ลอจิกแอนนาไลเซอร์

LOGIC ANALYZER

นายธีร ไชยธวัช
นายสุเมธ กิจรื่นภิรมย์สุข

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2546

ลอจิกแอนนาไลเซอร์

LOGIC ANALYZER

โดย

นายธีร ไชยธวัช

นายสุเมธ กิจรื่นภิรมย์สุข

อาจารย์ที่ปรึกษา

อ.เจริญ วงษ์ชุ่มเย็น

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ลอจิกแอนนาไลเซอร์

LOGIC ANALYZER

คณะผู้จัดทำ นายธีร ไชยธวัช รหัส 44015329

นายสุเมธ กิจรื่นภิรมย์สุข รหัส 44015358

.....อาจารย์ที่ปรึกษา

(อ.เจริญ วงษ์ชุ่มเย็น)

ลอจิกแอนนาไลเซอร์

นายธีร ไชยธวัช 44015329
นายสุเมธ กิจรื่นภิรมย์สุข 44015358
อ.เจริญ วงษ์ชุ่มเย็น อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

ในปัจจุบันการพัฒนาทางด้านเทคโนโลยีเป็นไปอย่างรวดเร็ว อุปกรณ์มีขนาดเล็กลง แต่มีคุณภาพสูงขึ้น อุปกรณ์พื้นฐานที่สามารถวัดสัญญาณดิจิตอลได้ก็คือ ออสซิลโลสโคป แต่ออสซิลโลสโคปสามารถใช้วัดสัญญาณได้เพียงชั่วขณะเท่านั้น เมื่อเวลาผ่านไปก็ไม่สามารถย้อนกลับมาดูสัญญาณเก่าได้ จึงไม่สามารถที่จะวิเคราะห์สัญญาณทางตรรกะได้ อีกทั้งออสซิลโลสโคปยังไม่สามารถวัดสัญญาณได้พร้อมๆกันหลายช่อง ซึ่งในบางครั้งการวิเคราะห์สัญญาณทางตรรกะจะต้องศึกษาและพิจารณาสัญญาณพร้อมกันหลายสัญญาณ ทำให้ไม่สามารถใช้ออสซิลโลสโคปช่วยในการวิเคราะห์สัญญาณทางตรรกะได้อย่างเต็มที่

เครื่องมือที่สามารถวิเคราะห์สัญญาณทางตรรกะได้ และสามารถนำสัญญาณที่เกิดขึ้นมาแล้วมาแสดงผลได้อีก โดยแสดงผลพร้อมกันได้หลายช่อง ก็คือ ลอจิกแอนนาไลเซอร์ ซึ่งมีขายอยู่ในปัจจุบันแต่มีราคาแพงมาก ดังนั้นจึงได้พัฒนาโครงงานนี้ขึ้นมามีชื่อว่า “ลอจิกแอนนาไลเซอร์” ซึ่งเป็นเครื่องมือวิเคราะห์สัญญาณทางตรรกะ ที่สามารถวัดสัญญาณดิจิตอลได้พร้อมๆกันถึง 16 สัญญาณหรือ 16 ช่อง และมีหน่วยความจำที่ใช้สำหรับเก็บข้อมูล สามารถที่จะนำข้อมูลมาแสดงผลได้อีก หลังจากที่สัญญาณเกิดขึ้นแล้ว

LOGIC ANALYZER

Mr.Teera Chaitawat

Mr.Sumate Kitraunpiromsuk

Mr.Chareon Vongchumyen Advisor

Academic Year 2003

ABSTRACT

Nowadays technology development grows fast.Many appliances become small but have high efficeincy.The device which can measure digital signal is called Oscilloscope.This device can only use at a time,we cannot get old results back. It also cannot perform multi-channels signal measurement simultaneously.With these limits using Oscilloscope for logic analysis is impractical because the analysis need to work with manysignals and statistical results.

Logic Analyzer is a device that can do all tasks need for logic analysis.It can get data back and performmulti-channels signal measuement.But this device is very expensive.So this project is developed to serve all conditions and also named Logic Analyzer.It can measure 16 signals in the same time and have a memory to store data which can restore it back.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี ด้วยคำแนะนำ คำปรึกษาและคอยดูแลจากหลายๆ ฝ่ายด้วยกัน โดยเฉพาะอาจารย์ที่ปรึกษาที่ให้โอกาสให้ข้าพเจ้าได้ทำปริญญานิพนธ์ฉบับนี้ คอยให้ความเอาใจใส่ แนะนำและความช่วยเหลือเสมอมา คือ อ.เจริญ วงษ์ชุ่มเย็น ซึ่งต้องขอขอบพระคุณเป็นอย่างสูง

ขอขอบคุณภาควิชาวิศวกรรมคอมพิวเตอร์ที่ได้จัดเตรียมสิ่งอำนวยความสะดวก เพื่อให้การวิจัยและพัฒนาโปรแกรมเป็นไปได้ด้วยความสะดวกและรวดเร็ว รวมทั้งยังมีอินเทอร์เน็ตความเร็วสูงให้บริการ สำหรับการค้นคว้าหาความรู้ต่างๆ ซึ่งท้ายที่สุดแล้วก็ประกอบกันเป็นส่วนหนึ่งของโครงงานนี้

ขอขอบคุณพี่ๆ เพื่อนๆ น้องๆ ในห้องปฏิบัติการฮาร์ดแวร์ที่คอยสร้างความคึกคึนยามอยู่ในห้อง เป็นกำลังใจเสมอมา และที่ขาดไม่ได้ต้องขอบคุณห้องฮาร์ดแวร์ที่ให้ที่อาศัย พักผ่อน และช่วยให้ท้องหายหิว มีแรงกายแรงใจในการทำงาน

และสุดท้ายต้องขอขอบคุณบุคคลที่สำคัญที่สุดในชีวิตที่ทำให้ข้าพเจ้ามีวันนี้ นั่นคือ บิดา มารดาและบุคคลในครอบครัว อันเป็นที่เคารพรัก ซึ่งได้เลี้ยงดู คอยสั่งสอนข้าพเจ้ามาเป็นอย่างดี พร้อมให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ ความรักเสมอมา ข้าพเจ้าขอกราบขอบพระคุณมา ณ ที่นี้ด้วย

ธีร ไชยธวัช

สุเมธ กิจรื่นภิรมย์สุข

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

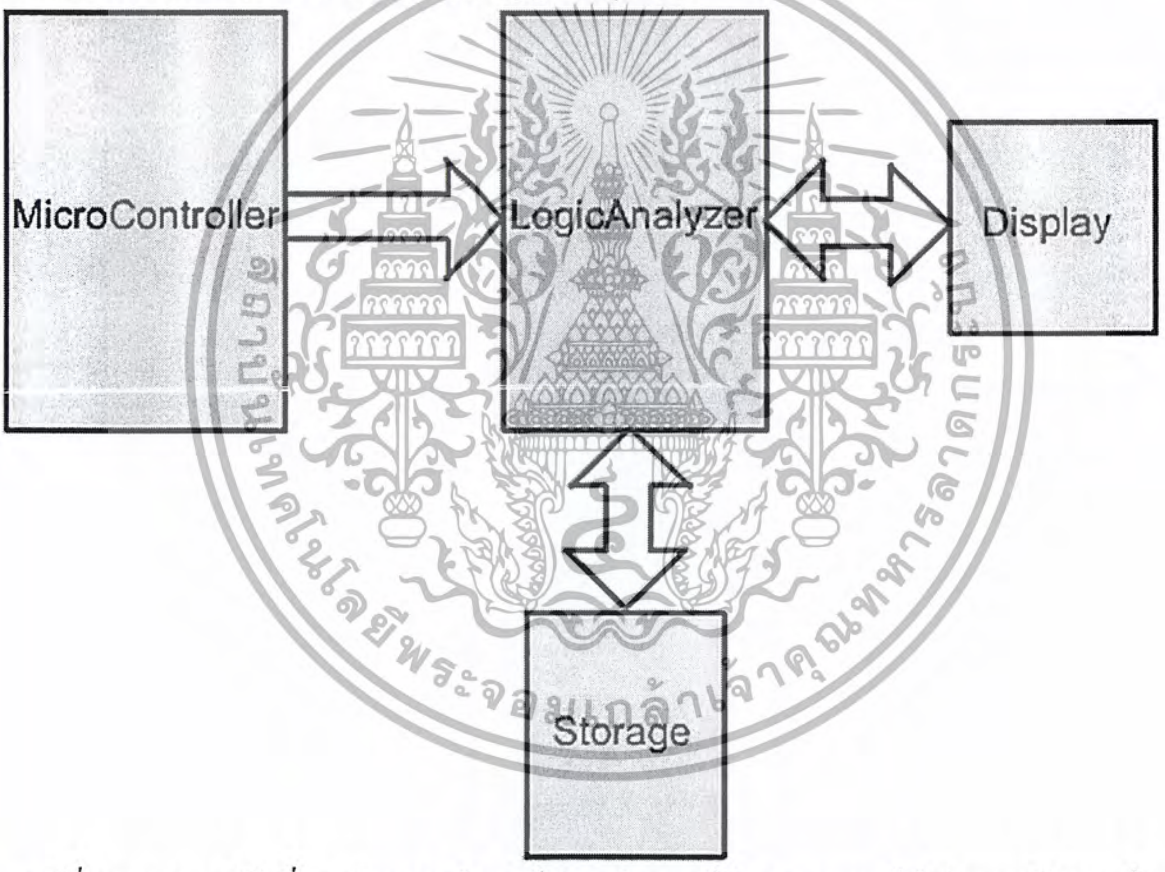
บทที่1

บทนำ

1.1 หลักการและเหตุผล

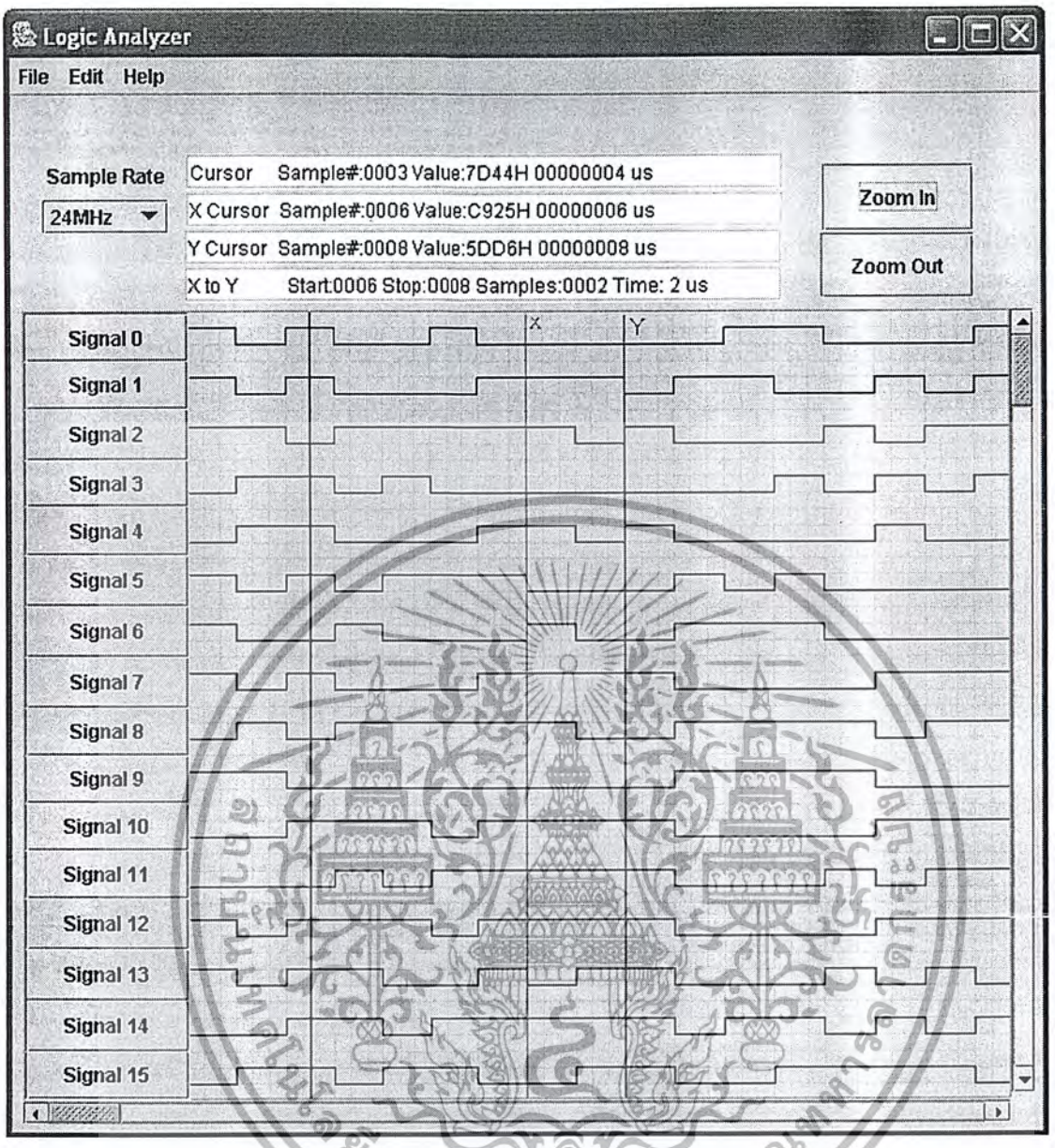
ในปัจจุบันนี้เทคโนโลยีทางด้านคอมพิวเตอร์ได้เจริญไปอย่างรวดเร็วโดยทั่วไปมักใช้ชิพ (CHIP) สำเร็จที่สร้างขึ้นมาเฉพาะงาน หรือใช้ไมโครคอนโทรเลอร์ การวิเคราะห์การทำงานถ้าใช้เครื่องมือวัดธรรมดา เช่น ออสซิสโลสโคป หรือ มิเตอร์ มาทำการวัดต่างๆ จะยุ่งยากมาก และขณะเดียวกันนั้น จะสามารถตรวจสอบได้เฉพาะความผิดพลาดทางฮาร์ดแวร์เท่านั้น ส่วนคำสั่งทางซอร์ฟแวร์จะวิเคราะห์ไม่ได้เลยด้วยเหตุนี้จึงมีการพัฒนาเครื่องวิเคราะห์สัญญาณทางตรรกะขึ้นมาใช้งาน

เครื่องลอจิกแอนนาไลเซอร์จะทำหน้าที่นำเอาสัญญาณที่วัดได้มาแสดงให้ผู้ใช้ได้ทราบถึง สภาวะทางตรรกะในช่วงเวลานั้นๆ ได้พร้อมกันหลายช่องสัญญาณดังรูปที่ 1-1



รูปที่ 1-1 แสดงการใช้เครื่องลอจิกแอนนาไลเซอร์ ทำการวิเคราะห์สัญญาณจากบอร์ดไมโครคอนโทรเลอร์

จากรูปที่ 1-1 สมมติต้องการทราบขั้นตอนการทำงานของบอร์ดไมโครคอนโทรเลอร์บอร์ดหนึ่ง จะกระทำโดยใช้เครื่องวิเคราะห์สัญญาณทางตรรกะเข้าช่วยโดยใช้สายนำสัญญาณจับสัญญาณที่พอร์ตข้อมูลดังรูป เครื่องวิเคราะห์ทางตรรกะจะแสดงผลที่จับได้ออกมาทางจอภาพในลักษณะสัญญาณทางตรรกะดังนี้



รูปที่ 1-2 แสดงลักษณะสัญญาณทางตรรกะที่จับได้ออกมาทางจอภาพ

จากรูปที่ 1-2 รูปที่ได้จะทำให้สามารถตรวจสอบได้ทันทีว่าในขณะนั้นไมโครคอนโทรเลอร์กำลังทำคำสั่งอะไรอยู่ ถูกต้องหรือไม่ ด้วยเหตุนี้จึงทำให้การวิเคราะห์การทำงานของระบบที่ต้องการตรวจสอบได้เร็วขึ้น

1.2 วัตถุประสงค์

เครื่องลอจิกแอนนาไลเซอร์ ที่พัฒนานี้เป็นการพัฒนา FPGA (Field Programmable Gate Array) ให้เป็นลอจิกแอนนาไลเซอร์ ที่สามารถตรวจจับสัญญาณดิจิตอลได้ถึง 16 ช่องสัญญาณและ อัตราสุ่มสัญญาณสุ่มได้ถึง 24 เมกกะเฮิทซ์ เพื่อการรองรับการตรวจจับวิเคราะห์สัญญาณของไมโครคอนโทรเลอร์ ซึ่งมีอัตราความเร็วของสัญญาณนาฬิกาอยู่ที่ 12-20 เมกกะเฮิทซ์ โดยมีความสามารถและฟังชั่นพิเศษต่างๆดังนี้

- สามารถตั้งชื่อและสีของสัญญาณนั้นๆได้
- สามารถเลือกอัตราการสุ่มข้อมูล (Sample Rate) ได้

- สามารถจัดการแสดงเป็นกลุ่มข้อมูลได้
- สามารถ ตั้งการ Trigger โดยเลือกเป็นแบบ Edge Trigger หรือเป็น Pattern Trigger
- สามารถค้นหาตาม Pattern ที่ต้องการได้
- สามารถย่อขยายขนาดของรูปสัญญาณได้
- สามารถกำหนดช่วงเวลาโดยตั้งจุดเริ่มต้น จุดสิ้นสุดโดยจะแสดงค่าของชุดข้อมูลนั้นและระยะเวลาที่ใช้
- สามารถเลือกสิ่งแวดล้อมตามต้องการได้

1.3 ผลที่คาดว่าจะได้รับ

1.3.1 ความรู้ความเข้าใจเกี่ยวกับความเร็วในการติดต่อกับคอมพิวเตอร์ ผ่านพอร์ต อนุกรม

1.3.2 ความรู้ความเข้าใจเกี่ยวกับ วงจรอิเล็กทรอนิกส์ที่จะใช้เป็นเครื่องลอจิกแอนนาไลเซอร์

1.3.3 ความรู้ความเข้าใจเกี่ยวกับ การออกแบบวงจร โดยใช้ FPGA

1.3.4 ความรู้ความเข้าใจเกี่ยวกับ การออกแบบและเขียนโปรแกรม ที่จะใช้ในการควบคุม FPGA

1.3.5 ความรู้ความเข้าใจเกี่ยวกับ การออกแบบและเขียนโปรแกรม ที่จะใช้ในการติดต่อผ่านทางพอร์ตอนุกรม และโปรแกรมเพื่อแสดงผลของเครื่องเครื่องลอจิกแอนนาไลเซอร์

1.3.6 ลอจิกแอนนาไลเซอร์ ที่สามารถตรวจจับสัญญาณดิจิตอลได้ถึง 16 ช่องสัญญาณและ อัตราสุ่มสัญญาณสุ่มได้ถึง 24 เมกกะเฮิทซ์ เพื่อการรองรับการตรวจจับวิเคราะห์สัญญาณของไมโครคอนโทรเลอร์

1.3.7 เครื่องลอจิกแอนนาไลเซอร์ ให้มีความสามารถ ฟังก์ชั่นพิเศษต่างๆ เพื่อสะดวกในการใช้งาน

1.4 ขอบเขตของโครงงาน

ขอบเขตโครงงานนี้ เพื่อต้องการพัฒนาอุปกรณ์เรียกว่า เครื่องลอจิกแอนนาไลเซอร์(Logic Analyzer) ที่มีประสิทธิภาพสูง และมีราคาไม่แพง เพื่อใช้เป็นเครื่องมือสำหรับตรวจสอบข้อมูลทางดรรกะ

เครื่องลอจิกแอนนาไลเซอร์ที่พัฒนานี้เป็นการพัฒนา FPGA (Field Programmable Gate Array)ให้เป็นเครื่องลอจิกแอนนาไลเซอร์ที่สามารถตรวจจับสัญญาณได้ถึง 16 ช่องสัญญาณและ อัตราสุ่มสัญญาณสุ่มได้ถึง 24 เมกกะเฮริทซ์ เพื่อการรองรับการตรวจจับวิเคราะห์สัญญาณของไมโครคอนโทรเลอร์ ซึ่งมีอัตราความเร็วของสัญญาณนาฬิกาอยู่ที่ 12-20 เมกกะเฮริทซ์

ลอจิกแอนนาไลเซอร์ที่พัฒนาขึ้นนี้มีขอบเขตในการพัฒนา 2 ส่วนด้วยกันคือ

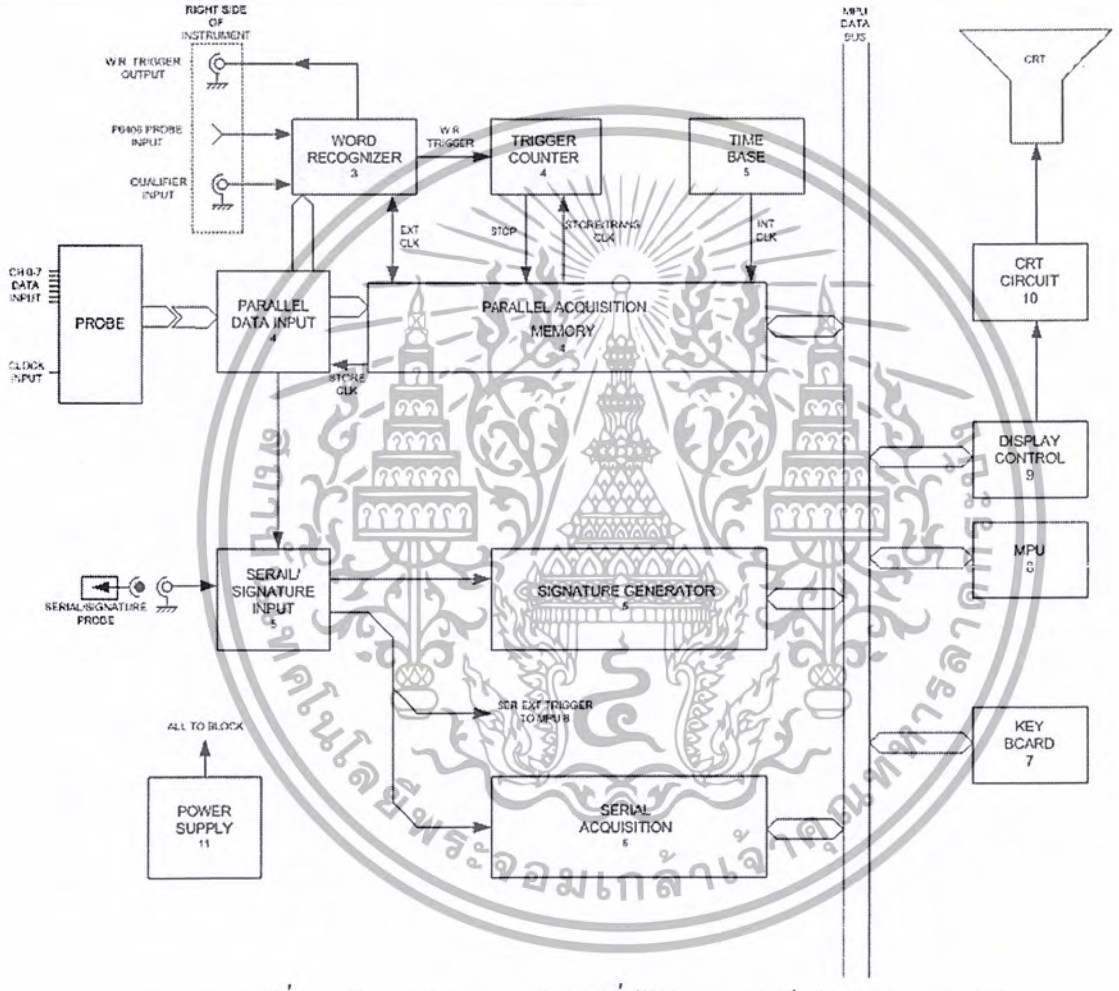
1.4.1วงจรประกอบด้วยวงจรของเครื่องลอจิกแอนนาไลเซอร์ซึ่งพัฒนาจาก FPGA (Field Programmable Gate Array) ซึ้งสามารถตรวจจับสัญญาณได้ถึง 16 ช่องสัญญาณและสามารถใช้อัตราสุ่มสัญญาณได้ที่ความถี่พอที่จะตรวจจับสัญญาณของไมโครคอนโทรเลอร์

1.4.2โปรแกรมควบคุมระบบของเครื่องลอจิกแอนนาไลเซอร์เพื่อเชื่อมต่อลอจิกแอนนาไลเซอร์เข้ากับคอมพิวเตอร์ส่วนบุคคล แสดงการวิเคราะห์สัญญาณที่สามารถตรวจจับจากไมโครคอนโทรเลอร์

บทที่ 2
ทฤษฎีและหลักการ

2.1 โครงสร้างและการทำงานทั่วไปของลอจิกแอนนาไลเซอร์

โครงสร้างและหลักการทำงานทั่วไปของเครื่องวิเคราะห์สัญญาณทางตรรกะสามารถแสดงดังบล๊อคไดอะแกรมในรูปที่ 2-1 ซึ่งประกอบด้วยส่วนย่อยๆ ดังนี้



รูปที่ 2-1 โครงสร้างและการทำงานทั่วไปของเคราะห์สัญญาณทางตรรกะ

2.1.1 ส่วนของสัญญาณด้านขาเข้าแบบขนาน (Parallel data input)

วงจรจะประกอบไปด้วย วงจรศักดาไฟฟ้าขีดเริ่มเปลี่ยน (Threshold voltage) , วงจรแปลงจาก ECT-TTL , วงจรหน่วงสัญญาณ , อินเวอร์เตอร์ , วงจรสุ่มสัญญาณตัวอย่าง (Sample/Latch) ซึ่งจุดประสงค์ ของส่วนนี้เป็นตัวปรับระดับของสัญญาณขาเข้าให้มีระดับที่เหมาะสมเพื่อนำไปประมวลผลต่อไป

2.1.2 ส่วนรู้จำของคำ (Word recognizer)

วงจรของส่วนรู้จำของคำจะให้สัญญาณขาออกเป็นลอจิก "1" (High) เมื่อสัญญาณทางตรรกะของสัญญาณรับเข้ามาจากแท่งรับสัญญาณ (Probe) เข้ากันได้กับสัญญาณขาที่เลือกไว้ (Qualifier input) ซึ่งสัญญาณจากส่วนรู้จำของคำนี้เป็นสัญญาณกระตุ้นแบบอะซิงโครนัส (Asynchronous trigger pulse)

2.1.3 ส่วนหน่วยความจำรับสัญญาณแบบขนาน(Parallel acquisition memory)

เป็นวงจรที่ประกอบไปด้วยหน่วยความจำ (RAM) , ตัวนับแอดเดรส (Address counter) ซึ่งหน่วยความจำนี้เป็นตัวเก็บสัญญาณขาเข้าซึ่งผ่านการปรับระดับของสัญญาณแล้ว และตัวนับแอดเดรสเป็นตัวเลือกบริเวณของแอดเดรสที่ต้องการพิจารณา

2.1.4 ส่วนหน่วงสัญญาณกระตุ้น (Trigger delay)

วงจรหน่วงสัญญาณกระตุ้นจะเป็นวงจรประกอบด้วย วงจรกระตุ้น (Trigger) และหน่วงสัญญาณต่างๆ เช่น Trigger delay counter , Trigger gate stage , Delay gate stage และอื่นๆ ซึ่งจะทำให้เกิดสัญญาณขาออกที่หน่วงเวลาการกระตุ้น (Delayed trigger output) เป็นสัญญาณที่ไปกระตุ้น วงจร Parallel Acquisition memory

2.1.5 วงจรคาบเวลา (Time base)

เป็นวงจรประกอบไปด้วยความถี่มาตรฐาน , วงจรแบ่งความถี่ ซึ่งหน้าที่ของวงจรจะเป็นส่วนให้สัญญาณนาฬิกากับตัวเอ็มพียู (MPU) และวงจรควบคุมการแสดงผล

2.1.6 วงจรสัญญาณขาเข้าแบบอนุกรม (Serial / Signature input)

เป็นวงจรทำหน้าที่เปรียบเทียบสัญญาณขาเข้า ปรับแต่งระดับของสัญญาณให้เหมาะสม เพื่อเข้าสู่วงจรในส่วนอนุกรม (Serial) อื่นๆต่อไป

2.1.7 ตัวกำเนิดสัญญาณ (Signature generator)

จะเป็นตัวรับสัญญาณแบบอนุกรมเข้ามาถอดรหัส (Decode) ของสัญญาณขาเข้า และทำให้เป็นสัญญาณขาออกเป็น 16 บิต (16-bit output) และส่งเข้าไปยังแรม(RAM) แสดงผลในหน่วยเอ็มพียู(MPU) ต่อไป

2.1.8 วงจรรับสัญญาณขาเข้าแบบอนุกรม (Serial data acquisition)

วงจรรับสัญญาณขาเข้าแบบอนุกรม ประกอบด้วย Baud rate generator, USART (Programmable communication interface) , Data buffer ซึ่งวงจรนี้จะเป็นกระจายสัญญาณนาฬิกาแบบภายนอกและภายใน (Internal / External) และรับสัญญาณขาเข้าแบบอนุกรมแล้วแปลงเป็นสัญญาณขาออกแบบขนาน โดยส่งผ่าน Data buffer เข้าสู่ Data bus MPU ต่อไป

2.1.9 คีย์บอร์ด และเอ็มพียู (Keyboard and MPU)

คีย์บอร์ดจะเป็นตัวส่งสัญญาณ ไปยัง MPU โดยผ่าน Latch และ Gate เพื่อจัดเวลาให้เหมาะสม ส่วน MPU นั้นเป็นหน่วยของไมโครโปรเซสเซอร์ ซึ่งเป็นหน่อยประมวลผลต่างๆนั้นเอง รวมถึง ROM, RAM ,Decode

2.1.10 หน่วยควบคุมการแสดงผล (Display control)

จะเป็นวงจรประกอบขึ้นด้วย RAM ซึ่งจะเก็บข้อมูลที่แสดงผลออกมา ซึ่งได้รับการประมวลผลโดยเอ็มพียู(MPU) ไว้แล้ว เพื่อส่งออกไปทางจอภาพ

2.1.11 จอภาพ (CRT)

เป็นหน่วยประมวลผลเพื่อแสดงสัญญาณออกที่จอภาพ

2.1.12 วงจรจ่ายกำลัง (Power supply)

เป็นวงจรจ่ายกำลัง หรือ จ่ายศักดาไฟฟ้าแบบ DC

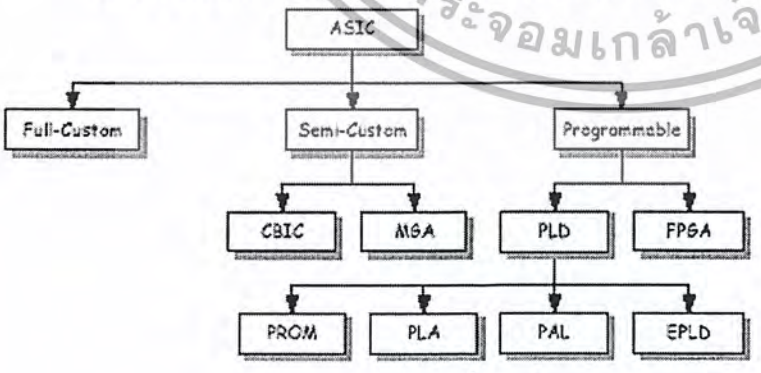
2.2 ความเป็นมาของ FPGA

ในช่วงก่อนทศวรรษ 1970 อุตสาหกรรมเซมิคอนดัคเตอร์ได้ถูกปลุกให้ตื่นตัวขึ้นหลังจากที่มีการประดิษฐ์วงจรรวมหรือไอซี ตัวแรกสำเร็จ ในยุคแรกนั้นไอซีขนาดเล็กหรือ SSI (Small-Scale Integration) ประกอบไปด้วยเกทดิจิตอลจำนวนไม่มาก นัก(ประมาณ 1 ถึง 10 ตัว) ต่อมาได้มีการเพิ่มปริมาณของเกทดิจิตอลและฟังก์ชันทางลอจิกให้มากขึ้นจนเป็น MSI (Medium- Scale Integration) การพัฒนาไอซีเป็นไปอย่างต่อเนื่องจนมาถึงยุคของ LSI (Large-Scale Integration) ซึ่งเป็นยุคที่มีการ สร้างไมโครโปรเซสเซอร์ตัวแรกขึ้น และในปัจจุบันเป็นยุคของ VLSI (Very Large-Scale Integration) ซึ่งเทคโนโลยีในการ สร้างไอซีรุดหน้าจนสามารถสร้างไมโครโปรเซสเซอร์ขนาด 64 บิต ที่มีหน่วยความจำแคชกับหน่วยคำนวณทางคณิตศาสตร์ ของโฟล์ทติงพอยน์ (Floating-Point Arithmetic Units) รวมอยู่ภายในตัวมันและเนื่องจากการปรับปรุงเทคโนโลยีของกระบวนการสร้างซีมอส ที่มีมาอย่างต่อเนื่องทำให้ขนาดของทรานซิสเตอร์ที่บรรจุอยู่ภายในไอซีมีขนาดเล็กลงเรื่อยๆ จนบางคนโดยเฉพาะ ในญี่ปุ่นใช้คำว่า ULSI (Ultra large Scale Integration) เพื่อใช้เรียกระดับของไอซีในปัจจุบันแต่คนส่วนมาก ยังมักนิยมเรียกเพียงแค่ VLSI

จากการปรากฏตัวของ VLSI ในช่วงทศวรรษ 1980 ทำให้วิศวกรเริ่มมีการออกแบบไอซีตามความต้องการของลูกค้าซึ่งใช้ใน ระบบที่เจาะจงนอกเหนือจากการใช้ไอซีมาตรฐานเพียงอย่างเดียว โดยไอซีเหล่านี้มีชื่อเรียกว่า ASIC : Application-Specific Integrated Circuit (ออกเสียงว่า เอ-ซิก) ซึ่งตัวอย่างของ ASIC ได้แก่ ชิพไอซีที่ใช้ สำหรับตู้กดาของเล่นพูดได้ คาวเทียม และ ชิพที่ภายในบรรจุด้วยไมโครโปรเซสเซอร์กับอุปกรณ์ทางลอจิกอื่นๆ

2.2.1 ประเภทของ ASIC

ASIC แบ่งเป็น 3 ประเภทใหญ่ๆ คือ Full-custom, Semi-custom และ Programmable ดังรูปที่ 2-2



รูปที่ 2-2 ประเภทของ ASIC

2.2.1.1 Full-custom

ASIC ประเภทนี้ลูกค้าจะเป็นผู้ออกแบบเซลล์ลอจิก (เช่น แอนด์เกท ออร์เกท มัลติเพล็กเซอร์ และฟลิบฟล็อบ) และลักษณะ การจัดวางอุปกรณ์บนตัวไอซีรวมถึงหน้ากากสำหรับควบคุมการเจือและสร้างชั้นสาร (Mask) ต่างๆ ที่ ใช้ในการทำไอซีเอง ดังนั้นค่าใช้จ่ายในการออกแบบและการผลิตจะสูงมาก

2.2.1.2 Semi-custom

ASIC ประเภทนี้เซลล์ลอจิกจะถูกออกแบบเอาไว้ก่อนแล้วในรูปแบบของไลบรารีและลูกค้าจะเป็นผู้ออกแบบ Mask ต่างๆเอง ตัวอย่างของไอซีประเภทนี้ได้แก่ Standard-Cell-Based ASIC และ Masked Gate-Array-Based ASIC

ก) Standard-Cell-Based ASIC

ไอซีประเภทนี้จะมีพื้นที่สำหรับจัดวางเซลล์ลอจิกมาตรฐานซึ่งถูกออกแบบเอาไว้แล้ว ในบางครั้งเซลล์ มาตรฐานเหล่านี้จะถูกนำมาประกอบกันเป็นเซลที่มีขนาดใหญ่ขึ้นเรียกว่า Megacell สำหรับการออกแบบนั้นผู้ออกแบบจะทำเพียงแค่กำหนดตำแหน่งของเซลล์มาตรฐานและการเชื่อมต่อภายในของแต่ ละเซลล์เท่านั้นแต่อย่างไรก็ดีเซลล์ต่างๆ เหล่านี้สามารถวางที่ตำแหน่งใดๆ ก็ได้บนแผ่นเวเฟอร์ซิลิกอน นั่นก็หมายความว่าชั้น Mask จะถูกจัดวางตามความต้องการของผู้ออกแบบ

