่ไม่มี RAM อยู่บนชิพ (off-chip) เพราะจะไม่มีค่าดีเลย์จาก I/O

การคอนฟิกหน่วยความจำ (RAM) และไทม์มิ่งโหมดของ CLB สำหรับโหมดแบบ single-port และ dual-port แสดงในตารางที่ 2-1

	16x1	**16x2**	**32x1**	**Edge-Triggered Timing**	**Level-Sensitive Timing**
Single-Port	✓	✓	✓	✓	✓
Dual-Port	✓	-	-	✓	-

ตารางที่ 2-1 โหมดแสดงพอร์ตของ RAM

RAM Configuration Options

ฟังก์ชันเจเนอเรเตอร์ใน CLB สามารถคอนฟิกเป็น RAM ได้ 2 ขนาดคือ

- 16x1 RAM 2 ตัว: มีอินพุทข้อมูล 2 อินพุทและเอาท์พุทข้อมูล 2 เอาท์พุท โดย RAM แต่ละตัวจะใช้แอดเดรสเดียวกันหรือถ้าต้องการจะใช้แยกกันก็ได้
- 32x1 RAM 1 ตัว: มีข้อมูลอินพุท 1 อินพุทและข้อมูลเอาท์พุท 1 เอาท์พุท

และ RAM ใน CLB มีไทม์มิ่งโหมด 2 โหมดคือ

- Edge-Triggered (Synchronous): มีการเขียนข้อมูลที่ขอบขาขึ้นหรือลงของสัญญาณนาฬิกาของ CLB
- Level-Sensitive (Asynchronous): มีสัญญาณ WE (Write Enable) จากภายนอกเป็นสัญญาณเพื่อการเขียนข้อมูล

ไทม์มิ่งโหมดนี้จะใช้ในฟังก์ชันเจเนอเรเตอร์ทั้ง 2 ชุดใน CLB เมื่อฟังก์ชันเจเนอเรเตอร์ทั้ง 2 ชุดนี้มีการคอนฟิกเป็น RAM และจำนวนพอร์ตที่ใช้อ่านสามารถจะโปรแกรมพอร์ตได้ 2 แบบคือ

- Single Port: ฟังก์ชันเจเนอเรเตอร์แต่ละชุดจะมีพอร์ตอ่าน 1 พอร์ต และพอร์ตเขียน 1 พอร์ต
- Dual Port: ฟังก์ชันเจเนอเรเตอร์ทั้ง 2 ชุดจะมีการคอนฟิกรวมกันเป็น RAM แบบ single 16x1 dual-port RAM โดยมีพอร์ตเขียน 1 พอร์ต และพอร์ตอ่าน 2 พอร์ต มีโอเปอเรชันอ่านและเขียนพร้อมกัน ซึ่งจะใช้แอดเดรสเดียวกันหรือแยกกันก็ได้

การเลือกโหมดการคอนฟิกของ RAM

การเลือกโหมดของ RAM ที่เพมาะสมกับฟังก์ชันที่ออกแบบนั้น จะดูจากไทม์มิ่งและทรัพยากรที่ฟังก์ชันต้องการ รวมทั้งเลือกให้มีโพรเซสการออกแบบที่ง่ายด้วย ดังแสดงในตารางที่ 2-2

RAM แบบ level-sensitive, edge-triggered และ dual-port นั้นจะต่างกันที่โอเปอเรชันการเขียนเท่านั้น ส่วนโอเปอเรชันการอ่านและไทม์มิ่งจะเหมือนกันทุกโหมด

การสร้างอาเรย์ของ RAM

RAM ใน XC4000 มีขนาด 16x1 และ 32x1 ดังนั้นถ้าต้องการใช้ RAM น้อยกว่า 16 words จะเห็นว่า RAM ขนาด 16x1 นั้นจะถูกใช้โดยมีแอดเดรสที่ไม่ได้ใช้ต่อลงกราวนด์หรือ Vcc

ผู้ใช้สามารถสร้าง RAM เป็นอาเรย์ขนาด 16xn หรือ 32xn ได้เองอย่างง่าย โดยนำเอา RAM เหล่านี้มาเชื่อมต่อขนานกัน ตัวอย่างเช่นการนำ RAM ขนาด 32x1 จำนวน 2 ตัวมาต่อกันเพื่อให้ได้ RAM ขนาด 64x1 จำนวน 1 ตัวโดย most significant address bit จะทำหน้าที่เลือก RAM ตัวใดตัวหนึ่งจาก RAM ทั้งสองตัว ในขณะที่แอดเดรสบิทที่เหลือนั้นทำหน้าที่ตามปกติ และในระหว่างไซเคิลของการอ่าน การเลือกระหว่าง RAM 2 ตัวนี้จะเกี่ยวข้องกับการมัลติเพลกซ์ข้อมูลที่จะเป็นเอาท์พุทด้วย ส่วนในไซเคิลของการเขียนนั้น ข้อมูลใน RAM ทั้งสองตัวจะเหมือนกัน และสัญญาณ WE จะถูกต่อเข้าด้วยกันโดยผ่านเกตเพื่อทำให้แน่ใจได้ว่าจะมีข้อมูลไหลเข้าใน RAM ตัวใดตัวหนึ่งเท่านั้น

	Level-Sensitive	Edge-Triggered	Dual-Port Edge-Triggered
Use for New Designs?	No	Yes	Yes
Size (16x1, Registered)	½ CLB	½ CLB	1 CLB
Simultaneous Read/Write	No	No	Yes
Relative Performance	X	2X	2X (4X effectively)

ตารางที่ 2-2 การเลือกโหมดสำหรับ RAM

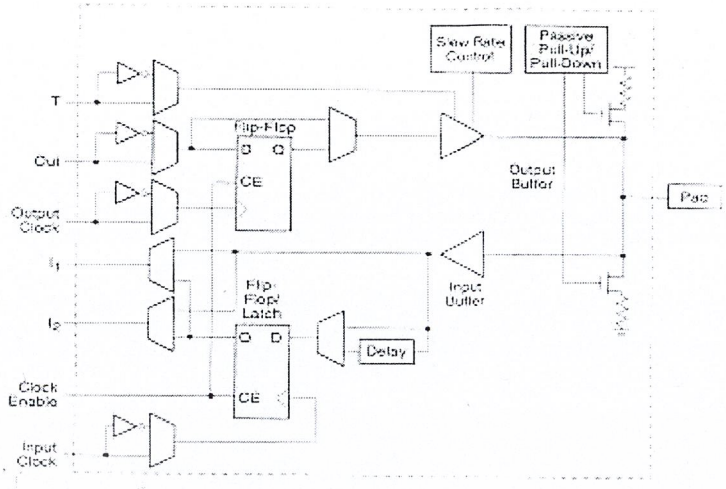
❖ **Input/Output Blocks (IOBs)**

IOB ใช้เป็นส่วนอินเทอร์เฟสระหว่างแพคเก็จพินภายนอกกับลอจิกภายใน โดยที่ IOB แต่ละตัวใช้ควบคุมแพคเก็จพิน 1 ชุด และสามารถจะคอนฟิกสัญญาณให้เป็นอินพุท, เอาท์พุท หรือสัญญาณแบบสองทิศทางก็ได้ ดังรูปที่ 2-3 แสดงบล็อกไดอะแกรมอย่างง่ายของ IOB

IOB นี้ยังสามารถโปรแกรมให้ต่อกับ high impedance pull-up register ได้ ซึ่งจะช่วยป้องกันการลอย (Floating) ในกรณีที่ User I/O ขาใดไม่ได้ต่อใช้งาน นอกจากนี้ค่าดีเลย์ของฟลิปฟลอปของ IOB จะมีค่าประมาณ 3 นาโนวินาที ซึ่งค่าดีเลย์สั้นๆ นี้ทำให้การใช้งานภายในสัญญาณนาฬิกาแบบอะซิงโครนัสมีประสิทธิภาพดี และเป็นการลดค่าความไม่แน่นอน (metastable) ให้น้อยที่สุด

บิทการคอนฟิกโปรแกรม (Configuration program bit) สามารถกำหนดการทำงานของ IOB ให้เป็นไปได้ในหลายลักษณะ เช่น Optimal output register, Signal inversion และ 3-State, slew rate control ของเอาท์พุท

ถึงแม้ว่าภายใน FPGA จะมีวงจรป้องกัน electrostatic discharge แล้วก็ตาม แต่การจับต้อง FPGA ก็ควรทำด้วยความระมัดระวัง



รูปที่ 2-3 บล็อกไดอะแกรมของ IOB

สัญญาณอินพุทของ IOB

แพธ I1 และ I2 ในรูปที่ 2-3 จะเป็นตัวนำสัญญาณอินพุทเข้าอาเรย์ และสัญญาณอินพุทนี้จะเชื่อมต่อเป็นอินพุทของรีจิสเตอร์ ซึ่งสามารถโปรแกรมให้เป็นฟลิปฟลอปแบบ edge-triggered หรือ level-sensitive latch ก็ได้

สัญญาณเอาท์พุทของ IOB

สัญญาณเอาท์พุทสามารถเปลี่ยนแปลงได้ภายใน IOB และสามารถส่งผ่านสัญญาณให้แพด (pad) โดยตรงหรือเก็บไว้ในฟลิปฟลอป (edge-triggered flip-flop) ก็ได้ ดังรูปที่ 2-3

❖ **Configuration**

การคอนฟิกเป็นกระบวนการโหลดข้อมูลที่ได้โปรแกรมไว้ลงใน FPGA เพื่อกำหนดฟังก์ชันการทำงานของบล็อกภายในและการเชื่อมต่อระหว่างกัน การคอนฟิกนี้ก็เหมือนกับการโหลดคำสั่งลงรีจิสเตอร์ของชีพที่ทำการโปรแกรมนั่นเอง

อุปกรณ์ XC4000 ใช้ข้อมูลการคอนฟิกหลายร้อยบิทต่อ CLB 1 ตัว ซึ่งบิทการคอนฟิกแต่ละบิทจะกำหนดสถานะ (state) ของเซลล์หน่วยความจำ 1 เซลล์ ที่ใช้ควบคุมฟังก์ชันลุกอัพเทเบิลบิท (Function look-up table bit) หรือมัลติเพล็กเซอร์อินพุท (Multiplexer input) หรือทรานซิสเตอร์ที่ใช้เชื่อมต่อ (Interconnect pass transistor)

Configuration Modes

XC4000 FPGA สามารถทำงานได้หลายแบบ โดยเลือกโหมดการทำงานได้จากบิทอินพุท 3 บิท คือ M2, M1 และ M0 ซึ่งมีโหมดการทำงานให้เลือก 6 โหมด ได้แก่ มาสเตอร์โหมด 3 โหมด, เพอริเฟอรัล โหมด 2 โหมด และสเลฟโหมด 1 โหมด การเลือกโหมดการทำงานของ FPGA แสดงดังตารางที่ 2-3

Mode	M2	M1	M0	CCLK	Data
Master Serial	0	0	0	Output	Bit-serial
Slave Serial	1	1	1	Input	Bit-serial
Master Parallel Up	1	0	0	Output	Byte-Wide, increment from 00000
Master Parallel Down	1	1	0	Output	Byte-Wide, decrement from 3FFFF
Peripheral Synchronous*	0	1	1	Input	Byte-Wide
Peripheral Asynchronous	1	0	1	Output	Byte-Wide
Reserved	0	1	0	-	-
Reserved	0	0	1	-	-

Note: *Peripheral Synchronous can be considered byte-wide Slave Parallel

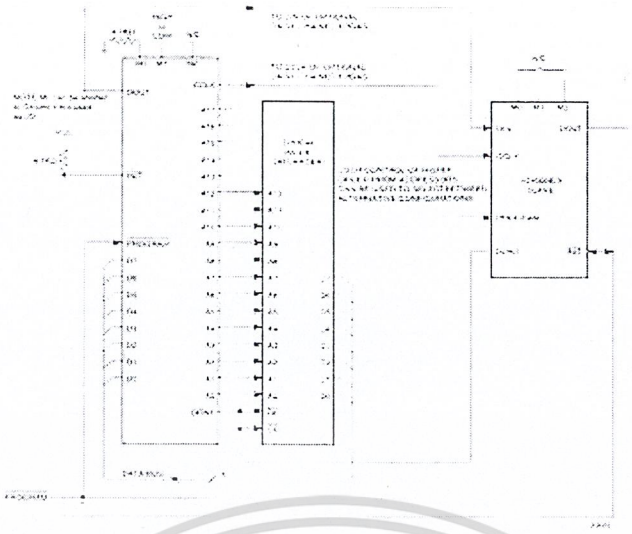
ตารางที่ 2-3 โหมดการคอนฟิก

จากตารางที่ 2-3 แต่ละโหมดมีลักษณะสำคัญ ดังนี้

มาสเตอร์โหมด

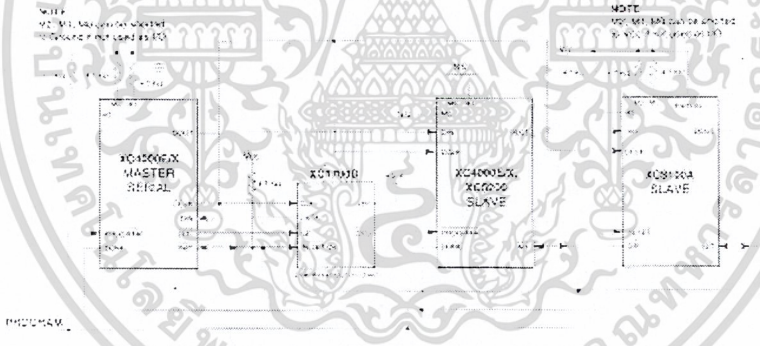
มาสเตอร์โหมดทั้ง 3 แบบ จะสร้างสัญญาณนาฬิกาเอง (CCLK) เพื่อขับอุปกรณ์ที่เป็นสเลฟ โดยสร้างจากออสซิลเลเตอร์ภายใน นอกจากนี้ยังสร้างแอดเดรสและไทม์มิ่งสำหรับ PROM ภายนอกที่เป็นตัวเก็บข้อมูลการคอนฟิก

มาสเตอร์โหมดแบบขนานทั้งแบบนับขึ้นและนับลง (Master Parallel Up or Down) ดังรูปที่ 2-4 จะสร้างสัญญาณ CCLK และ PROM แอดเดรส และรับข้อมูลเป็นไบต์แบบขนาน ข้อมูลจะเป็นแบบอนุกรมภายในรูปแบบเฟรมข้อมูลของ FPGA เอง ถ้าเป็นวงจรนับขึ้นจะเริ่มนับจากแอดเดรส 00000 แต่ถ้าเป็นวงจรนับลงจะเริ่มนับจากแอดเดรส 3FFFF



รูปที่ 2-4 วงจรไดอะแกรมของ FPGA ในมาสเตอร์โหมดแบบขนาน

มาสเตอร์โหมดแบบอนุกรมจะสร้าง CCLK และรับข้อมูลการคอนฟิกเป็นแบบอนุกรมจาก PROM ที่มีการคอนฟิกแบบอนุกรมเช่นกัน ดังรูปที่ 2-5



รูปที่ 2-5 วงจรไดอะแกรมของ FPGA ในมาสเตอร์/สเลฟโหมดแบบอนุกรม

ความเร็วของ CCLK สามารถเลือกได้ว่าจะเป็น 1 เมกะเฮิร์ต (เซตเป็นมาตรฐาน) หรือจะเป็น 8 เมกะเฮิร์ต และการคอนฟิกจะเริ่มที่ความถี่ต่ำเสมอ ซึ่งสามารถจะเปลี่ยนความถี่ให้สูงขึ้นได้โดยเปลี่ยนในเฟรมแรก

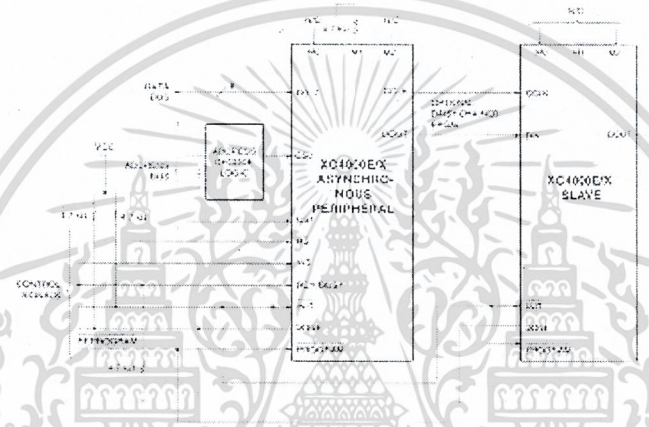
สเลฟโหมด

จากรูปที่ 2-4 และรูปที่ 2-5 สเลฟโหมดมีอยู่แบบเดียวคือ สเลฟโหมดแบบอนุกรม ซึ่ง FPGA จะรับข้อมูลการคอนฟิกแบบอนุกรมเมื่อขอบขาขึ้นของสัญญาณ CCLK และหลังจากโหลดการคอนฟิกและส่งข้อมูลออกแล้วก็จะทำการซิงโครไนซ์อีกครั้งที่ขอบขาลงของสัญญาณ CCLK ถัดไป

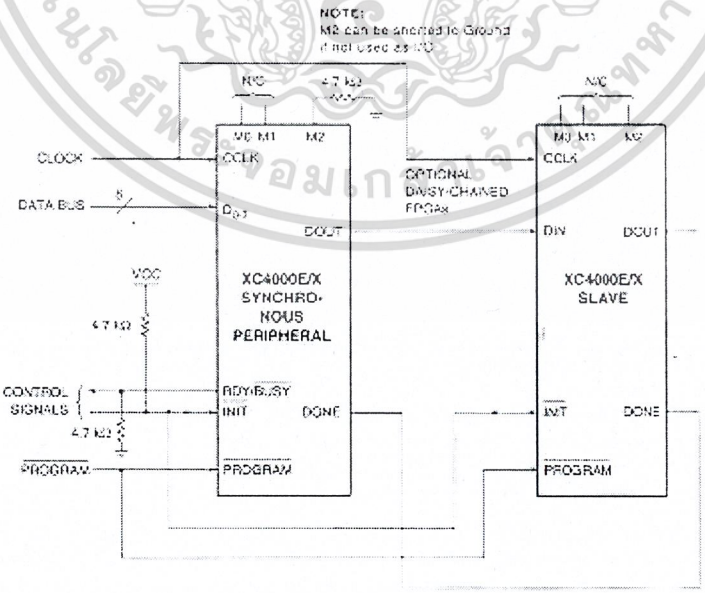
ถ้าต้องการต่ออุปกรณ์แบบมัลติเพิลสเลฟให้มีการคอนฟิกเหมือนกันนั้น สามารถทำได้โดยเชื่อมต่อกันด้วยขาอินพุท DIN แบบขนาน ซึ่งในกรณีนี้อุปกรณ์หลายตัวสามารถจะคอนฟิกไปพร้อมกันได้

<u>เพอริเฟอรัลโหมด</u>

เพอริเฟอรัลโหมดทั้ง 2 แบบจะรับข้อมูลเป็นไบต์จากบัส (Bus) สำหรับเพอริเฟอรัลโหมดแบบอะซิงโครนัสนั้น (ดังรูปที่ 2-6) ออสซิลเลเตอร์ภายในจะสร้างสัญญาณ CCLK ที่ซิงโครไนซ์กับข้อมูลแบบไบต์ และสัญญาณ CCLK ยังสามารถขับอุปกรณ์ที่เป็นสเลฟได้ด้วย ส่วนเพอริเฟอรัลโหมดแบบซิงโครนัสนั้น (ดังรูปที่ 2-7) จะรับสัญญาณนาฬิกาจากภายนอกมาเป็นอินพุทให้กับสัญญาณ CCLK ที่ซิงโครไนซ์กับข้อมูล



รูปที่ 2-6 วงจรไดอะแกรมของ FPGA ในเพอริเฟอรัลโหมดแบบอะซิงโครนัส



รูปที่ 2-7 วงจรไดอะแกรมของ FPGA ในเพอริเฟอรัลโหมดแบบซิงโครนัส

รูปแบบของข้อมูล (Data Stream Format)

รูปแบบของกระแสข้อมูล (Bitstream) จะเหมือนกันในโหมดการคอนฟิกทุกโหมด โดยข้อมูลแบบอนุกรมจะอ่านเป็นบิทจากซ้ายไปขวา และข้อมูลแบบขนานจะอ่านเป็นไบต์ที่มีบิทแรกในแต่ละไบต์เป็นบิท D0

กระแสข้อมูลการคอนฟิกจะเริ่มด้วยสตริง 1s ขนาด 8 บิท ตามด้วยโค้ด 0010b แล้วตามด้วยจำนวนสตริงที่มีความยาว 24 บิทและบิทเติมเต็ม (Separator field) ที่เป็น 1s ขนาด 4 บิท เฮดเดอร์ทั้งหมดนี้จะนำหน้าข้อมูลการคอนฟิกจริงในรูปเฟรมข้อมูล ซึ่งความยาวของแต่ละเฟรมและจำนวนเฟรมจะขึ้นอยู่กับรูปแบบของอุปกรณ์

เฟรมข้อมูลแต่ละเฟรมจะเริ่มด้วยบิทเริ่มต้น (Start bit) ซึ่งเป็น 0 และสิ้นสุดด้วยบิทที่ใช้ตรวจสอบข้อผิดพลาดขนาด 4 บิท

<u>หมายเหตุ</u>: โค้ด 0010b ในเฮดเดอร์ของกระแสข้อมูลการคอนฟิก ใช้เป็นตัวระบุว่าสตริงขนาด 24 บิทที่จะตามมานั้นเป็นตัวแสดงถึงการนับความยาวคือจำนวนของสัญญาณนาฬิกาในการคอนฟิกที่ต้องใช้ในการโหลดข้อมูลการคอนฟิกทั้งหมด

Cyclic Redundancy Check (CRC)

CRC เป็นวิธีการตรวจสอบข้อผิดพลาดในการส่งข้อมูลที่ใช้ในการรับ-ส่งข้อมูลของ FPGA โดยการส่งข้อมูลจะเพิ่มบิทสำหรับตรวจสอบต่อท้ายชุดข้อมูล และการรับข้อมูลจะตรวจสอบชุดข้อมูลแล้วเปรียบเทียบผลกับผลของเช็คซัม (Checksum) ที่ได้ ถ้าตรงกันจึงถือว่าการส่งถูกต้อง

ข้อมูลแต่ละเฟรมจะมีบิทสำหรับตรวจสอบต่อท้ายชุดข้อมูลจำนวน 4 บิท ถ้าเฟรมข้อมูลเกิดความผิดพลาดและถูกตรวจสอบได้ระหว่างการโหลดของ FPGA แล้วโพรเซสการคอนฟิกจะถูกยับยั้งและ FPGA จะดึงขาสัญญาณ INIT\ ลงและเข้าสู่เวทสเตท (Wait state)

2.2 ทฤษฎีพื้นฐานของ UART

2.2.1 ความรู้เบื้องต้นเกี่ยวกับ UART

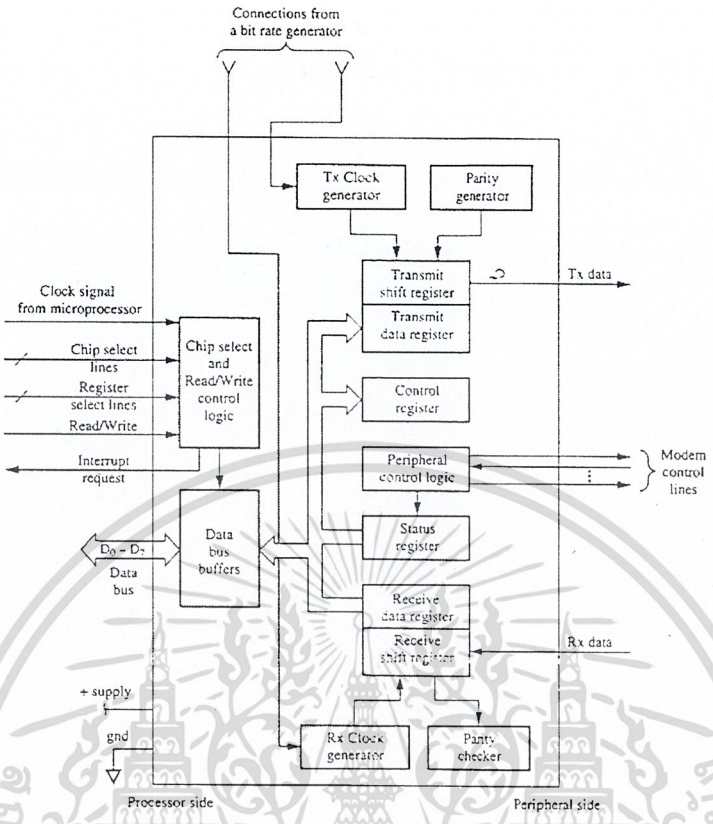
UART เป็นชื่อย่อที่มาจาก Universal Asynchronous Receiver/Transmitter ใช้อธิบายอุปกรณ์ I/O ที่ใช้รับ-ส่งข้อมูลอนุกรมแบบอะซิงโครนัสในระบบคอมพิวเตอร์และสามารถทำการโปรแกรมได้ ซึ่งมีบริษัทผู้ผลิตที่ผลิตผลิตภัณฑ์เกี่ยวกับ UART หลายราย ตัวอย่างเช่น อุปกรณ์ I/O 8 บิทแบบอนุกรมของบริษัทอินเทล (Intel Corporation) ตัวหนึ่งมีชื่อว่า Programmable Communication Interface (PCI) ซึ่ง PCI นี้มีความสามารถในการรับ-ส่งข้อมูลได้แบบซิงโครนัสและอะซิงโครนัส จึงเป็นที่รู้จักกันในนาม Universal Synchronous/Asynchronous Receiver/Transmitter (USART) แต่เนื่องจากโปรโตคอลในการติดต่อสื่อสารแบบซิงโครนัสและอะซิงโครนัสมีความแตกต่างกัน ดังนั้นอุปกรณ์ PCI จึงมักจะถูกนำมาใช้ในการติดต่อสื่อสารแบบอะซิงโครนัสเพียงอย่างเดียวและใช้อุปกรณ์ I/O อย่างอื่นในการติดต่อสื่อสารแบบซิงโครนัสแทน อีกตัวอย่างหนึ่ง เช่น UART 8 บิทของบริษัทโมโตโรล่า (Motorola) รู้จักกันในชื่อ Asynchronous Communication Interface Adapter (ACIA) ซึ่ง ACIA นี้ใช้ในการติอต่อสื่อสารแบบอะซิงโครนัสเท่านั้น

UART จะมีลักษณะเดียวกับอุปกรณ์เพอริเฟอรัลอื่นๆ คือ สามารถโปรแกรมการทำงานได้ และสามารถกำหนดแอดเดรสของรีจิสเตอร์ภายในได้ โดยปกติแล้ว UART ได้รับการออกแบบมาสำหรับทำงานบนชิพไมโครโปรเซสเซอร์ขนาด 8 บิท แต่อุปกรณ์ทางอิเล็คทรอนิคจะใช้ไมโครโปรเซสเซอร์ขนาด 16 บิทและ 32 บิทเป็นส่วนใหญ่ ดังนั้น ปัจจุบันจึงมีการปรับปรุง UART ให้สามารถใช้งานได้แพร่หลายขึ้น

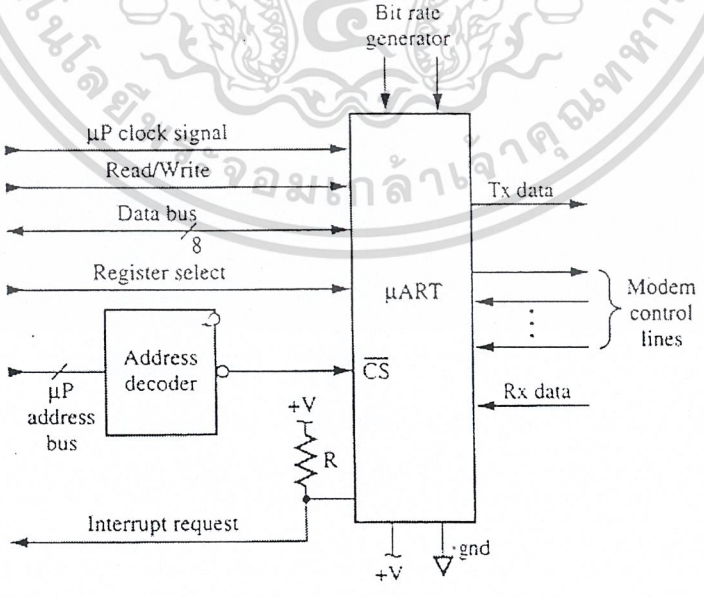
2.2.2 การเชื่อมต่อ UART กับอุปกรณ์ภายนอก

UART เป็นอุปกรณ์ที่นำมาใช้เชื่อมต่อไมโครโปรเซสเซอร์ (หรือ CPU) และเพอริเฟลรัล ซึ่งพินด้านโปรเซสเซอร์ (Processor side) จะเชื่อมต่อกับไมโครโปรเซสเซอร์โดยตรงหรือผ่านชิพอินเทอร์เฟสก่อนเชื่อมต่อกับไมโครโปรเซสเซอร์ และพินด้านเพอริเฟอรัล (Peripheral side) จะเชื่อมต่อกับอุปกรณ์ภายนอก แสดงดังรูปที่ 2-8

ไมโครโปรเซสเซอร์ทำการเลือกอุปกรณ์ UART ได้โดยใช้ **chip select pin** (หรือ chip enable pin) ของ UART ซึ่งก็คือการอีนาเบิล (enable) บัฟเฟอร์บัสข้อมูลนั่นเอง พินของ UART เหล่านี้เชื่อมต่อกับสายสัญญาณแอดเดรสของไมโครโปรเซสเซอร์ผ่านชิพดีโค้ดเดอร์ ดังรูปที่ 2-9 และไมโครโปรเซสเซอร์ทำการเลือกรีจิสเตอร์ภายใน UART ที่ต้องการได้โดยใช้ **register select pin** ซึ่งเชื่อมต่อกับสายสัญญาณแอดเดรสหมายเลขต่ำสุดของไมโครโปรเซสเซอร์ เช่น A0 หรือ A1



รูปที่ 2-8 การเชื่อมต่อบล็อกภายในของ UART กับอุปกรณ์ภายนอก



รูปที่ 2-9 การเชื่อมต่อสายสัญญาณกับ UART

ข้อมูลจะส่งผ่านระหว่างไมโครโปรเซสเซอร์และ UART ทางบัสข้อมูลแบบ 2 ทิศทางขนาด 8 บิทโดยข้อมูลเหล่านี้ส่งผ่านแบบซิงโครนัสตามสัญญาณนาฬิกาที่ขับไมโครโปรเซสเซอร์ ซึ่งการอ่าน-เขียนข้อมูลนั้น ไมโครโปรเซสเซอร์จะอ่าน-เขียนข้อมูลกับ UART ผ่านบัสข้อมูลโดยใช้สายสัญญาณอ่าน-เขียนข้อมูล (**read-write line**) นอกจากนี้ UART ยังมีสายสัญญาณร้องขอการอินเทอร์รัป (**IRQ**) ที่เป็นตัวให้สัญญาณกับไมโครโปรเซสเซอร์ว่าถึงเวลาที่ต้องรับ-ส่งข้อมูลชุดต่อไป หรือให้สัญญาณกับไมโครโปรเซสเซอร์ว่าไม่ควรรับ-ส่งข้อมูลเพราะมีปัญหาที่อุปกรณ์ สำหรับการเชื่อมต่อด้านเพอริเฟอรัลนั้น UART ใช้สายสัญญาณส่งข้อมูล (**Tx Data**), สายสัญญาณรับข้อมูล (**Rx Data**) และสายสัญญาณควบคุมอุปกรณ์เพอริเฟอรัล ดังรูปที่ 1-8