ข) Masked Gate-Array-Based ASIC

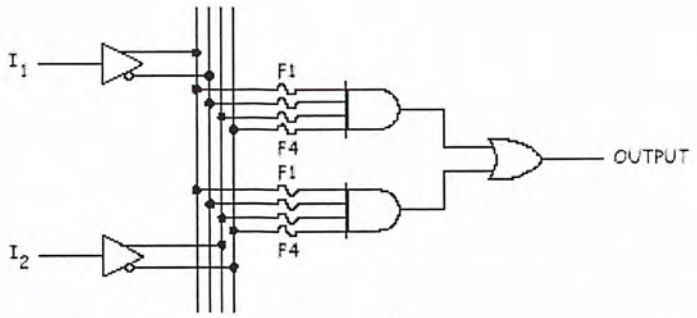
ไอซีชนิดนี้จะมีทรานซิสเตอร์หรือเกทถูกสร้างมาในลักษณะของอะเรย์สองมิติบนแผ่นเวเฟอร์ซิลิกอน และผู้ออกแบบจะทำการออกแบบ Mask เพื่อใช้สำหรับกำหนดการต่อเชื่อมของทรานซิสเตอร์แต่ละตัว

2.2.1.3 Programmable

ASIC ประเภทนี้เซลล์ลอจิกจะถูกออกแบบไว้ก่อนเช่นเดียวกับ Semi-Custom แต่ชั้นของ Mask จะไม่สามารถเปลี่ยนแปลงได้ ตามความต้องการของผู้ออกแบบ ไอซีประเภทนี้ยังแบ่งออกเป็น 2 ชนิดคือ Programmable Logic Device (PLD) และ Field Programmable Gate Array (FPGA)

ก) Programmable Logic Device (PLD)

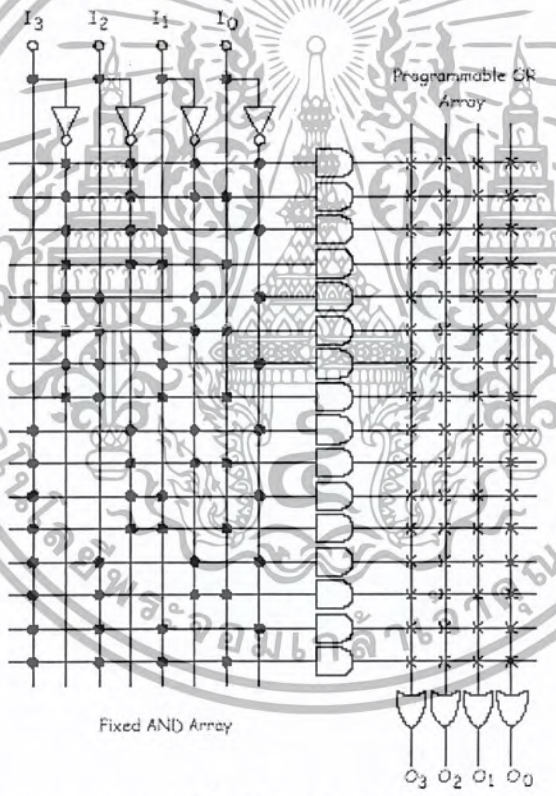
มีโครงสร้างภายในเป็นวงจรพื้นฐานทางด้านลอจิกต่อกันอยู่เป็นกลุ่มซึ่งมีทั้งวงจรคอมบิเนชัน (Combination) และซีเควนเชียล (Sequential) สำหรับเทคโนโลยีของวงจรที่ใช้สร้าง PLD จะมีทั้ง TTL, ECL และ CMOS ตามความเหมาะสมของแต่ละระบบ ไอซี PLD ทุกชนิดมีหลักการพื้นฐานของวงจรภายในที่เหมือนกันโดยมี วงจรคอมบิเนชันที่เป็นผลคูณร่วม บวก (Sum of product)ประกอบไปด้วยชุดของแอนด์เกทต่อร่วมกับออร์ เกทและในการ โปรแกรมจะเป็นการเลือกว่าอินพุทภายในของแอนด์เกทกับสัญญาณอินพุทใดบ้างที่จะต้อง ต่อ ถึงกันซึ่งมีทั้งจากภายนอกและสัญญาณป้อนกลับจากเอาท์พุทภายในเอง เช่น การติดต่ออินพุทของออร์ เกทกับเอาท์พุทของแอนด์เกทตัวต่างๆ สำหรับการโปรแกรมทางกายภาพนั้นอินพุท ต่างๆ ของอุปกรณ์ทุกตัว จะถูกต่อผ่านฟิวส์เข้ากับแหล่งสัญญาณ ซึ่งถ้าไม่ต้องการใช้สัญญาณ ใดก็จะตัดฟิวส์ตัวนั้นทิ้งทำให้สามารถ โปรแกรมได้เพียงครั้งเดียว ไอซี PLD บางชนิดใช้ มอสทรานซิสเตอร์แทนฟิวส์ทำให้สามารถโปรแกรมโดย ใช้กระแสไฟฟ้าและสามารถลบแล้ว โปรแกรมเข้าไปใหม่ได้อีก สำหรับไอซีในตระกูล PLD ได้แก่ PROM, PAL, PLA และ EPLD



รูปที่ 2-3 วงจรพื้นฐานของ PLD ซึ่งอยู่ในรูปผลคูณร่วมบวก

- PROM (Programmable Read Only Memory)

PROM คือหน่วยความจำประเภท ROM ซึ่งนับว่าเป็นไอซี PLD ชนิดหนึ่งซึ่งวงจรภายใน ของ PROM ประกอบไปด้วยอะเรย์ของแอนด์และออร์เกท (And - Or Array) ผลลัพธ์ที่ขา ดาต้าเอาท์พุทสามารถแสดงได้ในสมการของฟังก์ชันผลคูณร่วมบวก (Sum of product) ของสัญญาณอินพุทที่ขาแอดเดรส



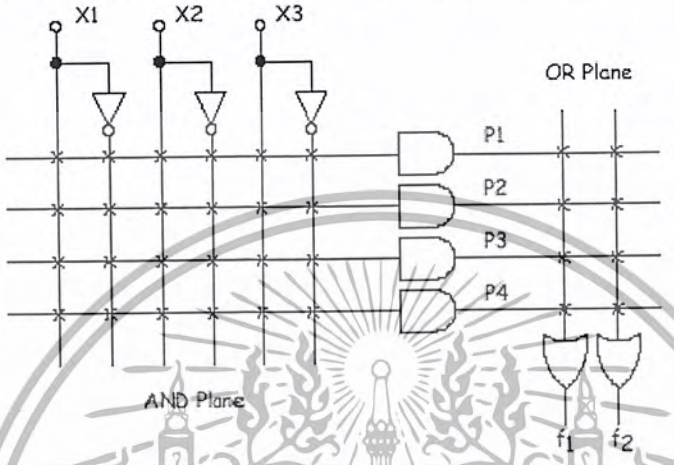
รูปที่ 2-4 ลักษณะของ PROM เมื่อเปรียบเทียบเป็นวงจรในรูปผลคูณร่วมบวก

รูปที่ 2-4 แสดงถึงลักษณะการเชื่อมต่อแอนด์เกทและออร์เกทของ PROM ขนาด 16x4 บิต วงจรทางด้านซ้ายบนสุดเป็นแอนด์เกทจะให้ผลคูณ (Product) ของกรณีที่อินพุทเป็น0000 แอนด์เกทที่อยู่ถัดลงมาเป็นผลคูณของกรณีที่อินพุทเป็น 0001, 0010, ...จนถึงตัวล่างสุดคือ ผลคูณในกรณีที่อินพุทเป็น 1111 ซึ่งสำหรับ PROM ที่มีจำนวนอินพุท n ตัวจะมีค่าอินพุทที่ เป็นไปได้ทั้งหมดเท่ากับ 2n และค่าอินพุทเหล่านี้จะถูกจัดวางอยู่ในส่วนอะเรย์ของ AND ซึ่งไม่สามารถแก้ไขได้ แต่ในส่วนของ OR จะเป็นส่วนที่อนุญาตให้ทำการโปรแกรมได้ และเนื่องจากการที่ด้าน AND ของ PROM มีการคอมบิเนชันของอินพุทที่เป็นไปได้ทั้งหมด

ดังนั้นผู้ออกแบบจึงไม่จำเป็นต้องทำการลดรูปของฟังก์ชันลอจิกที่ออกแบบไว้เลยแต่อย่าง ไรก็ดีการกระทำเช่นนี้อาจทำให้เกิดจำนวนวงจรที่ไม่มีประสิทธิภาพจำนวนมากบนตัว ชิพได้

- PLA (Programmable Logic Array)

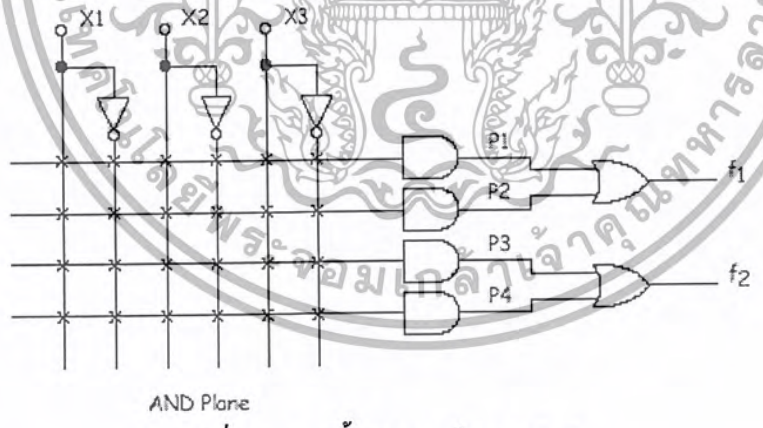
ลักษณะเด่นของ PLA คือสามารถโปรแกรมการเชื่อมต่อได้ทั้งทางด้าน AND และด้าน OR ทำให้มีความยืดหยุ่นในการใช้งานมาก แต่อย่างไรก็ดีข้อเสียที่เห็นได้อย่างชัดเจน ของ PLA คือความยุ่งยากในการสร้างและคุณสมบัติทางด้านความเร็วที่ลดลงเนื่องจาก สัญญาณจะต้องวิ่งผ่านอะเรย์ของ AND และ OR



รูปที่ 2-5 วงจรพื้นฐานภายในของ PLA

- PAL (Programmable Array Logic)

PAL มีลักษณะโครงสร้างที่ใกล้เคียงกับ PROM และ PLA มาก แต่การโปรแกรม PAL จะสามารถทำได้เพียงด้าน AND เท่านั้น



รูปที่ 2-6 วงจรพื้นฐานภายในของ PAL

- EPLD (Erasable Programmable Logic Device)

EPLD เป็นอุปกรณ์ที่สามารถทำการโปรแกรมได้หลายครั้งซึ่งเหมาะสำหรับการ ทำวงจรต้นแบบ สำหรับเทคโนโลยีที่ใช้ในการสร้างจะเหมือนกับ CMOS EPROM คือ ใช้มอสทรานซิสเตอร์เชื่อมต่อระหว่างสัญญาณอินพุทกับจุดที่ต้องการแทนการ ใช้ฟิวส์แบบเดิมทำให้สามารถโปรแกรมการต่อวงจรภายในอุปกรณ์ด้วยการจ่าย ไฟฟ้าตามขนาดที่กำหนดไว้และลบได้โดยใช้แสงอัลตร้าไวโอเลทฉาย ผ่านช่อง หน้าต่างกระจกของตัวชิพ

ข) Field-Programmable Gate Array (FPGA)

เป็นอุปกรณ์ที่มีความซับซ้อนมากกว่า PLD ไปอีกระดับหนึ่ง ซึ่งในความเป็นจริงแล้ว PLD และ FPGA แตก ต่างกันน้อยมาก สำหรับ FPGA แล้วนับว่าเป็นอุปกรณ์ตัวใหม่ในตระกูลของ ASIC ซึ่งมีการเจริญเติบโตอย่าง รวดเร็วและมีบทบาทที่สำคัญในการเข้ามาแทนที่ระบบอิเล็กทรอนิกส์ที่ใช้ TTL โครงสร้างภายในของ FPGA ประกอบไปด้วยอะเรย์ของลอจิกเกทต่างๆมากมาย ซึ่งในปัจจุบันความจุเกทภายใน ตัวชิพ FPGA ได้เพิ่มขึ้น จากระดับไม่กี่พันตัวจนถึงระดับล้านตัวซึ่งสามารถรองรับวงจรดิจิตอลที่มีความสลับซับซ้อนได้เป็นอย่างดี นอกจากนี้ในด้านการออกแบบพัฒนาและทดสอบก็ทำได้ง่ายซึ่งในปัจจุบัน การออกแบบวงจรโดยใช้ FPGA กำลังเป็นที่นิยมและมีแนวโน้มที่จะนำมาใช้งานมากขึ้นเรื่อย

2.3 สถาปัตยกรรม FPGA

ในปัจจุบันมี FPGA อยู่ 4 ชนิดที่วางขายอยู่ในท้องตลาดได้แก่ Symmetrical Array, Row-Based, Hierarchical PLD และ Sea-of-Gates ซึ่งแต่ละชนิดก็มีลักษณะการเชื่อมต่อภายในและการโปรแกรม ที่แตกต่างกันไป นอกจากนี้ในการแบ่งประเภทของ FPGA อาจแบ่งได้ตามเทคโนโลยีที่ใช้ในการโปรแกรม ซึ่งมีอยู่ 2 แบบคือ การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพของตัวชิพ และการโปรแกรม โดยการใช้หน่วยความจำ

รูปที่ 2- 7 แสดง FPGA แต่ละประเภท

2.3.1 การโปรแกรมโดยการทำให้เกิดการเปลี่ยนแปลงทางกายภาพ

- **Fuse** เป็นวิธีการโปรแกรมที่สามารถทำได้เพียงครั้งเดียว ซึ่งหลังจากที่โปรแกรมแล้ว จุดเชื่อม ต่อจะขาดจากกัน
- **Anti Fuse** เป็นวิธีการโปรแกรมที่คล้ายกับแบบ Fuse แต่ต่างกันที่หลังจากทำการโปรแกรมแล้วจุดเชื่อมต่อจะเชื่อมถึงกัน

2.3.2 การโปรแกรมโดยใช้หน่วยความจำ

- **EEPROM Based FPGA**

  FPGA ที่ใช้การโปรแกรมแบบนี้มักเรียกว่า CPLD ซึ่งเทคโนโลยีที่ใช้จะเหมือนกับ EEPROM ทำให้มีความจุของเกทต่ำ โดยทั่วไปจะน้อยกว่า 20,000 เกท แต่ข้อดีของ

EEPROM Based FPGA คือสามารถเก็บข้อมูลที่โปรแกรมลงไปได้โดยไม่จำเป็นต้องมีไฟเลี้ยง และในการโปรแกรมจะใช้ทรานซิสเตอร์ 1 ตัวต่อ 1 บิท ซึ่งการโปรแกรมสามารถทำได้ประมาณ 10,000 ครั้ง

- **SRAM Based FPGA**

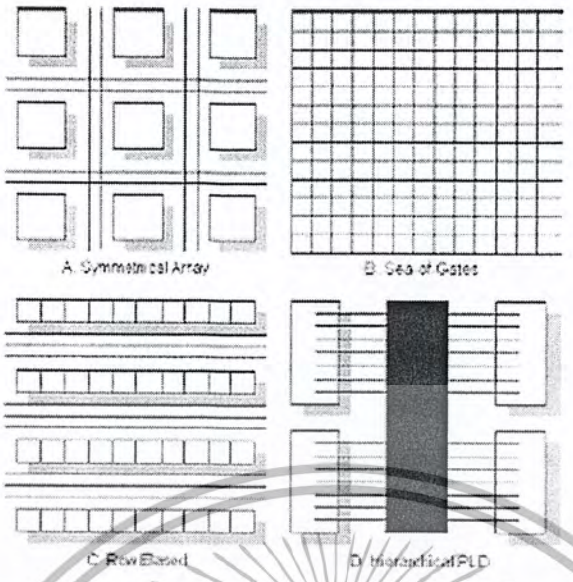
FPGA แบบนี้จะใช้เทคโนโลยีในการโปรแกรมเหมือนกับ SRAM (Static RAM) ทำให้สามารถ โปรแกรมซ้ำได้โดยไม่จำกัดจำนวนครั้ง นอกจากนี้ยังมีความจุของเกทในระดับปานกลางถึงสูงมาก (ประมาณ 10,000 - 1,000,000 เกท) ซึ่งข้อดีของ SRAM Based FPGA คือใช้เวลาในการ โปรแกรมน้อย (ระดับ ns) การโปรแกรมทำได้ง่ายเทียบได้กับการเขียน SRAM ทั่วไป และ เหมาะสำหรับการออกแบบวงจรที่มีความสลับซับซ้อน ส่วนข้อเสียคือไม่สามารถเก็บโปรแกรมใน ภาวะที่ไม่มีไฟเลี้ยงได้ ดังนั้น FPGA ชนิดนี้จึงมักใช้ควบคู่กับ ROM เพื่อเก็บโปรแกรมและทำการ โหลดโปรแกรมลงในตัวชิพในขณะที่เริ่มต้นใช้งาน

2.4 โครงสร้างภายในของ FPGA

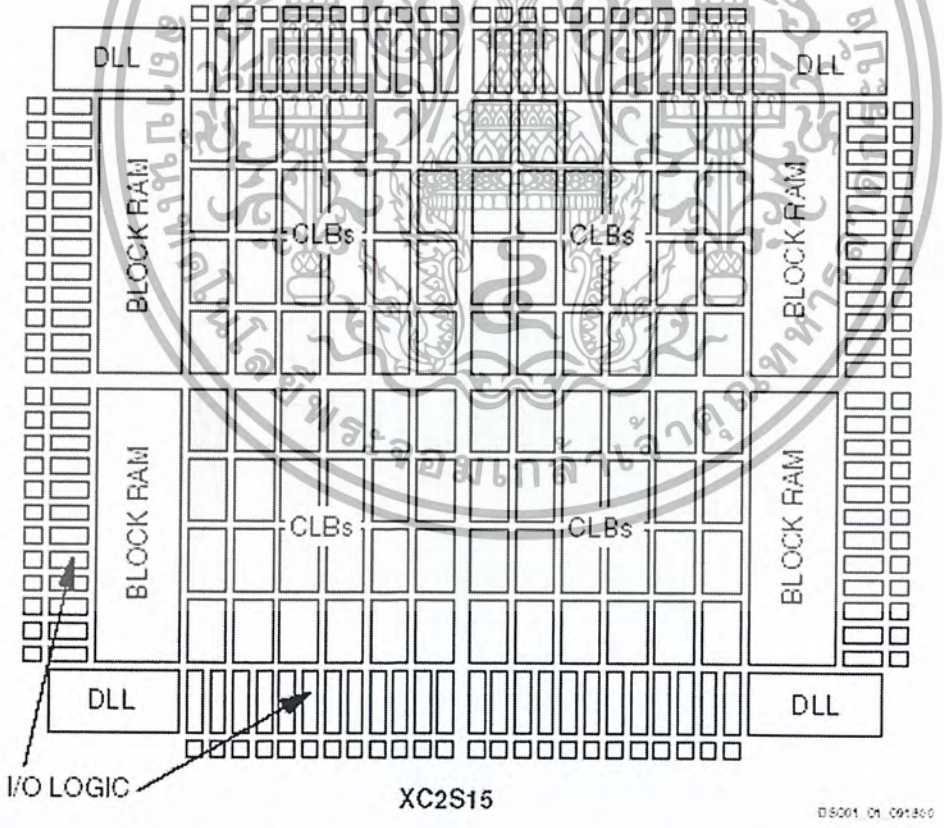
ลักษณะโครงสร้างภายในของ FPGA จะเป็นอะเรย์ของบล็อกลอจิกที่สามารถทำการโปรแกรมได้

FPGA ย่อมาจากคำว่า “Filed Programmable Gate Array” ซึ่งในที่นี้จะกล่าวถึงสถาปัตยกรรมของ FPGA โครงสร้างและหลักการทำงานของ FPGA บริษัท Xilinx ตระกูล XC2S50 ซึ่งใช้สถาปัตยกรรมของ FPGA แบบ Symmetrical Array ดังนี้

2.4.1 สถาปัตยกรรม FPGA แบ่งออกเป็น 4 ประเภท ด้วยกัน คือ Symmetrical Array, Row Based, Hierarchical PLD และ Sea of Gate ดังรูปที่ 2-8 ซึ่ง FPGA ของ Xilinx ตระกูล XC2S50 จะใช้สถาปัตยกรรมที่เรียกว่า Symmetrical Array มี “Static Random Access Memory” (SRAM) เป็นส่วนเก็บข้อมูลที่โปรแกรมรวมอยู่ภายใน Chip และสามารถเพิ่ม PROM เป็นส่วนที่เก็บข้อมูลภายนอกเพื่อให้ FPGA สามารถ Reconfigured ตัวเองได้ภายใน chip ประกอบด้วยบล็อกของลอจิก local routing เรียกว่า “Xilinx Configurable Logic Block”(CLB) โดย แต่ละบล็อกจะต่อถึงกันโดยผ่าน Switch Matrix แสดงดังรูปที่ 2-9



รูปที่ 2-8 แสดงสถาปัตยกรรม FPGA

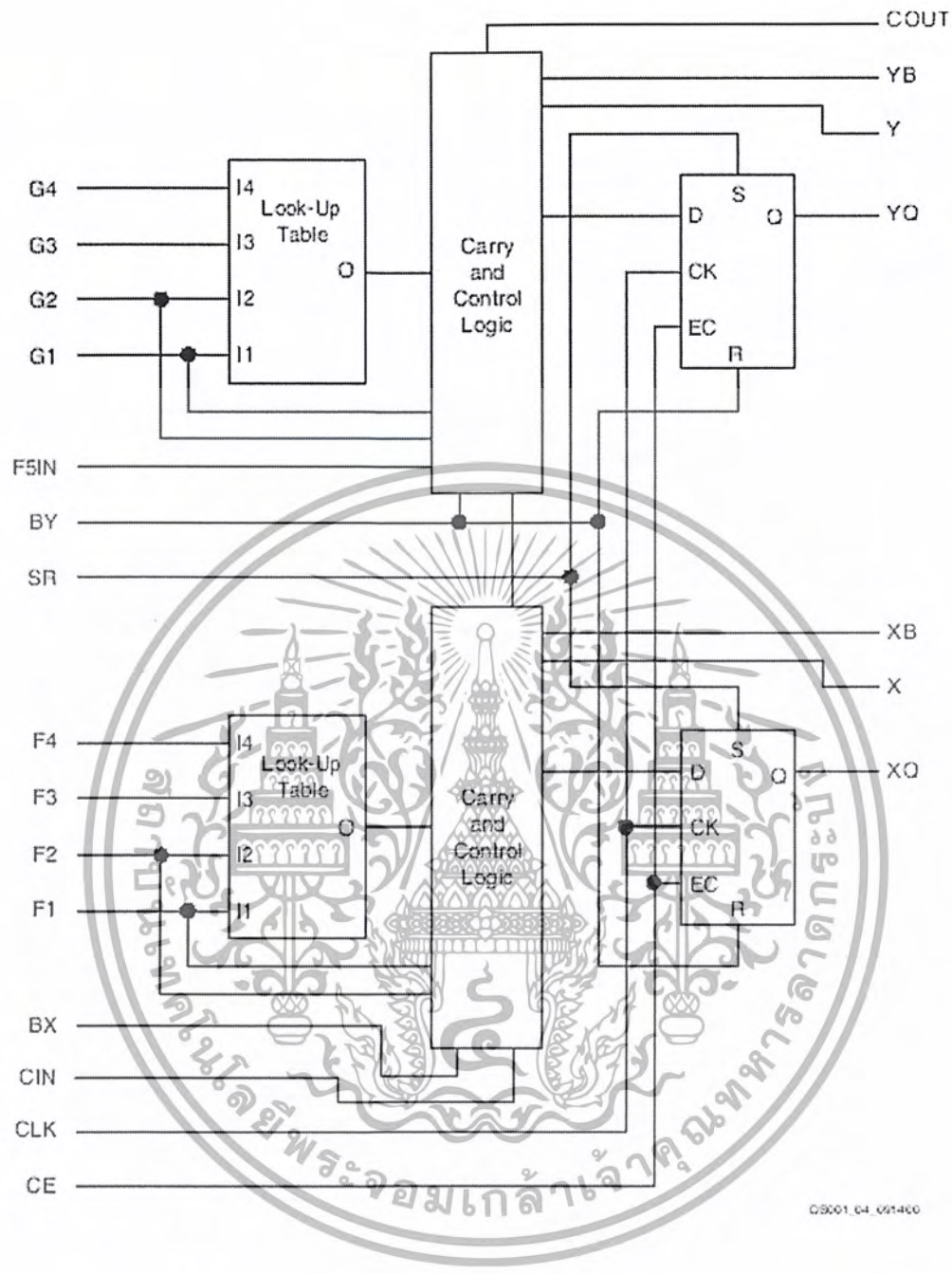


รูปที่ 2-9 แสดง Symmetrical Array FPGA ของ Xilinx ตระกูล XC2Sxx

โครงสร้างและหลักการทำงานของ FPGA ตระกูล XC2S50 สำหรับโครงสร้างของ FPGA แสดง ดังรูปที่ 2.9 ภายใน Logic Block ประกอบด้วย อุปกรณ์ที่สามารถโปรแกรมได้เรียกว่า "Configurable Logic

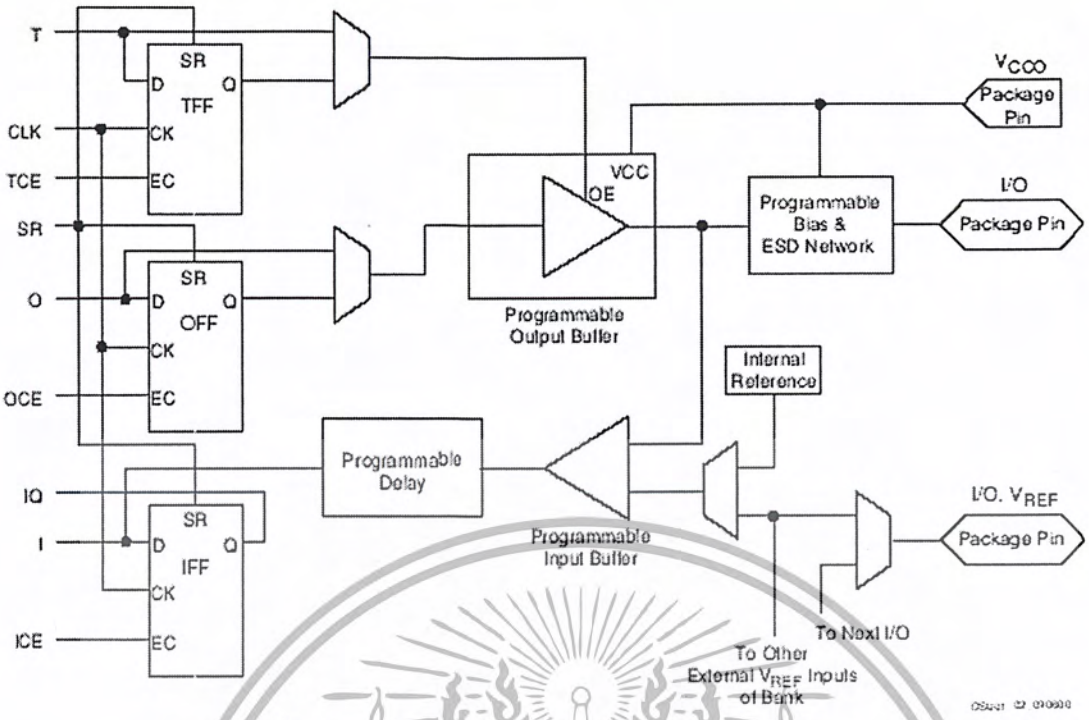
Block"(CLB) มีบล็อกของอินพุทกับเอาท์พุท Array เรียงกันไปตามของ Array ดังรูปที่ 2-9 เรียกว่า "Input / Output Block" (IOB) ซึ่งการต่อภายใน Array ของ CLB และ IOB แต่ละตัวผ่านทางช่องทางเดินไฟ (Wire) โดยมีอุปกรณ์ที่สามารถโปรแกรมไว้วางขั้นอยู่ในแต่ละช่วงดังรูปที่ 2-9 เรียกอุปกรณ์นี้ว่า Switch Matrix

2.4.2 วงจรอย่างง่ายของ CLB ดังรูปที่ 2-10 ใน 1 บล็อกสามารถรับอินพุทได้ 13 สัญญาณ ซึ่งรวมถึงสัญญาณนาฬิกา (K),H1,DIN,Global Set/Reset , EC และมีเอาท์พุท 4 สัญญาณ คือ X,Y,XQ,YQ จากรูปมี lookup table 2 ตัว คือ lookup table for G' และ lookup table for F' ในการรวมสัญญาณ G' F' และ H1 เอาท์พุทของ lookup table ทั้ง 3 จะส่งให้ Multiplexer เป็นตัวเลือกสัญญาณ โดยที่สัญญาณเอาท์พุท x จะมี Multiplexer เลือกสัญญาณระหว่าง F' กับ H1 สัญญาณเอาท์พุท Y จะเลือกสัญญาณระหว่าง G' กับ H' ในส่วนของสัญญาณเอาท์พุท XQ และ YQ มีส่วนควบคุมที่เหมือนกัน คือ จะมี Multiplexer ตัวที่ 1 ในการเลือกสัญญาณระหว่าง DIN, G' ,F' และ H' ส่งสัญญาณที่เลือกแล้วมาเข้าอินพุทของ D Flip-Flop ที่ถูกควบคุม โดยตัวควบคุมการ Set/Reset จากสัญญาณภายนอก ซึ่ง D Flip-Flop จะถูกควบคุมอีกครั้งโดยการเลือกสัญญาณของ Multiplexer ตัวที่ 2 ที่เลือกสัญญาณระหว่าง EC ภายนอกกับลอจิก "1"



รูปที่ 2-10 แสดงวงจรของ Configurable Logic Block (CLB)

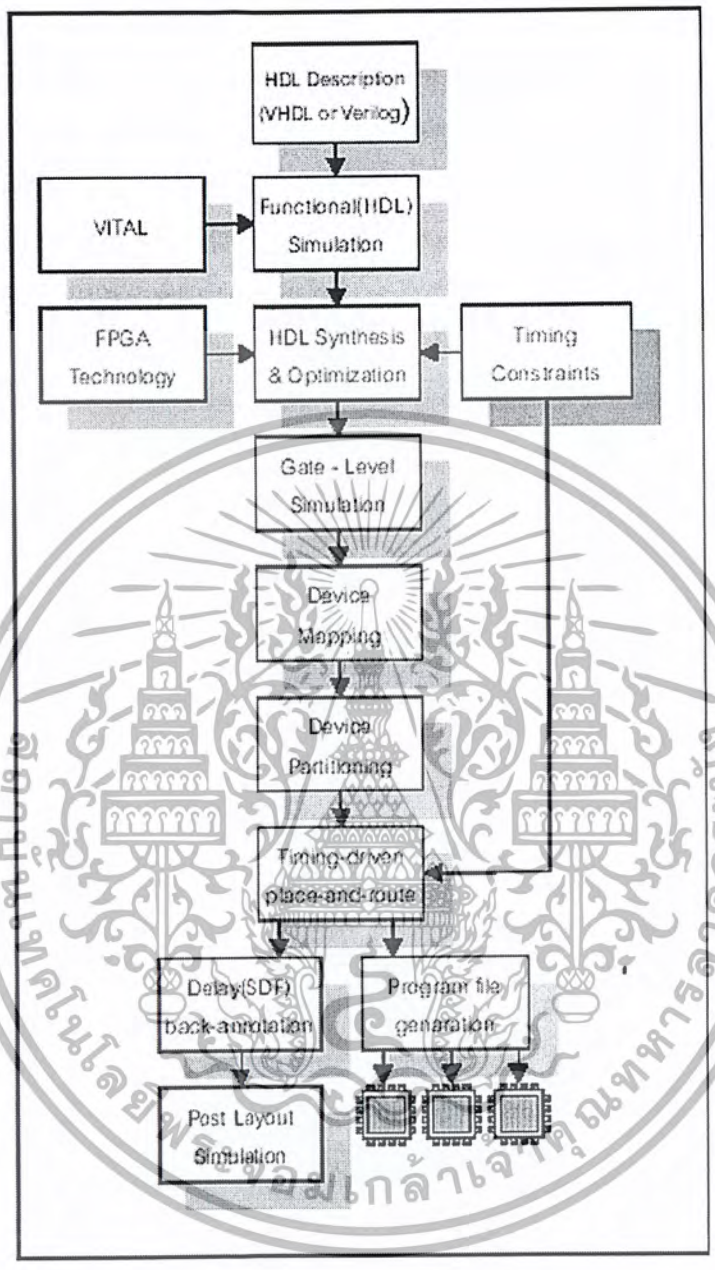
2.4.3 Input / Output Block (IOB) สามารถโปรแกรมการทำงานได้ จากรูปที่ 2-11 แสดงโครงสร้างอย่างง่าย ซึ่งในส่วนเอาท์พุทของ IOB จะส่งเอาท์พุท O ที่เก็บอยู่ใน D Flip-Flop ให้กับขา I/O โดยที่Tri-State(TS) เป็นตัวควบคุมขา I/O ว่าจะให้อินพุทหรือเอาท์พุท



รูปที่ 2-11 แสดงโครงสร้างของ Input / Output Block (IOB)

ในส่วนอินพุทของ IOB สัญญาณที่ขา I/o จะผ่านเข้ามาทาง Input buffer ผ่าน D Flip-Flop ส่งไปยัง Input data 1 และ Input data 2 ของตัว FPGA ต่อไป

2.5 แนวทางการออกแบบวงจรด้วย FPGA (FPGA Design)



รูปที่ 2.12 แสดงขั้นตอนการออกแบบโดยใช้ FPGA

ออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ (Hardware Description Language)

ในการออกแบบวงจรดิจิตอล (digital) นั้น ทำได้โดยการวาดวงจร (Schematic drawing) หรือใช้ภาษาอธิบายฮาร์ดแวร์ (Hardware Description Language) ในขั้นตอนนี้เป็นขั้นตอนที่ไม่แตกต่างกันระหว่างการออกแบบด้วย FPGA และ ASIC ในกรณีที่ใช้ภาษาอธิบายฮาร์ดแวร์ แต่ในกรณีที่ออกแบบโดยวิธีการวาดวงจรจะต่างกันโดยที่การทำวิธีนี้ผู้ทำการออกแบบต้องคำนึงถึงเทคโนโลยีที่จะใช้ซึ่งแต่ละเทคโนโลยีก็มีความแตกต่างกันไป จะเห็นได้ว่าออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ทำได้สะดวกกว่า เพราะการทำวิธีนี้ผู้ออกแบบไม่ต้องมาคำนึงถึงเทคโนโลยีที่จะใช้ และที่สำคัญการออกแบบโดยวิธีนี้สามารถแก้ไข (Model)

หรือเปลี่ยนเทคโนโลยีได้สะดวกกว่าเพราะไม่ต้องวาดวงจรใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยี สำหรับภาษาอธิบายฮาร์ดแวร์ที่ใช้ก็มี VHDL หรือ Verilog ในการเขียน (code) นั้นสิ่งที่ต้องคำนึงถึงคือจะต้องเขียนอย่างไรจึงจะสามารถสังเคราะห์เป็นวงจรได้และให้โมเดลที่มีคุณสมบัติถูกต้องตามข้อกำหนด เพราะลักษณะการเขียนโค้ดจะมีผลโดยตรงกับโมเดลที่ได้ เนื่องจากในการสังเคราะห์วงจรนั้น ซอฟต์แวร์สังเคราะห์วงจร (Synthesis Tools) จะสังเคราะห์วงจรตามโค้ด เช่นในการเขียนโค้ด VHDL อธิบายการทำงานของโมเดลวงจรอันเดียวกัน แต่เขียนโค้ดในลักษณะที่ต่างกัน เมื่อสังเคราะห์จะได้วงจรต่างกัน และจากวงจรที่ต่างกัน เมื่อนำไปทำต้นแบบด้วย FPGA หรือการทำ ASIC แล้วจะได้ไอซี ที่มีคุณสมบัติต่างกันทั้งในด้านของทางขนาดหรือความเร็ว (area and time) ดังนั้นการออกแบบในขั้นตอนนี้ผู้ออกแบบต้องระมัดระวังเป็นพิเศษ ส่วนการที่จะเขียนโค้ดในลักษณะใด เพื่อให้ได้ผลลัพธ์ที่ดีที่สุดนั้นก็ขึ้นอยู่กับประสบการณ์ของผู้ออกแบบ

2.5.1 การสังเคราะห์วงจร (Logic Synthesis)

ในขั้นตอนนี้จะใช้ซอฟต์แวร์สังเคราะห์วงจร (Synthesis tools) ทำการสังเคราะห์โค้ด VHGL หรือ Verilog เพื่อให้ได้เป็นวงจรขึ้นมา ซึ่งทำได้โดยใช้ซอฟต์แวร์อีกเช่นกันแต่ต้องตรวจสอบด้วยว่า ซอฟต์แวร์ นั้น ๆ สนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการใช้หรือไม่ ตัวอย่าง FPGA ที่มีการใช้งานแพร่หลาย เช่น FPGA ของบริษัท Xilinx และบริษัท Altera ในการทำ FPGA นั้นมีซอฟต์แวร์หลายตัวที่สามารถใช้ได้ เช่น Synopsys, Exemplar ฯลฯ ในขั้นตอนนี้ซอฟต์แวร์วงจรจะแปลงโค้ด VHDL และออพติไมซ์ (Optimization) เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ ในการสังเคราะห์วงจรเพื่อทำ FPGA นั้นวงจรระดับเกต (gate-level) ไม่เหมาะสมกับโครงสร้างที่มีอยู่ในอุปกรณ์ FPGA ดังนั้นในการออพติไมซ์นั้นซอฟต์แวร์สังเคราะห์วงจร จะต้องทำการออพติไมซ์ให้ได้เป็นวงจรที่ประกอบด้วยกลุ่มของลอจิก (logic) ที่เหมาะสมกับอุปกรณ์ FPGA นั้นๆ จึงจะทำให้ผลลัพธ์ที่ได้มีประสิทธิภาพ และในขั้นตอนการสังเคราะห์วงจรนี้ผู้ออกแบบสามารถกำหนดข้อบังคับสำหรับโมเดลได้เช่น ข้อบังคับในเรื่องของเวลา (timing constraints) หรือข้อบังคับในเรื่องของพื้นที่ (area) หรือกำหนดชนิดและตำแหน่งของ I/O ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอนออพติไมซ์เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนด ส่วนสำคัญในการออพติไมซ์คือการเทียบ (mapping) โมเดลให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้วงจรที่เหมาะสมกับโครงสร้างสถาปัตยกรรมภายในอุปกรณ์ FPGA ในกรณีของ Xilinx FPGA จะเทียบโดยใช้วิธี LUT(Look Up Table)

เมื่อทำการสังเคราะห์วงจรเสร็จแล้วซอฟต์แวร์สังเคราะห์วงจรก็จะมีการรายงานผลว่าโมเดลที่ออกแบบไปเป็นอย่างไร เช่น มีความหน่วง(delay) เท่าไร ใช้ทรัพยากรต่างFPGA อะไรบ้าง เมื่อมาถึงขั้นตอนนี้ผู้ออกแบบก็จะทราบว่าโมเดลเป็นไรใหม่จนกว่าจะเป็นไปตามที่กำหนด ในขั้นตอนการสังเคราะห์นี้มีเทคนิคต่าง ๆ ที่ผู้ออกแบบนำมาใช้เพื่อให้โมเดลเป็นไปตามข้อบังคับที่กำหนด

2.5.2 การแบ่งวงจร (Partitioning)

ขั้นตอนนี้เป็นการแบ่งวงจรที่ได้จากการสังเคราะห์ เป็นส่วนย่อยๆ สำหรับลงใน CLBs,IBOS หรือองค์ประกอบอื่นๆ ภายในอุปกรณ์ FPGA สำหรับเกณฑ์ที่ใช้ในการแบ่งคือให้แต่ละส่วนที่จะแยกออกจากกันมีจำนวนสัญญาณที่เชื่อมต่อกันระหว่างกันน้อยที่สุดเท่าที่จะทำได้ เพื่อช่วยลดความหนาแน่นในตอนทำการเชื่อมต่อสัญญาณ (routing) ในขั้นตอนนี้จะใช้ซอฟต์แวร์ทำ โดยซอฟต์แวร์จะเทียบส่วนประกอบของวงจรเช่น

เกท (gate), ฟลิปฟล็อป (flip-flop) ลงในทรัพยากรต่างๆ ที่มีอยู่ภายในอุปกรณ์ FPGA (CLBs,IOBs,BUFT และ edge decoder)

หลังจากทำขั้นตอนนี้เสร็จแล้วผู้ออกแบบสามารถที่จะทราบว่าวงจรใช้จำนวนทรัพยากรภายในอุปกรณ์ FPGA ไปเท่าไร ส่วนข้อมูลทางเวลานั้นผู้ออกแบบจะทราบเฉพาะความหน่วงภายในแต่ละส่วนเท่านั้น หรือที่เรียกว่าความหน่วงลอจิก (logic delay) ส่วนซอฟต์แวร์ที่ใช้ในซอฟต์แวร์ที่ใช้ในขั้นตอนนี้เช่น M1 ของ Xilinx ซึ่งซอฟต์แวร์ตัวนี้จะรวมเอาซอฟต์แวร์ ย่อยอื่นๆ อีกเพื่อให้การทำ PPR (Partitioning , Placement & Routing) เป็นไปอย่างต่อเนื่อง

2.5.3 การวางอุปกรณ์(Placement)

ขั้นตอนนี้ก็เป็นการเลือกที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (partitioning) มาแล้วว่าควรจะอยู่ ณ ตำแหน่งไหนในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด เช่น วงจรส่วนไหนควรอยู่ใกล้กันเพื่อจะได้ค้นหาเส้นทาง (Route) ได้ง่ายหรือช่วยลดความหน่วง จะเห็นได้ว่าตำแหน่งภายในอุปกรณ์ FPGA นั้นมีความสำคัญเพราะถ้าจัดวางวงจรลงในตำแหน่งที่ไม่เหมาะสมแล้วจะทำให้ความหน่วงเพิ่มขึ้นหรือ Router ทำการค้นหาเส้นทางสัญญาณได้ไม่หมด

การวางอุปกรณ์ที่ดีควรวางส่วนต่างให้อยู่ใกล้กันโดยเฉพาะส่วนที่มีการเชื่อมต่อสัญญาณด้วยกัน นอกจากนั้นการกำหนดตำแหน่งขา I/O (I/O pin) ตามตำแหน่งขา I/O ของ FPGA บนแผ่น PCB ก็จะมีผลโดยตารางเลยคือซอฟต์แวร์ จะวาง I/O ลงในตำแหน่งที่ผู้ออกแบบกำหนด ซึ่งบางครั้งตำแหน่งที่กำหนดไปไม่เหมาะสม ดังนั้นการกำหนดขา I/O ควรกำหนดตำแหน่งให้เหมาะสม หรือไม่ก็ให้ซอฟต์แวร์จัดการเอง ในกรณีที่ใช้ซอฟต์แวร์ M1 ผู้ออกแบบสามารถแก้ไขตำแหน่งใหม่ได้แต่ควรกระทำด้วย ความระมัดระวังเป็นอย่างยิ่ง แต่วิธีที่ดีที่สุด สำหรับเกณฑ์ที่ใช้ในการตัดสินคือ ความหน่วงที่ได้หลังจากทำการค้นหาเส้นทางแล้ว หรือเรียกว่า routing delay

2.5.4 การเชื่อมต่อสัญญาณ (Routing)

ในขั้นตอนนี้เป็นการเชื่อมต่อสัญญาณระหว่างองค์ประกอบต่างๆ ภายในอุปกรณ์ FPGA เช่นระหว่าง CLBs sหรือระหว่าง CLBs กับ IBOs ขั้นตอนนี้จะทำการต่อเนื่องจากการวางอุปกรณ์ ในกรณีที่ทำการวางอุปกรณ์ไว้ไม่ดี ซอฟต์แวร์ก็จะทำการเชื่อมต่อสัญญาณได้ไม่หมด (เนื่องจากจำนวนทรัพยากร (resource) สำหรับเชื่อมต่อสัญญาณนั้นมีอยู่จำกัด) หรือเกิดความหน่วงเกินค่าที่กำหนดในข้อบังคับ

ผู้ออกแบบสามารถทำขั้นตอนนี้ได้โดยใช้ซอฟต์แวร์เช่น M1 ของบริษัท Xilinx หรือผู้ออกแบบจะทำการเชื่อมต่อสัญญาณด้วยตัวเอง (manual layout) ก็ได้ แต่ทางที่ดีควรใช้ซอฟต์แวร์ทำดีกว่าโดยให้ ทำการค้นหาเส้นทางหลายๆ ครั้งเพื่อหาครั้งที่ดีที่สุด นอกจากนั้นการกำหนดข้อบังคับทางเวลา (timing constraints) จะช่วยให้ผลที่ได้จากการทำเชื่อมต่อสัญญาณดีขึ้นได้

2.5.5 ความหน่วงด้านเวลา (Delay)

ในการทำ FPGA นั้นความหน่วงที่เกิดขึ้นเป็นความหน่วงที่เกิดจากการวางตำแหน่ง (layout) ของอุปกรณ์ ซึ่งผู้ออกแบบไม่สามารถเข้าไปแก้ไขได้ แต่สามารถทำให้มีความหน่วงน้อยที่สุดได้ สำหรับความหน่วงที่เกิดขึ้นนั้นแยกได้เป็นสองประเภทคือ ความหน่วงลอจิก (Logic delay) เป็นความหน่วงภายในองค์ประกอบของอุปกรณ์ FPGA เอง เช่นความหน่วงภายใน CLBs และความหน่วงที่เกิดจากการเชื่อมต่อ

สัญญาณ (Routing delay) เป็นความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณระหว่างองค์ประกอบภายในของอุปกรณ์ FPGA โดยปกติแล้วค่าความหน่วงลอจิกไม่ควรเกิน 50 % ของค่าความหน่วงที่ยอมรับได้ เพราะว่าความหน่วงที่เกิดจากการเชื่อมต่อสัญญาณมักจะมีค่ามากกว่าค่าความหน่วงลอจิกดังนั้นในการวางอุปกรณ์ และเชื่อมต่อสัญญาณ ผู้ออกแบบควรกำหนดข้อบังคับทางเวลาเพื่อให้ ซอฟต์แวร์ได้ทำงานอย่างมีประสิทธิภาพเพิ่มขึ้น และเพื่อให้ได้ผลลัพธ์ที่ดีขึ้น

ค่าความหน่วง ที่ได้หลังจากการวางอุปกรณ์ และเชื่อมต่อสัญญาณแล้วเป็นค่าความหน่วงที่ค่อนข้างแน่นอน ซึ่งผู้ออกแบบสามารถทราบได้ว่าโมเดลที่ออกแบบนั้น เป็นไปตามข้อกำหนดหรือไม่

2.5.6 การโปรแกรมอุปกรณ์ FPGA (Configuration)

หลังจากที่โมเดลผ่านขั้นตอนต่างๆ จนกระทั่งผ่านการทำ PPR (Partitioning , Placement & Routing) แล้วนั้น ถึงตอนนี้ก็สามารถที่จะดาวน์โหลด (download) ลงในอุปกรณ์ FPGA ได้แล้ว ในการดาวน์โหลดนี้ก่อนอื่นต้องแปลงแบบ วงจรรวมที่ได้เป็นข้อมูลวงจร (Configuration data) ซึ่งอยู่ในรูปของบิตสตรีม (bit stream) ก่อน แล้วจึงดาวน์โหลดลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชั่นการทำงานตามโมเดลที่ผู้ออกแบบต้องการ ซึ่งในขั้นตอนนี้จะใช้วิธีที่แตกต่างกันออกไปสำหรับอุปกรณ์ FPGA ของแต่ละบริษัทผู้ผลิตคือ ในกรณีที่เป็นอุปกรณ์ FPGA ชนิดที่ต้องโปรแกรมโดยวิธี SRAM นั้น ในการใช้งานผู้ออกแบบต้องเก็บข้อมูลวงจร (Configuration data) ไว้ในหน่วยความจำประเภท EPROM หรือ serial PROM ด้วยเพื่อจะได้ใช้งานได้สะดวกขึ้น คือในการใช้งานโมเดลครั้งต่อไปไม่ต้องดาวน์โหลดข้อมูลวงจรจากเครื่องคอมพิวเตอร์อีก เพราะมีข้อมูลวงจรเก็บอยู่ในหน่วยความจำอยู่แล้ว แต่ในกรณีที่อุปกรณ์ FPGA เป็นชนิดที่โปรแกรมโดยใช้วิธี EPROM หรือ Antifuse ก็ไม่จำเป็นต้องมีหน่วยความจำไว้สำหรับเก็บข้อมูลวงจร เพราะว่าอุปกรณ์ FPGA ชนิดนี้เมื่อดาวน์โหลดลงไปแล้ว ข้อมูลที่ดาวน์โหลดลงไปก็ยังคงอยู่ในอุปกรณ์ FPGA และครั้งต่อไปก็ใช้งานโมเดลที่ออกแบบไว้ได้เลย

2.5.7 การจำลองการทำงานของวงจร (Simulation)

ในขั้นตอนนี้เป็นขั้นตอนสำคัญอีกขั้นตอนหนึ่ง เพราะเป็นขั้นตอนที่ผู้ออกแบบตรวจสอบฟังก์ชั่นการทำงานของโมเดลว่าถูกต้องหรือไม่ มีข้อผิดพลาดตรงไหนเพื่อจะได้ทำการแก้ไขให้ถูกต้อง ในขั้นตอนนี้จะมีซอฟต์แวร์ที่ใช้สำหรับทำการจำลองการทำงานของวงจร ที่ใช้อยู่เช่น ModelSim ของบริษัท Model Technology ในการจำลองการทำงานของวงจร ควรทำทุกครั้งหลังจากที่มีการำแต่ละขั้นตอนหลักเสร็จแล้วเพื่อจะได้ทราบว่าข้อผิดพลาดของโมเดลเกิดจากขั้นตอนไหนจะได้แก้ไขข้อผิดพลาดตรงขั้นนั้นๆ ได้เลย ไม่ต้องมาคอยตรวจหาขั้นตอนที่ทำให้เกิดการผิดพลาด นั่นคือการทำการจำลองการทำงานของวงจร ต้องทำหลังจากการเขียนโค้ด, การสังเคราะห์วงจร และการทำ PPR การจำลองการทำงานหลังจากที่เขียนโค้ดเสร็จแล้วนั้น ผู้ออกแบบสามารถทราบได้แค่โมเดลทำงานถูกต้องหรือไม่เท่านั้น (functional test) แต่ยังไม่สามารถตรวจสอบการทำงานในเชิงเวลาได้ถูกต้อง ในการจำลองการทำงานของวงจรหลังจากที่สังเคราะห์เป็นวงจรแล้วเพื่อตรวจสอบว่าฟังก์ชั่นการทำงานยังคงถูกต้องอยู่หรือไม่ และค่าความหน่วงที่เกิดขึ้นเป็นไปตามข้อบังคับหรือไม่ มีข้อผิดพลาดหรือไม่ ถ้ามีก็จะได้แก้ไขให้ถูกต้อง ในการจำลองการทำงานของวงจรหลังจากที่ทำการวางอุปกรณ์ และเชื่อมต่อสัญญาณ(Post Layout Simulation) แล้วก็มีความสำคัญเช่นกันเพราะผลที่ได้จากการจำลองการทำงานของวงจรในตอนนี้จะเป็นผลลัพธ์ของโมเดลเลย ซึ่งผู้ออกแบบนอกจากจะตรวจสอบ

ฟังก์ชั่นการทำงานแล้วยังต้องตรวจสอบคุณสมบัติอื่นๆ เช่น ความหน่วงที่ได้จากการทำ PPR ในรูปแบบค่าความหน่วงมาตรฐาน (Standard Delay Format :SDF) ว่าตรงตามที่กำหนดไว้หรือไม่ หรือตรวจสอบว่าแบบวงจรรวมสามารถใช้งานที่ความถี่สูงสุดเท่าไร นั่นเอง ในการจำลองการทำงานของวงจรควรใช้ ซอฟต์แวร์ตัวเดียวกันตลอดเพื่อจะได้เปรียบเทียบผลที่ได้จากขั้นตอนต่างๆ

2.5.8 ซอฟต์แวร์ FPGA (FPGA Design Tools)

จะเห็นได้ว่าการออกแบบเพื่อทำ FPGA นั้นทำได้สะดวกกว่าการทำ ASIC มากเพราะใช้เวลาน้อยกว่ามากด้วย ส่วนสำคัญที่ใช้ในการทำ FPGA คือ ซอฟต์แวร์ที่ใช้ตั้งแต่เขียนโค้ดอธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลดลงในอุปกรณ์ FPGA ซึ่งซอฟต์แวร์ที่ต้องเป็น ซอฟต์แวร์ที่ใช้ทำงานต่อเนื่องกันได้ คือ ซอฟต์แวร์ที่ใช้ทำการสังเคราะห์วงจรกับ ซอฟต์แวร์ ที่ใช้ทำ PPR เช่น Leonardo กับ M1 หรือ Synopsys กับ M1 นั่นคือ M1 ต้องสามารถทำ PPR วงจรที่สังเคราะห์โดยซอฟต์แวร์ Leonardo หรือ Synopsys ได้ สำหรับซอฟต์แวร์ที่ใช้ทำการจำลองการทำงานของวงจรนั้น ต้องสามารถใช้งานได้ต่อเนื่องกับซอฟต์แวร์ทั้งระบบเพราะโมเดลที่ได้จากการทำขั้นตอนต่างๆ (ด้วยซอฟต์แวร์ต่างๆ) ต้องเอามาจำลองการทำงานได้ และในการจำลองการทำงานของวงจร ควรใช้ ซอฟต์แวร์ตัวเดียวกันตลอดทั้งระบบ เพื่อจะได้เปรียบเทียบผลได้ง่าย ในอดีตซอฟต์แวร์ส่วนใหญ่จะใช้งานอยู่บนคอมพิวเตอร์สมรรถนะสูงอย่างเวิร์กสเตชั่น (Workstation) ในปัจจุบันมีการพัฒนาซอฟต์แวร์ที่ใช้งานบนพีซี (PC) มากขึ้น ซึ่งสามารถลดค่าใช้จ่ายในด้านอุปกรณ์คอมพิวเตอร์ ส่วนรายชื่อตัวอย่างซอฟต์แวร์และบริษัทผู้ผลิตที่ใช้ในการออกแบบแต่ละขั้นตอนแสดงดังตารางที่ 2.3

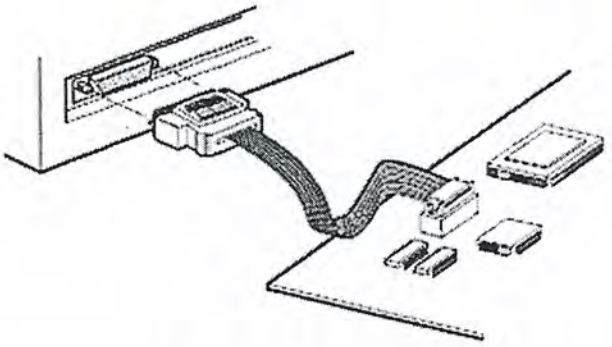
ขั้นตอนการออกแบบ	ซอฟต์แวร์ที่ใช้	ผู้ผลิต
HDL Simulation	ModelSim/VHDL	Model Technology
HDL Synthesis & Optimization	Leonardo	Exemplar logic
Partitioning,Placement and Routing	Foundation	Xilinx

ตารางที่ 2-1 แสดงตัวอย่างรายชื่อซอฟต์แวร์และผู้ผลิตที่ใช้ในการออกแบบแต่ละขั้นตอน

2.6 ปัจจัยที่ทำให้การออกแบบ FPGA ทำได้ง่ายและสะดวกรวดเร็ว

- ผู้ออกแบบไม่จำเป็นต้องทราบถึงโครงสร้างภายในของตัวชิพ เพียงแต่มีความรู้เกี่ยวกับขั้นตอนการออก แบบลอจิกก็เพียงพอแล้ว ต่างกับการใช้ไมโครโปรเซสเซอร์ซึ่งจำเป็นต้องศึกษาโครงสร้างภายในรวมถึง ภาษา Assembly ของไมโครโปรเซสเซอร์ตัวนั้นด้วย
- มีการออกแบบโดยใช้ภาษาในการอธิบายการทำงานของวงจร หรือ HDL (Hardware Description Language)เป็นเครื่องมือในการออกแบบ ซึ่งเป็นวิธีการที่มีความยืดหยุ่นสูง ทำได้รวดเร็ว และไม่จำเป็น ต้องทราบถึงลักษณะของวงจรที่ต้องการว่าจะเชื่อมต่อกันอย่างไร เพียงแต่กำหนดลักษณะการทำงานให้มัน จากนั้นตัวซอฟต์แวร์จะทำ Synthesis and Optimize ให้ทั้งหมด นอกจากนี้ภาษาที่ใช้ยังเป็นมาตรฐาน เดียวกันสามารถใช้ได้กับชิพทุกตัวและทุกบริษัท
- การโปรแกรมสามารถทำได้เองและใช้เวลาไม่นาน เพียงแค่ส่งข้อมูลผ่านสายดาวน์โหลดทางพอร์ตของ คอมพิวเตอร์ก็สามารถโปรแกรมตัวชิพขณะที่อยู่ในระบบได้ โดยไม่จำเป็นต้องถอดมาโปรแกรม

ข้างนอก ดังรูปที่ 2-13 และที่สำคัญสามารถโปรแกรมได้หลายครั้ง จึงทำให้ง่ายในการแก้ไขและพัฒนาโดยไม่ต้องเสีย ค่าใช้จ่ายเพิ่มแต่อย่างใด



รูปที่ 2-13 การโปรแกรมลงในชิพ

การออกแบบโดยใช้ภาษาอธิบายพฤติกรรมของฮาร์ดแวร์

ในการออกแบบวงจรดิจิตอลนั้นสามารถทำได้โดยการวาดวงจร (Schematic) หรือใช้ภาษาอธิบายพฤติกรรม (Hardware Description Language) ของฮาร์ดแวร์ จากที่ได้กล่าวไปแล้วในข้างต้น ในกรณีของการออก แบบวงจรด้วย ASIC ชนิด Full Custom ผู้ออกแบบจะต้องเขียนวงจรด้วย Schematic จากนั้นจะนำวงจรที่ออกแบบไว้ไปทำการจำลองการทำงาน (Simulate) ซึ่งหากผลออกมาเป็นที่พอใจก็จะต้อง Layout เป็นชั้นสาร และในการออกแบบ ASIC ชนิดนี้ผู้ออกแบบจำเป็นจะต้องทราบถึงเทคโนโลยีที่ใช้ในการสร้างด้วย หลังจากได้ Layout ที่สมบูรณ์แล้วจึงจะส่งไปเข้ากระบวนการสร้างไอซีหรือ Fabrication เพื่อสร้างเป็นชิพไอซีออกมา แต่ในการออกแบบวงจรด้วย FPGA โดยการใช้ Schematic หรือใช้ภาษาอธิบายการทำงานของวงจรจะทำ ได้สะดวกกว่า เนื่องจากวิธีการนี้ผู้ออกแบบไม่จำเป็นต้องคำนึงถึงเทคโนโลยีที่จะใช้สร้างไอซีและที่สำคัญ การออก แบบโดยวิธีนี้สามารถแก้ไขโมเดล (Model) หรือเปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่า เพราะไม่ต้องวาดวงจร ใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยีสำหรับภาษาที่ใช้ สำหรับอธิบายพฤติกรรมของฮาร์ดแวร์ที่ใช้กันก็มี VHDL, AHDL และ Verilog เป็นต้น

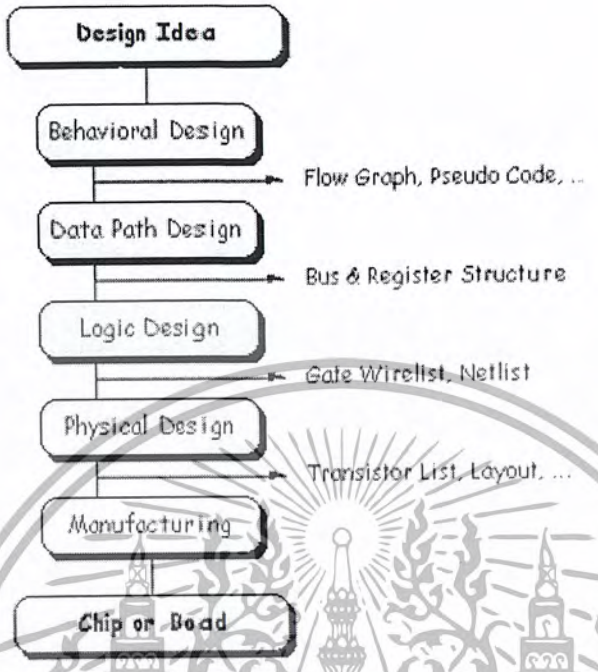
2.7 การเขียนภาษา VHDL

ความซับซ้อนและขนาดของระบบดิจิตอลในปัจจุบันได้เพิ่มมากขึ้นทุกขณะ ส่งผลให้มีการนำคอมพิวเตอร์เพื่อช่วยใน การออกแบบหรือ CAD มาใช้ในขบวนการออกแบบฮาร์ดแวร์เพิ่มขึ้นเช่นกัน อีกทั้งอุปกรณ์และวิธีการ ออกแบบใหม่ๆ ก็ถูกพัฒนาขึ้นมาเพื่อช่วยอำนวยความสะดวกให้กับนักออกแบบมากขึ้นด้วย สำหรับภาษาบรรยายอุปกรณ์ฮาร์ดแวร์ (HDL : Hardware Description Language) ก็เป็นเครื่องมืออย่างหนึ่งที่ได้รับการพัฒนามาอย่างต่อเนื่อง เพื่อช่วยให้การปรับปรุงขบวนการออกแบบระบบดิจิตอลเป็นไปอย่างมีประสิทธิภาพ

2.7.1 การออกแบบระบบดิจิตอล

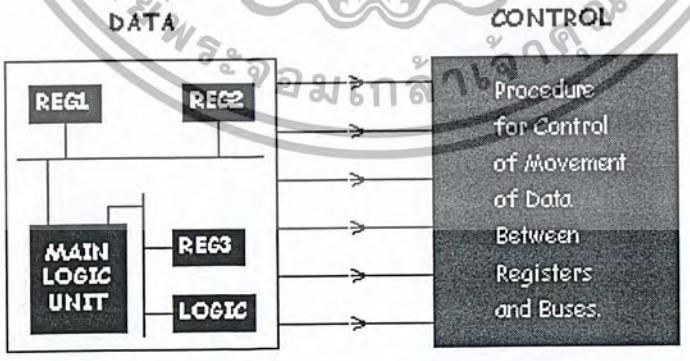
ในการออกแบบระบบดิจิตอล เริ่มตั้งแต่การกำหนดแนวความคิดเบื้องต้นจนกระทั่งได้ออกมาเป็นอุปกรณ์ฮาร์ดแวร์ ที่ใช้งานได้จะต้องผ่านขั้นตอนต่างๆ มากมาย และในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์ในแต่ละขั้น ก่อนเข้าสู่กระบวนการออกแบบในขั้นต่อไป รูปที่ 2-14 แสดงขั้นตอนปกติที่ใช้ในการออกแบบระบบดิจิตอลทั่วไป ขั้น แรกผู้ออกแบบจะกำหนดแนวความคิดในการออกแบบแล้วทำการ

พัฒนาให้สามารถนำมาใช้ได้อย่างสมบรูณ์ ซึ่งภาย ในขั้นตอนนี้ผู้ออกแบบจำเป็นต้องสร้างรูปแบบระบบในเชิงพฤติกรรมขึ้นมาตรวจสอบซึ่งอาจจะเป็นผังงานแสดงแบบหรือ รหัสคำสั่งเทียม (Pseudo code) ก็ได้



รูปที่ 2-14 แสดงขั้นตอนการออกแบบระบบดิจิตอล

ขั้นตอนต่อไปเป็นการออกแบบระบบเส้นทางของข้อมูล ผู้ออกแบบจะกำหนดส่วนประกอบของรีจิสเตอร์และวงจรลอจิก ที่จำเป็นทั้งหมดเพื่อนำมาประกอบเป็นระบบที่สมบรูณ์ โดยแต่ละองค์ประกอบสามารถเชื่อมต่อกันด้วยบัสหนึ่งหรือสอง ทิศทาง (Unidirectional or Bidirectional Bus) ส่วนกระบวนการในการควบคุมการเคลื่อนย้ายข้อมูลระหว่าง รีจิสเตอร์และวงจรลอจิกจะขึ้นอยู่กับพฤติกรรมของระบบที่กำหนดไว้ดังรูปที่ 2-15



รูปที่ 2-15 การออกแบบระบบเส้นทางของข้อมูล

ขั้นตอนถัดมาเป็นการออกแบบวงจรลอจิก ซึ่งจะเกี่ยวข้องกับการนำเกทดิจิตอลพื้นฐานและฟลิปฟล็อป (flip-flop) มาประกอบเป็นอุปกรณ์ย่อยต่างๆ เช่น รีจิสเตอร์เก็บข้อมูล บัสวงจรลอจิก และส่วนควบคุมฮาร์ดแวร์ ซึ่งผลลัพธ์ ที่ได้ในขั้นตอนนี้จะเป็นเครือข่ายของการโยงใยระหว่างเกทและ ฟลิปฟล็อป

นั่นเองการออกแบบในขั้นตอนถัดไป เป็นการเปลี่ยนเครือข่ายการโยงใยที่ได้จากขั้นตอนที่แล้วให้เป็นลำดับของทรานซิสเตอร์ (Transistor List) และ Layout ซึ่งขั้นตอนนี้จะเกี่ยวข้องโดยตรงกับการจัดวางทรานซิสเตอร์หรือไลบรารีเซลล์เพื่อ แทนเกทและ ฟลิปฟล๊อปต่างๆและในขั้นตอนสุดท้ายจะเป็นการส่งระบบที่ออกแบบไว้ไปทำการเจือสารที่โรงงานเพื่อผลิตออกมาเป็น วงจรรวมในที่สุด

2.7.2 ประวัติความเป็นมาของภาษา VHDL

VHDL ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC : Very High Speed Integrated Circuit) เป็นภาษาโปรแกรมระดับสูง (High Level Language) ที่ใช้สำหรับการออกแบบฮาร์ดแวร์ในระบบดิจิตอล ตัวของภาษาสามารถบรรยายพฤติกรรมการทำงานในรูปของลำดับชั้น (Hierarchy) และสามารถเขียนได้หลายรูปแบบ ด้วยเหตุผลนี้จึงทำให้ภาษา VHDL เป็นเครื่องมือที่ใช้ออกแบบตั้งแต่ขั้นตอนบนสุด คือ แนวความคิดที่จะแก้ปัญหา ลงไปทีละขั้นจนถึงขั้นตอนของการสร้างวงจรจริง และตัวภาษาก็เปิดโอกาสให้วิศวกร ได้พัฒนาและจำลองการทำงานของรูปแบบฟังก์ชันการทำงานของวงจรอย่างสังเขป โดยยังไม่ต้องคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรจริง นอกจากนั้น VHDL ยังเป็นภาษาที่สนับสนุนลักษณะต่างๆ ของระบบดิจิตอลที่มี ความซับซ้อนได้ทั้งหมด ดังนั้น VHDL จึงเป็นภาษาที่น่าสนใจในการศึกษาและนำไปใช้งานเป็นอย่างยิ่ง วิวัฒนาการของภาษา VHDL เริ่มต้นประมาณปี ค.ศ. 1981 เมื่อกระทรวงกลาโหมสหรัฐอเมริกา หรือ DoD (Department of Defense) ได้พยายามปรับปรุงอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการทางทหาร ให้มีความทันสมัยมากขึ้น ประกอบกับเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์มีการพัฒนาไปอย่างรวดเร็วดังจะเห็นได้ จากการนำวงจรดิจิตอลหลายๆ วงจรมาทำการผลิตอยู่บนแผ่นซิลิกอนที่มีพื้นที่เพียง 1 - 2 ตารางเซนติเมตรเท่านั้น ซึ่ง เป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้นตลอดจนความน่าเชื่อถือ ในการทำงานและความคงทนต่อสภาพ แวดล้อมสูง แต่เนื่องจากในขณะนั้นขั้นตอนของการออกแบบ การผลิต และการตรวจสอบวงจรต้นแบบ เป็นขบวนการที่ ต้องใช้วิศวกรและเวลาในดำเนินการมาก ฉะนั้นทาง DoD จึงจัดตั้งโครงการขึ้นมาเพื่อศึกษาวิธีการที่ช่วยในการพัฒนา วงจรอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่งวงจรระบบดิจิตอล ให้สามารถนำไปผลิตได้เร็วขึ้น ซึ่งโครงการดังกล่าวมีชื่อว่า "Very High Speed Integrated Circuits" หรือ VHSIC โดยในระยะแรกนั้นโครงการนี้ถือเป็นความลับทางด้านความมั่งคงของประเทศ และอยู่ภายใต้ความควบคุมดูแลของ United States International Traffic and Arms Regulations (ITAR) สำหรับมาตรฐานของภาษาที่ใช้บรรยาย พฤติกรรมวงจรหรือฮาร์ดแวร์ของระบบ สำหรับโครงการ VHSIC ที่ DoDได้ให้ไว้สามารถสรุปได้ดังนี้

- ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถเข้าใจได้ทั้งมนุษย์และเครื่อง คอมพิวเตอร์โดยไม่ต้องมีการแปลหรือเปลี่ยนแปลงอีก
- สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้
- ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร

ฉะนั้นภาษาดังกล่าวนี้จึงจัดเป็นภาษาโปรแกรมระดับสูง เช่นเดียวกับภาษาปาสคาล หรือภาษาซี ซึ่งในทางวิศวกรรม ภาษาที่ใช้ในการออกแบบฮาร์ดแวร์นี้เรียกว่า "Hardware Description Language" หรือ HDL ในตอนเริ่มแรกนั้น DoD ได้มอบหมายให้บริษัทไอบีเอ็ม เท็กซัสอินสตูเมนท์ และอินเตอร์เมทริกซ์ เป็นผู้ศึกษาและพัฒนา โครงการ ซึ่งการดำเนินงานเป็นไปอย่างต่อเนื่อง จนกระทั่งในปี ค.ศ.1985 ทาง ITAR ได้ยกเลิก

ข้อจำกัดในการถ่ายทอด เทคโนโลยีทางทหารออกจากโครงการนี้ ดังนั้นภาษา VHDL จึงเริ่มเป็นที่รู้จักกันโดยทั่วไป และประมาณปี ค.ศ. 1987 IEEE ได้ทำการกำหนดมาตรฐานของภาษานี้เป็น IEEE 1076-1987 และมีชื่อเรียกว่า VHDL ซึ่งมาตรฐานนี้ได้รับ การปรับปรุงจนเป็นมาตรฐาน IEEE 1076-1993 หรือ VHDL 1993 เนื่องจากในขณะนั้น DoD เป็นลูกค้ารายใหญ่ ของอุตสาหกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์ ดังนั้นจึงมีผู้รับโครงการต่างๆ จาก DoD ไปดำเนินการวิจัยและพัฒนา เป็นจำนวนมาก และเพื่อให้ทุกโครงการอยู่ในมาตรฐานเดียวกันหมด ดังนั้นทาง DoD จึงได้กำหนดว่า ทุกๆ โครงการต้อง เขียนอยู่ในรูปของภาษา VHDL เท่านั้น ซึ่งทำให้ DoD สามารถนำโครงการเหล่านี้ไปจำลองกับเครื่องคอมพิวเตอร์ได้ หลายๆระบบ