UART รับข้อมูลจากไมโครโปรเซสเซอร์แบบขนานผ่านทางบัสข้อมูลและส่งข้อมูลออกแบบอนุกรมผ่านทางสายสัญญาณข้อมุล Tx และรับข้อมูลเข้าแบบอนุกรมผ่านทางสายสัญญาณข้อมูล Rx และส่งข้อมูลแบบขนานให้กับไมโครโปรเซสเซอร์ผ่านทางบัสข้อมูล ดังนั้นฟังก์ชันพื้นฐานของ UART ก็คือ Parallel-to-serial converter และ Serial-to-parallel converter นั่นเอง สำหรับข้อมูลแบบอนุกรมที่จะส่งออกทาง Tx นั้น UART จะใส่บิทเริ่มต้น, บิทพาริตี้ และบิทสิ้นสุดลงในชุดข้อมูลก่อนที่จะส่งโดยอัตโนมัติ และจะตัดบิทเหล่านี้ออกจากชุดข้อมูลที่รับเข้าทาง Rx ก่อนที่จะส่งให้ไมโครโปรเซสเซอร์อ่านข้อมูล

บริษัทผู้ผลิตส่วนใหญ่มักจะสร้าง UART ให้สามารถเชื่อมต่อกับโมเดม โดยเพิ่มสายสัญญาณควบคุมอุปกรณ์เพอริเฟอรัล (**Peripheral control**) หรือควบคุมการทำแฮนเช็คกิ้งนั่นเอง ทั้งนี้เพื่อให้การส่งผ่านข้อมูลระหว่าง UART และโมเดมเป็นไปอย่างมีลำดับ

สำหรับอัตราการรับ-ส่งข้อมูลนั้น จะกำหนดด้วยความถี่ของสัญญาณนาฬิกาภายนอกซึ่งอาจมาจากคริสตอลภายนอกที่เชื่อมต่อกับอุปกรณ์โดยตรง หรือมาจากตัวกำเนิดอัตราบิท (Bit rate generator) หรือตัวกำเนิดอัตราบอด (Buad rate generator) ก็ได้

การทำงานของ UART ก็เหมือนกับอุปกรณ์อินพุท เอาท์พุทอื่นๆ ที่จะใช้เวลานาน ดังนั้นขณะที่ไมโครโปรเซสเซอร์ติดต่อสื่อสารกับ UART อยู่ก็จะไม่สามารถทำงานอย่างอื่นได้

2.2.3 บล็อกไดอะแกรมภายในของ UART

จากรูปที่ 1-8 ที่แสดงบล็อกภายในของ UART มีรีสเตอร์ที่เกี่ยวกับการส่งข้อมูลอนุกรมคือรีจิสเตอร์เลื่อนข้อมูลส่งออก (Transmit shift register) และรีจิสเตอร์ส่งข้อมูล (Transmit data register) ส่วนรีสเตอร์ที่เกี่ยวกับการรับข้อมูลอนุกรมคือรีจิสเตอร์เลื่อนข้อมูลรับเข้า (Receive shift register) และรีจิสเตอร์รับข้อมูล (Receive data register) โดยไมโครโปรเซสเซอร์จะส่งข้อมูลที่ต้องการส่งออกไปยังรีจิสเตอร์ส่งข้อมูลทีละ 1 บิทและรับข้อมูลจากรีจิสเตอร์รับข้อมูลทีละ 1 บิทเช่นกัน ซึ่งจะไม่ติดต่อสื่อสารกับรีจิสเตอร์เลื่อนข้อมูลส่งออกหรือรีจิสเตอร์เลื่อนข้อมูลรับเข้าโดยตรง แต่ UART จะส่งข้อมูลที่ต้องการส่งออกบิทถัดไปจากรีจิสเตอร์ส่งข้อมูลไปยังรีจิสเตอร์เลื่อนข้อมูลส่งออกโดยอัตโนมัติเมื่อรีจิสเตอร์เลื่อนข้อมูลส่งออกนั้นว่าง และข้อมูล (1 บิท) ที่รับเข้ามาจากรีจิสเตอร์เลื่อนข้อมูลรับเข้าจะไปเก็บอยู่ในรีจิสเตอร์รับข้อมูลโดยอัตโนมัติและจะยังคงอยู่ในรีจิสเตอร์รับข้อมูลจนกว่าไมโครโปรเซสเซอร์จะเรียกอ่านข้อมูล ซึ่ง

การทำงานลักษณะนี้เพื่อเป็นการบัฟเฟอร์ข้อมูลไว้ ไม่ต้องรับ-ส่งข้อมูลทันทีที่มีข้อมูลมาถึง การที่ใช้รีจิสเตอร์ 2 ตัวทำงานด้วยกันไม่ว่าจะเป็นการรับหรือการส่งข้อมูลเช่นนี้เรียกว่า "double buffering"

ไมโครโปรเซสเซอร์สามารถควบคุมการทำงานของ UART ได้โดยการส่งชุดข้อมูลควบคุม (Control word) ซึ่งมีลักษณะเป็นไบนารี่ไปเก็บในรีจิสเตอร์ควบคุม (Control register) ภายใน UART ก่อนที่จะเริ่มการติดต่อสื่อสารกับ UART นั้น ตัวอย่างชุดข้อมูลควบคุมเช่น การเซตอัตราการรับ-ส่งข้อมูล, จำนวนบิทของข้อมูลในการรับ-ส่งแต่ละตัวอักษร, รูปแบบของพาริตี้ (กรณีที่มีการตรวจสอบพาริตี้), จำนวนของบิทสิ้นสุด และกำหนดว่าจะให้ UART ส่งสัญญาณอินเทอร์รัปไปยังไมโครโปรเซสเซอร์หรือไม่เมื่อรีจิสเตอร์ส่งข้อมูลว่างหรือรีจิสเตอร์รับข้อมูลเต็มอยู่

สำหรับรีจิสเตอร์สถานะ (Status register) มีขนาด 8 บิท ใช้เป็นตัวเก็บสถานะภายในต่างๆ ของ UART แต่ละบิทจะบอกสถานะในรูปลอจิก ตัวอย่างเช่น บิทที่ใช้บอกสภาพของสายสัญญาณควบคุมเพอริเฟอรัล, บอกสถานะของรีจิสเตอร์ส่งข้อมูลหรือรีจิสเตอร์รับข้อมูลว่าว่างหรือเต็มอยู่ และสภาพของข้อมูลที่รับเข้ามา เช่น มีความผิดพลาดจากการตรวจสอบพาริตี้ (parity error), ความผิดพลาดจากการโอเวอร์รัน (overrun error) หรือเฟรมข้อมูลผิดพลาด(framing error) เกิดขึ้นหรือไม่ โดยความผิดพลาดจากการตรวจสอบพาริตี้เกิดขึ้นเมื่อพาริตี้ของข้อมูลที่รับเข้ามาไม่ตรงกับที่เซตไว้ในรีจิสเตอร์ควบคุมของ UART และความผิดพลาดจากการโอเวอร์รันเกิดขึ้นเมื่อ UART พยายามเก็บข้อมูลใหม่ลงในรีจิสเตอร์รับข้อมูลโดยที่ไมโครโปรเซสเซอร์ยังไม่ได้อ่านข้อมูลเก่าที่อยู่ในรีจิสเตอร์รับข้อมูลออกไป ส่วนเฟรมข้อมูลผิดพลาดเกิดขึ้นเมื่อ UART ได้รับจำนวนของบิทสิ้นสุดไม่ถูกต้องตามที่กำหนดไว้

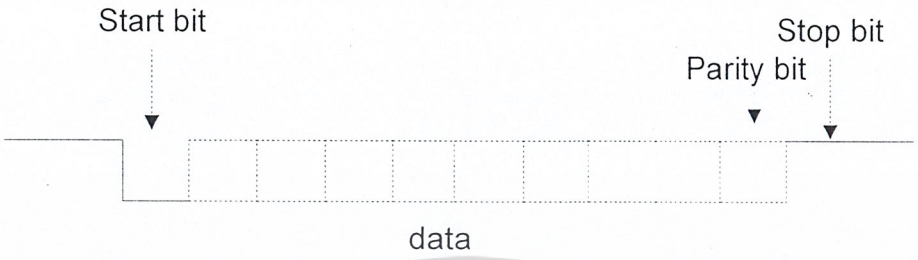
โดยทั่วไป ไมโครโปรเซสเซอร์จะติดต่อสื่อสารกับ UART ผ่านรีจิสเตอร์ 4 ตัว ต่อไปนี้

- **รีจิสเตอร์ส่งข้อมูล (Transmit data register):** ไมโครโปรเซสเซอร์ติดต่อสื่อสารโดยการเขียนข้อมูลลงรีจิสเตอร์ (Write-only)
- **รีจิสเตอร์ควบคุม (Control register):** ไมโครโปรเซสเซอร์ติดต่อสื่อสารโดยการเขียนข้อมูลลงรีจิสเตอร์ (Write-only)
- **รีจิสเตอร์รับข้อมูล (Receive data register):** ไมโครโปรเซสเซอร์ติดต่อสื่อสารโดยการอ่านข้อมูลจากรีจิสเตอร์ (Read-only)
- **รีจิสเตอร์สถานะ (Status register):** ไมโครโปรเซสเซอร์ติดต่อสื่อสารโดยการอ่านข้อมูลมาจากรีจิสเตอร์ (Read-only)

ปกติแล้ว ไมโครโปรเซสเซอร์จะไม่ทำการติดต่อสื่อสารกับรีจิสเตอร์ของ UART โดยตรง ดังนั้น UART จึงมีลอจิกควบคุม (control logic) และบัฟเฟอร์ของบัสข้อมูล (data bus buffer) เพื่อใช้อินเทอร์เฟสแทน ดังรูปที่ 1-8

2.2.4 รูปแบบของชุดข้อมูลภายใน UART

รูปแบบของข้อมูลจะประกอบด้วยบิทเริ่มต้น (start bit) ,บิทข้อมูล 8 บิท ,บิทพาริตี้ (parity bit) และ บิทสิ้นสุด (stop bit) ทำการรับส่งข้อมูลแบบฟูลดูเพล็กซ์ (full duplex) ดังรูปที่ 2-10

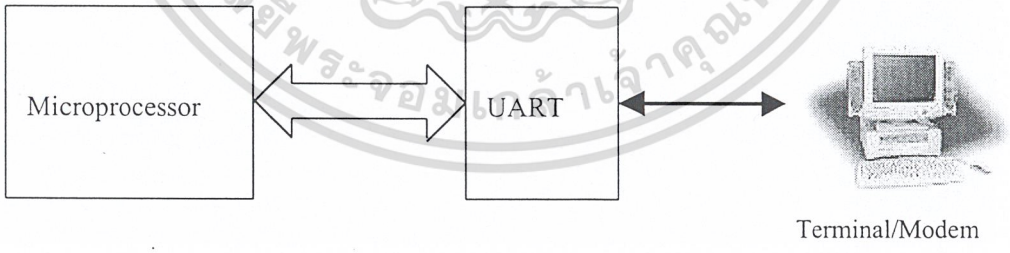


รูปที่ 2-10 รูปแบบของชุดข้อมูลแบบอะซิงโครนัสของ UART

2.2.5 MC6850 ACIA

MC6850 เป็นอุปกรณ์ UART ที่คณะผู้จัดทำเลือกมาเป็นต้นแบบสำหรับการออกแบบ UART ของคณะผู้จัดทำ MC6850 นี้เป็นอุปกรณ์ Asynchronous Communication Interface Adapter (ACIA) ซึ่งเป็น IC ของบริษัทโมโตโรล่า มีลักษณะการทำงานดังรูปที่ 2-11 คือ

1. รับข้อมูลอินพุทจากบัสข้อมูลแบบขนานของไมโครโปรเซสเซอร์ทีละ 1 ไบต์ และส่งอนุกรมไปยังอุปกรณ์เทอมินัลหรือโมเดมแบบอะซิงโครนัสทีละ 1 บิท
2. รับข้อมูลอนุกรมจากอุปกรณ์เทอมินัลหรือโมเดมแบบอะซิงโครนัส และส่งขนานไปยังบัสข้อมูลของไมโครโปรเซสเซอร์



รูปที่ 2-11 การเชื่อมต่อการทำงานของ UART

MC6850 ACIA มีพินทั้งหมด 24 พิน ดังรูปที่ 2-12 มีลักษณะต่อไปนี้

ก) พิน 1 และ 12 เป็นพินสำหรับจ่ายกระแสไฟ Vcc และ Vss

ข) พิน 2 และ 6 เป็นพินสำหรับรับ-ส่งข้อมูลอนุกรม

ค) พิน 3 และ 4 เป็นพินสำหรับสัญญาณนาฬิกาในการรับ-ส่งข้อมูล

ง) พิน 5,23 และ 24 เป็นพินสำหรับสายสัญญาณ V24/RS232 เป็นสัญญาณการร้องขอเพื่อจะส่งข้อมูล (Request to send), การตรวจสอบความผิดพลาดของข้อมูล (Data carrier detect) และการเคลียร์สัญญาณเพื่อจะส่งข้อมูล (Clear to send) ตามลำดับ

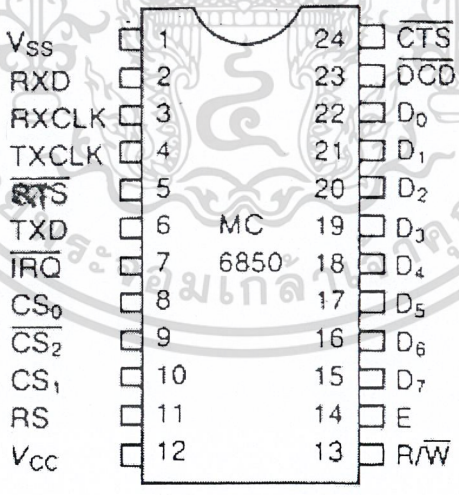
จ) พิน 7 เป็นพินสำหรับสัญญาณการร้องขออินเทอร์รัปที่ใช้ในการอินเทอร์รัปไมโครโปรเซสเซอร์ และเป็นสัญญาณบอกไมโครโปรเซสเซอร์เมื่อชิพมีการรับ-ส่งข้อมูล โดยสัญญาณนี้จะมีสถานะต่ำ (low) ขณะที่มีการอินเทอร์รัปเกิดขึ้น

ฉ) พิน 8, 9 และ 10 เป็นพินสำหรับสัญญาณการเลือกชิพ (chip select) ซึ่งพินเหล่านี้จะเชื่อมต่อกับบัสแอดเดรสของไมโครโปรเซสเซอร์ โดยชิพจะถูกเลือกเมื่อพิน CS0 และ CS1 มีสถานะสูงและพิน CS2 มีสถานะต่ำ

ช) พิน 11 และ 13 เป็นพินสำหรับเลือกรีจิสเตอร์และการอ่าน/เขียน ตามลำดับ โดยพินทั้งสองนี้จะใช้งานร่วมกันดังแสดงในตารางที่ 2-4 เพื่อเลือกรีจิสเตอร์ที่ไมโครโปรเซสเซอร์จะติดต่อสื่อสารด้วย ดังรูปที่ 2-13

ซ) พิน 14 เป็นพินสำหรับแสดงสถานะอีนาเบิล (enable) ซึ่งต้องมีสถานะสูงตลอดเมื่อต้องการใช้งานสัญญาณการเลือกชิพและการอ่าน/เขียนข้อมูล

ฌ) พิน 15 และ 22 เป็นพินสำหรับสายสัญญาณข้อมูล 8 บิทแบบสองทิศทาง มีตั้งแต่ D0-D7 โดยพินเหล่านี้จะเชื่อมต่อกับพินข้อมูลของไมโครโปรเซสเซอร์



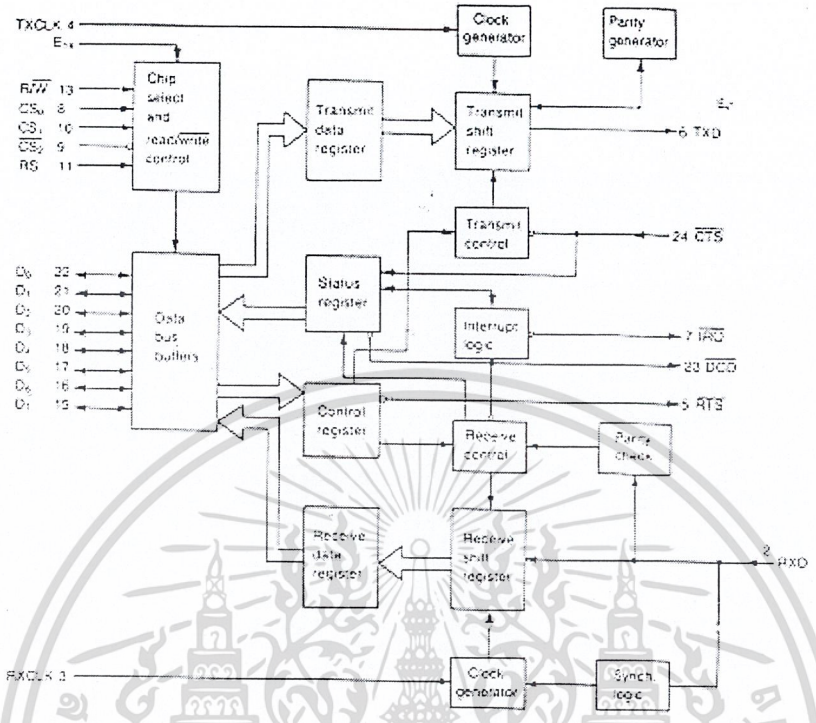
รูปที่ 2-12 พินของชิพ MC6850 ACIA

R/W	RS	Action
1	0	Read from status register
1	1	Read from data register
0	0	Write to control register
0	1	Write to transmit data register

ตารางที่ 2-4 ความหมายของการจับคู่พิน R/W และ RS แบบต่างๆ

บล็อกไดอะแกรมอย่างง่ายของ MC6850 ACIA แสดงดังรูปที่ 2-13 ไมโครโปรเซสเซอร์จะเขียนข้อมูลแต่ละไบต์ที่ต้องการส่งออกลงรีจิสเตอร์ส่งข้อมูล เมื่อรีจิสเตอร์เลื่อนข้อมูลส่งออกว่าง ข้อมูลไบต์ที่รออยู่ในรีจิสเตอร์ส่งข้อมูลก็จะถูกส่งแบบขนานจากรีจิสเตอร์ส่งข้อมูลไปยังรีจิสเตอร์เลื่อนข้อมูลส่งออกเพื่อใส่บิทเริ่มต้น, บิทสิ้นสุดและบิทพาริตี้ไปกับชุดข้อมูลนั้น แล้วจึงเลื่อนชุดข้อมูลนี้ออกไปทางพิน TxD แบบอนุกรม ซึ่งอัตราการส่งข้อมูลออกจะกำหนดโดยสัญญาณนาฬิกาภายนอก และจะส่งข้อมูลเมื่อไมโครโปรเซสเซอร์ตรวจพบว่ารีจิสเตอร์ส่งข้อมูลว่าง หรือไมโครโปรเซสเซอร์ได้รับสัญญาณร้องขอการอินเทอร์รัปให้ส่งข้อมูลไบต์ถัดไปได้ หรือโดยการตรวจสอบบิทในรีจิสเตอร์สถานะ

สำหรับข้อมูลอนุกรมนั้นจะรับมาจากสายสัญญาณผ่านรีจิสเตอร์เลื่อนข้อมูลรับเข้าทีละบิทจนครบจึงตัดบิทเริ่มต้น, บิทสิ้นสุดและบิทพาริตี้ออก จากนั้นเวิร์ดข้อมูลนี้จะถูกส่งแบบขนานไปยังรีจิสเตอร์รับข้อมูลเพื่อรอให้ไมโครโปรเซสเซอร์อ่านข้อมูลไปแบบขนานเช่นกัน ซึ่งก่อนที่ข้อมูลแต่เวิร์ดจะเลื่อนไปยังรีจิสเตอร์รับข้อมูลนั้น ACIA จะต้องตรวจสอบความผิดพลาดที่อาจเกิดขึ้นก่อน โดยมีความผิดพลาดจากการตรวจสอบพาริตี้, ความผิดพลาดจากการโอเวอร์รันและเฟรมข้อมูลผิดพลาด ตรวจสอบจากบิทต่างๆ ของรีจิสเตอร์สถานะ และไมโครโปรเซสเซอร์จะรู้ว่ารีจิสเตอร์รับข้อมูลเต็มพร้อมให้อ่านข้อมูลแล้วโดยได้รับสัญญาณร้องขอการอินเทอร์รัป หรือการตรวจสอบบิท 0 ในรีจิสเตอร์สถานะ ถ้ารีจิสเตอร์รับข้อมูลเต็มและไม่มีความผิดพลาดเกิดขึ้นแล้วไมโครโปรเซสเซอร์จะอ่านข้อมูลไปเก็บไว้ในหน่วยความจำ



รูปที่ 2-13 บล็อกไดอะแกรมของ MC6850 ACIA

2.2.5.1 รีจิสเตอร์ควบคุม (Control Register)

จากตารางที่ 1-4 รีจิสเตอร์ควบคุมทั้ง 8 บิทจะถูกเลือกจากไมโครโปรเซสเซอร์เมื่อ RS และ R/W มีค่าเท่ากับ 0 แล้วไมโครโปรเซสเซอร์จะส่งสัญญาณควบคุมผ่านทางพินข้อมูล D0-D7 ดังตารางที่ 2-5 ซึ่งมีรายละเอียดดังนี้

- การเลือกจำนวนหารที่เพื่อหาความถี่ของสัญญาณนาฬิกาที่จะใช้ในการรับข้อมูลของ MC6850 ACIA จะใช้บิท CR0 และ CR1 ในการเลือกว่าจะหารความถี่ด้วยจำนวน 1, 16, 64 หรือใช้มาสเตอร์รีเซตของ ACIA ดังตารางที่ 2-6 การหารด้วย 1 จะใช้เมื่อสัญญาณนาฬิกาในการรับข้อมูลซิงโครไนซ์กับสัญญาณข้อมูลที่รับเข้ามาเท่านั้น นอกนั้นจะใช้เมื่อเป็นการทำงานแบบอะซิงโครนัส
- บิท CR2, CR3 และ CR4 จะใช้เมื่อ ACIA เริ่มต้นการทำงานเพื่อกำหนดจำนวนบิทข้อมูลต่อเวิร์ด, จำนวนบิทสิ้นสุดที่จะใส่ต่อท้ายข้อมูลแต่ละชุด และกำหนดพาริตี้ว่าจะให้เป็นพาริตี้คู่หรือพาริตี้คี่ รายละเอียดดังแสดงในตารางที่ 2-7
- บิท CR5 และ CR6 เป็นสัญญาณควบคุมอินเทอร์รัปการส่ง (transmit interrupt control) และควบคุมสัญญาณพร้อมที่จะส่ง (ready to send) แสดงในตารางที่ 2-8 เมื่อบิทสัญญาณควบคุม

อินเทอร์รัปการส่งอีนาเบิลแล้ว ACIA จะไปอินเทอร์รัปไมโครโปรเซสเซอร์ (ในกรณีที่สัญญาณ IRQ\ เชื่อมต่อกับไมโครโปรเซสเซอร์) ทันทีเมื่อบัฟเฟอร์การส่งข้อมูลว่าง

- บิท CR7 เป็นบิทอีนาเบิลอินเทอร์รัปการรับข้อมูล ถ้าบิทนี้มีค่าเป็น 1 แล้ว ACIA จะอินเทอร์รัปไมโครโปรเซสเซอร์เมื่อรีจิสเตอร์รับข้อมูลเต็มหรือมีสัญญาณ DCD (Data Carrier Detect) เกิดขึ้น

ก่อนการส่งข้อมูลทุกครั้งควรใช้มาสเตอร์รีเซต เพื่อรีเซตสัญญาณที่กำหนดไว้ เช่น ความถี่ของสัญญาณนาฬิกา, ความยาวของเวิร์ดข้อมูล ฯลฯ และโปรแกรมใหม่ในการส่งข้อมูลแต่ละครั้ง

Data pin	**D0**	**D1**	**D2**	**D3**
Control register	CR0 counter divide select 1	CR1 Counter divide select 2	CR2 Word select 1	CR3 Word select 2
Data pin	**D4**	**D5**	**D6**	**D7**
Control register	CR4 Word select 3	CR5 Transmit control 1	CR6 Transmit control 2	CR7 Receive interrupt

ตารางที่ 2-5 บิทสัญญาณของรีจิสเตอร์ควบคุม

CR1	**CR0**	**Counter divide**
0	0	÷ 1
0	1	÷ 16
1	0	÷ 64
1	1	Master reset

ตารางที่ 2-6 จำนวนหารของรีจิสเตอร์ควบคุม

CR4	CR3	CR2	Character	Parity
0	0	0	7 bits: 2 stop bits	Even
0	0	1	7 bits: 2 stop bits	Odd
0	1	0	7 bits: 1 stop bit	Even
0	1	1	7 bits: 1 stop bit	Odd
1	0	0	8 bits: 2 stop bits	
1	0	1	8 bits: 1 stop bit	
1	1	0	8 bits: 1 stop bit	Even
1	1	1	8 bits: 1 stop bit	Odd

ตารางที่ 2-7 รูปแบบข้อมูลของรีจิสเตอร์ควบคุม

CR6	CR5	
0	0	RTS low: transmit interrupt disabled
0	1	RTS low: transmit interrupt enabled
1	0	RTS high: transmit interrupt disabled
1	1	RTS high: transmits constant 0 on TxD; transmit interrupt disabled

หมายเหตุ: สัญญาณ RTS คือ Ready to send

ตารางที่ 2-8 การควบคุมการส่งข้อมูลของรีจิสเตอร์ควบคุม

2.2.5.2 รีจิสเตอร์สถานะ

จากตารางที่ 1-4 รีจิสเตอร์สถานะจะถูกเลือกจากไมโครโปรเซสเซอร์เมื่อ RS มีค่าเท่ากับ 0 และ R/W มีค่าเท่ากับ 1 ซึ่งมีรายละเอียดของแต่ละบิทแสดงในตารางที่ 2-9 ดังนี้

บิท D0 จะมีสถานะสูงเมื่อ ACIA ได้รับข้อมูลและรีจิสเตอร์รับข้อมูลเต็ม ซึ่งไมโครโปรเซสเซอร์จะอ่านบิทนี้เพื่อตรวจสอบว่า ACIA รับข้อมูลเข้ามาและพร้อมที่จะส่งให้ไมโครโปรเซสเซอร์หรือยัง

บิท D1 จะมีสถานะสูงเพื่อแสดงให้ไมโครโปรเซสเซอร์รู้ กรณีที่รีจิสเตอร์ส่งข้อมูลว่างและไมโครโปรเซสเซอร์สามารถส่งข้อมูลไบต์ถัดไปให้กับ ACIA ได้แล้ว

บิท D2 และ D3 เป็นบิทสำหรับสายสัญญาณ V24/RS232 เมื่อ DCD\ มีสถานะสูง บิท D2 จะเซต ซึ่งจะทำให้บิท D1 มีสถานะต่ำ และเมื่อ CTS\ มีสถานะสูง บิท D3 จะเซตซึ่งจะทำให้บิท D1 มีสถานะต่ำเช่นกัน และทั้งสองกรณีนี้จะทำให้บิท D7 มีสถานะต่ำ

บิท D4 จะมีสถานะสูงเพื่อแสดงให้ไมโครโปรเซสเซอร์รู้ กรณีที่ข้อมูลที่เข้ามาไม่ซิงโครไนซ์และ/หรือให้หยุดการส่งข้อมูลมาให้ ACIA ซึ่งแฟลกนี้สามารถรีเซตได้โดยการให้ DCD\ (บิท D2) มีสถานะสูง

บิท D5 จะมีสถานะสูงเพื่อแสดงให้ไมโครโปรเซสเซอร์รู้ กรณีที่มีการรับข้อมูลมาจากสาย แต่ข้อมูลนั้นสูญหายก่อนที่รีจิสเตอร์รับข้อมูลจะอ่านเข้าไป ซึ่งแฟลกนี้สามารถรีเซตได้โดยการให้ DCD\ มีสถานะสูง

บิท D6 จะมีสถานะสูงเมื่อมีความผิดพลาดจากพาริตี้ ซึ่งแฟลกนี้สามารถรีเซตได้โดยการให้ DCD\ มีสถานะสูง

และบิท 7 เป็นบิทสำหรับการร้องขอการอินเทอร์รัปซึ่งจะมีสถานะสูงเมื่อต้องการอินเทอร์รัปไมโครโปรเซสเซอร์ โดยสถานะสูงอาจมาจากการที่ขณะนี้

1) รีจิสเตอร์ส่งข้อมูลว่าง
2) รีจิสเตอร์รับข้อมูลเต็ม และ
3) พิน DCD\ มีสถานะสูง ในกรณีที่ข้อมูลที่รับเข้ามาสูญหาย

Data pin	**D0**	**D1**	**D2**	**D3**
Status	Receive data register full	Transmit data register empty	Data carrier detect (DCD)	Clear to send (CTS)
Data pin	**D4**	**D5**	**D6**	**D7**
Status	Framing error	Receiver over-run	Parity error	Interrupt request

ตารางที่ 2-9 บิทสัญญาณของรีจิสเตอร์สถานะ

2.3 ทฤษฎีพื้นฐานของ VHDL

2.3.1 VHDL

The VHSIC Hardware Description Language เป็นภาษามาตรฐาน ที่ใช้อธิบายฮาร์ดแวร์ในระดับต่างๆ VHSIC ย่อมาจาก Very High Speed Integrated Circuit ที่ถือกำเนิดขึ้นในช่วงปลาย ค.ศ. 1970 ถึงต้น ค.ศ. 1980 VHSIC Program นี้เป็นต้นแบบทำให้เกิดภาษาใหม่ขึ้น เรียกกันว่าภาษา VHDL

2.3.2 VHDL ภาษามาตรฐาน

ภาษาสำหรับอธิบายฮาร์ดแวร์ที่มาจาก VHSIC Program นี้เป็นภาษาใหม่ที่เกิดขึ้นในปี 1981 มีชื่อว่า VHSIC Hardware Description Language หรือที่รู้จักกันในชื่อ VHDL ซึ่งภาษา VHDL เกิดขึ้นจากวัตถุประสงค์หลัก 2 ข้อ คือ ข้อแรก: นักออกแบบต้องการภาษาที่สามารถอธิบายวงจรที่ซับซ้อนที่นักออกแบบพยายามจะอธิบายได้ ข้อสอง: นักออกแบบต้องการภาษาที่เป็นมาตรฐานในปี 1986 VHDL ได้รับการจัดอยู่ใน IEEE Standard และมีการปรับปรุงเปลี่ยนแปลงจนกลายเป็น IEEE 1076 Standard ในเดือนธันวาคม ปี 1987

ในการทำโปรเจ็คนี้ คณะผู้จัดทำได้ใช้ภาษา VHDL เป็นเครื่องมือสำหรับช่วยสร้างโมเดล (model) ของระบบดิจิตอลที่ซับซ้อน โดยอาศัยขบวนการของการออกแบบแบบทอปดาวน์ (Top-Down Design) ขบวนการดังกล่าวคือการบรรยายระบบดิจิตอลในระดับบนสุด (top-level) ในรูปของแนวความคิดฟังก์ชันการทำงานอย่างสังเขป (abstract) แล้วเขียนโมเดลและจำลองการทำงานเพื่อตรวจสอบความถูกต้อง ซึ่งหลังจากที่ผ่านการตรวจสอบแล้ว แนวความคิดอย่างสังเขปนี้ จะถูกแบ่ง (partition) ให้เป็นส่วนย่อยๆ ตามกลุ่มของฟังก์ชันการทำงาน และเช่นเดียวกันส่วนย่อยๆ ที่สร้างขึ้นเหล่านั้น จะถูกจำลองการทำงาน (simulation) ตรวจสอบความถูกต้อง (test and verification) เป็นเช่นนี้ไปเรื่อยๆ จนกว่าโมเดลที่ออกแบบนั้นทำงานจะถูกต้องสมบูรณ์ และระดับสุดท้ายคือระดับเกตเป็นระดับล่างสุด (gate-level) ซึ่งสามารถที่จะนำไปเปรียบเทียบกับอุปกรณ์ฮาร์แวร์ทางดิจิตอลต่างๆ อาทิเช่น ไมโครโปรเซสเซอร์, RAM, ROM, PLD และ FPGA ได้โดยผ่านขั้นตอนของการสังเคราะห์วงจร (circuit synthesis)

2.3.3 โครงสร้างของภาษา VHDL

การออกแบบระบบดิจิตอลด้วยภาษา VHDL จะต้องประกอบด้วยส่วนต่างๆ ดังต่อไปนี้

1. ส่วนการออกแบบเอนติตี้ (Entity Design Unit): เป็นส่วนที่กำหนดการเชื่อมต่อภายนอกของคอมโพเนนต์แต่ละคอมโพเนนต์ (component) ที่ออกแบบ เช่น กำหนดพอร์ตอินพุท/เอาท์พุท, กำหนดลักษณะของสัญญาณที่เข้าออกคอมโพเนนต์แต่ละคอมโพเนนต์ เป็นต้น โดยโปรแกรมที่เขียนทั้งหมดจะบรรจุอยู่ในส่วนของเอนติตี้นี้ และเอนติตี้นี้ยังเป็นส่วนที่จะผ่านพารามิเตอร์ต่างๆ เข้าสู่โครงสร้างภายในเช่น ส่วนสถาปัตยกรรม

2. ส่วนการออกแบบสถาปัตยกรรม (Architecture Design Unit): เป็นส่วนที่อธิบายการทำงานของคอมโพเนนต์ที่ออกแบบในเอนติตี้ โดยการอธิบายการทำงานนี้สามารถอธิบายได้หลายลักษณะ ดัง

นั้นเอนติตี้หนึ่งตัวจึงสามารถประกอบด้วยส่วนสถาปัตยกรรมได้หลายตัวเช่นกัน และสามารถออกแบบสถาปัตยกรรมนี้ได้ใน 3 ระดับคือ

- ระดับโครงสร้าง (Structure): การออกแบบที่มีโมเดลแบบโครงสร้าง จะออกแบบโดยดูจากโครงสร้างของระบบในรูปของอุปกรณ์ต่างๆ ที่สามารถมองเห็นรูปร่างของวงจรได้ชัดเจน เป็นการเชื่อมต่อระดับเกตที่นำมาต่อกัน จัดอยู่ในประเภทการแสดงด้วยการแทนที่ด้วยอุปกรณ์ (ซึ่งในที่นี้หมายถึงอุปกรณ์ที่อยู่ในรูปแบบของ VHDL คือ ส่วนการออกแบบเอนติตี้และสถาปัตยกรรม) และการเชื่อมต่อภายในระหว่างอุปกรณ์เหล่านั้น (instantiation and interconnection) ด้วยโครงสร้างของคอมโพเนนต์ของ VHDL โดยการบรรยายออกมาในรูปที่เรียกว่า VHDL netlist
- ระดับพฤติกรรม (Behavior): การออกแบบที่มีโมเดลแบบพฤติกรรม จะออกแบบโดยดูจากพฤติกรรมการเปลี่ยนแปลงค่าสัญญาณของวงจร ซึ่งเป็นรูปแบบโมเดลอย่างสังเขป (abstract model) รูปแบบลักษณะนี้จะไม่ชี้ให้เห็นชัดว่าวงจรจะมีรูปร่างและโครงสร้างเป็นอย่างไร โดยถูกจัดให้อยู่ในประเภทของการบรรยายที่ไม่ต้องมีการอ้างถึงรูปแบบโมเดลย่อย (submodel) ภายในส่วนการออกแบบสถาปัตยกรรมนั้นอีก

  * การออกแบบระดับพฤติกรรมนี้บางครั้งเรียกว่าการบรรยายในรูปอัลกอริธึม (Algorithm Description)
- ระดับรีจิสเตอร์ (RTL): RTL ย่อมาจาก Register Transfer Level Description ระดับนี้มีลักษณะพิเศษคือการระบุถึงรีจิสเตอร์ทุกตัวและคอมไบเนชันลอจิก (combination logic) ระหว่างรีจิสเตอร์เหล่านั้นในการออกแบบ

  วิธีการอธิบายถึงรีจิสเตอร์ในการออกแบบมี 2 แบบคือ แบบชัดแจ้ง (explicit) โดยใช้ component instantiation ซึ่งจะเหมือนกับการออกแบบระดับโครงสร้าง และแบบโดยนัย (implicit) โดยใช้การ inference ซึ่งจะเหมือนกับการออกแบบระดับพฤติกรรม ส่วนการอธิบายคอมไบเนชันลอจิกนั้น อาจจะใช้สมการทางลอจิก (logical equation) , สเตทเมนต์ควบคุมการทำงานแบบเป็นลำดับ (sequantial control statement) เช่นการใช้ CASE ,IF then ELSE , subprogram เป็นต้น หรืออาจจะใช้สเตทเมนต์ควบคุมการทำงานแบบขนาน (concurrent statement)

  การออกแบบระดับ RTL มักจะถูกใช้กับงานออกแบบที่เป็นแบบซิงโครนัสและมีการอธิบายพฤติกรรมแบบ clock-by-clock

  <u>หมายเหตุ</u>: นอกจากนี้ภาษา VHDL สามารถใช้เขียนบรรยายรูปแบบของระบบดิจิตอลในลักษณะผสม (mix modeling) ระหว่างการออกแบบระดับพฤติกรรมและระดับโครงสร้างที่อยู่ภายในส่วนสถาปัตยกรรมเดียวกันเพื่อแสดงรายละเอียดของรูปแบบที่เขียนในระดับต่างๆ ซึ่งขั้นตอนที่กล่าวมานี้จะครอบคลุมวิธีการของโพรเซสการออกแบบแบบทอปดาวน์ทั้งหมด

3. ส่วนการออกแบบแพคเก็จ (Package Design Unit): ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อย (subprogram) ที่เป็นประโยชน์ต่อการเขียนโมเดลบรรยายระบบดิจิตอล สามารถเก็บไว้ในส่วนที่เรียกว่าแพคเก็จได้ และข้อมูลเหล่านี้สามารถถูกเรียกไปใช้ได้จากส่วนการออกแบบเอนติตี้, ส่วนการออกแบบสถาปัตยกรรม หรือจากส่วนการออกแบบแพคเก็จอื่นๆ ด้วยชุดคำสั่ง USE statement ดังนั้นแพคเก็จจึงเปรียบเสมือนกล่องเครื่องมือสำหรับบรรจุเครื่องมือที่ใช้ในการสร้างโปรแกรมรูปแบบต่างๆ

4. ส่วนการออกแบบการคอนฟิก (Configuration Design Unit): คอมโพเนนต์แต่ละคอมโพเนนต์ที่ออกแบบสามารถมีสถาปัตยกรรมได้หลายลักษณะ (เอนติตี้ 1 เอนติตี้จะมีสถาปัตยกรรมได้หลายแบบอยู่ในเอนติตี้นั้น) ซึ่งการคอนฟิกนี้จะใช้เป็นตัวกำหนดว่าเอนติตี้ใดจะใช้สถาปัตยกรรมแบบใด โดยจะอธิบายพฤติกรรมที่ใช้สำหรับเอนติตี้แต่ละเอนติตี้

2.3.4 ความสัมพันธ์ระหว่างส่วนต่างๆ ของการออกแบบ

การศึกษาเกี่ยวกับความสัมพันธ์ระหว่างหน่วย ตลอดจนกลไกของการทำงานที่มีผลต่อกันเมื่อแต่ละส่วนถูกนำมาประกอบเข้าด้วยกัน ซึ่งจะเป็นการศึกษาในหลักการของไลบรารี่ (Library) ของภาษา VHDL ในบางครั้งอาจจะมองไลบรารี่ในภาษา VHDL เป็นเสมือนกับตู้ที่ใช้เก็บอุปกรณ์อิเล็กทรอนิกส์ต่างๆ (หรือก็คือส่วนสถาปัตยกรรมนั่นเอง) ที่ถูกจัดให้อยู่ตามลิ้นชักต่างๆ ของตู้ (หรือก็คือส่วนเอนติตี้นั่นเอง) ดังนั้นจึงสามารถจึงสามารถแบ่งส่วนการออกแบบใน VHDL ออกได้เป็น 2 ส่วนคือ หน่วยหลัก (primary) และหน่วยรอง (secondary) ที่มีความสัมพันธ์กัน

2.3.4.1 ไลบรารี่ (Library)

หลักการของไลบรารี่คือพื้นฐานสำคัญที่จะสร้างความเข้าใจในความสัมพันธ์ระหว่างส่วนการออกแบบต่างๆ ของ VHDL ซึ่งหลังจากที่ส่วนการออกแบบถูกวิเคราะห์ตรวจสอบความถูกต้องตามกฎเกณฑ์การเขียนและความถูกต้องของฟังก์ชันหรือพฤติกรรมการทำงานแล้ว (complied and simulated) ผลลัพธ์ที่ได้จากการวิเคราะห์จะถูกเก็บไว้ในส่วนที่เรียกว่าไลบรารี่

ส่วนที่เป็นการไลบรารี่ของ VHDL นั้น สามารถนำไปใช้ได้ในส่วนการออกแบบอื่นๆ โดยการอ้างถึงชื่อของไลบรารี่นั้นๆ ซึ่งชื่อของไลบรารี่นี้เรียกว่าชื่อสัญลักษณ์ (symbolic name) และการอ้างถึงดังกล่าวนี้ก็เพื่อที่จะเข้าสู่สิ่งที่อยู่ภายในไลบรารี่นั่นเอง โดยข้อมูลที่อยู่ภายในจะแบ่งเป็น

- หน่วยหลัก (primary units) ได้แก่
  - ส่วนการประกาศเอนติตี้ (entity declarations)
  - ส่วนการประกาศแพคเก็จ (package declarations)
  - ส่วนการกำหนดการคอนฟิก (configuration specifications)
- หน่วยรอง (secondary units) ได้แก่
  - ส่วนบอดี้ของสถาปัตยกรรม (architecture bodies)
  - ส่วนบอดี้ของแพคเก็จ (package bodies)

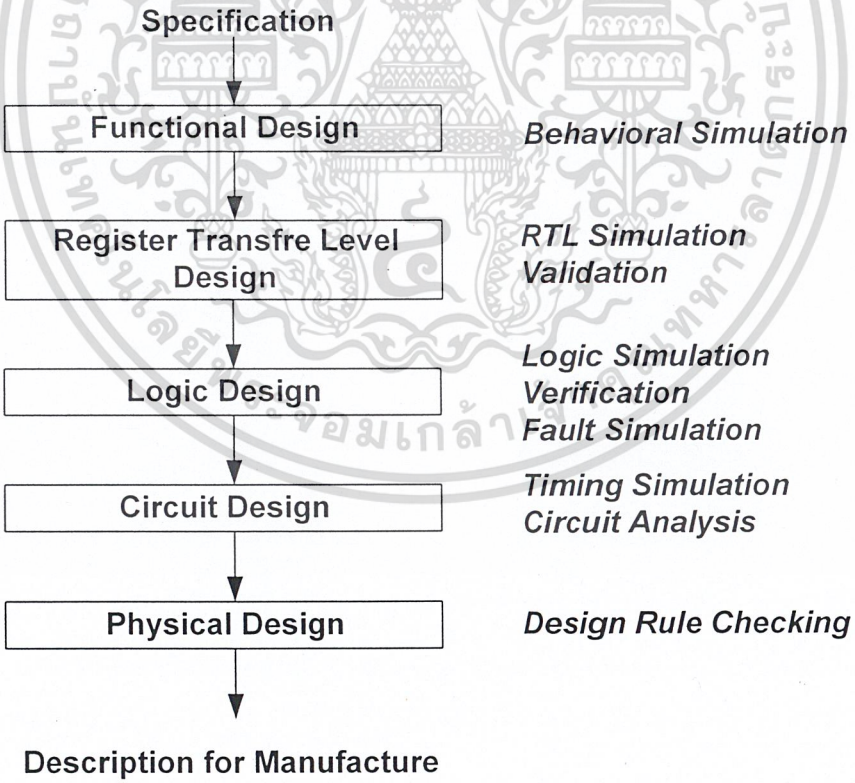
เนื่องจากหน่วยรองมีความสัมพันธ์กับหน่วยหลัก ฉะนั้นหน่วยรองจึงต้องได้รับการวิเคราะห์ภายหลังจากที่หน่วยหลักได้รับการวิเคราะห์แล้ว โดยส่วนการออกแบบเอนติตี้, สถาปัตยกรรม, แพคเก็จ และการคอนฟิกที่อยู่ภายในเหล่านี้จะต้องถูกกฎเกณฑ์การเขียน ซึ่งขบวนการที่บรรจุส่วนการออกแบบต่างๆ ลงในไลบรารี่จะเป็นไปอย่างอัตโนมัติเมื่อไฟล์ของ VHDL ที่ผู้ออกแบบเขียนขึ้นผ่านการวิเคราะห์และตรวจสอบว่าถูกต้องตามกฎเกณฑ์การเขียนแล้ว หรือเรียกได้อีกอย่างหนึ่งว่าผ่านการคอมไพล์ (compile) แล้วนั่นเอง โดยที่หน่วยรองจะต้องอยู่ในไลบรารี่เดียวกันกับหน่วยหลักที่เกี่ยวข้องกัน

บทที่ 3
การออกแบบระบบดิจิตอลเลียนแบบการทำงานของ UART

ในบทนี้ได้อธิบายถึงขั้นตอนการออกแบบวงจรดิจิตอลในโครงงานนี้ รายละเอียดการทำงานของคอมโพเนนต์ย่อยของระบบที่ได้ออกแบบ รวมทั้งแนวทางและขั้นตอนในการทดสอบวงจรที่ได้ทำการออกแบบขึ้นมาในโครงงานนี้ โดยจะอธิบายถึงการออกแบบระระบบดิจิตอลเลียนแบบการทำงานของ UART ตามหลักการดังกล่าว ในโครงงานนี้ใช้ ซอฟต์แวร์ Leonardo Spectrum ในการสังเคราะห์วงจร (VHDL Synthesis) และใช้ซอฟต์แวร์ Xilinx ในการ Place & Route

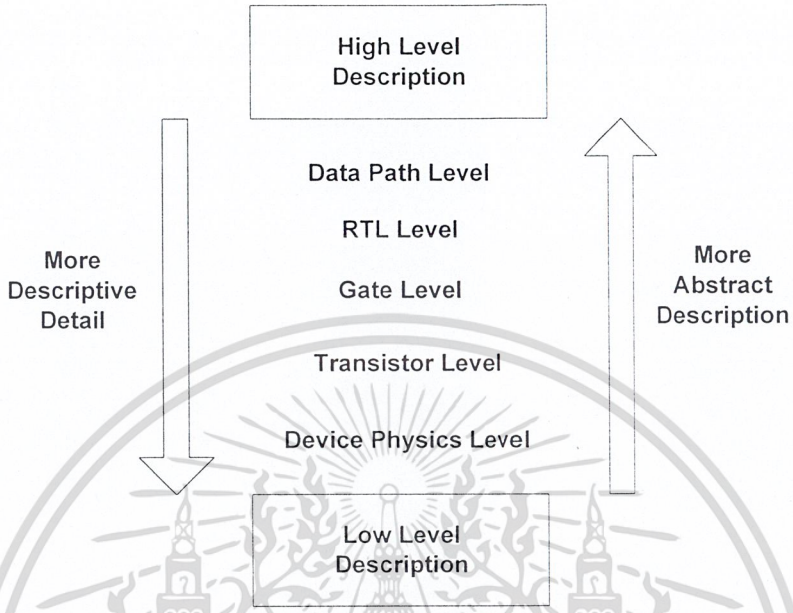
3.1 ขั้นตอนในการออกแบบวงจรดิจิตอล

การออกแบบระบบดิจิตอลเป็นกระบวนการที่เริ่มต้นจากการกำหนดลักษณะของระบบที่ต้องการ (Specification of requirements) การออกแบบในระดับหน้าที่การทำงาน (Functional Design) ไปจนถึงการสร้างระบบจริง (Physical Implementation) ดังรูปที่ 3-1



รูปที่ 3-1 ขั้นตอนการออกแบบระบบดิจิตอล

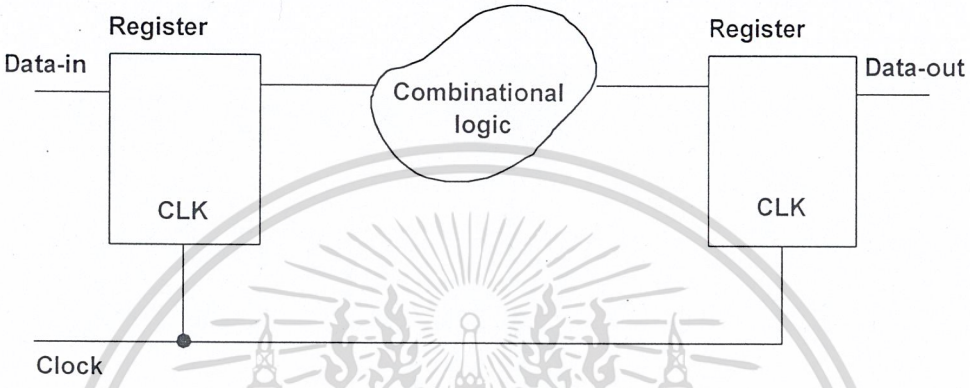
ในการออกแบบระบบดิจิตอลจะทำโดยการระบุและอธิบายถึงฮาร์ดแวร์ แบ่งระดับในการอธิบายเป็นดังรูปที่ 3-2



รูปที่ 3-2 ระดับการออกแบบอย่างสังเขป (Level of Abtraction)

- ระดับแพธข้อมูล (Data Path Level): เป็นการออกแบบระบบโดยดูจากพฤติกรรมการเปลี่ยนแปลงค่าของสัญญาณอินพุทและเอาท์พุทของระบบ การออกแบบในระดับนี้จะไม่สนใจถึงโครงสร้างการทำงานภายในของระบบ ผลของการออกแบบในระดับนี้จะอยู่ในรูปของ Block Diagram และโปรแกรมภาษา VHDL ในระดับ Behavior การออกแบบระบบแบบ behavior ก่อนเพื่อให้เข้าใจระบบที่จะออกแบบมากขึ้นเพราะสามารถจำลองการทำงานเพื่อดูรูปแบบสัญญาณ (waveform) ของสัญญาณอินพุทและเอาท์พุทของระบบว่าถูกต้องตาม specification และนำรูปแบบสัญญาณนั้นมาใช้เป็น specification ของการออกแบบในระดับต่อไป คือ RTL

- ระดับรีจิสเตอร์ (Register Transfer Level (RTL)): เป็นการออกแบบระบบโดยอธิบายถึงระบบในระดับที่สนใจการถ่ายโอนข้อมูลระหว่างรีจิสเตอร์ โดยการระบุถึงรีจิสเตอร์ทุกตัวในการออกแบบและ combinational logic ระหว่างรีจิสเตอร์เหล่านั้น ดังแสดงในไดอะแกรม "register and cloud" ในรูปที่ 3-3



รูปที่ 3-3 ลักษณะของ RTL

- ระดับเกต (Gate Level): เป็นระดับที่มองรายละเอียดไปถึงการเชื่อมต่อกันของเกตต่างๆ ที่มาสร้างเป็นระบบ การออกแบบในระดับนี้จะมี Software Tool ช่วยในการเปลี่ยนจากการออกแบบในลักษณะ RTL เป็นระบบในระดับ Gate Level Netlist
- ระดับทรานซิสเตอร์ (Transistor Level): เป็นการออกแบบในระดับที่ต้องคำนึงถึงมาตรฐานของไลบรารี่ (Standard library) ของโรงงานที่เราจะนำระบบไปทำการผลิตเป็นชิพ ต้องทำการวาง lay out ของคอมโพเนนต์ของระบบให้ได้ตามกฎเกณฑ์ของเทคโนโลยีที่ใช้
- ระดับอุปกรณ์ทางฟิสิกส์ (Device Physics Level): เป็นการออกแบบโดยนำคอมโนเนนต์ที่มีอยู่แล้วมาทำการเชื่อมต่อกันในระดับทางฟิสิกส์

3.2 ขั้นตอนการออกแบบระบบดิจิตอลเลียนแบบการทำงานของ UART

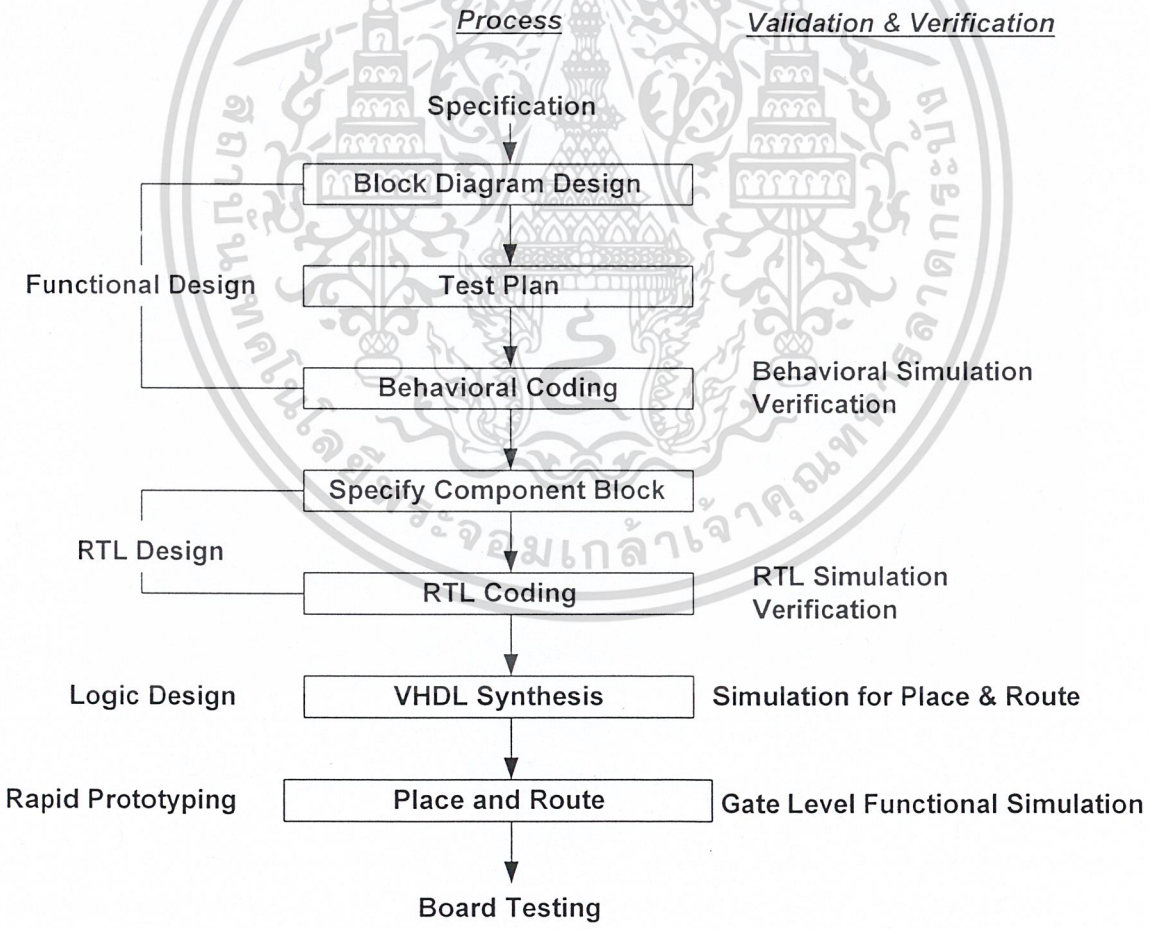
ขั้นตอนคล้ายกับการออกแบบระบบดิจดตอลโดยทั่วไปตามที่กล่าวมาแล้วข้างต้นแต่แตกต่างกันตรงที่ในโครงงานเป็นการออกแบบแล้วนำไปสร้างลงบน FPGA เพื่อทดสอบ ดังนั้นในขั้นตอนของ Circuit Design และ Physical Design จะเป็นการทำ Rapid Prototyping โดยใช้ FPGA แทน

ขั้นตอนการออกแบบระบบดิจิตอลของ UART มีรายละเอียดแสดงในรูปที่ 3-4 ดังต่อไปนี้

- Preliminary Design
  - Development Plan: วางแผนในการออกแบบและทดสอบระบบ
  - Design Specification: กำหนดหน้าที่ คุณสมบัติและลักษณะการทำงานของระบบ

- Block Diagram Design
  - แบ่งระบบเป็นคอมโพเนนต์ตามหน้าที่การทำงาน
  - กำหนดอินพุท/เอาท์พุทของระบบและการเชื่อมต่อกันของคอมโพเนนต์
- Test Plan
  - Test Strategy : วางแผนการทดสอบระบบทั้งในส่วนซอฟต์แวร์และฮาร์ดแวร์
  - Test Design: ออกแบบวิธีการตรวจสอบ คิดกรณีที่จะใช้ทดสอบทั้งคอมโพเนนต์และระบบทั้งหมด โดยคิดจาก specification และบล็อกไดอะแกรมของระบบ
  - Test System Design: ทำการออกแบบระบบที่ใช้ตรวจสอบ โดยการตรวจสอบในส่วนซอฟต์แวร์ใช้ Test Bench ในการจำลองการทำงาน และในส่วนฮาร์ดแวร์ในบอร์ด FPGA
- Behavior Coding
  - เขียนโปรแกรมสร้างแต่ละคอมโพเนนต์ในเชิงพฤติกรรม (behavior) เพื่อให้เข้าใจการทำงานของระบบมากขึ้นและจะนำไปเป็น specification ในการออกแบบในระดับต่อไป
  - นำแต่ละคอมโพเนนต์มารวมกันเป็นระบบ
  - สร้าง Test Bench เพื่อใช้ทดสอบคอมโพเนนต์และระบบ
- Simulation & Verification
  - จำลองการทำงานของคอมโพเนนต์และระบบ เพื่อให้เข้าใจการทำงานของคอมโพเนนต์และระบบ
  - ทำการแก้ไขโปรแกรมเพื่อให้ผลการจำลองการทำงานถูกต้องตาม specification
- Specify Component Block
  - กำหนดคุณลักษณะของแต่ละคอมโพเนนต์และระบบ รวมทั้งส่วนเชื่อมต่อทั้งหมดให้ชัดเจนตาม specification
- RTL Coding
  - เขียนโปรแกรมสร้างแต่ละคอมโพเนนต์ให้อยู่ในรูปแบบ RTL เพราะ โปรแกรมในลักษณะ behavior ซอฟต์แวร์ในการสังเคราะห์ที่โครงงานนี้ใช้ไม่สามารถนำไปสังเคราะห์ได้
- Simulation & Verification
  - ทำการจำลองการทำงานของแต่ละคอมโพเนนต์และระบบ โดยใช้ Test Bench เดียวกับในระดับ Behavior
  - ทำการแก้ไขโปรแกรมเพื่อให้ผลการจำลองการทำงานถูกต้องตาม specification ซึ่งก็คือรูปแบบของสัญญาณที่ได้จากการจำลองการทำงานในระดับ behavior
- VHDL Synthesis
  - ใช้ซอฟต์แวร์ Leonardo Spectrum ทำการสังเคราะห์โปรแกรมที่สร้างขึ้นให้อยู่ในรูปแบบที่สามารถแปลงเป็น gate level netlist ได้

- Simulation For Place & Route
  - นำไฟล์ภาษา VHDL และไฟล์ดีเลย์ที่ได้จากซอฟต์แวร์ที่ใช้ในการสังเคราะห์มาจำลองการทำงานโดยใช้ Test Bench เดิม โดยค่าดีเลย์ในระดับนี้จะเป็นค่าดีเลย์ระดับเกต (gate delay)
- Place & Route
  - ทำการสร้างระบบที่ได้ออกแบบลงบน FPGA โดยใช้ซอฟต์แวร์ทำโดยอัตโนมัติ ในโครงงานนี้ใช้ FPGA ของ Xilinx และใช้ซอฟต์แวร์ Xilinx Foundation Series ในการทำ Placement และ Routing
- Gate Level Functional Simulation
  - นำไฟล์ภาษา VHDL และไฟล์ดีเลย์ที่ได้จากซอฟต์แวร์ที่ใช้ในการสังเคราะห์มาจำลองการทำงานโดยใช้ Test Bench เดิม โดยค่าดีเลย์ในระดับนี้จะรวมค่าดีเลย์ทั้งหมดของ FPGA
- Board Testing
  - ทำการโปรแกรมระบบที่ได้ออกแบบไว้ลงบน FPGA เพื่อทำการทดสอบในระดับฮาร์ดแวร์



รูปที่ 3-4 ขั้นตอนการออกแบบระบบดิจิตอลเลียนแบบการทำงานของ UART

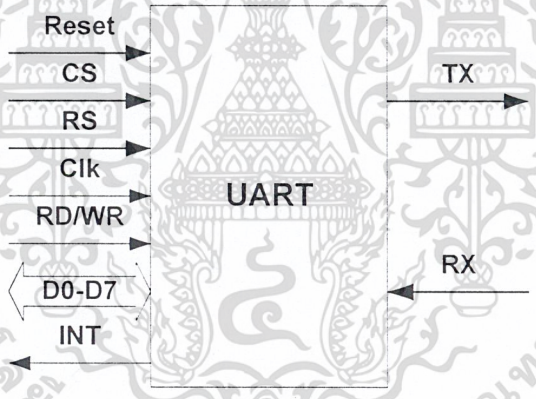
3.3 การสร้างระบบดิจิตอลเลียนแบบการทำงานของ UART โดยใช้ภาษา VHDL

3.3.1 Specification

- ในส่วนของการทำงานจะอ้างอิงชิพ MC6850 ACIA ของ Motorola และในส่วนการอินเทอร์เฟสจะอ้างอิงการอินเทอร์เฟสกับไมโครคอนโทรลเลอร์ MCS51
- Asynchronous Communication Protocol
  - Full Duplex 8 bit data transfer
  - Start bit and stop bit generation and detection
  - Odd/Even or no Parity generation and detection
  - Receive Overrun, framing error and parity error detection

3.3.2 การออกแบบบล็อกไดอะแกรมของ UART

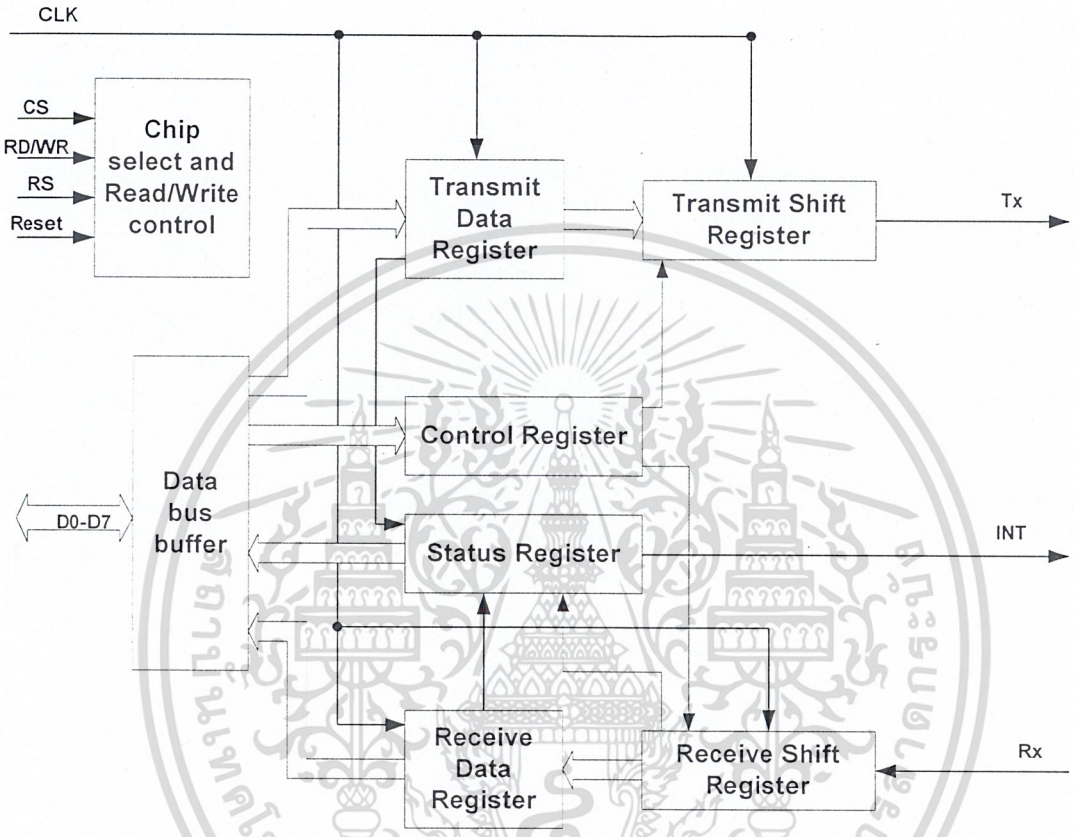
บล็อกไดอะแกรมของ UART ที่คณะผู้จัดทำออกแบบแสดงในรูปที่ 3-5 และมีรายละเอียดดังแสดงในรูปที่ 3-6 และแสดงรายละเอียดของพินในตารางที่ 3-1



รูปที่ 3-5 อินเทอร์เฟสของ UART (UART Interface Block)

Signal	Signal Direction	Description
UART Interface Signals For MPU		
RESET	Input	Chip reset . This input forces UART to be in predefined state ; all flags are reset.
CS	Input	Chip select. Indicate UART to work. Active high.
RS	Input	Register select. Select the internal register for read or write.
RD/WR	Input	Indicates read /write operation to UART ; 0: write,1: read
D0-D7	In/Out	Bidirectional data bus carries data to be written to internal registers; also data from registers during read cycle.
INT	Output	Interrupt to uP. Active low.
Clock Input		
CLK	Input	Master clock input.
Serial Input/Output Lines		
TX	Output	Serial data output from transmit shift register.
RX	Input	Serial data input to receive shift register.