ข้อกำหนดDoD ได้ตั้งข้อกำหนดสำหรับภาษา VHDL ในเดือนมกราคมปี ค.ศ.1983 ไว้ดังนี้

- ลักษณะทั่วไป

DoD ได้กำหนดให้ VHDL เป็นภาษาสำหรับการออกแบบและบรรยายของฮาร์ดแวร์ ซึ่งหมายถึงความสามารถ ในการอธิบายและออกแบบในระดับสูง การจำลอง (Simulation) การสังเคราะห์ (Synthesis) และการทดสอบ (Testing) นอกจากนั้น VHDL ยังถูกกำหนดไว้สำหรับการบรรยายฮาร์ดแวร์ตั้งแต่ระดับบนซึ่งก็คือระบบจนถึง ระดับเกทอีกด้วย เนื่องจากในการทำงานของระบบดิจิตอลนั้น ทุกๆ องค์ประกอบภายในระบบไม่ว่าเล็กหรือใหญ่ จะทำงานไปพร้อมๆ กัน ซึ่งในเรื่องของความพร้อมเพรียงในการทำงานนี้ก็ถือเป็นข้อกำหนดที่สำคัญอย่างหนึ่งของ VHDL ด้วยเช่นกัน (สำหรับในภาษาที่ใช้ในการบรรยายฮาร์ดแวร์นั้นความพร้อมเพรียงจะหมายถึงทุกๆ คำสั่ง องค์ประกอบ เกทหรือวงจรลอจิกจะถูกนำมาปฏิบัติทั้งหมด ดังนั้นในที่สุดแล้วก็จะดูเหมือนว่าได้มีการปฏิบัติไป พร้อมๆ กัน)

- สนับสนุนการออกแบบแบบลำดับขั้น

การออกแบบแบบลำดับขั้นเป็นลักษณะที่สำคัญอย่างหนึ่งสำหรับการออกแบบระบบที่มีหลายๆ ระดับ โดยในการ ออกแบบจะประกอบด้วยส่วนการบรรยายการเชื่อมต่อ และส่วนการบรรยายหน้าที่การทำงาน ซึ่งหน้าที่การทำงาน ของระบบสามารถกำหนดได้ด้วยตัวเอง หรืออาจถูกกำหนดโดยโครงสร้างที่ประกอบด้วยองค์ประกอบย่อยๆ ลง ไปได้เช่นกัน แต่ที่ระดับล่างสุด องค์ประกอบต้องถูกบรรยายหน้าที่การทำงานด้วยตัวมันเอง และไม่สามารถกำหนด การทำงานโดยลักษณะแบบโครงสร้างได้

- ไลบรารี

VHDL ได้สนับสนุนการมีไลบรารีเพื่อระบบการจัดการที่ดี ผู้ออกแบบสามารถกำหนดลักษณะและการทำงานของ อุปกรณ์พื้นฐานไว้ในระบบไลบรารี หรือจะใช้ไลบรารีที่ระบบได้จัดเตรียมไว้แล้วก็ได้ โมเดลและการบรรยายที่ถูก ต้องควรจัดเก็บไว้ในไลบรารีหลังจากที่ได้ผ่านการคอมไพล์เรียบร้อยแล้วเพื่อให้ผู้ออกแบบคนอื่นๆ สามารถนำไป ใช้ได้ด้วย

- ลำดับคำสั่ง

แม้ว่าการปฏิบัติคำสั่งหรือกระบวนการโดยพร้อมเพรียงกันจะเป็นคุณสมบัติที่สำคัญของ VHDL ก็ตาม ตัวภาษา เองก็ยังมีการจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งไว้ให้ด้วย เมื่อผู้ออกแบบได้กำหนดหน้าที่และองค์ประกอบ ที่ทำงานพร้อมกันของระบบไว้เรียบร้อยแล้ว ผู้ออกแบบยังสามารถบรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดภายใน ของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรมที่ประกอบด้วยโครงสร้างแบบ case, if - then - else และ loop ทั่วๆ ไปได้ การบรรยายแบบลำดับคำสั่งทำให้การ

ออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้ สะดวกและง่ายขึ้น อย่างไรก็ตามโครงสร้างทั้งหมดของ VHDL ก็ยังคงเป็นการทำงานแบบพร้อมเพรียงกันเช่นเดิม

- **การกำหนดคุณสมบัติ**

นอกจากการกำหนดอินพุทและเอาท์พุทแล้ว เงื่อนไขอื่นๆ ก็มีผลต่อการปฏิบัติหน้าที่ของอุปกรณ์ฮาร์ดแวร์ด้วยเช่นกัน โดยสิ่งนี้รวมถึงสภาพแวดล้อมและลักษณะทางกายภาพของอุปกรณ์นั้นๆ ด้วย ซึ่งภาษาสำหรับการออกแบบที่ดีควร ให้ผู้ออกแบบกำหนดคุณสมบัติของอุปกรณ์ที่ใช้ได้ด้วย เช่น สามารถกำหนดขนาด ลักษณะทางกายภาพเวลา โหลด และเงื่อนไขทางสภาพแวดล้อมอื่นๆ ซึ่งความสามารถในการกำหนดคุณสมบัตินี้ก็เป็นส่วนหนึ่งที่มีอยู่ในภาษา VHDL ด้วยเช่นกัน

- ชนิดของข้อมูล

VHDL สามารถกำหนดชนิดของข้อมูลไม่เพียงแต่ชนิด BIT และ BOOLEAN เท่านั้น แต่ยังสามารถกำหนดชนิด ของข้อมูลเป็นจำนวนเต็ม จำนวนจริง จุดทศนิยม และชนิดลำดับการนับ (Enumerate Type) หรือแม้แต่ชนิดของ ข้อมูลที่ผู้ออกแบบกำหนดขึ้นมาเองก็ได้

- โปรแกรมย่อย

ความสามารถในการใช้ฟังก์ชันและโพรซีเจอร์ (Procedure) ก็เป็นข้อกำหนดอีกอย่างหนึ่งใน VHDL ซึ่งผู้ออกแบบ สามารถนำโปรแกรมย่อยมาใช้ในการเปลี่ยนแปลงชนิดของข้อมูล การกำหนดหน่วยของลอจิก การกำหนดตัวกระทำต่างๆ หรือหน้าที่อื่นๆ ตามที่ต้องการได้เช่นเดียวกับการเขียนโปรแกรมทั่วไป

- การควบคุมเวลา

VHDL อนุญาตให้ผู้ออกแบบสามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ตามต้องการ การตรวจสอบ การออกแบบเกทหรือการหน่วงเวลาก็สามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอย เหตุการณ์ (Event) นอกจากนี้ก็ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้อีกด้วย

- การกำหนดแบบโครงสร้าง

การกำหนดโครงสร้างขององค์ประกอบต่างๆ สามารถกระทำได้ในทุกระดับของการออกแบบ โดยการกำหนดโครงสร้างขององค์ประกอบร่วมที่เกิดจากองค์ประกอบย่อยซึ่งแตกต่างกันหรือ เหมือนกันก็เป็นข้อกำหนดอย่างหนึ่งของ VHDL เช่นกัน

2.7.3 องค์ประกอบพื้นฐานของ VHDL

รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบของ VHDL จะประกอบไปด้วยส่วนกำหนดการเชื่อมต่อ (Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม (Architecture) ดังแสดงในรูปที่ 2-16 โดยในการบรรยายการเชื่อมต่อจะขึ้น ต้นด้วยคำว่า ENTITY แล้วตามด้วยชื่อขององค์ประกอบจากนั้นตามด้วยคำว่า IS และถัดมาจะเป็นการบรรยายถึงพอร์ต การติดต่อ อินพุท - เอาท์พุท ขององค์ประกอบ ส่วนลักษณะภายนอกอื่น ๆ เช่น เวลา อุณหภูมิก็สามารถรวมเข้าไปในส่วนนี้ ได้เช่นกัน ในส่วนของการกำหนดลักษณะเชิงสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้ บรรยายหน้าที่การทำงานขององค์ประกอบ โดยหน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาณอินพุท เอาท์พุทและพารามิเตอร์ อื่นๆ ที่ได้กำหนด

ไว้ในส่วนของการเชื่อมต่อดังรูปที่ 2-16 และสำหรับการบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังจาก คำว่า BEGIN เป็นต้นไป

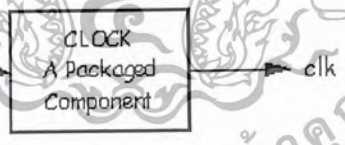
```
ENTITY component_name IS
    Input and output ports
    Physical and other parameters
END [component_name] ;

ARCHITECTURE identifier OF component_name IS
    [declartion]
BEGIN
    specification of the functionality of the component
    in terms of its input lines and as influenced
    by physical and other parameters
END [identifier];
```

รูปที่ 2-16 การกำหนดการเชื่อมต่อและสถาปัตยกรรม

- การกำหนดการเชื่อมต่อ

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบ โดยในระดับนี้ต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบ ภายนอกอื่นๆ ดังตัวอย่างในรูปที่ 2-17 ซึ่งเป็นบล็อกไดอะแกรม และการบรรยายการเชื่อมต่อขององค์ประกอบสำหรับตัวจ่าย สัญญาณนาฬิกา ในบรรทัดแรกของการบรรยายการเชื่อมต่อเป็นการกำหนดชื่อขององค์ประกอบซึ่งกำหนดเป็น clock_ component ตามด้วยคำว่า PORT และชื่อของพอร์ตอยู่ภายในวงเล็บ ส่วน IN และ OUT เป็นการกำหนด โหนดของสัญญาณให้เป็นอินพุทหรือเอาท์พุท และ BIT เป็นการแสดงชนิดของข้อมูล




```
ENTITY clock_component IS
    PORT (en : IN BIT;ck : OUT BIT)
END clock_name;
```

รูปที่ 2-17 บล็อกไดอะแกรมและการบรรยายการเชื่อมต่อของ clock_ component

- การกำหนดรูปแบบการบรรยาย

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้ ซึ่งในการบรรยายสามารถกำหนดค่าของสัญญาณ เอาท์พุทในเทอมของอินพุทหรือในรูปขององค์ประกอบอื่นๆ หรือทั้งสองอย่างรวมกันก็ได้ ดังตัวอย่างการบรรยายของ clock_component ในรูปที่ 2-18 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมโดยมี en เป็นอินพุทและ ck เป็นเอาท์พุท PROCESS เป็นคำที่ใช้ในการเริ่มต้นสำหรับการบรรยายในเชิงพฤติกรรม และภายในโปรเซสกำหนดให้ periodic เป็นตัวแปรที่มีคำเริ่มต้นเป็น "0" ถ้าสัญญาณ en มีค่าเป็น "1"' จะทำให้ตัว

แปร periodic ถูกคอมพลีเมนท์ (complement) และส่งค่าให้กับ ck ซึ่งเป็นสัญญาณเอาท์พุท และสำหรับคำสั่ง WAIT จะเป็นการกำหนดให้สัญญาณมีคาบเวลาเท่ากับ 1 ไมโครวินาที

```
ARCHITECTURE behavioral OF clock_component IS

BEGIN
   PEOCESS
      VARIABLE periodic : BIT := '0';
   BEGIN
      IF en='1' THEN
       periodic := Not periodic;
      END IF;
       ck <= periodic;
       WAIT FOR 1 US;
  END PROCESS;
END behavioral;
```

รูปที่ 2-18 การบรรยายเชิงพฤติกรรมของ clock_ component

- หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อย ที่เป็นประโยชน์ต่อการเขียนรูปแบบการบรรยายระบบดิจิตอล สามารถเก็บไว้ใน ส่วนของแพ็กเก็ต ซึ่งหน่วยการออกแบบต่างๆ เช่น หน่วยการออกแบบ Entity หน่วยการออกแบบสถาปัตยกรรมหรือ หน่วยการออกแบบแพ็กเกจอื่นๆ สามารถเรียกข้อมูลเหล่านี้ไปใช้ได้ นอกจากนั้น สิ่งที่นิยมทำกันมากคือการนำรูปแบบ มาตรฐานต่างๆ เช่น อุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 74XX เป็นต้น) มาเก็บไว้ในรูปของแพ็กเกจ ที่ทุกคนสามารถ เข้าถึงได้ ตามปกติแล้วแพ็กเกจจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเกจ (Package declaration)และ ส่วนของบอดี้แพ็กเกจ (Package body) เนื่องจาก แพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่ กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษา VHDL สามารถ กระทำได้ด้วยชุดคำสั่ง USE

ก) PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ส่วนการประกาศแพ็กเกจ เนื่องจากเป็นส่วนที่ใช้คำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเกจ สำหรับนำไปใช้ภายนอกตัวของแพ็กเกจเอง ถ้ามี การประกาศสิ่งใดๆ ในส่วนของส่วนบอดี้แพ็กเกจ แต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเกจจะทำให้ค่าและพฤติ กรรมไม่สามารถนำไปใช้งานในส่วนนอกได้ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ Entity คือ จุดเชื่อมต่อ หรือ พอร์ต ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเกจสามารถสร้างขึ้นได้โดยไม่จำเป็น ต้องมีส่วนบอดี้ และยังสามารถนำไปใช้งานจากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศ ชนิด (Type) หรือสัญญาณ เช่นเดียวกับ ส่วนบอดี้แพ็กเกจที่ไม่จำเป็นต้องมี ส่วนของการประกาศแพ็กเกจ แต่แพ็กเกจนั้นจะไม่สามารถนำไปใช้จาก รูปแบบอื่นได้

```
PACKAGE package_name IS
    Package_declarative_part
END package_name;
```

รูปที่ 2-19 โครงสร้างทั่วไปของส่วนการประกาศแพ็กเกจ

ข) PACKAGE BODY

โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อยทั้งหลาย ซึ่งชื่อของโปรแกรมย่อยนั้นๆ ได้ถูกประกาศไปแล้วในส่วนของการประกาศแพ็กเกจ จะถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจ ทั้งนี้รวมถึง การกำหนดค่าคงที่ต่างๆ อันได้แก่ค่าคงที่ที่ถูกประกาศชื่อไว้ก่อนในส่วนของการประกาศแพ็กเกจ และถูกกำหนดค่าใน ส่วนของบอดี้แพ็กเกจ ฉะนั้นในส่วนของบอดี้แพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการ ประกาศชื่อที่เป็นโปรแกรมย่อย หรือค่าคงที่ การเขียนบอดี้แพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังแสดงในรูปที่ 2-20

```
PACKAGE BODY package_name IS
    declarative part
END package_name;
```

รูปที่ 2-20 โครงสร้างของบอดี้แพ็กเกจ

- **หน่วยการออกแบบ Configuration**

ดังที่ทราบกันแล้วว่าระบบดิจิตอลรูปแบบหนึ่งไม่ว่าจะเป็นอะไรก็ตาม จะสามารถมีหน่วยการออกแบบ Entity ได้ เพียงหนึ่งเดียวเท่านั้น ซึ่งในหน่วยการออกแบบ Entity หนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลาย หน่วย ดังนั้นจะต้องมีหน่วยการออกแบบ Configuration มาเพื่อกำหนดการใช้ Configuration ของการประกอบ Entity กับหน่วยการออกแบบสถาปัตยกรรมหน่วยใดๆ เข้าด้วยกัน

```
COMFIGURTRION identifier OF entiry_name IS
    Configuration_declarative_part
END ;
```

รูปที่ 2-21 สร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

- **โปรแกรมย่อย**

การใช้ฟังก์ชันและโพรซีเจอร์ใน VHDL เปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไปค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อยอาจจะมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่นถ้าใช้ฟังก์ชัน แทนการกระทำในสมการบูลีนก็จะมีผลต่อวงจรลอจิกจริงๆในขณะที่ถ้าใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูล หรือในการคำนวณค่าการหน่วงเวลาแล้วก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์ รูปที่ 2-22 แสดงการใช้โพรซีเจอร์ เพื่อเปลี่ยนข้อมูลชนิด 8 บิตเป็นค่าจำนวนเต็ม และรูปที่ 2-23 แสดงการใช้ฟังก์ชันโดยกำหนดให้ X เป็นตัวแปรชนิด บิตแทนการกระทำในสมการบูลีน

```
TYPE byte IS ARRAY (7 DOWNTO 0) OF BIT;
 ...
PROCEDURE byte_to_integer (ib : IN byte; oi : OUT INTEGER) IS
        VARIABLE result: INTEGER := 0;
BEGIN
        FOR i IN 0 TO 7 LOOP
                IF ib(i) = '1' THEN
                result  := result + 2**i;
                END IF;
        END LOOP;
        oi := result;
END byte_to_integer
```

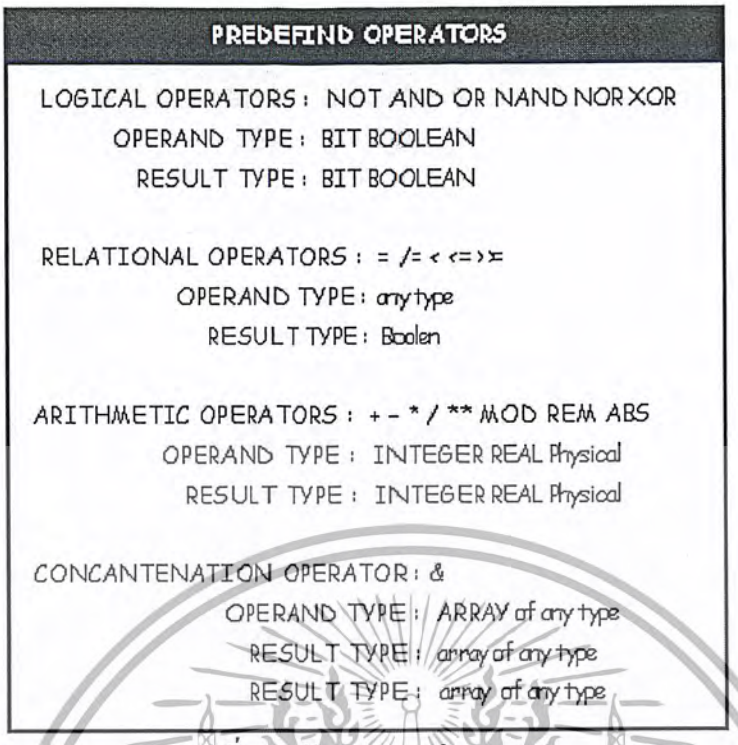
รูปที่ 2-22 การใช้โพรซีเจอร์

```
FUNCTION f (a, b, c: BIT) RETURN BIT  IS
        VARIABLE x: BIT;
BEGIN
        x := ((NOT a) AND (NOT b) AND c);
        RETURN x;
END f;
```

รูปที่ 2-23 การใช้ฟังก์ชัน

- โอเปอร์เรเตอร์

การบรรยายเชิงพฤติกรรมในภาษา VHDL มีตัวดำเนินการหรือโอเปอร์เรเตอร์ทางลอจิกและคณิตศาสตร์เช่นเดียว กับภาษาซอฟต์แวร์ทั่วไปดังรูปที่ 2-24



รูปที่ 2-24 ตัวดำเนินการใน VHDL

- เวลาและความพร้อมเพรียง

ในวงจรอิเล็กทรอนิกส์อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ (Always Active) และจะมีเรื่องของเวลา เข้ามาเกี่ยวข้องในทุกๆเหตุการณ์ที่เกิดขึ้นเสมอ VHDL เป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถบรรยายรูป แบบและการพ้องกันของเวลาสำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วน ของการบรรยายสถาปัตยกรรม จะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซสซึ่งมีการทำงานภายในเป็น แบบลำดับคำสั่งก็ตาม ซึ่งหากมีหลายๆ โปรเซสอยู่ภายในโครงสร้างเดียวกัน ทุกๆ โปรเซสก็จะทำงานไปพร้อมๆ กันด้วย

- สัญญาณและตัวแปร

สัญญาณมีลักษณะเป็นเสมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูลและมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยการ กำหนดค่าให้กับสัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่าและสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการ ส่งผ่านค่าของสัญญาณ เช่น w <= a AFTER 12 NS หมายถึงการกำหนดค่าสัญญาณ a ให้กับ w หลังจากเวลา ผ่านไป 12 นาโนวินาที ในทางตรงข้ามตัวแปรมีลักษณะเป็นเสมือนตัวกลางที่ใช้ในการส่งผ่านข้อมูลและไม่มีเรื่องของ เวลาเข้ามาเกี่ยวข้องด้วย ซึ่งตัวแปรจะถูกใช้ในส่วนที่มีการทำงานเป็นแบบลำดับคำสั่งเช่นใน ฟังก์ชัน โพรซีเจอร์ และ โปรเซส สำหรับการกำหนดค่าให้กับตัวแปรจะใช้สัญลักษณ์ :=

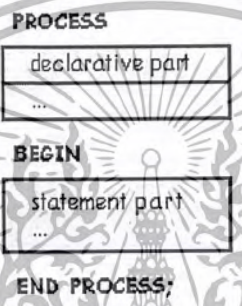
2.7.4 การบรรยายเชิงพฤติกรรม

การบรรยายลักษณะการทำงานของอุปกรณ์ฮาร์ดแวร์ในเชิงพฤติกรรม เป็นการบรรยายลักษณะการเปลี่ยนแปลงของ ข้อมูลในรูปแบบของอัลกอริธึมสำหรับการคำนวณผลลัพธ์ที่เกิดขึ้นซึ่งสืบเนื่องมาจากการเปลี่ยนแปลงสภาวะของข้อมูล ที่เข้ามาโดยไม่คำนึงถึงลักษณะโครงสร้างหรือความสัมพันธ์ของอุปกรณ์ที่อยู่

ภายในว่าจะเป็นอย่างไร ในหัวข้อนี้จะ แสดงถึงการบรรยายเชิงพฤติกรรม แทนการใช้โมดูลฮาร์ดแวร์รวมถึงข้อกำหนดต่างๆ ที่ควรรู้

2.7.5 โปรเซส

โปรเซสเป็นรูปแบบพื้นฐานอย่างหนึ่งที่ใช้ในการกำหนดให้กับสัญญาณ โปรเซสจะอยู่ในสถานะที่เตรียมพร้อมอยู่เสมอ และจะปฏิบัติคำสั่งพร้อมๆ กันกับโปรเซสอื่นๆ ที่อยู่ในสถาปัตยกรรมบรรยายเดียวกัน โดยโปรเซสจะปฏิบัติงานตามคำ สั่งทันทีที่มีเหตุการณ์เกิดขึ้นกับสัญญาณที่อยู่ทางด้านขวามือของสัญลักษณ์กำหนดค่าให้กับสัญญาณ (<=) การบรรยาย โปรเซสจะเริ่มต้นด้วยคำสั่ง PROCESS และจบด้วยคำสั่ง END PROCESS ในรูปที่ 2-25 เป็นการแสดงส่วน ประกอบของการบรรยายแบบโปรเซส ซึ่งประกอบด้วยส่วนของการประกาศตัวแปรที่ต้องใช้และส่วนของการปฏิบัติ คำสั่งเพื่อให้ได้ผลลัพธ์ที่ต้องการ



รูปที่ 2-25 รูปแบบของการบรรยายแบบโปรเซส

2.7.6 การกำหนดตัวดำเนินการภายในโปรเซส

ตัวดำเนินการภายในโปรเซสมี 3 ชนิดคือ ตัวแปร (Variable) ไฟล์ (File) และตัวคงที่ (Constant) ซึ่งตัวดำเนิน การทั้งสามชนิดนี้หากมีการประกาศไว้ในโปรเซสใดก็จะใช้ได้เฉพาะภายในโปรเซสนั้นเท่านั้นสำหรับการติดต่อกับภาย นอกหรือระหว่างโปรเซสสามารถทำได้โดยใช้สัญญาณ (Signal) หรือตัวคงที่ที่ได้ประกาศไว้ในส่วนของ ARCHITECTURE ในรูปที่ 2-26 แสดงตัวอย่างการประกาศตัวกระทำภายในโปรเซส ซึ่งจะอยู่ระหว่างคำสั่ง PROCESS และ BEGIN และค่าเริ่มต้นที่ถูกกำหนดให้กับตัวดำเนินการภายในโปรเซสจะถูกนำมาใช้ในตอนเริ่มต้น ของการปฏิบัติเพียงครั้งเดียวเท่านั้น ต่างกับค่าเริ่มต้นที่อยู่ภายในโปรแกรมย่อยจะถูกนำมาใช้ทุกครั้งที่มีการเรียกใช้ โปรแกรมย่อยนั้น ๆ

```
PROCESS
   FILE flush : TEXT IS IN "filename.dat";
   VARIABLE var : BIT;
   CONSTANT n : INTEGER := 0;
BEGIN
   ....
END PROCESS;
```

รูปที่ 2-26 ตัวอย่างการประกาศตัวดำเนินการภายในโปรเซส

2.7.7 การกำหนดการกระทำภายในโปรเซส

การกระทำใดๆ ภายในโปรเซสจะเป็นการปฏิบัติแบบลำดับ (Sequential) เสมอ ซึ่งภายในโปรเซสสามารถใช้ประโยค เงื่อนไขหรือการทำซ้ำได้เช่น IF-THEN - ELSE,CASE - WHEN, FOR LOOP และ WHILE LOOP ดังตัวอย่างในรูปที่ 2-27 และ 2-28

```
ARCHITECTURE demo OF partial_process IS
 ....
BEGIN
  PROCESS
  ....
   BEGIN
          ....
          x <= '1';
          IF x = '1' THEN
            perform action_1
          ELSE
            perform action_2
          END IF;
          ...
   END PROCESS;
END demo;
```

รูปที่ 2-27 เงื่อนไขการกระทำในโปรเซส

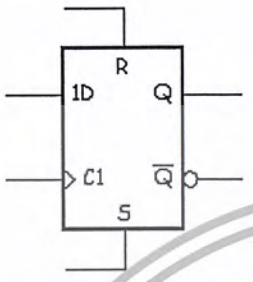
```
ARCHITECTURE demo OF partial_process IS
  ...
BEGIN
  PROCESS
   BEGIN
          ...
          x <= a AFTER 10 NS;
          y <= b AFTER 6 NS;
          ...
   END PROCESS;
END demo;
```

รูปที่ 2-28 แสดงการกระทำในโปรเซส

2.7.8 การกระตุ้นและยับยั้งการกระทำของโปรเซส

การกระทำภายในโปรเซสจะอยู่ในสภาพเตรียมพร้อม และมีการปฏิบัติงานอยู่ตลอดเวลาที่มีการเปลี่ยนแปลงของเหตุการณ์ เกิดขึ้น อย่างไรก็ดีเราสามารถกระตุ้นหรือยับยั้งการกระทำภายในโปรเซสได้โดยการกำหนดรายการของสัญญาณที่ต้อง การให้โปรเซสปฏิบัติงานเมื่อมีเหตุการณ์เกิดขึ้นกับสัญญาณที่กำหนดไว้เท่านั้น ส่วนเหตุการณ์ใดๆ ที่เกิดขึ้นกับสัญญาณ ที่ไม่ได้กำหนดไว้ในรายการก็จะไม่ส่งผลให้มีการกระทำ

ภายในโปรเซส ซึ่งรายการของสัญญาณนี้เรียกว่า Sensitivity List และจะกำหนดไว้ภายในวงเล็บหลังคำสั่ง PROCESS รูปที่ 2-29 (a) แสดงตัวอย่างโมเดล และรูปที่ 2-29 (b) เป็นตัวอย่างการบรรยายการเชื่อมต่อของ D-Flip Flop ส่วนรูปที่ 2-30 แสดงถึงการบรรยายเชิงพฤติกรรมของ D-Flip Flop โดยในรูปที่ 2-30 (a) เป็นการใช้ตัวกระทำภายนอกโปรเซส และรูปที่ 2-30 (b) เป็นการใช้ตัวกระทำภายในโปรเซส โดยมีรายการของสัญญาณ (rst, set, clk) เป็นตัวกระตุ้นการปฏิบัติงานภายในโปรเซส



(a)

```
ENTITY component_name IS
    Input and output ports
    Physical and other parameters
END component_name;
```

(b)

รูปที่ 2-29 (a) ตัวอย่างโมเดล D-Flip Flop (b) การบรรยายการเชื่อมต่อของ D-Flip Flop

```
ARCHITECTURE behavioral OF d_sr_flipflop IS
  SIGNAL state : BIT := '0';
BEGIN
  dff : PROCESS (rst, set, clk)
        BEGIN
      IF set= '1' THEN
          state <= '1' AFTER sq_delay;
        ELSIF rst = '1' THEN
          state <= '0' AFTER rq_delay;
        ELSIF clk = '1' AND clk ' EVENT THEN
          state <= d AFTER cq_delay;
      END IF;
  END PROCESS dff;
  q <= state;
  qb <= NOT state;
END behavioral;
```

(a)

```
ARCHITECTURE average_dalay_ behavioral OF d_sr_flipflop IS
BEGIN
  dff : PEOCESS (rst, set, clk)
          VARIABLE state : BIT := '0';
        BEGIN
      IF set= '1' THEN
          state <= '1';
        ELSIF rst = '1' THEN
          state <= '0';
        ELSIF clk = '1' AND clk ' EVENT THEN
          state <= d;
      END IF;
         q <= state AFTER (sq_delay + rq_delay + cq_delay)/3;
        qb <= NOT state AFTER(sq_delay + rq_delay + cq_delay)/3;
  END PROCESS dff;
END behavioral;
```

(b)

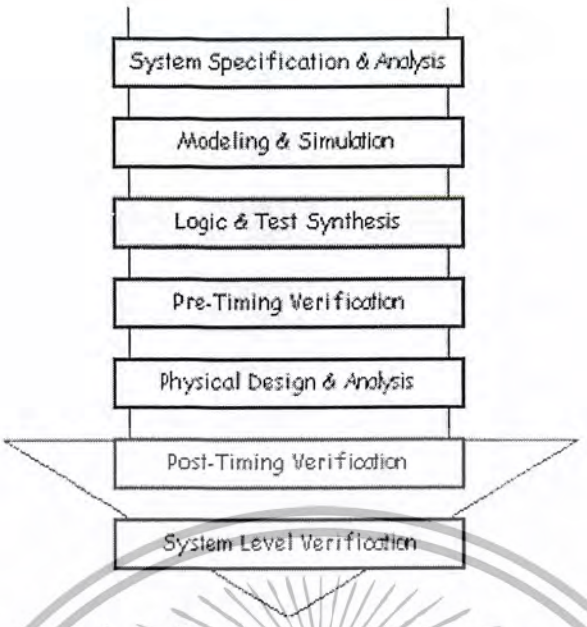
รูปที่ 2-30 การบรรยายเชิงพฤติกรรมของ D-FlipFlop

(a) การใช้ตัวกระทำภายนอกโปรเซส

(b) การใช้ตัวกระทำภายในโปรเซส

2.7.9 การออกแบบจากบนลงล่าง

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน วิศวกรหรือผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของ บล็อกไดอะแกรมก่อนที่ทำวิเคราะห์ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษา VHDL นั้นอนุญาตให้อธิบายและวิเคราะห์การ ทำงานของแต่ละบล็อก รวมถึงการปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามต้องการ นอกจากนี้ยัง สามารถเพิ่มเติมในรายละเอียดในแต่ละขั้นตอนได้ ซึ่งหลักการนี้สอดคล้องกับหลักการออกแบบจากบนลงล่าง (Top - Down Design) นั่นเอง ถ้าทดลองเปรียบเทียบกับการออกแบบจากล่างขึ้นบน (Bottom - Up Design) จะเห็นได้ว่า การออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า 90% เนื่องจากเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบ ก่อนแล้วจึงทำการจำลองการทำงาน และตรวจ สอบความถูกต้อง ดังนั้นการใช้ภาษา VHDL กับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับวิศวกรให้สามารถ ออกแบบและพัฒนาวงจรที่มีความซับซ้อนได้มากขึ้น ทั้งยังช่วยลดเวลาและค่าใช้จ่ายในการออกแบบด้วย



รูปที่ 2-31 ขั้นตอนการออกแบบจากบนลงล่าง

จากรูปที่ 2-31 แสดงถึงขั้นตอนของการออกแบบจากบนลงล่าง ทั้งนี้ในทางปฏิบัติอาจมีข้อแตกต่างไปจากนี้บ้าง เล็กน้อยเนื่องจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้หลายเทคโนโลยี สำหรับรายละเอียดของขั้นตอน การออกแบบจากบนลงล่างในแต่ละขั้นตอนมีดังนี้

2.7.9.1 สร้างข้อกำหนดของความต้องการ และวิเคราะห์ระบบ เพื่อหาแนวความคิดและหลักการ (Idea and Concept) ใน การแก้ปัญหา

2.7.9.2 เขียนรูปแบบของระบบที่ต้องการออกแบบโดยใช้ภาษา VHDL หรือ ภาษา HDL อื่น ๆ สำหรับบรรยายพฤติกรรมการ ทำงาน พร้อมทั้งจำลองการทำงาน เพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด

2.7.9.3 หลังจากที่ได้หลักการขั้นต้นพร้อมแนวความคิดที่ผ่านการตรวจสอบแล้วหลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมา เป็นลำดับขั้นที่สอง จนกระทั่งอยู่ในระดับที่จะนำไปผลิตวงจริง หรือสังเคราะห์ในขั้นตอนนี้เองเทคโนโลยีที่จะมารองรับ วงจรออกแบบจะถูกกำหนดขึ้น และระบบช่วยการออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้นให้อยู่ในรูปของ วงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือวงจรในระดับเกท และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้นหรือ ไม่ก็อยู่ในรูปของ Netlist ที่สามารถนำไปผลิตในอุปกรณ์อื่นได้

2.7.9.4 หลังจากการสังเคราะห์วงจรให้อยู่ในระดับเกทหรือ Netlist แล้ว ข้อมูลนี้จะถูกใช้สำหรับจำลองการทำงานในเรื่อง ความถูกต้องของฟังก์ชัน พร้อมกับนำข้อมูลที่เกี่ยวข้องกับเวลาเข้ามาประกอบการพิจารณาด้วย ซึ่งตามปกติแล้วอุปกรณ์ ทางอิเล็กทรอนิกส์ทุกชิ้นจะมีเวลาหน่วงของการแพร่กระจาย (Propagation Delay Time) เสมอ ถึงแม้ว่าจะเป็น เวลาที่น้อยมากในระดับนาโนวินาทีก็ตาม แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกทของฟังก์ชันต่างๆ จำนวน 10,000 เกท ขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้น จนอาจทำให้การทำงานของวงจรรวมทั้งหมดผิดพลาดไป หรือไม่สามารถทำ งานในย่านความถี่สัญญาณนาฬิกาที่สูงได้

2.7.9.5 ผลิตเป็นวงจรจริง (Technology and device mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจ จะอยู่ในรูปของแผงวงจรไฟฟ้า ที่ประกอบด้วยอุปกรณ์หลายๆ ชิ้นหรืออยู่ในรูปของวงจรรวม ASIC

2.7.9.6 ทำการตรวจสอบการทำงานและตัวแปรทางด้านเวลาทั้งหมด เพื่อความถูกต้องของวงจรเป็นครั้งสุดท้ายก่อนนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เนื่องจากในขั้นตอนนี้ วงจรที่ออกแบบ จะประกอบด้วยจุดต่อทางอินพุทและเอาท์พุท ซึ่งเป็นจุดต่อสำหรับการรับและส่งสัญญาณกับภายนอก

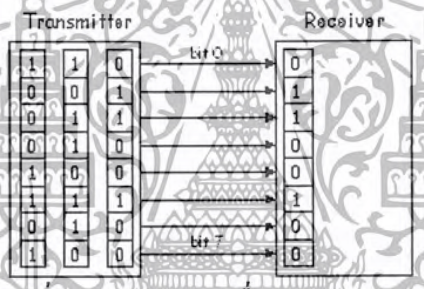
2.7.9.7 นำวงจรที่ออกแบบไว้ประกอบเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบที่สมบูรณ์ แล้วทำการทดสอบการทำงานทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้งเพื่อควบคุมคุณภาพของผลิตภัณฑ์

2.8 การรับ – ส่งข้อมูลแบบอนุกรมด้วย FPGA

ในการสื่อสารหรือการส่งข้อมูลในระบบดิจิตอลนั้นมีรูปแบบในการสื่อสารที่สำคัญอยู่ 2 แบบ คือ

- **การสื่อสารแบบขนาน (Parallel Communication)**

การสื่อสารในรูปแบบที่ส่งข้อมูลออกไปทีเดียวพร้อมๆกันทั้ง 8 bits ผ่านสายสัญญาณทั้ง 8 เส้น การสื่อสารและการส่งข้อมูลแบบขนานจะแสดงให้เห็นดังรูปที่ 2-32