ตารางที่ 3-1 ฟังก์ชันของอินพุท/เอาท์พุทของคอร์ของ UART ที่ออกแบบ

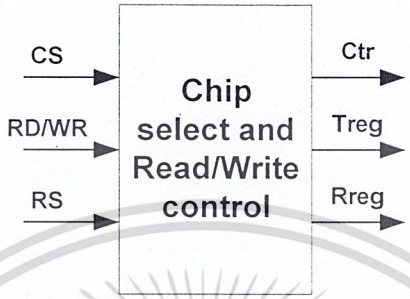


รูปที่ 3-6 บล็อกไดอะแกรมภายในของ UART ที่ได้ออกแบบ

3.3.3 รายละเอียดของแต่ละคอมโพเนนต์

3.3.3.1 สัญญาณควบคุมการเลือกชิพและอ่าน/เขียน (Chip select and Read/Write control)

ทำหน้าที่เลือกในการอ่านหรือเขียนรีจิสเตอร์ภายใน UART มีบล็อกไดอะแกรมดังรูปที่ 3-7 และมีฟังก์ชันการทำงานดังตารางที่ 3-2



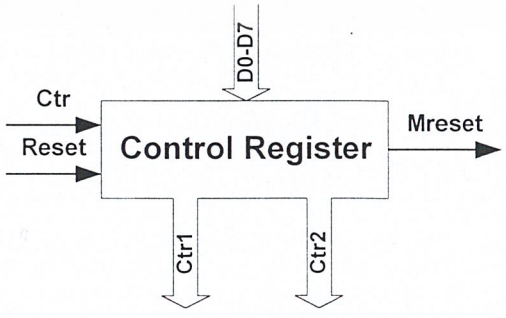
รูปที่ 3-7 บล็อกไดอะแกรมของพิน Chip select and read/write control

RS	R/W	Direction of data
0	0	From uP to control register
0	1	From status register to uP
1	0	From uP to transmit data register
1	1	From receive data register to uP

ตารางที่ 3-2 ฟังก์ชันการทำงานของพิน RS กับ RD/WR

3.3.3.2 รีจิสเตอร์ควบคุม (Control Register)

รีจิสเตอร์นี้ทำหน้าที่ในการกำหนดว่าจะให้ทำการรับ-ส่งข้อมูลแบบใด ดังบล็อกไดอะแกรมในรูปที่ 3-8 โดยแต่ละบิทมีหน้าที่การทำงานแสดงในรูปที่ 3-9



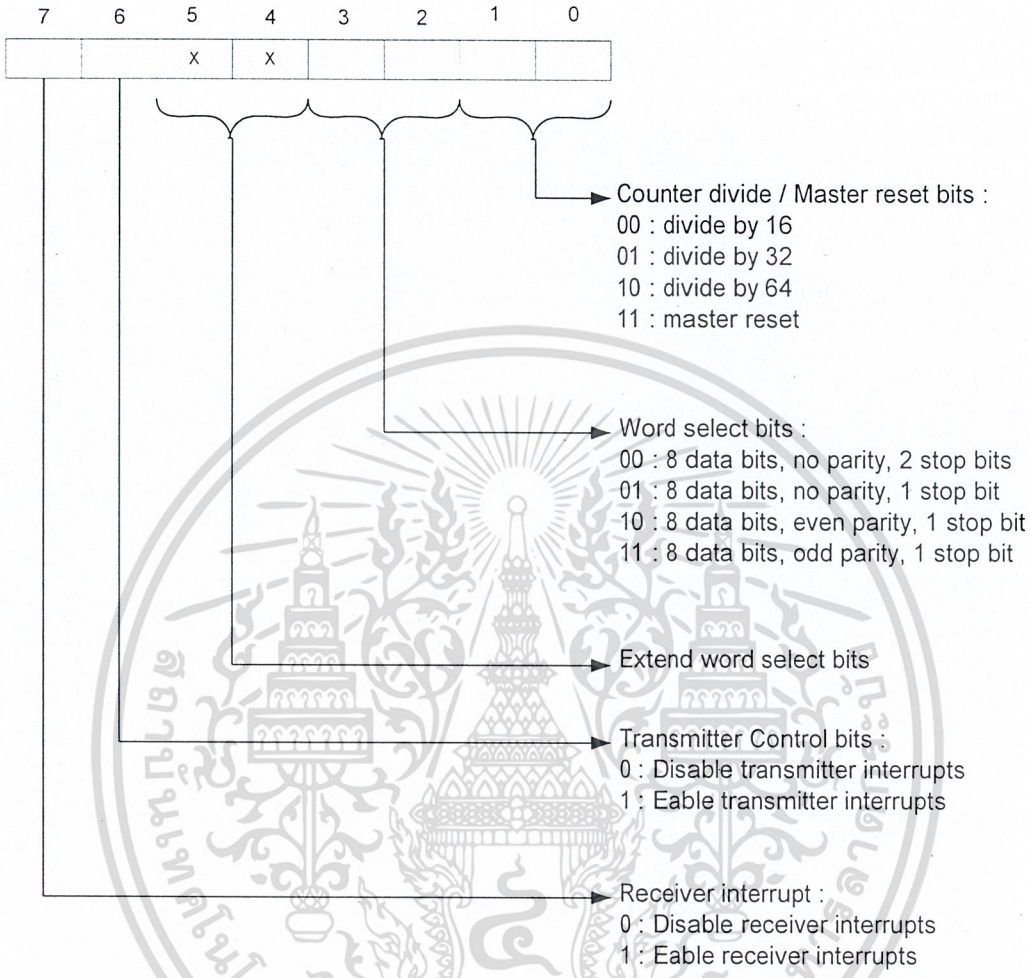
รูปที่ 3-8 บล็อกไดอะแกรมของรีจิสเตอร์ควบคุม

- Counter divide and Master reset (บิท 1และ บิท 0) : เป็นบิทที่ใช้กำหนดสัญญาณนาฬิกาที่ใช้ในการรับ-ส่งข้อมูล โดยการกำหนดค่าในแต่ละบิทเป็นดังตารางที่ 3-3

บิท 1	บิท 0	Function
0	0	Divide by 16
0	1	Divide by 32
1	0	Divide by 64
1	1	Master reset

ตารางที่ 3-3 การกำหนดสัญญาณนาฬิกาสำหรับ UART

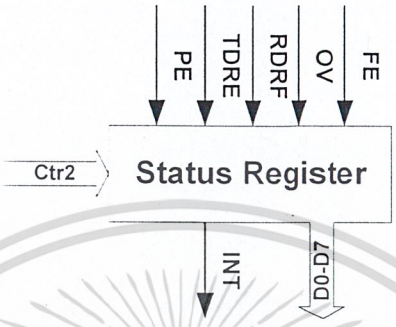
- Word select bit (บิท 3 และบิท 2) : ใช้กำหนดรูปแบบของข้อมูลที่ใช้ในการรับ-ส่ง ได้แก่ ความยาว(บิท), จำนวนบิทสิ้นสุด, รูปแบบบิทพาริตี้
- Transmit control bit (บิท6): เป็นบิทที่ใช้กำหนดการอินเทอร์รัปไมโครคอนโทรลเลอร์เมื่อรีจิสเตอร์ส่งข้อมูลว่าง โดยจะมีค่าเป็น 0 เมื่อ ไม่อนุญาตให้มีการอินเทอร์รัปและมีค่าเป็น 1 เมื่ออนุญาตให้มีการอินเทอร์รัป
- Receive interrupt bit (บิท7): เป็นบิทที่ใช้กำหนดการอินเทอร์รัปไมโครคอนโทรลเลอร์เมื่อรีจิสเตอร์รับข้อมูลเต็ม โดยจะมีค่าเป็น 0 เมื่อ ไม่อนุญาตให้มีการอินเทอร์รัปและมีค่าเป็น 1 เมื่ออนุญาตให้มีการอินเทอร์รัป



รูปที่ 3-9 การทำงานในแต่ละบิทของรีจิสเตอร์ควบคุม

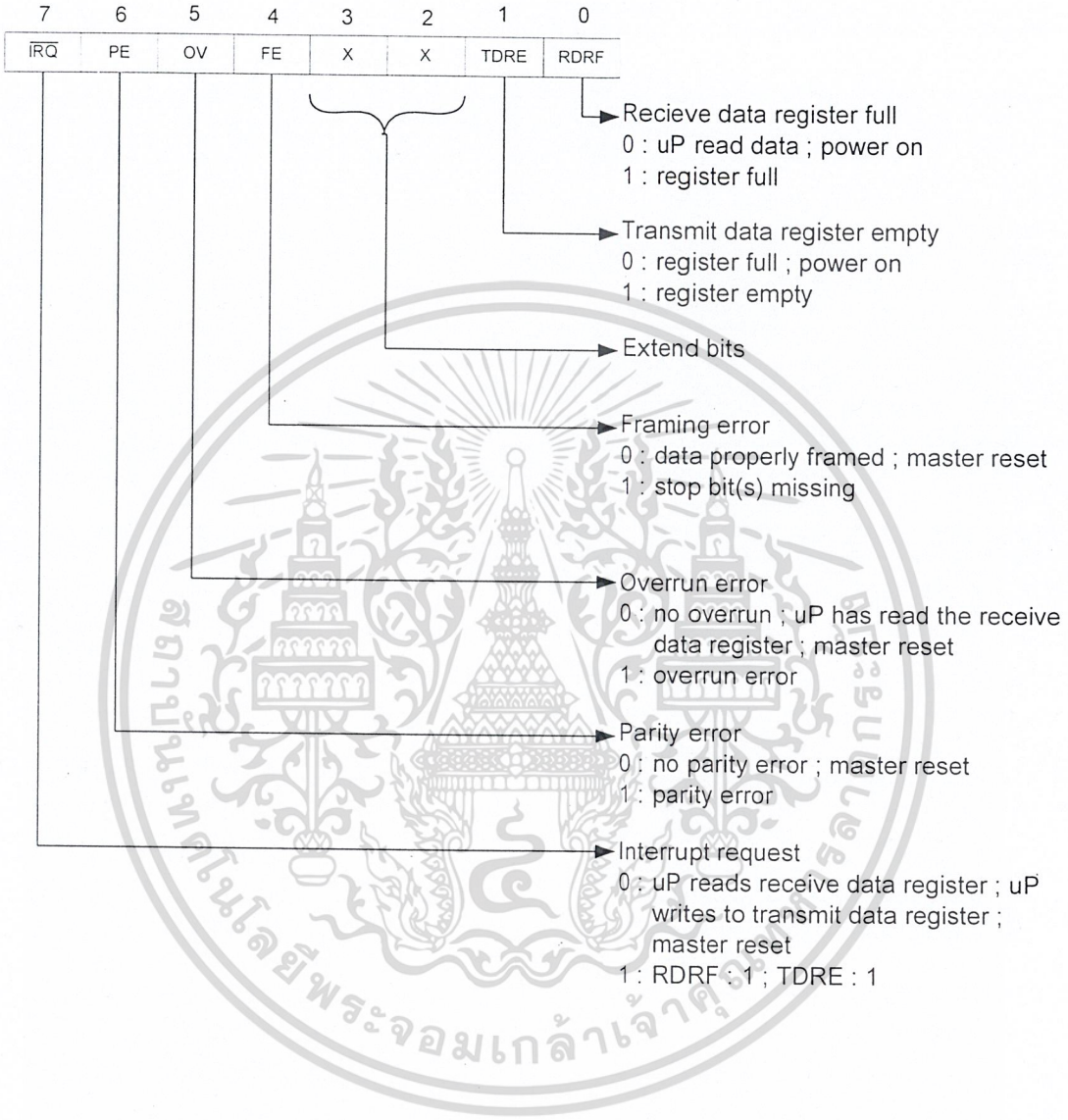
3.3.3.3 รีจิสเตอร์สถานะ (Status Register)

เป็นรีจิสเตอร์ที่ใช้รายงานสถานะการทำงานของการสื่อสารระหว่างฝ่ายรับและฝ่ายส่ง และสถานะของข้อมูลที่รับเข้ามา ดังบล็อกไดอะแกรมในรูปที่ 3-10 โดยแต่ละบิทในรีจิสเตอร์มีหน้าที่การทำงานดังแสดงในรูป 3-11



รูปที่ 3-10 บล็อกไดอะแกรมของรีจิสเตอร์สถานะ

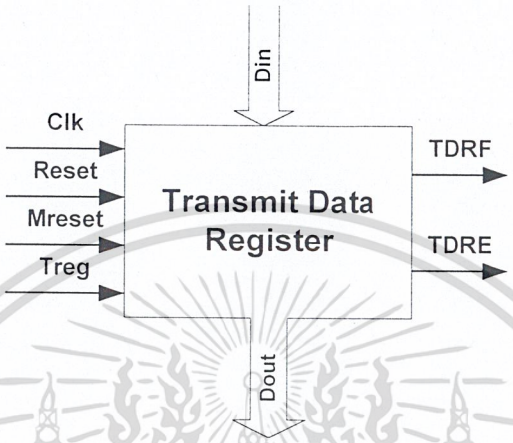
- RDRF : บิทนี้จะมีค่าเป็น '0' เมื่อรีเซตหรือ power-on หรือเมื่อ CPU อ่านค่าจากรีจิสเตอร์รับข้อมูลและจะมีค่าเป็น '1' เมื่อรับบิทข้อมูลมาครบ 1 เฟรม
- TDRE : บิทนี้จะมีค่าเป็น '0' เมื่อรีเซตหรือ power-on หรือ เมื่อ CPU เขียนค่าใส่รีจิสเตอร์ส่งข้อมูลและจะมีค่าเป็น '1' เมื่อรีจิสเตอร์ส่งข้อมูลทำการส่งค่าไปยังรีจิสเตอร์เลื่อนข้อมูลส่งออกแล้ว
- FE : บิทนี้จะมีค่าเป็น '0' เมื่อรีเซตหรือ power-on และจะมีค่าเป็น '1' เมื่อเฟรมข้อมูลที่รับเข้ามามีรูปแบบไม่ถูกต้อง ซึ่งเกิดจากมีจำนวนบิทสิ้นสุดไม่ตรงตามที่กำหนดไว้ในรีจิสเตอร์ควบคุม
- OV : บิทนี้จะมีค่าเป็น '0' เมื่อรีเซตหรือ power-on และจะมีค่าเป็น '1' เมื่อ UART รับข้อมูลใหม่เข้ามาแล้วจะนำไปเก็บในรีจิสเตอร์รับข้อมูล ในขณะที่ข้อมูลเดิมในรีจิสเตอร์รับข้อมูลนั้นยังไม่ได้ถูก CPU อ่านออกไป
- PE : บิทนี้จะมีค่าเป็น '0' เมื่อรีเซตหรือ power-on และจะมีค่าเป็น '1' เมื่อเฟรมข้อมูลที่รับเข้ามามีรูปแบบของบิทพาริตี้ไม่ถูกต้องตามที่กำหนดไว้ในรีจิสเตอร์ควบคุม
- IRQ : บิทนี้จะมีค่าเป็น '0' เมื่อรีเซตหรือ power-on และจะมีค่าเป็น '1' เมื่อ UART ทำการอินเทอร์รัป CPU ซึ่งจะสามารถทำการอินเทอร์รัปได้ถ้าค่า บิท 7 หรือ บิท 6 ใน Control register มีค่าเป็น '1' เพราะทั้ง 2 บิทนี้เป็นบิทที่กำหนดว่าเมื่อรีจิสเตอร์รับข้อมูลเต็ม (บิท RDRF มีค่าเป็น '1') หรือเมื่อรีจิสเตอร์ส่งข้อมูลว่าง (บิท TDRE มีค่าเป็น '1') จะให้ทำการอินเทอร์รัปหรือไม่



รูปที่ 3-11 การทำงานในแต่ละบิทของรีจิสเตอร์สถานะ

3.3.3.4 รีจิสเตอร์ส่งข้อมูล (Transmit Data Register)

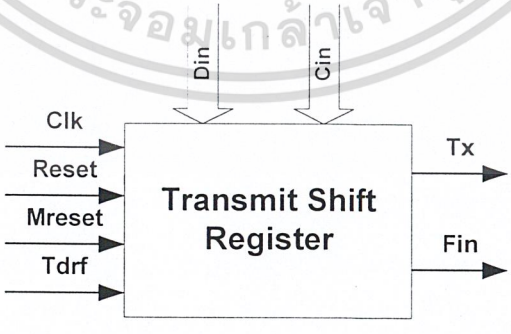
รับข้อมูล 8 บิทจาก MPU และส่งให้รีจิสเตอร์เลื่อนข้อมูลส่งออกเมื่อรีจิสเตอร์เลื่อนข้อมูลส่งออกนี้ส่งข้อมูลของมันเสร็จ และเมื่อรีจิสเตอร์เลื่อนข้อมูลส่งออกมารับข้อมูลไปแล้วรีจิสเตอร์ส่งข้อมูลจะว่างและเซตสัญญาณ TDRE ซึ่งเป็นอินพุทของรีจิสเตอร์สถานะให้เป็น '1' มีบล็อกไดอะแกรมดังรูปที่ 3-12



รูปที่ 3-12 บล็อกไดอะแกรมของรีจิสเตอร์ส่งข้อมูล

3.3.3.5 รีจิสเตอร์เลื่อนข้อมูลส่งออก (Transmit Shift Register)

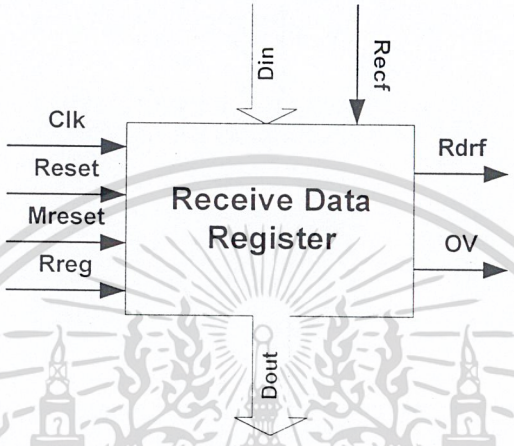
ทำหน้าที่ส่งข้อมูลอนุกรมออกไปทางขา Tx โดยทำการรับข้อมูล 8 บิท จากรีจิสเตอร์ส่งข้อมูลแล้วนำมาจัดเป็นเฟรม คือเพิ่มบิทเริ่มต้น, บิทพาริตี้และบิทสิ้นสุดเข้าไปตามรูปแบบที่ระบุไว้ในรีจิสเตอร์ควบคุมและส่งข้อมูลออกไปแบบอนุกรมด้วยความเร็วตาม counter divide bit ในรีจิสเตอร์ควบคุม มีบล็อกไดอะแกรมดังรูปที่ 3-13



รูปที่ 3-13 บล็อกไดอะแกรมของรีจิสเตอร์เลื่อนข้อมูลส่งออก

3.3.3.6 รีจิสเตอร์รับข้อมูล (Receive Data Register)

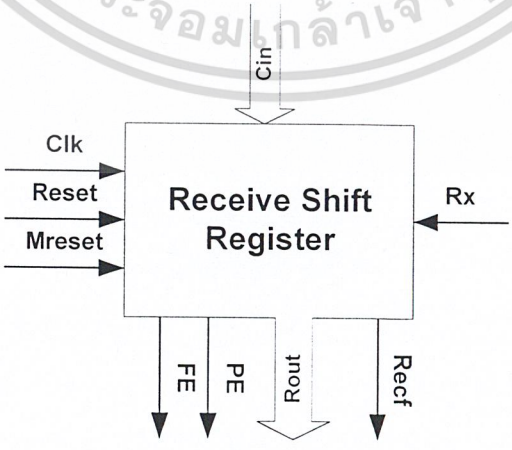
รับข้อมูล 8 บิทจากรีจิสเตอร์เลื่อนข้อมูลรับเข้ามาเก็บไว้และเซตสัญญาณ Rdrf เป็น '1' และถ้ารีจิสเตอร์เลื่อนข้อมูลรับเข้าส่งข้อมูลมาให้ในขณะที่ข้อมูลเดิมในรีจิสเตอร์รับข้อมูลยังไม่ได้ถูกอ่านจะเกิดการโอเวอร์รัน (overrun) และจะเซตสัญญาณ OV เป็น '1' โดยสัญญาณ Rdrf และ OV นั้นจะไปเป็นอินพุทของรีจิสเตอร์สถานะ มีบล็อกไดอะแกรมดังรูปที่ 3-14



รูปที่ 3-14 บล็อกไดอะแกรมของรีจิสเตอร์รับข้อมูล

3.3.3.7 Receive Shift Register

รับข้อมูลอนุกรมจาก Rx โดยใช้ความเร็วในการรับตาม counter divide bit ในรีจิสเตอร์ควบคุม แล้วตรวจสอบเฟรมข้อมูลและบิทพาริตี้ว่าตรงตามที่กำหนดไว้ในรีจิสเตอร์ควบคุม (cin) หรือไม่ แล้วส่งข้อมูล 8 บิทให้รีจิสเตอร์รับข้อมูลและส่งผลการตรวจสอบรูปแบบข้อมูล (FE) บิทพาริตี้ (PE) ให้รีจิสเตอร์สถานะ มีบล็อกไดอะแกรมดังรูปที่ 3-15

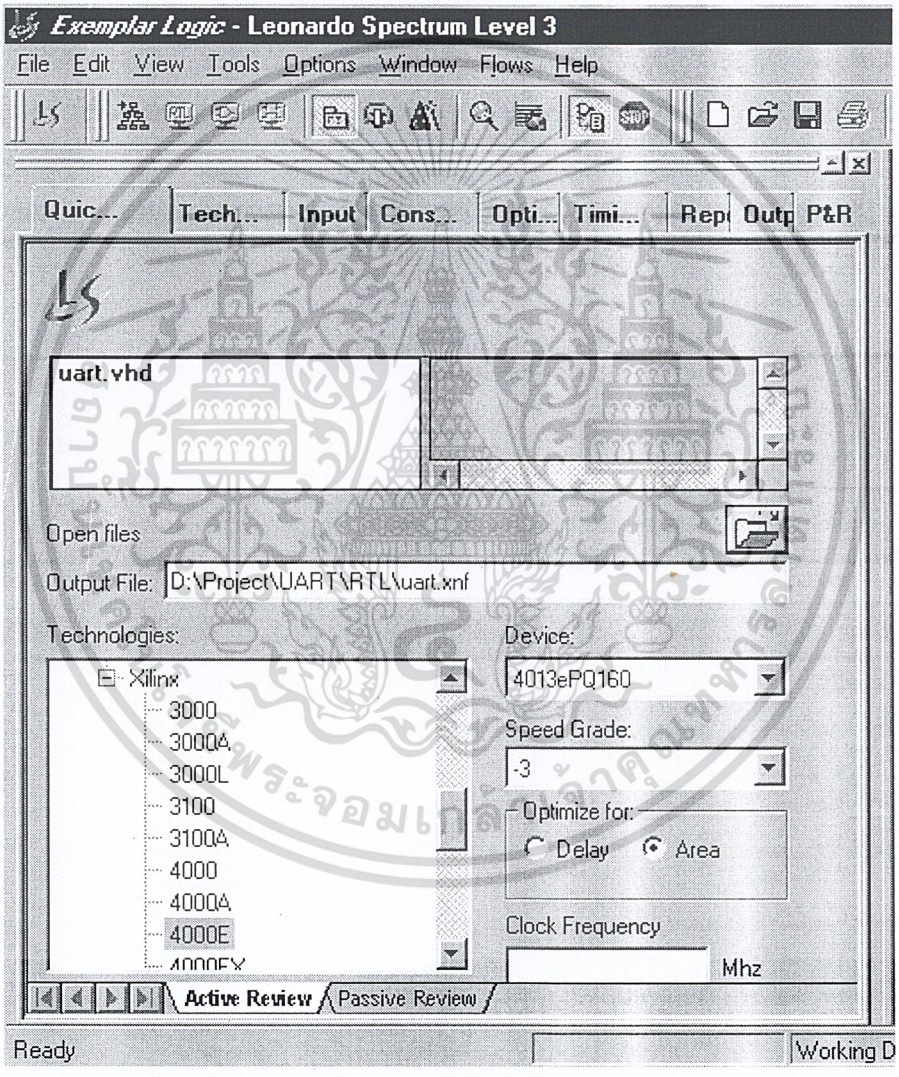


รูปที่ 3-15 บล็อกไดอะแกรมของรีจิสเตอร์เลื่อนข้อมูลรับเข้า

3.4 การสังเคราะห์วงจร (Synthesis)

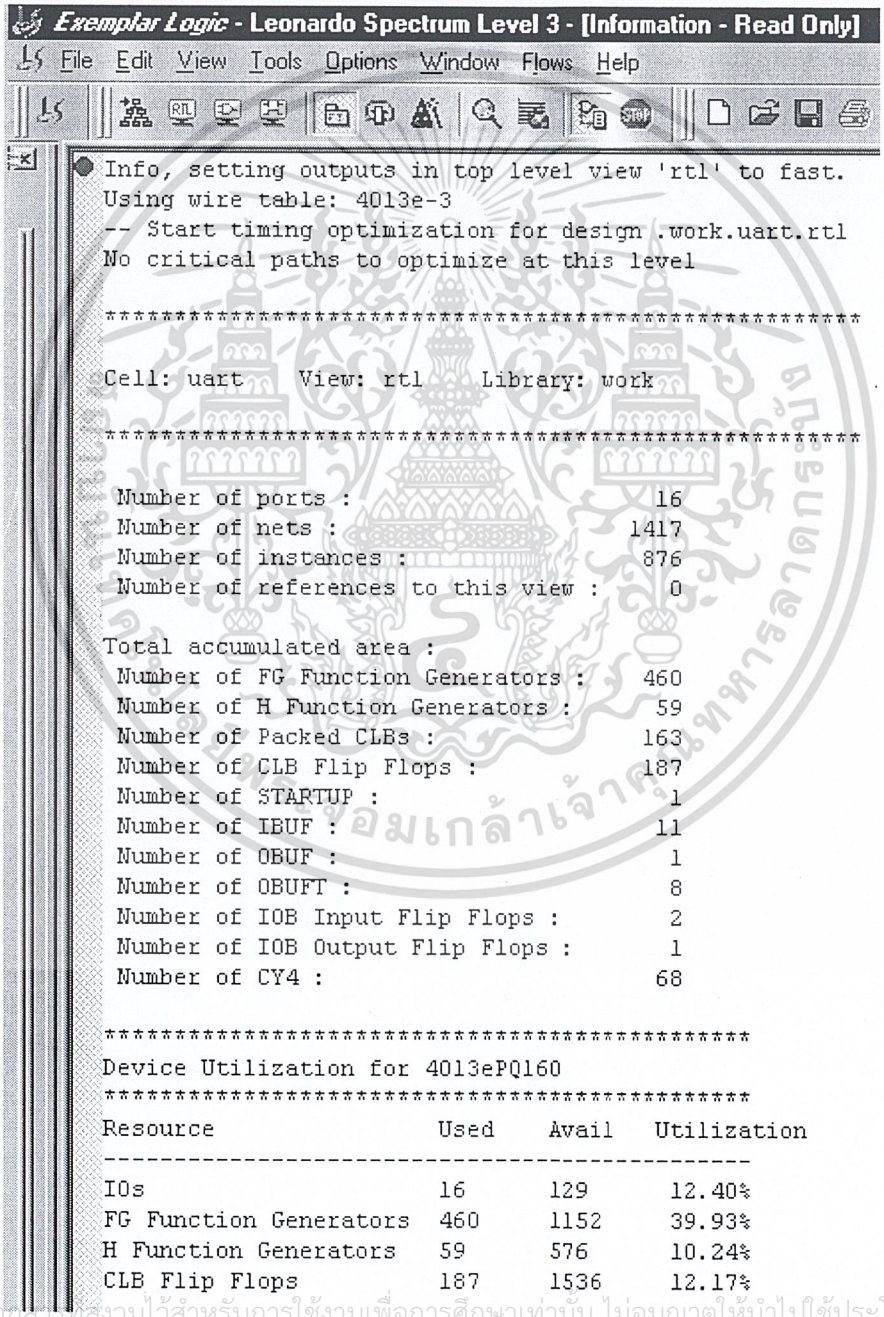
การสังเคราะห์วงจรเป็นการเปลี่ยนระบบที่ออกแบบมาซึ่งอยู่ในลักษณะ Register Transfer Level (RTL) description เป็น gate-level netlist ที่สามารถนำไปทำการแมป (Mapping) กับ FPGA ได้ โดยในโครงงานนี้ใช้ซอฟต์แวร์ Leonardo Spectrum ทำการสังเคราะห์วงจร โดยมีขั้นตอนดังนี้

- เลือกไฟล์ vhdl ที่จะทำการสังเคราะห์
- เลือกอุปกรณ์ที่จะใช้ในการแมป โดยในโครงงานนี้ใช้ FPGA : XC4013e PQ160 ของ Xilinx ดังรูปที่ 3-16



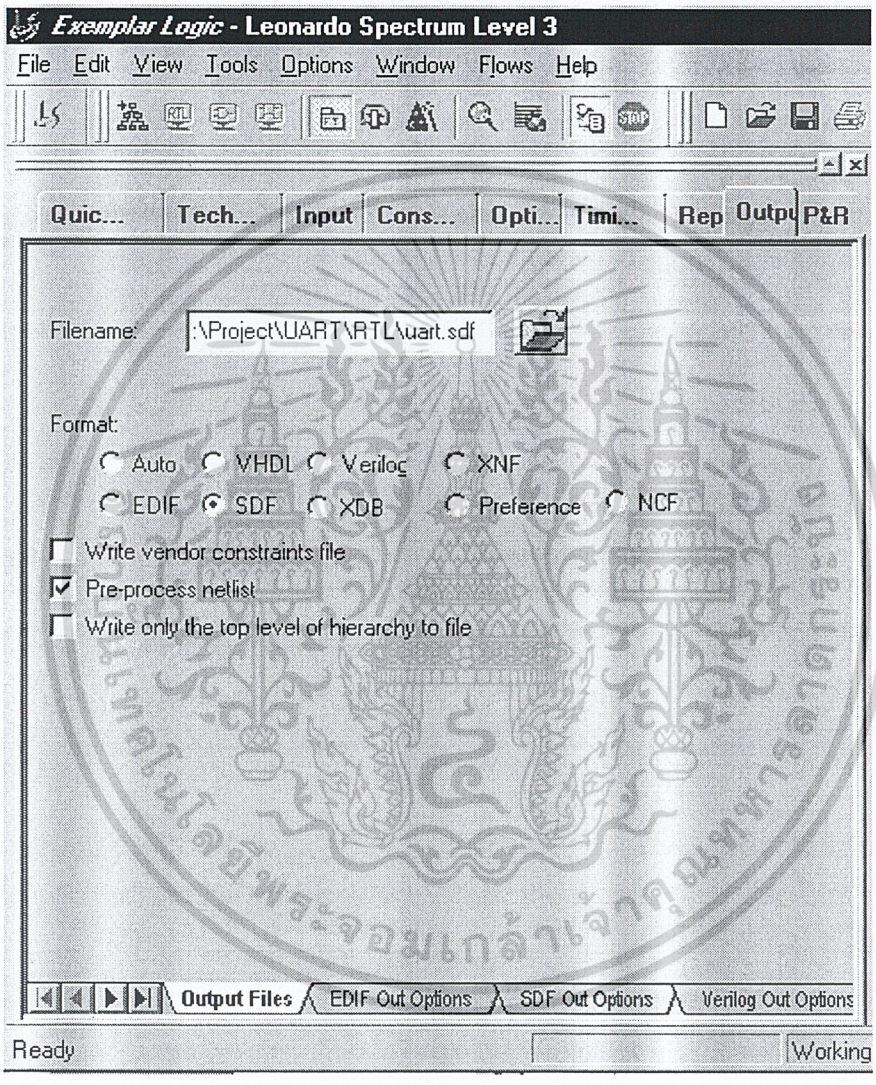
รูปที่ 3-16 การใช้ Leonardo Spectrum ในการสังเคราะห์วงจร

- ใช้คำสั่ง Run Flow ซอฟต์แวร์จะทำการตรวจสอบไฟล์ Vhdl ว่ามีรูปแบบถูกต้องที่สามารถจะ Map เป็น gate level netlist ได้หรือไม่ เพราะการเขียนโปรแกรมภาษา VHDL ในบางลักษณะจะไม่สามารถแมปได้ ถ้าเกิดสังเคราะห์ไม่ผ่านจะแสดงข้อความแสดงความผิดพลาด (error message) ขึ้นมาว่าเกิดจากอะไร แล้วต้องไปแก้โปรแกรมให้อยู่ในลักษณะที่สามารถสังเคราะห์ได้
- เมื่อทำการสังเคราะห์ผ่านจะได้ผลดังรูปที่ 3-17 ซึ่งประกอบด้วยข้อมูลของ UART ที่ได้ออกแบบมาว่ามีการใช้ทรัพยากร (resource) ใน FPGA เป็นจำนวนเท่าไหร่



รูปที่ 3-17 ผลการสังเคราะห์วงจรของ UART

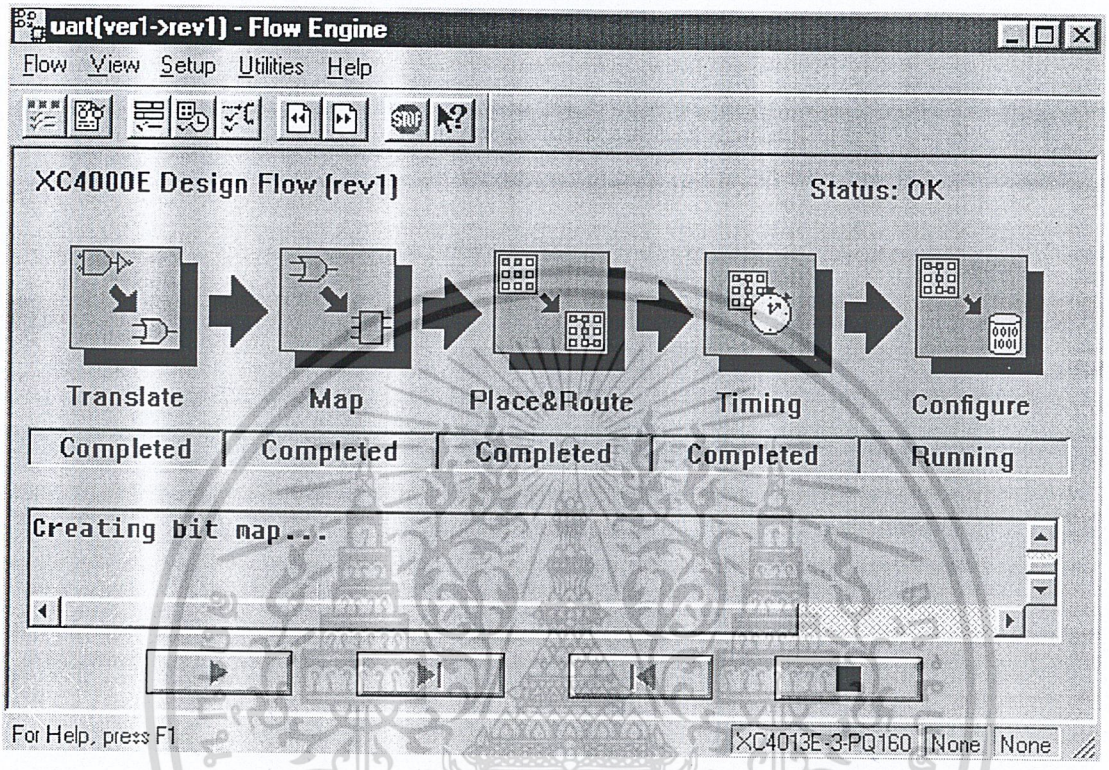
- เมื่อสังเคราะห์ผ่านแล้ว ก็สั่งให้ซอฟต์แวร์เขียนไฟล์เอาท์พุทออกมาใน 3 รูปแบบ คือ ไฟล์ .vhd , ไฟล์ .sdf และ ไฟล์ .xnf โดยไฟล์ .vhd และไฟล์ .sdf จะนำไปใช้ในขั้นตอนการ Simulation for Place & Route เพื่อตรวจสอบว่าทำงานได้ถูกต้องตาม specification แล้วจึงนำไฟล์ .xnf ไปใช้เป็นอินพุทไฟล์ในการ Place & Route โดย Xilinx Foundation Series ดังรูปที่ 3-18



รูปที่ 3-18 ไฟล์เอาท์พุทจาก Leonardo Spectrum

3.5 การ Place & Route

ในโครงงานนี้ใช้ซอฟต์แวร์ Xilinx Foundation Series เพื่อการ Place & Route ซึ่งซอฟต์แวร์จะทำงานตามขั้นตอนดังรูปที่ 3-19



รูปที่ 3-19 ขั้นตอนการทำงานของ Xilinx Foundation Series

โดยในขั้นตอนไทม์มิ่ง (Timing) จะได้ไฟล์ timsim.vhd และ timsim.sdf ซึ่งจะนำไปใช้ในขั้นตอนการทำ Gate Leval Functional Simulation และในขั้นตอนการคอนฟิกจะได้ไฟล์ .bit ซึ่งจะนำไปใช้ในการโปรแกรมระบบที่เราออกแบบลงใน FPGA ซึ่งมีผลการ Place & Route แสดงดังรูปที่ 3-20

หลังจากทำการ Place & Route แล้วจะได้ไฟล์รายงานการแมปและ Place & Route ว่าคอร์ของ UART ที่ได้ออกแบบไว้มีขนาดเท่าใด และมีการใช้ทรัพยากรต่างๆ ใน FPGA อย่างไรบ้าง โดยคอร์ของ UART ที่ได้ออกแบบในโครงงานนี้เมื่อลง FPGA แล้ว มีขนาดที่วัดเป็นจำนวน CLB เท่ากับ 351 CLB และขนาดที่วัดเป็นจำนวนเกตเท่ากับ 5398 เกต

```
                Xilinx Mapping Report File for Design "Uart"
          Copyright (c) 1995-1997 Xilinx, Inc.  All rights reserved.

Design Information
------------------
Command Line   : map -p xc4013e-3-pq160 -o map.ncd -r -pr b ../xc4000e.ngd
Uart.pcf
Target Device  : x4013e
Target Package : pq160
Target Speed   : -3
Mapper Version : xc4000e -- M1.4.12
Mapped Date    : Mon Apr 12 17:53:32 1999

Design Summary
--------------
   Number of errors:        0
   Number of warnings:      4
   Number of CLBs:            351 out of   576   60%
      CLB Flip Flops:     190
      4 input LUTs:       590 (128 used as route-throughs)
      3 input LUTs:        60
   Number of bonded IOBs:     16 out of   129   12%
      IOB Flops:           0
      IOB Latches:         0
   Number of global buffers:   1 out of     8   12%
   Number of primary CLKs:     1 out of     4   25%
   Number of RPM macros:       4
Total equivalent gate count for design: 5398
Additional JTAG gate count for IOBs:    768
```

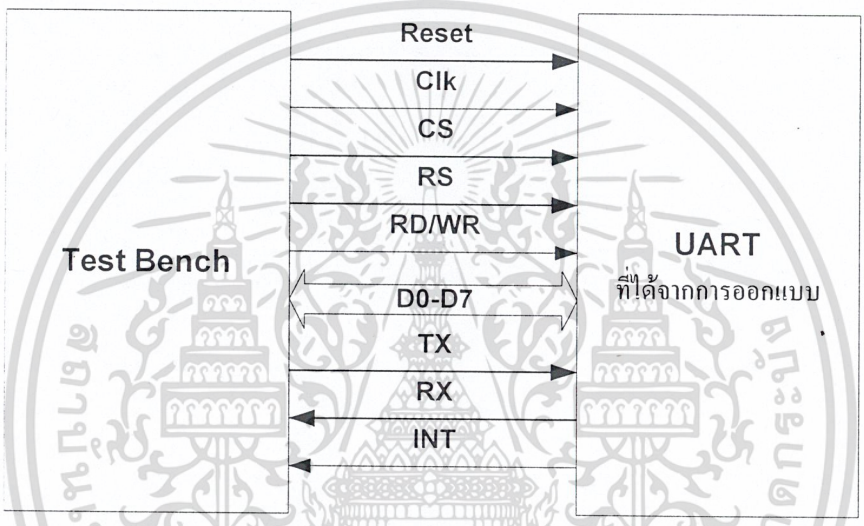
รูปที่ 3-20 ไฟล์รายงานผลการ Place & Route ของ Xilinx

3.6 แนวทางการทดสอบ (Test Strategy)

ในการทดสอบระบบจะต้องสามารถตรวจสอบได้ว่าระบบที่ออกแบบมานั้นทำงานได้ถูกต้องตาม specification โดยแบ่งการทดสอบออกเป็น 2 ส่วน ได้แก่ การทดสอบโดยการจำลองการทำงานและการทดสอบในระดับฮาร์ดแวร์โดยใช้ FPGA

- การทดสอบโดยการจำลองการทำงาน

ใช้การสร้าง Test Bench ขึ้นมาทำหน้าที่ในการป้อนสัญญาณและควบคุมการทำงานของ UART ที่ได้ออกแบบมาให้ทำการรับ-ส่ง ข้อมูล ดังรูปที่ 3-21

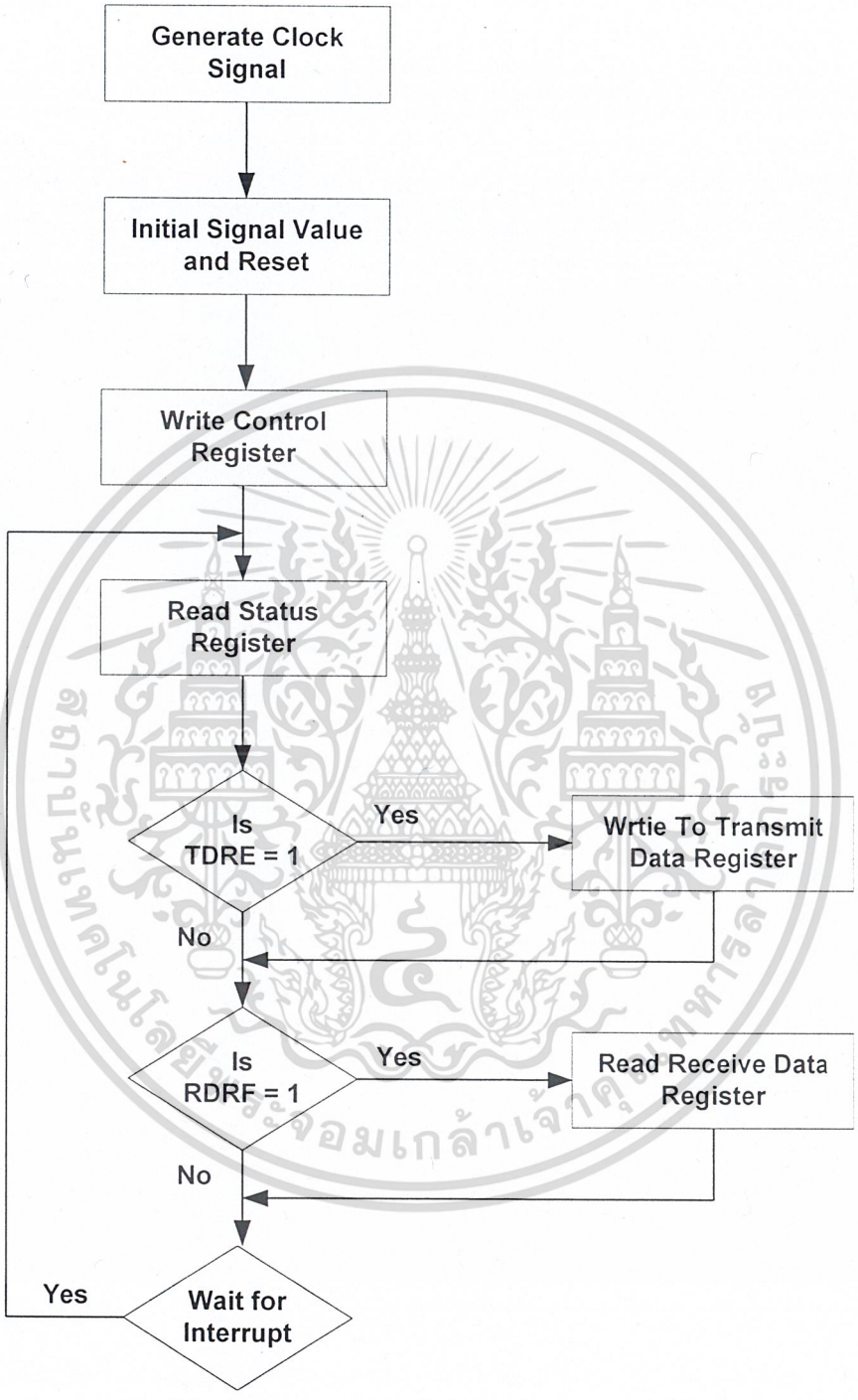


รูปที่ 3-21 การทดสอบโดยใช้ Test Bench

โดยจะทำการตรวจสอบในคุณสมบัติของระบบดังต่อไปนี้

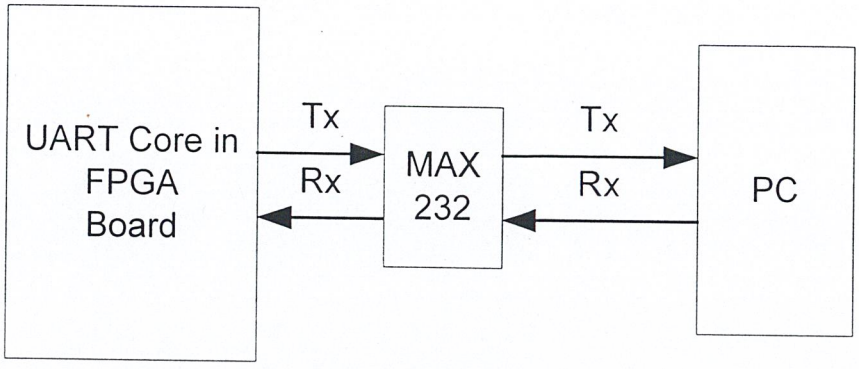
- สามารถรับส่งข้อมูลอนุกรมได้ถูกต้องตาม Specification โดยทดสอบระบบในทุกโหมดการทำงาน คือ
  - 8 bits , Non parity , 2 stop bits
  - 8 bits , Non parity , 1 stop bit
  - 8 bits , Even parity , 1 stop bit
  - 8 bits , Odd parity , 1 stop bit
- สามารถทำงานสัมพันธ์กับคำสั่งควบคุมในรีจิสเตอร์ควบคุมได้ถูกต้อง
- สามารถทำการอินเทอร์รัปและทำงานได้ถูกต้องตามการควบคุมของ Test Bench
- ในการรับ-ส่งข้อมูลอนุกรม บิทสัญญาณในรีจิสเตอร์สถานะต้องสามารถแสดงสถานะได้อย่างถูกต้อง

Test Bench ที่ใช้ในการตรวจสอบมีแผนผังการทำงานดังรูปที่ 3-22 นี้



รูปที่ 3-22 แผนผังการทำงานของ Test Bench

- การทดสอบในระดับฮาร์ดแวร์ (FPGA Prototype) ดังรูปที่ 3-23



รูปที่ 3-23 ระบบที่ใช้ทดสอบคอร์ของ UART

ขั้นตอนการทดสอบในระดับฮาร์ดแวร์จะใช้ FPGA ซึ่งถูกโปรแกรมเป็น UART ที่ได้ออกแบบไว้มาต่อกับบอร์ดไมโครคอนโทรลเลอร์ 89C51 ในส่วนของ MPU interface ของ UART และในส่วนของการสื่อสารอนุกรมจะต่อกับพอร์ตสื่อสารอนุกรมของคอมพิวเตอร์ PC โดยการตรวจสอบความถูกต้องของการสื่อสารจะใช้โปรแกรม PCPLUS เพื่อดูค่าของข้อมูลที่ไมโครคอนโทรลเลอร์ส่งมาโดยผ่าน UART ใน FPGA และดูค่าข้อมูลที่คอมพิวเตอร์ส่งให้ไมโครคอนโทรลเลอร์จาก LED บนบอร์ดไมโครคอนโทรลเลอร์

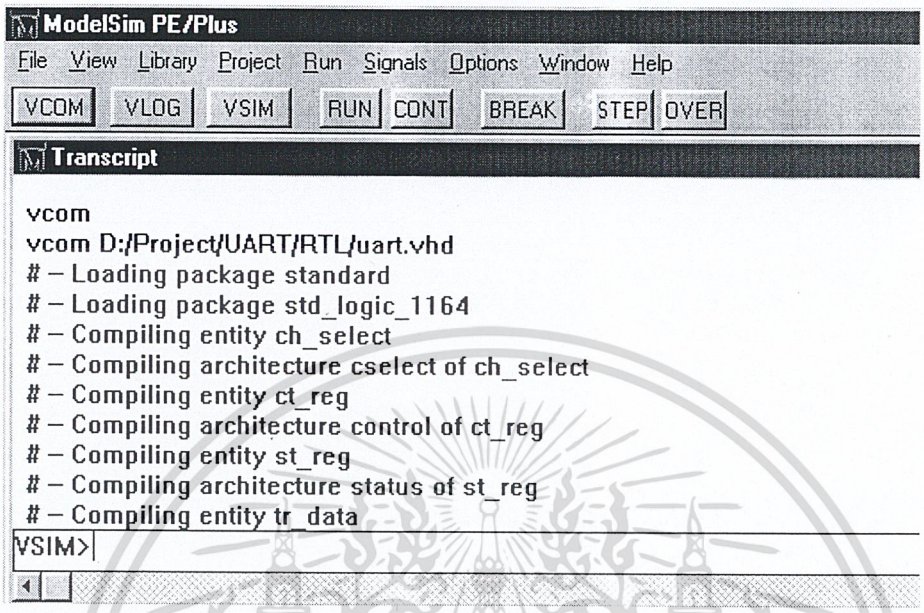
3.7 ขั้นตอนการทดสอบ (Test Plan)

ในโครงงานนี้ได้แบ่งขั้นตอนการทดสอบเป็น 2 ส่วนคือ

3.7.2 การทดสอบโดยการจำลองการทำงาน

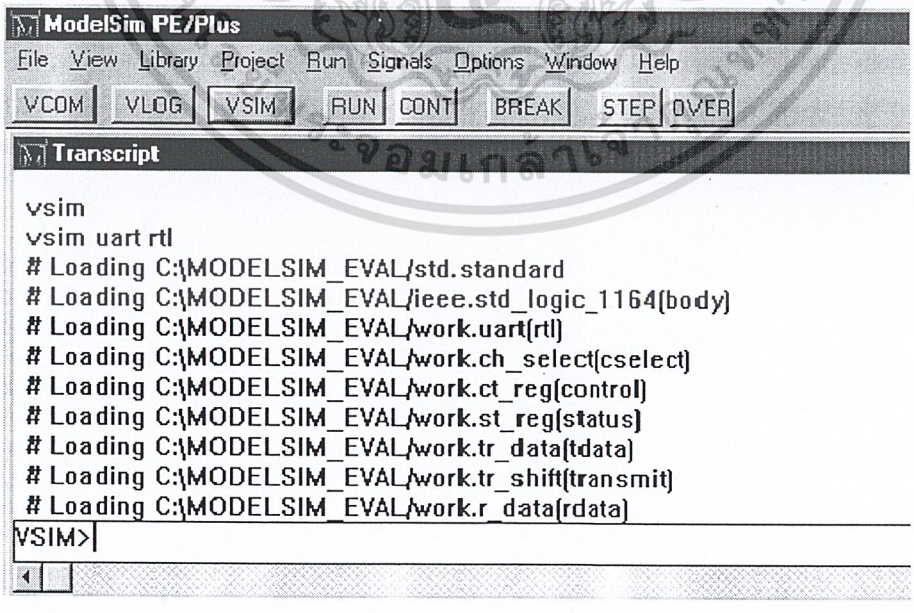
ขั้นตอนในการจำลองการทำงานเริ่มจากการคอมไพล์ไฟล์ VHDL โดยใช้ซอฟต์แวร์ ModelSim แล้วก็จะเริ่มตรวจสอบการทำงานของระบบ โดยใช้การจำลองการทำงานบนซอฟต์แวร์ ModelSim เช่นเดียวกัน โดยมีขั้นตอนดังนี้

1. ทำการคอมไพล์ไฟล์ VHDL ของระบบที่ได้ออกแบบไว้ โดยการใช้คำสั่ง Vcom หรือเรียกใช้จากเมนู ดังรูปที่ 3-24



รูปที่ 3-24 การคอมไพล์ไฟล์ VHDL โดยใช้ ModelSim

2. เมื่อผลการตรวจสอบไฟล์ VHDL โดย ModelSim ไม่มีปัญหา ก็จะทำการจำลองการทำงานโดยการเรียกใช้คำสั่ง Vsim แล้วเลือกเอนติตี้ (Entity) ของระบบที่เราออกแบบ ดังรูปที่ 3-25



รูปที่ 3-25 การจำลองการทำงานโดยใช้คำสั่ง Vsim

3. ทำการป้อนค่าสัญญาณอินพุทของระบบ แล้วใช้คำสั่ง Run ดูผลการทำงานของระบบว่าตรงกับที่กำหนดไว้หรือไม่ โดยการป้อนค่าอาจจะป้อนโดยตรงโดยใช้คำสั่ง Force หรือเขียนโปรแกรม VHDL ขึ้นมาเพื่อทำงานแทนในการป้อนค่าให้กับระบบ

โดยจะต้องทำการจำลองการทำงานของระบบทั้งหมด 4 ระดับ โดยในการจำลองการทำงานในแต่ละระดับจะเป็นเพิ่มความถูกต้องของระบบมากขึ้นเพราะจะเพิ่มปัจจัยต่าง ๆ ที่จะเกิดขึ้นจริงทางฮาร์ดแวร์เข้ามาด้วย โดยมีรายละเอียดของการจำลองการทำงานในแต่ละระดับดังนี้

3.7.2.1 Behavioral Simulation

นำไฟล์ VHDL ที่เขียนแบบ behavior มาทำการจำลองการทำงานโดยใช้ Testbench ถ้าผลการจำลองการทำงานยังไม่ถูกต้องก็กลับไปแก้ไขโปรแกรมแล้วนำมาจำลองการทำงานดูใหม่ จนกว่าผลของสัญญาณจากการทำงานของระบบจะถูกต้องตาม specification และผลของสัญญาณในการจำลองการทำงานในระดับนี้จะถูกนำไปใช้เป็น specification ของสัญญาณ ของการจำลองการทำงานของระบบในระดับต่อ ๆ ไป

3.7.2.2 RTL Simulation

นำไฟล์ VHDL ที่เขียนแบบ RTL มาทำการจำลองการทำงานโดยใช้ Test Bench ถ้าผลการจำลองการทำงานยังไม่ถูกต้องก็กลับไปแก้ไขโปรแกรมแล้วนำมาจำลองการทำงานดูใหม่ จนกว่าผลของสัญญาณจากการทำงานของระบบจะถูกต้องตาม specification

3.7.2.3 Simulation For Place and Route

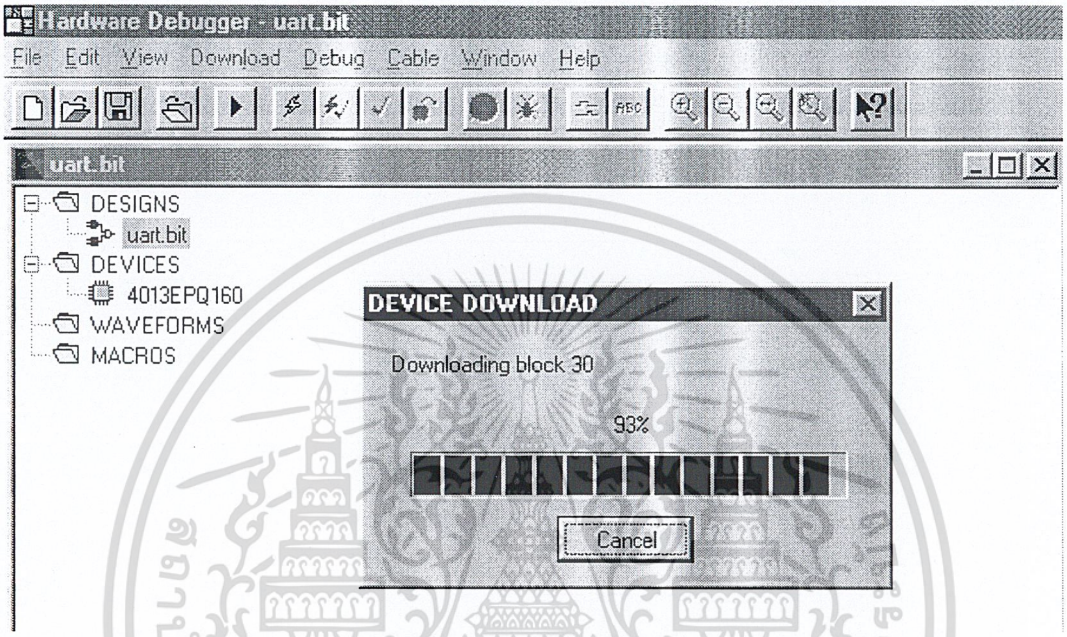
หลังจากการสังเคราะห์โดยใช้ Leonardo Spectrum จะใช้ฟล์ภาษา VHDL และ ไฟล์นามสกุล sdf ซึ่งเป็นไฟล์ดีเลย์ที่ได้จากซอฟต์แวร์ Leonardo Spectrum มาจำลองการทำงานโดยใช้ Test Bench เดิม โดยการจำลองการทำงานในระดับนี้ ระบบจะทำงานโดยมีค่าดีเลย์จากเกตด้วย

3.7.2.4 Gate Level Funtional Simulation

หลังจากการ Place & Route โดยใช้ Xilinx Foundation Series จะใช้ไฟล์ภาษา VHDL และ ไฟล์นามสกุล sdf ซึ่งเป็นไฟล์ดีเลย์ที่ได้จากซอฟต์แวร์ Xilinx Foundation Series มาจำลองการทำงานโดยใช้ Test Bench เดิม โดยการจำลองการทำงานในระดับนี้ ระบบจะทำงานโดยมีดีเลย์จากเกต, จาก setup time, hold time และค่าดีเลย์จริงของ FPGA ที่ใช้ในการ Place & Route

3.7.3 การทดสอบในระดับฮาร์ดแวร์ (FPGA Prototype)

หลังจากใช้ซอฟต์แวร์ Xilinx Foundation Series ทำการ Place & Route และทำการทดสอบในขั้นตอน Gate Level Functional Simulation แล้ว ก็เรียกส่วน Hardware Debugger ขึ้นมาเพื่อทำการโปรแกรม UART ที่ได้ออกแบบลงบน FPGA โดยใช้ไฟล์ .bit ที่ได้จากขั้นตอนการคอนฟิก ดังรูปที่ 3-26



รูปที่ 3-26 การโปรแกรม UART ลงบน FPGA XC4013EPQ160

3.7.3.1 ระบบทดสอบ UART core ในระดับฮาร์ดแวร์

การทดสอบในระดับนี้ต้องทำการเซตอัพ (set up) และทดสอบการทำงานของอุปกรณ์และบอร์ดที่ใช้ให้แน่ใจว่าทำงานได้ถูกต้องก่อนจึงนำมาทดสอบคอร์ของ UART ดังรูปที่ 3-27