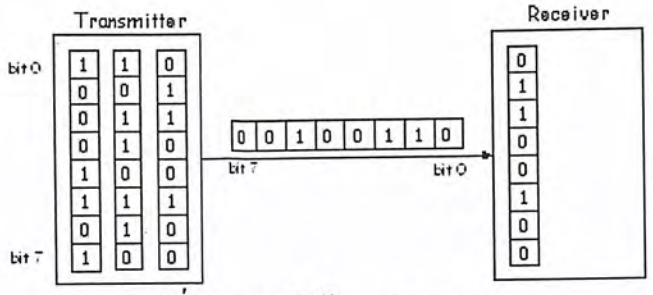


รูปที่ 2-32 ลักษณะการสื่อสารแบบขนาน

จะเห็นได้ว่าการสื่อสารแบบขนานมีข้อดีคือทำให้สามารถส่งข้อมูลทีละมากๆ และรวดเร็วกว่าการส่งแบบอนุกรม แต่การสื่อสารแบบขนานมีข้อจำกัดคือ ไม่สามารถส่งข้อมูลในระยะไกลๆได้และยังต้องใช้สายสัญญาณหลายเส้นในการส่งข้อมูล ทำให้สิ้นเปลืองกว่าการสื่อสารแบบอนุกรม ทำให้ไม่สะดวกในการใช้งาน ตัวอย่างของการสื่อสารแบบขนาน เช่น เครื่องพิมพ์ (Printer) และการสื่อสารทางพอร์ตขนาน (ECP Printer Port) เป็นต้น

- **การสื่อสารแบบอนุกรม (Serial Communication)**

เป็นการสื่อสารโดยการส่งข้อมูลทีละบิท ผ่านสายสัญญาณเส้นเดียว จนครบทั้ง 8 bits หรือ ไบท์ โดยจะส่งบิทต่ำ (LSB) ออกไปก่อน สามารถแสดงให้เห็นหลักการส่งข้อมูล แบบอนุกรมได้ดังรูปที่ 2-33



รูปที่ 2-33 การส่งข้อมูลแบบอนุกรม

2.8.1 การอินเตอร์เฟสตามมาตรฐาน RS-232

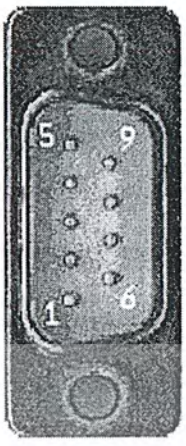
มาตรฐาน RS-232 เป็นมาตรฐานที่ได้รับการพัฒนามานานและถูกใช้งานกันอย่างแพร่หลาย เราใช้ RS-232 เชื่อมต่อ DTE (Data Terminal Equipment) เข้ากับ DCE (Data Communication Equipment) เช่นการต่อเทอร์มินัลเข้ากับโมเด็ม มาตรฐาน RS-232 กล่าวถึงลักษณะทางกล,ลักษณะของสัญญาณไฟฟ้าและลักษณะการทำงานที่ใช้การอินเทอร์เฟส ตัวอย่างของอุปกรณ์ที่ใช้ในการอินเทอร์เฟสตามมาตรฐาน RS-232 ได้แก่ เทอร์มินัล , พล็อตเตอร์ , ลอจิกแอนนาไลเซอร์ (Logic Analyzer) และเครื่องพิมพ์ ถ้าการประยุกต์ใช้งานของเราต้องการทำอินเทอร์เฟสอุปกรณ์ เข้ากับอินเทอร์เฟสมาตรฐาน RS-232 เราจำเป็นต้องแปลงระดับสัญญาณ TTL ให้เป็นระดับสัญญาณแบบอื่น ซึ่งรายละเอียดของระดับสัญญาณที่ใช้สำหรับ RS-232 จะได้กล่าวต่อไป

2.8.2 ลักษณะสัญญาณที่ใช้ในการอินเทอร์เฟส

มาตรฐาน RS-232 ใช้สัญญาณเพียงเส้นเดียวในการส่งสัญญาณ โดยสัญญาณจะส่งไปในทิศทางเดียวกัน ในกรณีที่อัตราเร็วในการส่งข้อมูลมีค่าเท่ากับ 20 kbps (กิโลบิทต่อวินาที) ซึ่งค่านี้เป็นค่าสูงสุดที่ใช้ในการสื่อสารข้อมูล (ในปัจจุบันพัฒนาให้สามารถส่งข้อมูลได้มากกว่านี้) ระยะทางในการส่งข้อมูลไม่ควรเกิน 50 ฟุต (ตามข้อกำหนดในมาตรฐาน) สำหรับการแทนแรงดันของระดับสัญญาณจะแทนระดับสัญญาณของลอจิก "0" ด้วยค่าแรงดัน +3 โวลท์ ถึง +12 โวลท์ ส่วนลอจิก "1" จะแทนระดับสัญญาณด้วยค่าแรงดันระหว่าง –3 โวลท์ ถึง –12 โวลท์

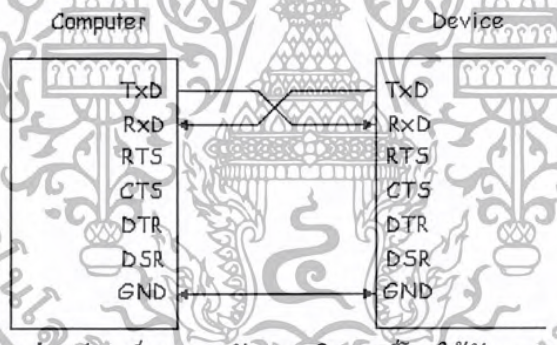
2.8.3 การออกแบบตัวแปลงสัญญาณ

การเชื่อมต่อกับพอร์ทสื่อสารของคอมพิวเตอร์ส่วนบุคคลจะเลือกใช้พอร์ตสื่อสารแบบ อนุกรม 9 ขา (DB-9) ซึ่งสามารถทำการรับส่งข้อมูลได้แบบอนุกรม โดยลักษณะของสัญญาณจะเป็นไปตามมาตรฐาน RS-232 โดยลักษณะของการเชื่อมต่อของพอร์ตสื่อสารสำหรับคอนเน็คเตอร์แบบ DB-9 สามารถแสดงให้เห็นได้ดังรูปที่ 2-34 และ รูปที่ 2-35



ตำแหน่งขา DB-9	สัญญาณ
1	Data Carrier Detect : DCD
2	Received Data : RxD
3	Transmitted Data : TxD
4	Data Terminal Ready : DTR
5	Signal Ground : GND
6	Data Set Ready : DSR
7	Request To Send : RTS
8	Clear To Send : CTS
9	Ring Indicator : RI

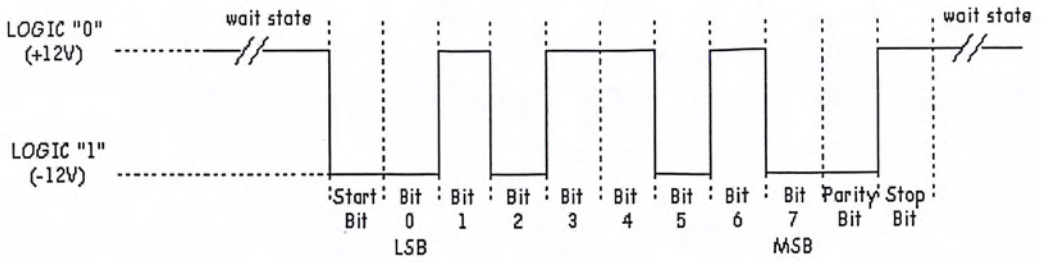
รูปที่ 2-34 การจัดขาของคอนเน็คเตอร์พอร์ตอนุกรมแบบ DB-9



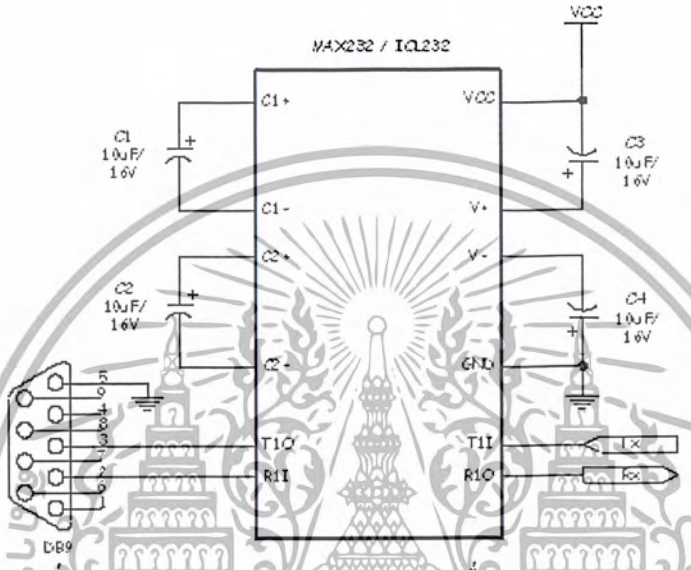
รูปที่ 2.35 การต่ออุปกรณ์ภายนอกกับคอมพิวเตอร์โดยใช้สัญญาณเพียง 3 เส้น

2.8.4 วงจรอแดปเตอร์

จากหัวข้อที่ผ่านมาตอนนี้เราทราบแล้วว่าในการสื่อสารแบบอนุกรมนั้น ในระดับของลอจิก "0" จะแทนระดับลอจิกด้วยระดับแรงดันระหว่าง +3 โวลท์ ถึง +12 โวลท์และในระดับของลอจิก "1" จะแทนระดับลอจิกด้วยระดับแรงดันระหว่าง –3 โวลท์ ถึง –12 โวลท์ เพราะฉะนั้นต้องทำการแปลง ระดับของลอจิก "1" และ "0" ให้เป็นระดับแรงดันดังกล่าว ซึ่งจะต้องใช้วงจรในการแปลงระดับแรงดัน สามารถแสดงลักษณะการส่งข้อมูลแบบอนุกรมที่ผ่านตัวแปลงแรงดันได้ดังรูปที่ 5 ในส่วนของวงจรแปลงระดับแรงดันสำหรับการสื่อสาร RS-232 นั้นจะใช้ไอซีเบอร์ MAX232 หรือ ICL232 ดังรูปที่ 2-37



รูปที่ 2-36 รูปแบบการสื่อสารแบบอนุกรม



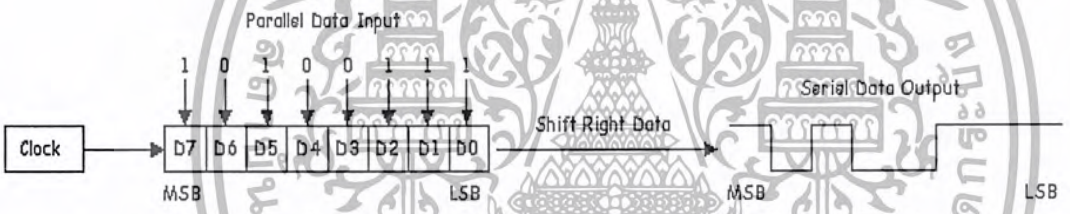
รูปที่ 2-37 วงจรแปลงระดับแรงดันของการสื่อสารแบบอนุกรม

2.8.5 รูปแบบของการส่งข้อมูลแบบอนุกรมและอัตราบอดในการสื่อสาร

อัตราบอด (Baud Rate) คือความเร็วในการรับ - ส่งข้อมูลอนุกรมมีหน่วยเป็นบิตต่อวินาที ซึ่งจะบอกจำนวนบิตที่รับ - ส่งในเวลา 1 วินาที เช่น ส่งข้อมูลด้วยอัตรา 9600 บิตต่อวินาที ก็คือการส่งข้อมูลตัวอักขระขนาด 10 บิต (บิต Start 1 บิต บิตข้อมูล 8 บิต บิต Stop 1 บิต) ได้ 960 ตัวอักขระใน 1 วินาที ซึ่งตารางที่ 1 แสดงอัตราบอดของ UART ที่ใช้กันทั่ว ตารางที่ 1 เป็นการแสดงอัตราบอดทั่วไปที่ใช้ในการโอนย้ายข้อมูลแบบอนุกรม

อัตราบอด	ช่วงเวลาของแต่ละบิต
110	9.91 ms
150	6.67 ms
300	3.33 ms
600	1.67 ms
1200	0.833 ms
2400	0.417 ms
4800	0.208 ms
9600	0.104 ms
19200	0.052 ms

ตารางที่ 2-2 อัตราบอดที่ใช้กันทั่วไป



รูปที่ 2-38 การแปลงข้อมูลแบบขนานเป็นข้อมูลแบบอนุกรม

ในรูปที่ 2-38 เป็นการแปลงข้อมูลแบบขนานเป็นข้อมูลแบบอนุกรม โดยเริ่มแรกข้อมูลแบบขนานจะถูกนำไปเก็บไว้ใน Shift Register หลังจากนั้นจะใช้สัญญาณนาฬิกาในการเลื่อนค่าในรีจิสเตอร์ออกมาทีละบิต (โดยการเลื่อนค่าไปทางขวามือ) โดยบิตแรกที่ถูกเลื่อนออกมาคือบิต LSB ของข้อมูลและบิตที่สองที่ถูกเลื่อนออกมาก็คือบิตที่อยู่ถัดจากบิต LSB และบิตต่อๆไป สำหรับบิตสุดท้ายที่ถูกเลื่อนออกมาก็คือบิต MSB ของข้อมูล

การแปลงข้อมูลแบบอนุกรมไปเป็นข้อมูลแบบขนานนั้นจะมีขั้นตอนตรงกันข้ามกับที่กล่าวมา นั่นคือข้อมูลแบบอนุกรมจะถูกเคลื่อนเขาไปใน Shift Register โดยใช้สัญญาณนาฬิกาเป็นตัวควบคุม และหลังจากที่มีการเคลื่อนข้อมูลทุกบิตเข้าไปใน Shift Register ได้หมดแล้ว ข้อมูลในรีจิสเตอร์นี้ จะถูกนำออกมาแบบขนานเพื่อนำไปใช้งานต่อไป

อุปกรณ์ที่ทำหน้าที่แปลงข้อมูลแบบอนุกรมเป็นข้อมูลแบบขนานและแปลงข้อมูลแบบขนานเป็นข้อมูลแบบอนุกรมเรียกว่า UART (Universal Asynchronous Receiver-Transmitter) ซึ่งนอกจากจะมีหน้าที่ในการแปลงข้อมูลแล้ว UART ยังมีหน่วยควบคุมและหน่วยตรวจสอบการทำงานอีกด้วย ในการส่งข้อมูลขนาด 8 บิต แบบอนุกรมนี้จะต้องมีบิตเริ่มต้น (Start Bit) และบิตหยุด (Stop Bit) เพิ่มขึ้นมาซึ่งจะทำให้ข้อมูลที่ถูกส่งไปจริงๆนั้นมีขนาด 10 บิต

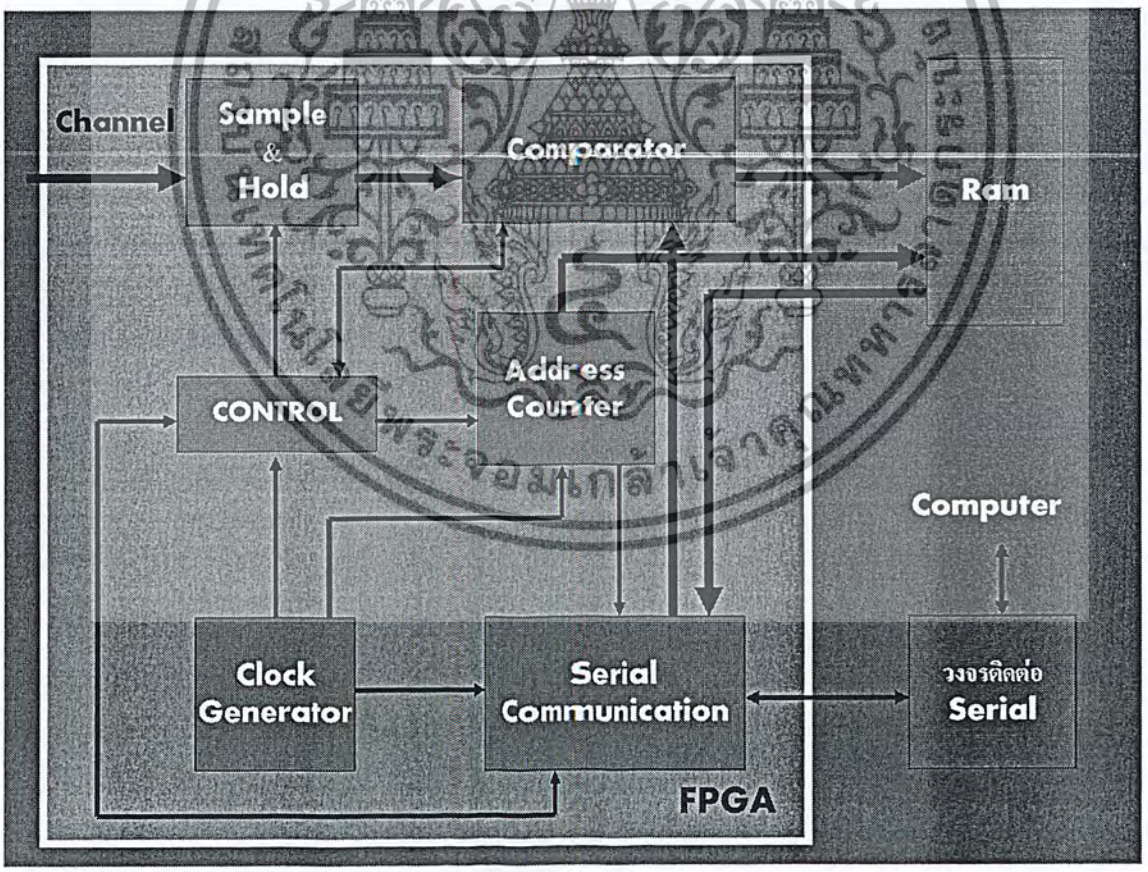
บทที่ 3

โครงสร้างและการออกแบบ

3.1 ทางการออกแบบทางฮาร์ดแวร์

แนวทางการออกแบบโดยอาศัยโครงสร้างและหลักการทำงานทั่วไปของลอจิกแอนนาไลเซอร์ที่กล่าวข้างต้นมาออกแบบและกำหนดรูปแบบที่ต้องการใหม่ดังนี้

- เครื่องลอจิกแอนนาไลเซอร์ที่สร้างขึ้น ต้องติดต่อใช้งานร่วมกับ เครื่องคอมพิวเตอร์ส่วนบุคคลเพื่ออาศัยจอภาพและ Disk เป็นตัวแสดงผลที่วัดได้และเก็บข้อมูลที่วัดได้เพื่อมาใช้ภายหลัง
- เครื่องวิเคราะห์สัญญาณทางตรรกะที่สร้างขึ้นจะรับส่งข้อมูลระหว่างเครื่องกับคอมพิวเตอร์ผ่านทางพอร์ตอนุกรม ทั้งนี้เนื่องจากพอร์ตอนุกรมมีอยู่แล้วในคอมพิวเตอร์ทั่วไป ทำให้สะดวกในการใช้งาน
- ช่องสัญญาณที่ต้องมีขนาด 16 ช่องสัญญาณ ซึ่งสามารถรองรับการใช้งานการตรวจจับสัญญาณของไมโครคอนโทรเลอร์ได้
- อัตราสุ่มสัญญาณสุ่มได้ถึง 24 เมกกะเฮิทซ์ เพื่อการรองรับการตรวจจับวิเคราะห์สัญญาณของไมโครคอนโทรเลอร์ ซึ่งมีอัตราความเร็วของสัญญาณนาฬิกาอยู่ที่ 12-20 เมกกะเฮิทซ์



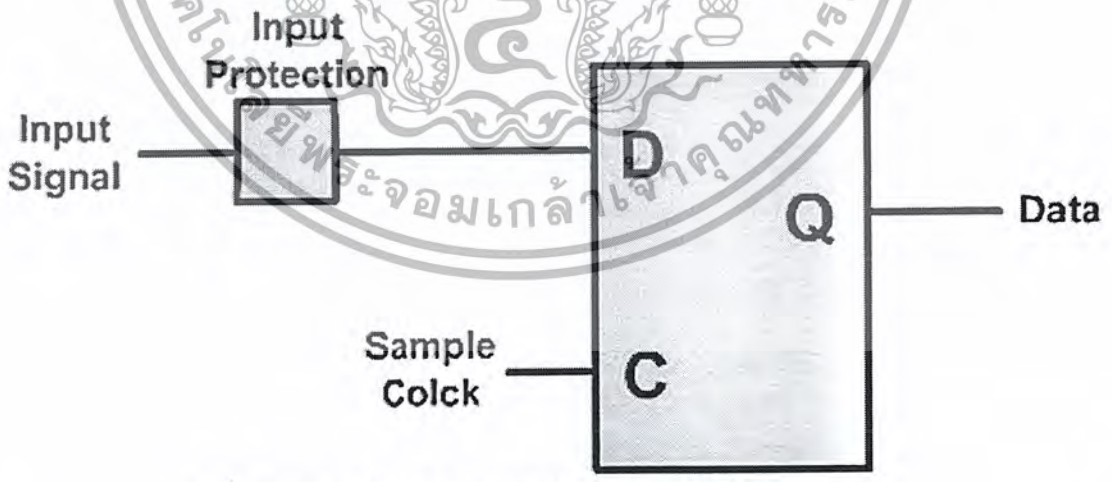
รูปที่ 3-1 แสดงภาพบล็อกของลอจิกแอนนาไลเซอร์

โดยอาศัยข้อกำหนดดังกล่าวข้างต้นจะสามารถเขียนแผนผังของบล็อกได้ดังรูป 3-1 สัญญาณทางตรรกะที่ได้รับเข้ามาจะผ่านสุ่มและคงค่า (Sampling and Hold) เพื่อทำให้สัญญาณที่ได้รับเข้ามาคงที่ (Stable) เสียก่อน จากนั้นสัญญาณที่รับเข้ามาจะรอการทริกสัญญาณจากภายนอก (Word trigger) โดยเป็นหน้าที่ของวงจรควบคุม(Control) จะส่งสัญญาณไปส่วนนับตำแหน่งแอดเดรส (Address counter) ทำงานส่วนนับตำแหน่งแอดเดรส จะรับสัญญาณจากส่วนสร้างสัญญาณนาฬิกา (Clock generator) และจะเพิ่มตำแหน่งแอดเดรสขึ้นไปทีละ หนึ่งเรื่อยๆ สัญญาณด้านออกของส่วนนับตำแหน่งจะถูกต่อเข้ากับหน่วยความจำ ที่ขาแอดเดรสที่สอดคล่องกัน ข้อมูลที่ผ่านการสุ่มและคงค่าจะถูกเข้าที่หน่วยความจำ เมื่อถึงขั้นตอนนี้จะเห็นได้ว่าข้อมูลที่รับเข้ามาจะถูกเก็บเข้าไว้ที่หน่วยความจำทั้งหมด ต่อจากนี้ไมโครโปรเซสเซอร์จะทำการส่งข้อมูลไปยังเครื่องคอมพิวเตอร์ เพื่อทำการแสดงผลต่อไป

แบ่งการทำงานของแต่ละขั้นตอนได้ดังนี้

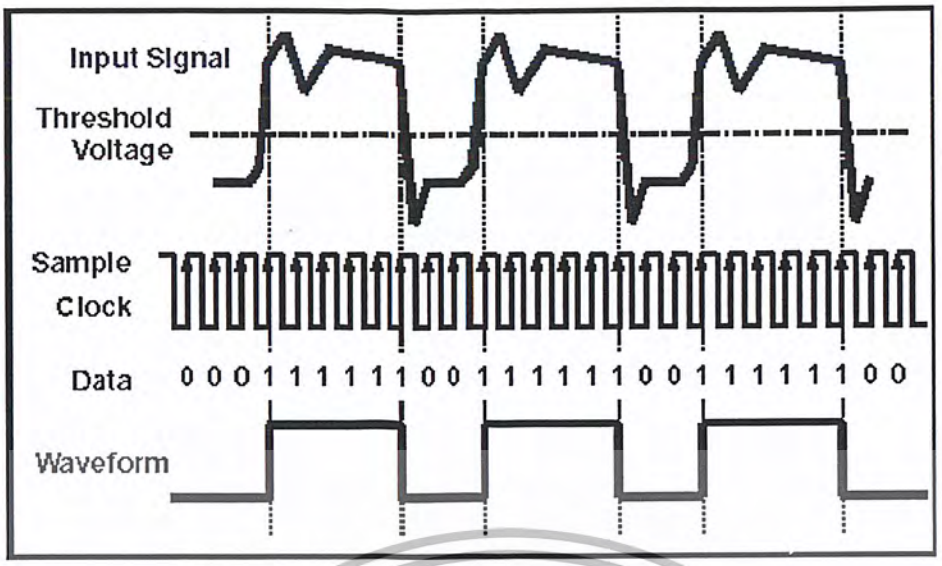
3.1.1 การสุ่มและคงค่าข้อมูล

การสุ่มสัญญาณให้ได้ค่าที่ถูกต้องและใกล้เคียงกับสัญญาณที่แท้จริงนั้น ความถี่ที่ใช้ในการสุ่มสัญญาณจะต้องเป็นอย่างน้อย 2 เท่าของความถี่ของสัญญาณที่ถูกสุ่ม ถ้าความถี่ที่ใช้ในการสุ่มสัญญาณมีค่ามากกว่าความถี่ของสัญญาณที่สุ่มมาเท่าไรความถูกต้องเหมือนสัญญาณจริงก็ยิ่งมากขึ้น แต่ตัวที่จะนำมาใช้กำหนดความถี่ที่ใช้ในการสุ่มก็คือ ความเร็วในการทำงาน โดยเฉพาะหน่วยความจำ เช่น หน่วยความจำมีความเร็วในการอ่านเขียนข้อมูลประมาณ 6 MHz หรือแต่ละรอบการทำงานจะใช้เวลา 150 ns ซึ่งจาเห็นได้ว่าถ้าการสุ่มสัญญาณได้สูงสุดเพียง 3 MHz ดังนั้นเราจึงนำหน่วยความจำธรรมดานี้จำนวน 8 ตัวมาต่อกัน โดยให้แต่ละตัวผลัดกันทำงานที่ความถี่ไม่เกิน 24 MHz หน่วยความจำแต่ละตัวจะทำงานได้ทันก่อนที่จะทำงานครั้งต่อไปจะมาถึง ถ้าเรามองหน่วยความจำทั้ง 8 ตัวนี้เป็นหน่วยความจำเพียงตัวเดียว จะเห็นได้ว่าหน่วยความจำตัวนี้ทำงานได้เร็วถึง 8 เท่าเราจึงเรียกว่า “หน่วยความจำความเร็วสูง”(High Speed Ram)



รูปที่ 3-2 แสดงรูปวงจรทำงานของการสุ่มและคงค่าข้อมูล

จากรูปที่ 3-2 การสุ่มสัญญาณให้ได้ค่าที่ถูกต้องและใกล้เคียงกับสัญญาณที่แท้จริงนั้นเมื่อผ่านการสุ่มสัญญาณส่งต่อให้ ดี-ฟลิปฟล็อป (D-Flip Flop) เพื่อคงค่าข้อมูล โดยจะทำงานการสุ่มและคงค่าข้อมูลตามสัญญาณนาฬิกา (Clock) ที่ส่งมาจากส่วนสร้างสัญญาณนาฬิกา (Clock generator) ได้สัญญาณที่มีความถูกต้องดังรูป 3-3



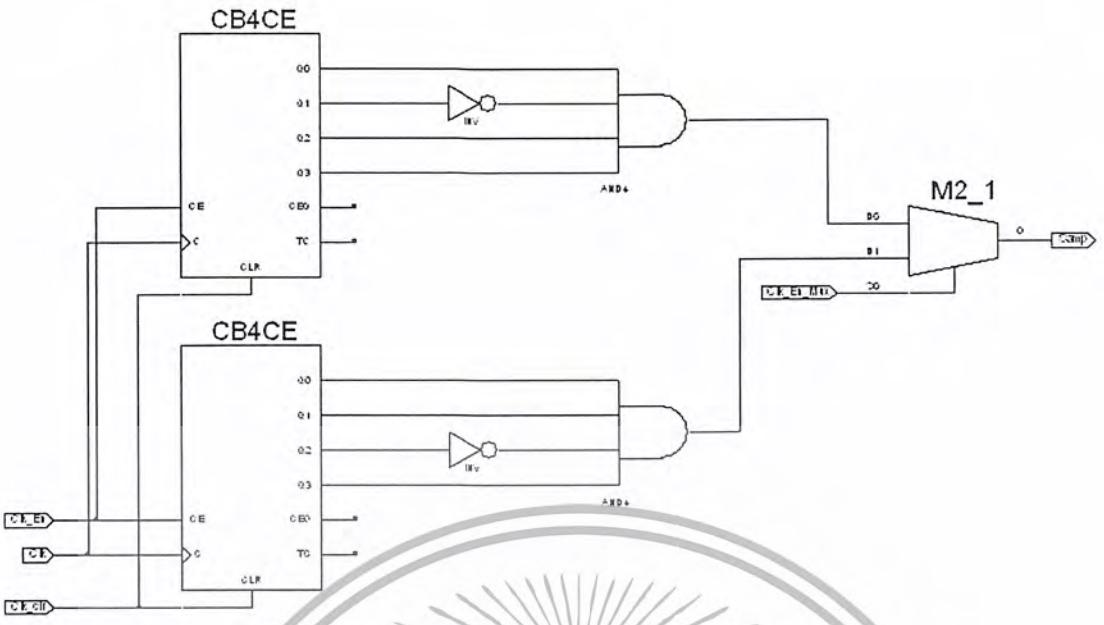
รูปที่ 3-3 แสดงรูปสัญญาณที่ได้จากวงจรสุ่มและคงค่าข้อมูล

3.1.2 การสร้างสัญญาณนาฬิกา

การสร้างสัญญาณนาฬิกา (Clock) เพื่อทำการจัดสรรสัญญาณนาฬิกาให้แต่วงจรให้ทำงานพร้อมกัน และทำการหารสัญญาณนาฬิกาให้เท่ากับความต้องการ

ความถี่แน่นอนเพื่อทำการสุ่มค่าสัญญาณจะใช้วงจรออสซิลเลเตอร์ (Oscillator) สร้างสัญญาณนาฬิกาผ่านให้วงจรนับ (Counter) เมื่อสัญญาณนาฬิกาผ่านเข้าวงจรนับ ก็จะนับเพิ่มขึ้นหนึ่ง หมายถึงการทำการสุ่มค่าสัญญาณครั้งหนึ่ง หรือเขียนข้อมูลลงในลงในหน่วยความจำค่าหนึ่ง วงจรนับจะนับจากศูนย์จนไปถึงค่าสุดท้าย (เนื้อที่ในหน่อยความจำ) แล้วจะวนนับศูนย์ใหม่อีกครั้งหนึ่ง ซึ่งหมายความว่า ได้เขียนข้อมูลลงในหน่วยความจำเต็มแล้ว

วงจรสร้างสัญญาณประกอบด้วยวงจรนับ (Counter) และวงจรเลือกค่าสัญญาณ (Multiplexer) เพื่อเลือกสัญญาณนาฬิกาที่ต้องการดังรูปที่ 3-4 โดยจะได้รับสัญญาณจากวงจรควบคุม(Control) การทำงานเลือกสัญญาณนาฬิกาที่ต้องการ



รูปที่ 3-4 แสดงวงจรสร้างสัญญาณนาฬิกา

3.1.3 การตรวจสอบสภาวะของสัญญาณและควบคุมการทำงาน

จะต้องมีวงจรหนึ่งทำการตรวจสอบสภาวะของสัญญาณ แล้วไปควบคุมการทำงานของเครื่องลอจิกแอนนาไลเซอร์โดยสภาวะที่จะตรวจสอบก็คือ

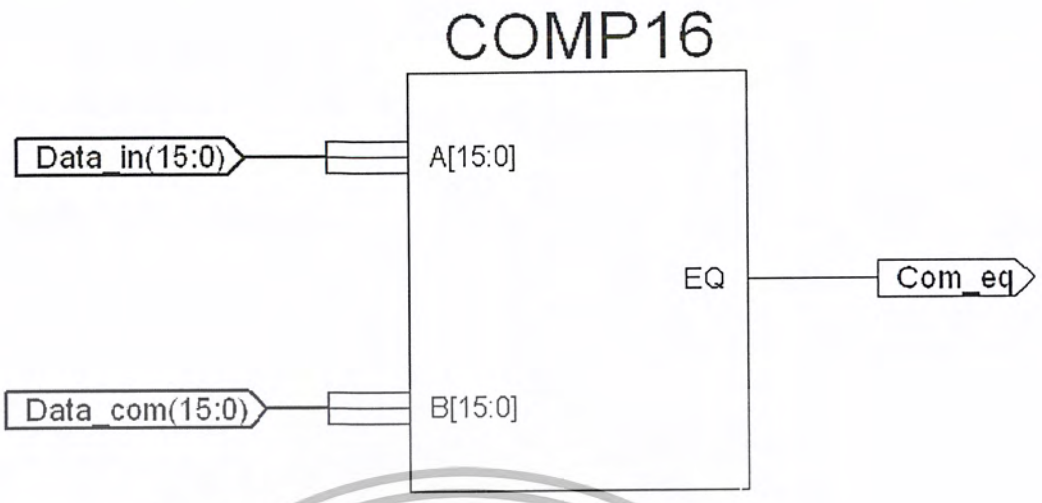
สัญญาณทริกเกอร์ (Trig) จากภายนอก ถ้าไม่มีสัญญาณทริกเกอร์จากภายนอกออกมาทริก เครื่องลอจิกแอนนาไลเซอร์ จะยังไม่เริ่มสุ่มสัญญาณ แต่ถ้ามีสัญญาณทริกเกอร์จากภายนอกแล้ว วงจรนับจะเริ่มนับหรือจาเริ่มสุ่มค่าสัญญาณและ ตรวจสอบว่าหน่วยความจำเต็มหรือยัง การตรวจสอบสภาวะนี้มีความสำคัญมาก ถ้าหากว่าหน่วยความจำถูกเขียนข้อมูลเต็มแล้ว แต่ยังมีการสุ่มค่าสัญญาณอีก จะทำให้ข้อมูลใหม่ไปทับข้อมูลเก่า ดังนั้นจึงต้องมีการตรวจสอบสภาวะนี้ ถ้าข้อมูลเต็มแล้วก็ให้วงจรนับหยุดนับหรือหยุดสุ่มค่าสัญญาณ

วงจรในส่วนการตรวจสอบสภาวะของสัญญาณและควบคุมการทำงานเป็นวงจรที่ทำงานตามเงื่อนไขที่แน่นอนก็คือถ้ามีสัญญาณริกจากภายนอกก็เริ่มสุ่มสัญญาณและจัดเก็บลงในหน่วยความจำถ้าหน่วยความจำถูกเขียนข้อมูลเต็มแล้ววงจรนับก็หยุดนับหรือหยุดสุ่มค่าสัญญาณกับหยุดจัดเก็บลงในหน่วยความจำนั้นเอง