ระบบที่ใช้ทดสอบประกอบด้วย

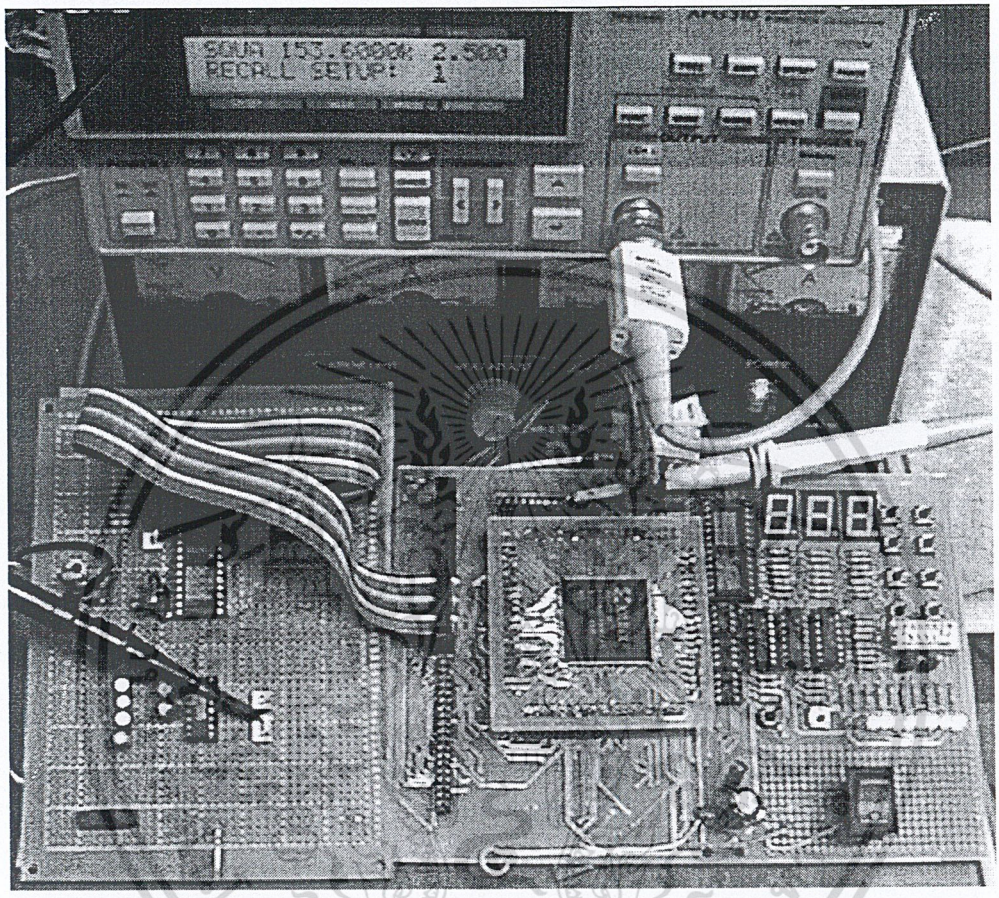
- บอร์ดไมโครคอนโทรลเลอร์ MCS51

ทำการทดสอบโดยการเขียนโปรแกรมแสดงผล LED และทดสอบ MAX232 โดยการเขียนโปรแกรมให้ MCS51 สื่อสารข้อมูลอนุกรม UART กับคอมพิวเตอร์พีซี ผ่าน MZX232

- บอร์ด FPGA

ในโครงงานนี้ใช้ FPGA ของ Xilinx เบอร์ XC4013EPQ160 ในการทดสอบบอร์ด FPGA ประกอบด้วยการทดสอบการทำงานภายใน FPGA โดยการเขียนโปรแกรม VHDL ให้ FPGA ทำงานเป็น Counter โดยรับสัญญาณนาฬิกาจาก Function Generator แล้วนำมานับและแสดงผลออกทาง LED และการทดสอบส่วนอินเทอร์เฟสของ FPGA กับ บอร์ดไมโครคอนโทรลเลอร์ โดยการเขียนโปรแกรม Counter บน MCS51 แล้วให้ส่งค่าจากบอร์ด MCS51 มายังบอร์ด FPGA ให้นำมาแสดงผลที่ LED บนบอร์ด FPGA

- Function Generator ใช้จ่ายสัญญาณนาฬิกาให้กับ FPGA โดยในการทดสอบ UART จะใช้ความถี่สัญญาณนาฬิกา 153.6 KHz
- Power Supply ใช้จ่ายไฟให้กับบอร์ดไมโครคอนโทรลเลอร์และบอร์ด FPGA



รูปที่ 3-27 ระบบที่ใช้ทดสอบคอร์ของ UART ในระดับฮาร์ดแวร์

บทที่ 4
ผลการทดสอบระบบ

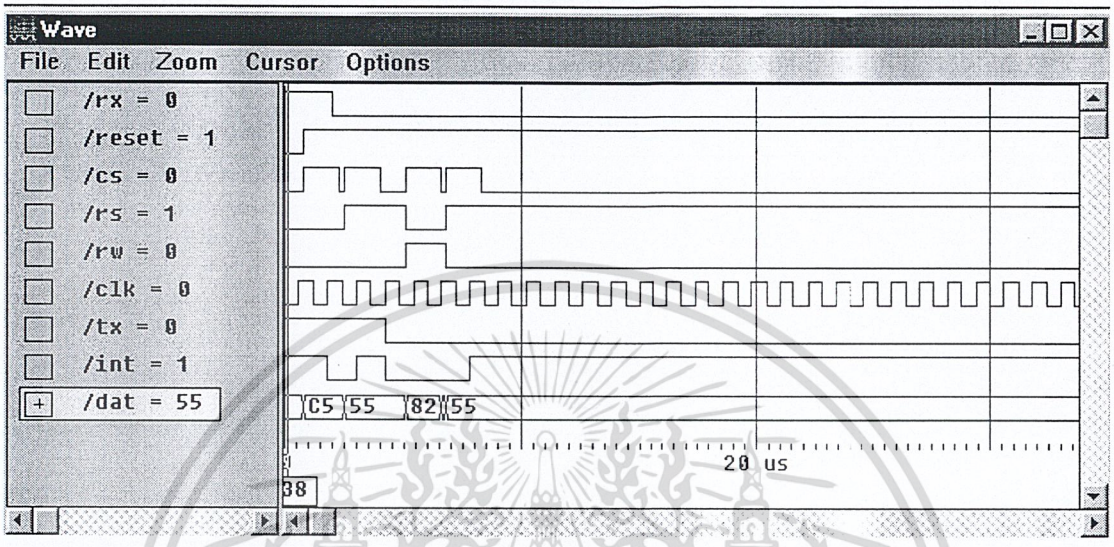
4.1 ผลการทดสอบโดยการจำลองการทำงาน

ผลการตรวจสอบโดยการจำลองการทำงานในแต่ละหน้าที่การทำงานของระบบเป็นดังตาราง 4-1

Chip select and read/write control	สามารถรับสัญญาณ cs,rs และ rw แล้วทำการเลือกการอ่านเขียนรีจิสเตอร์ในระบบได้ถูกต้อง
รีจิสเตอร์ควบคุม	เมื่อถูกเขียนค่า 8 บิทที่ใช้ควบคุมได้แล้วสามารถส่งสัญญาณการควบคุมไปยังรีจิสเตอร์อื่นได้ถูกต้อง
รีจิสเตอร์สถานะ	RDRF : มีค่าเป็น '0' เมื่อรีเซตหรือ power-on หรือCPU อ่านค่าจากรีจิสเตอร์รับข้อมูลและจะมีค่าเป็น '1' เมื่อรับบิทข้อมูลมาครบ 1 เฟรม TDRE : มีค่าเป็น '0' เมื่อรีเซตหรือ power-on หรือ เมื่อ CPU เขียนค่าใส่รีจิสเตอร์ส่งข้อมูลและจะมีค่าเป็น '1' เมื่อรีจิสเตอร์ส่งข้อมูลทำการส่งค่าไปยังรีจิสเตอร์เลื่อนข้อมูลส่งออกแล้ว FE : มีค่าเป็น '0' เมื่อรีเซตหรือ power-on และมีค่าเป็น '1' เมื่อเฟรมข้อมูลที่รับเข้ามามีรูปแบบไม่ถูกต้อง OV: มีค่าเป็น '0' เมื่อรีเซตหรือ power-on และจะมีค่าเป็น '1' เมื่อเกิด overrun error PE: มีค่าเป็น '0' เมื่อรีเซตหรือ power-on และจะมีค่าเป็น '1' เมื่อเฟรมข้อมูลที่รับเข้ามามีรูปแบบของบิทพาริตี้ผิด IRQ: มีค่าเป็น '0' เมื่อรีเซตหรือ power-on และจะมีค่าเป็น '1' เมื่อ UART ทำการอินเทอร์รัป CPU
รีจิสเตอร์ส่งข้อมูล	ส่งข้อมูลที่ถูกเขียนค่ามาให้รีจิสเตอร์เลื่อนข้อมูลส่งออกได้ถูกต้อง และส่งสัญญาณให้รีจิสเตอร์สถานะได้ถูกต้อง
รีจิสเตอร์รับข้อมูล	รับข้อมูลจากรีจิสเตอร์เลื่อนข้อมูลรับเข้าได้ถูกต้อง และส่งสัญญาณให้รีจิสเตอร์สถานะได้ถูกต้อง
รีจิสเตอร์เลื่อนข้อมูลส่งออก	ส่งข้อมูลออกไปได้ถุกต้องตามที่รับมาจากรีจิสเตอร์ส่งข้อมูล และส่งในตามรูปแแบที่กำหนดไว้ในรีจิสเตอร์ควบคุม
รีจิสเตอร์เลื่อนข้อมูลรับเข้า	รับข้อมูลเข้าได้ถูกต้องตามโปรโตคอล และตรวจสอบลักษณะข้อมูลและส่งสัญญาณให้รีจิสเตอร์สถานะได้ถูกต้อง

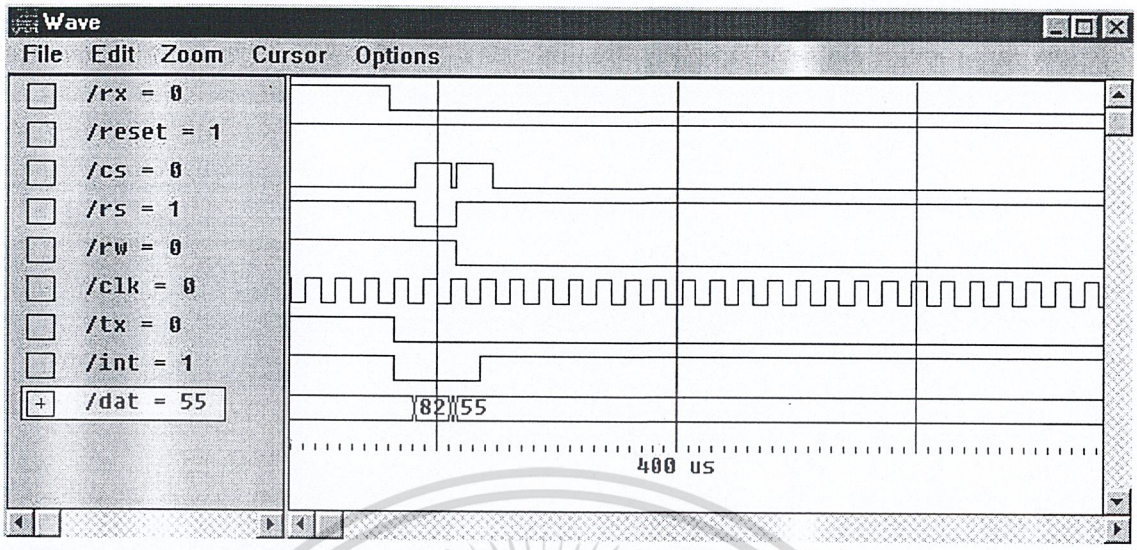
ตาราง 4-1 ผลการทดสอบแต่ละคอมโนเนนต์ในระบบ

การทดสอบโดยการจำลองการทำงานจะตรวจสอบโดยการดูค่าจากรีจิสเตอร์ควบคุมและรีจิสเตอร์สถานะในขณะต่างๆ ของการทำงาน และตรวจสอบข้อมูลที่รับและส่งว่ามีความถูกต้องหรือไม่ และดูความสัมพันธ์ในการทำงานของสัญญาณอินเทอร์รัป ดังแสดงในรูปที่ 4-1, 4-2 และ 4-3



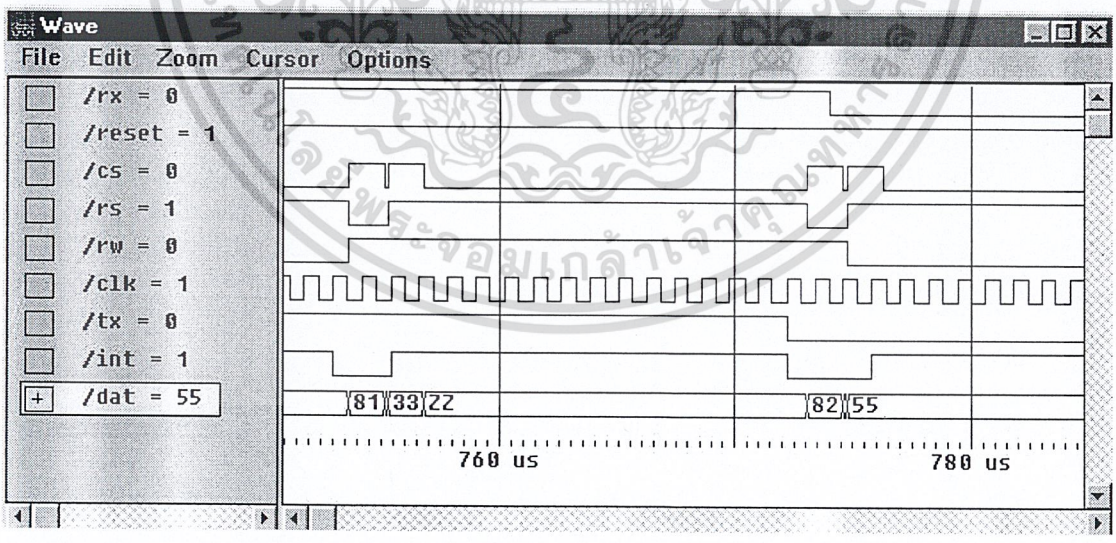
รูปที่ 4-1 การเขียนค่าลงในรีจิสเตอร์ควบคุม

จากรูปที่ 4-1 เป็นการเลือกที่จะเขียนค่า C5H ลงในรีจิสเตอร์ควบคุม โดยให้ rs = '0' และ rw = '0' แล้วให้ rs = '1' และ rw = '0' เพื่อเลือกรีจิสเตอร์ส่งข้อมูลแล้วเขียนค่า 55H



รูปที่ 4-2 การอ่านค่ารีจิสเตอร์สถานะและส่งข้อมูล

เมื่อมีอินเทอร์รัปจาก UART เมื่อมีอินเทอร์รัปเกิดขึ้น (int = '0') ก็จะทำการอ่านรีจิสเตอร์สถานะ เพื่อดูว่าอินเทอร์รัปเกิดจากรีจิสเตอร์ด้านส่งว่าง หรือรีจิสเตอร์ด้านรับเต็ม จากรูปที่ 4-2 เมื่อมีการอ่านค่ามาจากรีจิสเตอร์สถานะได้ 82H ("10000010") แสดงว่าบิท TDRE (บิทที่ 1) มีค่าเท่ากับ 1 แล้วจึงทำการส่งข้อมูล 55H ต่อไป

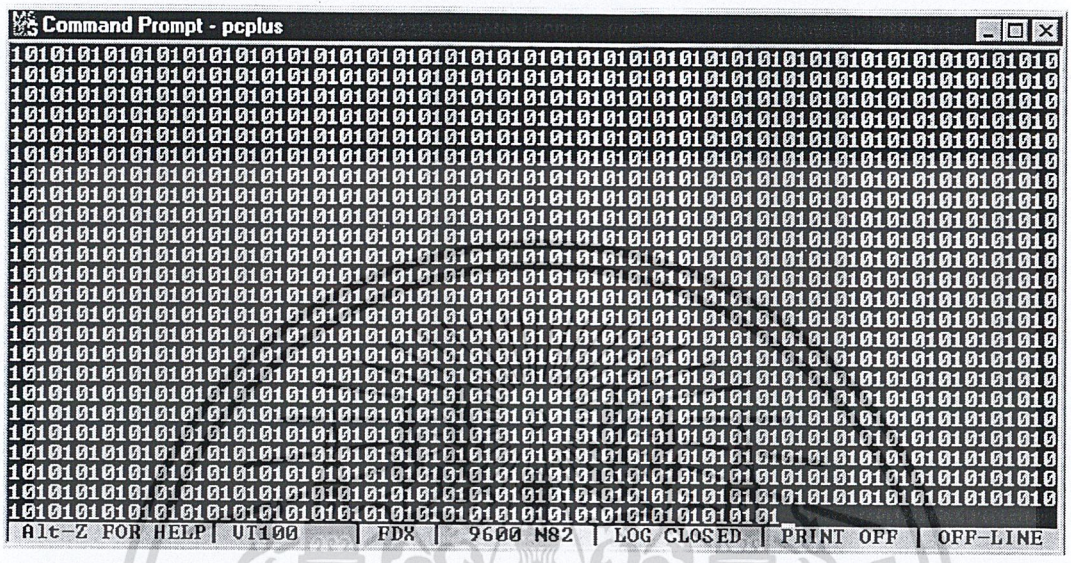


รูปที่ 4-3 การอ่านค่ารีจิสเตอร์สถานะและรับข้อมูล

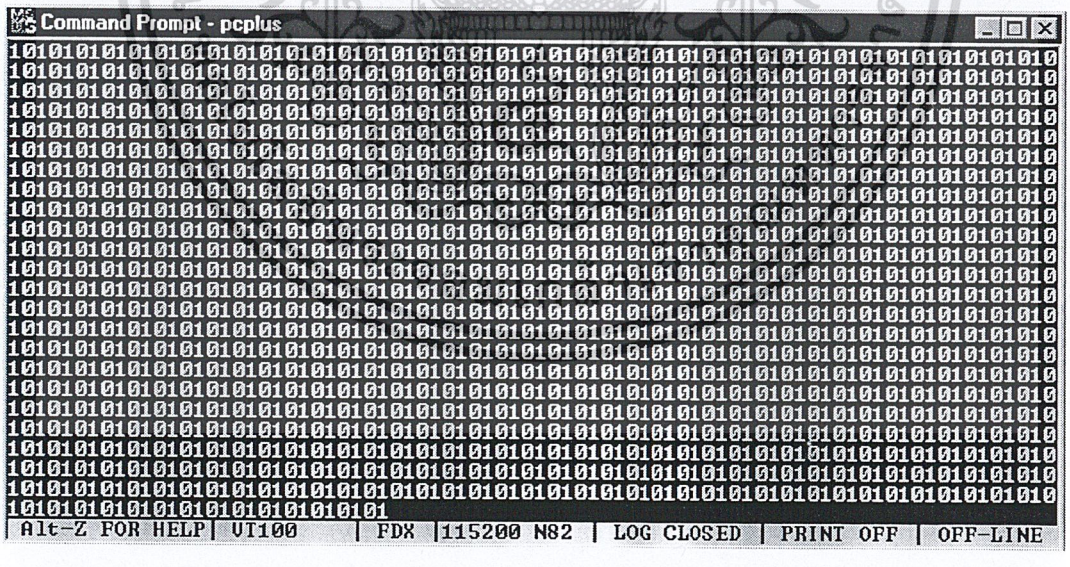
จากรูปที่ 4-3 เมื่อเกิดอินเทอร์รัปและอ่านค่าในรีจิสเตอร์สถานะได้ 81H ("10000001") ก็แสดงว่าบิท RDRF (บิทที่ 0) มีค่าเท่ากับ 1 จึงทำการอ่านข้อมูลจากรีจิสเตอร์รับข้อมูล ได้ค่า 33H

4.2 ผลการทดสอบในระดับฮาร์ดแวร์ (FPGA Prototype)

ในการทดสอบได้เขียนโปรแกรมลงใน FPGA ควบคุมให้ UART ใน FPGA ส่งเลข 0 และ 1 ออกมาโดยให้ทำการส่งข้อมูลกับพอร์ตอนุกรมของคอมพิวเตอร์พีซีผ่านโปรแกรม PCPLUS ที่ความเร็ว 9600 บิทต่อวินาที และ 115200 บิทต่อวินาที ซึ่งผลปรากฏว่าสามารถส่งข้อมูลมาได้ถูกต้องทั้งหมด



รูปที่ 4-4 ผลการส่งข้อมูลอนุกรมจาก UART มายังคอมพิวเตอร์พีซี ที่ความเร็ว 9600 บิทต่อวินาที



รูปที่ 4-5 ผลการส่งข้อมูลอนุกรมจาก UART มายังคอมพิวเตอร์พีซี ที่ความเร็ว 115200 บิทต่อวินาที

ส่วนในด้านการรับข้อมูลอนุกรมจากคอมพิวเตอร์พีซี จะทำโดยการส่งข้อมูลจากคีย์บอร์ดของคอมพิวเตอร์แล้วให้ UART รับแล้วแสดงผลออกมาที่ LED บนบอร์ด FPGA ซึ่งผลปรากฏว่าสามารถรับข้อมูลได้ถูกต้องทั้งหมด

บทที่ 5
สรุปและวิจารณ์

5.1 วิธีการดำเนินงาน

การดำเนินงานโครงงานนี้ในช่วงแรกเป็นการศึกษาการออกแบบระบบดิจิตอลโดยใช้ภาษา VHDL และการสื่อสารอนุกรม UART โดยได้ศึกษาโครงสร้างและการทำงานของชิพที่ใช้เป็นต้นแบบคือ MC6850 ACIA ของบริษัทโมโตโรล่า และการทำงานในส่วน UART ของไมโครคอนโทรลเลอร์ MCS51 ต่อมาเป็นขั้นตอนในการออกแบบ โดยเริ่มจากออกแบบในระดับบล็อกไดอะแกรมและคอมโพเนนต์ของระบบ แล้วจึงทำการเขียนโค้ดของ UART และ Test Bench เมื่อทำการออกแบบเสร็จแล้วขั้นตอนต่อไปจะเป็นการทดสอบระบบที่ได้ออกแบบโดยการจำลองการทำงานโดยใช้ซอฟต์แวร์ Model Sim และทำการสังเคราะห์โดยใช้ซอฟต์แวร์ Leonardo Spectrum และทำการ Place & Route โดยใช้ Xilinx Foundation Series จากนั้นก็มาถึงขั้นตอนสุดท้ายเป็นขั้นตอนการทดสอบในระดับฮาร์ดแวร์โดยใช้ FPGA ของบริษัท Xilinx เบอร์ XC4013EPQ160 แต่เนื่องจากมีเวลาไม่พอจึงสามารถทำการทดสอบได้ที่ความเร็ว 9600 บิทต่อวินาที และ 115200 บิทต่อวินาทีเท่านั้น

โครงงานนี้ถือว่าประสบสำเร็จตามเป้าหมายที่ได้กำหนดไว้คือสามารถสร้างระบบลงบน FPGA และทำการติดต่อสื่อสารกับ UART จริงของคอมพิวเตอร์พีซีได้

5.2 การออกแบบและทดสอบ

จากการศึกษาและออกแบบคอร์ของ UART ด้วยภาษา VHDL มาทำการสังเคราะห์และ Place & Route ลง FPGA ได้ผลการออกแบบดังนี้

ส่วนของโค้ดภาษา VHDL ของ UART ประกอบด้วย

- จำนวนเอนติตี้ 8 เอนติตี้
- จำนวนโพรเซส 4 โพรเซส
- จำนวนบรรทัด 460 บรรทัด

ส่วนของโค้ดภาษา VHDL ของ Test Bench ประกอบด้วย

- จำนวนเอนติตี้ 6 เอนติตี้
- จำนวนโพรเซส 10 โพรเซส
- จำนวนบรรทัด 634 บรรทัด

ส่วนของคอร์ของ UART หลังจากทำการสังเคราะห์และสร้างลง FPGA สรุปผลได้ดังนี้

Xilinx Mapping Report File for Design "uart"

Design Information :

Target Device : x4013e

Target Package : pq160

Target Speed : -3

Mapper Version : xc4000e -- M1.4.12

Mapped Date : Tue Mar 23 10:09:54 1999

Design Summary :

Number of CLBs : 351 out of 576 60%

CLB Flip Flops: 190

4 input LUTs : 590 (128 used as route-throughs)

3 input LUTs : 60

Number of bonded IOBs: 16 out of 129 12%

IOB Flops : 0

IOB Latches : 0

Number of global buffers: 1 out of 8 12%

Number of primary CLKs: 1 out of 4 25%

Number of RPM macros : 4

Total equivalent gate count for design: 5398

Additional JTAG gate count for IOBs: 768

จากผลการทดสอบโดยการจำลองการทำงานนั้น ในแต่ละระดับของคอร์ของ UART ที่ออกแบบสามารถรับ-ส่งข้อมูลได้ถูกต้อง และสำหรับการทดสอบในระดับฮาร์ดแวร์โดยใช้ FPGA นั้น คอร์ของ UART สามารถที่จะรับ-ส่งข้อมูลแบบอนุกรมกับคอมพิวเตอร์พีซีได้ตาม specification ที่ความเร็ว 9600 บิทต่อวินาที และ 115200 บิทต่อวินาที

5.3 ปัญหาและอุปสรรคในการทำงาน

ในขั้นตอนของการออกแบบและทดสอบในระดับซอฟต์แวร์นั้นมีปัญหาในการเขียนโปรแกรมภาษา VHDL ให้สามารถทำการสังเคราะห์ได้ เนื่องจากยังไม่เข้าใจในหลักการทำงานของซอฟต์แวร์ที่ใช้ในการสังเคราะห์ จึงไม่ทราบว่าควรจะเขียนในลักษณะใดทำให้ต้องแก้ไขโปรแกรมหลายรอบจึงจะสามารถสังเคราะห์ผ่านได้ และหลังจากทำการ Place & Route แล้วนำไฟล์ VHDL และไฟล์ดีเลย์ไปทำการจำลองการทำงานก็มีปัญหาในเรื่องไทม์มิ่ง (Timing) ของ Test Bench ไม่เหมาะสมกับดีเลย์ของระบบทำให้ต้องมีการแก้ไขไทม์มิ่งของ Test Bench ใหม่จึงจะสามารถตรวจสอบโดยการจำลองการทำงานในระดับนี้ได้ ส่วนในด้านการทดสอบทางฮาร์ดแวร์โดยใช้ FPGA นั้นในตอนแรกของการทดสอบการรับ-ส่งข้อมูล ปรากฏว่าข้อมูลไม่ถูกต้องทั้งหมด จึงใช้ Logic Analyzer ทำการตรวจสอบสัญญาณต่างๆ ที่เกี่ยวข้องกับระบบและพบว่ามีปัญหาเรื่องสัญญาณนาฬิกาที่จ่ายให้กับ FPGA ซึ่งในการทดสอบใช้สัญญาณนาฬิกาจาก Function Generator เนื่องจากสายนำสัญญาณที่ต่อจาก Function Generator มายังบอร์ด FPGA ไม่ดีทำให้ส่งสัญญาณนาฬิกามาไม่สม่ำเสมอและไม่ต่อเนื่อง ทำให้การรับ-ส่งข้อมูลผิดพลาด จึงได้ทำการเปลี่ยนสายนำสัญญาณใหม่ผลการรับ-ส่งข้อมูลจึงถูกต้อง 100%

5.4 แนวทางในการพัฒนาต่อไป

เนื่องจากคอร์ของ UART ที่ได้ทำการออกแบบในโครงงานนี้ เมื่อลง FPGA แล้วได้ทดสอบการรับ-ส่งข้อมูลที่ความเร็ว 9600 บิทต่อวินาที และ 115200 บิทต่อวินาที ยังไม่ได้ทดสอบที่ความเร็วระดับอื่นๆ ที่สูงกว่านี้ การพัฒนาต่อไปควรจะออกแบบและพัฒนาให้สามารถรับ-ส่งข้อมูลได้ที่ความเร็วสูง (High Speed UART) และในส่วนของ System Interface ควรพัฒนาให้มีความสมบูรณ์และยืดหยุ่นในการนำไปใช้งานได้กว้างขวางมากขึ้น และพัฒนาเทคนิคในการออกแบบให้ได้ชิพที่มีประสิทธิภาพดีขึ้นและมีขนาดเล็กลง

ภาคผนวก ก

MOTOROLA
SEMICONDUCTORS
3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721

MC6850

ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER (ACIA)

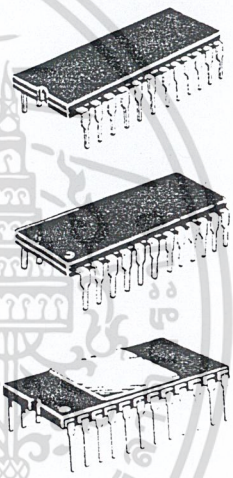
The MC6850 Asynchronous Communications Interface Adapter provides the data formatting and control to interface serial asynchronous data communications information to bus organized systems such as the MC6800 Microprocessing Unit.

The bus interface of the MC6850 includes select, enable, read/write, interrupt and bus interface logic to allow data transfer over an 8-bit bidirectional data bus. The parallel data of the bus system is serially transmitted and received by the asynchronous data interface, with proper formatting and error checking. The functional configuration of the ACIA is programmed via the data bus during system initialization. A programmable Control Register provides variable word lengths, clock division ratios, transmit control, receive control, and interrupt control. For peripheral or modem operation, three control lines are provided. These lines allow the ACIA to interface directly with the MC6860L 0-600 bps digital modem.

- 8- and 9-Bit Transmission
- Optional Even and Odd Parity
- Parity, Overrun and Framing Error Checking
- Programmable Control Register
- Optional ÷1, ÷16, and ÷64 Clock Modes
- Up to 1.0 Mbps Transmission
- False Start Bit Deletion
- Peripheral/Modem Control Functions
- Double Buffered
- One- or Two-Stop Bit Operation

MOS
(N-CHANNEL, SILICON-GATE)

ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER

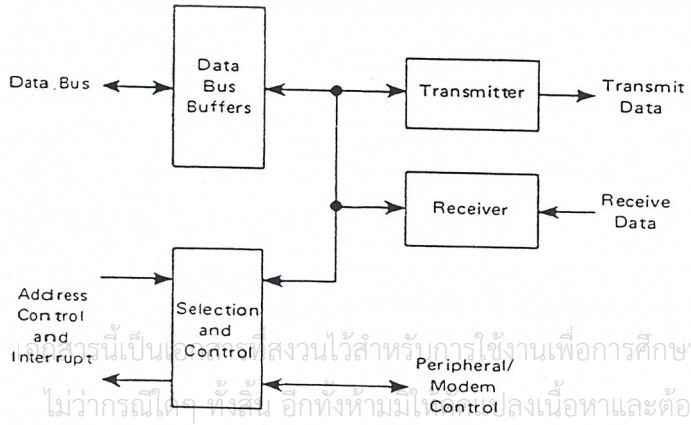


S SUFFIX
CERDIP PACKAGE
CASE 623

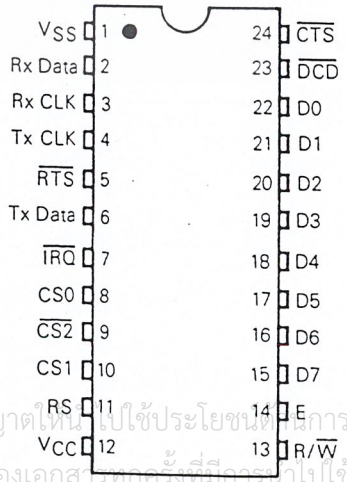
P SUFFIX
PLASTIC PACKAGE
CASE 709

L SUFFIX
CERAMIC PACKAGE
CASE 716

MC6850 ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER
BLOCK DIAGRAM



PIN ASSIGNMENT



DS9493R4

MAXIMUM RATINGS

Characteristics	Symbol	Value	Unit
Supply Voltage	V_{CC}	−0.3 to +7.0	V
Input Voltage	V_{in}	−0.3 to +7.0	V
Operating Temperature Range MC6850, MC68A50, MC68B50 MC6850C, MC68A50C	T_A	T_L to T_H 0 to 70 −40 to +85	°C
Storage Temperature Range	T_{stg}	−55 to +150	°C

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{CC}).

THERMAL CHARACTERISTICS

Characteristic	Symbol	Value	Unit
Thermal Resistance Plastic Ceramic Cerdip	θ_{JA}	 120 60 65	°C/W

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \theta_{JA}) \quad (1)$$

Where:

$T_A \equiv$ Ambient Temperature, °C

$\theta_{JA} \equiv$ Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D \equiv P_{INT} + P_{PORT}$

$P_{INT} \equiv I_{CC} \times V_{CC}$, Watts — Chip Internal Power

$P_{PORT} \equiv$ Port Power Dissipation, Watts — User Determined

For most applications $P_{PORT} \ll P_{INT}$ and can be neglected. P_{PORT} may become significant if the device is configured to drive Darlington bases or sink LED loads.

An approximate relationship between P_D and T_J (if P_{PORT} is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations (1) and (2) for K gives:

$$K = P_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D 2 \quad (3)$$

Where K is a constant pertaining to the particular part. K can be determined from equation (3) by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

DC ELECTRICAL CHARACTERISTICS (V_{CC} = 5.0 Vdc ±5%, V_{SS} = 0, $T_A = T_L$ to T_H unless otherwise noted.)

Characteristic		Symbol	Min	Typ	Max	Unit
Input High Voltage		V_{IH}	V_{SS} + 2.0	–	V_{CC}	V
Input Low Voltage		V_{IL}	V_{SS} − 0.3	–	V_{SS} + 0.8	V
Input Leakage Current (V_{in} = 0 to 5.25 V)	R/$\overline{W}$, CS0, CS1, $\overline{CS2}$, Enable RS, Rx D, Rx C, $\overline{CTS}$, $\overline{DCD}$	I_{in}	–	1.0	2.5	μA
Hi-Z (Off State) Input Current (V_{in} = 0.4 to 2.4 V)	D0-D7	I_{TSI}	–	2.0	10	μA
Output High Voltage (I_{Load} = − 205 μA, Enable Pulse Width < 25 μs) (I_{Load} = − 100 μA, Enable Pulse Width < 25 μs)	D0-D7 Tx Data, $\overline{RTS}$	V_{OH}	V_{SS} + 2.4 V_{SS} + 2.4	– –	– –	V
Output Low Voltage (I_{Load} = 1.6 mA, Enable Pulse Width < 25 μs)		V_{OL}	–	–	V_{SS} + 0.4	V
Output Leakage Current (Off State) (V_{OH} = 2.4 V)	$\overline{IRQ}$	I_{LOH}	–	1.0	10	μA
Internal Power Dissipation (Measured at T_A = 0°C)		P_{INT}	–	300	525*	mW
Internal Input Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz) D0-D7 E, Tx CLK, Rx CLK, R/$\overline{W}$, RS, Rx Data, CS0, CS1, $\overline{CS2}$, $\overline{CTS}$, $\overline{DCD}$		C_{in}	– –	10 7.0	12.5 7.5	pF
Output Capacitance (V_{in} = 0, T_A = 25°C, f = 1.0 MHz) RTS, Tx Data $\overline{IRQ}$		C_{out}	– –	– –	10 5.0	pF

* For temperatures less than T_A = 0°C, P_{INT} maximum will increase.

SERIAL DATA TIMING CHARACTERISTICS

Characteristic		Symbol	MC6850		MC68A50		MC68B50		Unit
			Min	Max	Min	Max	Min	Max	
Data Clock Pulse Width, Low (See Figure 1)	÷ 16, ÷ 64 Modes	PW_{CL}	600	–	450	–	280	–	ns
	÷ 1 Mode		900	–	650	–	500	–	
Data Clock Pulse Width, High (See Figure 2)	÷ 16, ÷ 64 Modes	PW_{CH}	600	–	450	–	280	–	ns
	÷ 1 Mode		900	–	650	–	500	–	
Data Clock Frequency	÷ 16, ÷ 64 Modes	f_C	–	0.8	–	1.0	–	1.5	MHz
	÷ 1 Mode		–	500	–	750	–	1000	kHz
Data Clock-to-Data Delay for Transmitter (See Figure 3)		t_{TDD}	–	600	–	540	–	460	ns
Receive Data Setup Time (See Figure 4)	÷ 1 Mode	t_{RDS}	250	–	100	–	30	–	ns
Receive Data Hold Time (See Figure 5)	÷ 1 Mode	t_{RDH}	250	–	100	–	30	–	ns
Interrupt Request Release Time (See Figure 6)		t_{IR}	–	1.2	–	0.9	–	0.7	μs
Request-to-Send Delay Time (See Figure 6)		t_{RTS}	–	560	–	480	–	400	ns
Input Rise and Fall Times (or 10% of the pulse width if smaller)		t_r, t_f	–	1.0	–	0.5	–	0.25	μs

FIGURE 1 — CLOCK PULSE WIDTH, LOW-STATE

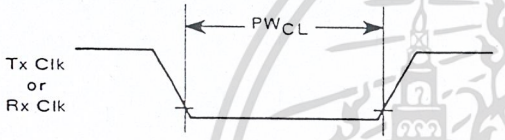


FIGURE 2 — CLOCK PULSE WIDTH, HIGH-STATE

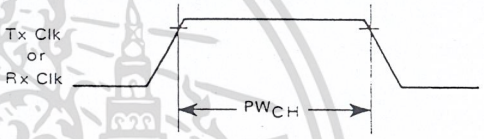


FIGURE 3 — TRANSMIT DATA OUTPUT DELAY

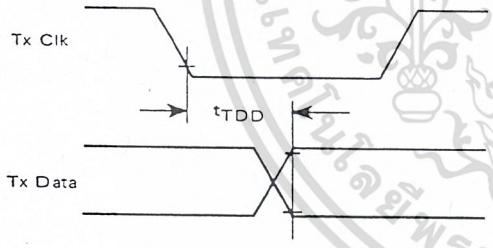


FIGURE 4 — RECEIVE DATA SETUP TIME (÷ 1 Mode)

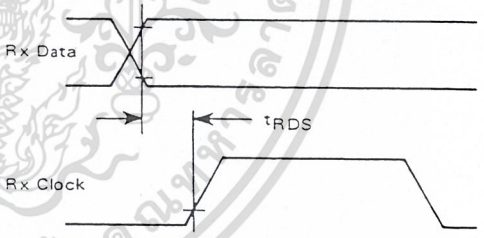


FIGURE 5 — RECEIVE DATA HOLD TIME (÷ 1 Mode)

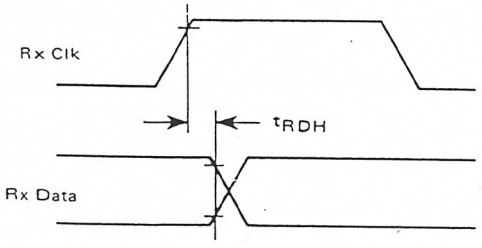
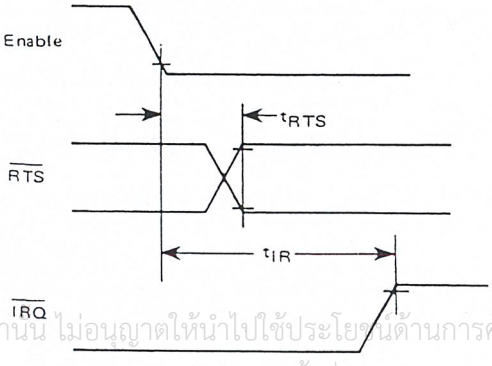


FIGURE 6 — REQUEST-TO-SEND DELAY AND INTERRUPT-REQUEST RELEASE TIMES



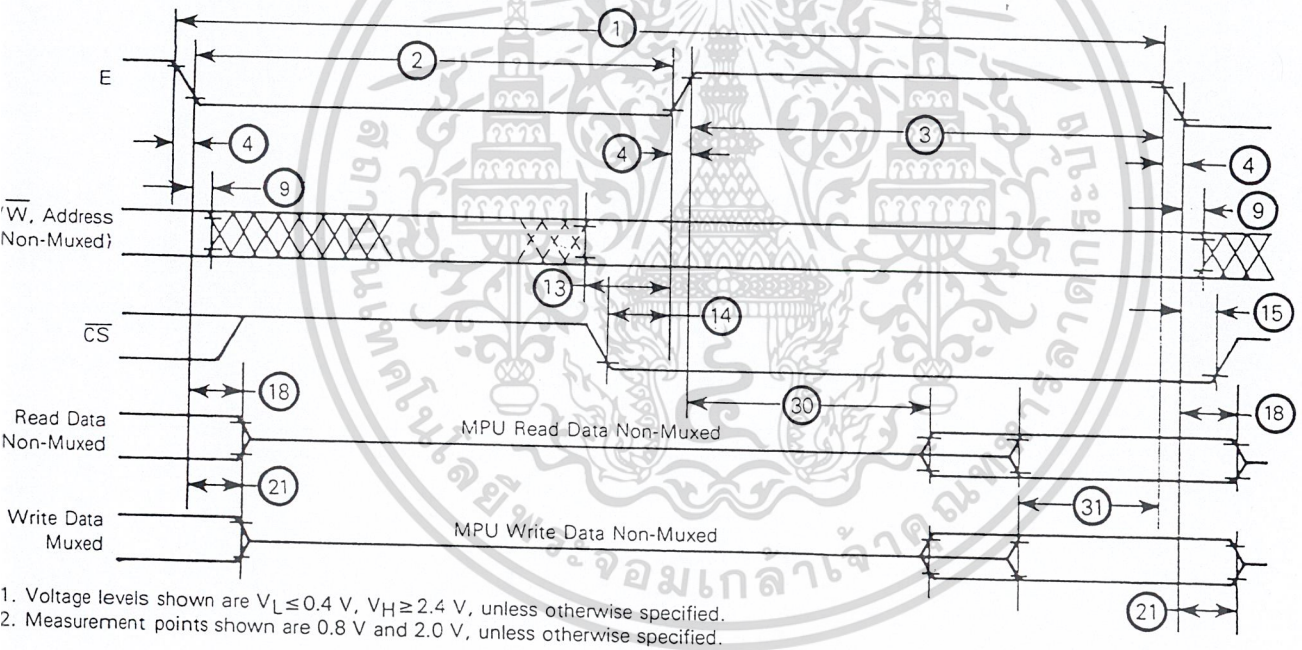
Note: Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.

BUS TIMING CHARACTERISTICS (See Notes 1 and 2 and Figure 7)

Ident. Number	Characteristic	Symbol	MC6850		MC68A50		MC68B50		Unit
			Min	Max	Min	Max	Min	Max	
1	Cycle Time	t_{cyc}	1.0	10	0.67	10	0.5	10	μs
2	Pulse Width, E Low	PW_{EL}	430	9500	280	9500	210	9500	ns
3	Pulse Width, E High	PW_{EH}	450	9500	280	9500	220	9500	ns
4	Clock Rise and Fall Time	t_r, t_f	–	25	–	25	–	20	ns
9	Address Hold Time	t_{AH}	10	–	10	–	10	–	ns
13	Address Setup Time Before E	t_{AS}	80	–	60	–	40	–	ns
14	Chip Select Setup Time Before E	t_{CS}	80	–	60	–	40	–	ns
15	Chip Select Hold Time	t_{CH}	10	–	10	–	10	–	ns
18	Read Data Hold Time	t_{DHR}	20	50*	20	50*	20	50*	ns
21	Write Data Hold Time	t_{DHW}	10	–	10	–	10	–	ns
30	Output Data Delay Time	t_{DDR}	–	290	–	180	–	150	ns
31	Input Data Setup Time	t_{DSW}	165	–	80	–	60	–	ns

The data bus output buffers are no longer sourcing or sinking current by t_{DHR}max (High Impedance).

FIGURE 7 — BUS TIMING CHARACTERISTICS



1. Voltage levels shown are $V_L \leq 0.4$ V, $V_H \geq 2.4$ V, unless otherwise specified.
2. Measurement points shown are 0.8 V and 2.0 V, unless otherwise specified.

FIGURE 8 — BUS TIMING TEST LOADS

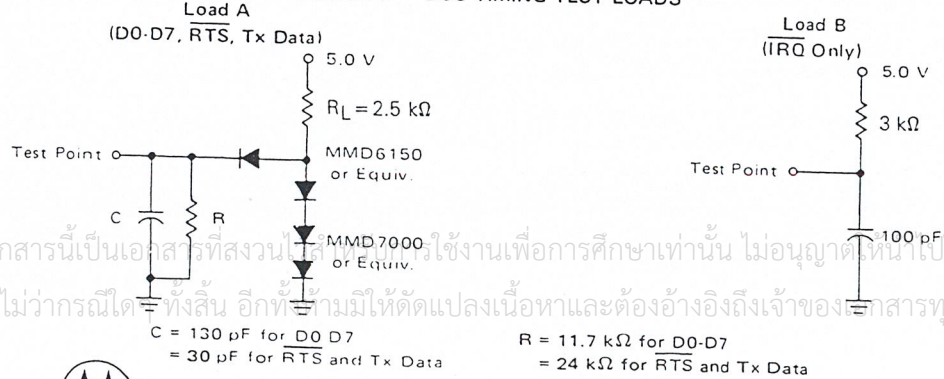
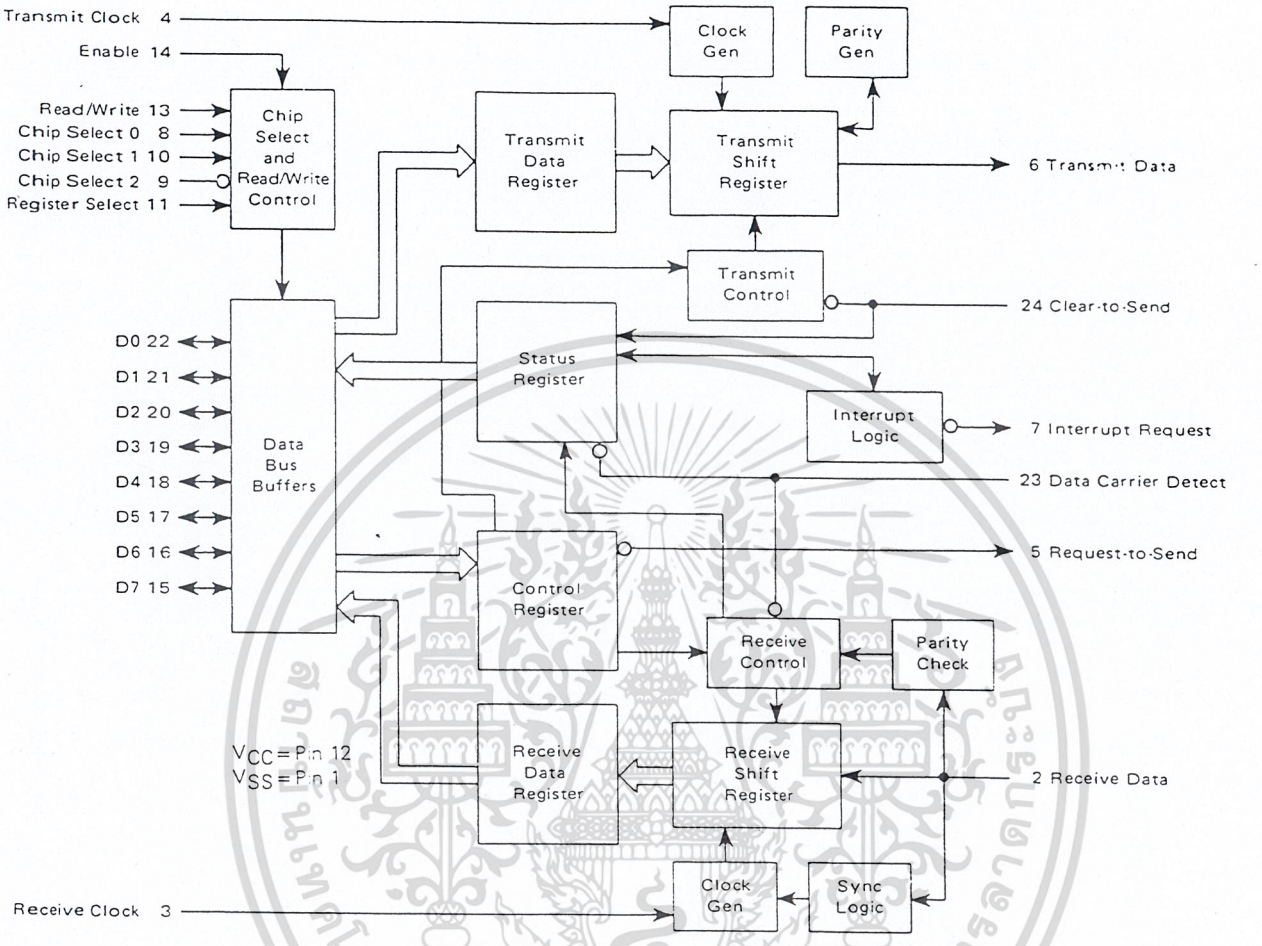


FIGURE 9 — EXPANDED BLOCK DIAGRAM



DEVICE OPERATION

At the bus interface, the ACIA appears as two addressable memory locations. Internally, there are four registers: two read-only and two write-only registers. The read-only registers are Status and Receive Data; the write-only registers are Control and Transmit Data. The serial interface consists of serial input and output lines with independent clocks, and three peripheral/modem control lines.

MASTER RESET

The master reset (CR0, CR1) must be set immediately after power-up to insure the reset condition and prepare for programming the ACIA functional configuration when the communications channel is required. During the first master reset, the $\overline{IRQ}$ and $\overline{RTS}$ outputs are held at level 1. On all other master resets, the $\overline{RTS}$ output can be programmed high or low with the $\overline{IRQ}$ output held high. Control bits CR5 and CR6 should also be programmed to define the state of $\overline{RTS}$ whenever master reset is utilized. After master resetting the ACIA, the programmable Control Register can be set for a number of options such as variable clock divider ratios, variable word length, one or two stop bits, and parity (even, odd, or none).

TRANSMIT

A typical transmitting sequence consists of reading the ACIA Status Register either as a result of an interrupt or in the ACIA's turn in a polling sequence. A character may be written into the Transmit Data Register if the status read operation has indicated that the Transmit Data Register is empty. This character is transferred to a Shift Register where it is serialized and transmitted from the Transmit Data output preceded by a start bit and followed by one or two stop bits. Internal parity (odd or even) can be optionally added to the character and will occur between the last data bit and the first stop bit. After the first character is written in the Data Register, the Status Register can be read again to check for a Transmit Data Register Empty condition and current peripheral status. If the register is empty, another character can be loaded for transmission even through the first character is in the process of being transmitted (because of

double buffering). The second character will be automatically transferred into the Shift Register when the first character transmission is completed. This sequence continues until all the characters have been transmitted.

RECEIVE

Data is received from a peripheral by means of the Receive Data input. A divide-by-one clock ratio is provided for an externally synchronized clock (to its data) while the divide-by-16 and 64 ratios are provided for internal synchronization. Bit synchronization in the divide-by-16 and 64 modes is initiated by the detection of 8 or 32 low samples on the receive line in the divide-by-16 and 64 modes respectively. False start bit deletion capability insures that a full half bit of a start bit has been received before the internal clock is synchronized to the bit time. As a character is being received, parity (odd or even) will be checked and the error indication will be available in the Status Register along with framing error, overrun error, and Receive Data Register full. In a typical receiving sequence, the Status Register is read to determine if a character has been received from a peripheral. If the Receiver Data Register is full, the character is placed on the 8-bit ACIA bus when a Read Data command is received from the MPU. When parity has been selected for a 7-bit word (7 bits plus parity), the receiver strips the parity bit (D7 = 0) so that data alone is transferred to the MPU. This feature reduces MPU programming. The Status Register can continue to be read to determine when another character is available in the Receive Data Register. The receiver is also double buffered so that a character can be read from the data register as another character is being received in the shift register. The above sequence continues until all characters have been received.

INPUT/OUTPUT FUNCTIONS

ACIA INTERFACE SIGNALS FOR MPU

The ACIA interfaces to the M6800 MPU with an 8-bit bidirectional data bus, three chip select lines, a register select line, an interrupt request line, read/write line, and enable line. These signals permit the MPU to have complete control over the ACIA.

ACIA Bidirectional Data (D0-D7) — The bidirectional data lines (D0-D7) allow for data transfer between the ACIA and the MPU. The data bus output drivers are three-state devices that remain in the high-impedance (off) state except when the MPU performs an ACIA read operation.

ACIA Enable (E) — The Enable signal, E, is a high-impedance TTL-compatible input that enables the bus input/output data buffers and clocks data to and from the ACIA. This signal will normally be a derivative of the MC6800 $\phi 2$ Clock or MC6809 E clock.

Read/Write (R/$\overline{W}$) — The Read/Write line is a high-impedance input that is TTL compatible and is used to control the direction of data flow through the ACIA's input/output data bus interface. When Read/Write is high (MPU Read cycle), ACIA output drivers are turned on and a selected register is read. When it is low, the ACIA output drivers are turned off and the MPU writes into a selected register. Therefore, the Read/Write signal is used to select read-only or write-only registers within the ACIA.

Chip Select (CS0, CS1, $\overline{CS2}$) — These three high-impedance TTL-compatible input lines are used to address the ACIA. The ACIA is selected when CS0 and CS1 are high and $\overline{CS2}$ is low. Transfers of data to and from the ACIA are then performed under the control of the Enable Signal, Read/Write, and Register Select.

Register Select (RS) — The Register Select line is a high-impedance input that is TTL compatible. A high level is used to select the Transmit/Receive Data Registers and a low level the Control/Status Registers. The Read/Write signal line is used in conjunction with Register Select to select the read-only or write-only register in each register pair.

Interrupt Request ($\overline{IRQ}$) — Interrupt Request is a TTL-compatible, open-drain (no internal pullup), active low output that is used to interrupt the MPU. The $\overline{IRQ}$ output remains low as long as the cause of the interrupt is present and the appropriate interrupt enable within the ACIA is set. The $\overline{IRQ}$ status bit, when high, indicates the $\overline{IRQ}$ output is in the active state.

Interrupts result from conditions in both the transmitter and receiver sections of the ACIA. The transmitter section causes an interrupt when the Transmitter Interrupt Enabled condition is selected (CR5•$\overline{CR6}$), and the Transmit Data Register Empty (TDRE) status bit is high. The TDRE status bit indicates the current status of the Transmitter Data Register except when inhibited by Clear-to-Send ($\overline{CTS}$) being high or the ACIA being maintained in the Reset condition. The interrupt is cleared by writing data into the Transmit Data Register. The interrupt is masked by disabling the Transmitter Interrupt via CR5 or CR6 or by the loss of $\overline{CTS}$ which inhibits the TDRE status bit. The Receiver section causes an interrupt when the Receiver Interrupt Enable is set and the Receive Data Register Full (RDRF) status bit is high, an Overrun has occurred, or Data Carrier Detect ($\overline{DCD}$) has gone high. An interrupt resulting from the RDRF status bit can be cleared by reading data or resetting the ACIA. Interrupts caused by Overrun or loss of $\overline{DCD}$ are cleared by reading the status register after the error condition has occurred and then reading the Receive Data Register or resetting the ACIA. The receiver interrupt is masked by resetting the Receiver Interrupt Enable.

CLOCK INPUTS

Separate high-impedance TTL-compatible inputs are provided for clocking of transmitted and received data. Clock frequencies of 1, 16, or 64 times the data rate may be selected.

Transmit Clock (Tx CLK) — The Transmit Clock input is used for the clocking of transmitted data. The transmitter initiates data on the negative transition of the clock.

Receive Clock (Rx CLK) — The Receive Clock input is used for synchronization of received data. (In the ÷1 mode, the clock and data must be synchronized externally.) The receiver samples the data on the positive transition of the clock.

SERIAL INPUT/OUTPUT LINES

Receive Data (Rx Data) — The Receive Data line is a high-impedance TTL-compatible input through which data is received in a serial format. Synchronization with a clock for detection of data is accomplished internally when clock rates of 16 or 64 times the bit rate are used.

Transmit Data (Tx Data) — The Transmit Data output line transfers serial data to a modem or other peripheral.

PERIPHERAL/MODEM CONTROL

The ACIA includes several functions that permit limited control of a peripheral or modem. The functions included are Clear-to-Send, Request-to-Send and Data Carrier Detect.

Clear-to-Send ($\overline{CTS}$) — This high-impedance TTL-compatible input provides automatic control of the transmitting end of a communications link via the modem Clear-to-Send active low output by inhibiting the Transmit Data Register Empty (TDRE) status bit.

Request-to-Send ($\overline{RTS}$) — The Request-to-Send output enables the MPU to control a peripheral or modem via the data bus. The RTS output corresponds to the state of the Control Register bits CR5 and CR6. When CR6=0 or both CR5 and CR6=1, the $\overline{RTS}$ output is low (the active state). This output can also be used for Data Terminal Ready (DTR).

Data Carrier Detect ($\overline{DCD}$) — This high-impedance TTL-compatible input provides automatic control, such as in the receiving end of a communications link by means of a modem Data Carrier Detect output. The $\overline{DCD}$ input inhibits and initializes the receiver section of the ACIA when high. A low-to-high transition of the Data Carrier Detect initiates an interrupt to the MPU to indicate the occurrence of a loss of carrier when the Receive Interrupt Enable bit is set. The Rx CLK must be running for proper $\overline{DCD}$ operation.

ACIA REGISTERS

The expanded block diagram for the ACIA indicates the internal registers on the chip that are used for the status, control, receiving, and transmitting of data. The content of each of the registers is summarized in Table 1.

TRANSMIT DATA REGISTER (TDR)

Data is written in the Transmit Data Register during the negative transition of the enable (E) when the ACIA has been addressed with RS high and R/$\overline{W}$ low. Writing data into the register causes the Transmit Data Register Empty bit in the Status Register to go low. Data can then be transmitted. If the transmitter is idling and no character is being transmitted, then the transfer will take place within 1-bit time of the trailing edge of the Write command. If a character is being transmitted, the new data character will commence as soon as the previous character is complete. The transfer of data causes the Transmit Data Register Empty (TDRE) bit to indicate empty.

RECEIVE DATA REGISTER (RDR)

Data is automatically transferred to the empty Receive Data Register (RDR) from the receiver deserializer (a shift register) upon receiving a complete character. This event causes the Receive Data Register Full bit (RDRF) in the status buffer to go high (full). Data may then be read through the bus by addressing the ACIA and selecting the Receive Data Register with RS and R/$\overline{W}$ high when the ACIA is enabled. The non-destructive read cycle causes the RDRF bit to be cleared to empty although the data is retained in the RDR. The status is maintained by RDRF as to whether or not the data is current. When the Receive Data Register is full, the automatic transfer of data from the Receiver Shift Register to the Data Register is inhibited and the RDR contents remain valid with its current status stored in the Status Register.

TABLE 1 — DEFINITION OF ACIA REGISTER CONTENTS

Data Bus Line Number	Buffer Address			
	RS • R/$\overline{W}$ Transmit Data Register	RS • R/$\overline{W}$ Receive Data Register	$\overline{RS}$ • $\overline{R/W}$ Control Register	$\overline{RS}$ • R/$\overline{W}$ Status Register
	(Write Only)	(Read Only)	(Write Only)	(Read Only)
0	Data Bit 0*	Data Bit 0	Counter Divide Select 1 (CR0)	Receive Data Register Full (RDRF)
1	Data Bit 1	Data Bit 1	Counter Divide Select 2 (CR1)	Transmit Data Register Empty (TDRE)
2	Data Bit 2	Data Bit 2	Word Select 1 (CR2)	Data Carrier Detect ($\overline{DCD}$)
3	Data Bit 3	Data Bit 3	Word Select 2 (CR3)	Clear-to-Send ($\overline{CTS}$)
4	Data Bit 4	Data Bit 4	Word Select 3 (CR4)	Framing Error (FE)
5	Data Bit 5	Data Bit 5	Transmit Control 1 (CR5)	Receiver Overrun (OVRN)
6	Data Bit 6	Data Bit 6	Transmit Control 2 (CR6)	Parity Error (PE)
7	Data Bit 7***	Data Bit 7**	Receive Interrupt Enable (CR7)	Interrupt Request ($\overline{IRQ}$)

* Leading bit = LSB = Bit 0
** Data bit will be zero in 7-bit plus parity modes.
*** Data bit is "don't care" in 7-bit plus parity modes.

CONTROL REGISTER

The ACIA Control Register consists of eight bits of write-only buffer that are selected when RS and R/$\overline{W}$ are low. This register controls the function of the receiver, transmitter, interrupt enables, and the Request-to-Send peripheral/modem control output.

Counter Divide Select Bits (CR0 and CR1) — The Counter Divide Select Bits (CR0 and CR1) determine the divide ratios utilized in both the transmitter and receiver sections of the ACIA. Additionally, these bits are used to provide a master reset for the ACIA which clears the Status Register (except for external conditions on $\overline{CTS}$ and $\overline{DCD}$) and initializes both the receiver and transmitter. Master reset does not affect other Control Register bits. Note that after power-on or a power fail/restart, these bits must be set high to reset the ACIA. After resetting, the clock divide ratio may be selected. These counter select bits provide for the following clock divide ratios:

CR1	CR0	Function
0	0	÷ 1
0	1	÷ 16
1	0	÷ 64
1	1	Master Reset

Word Select Bits (CR2, CR3, and CR4) — The Word Select bits are used to select word length, parity, and the number of stop bits. The encoding format is as follows:

CR4	CR3	CR2	Function
0	0	0	7 Bits + Even Parity + 2 Stop Bits
0	0	1	7 Bits + Odd Parity + 2 Stop Bits
0	1	0	7 Bits + Even Parity + 1 Stop Bit
0	1	1	7 Bits + Odd Parity + 1 Stop Bit
1	0	0	8 Bits + 2 Stop Bits
1	0	1	8 Bits + 1 Stop Bit
1	1	0	8 Bits + Even parity + 1 Stop Bit
1	1	1	8 Bits + Odd Parity + 1 Stop Bit

Word length, Parity Select, and Stop Bit changes are not buffered and therefore become effective immediately.

Transmitter Control Bits (CR5 and CR6) — Two Transmitter Control bits provide for the control of the interrupt from the Transmit Data Register Empty condition, the Request-to-Send ($\overline{RTS}$) output, and the transmission of a Break level (space). The following encoding format is used:

CR6	CR5	Function
0	0	$\overline{RTS}$ = low, Transmitting Interrupt Disabled.
0	1	$\overline{RTS}$ = low, Transmitting Interrupt Enabled.
1	0	$\overline{RTS}$ = high, Transmitting Interrupt Disabled.
1	1	$\overline{RTS}$ = low, Transmits a Break level on the Transmit Data Output. Transmitting Interrupt Disabled.

Receive Interrupt Enable Bit (CR7) — The following interrupts will be enabled by a high level in bit position 7 of the Control Register (CR7): Receive Data Register Full, Overrun, or a low-to-high transition on the Data Carrier Detect ($\overline{DCD}$) signal line.

STATUS REGISTER

Information on the status of the ACIA is available to the MPU by reading the ACIA Status Register. This read-only register is selected when RS is low and R/$\overline{W}$ is high. Information stored in this register indicates the status of the Transmit Data Register, the Receive Data Register and error logic, and the peripheral/modem status inputs of the ACIA.

Receive Data Register Full (RDRF), Bit 0 — Receive Data Register Full indicates that received data has been transferred to the Receive Data Register. RDRF is cleared after an MPU read of the Receive Data Register or by a master reset. The cleared or empty state indicates that the contents of the Receive Data Register are not current. Data Carrier Detect being high also causes RDRF to indicate empty.

Transmit Data Register Empty (TDRE), Bit 1 — The Transmit Data Register Empty bit being set high indicates that the Transmit Data Register contents have been transferred and that new data may be entered. The low state indicates that the register is full and that transmission of a new character has not begun since the last write data command.

Data Carrier Detect ($\overline{DCD}$), Bit 2 — The Data Carrier Detect bit will be high when the $\overline{DCD}$ input from a modem has gone high to indicate that a carrier is not present. This bit going high causes an Interrupt Request to be generated when the Receive Interrupt Enable is set. It remains high after the $\overline{DCD}$ input is returned low until cleared by first reading the Status Register and then the Data Register or until a master reset occurs. If the $\overline{DCD}$ input remains high after read status and read data or master reset has occurred, the interrupt is cleared, the $\overline{DCD}$ status bit remains high and will follow the $\overline{DCD}$ input.