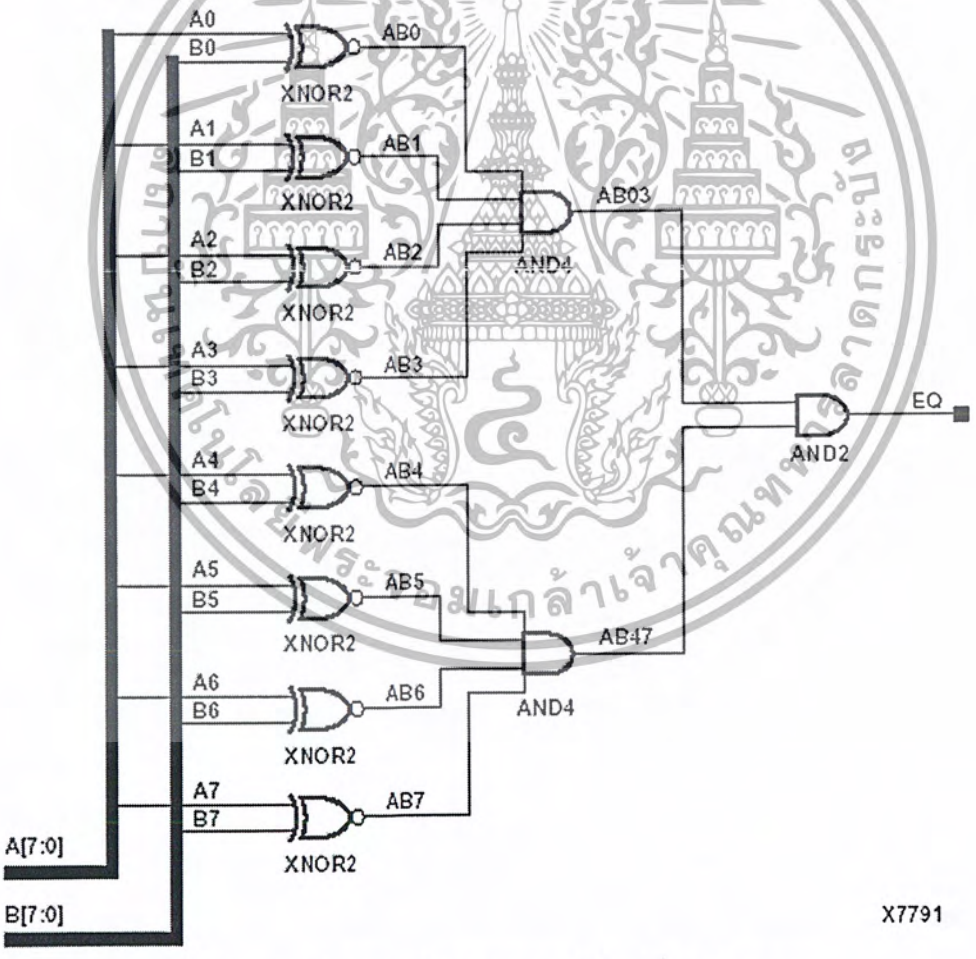
3.1.4 การเปรียบเทียบค่าสัญญาณ

การเปรียบเทียบสัญญาณที่เข้ามากับสัญญาณที่กำหนดจากคอมพิวเตอร์ในกรณีแบบมีทริกเกอร์ (Trigger) เมื่อเปรียบเทียบแล้วเท่ากันก็จะส่งผลให้กับวงจรควบคุมเพื่อส่งต่อให้กับคอมพิวเตอร์ต่อไป การตรวจจับสัญญาณแบบมีทริกเกอร์คือเมื่อเปรียบเทียบสัญญาณที่วัดได้กับสัญญาณที่กำหนดโดยผู้ใช้ถึงเริ่มการตรวจจับสัญญาณนั้นเอง

วงจรในส่วนของการเปรียบเทียบค่าสัญญาณนั้นประกอบด้วย ตัวเปรียบเทียบสัญญาณ(Comparator) ขนาด 16 บิต ส่งผลให้กับวงจรควบคุมเพื่อส่งต่อให้กับคอมพิวเตอร์ต่อไปดังรูปที่ 3-5 โดยจะมีวงจรภายในตามรูป 3-6



รูปที่ 3-5 แสดงรูปวงจรเปรียบเทียบค่าสัญญาณขนาด 16 บิต



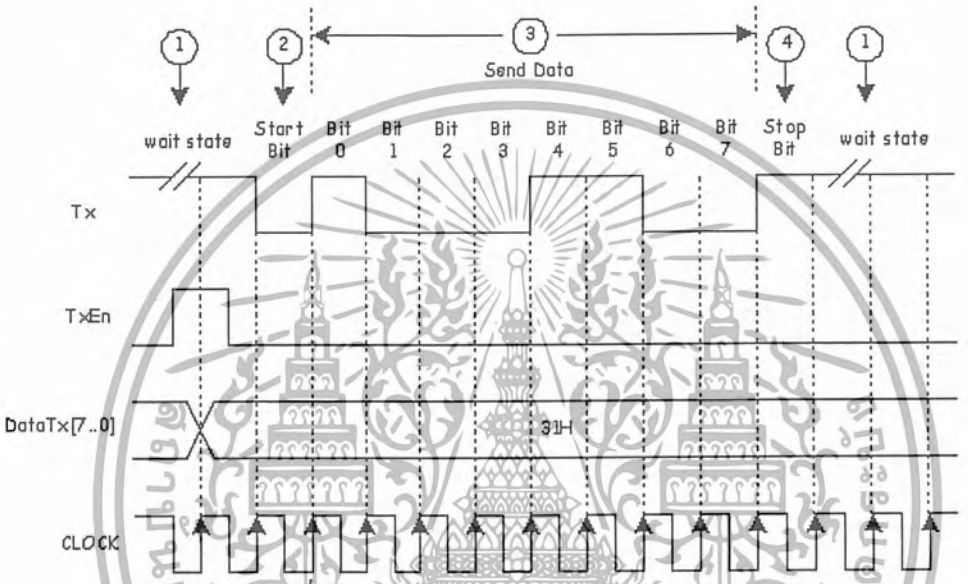
รูปที่ 3-6 แสดงรูปวงจรภายในของวงจรเปรียบเทียบค่าสัญญาณ

3.1.5 การติดต่อกับคอมพิวเตอร์

หลักการออกแบบวงจรรับ – ส่งข้อมูลแบบอนุกรมด้วย FPGA สำหรับวงจรสื่อสารข้อมูลแบบอนุกรมที่เรากำลังจะออกแบบกันนั้น จะมีคุณสมบัติดังนี้ Baud Rate = 9600 Bits/Sec, Data = 8 Bits, Start Bit = 1 Bit, Stop Bit = 1 Bit, Parity = none

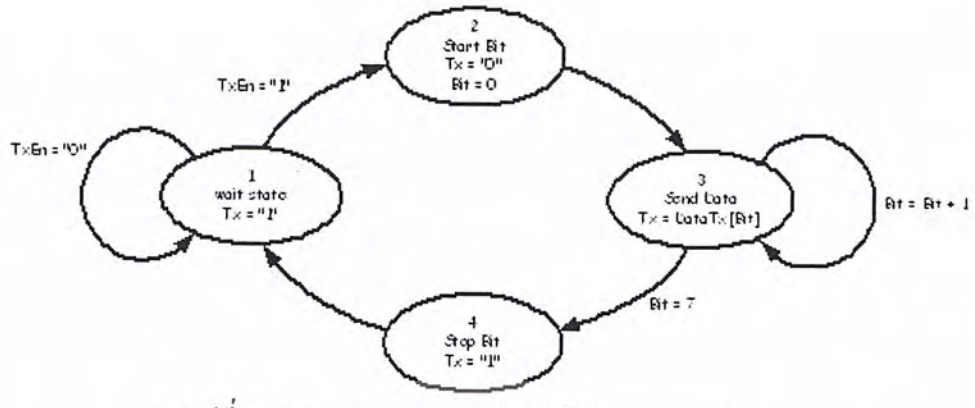
3.1.5.1 ภาคส่งข้อมูลแบบอนุกรม

ในส่วนนี้เป็นการออกแบบ FPGA ให้ทำหน้าที่เป็นตัวส่งข้อมูลแบบอนุกรมให้แก่อุปกรณ์ต่างๆ เช่น คอมพิวเตอร์ , ไมโครคอนโทรลเลอร์ เป็นต้น รูปแบบการส่งข้อมูลแบบอนุกรม แสดงดังรูปที่ 3-7



รูปที่ 3-7 รูปแบบการส่งข้อมูลแบบอนุกรม

จากรูปที่ 3-7 สัญญาณ Tx เป็นสัญญาณขนาด 1 บิต ใช้สำหรับส่งข้อมูลแบบอนุกรมออกไป ในการออกแบบวงจรส่งข้อมูลแบบอนุกรมจะต้องมีสัญญาณ CLOCK เป็นสัญญาณอ้างอิง เพื่อกำหนดความเร็วในการส่งข้อมูลซึ่งจะต้องตรงกับทางภาครับด้วย ความเร็วในการส่งข้อมูลนี้ก็คือ Baud Rate นั้นเอง เช่น ต้องการส่งข้อมูลด้วยความเร็ว 9600 บิตต่อวินาที ก็จะต้องกำหนดให้สัญญาณ CLOCK มีค่าเท่ากับ 9600 Hz เมื่อสัญญาณ TxEn มีค่าเป็นลอจิก “0” แสดงว่ายังไม่ต้องการส่งข้อมูล จะทำให้สัญญาณ Tx มีสถานะเป็นลอจิก “1” จนกว่าสัญญาณ TxEn จะมีสถานะเป็นลอจิก “1” แสดงว่าต้องการส่งข้อมูลออกไป จะทำให้สัญญาณ Tx มีค่าเป็นลอจิก “0” เพื่อเป็นการบอกทางภาครับว่าจะเริ่มต้นส่งข้อมูลออกไป หลังจากนั้นจะทำการส่งข้อมูลออกไป DataTx[7..0] เป็นข้อมูลที่ต้องการส่งแบบอนุกรม โดยส่งบิตที่มีความสำคัญต่ำสุด (Bit 0) ออกไปก่อน จะส่งข้อมูลออกไปที่ละบิตจนครบทั้ง 8 บิต จากนั้นสัญญาณ Tx จึงมีสถานะเป็นลอจิก “1” เพื่อบอกทางภาครับว่าเป็นการสิ้นสุดการส่งข้อมูล เราสามารถนำวิธีการส่งข้อมูลแบบอนุกรมที่ได้กล่าวมาแล้วมาเขียนเป็น State Diagram ได้ดังรูปที่ 3-7

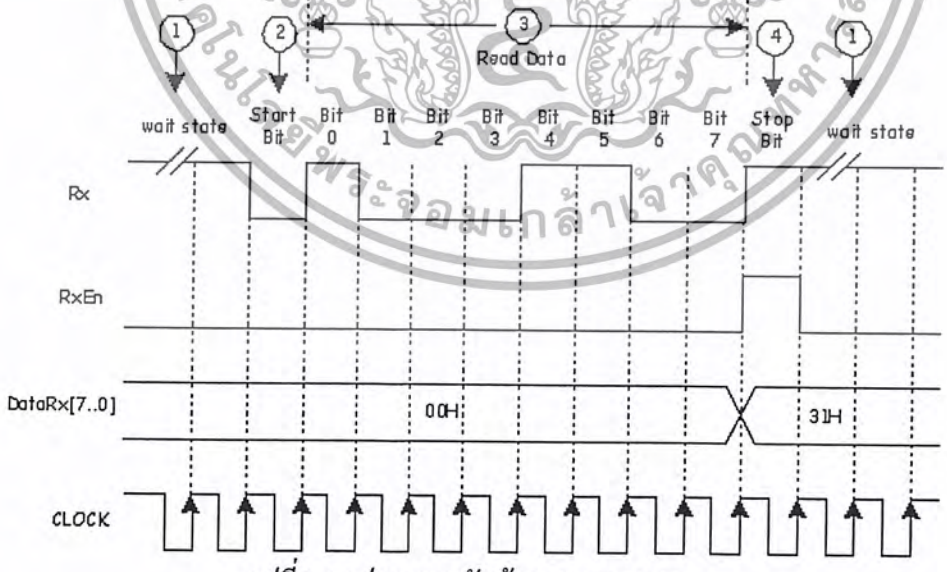


รูปที่ 3-8 State Diagram ของการส่งข้อมูลแบบอนุกรม

จาก State Diagram ในรูปที่ 3-8 เมื่อ TxEn มีค่าเป็นลอจิก "0" จะทำให้ Tx มีค่าเท่ากับลอจิก "1" และจะยังอยู่ใน State 1 จนกว่า TxEn จะมีค่าเป็นลอจิก "1" จึงจะเข้าสู่ State 2 ใน State 2 สัญญาณ Tx จะมีค่าเป็นลอจิก "0" เป็นการกำหนดบิตเริ่มต้นการส่งข้อมูล หลังจากนั้นจะเข้าสู่ State 3 ซึ่งจะเป็นการส่งข้อมูลออกไปที่ละบิต เริ่มต้นจะส่งข้อมูลบิตที่ 0 ออกไปก่อนและวนส่งข้อมูลออกไปจนครบ 8 บิต จึงจะหลุดเข้าสู่ State 4 จะกำหนดให้ Tx มีค่าเป็นลอจิก "1" เป็นการกำหนดบิตสิ้นสุดการส่งข้อมูล หลังจากนั้นจึงกลับเข้าสู่ State 1 ใหม่อีกครั้ง

3.1.5.2 ภาครับข้อมูลแบบอนุกรม

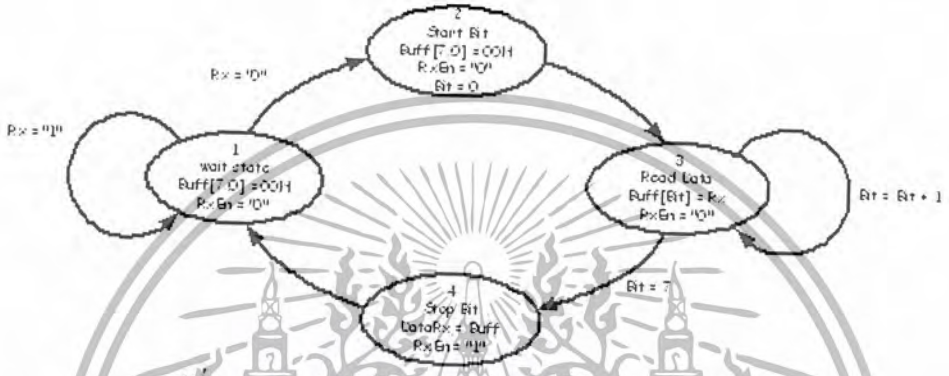
สำหรับภาครับข้อมูลแบบอนุกรมการทำงานจะคล้ายๆ กับทางภาคส่ง ข้อมูลแบบอนุกรมจะถูกส่งมาจากตัวส่ง เช่นคอมพิวเตอร์ , ไมโครคอนโทรลเลอร์ เป็นต้น ข้อมูลแบบอนุกรมที่ส่งมาจะมีลักษณะดังรูปที่ 10



รูปที่ 3-9 รูปแบบการรับข้อมูลแบบอนุกรม

จากรูปที่ 3-9 สัญญาณ Rx เป็นสัญญาณขนาด 1 บิตใช้สำหรับรับข้อมูลแบบอนุกรมที่ส่งมาจากตัวส่ง ในการออกแบบภาครับข้อมูลจะต้องมีสัญญาณ CLOCK เป็นสัญญาณอ้างอิง เพื่อกำหนดความเร็วในการรับข้อมูลซึ่งจะต้องสอดคล้องกับทางภาคส่ง เช่นภาคส่งใช้อัตราเร็วในการส่งข้อมูลเท่ากับ 9600 บิตต่อวินาที ทาง

ภาครับก็จะต้องกำหนดให้มีอัตราเร็วในการรับข้อมูลเป็น 9600 บิตต่อวินาทีเช่นกัน เพราะฉะนั้นจะต้องกำหนดให้สัญญาณ CLOCK มีความถี่เท่ากับ 9600 Hz เมื่อสัญญาณ Rx มีค่าเป็นลอจิก "1" แสดงว่ายังไม่มีการส่งข้อมูลออกมา จะรอจนกว่า Rx มีค่าเป็นลอจิก "0" แสดงว่าทางภาคส่งจะเริ่มส่งข้อมูลมาแล้ว หลังจากนั้นจะทำการอ่านข้อมูลเข้ามาเก็บไว้ที่ละบิตจนครบทั้ง 8 บิต และตรวจสอบสัญญาณ Rx ว่าเป็นลอจิก "1" หรือไม่ หากเป็นลอจิก "1" แสดงว่าสิ้นสุดการส่งข้อมูล และกำหนดให้สัญญาณ DataRx[7..0] มีค่าเท่ากับสัญญาณที่รับมาได้ และกำหนดให้สัญญาณ RxEn มีค่าเป็นลอจิก "1" เพื่อเป็นสัญญาณกระตุ้นให้วงจรต่อไปนำข้อมูล DataRx[7..0] ไปใช้งาน เราสามารถนำวิธีการรับข้อมูลแบบอนุกรมที่ได้กล่าวมาแล้วมาเขียนเป็น State Diagram ได้ดังรูปที่ 3-10



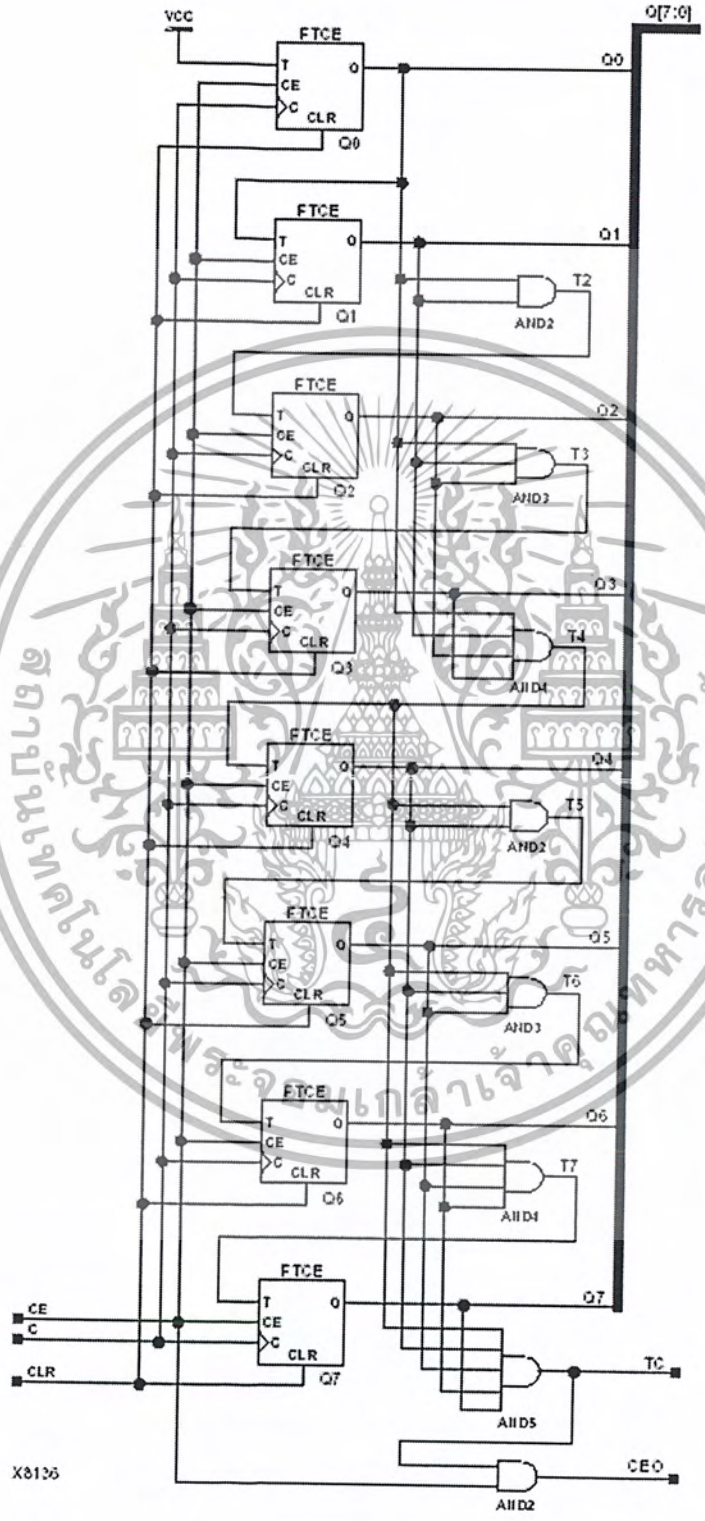
รูปที่ 3-10 State Diagram ของการรับข้อมูลแบบอนุกรม

จาก State Diagram ในรูปที่ 3-10 เริ่มต้นจะอยู่ใน State 1 หากสัญญาณ Rx มีค่าเป็นลอจิก "1" จะไม่มีการเปลี่ยน Stateจะให้ค่า RxEn เป็นลอจิก "0" และ Buff[7..0] ซึ่งเป็น Buffer สำหรับเก็บข้อมูลที่อ่านมาได้จากสัญญาณ Rx มีค่าเป็น "00000000" เมื่อสัญญาณ Rx มีค่าเป็นลอจิก "0" แสดงว่าเป็นบิตเริ่มต้นของข้อมูลจะเปลี่ยนเป็น State 2 เมื่อเข้ามาที่ State 2 แสดงว่าได้มีการส่งบิตเริ่มต้นข้อมูลมาแล้ว ใน State นี้จะยังคงกำหนดให้ Buff[7..0] มีค่าเท่ากับ "00000000" , RxEn เป็นลอจิก "0" และ Bit มีค่าเท่ากับ 0 หลังจากนั้นจะเข้าสู่ State 3 โดยอัตโนมัติ ภายใน State 3 จะมีการอ่านข้อมูลจาก Rx เข้ามาเก็บไว้ใน Buff จะอ่านข้อมูลจนครบทั้ง 8 บิต นำข้อมูลที่ได้ทั้งหมดเก็บไว้ใน Buff และกำหนดให้สัญญาณ RxEn มีค่าเป็นลอจิก "0" เมื่ออ่านข้อมูลจนครบทั้ง 8 บิตแล้วก็เปลี่ยน State เป็น State 4 ภายใน State 4 จะมีการโอนย้ายข้อมูลจาก Buff มาเก็บไว้ใน DataRx ซึ่งเป็นข้อมูลขนาด 8 บิตทั้งหมดที่รับมาได้ และกำหนดให้ RxEn มีค่าเป็นลอจิก "1" เพื่อเป็นสัญญาณกระตุ้นให้ภาคอื่นๆ รับทราบว่าได้รับข้อมูลมาครบ 8 บิตแล้ว หลังจากนั้นก็จะกลับมา State 1 เพื่อรอรับข้อมูลอีกครั้ง

3.1.6 การนับตำแหน่งแอดเดรส (Address counter)

เนื่องจากหน่วยความจำความเร็วสูงมีขนาด 64 K x 8 และมีสายแสดงตำแหน่ง (Address) 16 เส้นโดยคิดจาก 2^{16} = 64 K ดังนั้นวงจรนับจึงนับถอยหลังจาก 0000 ถึง FFFF ซึ่งแสดงถึง 64 K ตำแหน่ง จึงใช้ตัวนับขนาด 16 บิต สายสัญญาณเอาท์พุทนี้จะต้องต่อกับสายสัญญาณตำแหน่งของหน่วยความจำความเร็วสูงโดยผ่านมัลติเพลกเซอร์ก่อน เมื่อสัญญาณนาฬิกาจากวงจรสร้างสัญญาณนาฬิกาเข้ามา ที่ขา CLK ของวงจรนับและขา CLR เป็นศูนย์แล้ว วงจรนับก็จะนับเพิ่มขึ้นเรื่อยๆจนถึง FFFF แล้วก็เริ่มวนนับใหม่ที่ 0000 เป็นเช่นนี้ไป

เรื่อยๆ จนกว่าจะมีการเปลี่ยนแปลงสัญญาณนาฬิกาดังนั้นเราจึงสามารถควบคุมวงจรนับได้จากสัญญาณขา CLR และ CLK ของรูปที่ 3-11



รูปที่ 3-11 แสดงวงจรนับตำแหน่งแอดเดรส

โดยวงจรควบคุมวงจรนับนั้นเป็นวงจรตรวจสอบสภาวะการทำงานของลอจิกแอนนาไลเซอร์ จะไปเคลียร์วงจรนับไม่ให้นับ จนกว่าจะมีสัญญาณทริกจากภายนอก เข้ามากระตุ้นให้เริ่มเขียนข้อมูลลงไปในหน่วยความจำ ดังนั้นลอจิกแอนนาไลเซอร์เริ่มสุ่มค่าสัญญาณ และเมื่อทริกวงจรนับทำงานแล้ว วงจรตรวจสอบสถาวะการทำงานก็ตรวจสอบว่านับถึง FFFF แล้วหรือยังถ้ายังนับไม่ถึง FFFF ก็นับต่อไป แต่ถ้านับถึง FFFF วงจรตรวจสอบนี้ก็จะให้สัญญาณมาควบคุมวงจรนับให้หยุดนับและส่งสัญญาณไปบอกเพื่อส่งข้อมูลให้กับคอมพิวเตอร์ต่อไป

3.1.7 การจัดการข้อมูล

เมื่อมีการสุ่มค่าสัญญาณ และเก็บค่าเหล่านั้นลงในหน่วยความจำจนเต็มแล้ว ก็จะเป็นการจัดการนำเอาข้อมูลที่มีอยู่ในหน่วยความจำ มาแสดงผล โดยมีไมโครโปรเซสเซอร์เป็นตัวจัดการข้อมูล

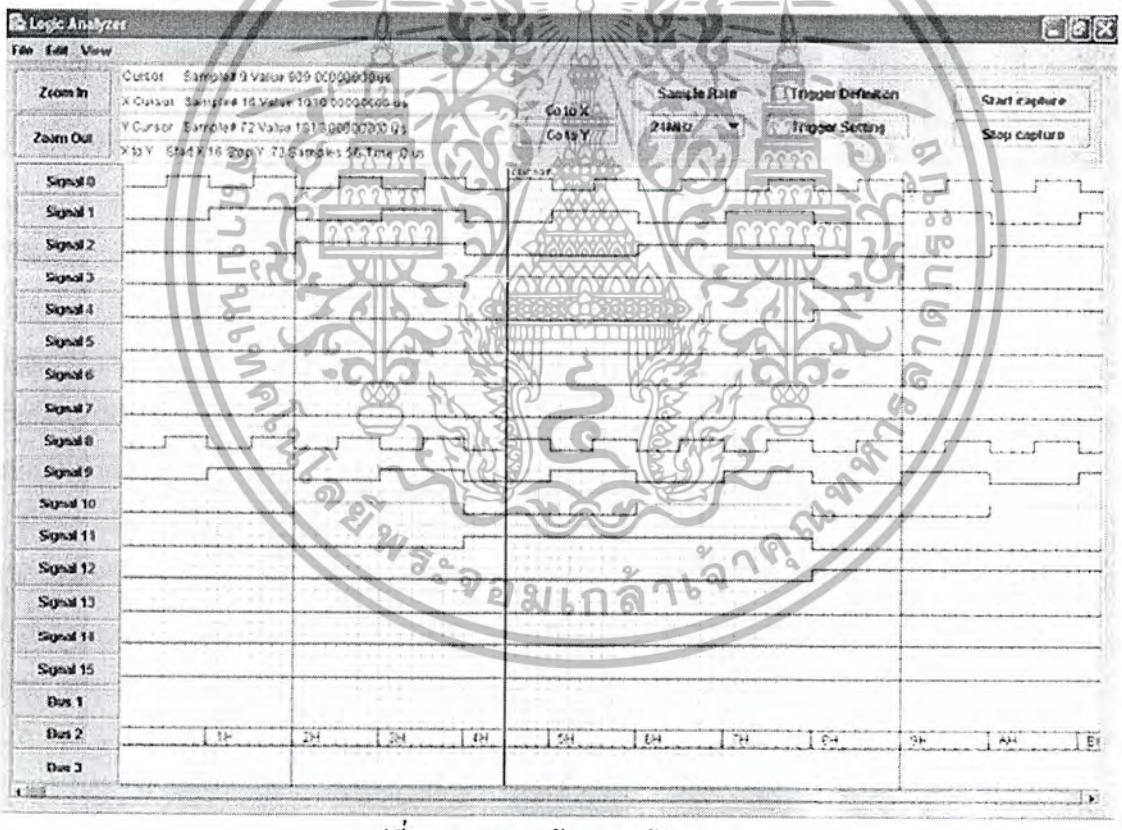
เมื่อต้องการแสดงผล คอมพิวเตอร์จะส่งสัญญาณมาบอกไมโครโปรเซสเซอร์ ไมโครโปรเซสเซอร์ก็จะอ่านข้อมูลจากหน่วยความจำความเร็วสูงในแต่ละตำแหน่งแล้วส่งไปให้คอมพิวเตอร์ โดยการอ่านข้อมูลจากตำแหน่งสูงถึงตำแหน่งสุดท้าย การอ่านข้อมูลจากหน่วยความจำจะเป็นรับข้อมูลแบบขนาน และจะส่งไปให้คอมพิวเตอร์แบบอนุกรม

3.1.7 การแสดงผล

ในส่วนนี้เป็นซอฟต์แวร์ (Software) เมื่อคอมพิวเตอร์รับข้อมูลจากเครื่องลอจิกแอนนาไลเซอร์เนื่องจากรับข้อมูลมาเป็นแบบขนาน ขนาด 16 บิตโดยแต่ละบิตคือค่าแต่ละช่องสัญญาณ ดังนั้นจึงต้องนำค่ามาต่อกัน โดยนำข้อมูลจากตำแหน่งต่ำสุดจนไปถึงสูงสุด ซึ่งผลที่ได้จะเป็นเส้นไดอะแกรม (Diagram) แสดงผลออกทางคอมพิวเตอร์

3.2 แนวทางการออกแบบทางซอร์ฟแวร์

หน้าที่ของส่วนโปรแกรม คือ แสดงผลสัญญาณดิจิตอลที่เครื่องวัดวัดได้ และส่งเข้ามาทางพอร์ตอนุกรม เมื่อโปรแกรมได้รับค่าจะทำการเก็บค่าสัญญาณไว้ในตัวแปรที่มีชื่อว่าเวคเตอร์ และส่วนวาดรูปสัญญาณจะทำการอ่านค่าที่เก็บไว้ในตัวแปรเวคเตอร์ ผ่านผ่านฟังก์ชั่นวาดกราฟออกทางหน้าจอ โดยการแสดงผลออกทางหน้าจอ ตัวโปรแกรมจะมีฟังก์ชั่นต่างๆ คอยจัดการการแสดงผลนั้นๆ เพื่อเพิ่มความสะดวกให้กับผู้ใช้งาน เช่น การอ่านค่าจากเคอร์เซอร์เมาส์เพื่อคำนวณค่าออกเป็นค่าฐาน 16, แสดงตำแหน่งของลูกสัญญาณที่เคอร์เซอร์ชี้ในขณะนั้นว่า เป็นลูกสัญญาณที่เท่าใด, การวัดระยะห่างของเส้นเคอร์เซอร์ทั้ง 2 เส้น, การแสดงผลแบบขยายเข้า(zoom in) ขยายออก(zoom out) ของลูกคลื่นสัญญาณ, การสลับเส้นสัญญาณในการแสดงผล, การแสดงผลแบบจับกลุ่ม, การตั้งค่าทริกเกอร์ เพื่อเริ่มรับค่า, ตั้งค่าการแสดงผลแบบรูปคลื่น หรือการแสดงผลแบบตรรกะ, การค้นหารูปแบบลูกคลื่นที่ได้รับเข้ามาแล้ว, การตั้งค่าสภาพแวดล้อมในการแสดงผล, การเก็บบันทึกข้อมูลที่ได้วัดแล้ว, การโหลดข้อมูลที่เคยเก็บบันทึกไว้ และการพิมพ์ลูกคลื่นออกทางเครื่องปริ๊นท์

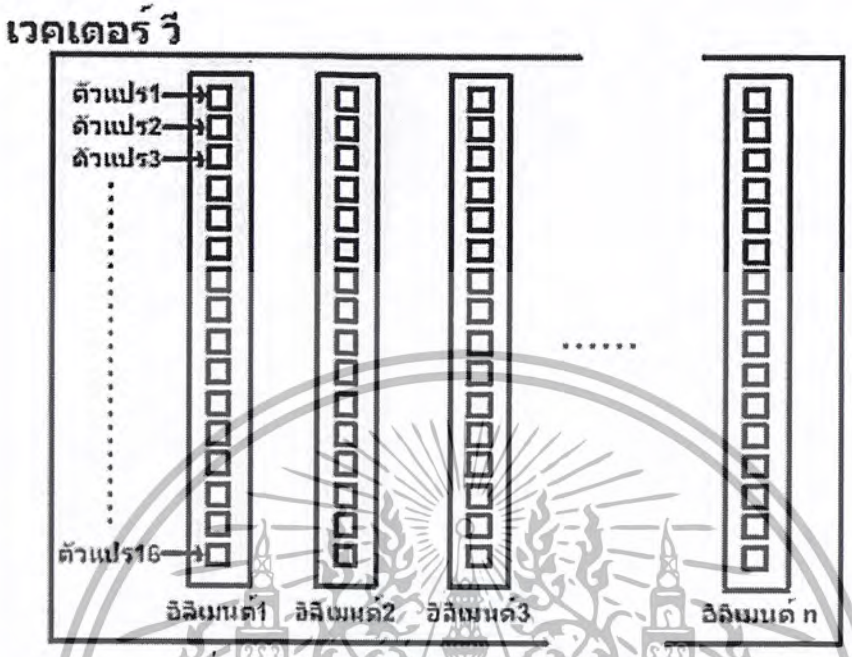


รูปที่ 3-12 แสดงหน้าตาของโปรแกรม

จากรูปแสดงหน้าจอหลักของส่วนโปรแกรม โดยตัวโปรแกรมจะแสดงรูปคลื่นที่ได้รับจากเครื่องวัดออกมาแสดงผล ผ่านฟังก์ชั่นต่างๆ โดยการทำงานของฟังก์ชั่นต่างๆจะอธิบายดังต่อไปนี้

3.2.1 ส่วนเก็บค่าสัญญาณ

การเก็บค่าสัญญาณที่ได้รับจากพอร์ตอนุกรมนั้นจะใช้ตัวแปรที่เรียกว่า "เวคเตอร์" โดยจะใช้ชื่อว่า เวคเตอร์ วี



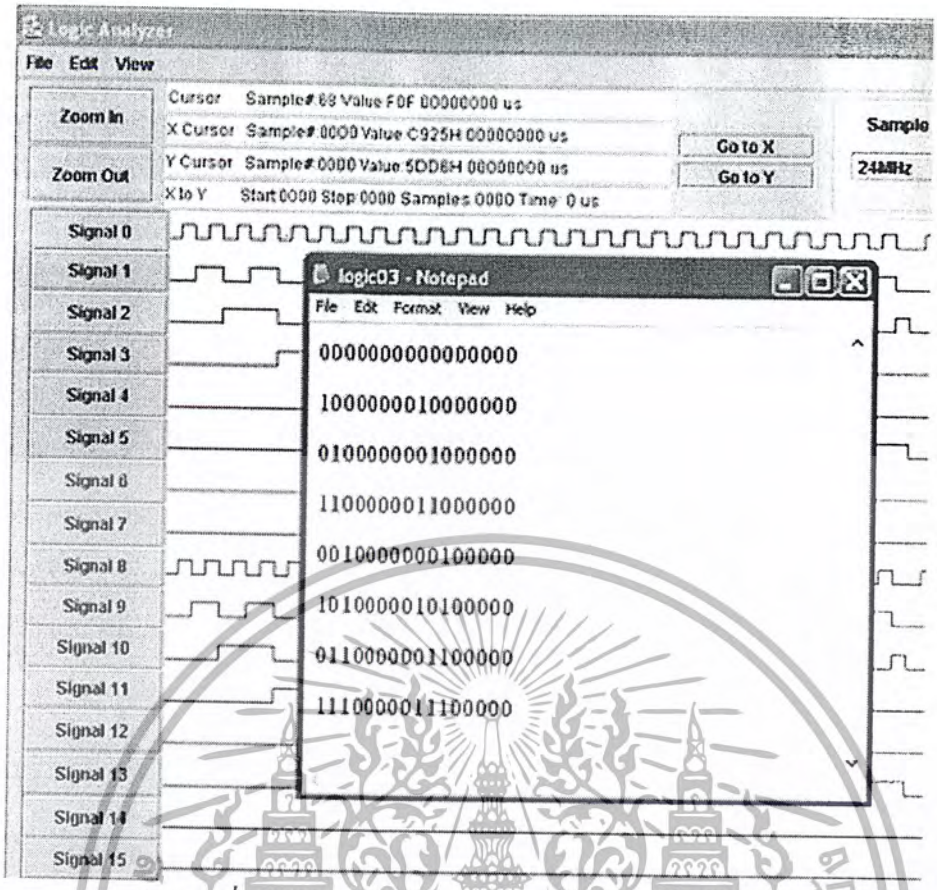
รูปที่ 3-13 รูปแสดงโครงสร้างการเก็บค่าสัญญาณ