Clear-to-Send ($\overline{CTS}$), Bit 3 — The Clear-to-Send bit indicates the state of the Clear-to-Send input from a modem. A low $\overline{CTS}$ indicates that there is a Clear-to-Send from the modem. In the high state, the Transmit Data Register Empty bit is inhibited and the Clear-to-Send status bit will be high. Master reset does not affect the Clear-to-Send status bit.

Framing Error (FE), Bit 4 — Framing error indicates that the received character is improperly framed by a start and a stop bit and is detected by the absence of the first stop bit. This error indicates a synchronization error, faulty transmission, or a break condition. The framing error flag is set or reset during the receive data transfer time. Therefore, this error indicator is present throughout the time that the associated character is available.

Receiver Overrun (OVRN), Bit 5 — Overrun is an error flag that indicates that one or more characters in the data stream were lost. That is, a character or a number of characters were received but not read from the Receive Data Register (RDR) prior to subsequent characters being received. The overrun condition begins at the midpoint of the last bit of the second character received in succession without a read of the RDR having occurred. The Overrun does not occur in the Status Register until the valid character prior to Overrun has

been read. The RDRF bit remains set until the Overrun is reset. Character synchronization is maintained during the Overrun condition. The Overrun indication is reset after the reading of data from the Receive Data Register or by a Master Reset.

Parity Error (PE), Bit 6 — The parity error flag indicates that the number of highs (ones) in the character does not agree with the preselected odd or even parity. Odd parity is defined to be when the total number of ones is odd. The parity error indication will be present as long as the data character is in the RDR. If no parity is selected, then both the transmitter parity generator output and the receiver partiy check results are inhibited.

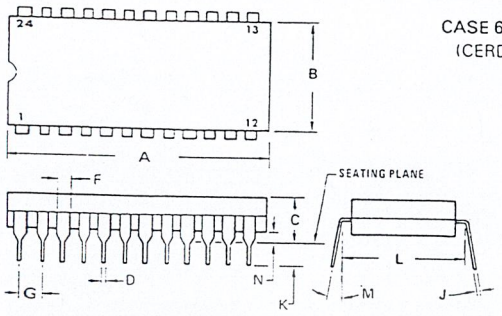
Interrupt Request ($\overline{IRQ}$), Bit 7 — The $\overline{IRQ}$ bit indicates the state of the $\overline{IRQ}$ output. Any interrupt condition with its applicable enable will be indicated in this status bit. Anytime the $\overline{IRQ}$ output is low the $\overline{IRQ}$ bit will be high to indicate the interrupt or service request status. $\overline{IRQ}$ is cleared by a read operation to the Receive Data Register or a write operation to the Transmit Data Register.

ORDERING INFORMATION

Package Type	Frequency (MHz)	Temperature	Order Number
Ceramic L Suffix	1.0	0°C to 70°C	MC6850L
	1.0	−40°C to 85°C	MC6850CL
	1.5	0°C to 70°C	MC68A50L
	1.5	−40°C to 85°C	MC68A50CL
	2.0	0°C to 70°C	MC68B50C
Cerdip S Suffix	1.0	0°C to 70°C	MC6850S
	1.0	−40°C to 85°C	MC6850CS
	1.5	0°C to 70°C	MC68A50S
	1.5	−40°C to 85°C	MC68A50CS
	2.0	0°C to 70°C	MC68B50S
Plastic P Suffix	1.0	0°C to 70°C	MC6850P
	1.0	−40°C to 85°C	MC6850CP
	1.5	0°C to 70°C	MC68A50P
	1.5	−40°C to 85°C	MC68A50CP
	2.0	0°C to 70°C	MC68B50P

Literature Distribution Centers:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.
ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

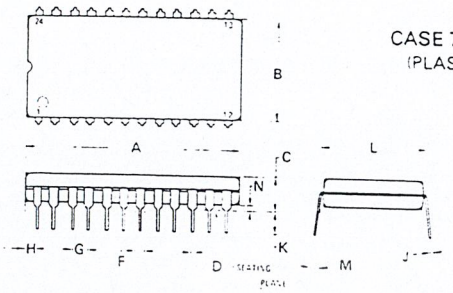
PACKAGE DIMENSIONS



CASE 623-03
(CERDIP)

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	31.24	32.77	1.230	1.290
B	12.70	15.49	0.500	0.610
C	4.06	5.59	0.160	0.220
D	0.41	0.51	0.016	0.020
F	1.27	1.52	0.050	0.060
G	2.54 BSC		0.100 BSC	
J	0.20	0.30	0.008	0.012
K	2.29	4.06	0.090	0.160
L	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.27	0.020	0.050

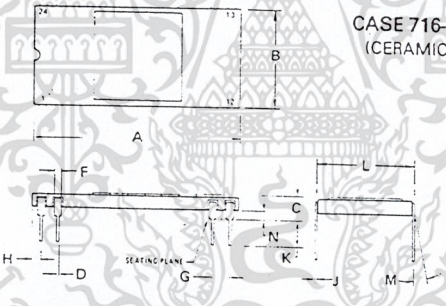
NOTES:
1. DIM "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.
2. LEADS WITHIN 0.13 mm (0.005) RADIUS OF TRUE POSITION AT SEATING PLANE AT MAXIMUM MATERIAL CONDITION. (WHEN FORMED PARALLEL)



CASE 709-02
(PLASTIC)

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	31.37	32.13	1.235	1.265
B	13.72	14.22	0.540	0.560
C	3.94	5.08	0.155	0.200
D	0.36	0.56	0.014	0.022
F	1.02	1.52	0.040	0.060
G	2.54 BSC		0.100 BSC	
H	1.65	2.03	0.065	0.080
J	0.20	0.38	0.008	0.015
K	2.92	3.43	0.115	0.135
L	15.24 BSC		0.600 BSC	
M	0°	15°	0°	15°
N	0.51	1.02	0.020	0.040

NOTES:
1. POSITIONAL TOLERANCE OF LEADS (D), SHALL BE WITHIN 0.25 mm (0.010) AT MAXIMUM MATERIAL CONDITION, IN RELATION TO SEATING PLANE AND EACH OTHER.
2. DIMENSION L TO CENTER OF LEADS WHEN FORMED PARALLEL.
3. DIMENSION B DOES NOT INCLUDE MOLD FLASH.



CASE 716-06
(CERAMIC)

DIM	MILLIMETERS MIN	MILLIMETERS MAX	INCHES MIN	INCHES MAX
A	27.64	30.99	1.088	1.220
B	14.73	15.34	0.580	0.604
C	2.67	4.32	0.105	0.170
D	0.38	0.53	0.015	0.021
F	0.76	1.40	0.030	0.055
G	2.54 BSC		0.100 BSC	
H	0.76	1.78	0.030	0.070
J	0.20	0.30	0.008	0.012
K	2.54	4.57	0.100	0.180
L	14.99	15.49	0.590	0.610
M	–	10°	–	10°
N	1.02	1.52	0.040	0.060

NOTE:
1. LEADS TRUE POSITIONED WITHIN 0.25mm (0.010) DIA (AT SEATING PLANE) AT MAXIMUM MATERIAL CONDITION.
2. DIM "L" TO CENTER OF LEADS WHEN FORMED PARALLEL.

ภาคผนวก ข

XC4010E/XL Pad Name	PC 84	PQ 100††	TQ 144††	PQ 160	TQ 176††	PG 191†	PQ/HQ 208	BG 225†	BG 256††	Bndry Scan
GND	P1	P91	P127	P141	P154	GND*	P182	GND*	GND*	-

6/19/97

* Pads labelled GND* or VCC* are internally bonded to Ground or VCC planes within the package. They have no direct connection to any specific package pin.
† = E only
†† = XL only

Additional XC4010E/XL Package Pins

PQ/HQ208

Not Connected Pins						
P1	P3	P51	P52	P53	P54	P102
P104	P105	P107	P155	P156	P157	P158
P206	P207	P208	-	-	-	-

5/27/97

PG191

VCC Pins						
D3	D10	D16	J4	J15	R4	R10
R15	-	-	-	-	-	-
GND Pins						
C7	C12	D4	D9	D15	G3	G16
K4	K15	M3	M16	R3	R9	R16
T7	T12	-	-	-	-	-

5/27/97

BG225

VCC Pins						
B2	B14	D8	H1	H15	R1	R8
R15	-	-	-	-	-	-
GND Pins						
A1	A8	D12	F8	G7	G8	G9
H2	H6	H7	H8	H9	H10	J7
J8	J9	K8	M8	-	-	-
Not Connected Pins						
A3	B10	C4	C6	C10	D11	E2
E3	E14	E15	F1	F2	F7	F9
F12	G10	J5	K1	K4	K12	L2
L6	L15	M10	M14	N7	N11	N15
P5	P7	P10	R10	-	-	-

6/16/97

BG256

VCC Pins						
C14	D6	D7	D11	D14	D15	E20
F1	F4	F17	G4	G17	K4	L17
P4	P17	P19	R2	R4	R17	U6
U7	U10	U14	U15	V7	W20	-
GND Pins						
A1	B7	D4	D8	D13	D17	G20
H4	H17	N3	N4	N17	U4	U8
U13	U17	W14	-	-	-	-
Not Connected Pins						
A6	A7	A13	B13	B16	C4	C7
C8	C13	C16	D5	D12	E19	F2
F3	F18	F19	G18	H1	H2	H20
J3	J4	M4	M19	N1	N2	N18
P20	R3	T1	T18	U20	V9	V13
V15	W6	W9	W10	W13	W16	Y6
Y9	Y13	Y14	-	-	-	-

5/27/97

Pin Locations for XC4013E/XL Devices

The following table may contain pinout information for unsupported device/package combinations. Please see the availability charts elsewhere in the XC4000 Series data sheet for availability information.

XC4013E /XL Pad Name	HT 144††	PQ 160	HT 176††	PQ/HQ 208	PG 223†	BG 225†	PQ/HQ 240	BG 256††	Bndry Scan
VCC	P128	P142	P155	P183	VCC*	VCC*	P212	VCC*	-
I/O (A8)	P129	P143	P156	P184	J3	E8	P213	C10	74
I/O (A9)	P130	P144	P157	P185	J2	B7	P214	D10	77
I/O (A19) ††	P131	P145	P158	P186	J1	A7	P215	A9	80
I/O (A18) ††	P132	P146	P159	P187	H1	C7	P216	B9	83
I/O	-	-	P160	P188	H2	D7	P217	C9	86
I/O	-	-	P161	P189	H3	E7	P218	D9	89
I/O (A10)	P133	P147	P162	P190	G1	A6	P220	A8	92
I/O (A11)	P134	P148	P163	P191	G2	B6	P221	B8	95
VCC	-	-	-	-	VCC*	VCC*	P222	VCC*	-
I/O	-	-	-	-	H4	C6	P223	A6	98
I/O	-	-	-	-	G4	F7	P224	C7	101
I/O	P135	P149	P164	P192	F1	A5	P225	B6	104
I/O	P136	P150	P165	P193	E1	B5	P226	A5	107
GND	P137	P151	P166	P194	GND*	GND*	P227	GND*	-
I/O	-	-	-	P195	F2	D6	P228	C6	110
I/O	-	-	P167	P196	D1	C5	P229	B5	113
I/O	-	P152	P168	P197	C1	A4	P230	A4	116
I/O	-	P153	P169	P198	E2	E6	P231	C5	119
I/O (A12)	P138	P154	P170	P199	F3	B4	P232	B4	122
I/O (A13)	P139	P155	P171	P200	D2	D5	P233	A3	125
I/O	-	-	-	-	F4	A3	P234	D5	128
I/O	-	-	-	-	E4	C4	P235	C4	131
I/O	P140	P156	P172	P201	B1	B3	P236	B3	134
I/O	P141	P157	P173	P202	E3	F6	P237	B2	137

XC4013E /XL Pad Name	HT 144††	PQ 160	HT 176††	PQ/HQ 208	PG 223†	BG 225†	PQ/HQ 240	BG 256††	Bndry Scan
I/O (A14)	P142	P158	P174	P203	C2	A2	P238	A2	140
I/O, SGCK1 †, GCK8 †† (A15)	P143	P159	P175	P204	B2	C3	P239	C3	143
VCC	P144	P160	P176	P205	VCC*	VCC*	P240	VCC*	-
GND	P1	P1	P1	P2	GND*	GND*	P1	GND*	-
I/O, PGCK1 †, GCK1 †† (A16)	P2	P2	P2	P4	C3	D4	P2	B1	146
I/O (A17)	P3	P3	P3	P5	C4	B1	P3	C2	149
I/O	P4	P4	P4	P6	B3	C2	P4	D2	152
I/O	P5	P5	P5	P7	C5	E5	P5	D3	155
I/O, TDI	P6	P6	P6	P8	A2	D3	P6	E4	158
I/O, TCK	P7	P7	P7	P9	B4	C1	P7	C1	161
I/O	-	P8	P8	P10	C6	D2	P8	D1	164
I/O	-	P9	P9	P11	A3	G6	P9	E3	167
I/O	-	-	-	P12	B5	E4	P10	E2	170
I/O	-	-	-	P13	B6	D1	P11	E1	173
I/O	-	-	-	-	D5	E3	P12	F3	176
I/O	-	-	-	-	D6	E2	P13	F2	179
GND	P8	P10	P10	P14	GND*	GND*	P14	GND*	-
I/O	P9	P11	P11	P15	A4	F5	P15	G3	182
I/O	P10	P12	P12	P16	A5	E1	P16	G2	185
I/O, TMS	P11	P13	P13	P17	B7	F4	P17	G1	188
I/O	P12	P14	P14	P18	A6	F3	P18	H3	191
VCC	-	-	-	-	VCC*	VCC*	P19	VCC*	-

XC4013E /XL Pad Name	HT 144††	PQ 160	HT 176††	PQ/HQ 208	PG 223†	BG 225†	PQ/HQ 240	BG 256††	Bndry Scan
I/O	-	-	-	-	D7	F2	P20	H2	194
I/O	-	-	-	-	D8	F1	P21	H1	197
I/O	-	-	P15	P19	C8	G4	P23	J2	200
I/O	-	-	P16	P20	A7	G3	P24	J1	203
I/O	P13	P15	P17	P21	B8	G2	P25	K2	206
I/O	P14	P16	P18	P22	A8	G1	P26	K3	209
I/O	P15	P17	P19	P23	B9	G5	P27	K1	212
I/O	P16	P18	P20	P24	C9	H3	P28	L1	215
GND	P17	P19	P21	P25	GND*	GND*	P29	GND*	-
VCC	P18	P20	P22	P26	VCC*	VCC*	P30	VCC*	-
I/O	P19	P21	P23	P27	C10	H4	P31	L2	218
I/O	P20	P22	P24	P28	B10	H5	P32	L3	221
I/O	P21	P23	P25	P29	A9	J2	P33	L4	224
I/O	P22	P24	P26	P30	A10	J1	P34	M1	227
I/O	-	-	P27	P31	A11	J3	P35	M2	230
I/O	-	-	P28	P32	C11	J4	P36	M3	233
I/O	-	-	-	-	D11	J5	P38	N1	236
I/O	-	-	-	-	D12	K1	P39	N2	239
VCC	-	-	-	-	VCC*	VCC*	P40	VCC*	-
I/O	P23	P25	P29	P33	B11	K2	P41	P1	242
I/O	P24	P26	P30	P34	A12	K3	P42	P2	245
I/O	P25	P27	P31	P35	B12	J6	P43	R1	248
I/O	P26	P28	P32	P36	A13	L1	P44	P3	251
GND	P27	P29	P33	P37	GND*	GND*	P45	GND*	-
I/O	-	-	-	-	D13	L2	P46	T1	254
I/O	-	-	-	-	D14	K4	P47	R3	257
I/O	-	-	-	P38	B13	L3	P48	T2	260
I/O	-	-	-	P39	A14	M1	P49	U1	263
I/O	-	P30	P34	P40	A15	K5	P50	T3	266
I/O	-	P31	P35	P41	C13	M2	P51	U2	269
I/O	P28	P32	P36	P42	B14	L4	P52	V1	272
I/O	P29	P33	P37	P43	A16	N1	P53	T4	275
I/O	P30	P34	P38	P44	B15	M3	P54	U3	278
I/O	P31	P35	P39	P45	C14	N2	P55	V2	281
I/O	P32	P36	P40	P46	A17	K6	P56	W1	284
I/O, SGCK2 †, GCK2 ††	P33	P37	P41	P47	B16	P1	P57	V3	287
O (M1)	P34	P38	P42	P48	C15	N3	P58	W2	290
GND	P35	P39	P43	P49	GND*	GND*	P59	GND*	-
I (M0)	P36	P40	P44	P50	A18	P2	P60	Y1	293
VCC	P37	P41	P45	P55	VCC*	VCC*	P61	VCC*	-
I (M2)	P38	P42	P46	P56	C16	M4	P62	W3	294
I/O, PGCK2 †, GCK3 ††	P39	P43	P47	P57	B17	R2	P63	Y2	295
I/O (HDC)	P40	P44	P48	P58	E16	P3	P64	W4	298
I/O	P41	P45	P49	P59	C17	L5	P65	V4	301
I/O	P42	P46	P50	P60	D17	N4	P66	U5	304
I/O	P43	P47	P51	P61	B18	R3	P67	Y3	307
I/O ($\overline{LDC}$)	P44	P48	P52	P62	E17	P4	P68	Y4	310
I/O	-	P49	P53	P63	F16	K7	P69	V5	313
I/O	-	P50	P54	P64	C18	M5	P70	W5	316
I/O	-	-	-	P65	D18	R4	P71	Y5	319
I/O	-	-	-	P66	F17	N5	P72	V6	322
I/O	-	-	-	-	E15	P5	P73	W6	325
I/O	-	-	-	-	F15	L6	P74	Y6	328
GND	P45	P51	P55	P67	GND*	GND*	P75	GND*	-
I/O	P46	P52	P56	P68	E18	R5	P76	W7	331
I/O	P47	P53	P57	P69	F18	M6	P77	Y7	334
I/O	P48	P54	P58	P70	G17	N6	P78	V8	337
I/O	P49	P55	P59	P71	G18	P6	P79	W8	340
VCC	-	-	-	-	VCC*	VCC*	P80	VCC*	-
I/O	-	-	P60	P72	H16	R6	P81	Y8	343
I/O	-	-	P61	P73	H17	M7	P82	U9	346
I/O	-	-	-	-	G15	N7	P84	Y9	349
I/O	-	-	-	-	H15	P7	P85	W10	352
I/O	P50	P56	P62	P74	H18	R7	P86	V10	355
I/O	P51	P57	P63	P75	J18	L7	P87	Y10	358
I/O	P52	P58	P64	P76	J17	N8	P88	Y11	361
I/O ($\overline{INIT}$)	P53	P59	P65	P77	J16	P8	P89	W11	364
VCC	P54	P60	P66	P78	VCC*	VCC*	P90	VCC*	-
GND	P55	P61	P67	P79	GND*	GND*	P91	GND*	-
I/O	P56	P62	P68	P80	K16	L8	P92	V11	367

XC4013E /XL Pad Name	HT 144††	PQ 160	HT 176††	PQ/HQ 208	PG 223†	BG 225†	PQ/HQ 240	BG 256††	Bndry Scan
I/O	P57	P63	P69	P81	K17	P9	P93	U11	370
I/O	P58	P64	P70	P82	K18	R9	P94	Y12	373
I/O	P59	P65	P71	P83	L18	N9	P95	W12	376
I/O	-	-	P72	P84	L17	M9	P96	V12	379
I/O	-	-	P73	P85	L16	L9	P97	U12	382
I/O	-	-	-	-	L15	R10	P99	V13	385
I/O	-	-	-	-	M15	P10	P100	Y14	388
VCC	-	-	-	-	VCC*	VCC*	P101	VCC*	-
I/O	P60	P66	P74	P86	M18	N10	P102	Y15	391
I/O	P61	P67	P75	P87	M17	K9	P103	V14	394
I/O	P62	P68	P76	P88	N18	R11	P104	W15	397
I/O	P63	P69	P77	P89	P18	P11	P105	Y16	400
GND	P64	P70	P78	P90	GND*	GND*	P106	GND*	-
I/O	-	-	-	-	N15	M10	P107	V15	403
I/O	-	-	-	-	P15	N11	P108	W16	406
I/O	-	-	-	P91	N17	R12	P109	Y17	409
I/O	-	-	-	P92	R18	L10	P110	V16	412
I/O	-	P71	P79	P93	T18	P12	P111	W17	415
I/O	-	P72	P80	P94	P17	M11	P112	Y18	418
I/O	P65	P73	P81	P95	N16	R13	P113	U16	421
I/O	P66	P74	P82	P96	T17	N12	P114	V17	424
I/O	P67	P75	P83	P97	R17	P13	P115	W18	427
I/O	P68	P76	P84	P98	P16	K10	P116	Y19	430
I/O	P69	P77	P85	P99	U18	R14	P117	V18	433
I/O, SGCK3 †, GCK4 ††	P70	P78	P86	P100	T16	N13	P118	W19	436
GND	P71	P79	P87	P101	GND*	GND*	P119	GND*	-
DONE	P72	P80	P88	P103	U17	P14	P120	Y20	-
VCC	P73	P81	P89	P106	VCC*	VCC*	P121	VCC*	-
PROGRAM	P74	P82	P90	P108	V18	M12	P122	V19	-
I/O (D7)	P75	P83	P91	P109	T15	P15	P123	U19	439
I/O, PGCK3 †, GCK5 ††	P76	P84	P92	P110	U16	N14	P124	U18	442
I/O	P77	P85	P93	P111	T14	L11	P125	T17	445
I/O	P78	P86	P94	P112	U15	M13	P126	V20	448
I/O	-	-	-	-	R14	N15	P127	U20	451
I/O	-	-	-	-	R13	M14	P128	T18	454
I/O (D6)	P79	P87	P95	P113	V17	J10	P129	T19	457
I/O	P80	P88	P96	P114	V16	L12	P130	T20	460
I/O	-	P89	P97	P115	T13	M15	P131	R18	463
I/O	-	P90	P98	P116	U14	L13	P132	R19	466
I/O	-	-	-	P117	V15	L14	P133	R20	469
I/O	-	-	-	P118	V14	K11	P134	P18	472
GND	P81	P91	P99	P119	GND*	GND*	P135	GND*	-
I/O	-	-	-	-	R12	L15	P136	P20	475
I/O	-	-	-	-	R11	K12	P137	N18	478
I/O	P82	P92	P100	P120	U13	K13	P138	N19	481
I/O	P83	P93	P101	P121	V13	K14	P139	N20	484
VCC	-	-	-	-	VCC*	VCC*	P140	VCC*	-
I/O (D5)	P84	P94	P102	P122	U12	K15	P141	M17	487
I/O ($\overline{CS0}$)	P85	P95	P103	P123	V12	J12	P142	M18	490
I/O	-	-	P104	P124	T11	J13	P144	M20	493
I/O	-	-	P105	P125	U11	J14	P145	L19	496
I/O	P86	P96	P106	P126	V11	J15	P146	L18	499
I/O	P87	P97	P107	P127	V10	J11	P147	L20	502
I/O (D4)	P88	P98	P108	P128	U10	H13	P148	K20	505
I/O	P89	P99	P109	P129	T10	H14	P149	K19	508
VCC	P90	P100	P110	P130	VCC*	VCC*	P150	VCC*	-
GND	P91	P101	P111	P131	GND*	GND*	P151	GND*	-
I/O (D3)	P92	P102	P112	P132	T9	H12	P152	K18	511
I/O ($\overline{RS}$)	P93	P103	P113	P133	U9	H11	P153	K17	514
I/O	P94	P104	P114	P134	V9	G14	P154	J20	517
I/O	P95	P105	P115	P135	V8	G15	P155	J19	520
I/O	-	-	P116	P136	U8	G13	P156	J18	523
I/O	-	-	P117	P137	T8	G12	P157	J17	526
I/O (D2)	P96	P106	P118	P138	V7	G11	P159	H19	529
I/O	P97	P107	P119	P139	U7	F15	P160	H18	532
VCC	-	-	-	-	VCC*	VCC*	P161	VCC*	-
I/O	P98	P108	P120	P140	V6	F14	P162	G19	535
I/O	P99	P109	P121	P141	U6	F13	P163	F20	538
I/O	-	-	-	-	R8	G10	P164	G18	541

XC4013E /XL Pad Name	HT 144††	PQ 160	HT 176††	PQ/HQ 208	PG 223†	BG 225†	PQ/HQ 240	BG 256††	Bndry Scan
I/O	-	-	-	-	R7	E15	P165	F19	544
GND	P100	P110	P122	P142	GND*	GND*	P166	GND*	-
I/O	-	-	-	-	R6	E14	P167	F18	547
I/O	-	-	-	-	R5	F12	P168	E19	550
I/O	-	-	-	P143	V5	E13	P169	D20	553
I/O	-	-	-	P144	V4	D15	P170	E18	556
I/O	-	P111	P123	P145	U5	F11	P171	D19	559
I/O	-	P112	P124	P146	T6	D14	P172	C20	562
I/O (D1)	P101	P113	P125	P147	V3	E12	P173	E17	565
I/O ($\overline{RCLK}$, RDY/$\overline{BUSY}$)	P102	P114	P126	P148	V2	C15	P174	D18	568
I/O	P103	P115	P127	P149	U4	D13	P175	C19	571
I/O	P104	P116	P128	P150	T5	C14	P176	B20	574
I/O (D0, DIN)	P105	P117	P129	P151	U3	F10	P177	C18	577
I/O, SGCK4 †, GCK6 †† (DOUT)	P106	P118	P130	P152	T4	B15	P178	B19	580
CCLK	P107	P119	P131	P153	V1	C13	P179	A20	-
VCC	P108	P120	P132	P154	VCC*	VCC*	P180	VCC*	-
O, TDO	P109	P121	P133	P159	U2	A15	P181	A19	0
GND	P110	P122	P134	P160	GND*	GND*	P182	GND*	-
I/O (A0, $\overline{WS}$)	P111	P123	P135	P161	T3	A14	P183	B18	2
I/O, PGCK4 †, GCK7 †† (A1)	P112	P124	P136	P162	U1	B13	P184	B17	5
I/O	P113	P125	P137	P163	P3	E11	P185	C17	8
I/O	P114	P126	P138	P164	R2	C12	P186	D16	11
I/O (CS1, A2)	P115	P127	P139	P165	T2	A13	P187	A18	14
I/O (A3)	P116	P128	P140	P166	N3	B12	P188	A17	17
I/O	-	-	-	-	P4	F9	P189	C16	20
I/O	-	-	-	-	N4	D11	P190	B16	23
I/O	P117	P129	P141	P167	P2	A12	P191	A16	26
I/O	-	P130	P142	P168	T1	C11	P192	C15	29
I/O	-	-	-	P169	R1	B11	P193	B15	32
I/O	-	-	-	P170	N2	E10	P194	A15	35
GND	P118	P131	P143	P171	GND*	GND*	P196	GND*	-
I/O	P119	P132	P144	P172	P1	A11	P197	B14	38
I/O	P120	P133	P145	P173	N1	D10	P198	A14	41
I/O	-	-	-	-	M4	C10	P199	C13	44
I/O	-	-	-	-	L4	B10	P200	B13	47
VCC	-	-	-	-	VCC*	VCC*	P201	VCC*	-
I/O (A4)	P121	P134	P146	P174	M2	A10	P202	C12	50
I/O (A5)	P122	P135	P147	P175	M1	D9	P203	B12	53
I/O	-	-	P148	P176	L3	C9	P205	A12	56
I/O	-	P136	P149	P177	L2	B9	P206	B11	59
I/O (A21) ††	P123	P137	P150	P178	L1	A9	P207	C11	62
I/O (A20) ††	P124	P138	P151	P179	K1	E9	P208	A11	65
I/O (A6)	P125	P139	P152	P180	K2	C8	P209	A10	68
I/O (A7)	P126	P140	P153	P181	K3	B8	P210	B10	71
GND	P127	P141	P154	P182	GND*	GND*	P211	GND*	-

6/9/97

* Pads labelled GND* or VCC* are internally bonded to Ground or VCC planes within the package. They have no direct connection to any specific package pin.

† = E only, †† = XL only

Additional XC4013E/XL Package Pins

PQ/HQ208

Not Connected Pins					
P1	P3	P51	P52	P53	P54
P102	P104	P105	P107	P155	P156
P157	P158	P206	P207	P208	-

5/5/97

PG223

VCC Pins					
D3	D10	D16	J4	J15	R4
R10	R15	-	-	-	-
GND Pins					
C7	C12	D4	D9	D15	G3
G16	K4	K15	M3	M16	R3
R9	R16	T7	T12	-	-

5/5/97

BG225

VCC Pins				
B2	B14	D8	H1	H15
R1	R8	R15	-	-
GND Pins				
A1	A8	D12	F8	G7
G8	G9	H2	H6	H7
H8	H9	H10	J7	J8
J9	K8	M8	-	-

5/5/97

The BG225 package pins in this table are bonded to an internal Ground plane on the XC4013E die. They must all be externally connected to Ground.

PQ/HQ240

GND Pins					
P22‡	P37‡	P83‡	P98‡	P143‡	P158‡
P204‡	P219‡	-	-	-	-
Not Connected Pins					
P195	-	-	-	-	-

6/9/97

‡ Pins marked with this symbol are used for Ground connections on some revisions of the device. These pins may not physically connect to anything on the current device revision. However, they should be externally connected to Ground, if possible.

BG256

VCC Pins					
C14	D6	D7	D11	D14	D15
E20	F1	F4	F17	G4	G17
K4	L17	P4	P17	P19	R2
R4	R17	U6	U7	U10	U14
U15	V7	W20	-	-	-
GND Pins					
A1	B7	D4	D8	D13	D17
G20	H4	H17	N3	N4	N17
U4	U8	U13	U17	W14	-
Not Connected Pins					
A7	A13	C8	D12	H20	J3
J4	M4	M19	V9	W9	W13
Y13	-	-	-	-	-

6/4/97

บรรณานุกรม

[1] Douglas L. Perry, "*VHDL*", (2nd Ed.), McGraw-Hill,Inc.

[2] Roger Lipsett, Carl R. Schaefer, Cary Ussery, "*VHDL: Hardware Description And Design*", Kluwer Acadamic Publishers

[3] Sudhakar Yalamanchili, "*VHDL Starter's Guide*", Prentice Hall

[4] Steve Carlson, "*Introduction To HDL-Based Design Using VHDL*", Synopsys Inc.

[5] George Milne , "*Formal Specification And Verification Of Digital Systems*", McGraw-Hill Book Company

[6] Fredrick F. Driscoll, "*Data Communication*", (International Ed.), Saurders College Publishing, 1992

[7] David H. Stein, "*Introduction To Digital Data Communications*", Delmar Publishers Inc., 1985

[8] William A. Shay, "*Understanding Data Communications And Networks*", PWS Publishing Company, 1995

[9] D. C. Green, "*Data Communication*", (2nd Ed.), Longman Scientific & Technical, 1995

[10] Stephen L. Wasson, "*Top-Down FPGA Design A 12-Step Program*"

[11] Motorola Semiconductors, "*Data Sheet MC6850*"

[12] IEEE Design & Test of Computers, " Embedded Core-Base Systems " , IEEE Circuits and Systems Society

[13] สุนทร วิทูสุรพจน์, "*การใช้งานไมโครคอนโทรลเลอร์ ตระกูล 8051*", บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)

[14] จิระศักดิ์ เหลืองอุไร, "*คัมภีร์การใช้งานการสื่อสารอนุกรมบน PC*", บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)

[15] The Programmable Logic Databook 1998, Xilinx