แสดงรูปแสดงโครงสร้างการเก็บค่าสัญญาณที่รับจากพอร์ตอนุกรมโดยเก็บลงในตัวแปรเวคเตอร์ ซึ่งในแต่ละอิลิเมนต์ของเวคเตอร์ จะเก็บค่าสัญญาณจำนวน 16 ตัวแปร ตัวแปร 1 ตัวจะแทนสถานะ "0" หรือ "1" ของ 1 ช่องสัญญาณ ใน 1 ช่วงเวลา เหตุที่ใช้ตัวแปรประเภทเวคเตอร์เก็บค่าสัญญาณเพราะว่า ตัวแปรเวคเตอร์จะเป็นตัวแปรที่คล้ายกับตัวแปรประเภทอะเรย์ แต่ตัวแปรประเภทอะเรย์นั้น จะต้องประกาศขอบเขต หรืออิลิเมนต์ในแน่ชัด แต่ในตัวแปรประเภทเวคเตอร์นั้น ไม่จำเป็นจะต้องประกาศขอบเขตขนาดของเวคเตอร์ ขนาดของเวคเตอร์สามารถเพิ่มจำนวนของอิลิเมนต์ไปได้เรื่อยๆ และยังสามารถลบอิลิเมนต์ หรือ แทรกค่าลงไปในตำแหน่งของอิลิเมนต์ ได้

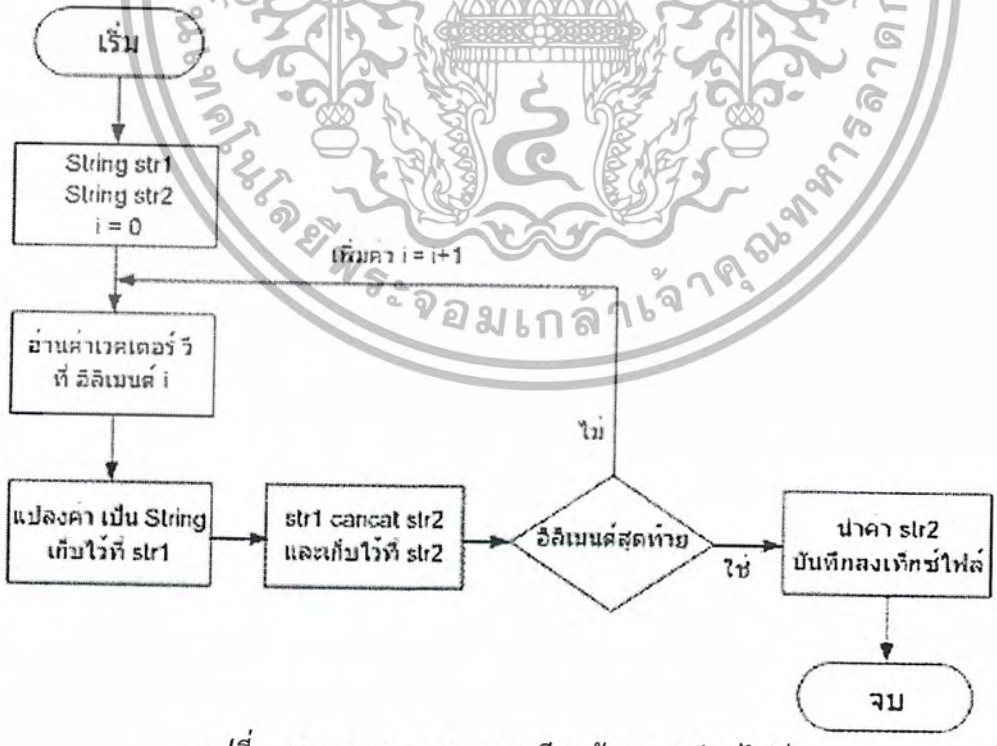
3.2.2 ส่วนเก็บบันทึกข้อมูลลงเท็กซ์ไฟล์

ในส่วนนี้จะทำการเก็บค่าข้อมูลสัญญาณดิจิตอลที่ได้วัดเข้ามาแล้ว และจะทำการบันทึกออกเป็นเท็กซ์ไฟล์ เพื่อที่จะสามารถเปิดดูข้อมูลที่เคยวัดไปแล้วได้อีกครั้ง การเก็บลงเท็กซ์ไฟล์จะเก็บเรียงบรรทัดละ 16 ตัวอักษร แทนค่า"0"และ"1" ในแต่ละ 1 ช่วงเวลา

จากรูปจะแสดงให้เห็นรูปแบบการเขียนข้อมูลลงเท็กซ์ไฟล์ โดยในช่องคาบเวลาที่1 ตัวโปรแกรมจะไปดูค่าข้อมูลจากเวคเตอร์ วี ในอิลิเมนต์ที่ 0 จากรูปในอิลิเมนต์ที่ 0 เก็บค่า "0000000000000000" ดังนั้นจึงเขียนลงเท็กซ์ไฟล์ในบรรทัดแรกมีค่า "0000000000000000" จากโปรแกรมในคาบเวลาที่2 โปรแกรมจะไปอ่านค่าในเวคเตอร์ วี ในอิลิเมนต์ที่ 1 จากรูปมีค่า "1000000010000000" จึงเขียนค่าลงเท็กซ์ไฟล์ในบรรทัดที่ 2 ว่า "1000000010000000" และจากโปรแกรมในคาบเวลาที่3 โปรแกรมจะไปอ่านค่าในเวคเตอร์ วี ในอิลิเมนต์ที่ 2 จากรูปมีค่า "0100000001000000" จึงเขียนค่าลงเท็กซ์ไฟล์ในบรรทัดที่ 3 ว่า "0100000001000000" การทำงานจะเป็นเช่นนี้ไปเรื่อยๆจนกว่าจะถึงคาบเวลาสุดท้าย



รูปที่ 3-14 แสดงรูปแบบการเขียนข้อมูลลงเท็กซ์ไฟล์



รูปที่ 3-15 แสดง flowchart การเขียนข้อมูลลงเท็กซ์ไฟล์

3.3.3 ส่วนโหลดข้อมูลที่เคยเก็บบันทึกไว้จากเท็กซ์ไฟล์

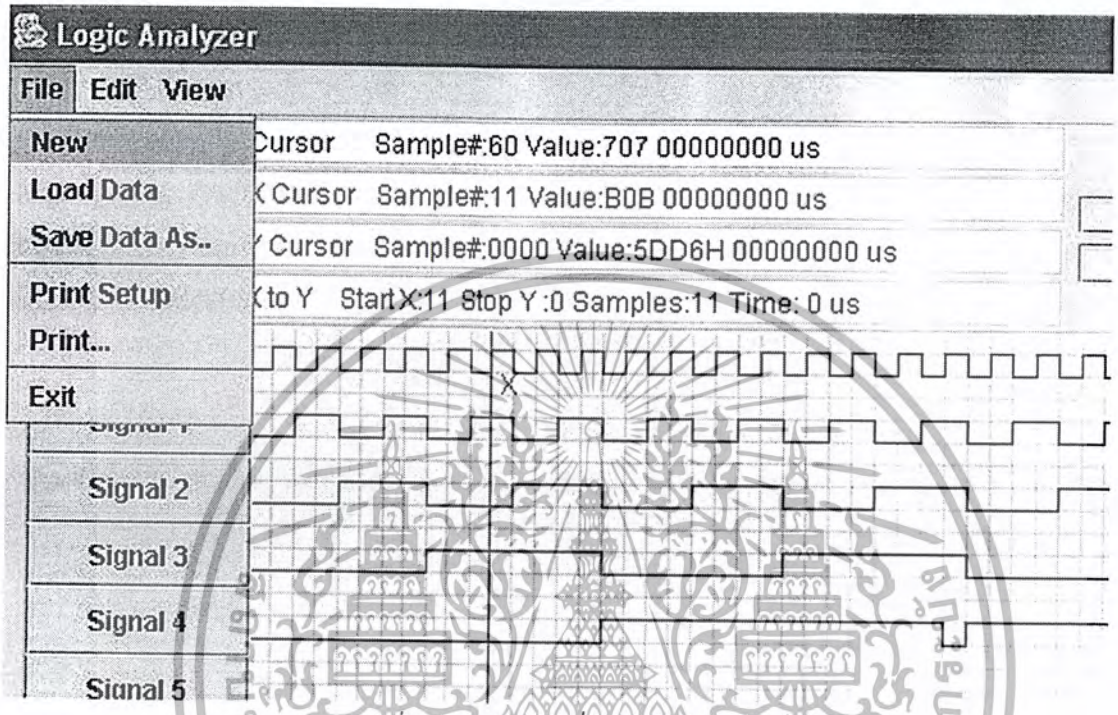
ในครั้งแรกหลังจากเลือกเท็กซ์ไฟล์ที่เก็บข้อมูลขึ้นมาแล้ว โปรแกรมจะทำการโหลดค่าจากเท็กซ์ไฟล์เป็นค่าข้อมูลทั้งหมดที่มีอยู่ในเท็กซ์ไฟล์มาเก็บไว้ในตัวแปรสตริง แต่ก่อนที่จะเก็บค่าข้อมูลสตริงทั้งหมดนี้ลงในตัวแปรเวคเตอร์ซึ่งเป็นตัวเก็บค่าสัญญาณดิจิตอลที่อยู่ในโปรแกรมนั้น เราจะต้องแบ่งค่าสตริงออกเป็นส่วนย่อยๆ โดยแบ่งออกเป็น ส่วนๆ ส่วนละ16ตัวอักษร และนำค่า 16 ตัวอักษรแปลงเป็นค่าบูลลีน และส่งค่าทั้ง 16 ตัว นี้เข้าไปในแต่ละอิลิเมนต์ของเวคเตอร์ วี การทำงานอธิบายดังรูป ดังนี้

รูปที่ 3-16 แสดง flowchart การโหลดข้อมูลที่เคยเก็บบันทึกไว้จากเท็กซ์ไฟล์

โดยการโหลดข้อมูลนั้นตัวโปรแกรมจะไปเปิดเท็กซ์ไฟล์ที่ได้ทำการเลือกไว้ ซึ่งเมื่อเปิดไฟล์ตัวโปรแกรมจะทำการอ่านค่าทั้งหมดที่อยู่ในเท็กซ์ไฟล์ไปเก็บไว้ และการที่จะนำข้อมูลที่อ่านขึ้นมาได้นำไปแสดงผลนั้นจะต้องส่งข้อมูลไปเก็บไว้ในตัวแปรเวคเตอร์ วี ก่อน ซึ่งการเก็บค่าลงในเวคเตอร์ วี จะต้องเก็บลงครั้งละ 16 ค่าสัญญาณ ต่อ 1 อิลิเมนต์ของเวคเตอร์ การที่จะแบ่งสตริงที่อ่านมาได้จากเท็กซ์ไฟล์นั้น ในครั้งแรกโปรแกรมจะเริ่มอ่านจากตำแหน่งที่ 0 ของสตริงและไปถึงตำแหน่งที่ 15 ของสตริง โปรแกรมก็จะได้ค่าสตริงตัวใหม่ที่มีจำนวน 16 ตัวอักษร หลังจากนั้นโปรแกรมจะต้องแปลงค่าสตริงนี้ให้เป็นค่าบูลลีนเพื่อที่จะส่งไปเก็บในอิลิเมนต์ของเวคเตอร์ และเมื่อแปลงและเก็บค่าลงในเวคเตอร์แล้ว ครั้งต่อไปโปรแกรมจะต้องมาอ่านค่าสตริงตัวต่อไปหลังจาก 16 ตัวแรกที่ได้อ่านไปแล้ว แต่ข้อมูลที่เก็บอยู่ในเท็กซ์ไฟล์ เมื่อจบ 16 ตัวอักษรแล้วจะมีการขึ้นบรรทัดใหม่ ดังนั้นการอ่านค่า 16 ตัวอักษรถัดไป จะต้องบวกค่าตำแหน่งไปอีก 2 ตำแหน่ง ดังนั้นค่าตำแหล่งที่จะอ่านต่อไปคือ 18 ถึง 34 เพื่อให้ได้ค่าสตริง16 ตัวอักษรถัดไป การทำงานจะเป็นเช่นนี้ไปเรื่อยๆ จนกระทั่งจบไฟล์

3.3.4 ส่วนเริ่มค่าการวัดใหม่

ในส่วนนี้ในกรณีที่ผู้ใช้ได้เคยทำการวัดค่าสัญญาณไปแล้ว ทำให้มีรูปสัญญาณค้างอยู่ที่หน้าจอ และผู้ใช้อยากที่จะเริ่มทำงานใหม่โดยเริ่มทำการวัดใหม่ หรือต้องหารที่จะลบข้อมูล,ที่ได้เคยวัดไปแล้วก็สามารถลบข้อมูลที่เคยวัดไปแล้วได้

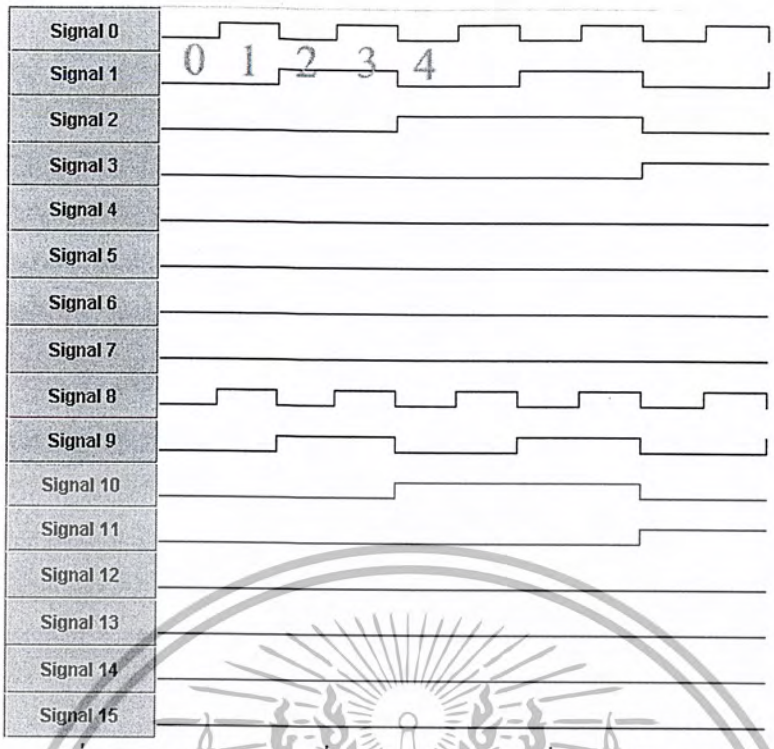


รูปที่ 3-17 แสดงการเริ่มค่าการวัดใหม่

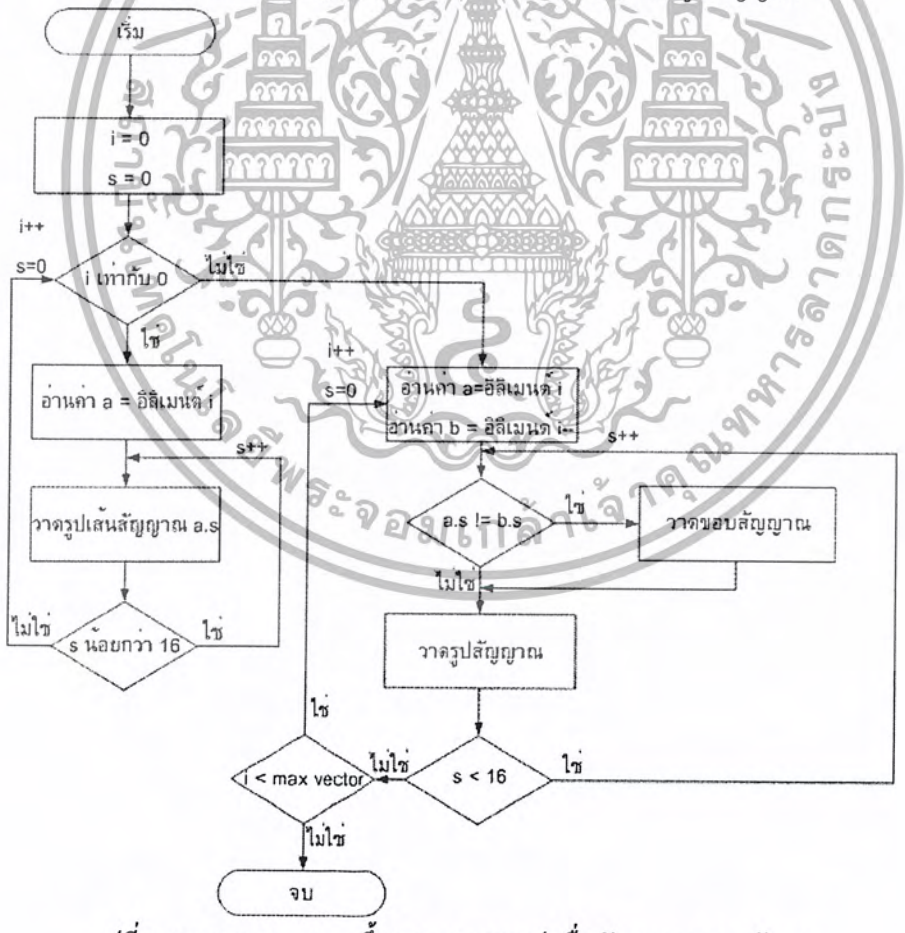
โดยทำการเลือกที่ File>New โดยเมื่อผู้ใช้ต้องการที่จะลบค่าที่เคยวัดไปแล้ว ภายในโปรแกรมจะทำการเข้าไปจัดการกับตัวแปรเวคเตอร์ วี ซึ่งเป็นตัวเก็บค่าต่างๆที่เคยได้ทำการวัดไปในครั้งล่าสุด

3.3.5 ส่วนวาดรูปสัญญาณ

ในส่วนนี้โปรแกรมจะไปอ่านค่าสัญญาณที่เก็บไว้ในตัวแปรเวคเตอร์ วี และผ่านฟังก์ชั่นและเงื่อนไขต่างๆหลังจากนั้นจะส่งค่าตำแหล่ง x และ y ไปยังฟังก์ชั่นวาดรูปสัญญาณ



รูปที่ 3-18 แสดงรูปสัญญาณที่ผ่านกาดวาดโดยฟังก์ชั่นวาดรูปสัญญาณ

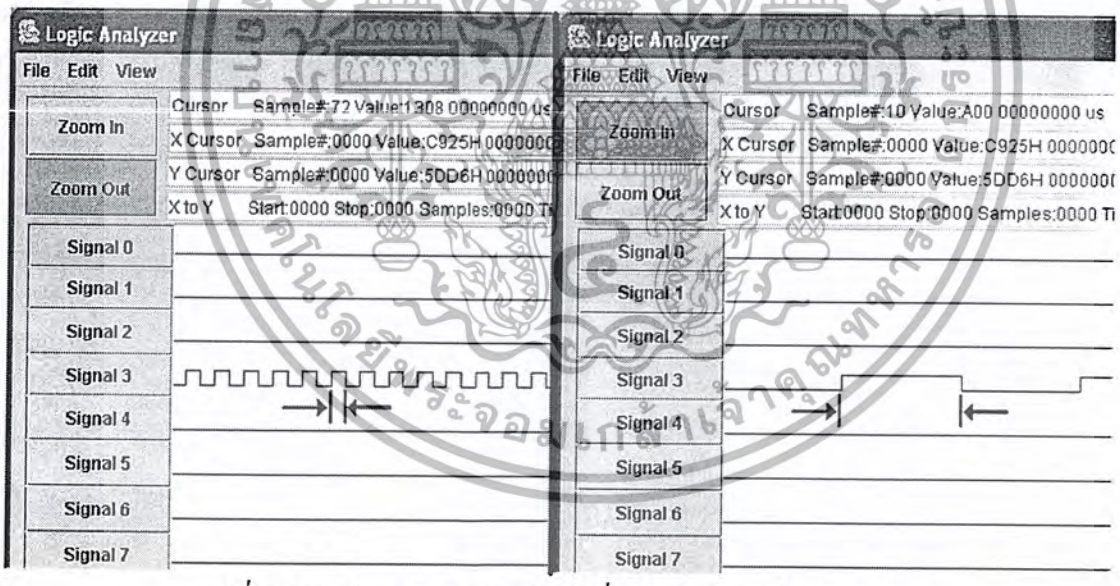


รูปที่ 3-19 แสดง flowchart ขั้นตอนการวาดรูปคลื่นสัญญาณออกหน้าจอ

จากรูป โปรแกรมจะเริ่มต้นอ่านข้อมูลในอิลิเมนต์ 0 ของเวคเตอร์ จึงตั้งค่าให้ i = 0 และถ้า i = 0 แล้ว จะให้ค่า a เก็บค่าที่อยู่ในอิลิเมนต์ 0 ทั้งหมด ซึ่งค่าที่อยู่ในอิลิเมนต์นั้นจะเป็นค่าอาร์เรย์16ค่า จึงอ้างค่าได้อีกเป็น a.0 – a.15 ซึ่งจะวนอ่านออกมาทีละ 1 ค่า เพื่อนำมาเปรียบเทียบว่าค่านั้นเป็นค่า “1” หรือค่า “0” เมื่อรู้ว่าเป็นค่า “1” หรือค่า “0” แล้วก็วาดเส้นตามตำแหน่ง x และ y ที่กำหนด เมื่อโปแกรมวนอ่านค่าทั้งหมด 16 รอบแล้ว ก็จะเพิ่มค่า i++ ดังนั้นจะมาโปรเซสทางด้านขวามือ เมื่อเริ่มโปรเซสทางด้านขวามือโปรแกรมจะเริ่มอ่านค่าในอิลิเมนต์ที่ 1 คือจะให้ a เก็บค่าในอิลิเมนต์ที่1 และให้ b เก็บค่าในอิลิเมนต์ที่แล้ว เพื่อที่ว่า จะได้นำมาเปรียบเทียบว่า ในอิลิเมนต์ที่1 ในอาร์เรย์ที่0 และในอิลิเมนต์ที่ 0 ในอาร์เรย์ที่0 นั้นมีค่าสัญญาณสูงต่ำเหมือนกันเหมือนไม่ซึ่งถ้าเหมือนกันโปรแกรมก็จะวาดรูปสัญญาณตามปกติ แล้วถ้ามีค่าไม่เหมือนกันแล้วโปรแกรมจะต้องวาดขอบขาขึ้นของสัญญาณด้วย ดังนั้นจึงจำเป็นที่จะต้องอ่านค่าทั้งในอิลิเมนต์ปัจจุบัน และในอิลิเมนต์ตัวที่แล้วด้วย หลังจากนั้นโปแกรมจะวนทำงานต่อไปเรื่อยๆ จนกว่าจะถึงอิลิเมนต์สุดท้ายของเวคเตอร์ วี ตัวโปรแกรมถึงจะหยุดการวาดรูปสัญญาณ

3.3.6 ส่วนย่อและขยายในการแสดงรูปสัญญาณ

โดยปกติแล้วโปแกรมจะวาดรูปสัญญาณ 1 ลูกคลื่นขนาด 10 พิกเซล แต่เพื่อความสะดวกของผู้ใช้ตัวโปแกรมสามารถย่อ และขยายการแสดงผลของรูปคลื่นสัญญาณได้ เป็น 1ลูกคลื่นขนาด 80พิกเซล

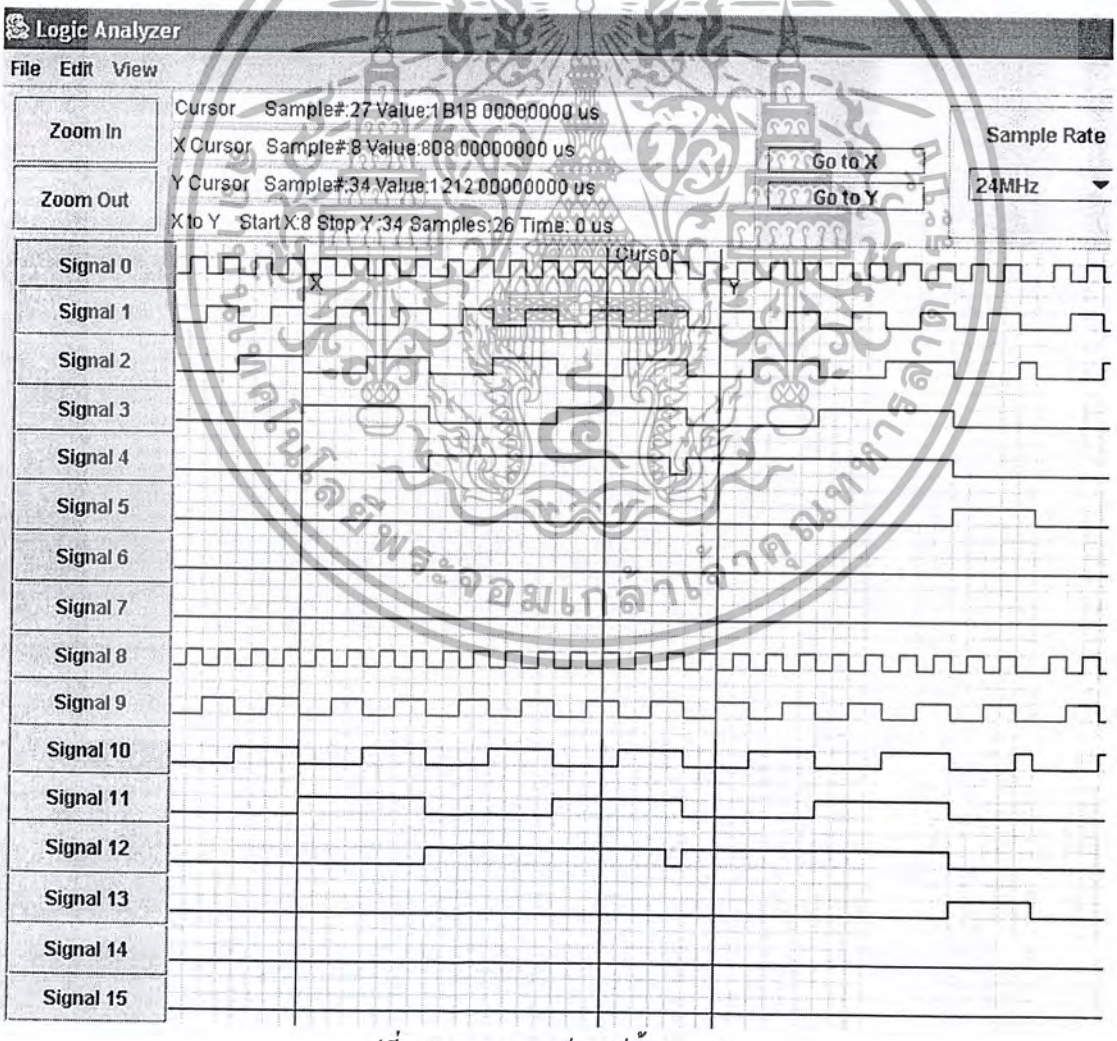


รูปที่ 3-20 แสดงการแสดงผลของรูปคลื่นสัญญาณแบบย่อ และขยาย

ในขั้นตอนของฟังก์ชันวาดรูปคลื่นสัญญาณตัวโปรแกรมได้มีตัวคูณขนาดของสัญญาณ 1 ลูกคลื่นซึ่งโดยปกติแล้วขนาดของสัญญาณ 1 ลูกคลื่นจะเท่ากับ 10 พิกเซลโดยให้ตัวคูณเท่ากับ 1 แต่ถ้าผู้ใช้ได้ขยาย(Zoom in) การแสดงผลตัวคูณจะกลายเป็น 2, 3, 4 ตามลำดับ ซึ่งขนาดของสัญญาณที่ขยายได้สุดคือ คูณ8เท่าของสัญญาณขนาดปกติ

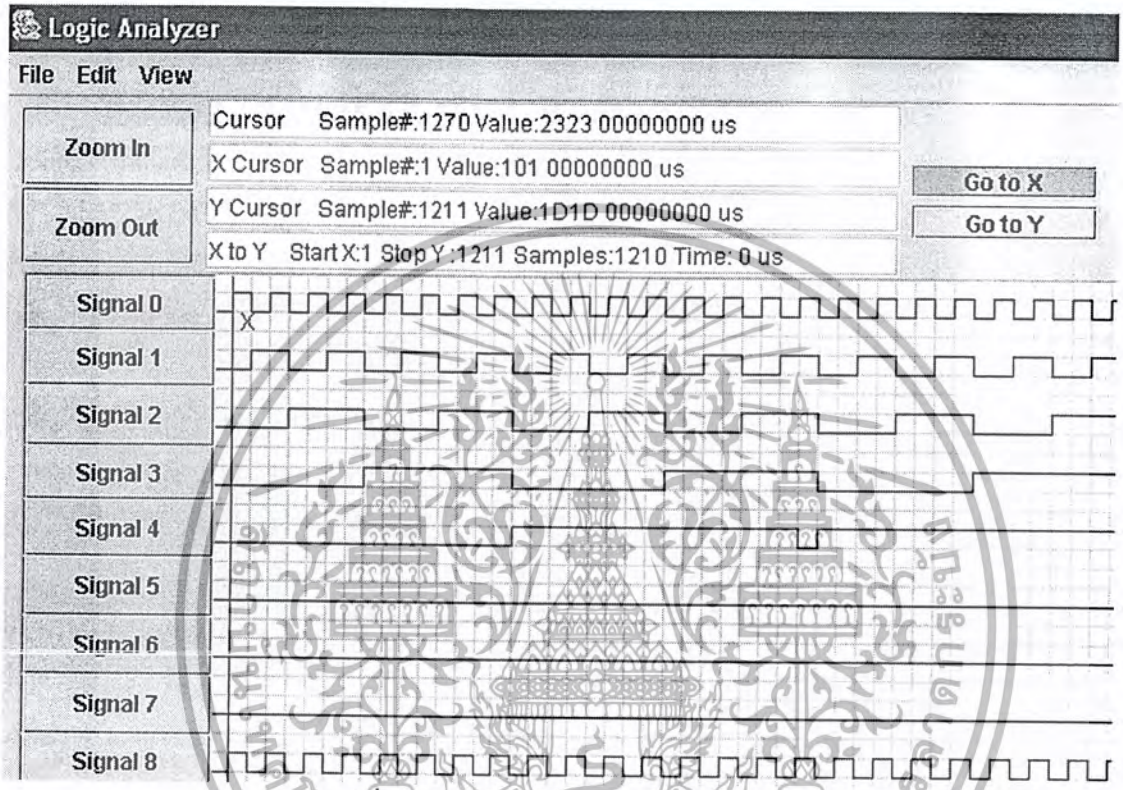
3.3.7 การอ่านค่าจากเคอร์เซอร์เมาส์เพื่อข้อมูลของสัญญาณ

ในโปรแกรมนี้จะมีเคอร์เซอร์อยู่ทั้งหมด 3 เคอร์เซอร์ คือ เคอร์เซอร์เมาส์ การเกิดของเคอร์เซอร์เมาส์คือเมื่อเมาส์เลื่อนเข้ามาอยู่ในพื้นที่แสดงรูปสัญญาณเคอร์เซอร์นี้จะเกิดขึ้น ,เคอร์เซอร์ที่ 2 คือ เคอร์เซอร์ x เป็นเคอร์เซอร์ที่เกิดการการคลิ๊กเมาส์ซ้ายในกรอบการแสดงรูปสัญญาณ ,เคอร์เซอร์ที่ 3 คือ เคอร์เซอร์ที่เกิดจากการคลิ๊กเมาส์ขวาในกรอบการแสดงรูปสัญญาณ โดยประโยชน์ของเคอร์เซอร์ คือ ในตำแหน่งของเคอร์เซอร์ใดเคอร์เซอร์หนึ่งจะบอกข้อมูลรายละเอียดของตำแหน่ง ว่า เคอร์เซอร์ได้อยู่ ณ.ตำแหล่งใดเช่น ดังรูปเคอร์เซอร์ x ได้อยู่ที่ตำแหน่งลูกคลื่นที่ 27 ลูกคลื่น 27 มีค่า ในฐาน 16 คือ 1B1BH และบอกเวลาที่ลูกคลื่นที่ 27 เกิด โดยเริ่มนับตั้งแต่ ลูกคลื่นที่ 0 และในเคอร์เซอร์ y และเคอร์เซอร์เมาส์ ก็จะมีข้อมูลแจ้งให้กับผู้ใช้ทราบเช่นกัน แต่จากที่ได้กล่าวไปว่า เมื่อคลิ๊กซ้ายและ ขวา ก็จะเกิดเคอร์เซอร์x และ เคอร์เซอร์ y เมื่อเคอร์เซอร์ x และ y เกิดแล้วใน textfield4 ตามรูปจะบอกข้อมูลว่า เคอร์เซอร์ x อยู่ที่ตำแหน่งที่27 คือวัดลูกคลื่นที่ 27 และเคอร์เซอร์ y อยู่ที่ตำแหน่งที่ 34 คือวัดลูกคลื่นที่34 และจะบอกว่าระยะห่างจากเคอร์เซอร์ x และเคอร์เซอร์ y ห่างกัน 26 ลูกคลื่น แล้วจะบอกช่วงเวลาระหว่างเคอร์เซอร์ x และเคอร์เซอร์ y ตามลำดับ



รูปที่ 3-21 แสดงเคอร์เซอร์ทั้ง 3 แบบ

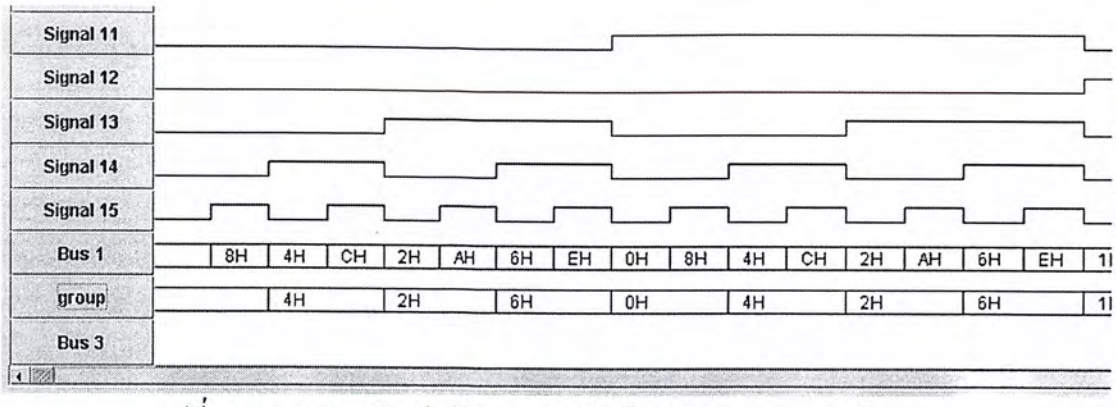
ในกรณีที่เราคลิ๊กซ้ายเพื่อให้เกิดเคอร์เซอร์ x ถ้าถ้าผู้ใช้เลื่อนหน้าจอให้ไปแสดงผลในคลื่นสัญญาณอื่นๆ เช่นเคอร์เซอร์ x อยู่ที่ตำแหน่ง 100 แต่ในขณะที่หน้าจอแสดงสัญญาณดิจิตอลในตำแหน่งที่ 2000 และผู้ใช้ต้องการให้กลับมาแสดงยังที่ตำแหน่งเคอร์เซอร์ x อยู่นั้น ผู้ใช้ก็แค่เพียงคลิ๊กที่ปุ่ม Go to X ตัวโปรแกรมก็จะไปอ่านที่ที่เคอร์เซอร์ x อยู่ และจะเปลี่ยนค่า jSbar ให้มีค่าเท่ากับตำแหน่งที่เคอร์เซอร์ x อยู่ ทำให้หน้าจอกระโดดกลับไปแสดงยังตำแหน่งที่เคอร์เซอร์ ดังรูป



รูปที่ 3-22 แสดงการใช้งานฟังก์ชั่น Go to X

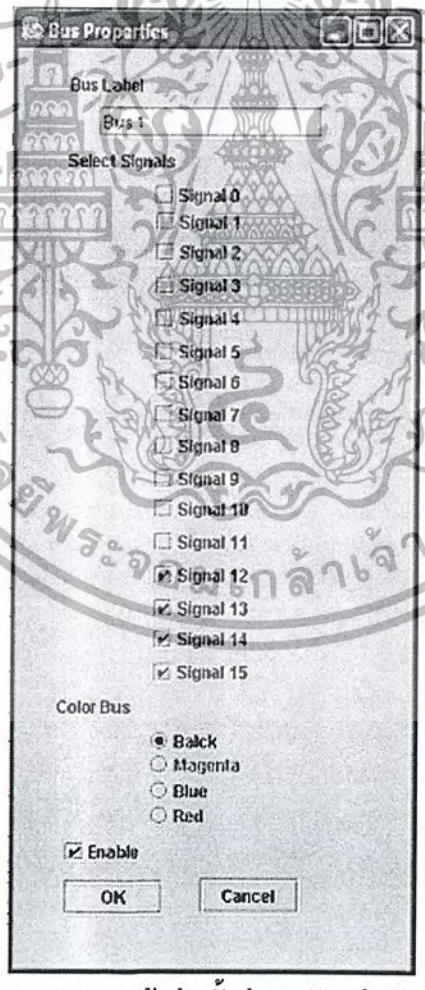
3.3.8 ส่วนจัดกลุ่มสัญญาณ

ในส่วนจัดกลุ่มสัญญาณจะเป็นการรวมเอาหลายๆสัญญาณมาแสดงผลร่วมกัน โดยจะแสดงค่าของการรวมออกเป็นเลขฐาน 16 โดยเงื่อนไขในการจัดกลุ่มผู้ใช้สามารถตั้งค่าได้ว่าต้องการที่จะนำสัญญาณในช่องไหนมาจัดกลุ่มรวมกันบ้างโดยจะต้องมีตั้งแต่อย่างน้อยที่สุดคือ 2 สัญญาณ และมากที่สุดคือรวมทุกสัญญาณเข้ามาอยู่ในกลุ่มเดียวกันคือ 16 ช่องสัญญาณ, สามารถตั้งค่าชื่อของกลุ่มที่จะจัดได้, ตั้งค่าสีของสัญญาณว่าจะให้แสดงผลออกเป็นสีตามที่ตั้งหรือไม่, ตั้ง Enable หรือ Disable เพื่อเปิด-ปิดกรแสดงผลของกลุ่มนั้นๆ



รูปที่ 3-23 แสดงการจัดกลุ่มสัญญาณ 2 กลุ่ม คือ กลุ่ม Bus 1 และ กลุ่ม group

จากรูปเป็นการแสดงผลแบบจัดกลุ่ม 2 กลุ่ม โดยมีกลุ่ม Bus 1 รวมสัญญาณ Signal 12, Signal 13, Signal 14, Signal 15 เข้าด้วยกัน จะเห็นว่าถ้า ช่องสัญญาณ Signal 12, Signal 13, Signal 14, Signal 15 ช่องใดช่องหนึ่งมีการเปลี่ยนแปลงค่า Bus1 จะทำการแสดงค่าที่รวมจากทั้ง 4 ช่องสัญญาณออกมาเป็นค่าฐาน 16 ออกหน้าจอ และอีกลุ่มคือกลุ่มชื่อ group ได้รวมสัญญาณ Signal 12, Signal 13, Signal 14 เข้าด้วยกัน

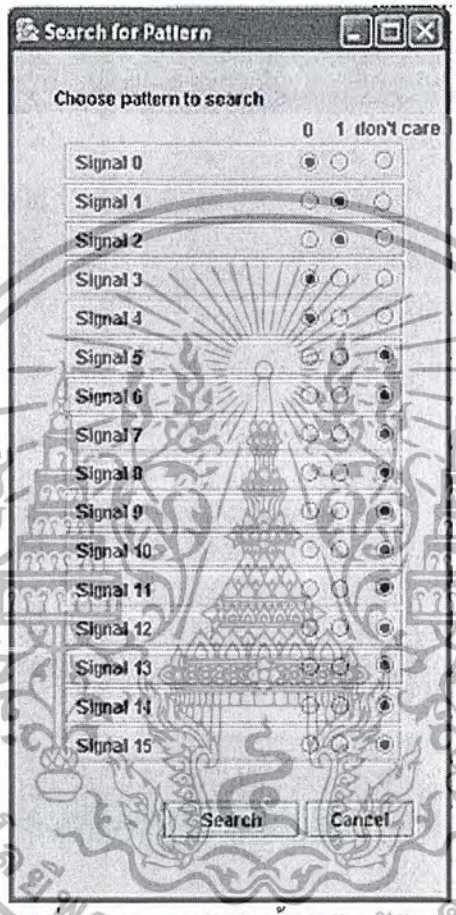


รูปที่ 3-24 แสดงหน้าต่างตั้งค่าการจัดกลุ่มสัญญาณ

จากรูปเป็นการตั้งค่ากลุ่มที่ให้ชื่อกลุ่มว่า Bus 1 และรวมเอาช่องสัญญาณ Signal 12, Signal 13, Signal 14, Signal 15 รวมไว้ด้วยกัน และให้แสดงผลสีของกลุ่มออกเป็นสีดำ

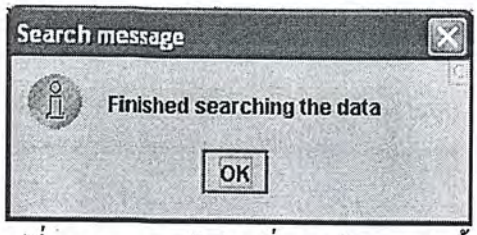
3.3.9 ส่วนค้นหาข้อมูล

ตัวโปรแกรมสามารถค้นหารูปแบบข้อมูลที่เคยรับเข้ามาจากเครื่องวัดได้ โดยผ่านฟังก์ชั่นค้นหาโดยฟังก์ชั่นค้นหานี้จะให้ผู้ใช้ตั้งค่าที่ต้องการค้นหา ว่าเป็น "0" หรือ "1" หรือจะไม่สนใจได้



รูปที่ 3-25 แสดงหน้าต่างตั้งค่าค้นหาข้อมูล

จากรูป เป็นการตั้งค่าค้นหาข้อมูล 01100xxxxxxxxxx โดย x หมายถึงไม่สนใจ จะเป็น "0" หรือ "1" ก็ได้ เมื่อตั้งค่าและกดปุ่ม search แล้ว ฟังก์ชั่นค้นหาข้อมูลจะไปอ่านค่าจากตัวแปรเวคเตอร์ วี และตรวจสอบว่าตรงตามเงื่อนไขที่ตั้งไว้หรือไม่ ถ้าตรงตามเงื่อนไข ฟังก์ชั่นจะนำตำแหน่งของอิลิเมนต์ที่ตรงตามเงื่อนไขไปตั้งค่าของ jSbar เพื่อที่จะให้โปแกรมกระโดดไปแสดงผลในตำแหน่งที่เจอข้อมูล แล้วตรงตามเงื่อนไขของการค้นหา และเมื่อกด search อีกครั้งฟังก์ชั่นจะทำการค้นหาต่อจากเดิมและถ้าตรงตามเงื่อนไขก็จะกระโดดไปแสดงผลยังตำแหน่งดังกล่าว และถ้าฟังก์ชั่นทำการค้นหาจนถึงตำแหน่งสุดท้ายของอิลิเมนต์แล้วจะแสดงผลดังนี้

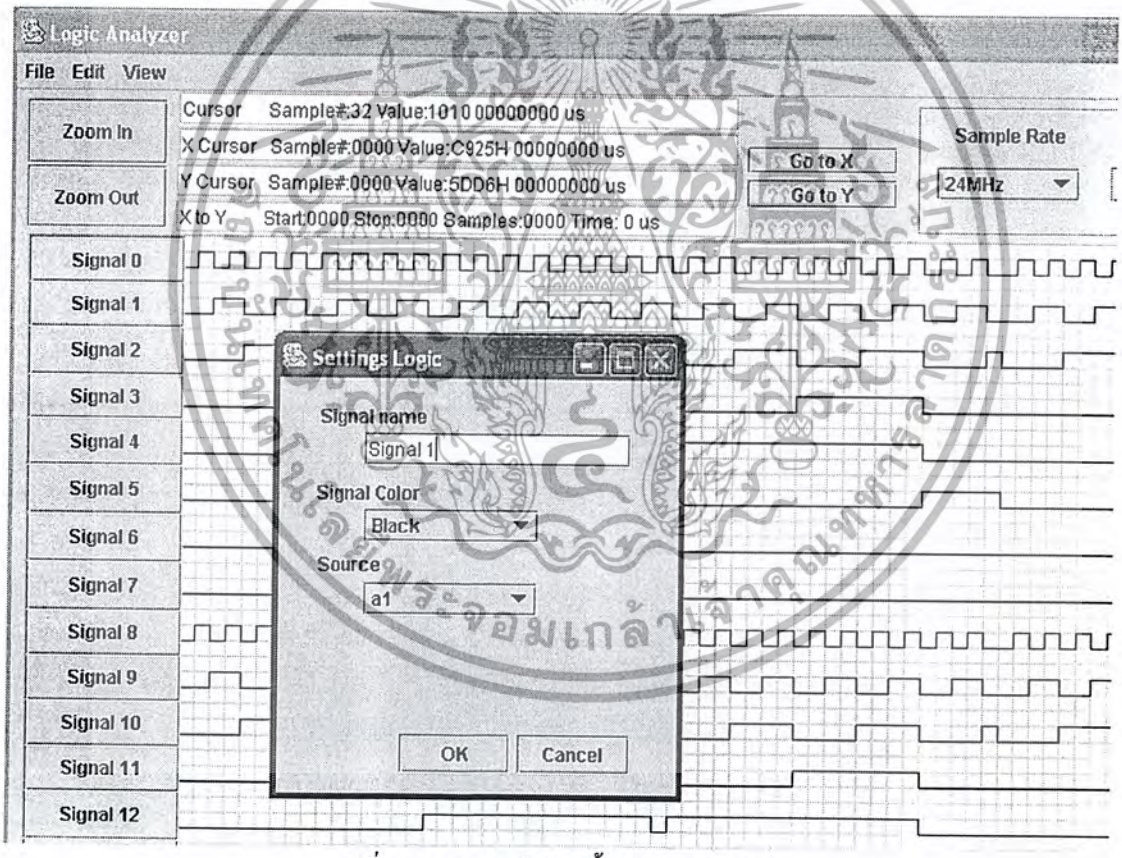


รูปที่ 3-26 แสดงหน้าต่างเมื่อการค้นหาเสร็จสิ้น

3.3.10 ส่วนตั้งค่าให้กับช่องสัญญาณ

โปรแกรมนี้จะแสดงค่าสัญญาณเป็นจำนวน 16 ช่องสัญญาณ โดยที่ผู้ใช้สามารถตั้งค่าต่างๆให้กับแต่ละช่องสัญญาณได้ ดังนี้

- ตั้งชื่อช่องสัญญาณ
- ตั้งค่าสีให้การแสดงผลในแต่ละช่องสัญญาณ
- ตั้งค่าแหล่งสัญญาณของช่องสัญญาณ



รูปที่ 3-27 แสดงหน้าต่างตั้งค่าช่องสัญญาณ

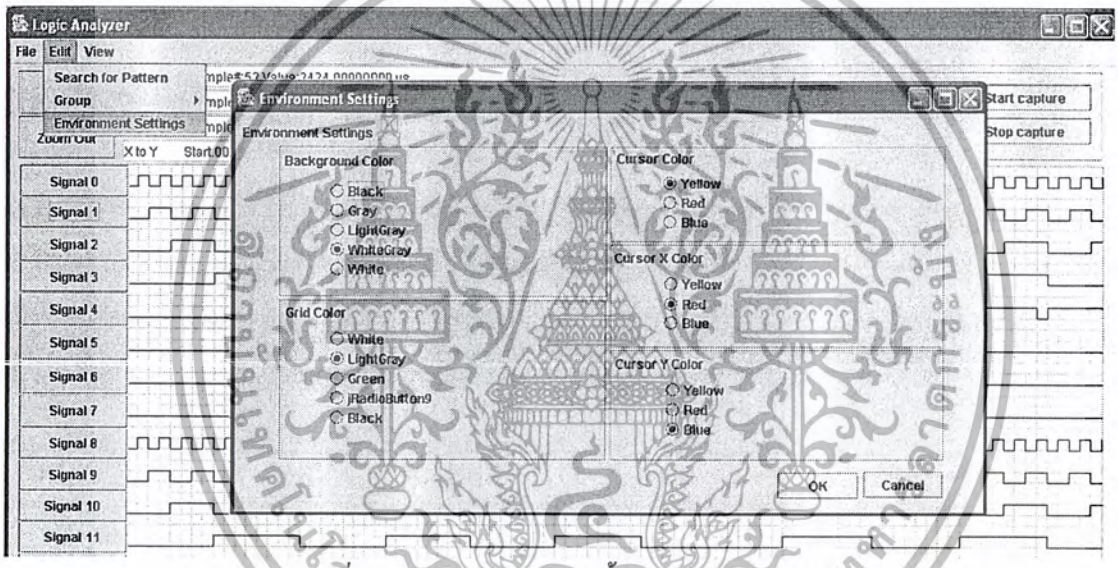
จากรูป เมื่อผู้ใช้คลิ๊กที่ปุ่มของช่องสัญญาณโปแกรมจะแสดงหน้าต่างเพื่อให้ผู้ใช้สามารถตั้งค่าต่างๆของช่องสัญญาณนั้นๆได้ จากรูปเป็นการตั้งค่าให้กับช่องสัญญาณช่องที่1 โดยให้ชื่อช่องสัญญาณว่า Signal 1 แล้วตั้งค่าสีให้กับช่องสัญญาณเป็นสีดำ แล้วแหล่งสัญญาณมาจาก a1 เราสามารถตั้งค่าใหม่โดยเอาตั้งค่าแหล่งสัญญาณมาจาก a0 – a15 ได้ เพื่อที่กรณีที่เวลาเครื่องวัดของเราสลับสายกันไม่เรียงตามลำดับ ผู้ใช้ก็ไม่จำเป็น

จะต้องไปสลับสาย ผู้ใช้เพียงแต่ตั้งค่าภายในโปรแกรมว่าจะให้ช่องสัญญาณดังกล่าว ไปอ่านค่ามาจากสายสัญญาณเส้นใดจากเครื่องวัด

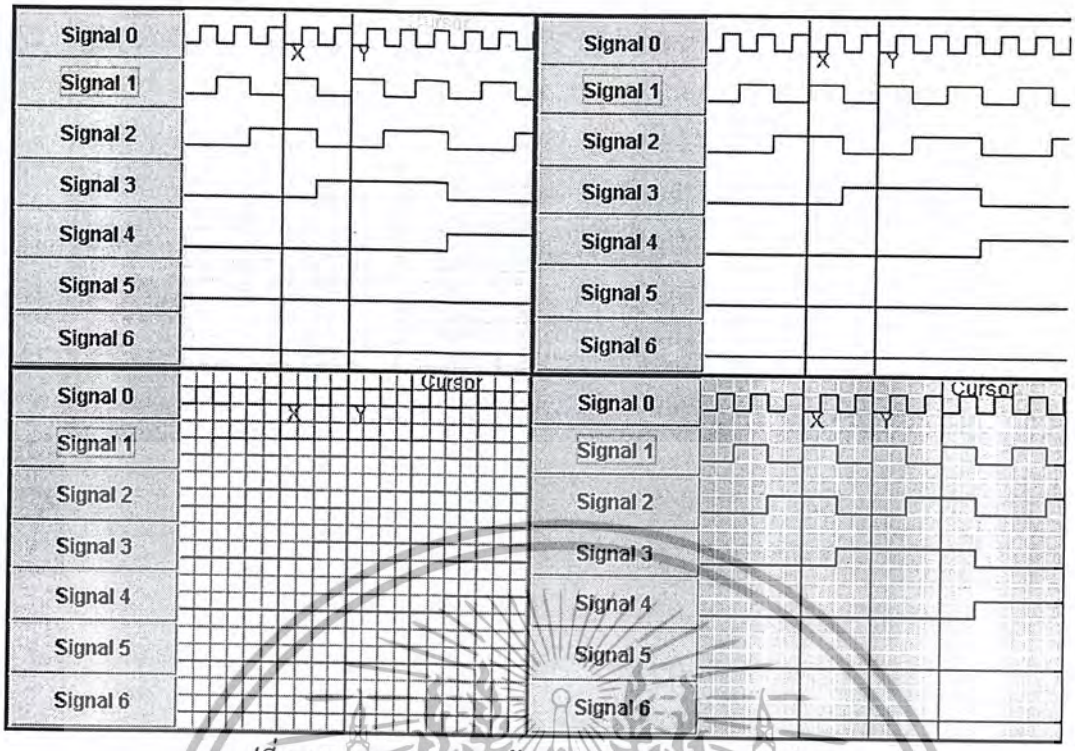
3.3.11 ส่วนตั้งค่าสภาพแวดล้อมในการแสดงผล

โปรแกรมสามารถตั้งค่าสภาพแวดล้อมในการแสดงผลในส่วนต่างๆ โดยการตั้งค่าสีต่างๆเพื่อความพอใจของผู้ใช้ โดยส่วนที่สามารถตั้งค่าได้มีดังนี้

- ตั้งค่าพื้นหลัง
- ตั้งค่ากริด
- ตั้งค่าเคอร์เซอร์เมาส์
- ตั้งค่าเคอร์เซอร์ x
- ตั้งค่าเคอร์เซอร์ y



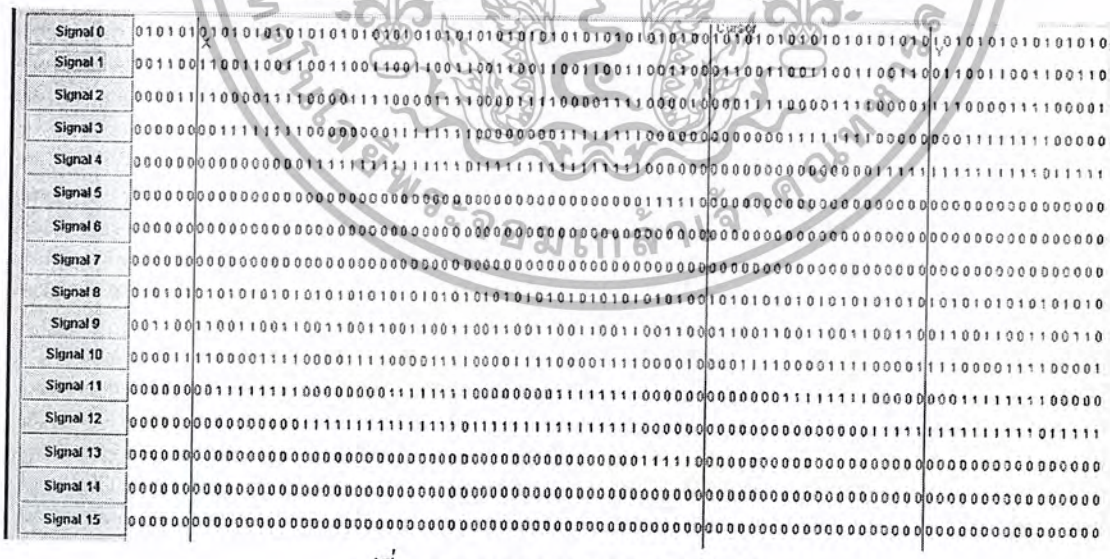
รูปที่ 3-28 แสดงหน้าต่างการตั้งค่าสภาพแวดล้อม



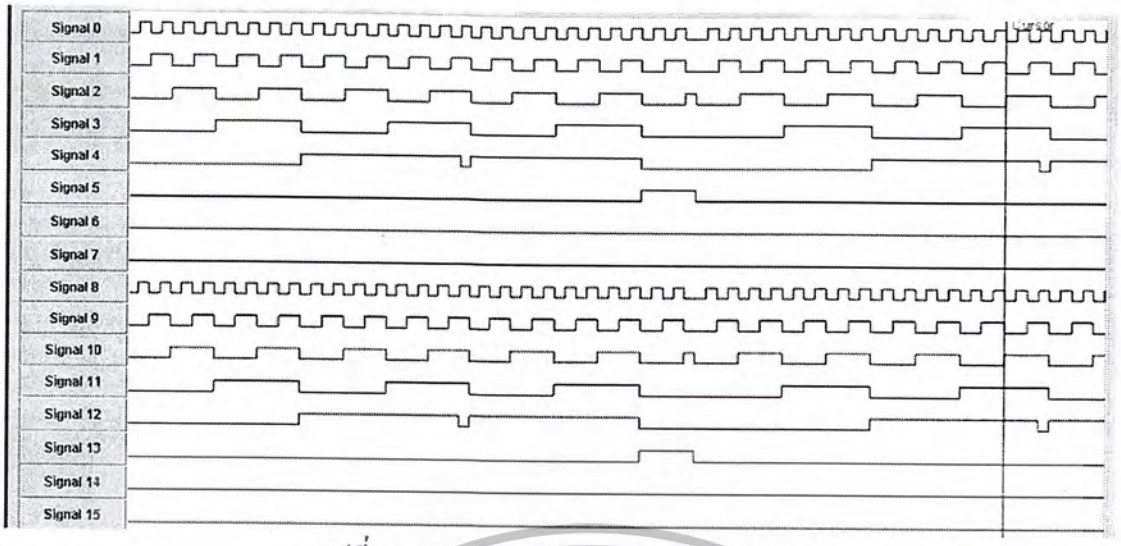
รูปที่ 3-29 แสดงสภาพแวดล้อมของการแสดงผลแบบต่างๆ

3.3.12 ส่วนแสดงผลรูปสัญญาณและแสดงผลตัวเลข

โดยปกติแล้วตัวโปรแกรมจะแสดงผลการวัดสัญญาณออกเป็นรูปคลื่นสัญญาณหรือแบบรูปสัญญาณนั้นเอง แต่ตัวโปแกรมเองยังสามารถแสดงผลออกเป็นตัวเลขทางดิจิตอล หรือแสดงผลออกเป็นเลข “0” หรือ “1” ได้ โดยตั้งค่าให้ตัวโปรแกรมแสดงผลเป็นแบบ List



รูปที่ 3-30 แสดงการแสดงผลแบบตัวเลข

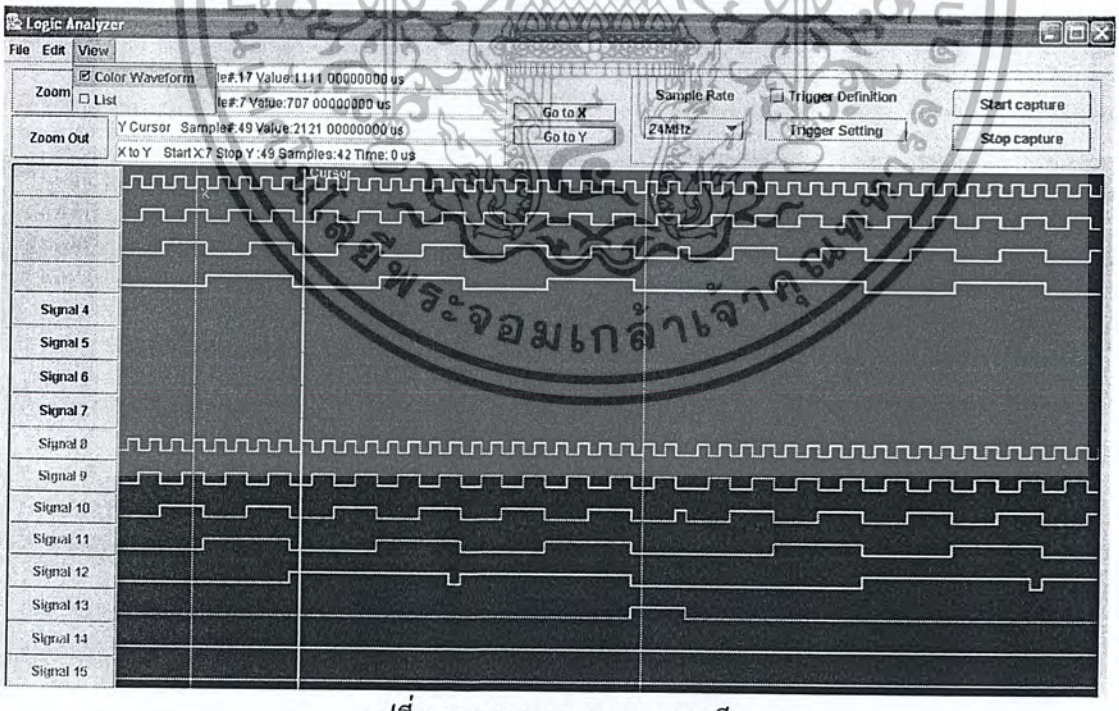


รูปที่ 3-31 แสดงการแสดงผลแบบรูปสัญญาณ

แต่การแสดงผลแบบตัวเลขจะไม่สามารถขยาย หรือลดขนาดของการแสดงผลได้เหมือนกับการแสดงผลแบบรูปสัญญาณ แต่ฟังก์ชั่นอย่างอื่นเช่น ค้นหาข้อมูล, ส่วนตั้งค่าให้กับช่องสัญญาณ, ตั้งค่าสภาพแวดล้อมการแสดงผล ยังสามารถใช้ได้กับการแสดงผลแบบตัวเลขได้

3.3.13 ส่วนการแสดงผลแบบสี

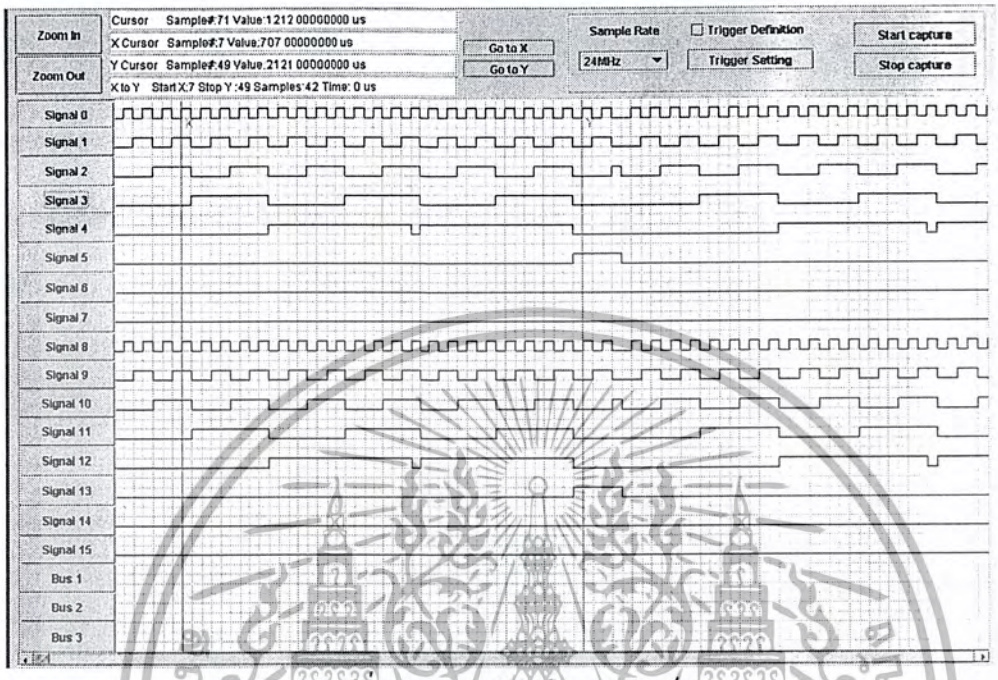
ส่วนนี้เราสามารถตั้งค่าให้แต่ละสัญญาณมีค่าการแสดงผลแต่ละสีได้ ซึ่งโดยปกติจะแสดงผลได้แค่สีดำเพียงสีเดียว ซึ่งถ้าให้แสดงผลแบบสีแล้วจะช่วยให้ผู้ใช้สะดวกต่อการแยกกลุ่มได้ง่ายยิ่งขึ้น



รูปที่ 3-32 แสดงการแสดงผลแบบสี

3.3.14 ส่วนพิมพ์ข้อมูล

โปรแกรมสามารถพิมพ์รูปสัญญาณลงบนในกระดาษโดยรูปสัญญาณที่ถูกพิมพ์ออกไปนั้นจะเป็นรูปสัญญาณที่กำลังแสดงอยู่ในหน้าตอนที่สั่งคำสั่งพิมพ์เท่านั้น



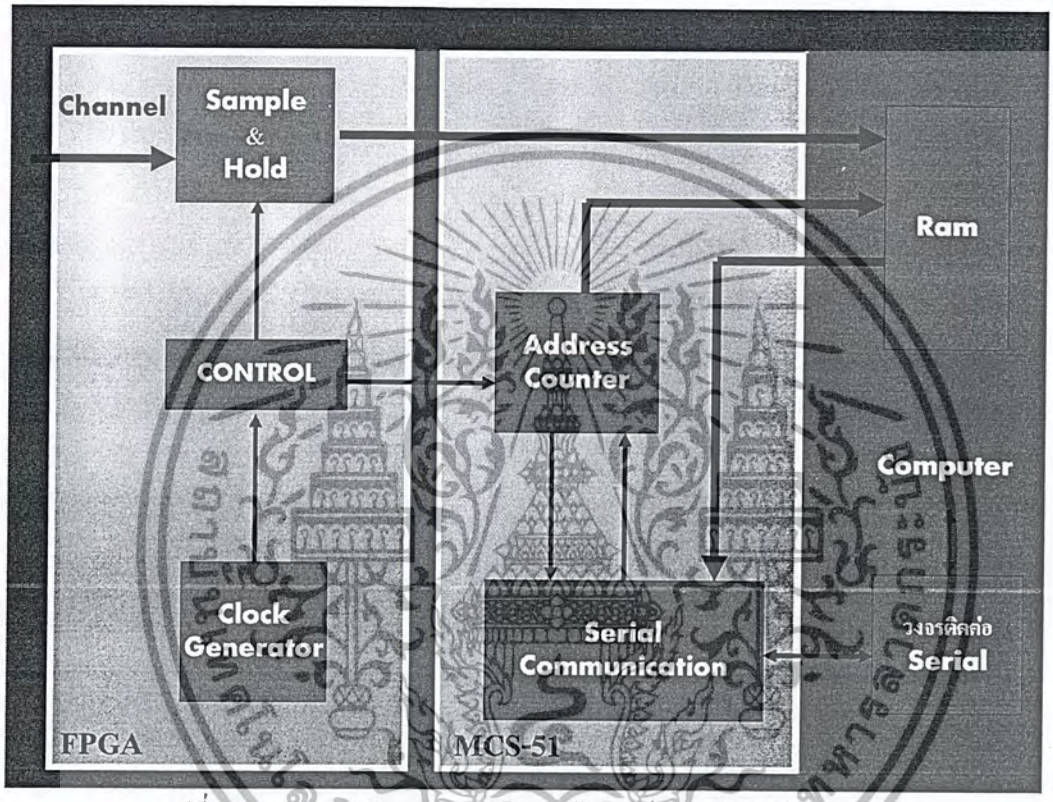
รูปที่ 3-33 แสดงตัวอย่างงานพิมพ์

บทที่ 4

ผลการทดลอง/การทดสอบ

4.1 การทดสอบ

จากการทศสอบตามทฤษฎีที่ที่ได้ออกแบบไว้ในบทที่ 3 ในส่วนของฮาร์ดแวร์ไม่สามารถทำวงจร การติดต่อกับคอมพิวเตอร์ และวงจรในภาคของหน่วยความจำจึงได้ทำการย้ายการทำงานทั้ง 2 ส่วนนี้มาทำงานบนไมโครคอนโทรเลอร์ เบอร์ 89C51 แทนดังรูปที่ 4-1



รูปที่4-1 แสดงภาพบล็อกของลอจิกแอนนาไลเซอร์จริงหลังทำการทศสอบ

จากรูปจะเห็นว่าการทำงานในส่วนของวงจรนับตำแหน่งแอดเดรส (Address counter) และวงจรติดต่อกับคอมพิวเตอร์ (Serial Communication) ย้ายมาทำงานบนไมโครคอนโทรเลอร์ จึงทำให้เครื่องลอจิกแอนนาไลเซอร์ชุดนี้มีอัตราการสุ่มและคงค่า (Sampling and Hold) ลดลงหลักการทำงานก็คือ เมื่อ FPGA ทำการสุ่มและคงค่า เข้ามาก็จะส่งสัญญาขัดจังหวะ (Interrupt) มาให้ MCS-51 ทำการนับตำแหน่งแอดเดรสเพื่อทำการเก็บข้อมูลที่ได้ลงหน่วยความจำเมื่อข้อมูลที่ทำการสุ่มและคงค่าเข้ามาเต็มหน่วยความจำ MCS-51 ก็จะส่งสัญญาณไปบอกวงจรควบคุมการทำงานของ FPGA ให้ยุดการสุ่มและคงค่าข้อมูลเข้ามาเสียก่อน และทำการส่งค่าข้อมูลที่ได้ในหน่วยความจำให้กับคอมพิวเตอร์ จนเสร็จก่อนถึงเริ่มสุ่มและคงค่าสัญญาณชุดต่อไปความเร็วในการทำงาน ของเครื่องลอจิกแอนนาไลเซอร์ชุดนี้ลดลงเพราะต้องรอการทำงานของ MCS-51

4.2 วิธีการทดลอง

ทดลองโดยเขียนไมโครคอนโทรเลอร์ MCS-51 ผลิตสัญญาณนับขึ้นโดยเป็นสัญญาณที่มีความถี่คงที่ขนาด 10 มิลิวินาที ใช้ลอจิกแอนนาไลเซอร์วัดสัญญาณจากไมโครคอนโทรเลอร์ 2 พอร์ตรวมเป็น 16 ช่องสัญญาณ

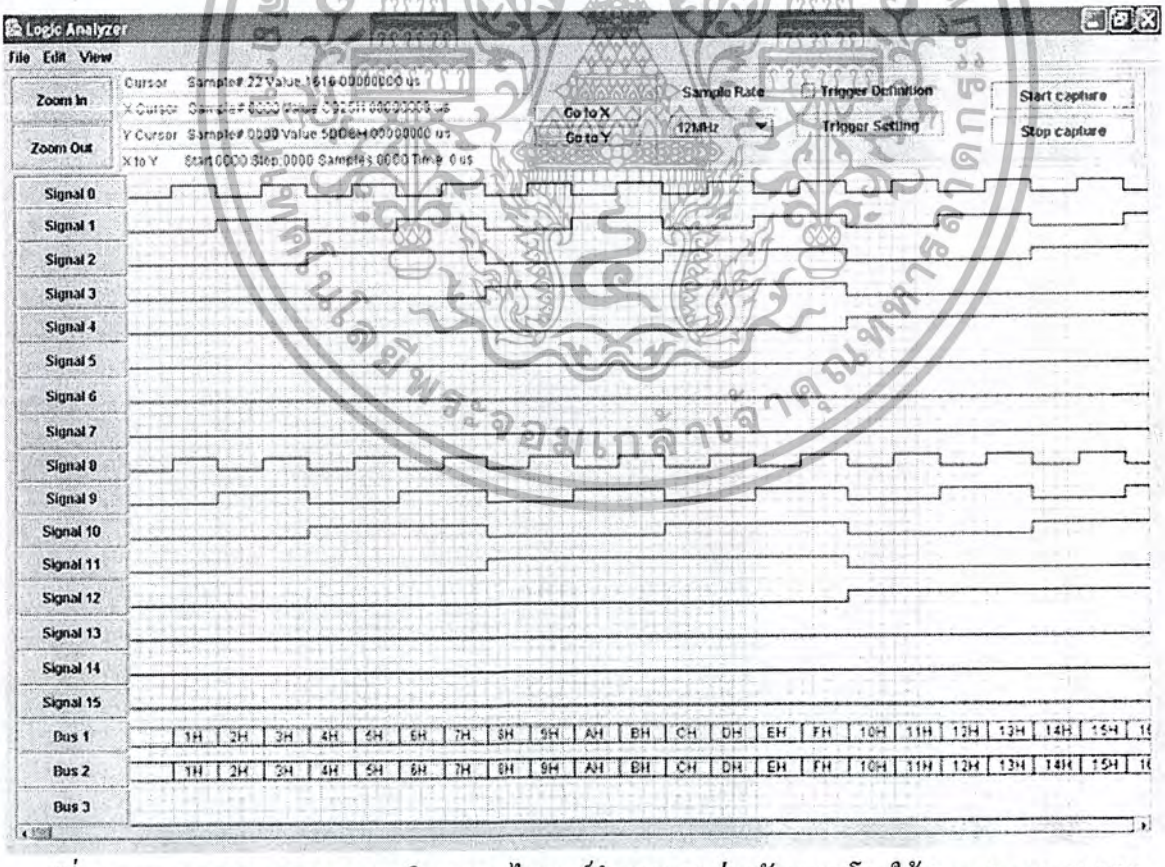
โดยวัดที่ใช้ความถี่ในการสุ่มสัญญาณ 12MHz และ 1MHz โดยไมโครคอนโทรเลอร์ผลิตสัญญาณความถี่ดังนี้

- สัญญาณที่มีความถี่คงที่ขนาด 1 MHz
- สัญญาณที่มีความถี่คงที่ขนาด 500 kHz
- สัญญาณที่มีความถี่คงที่ขนาด 100 kHz
- สัญญาณที่มีความถี่คงที่ขนาด 50 kHz

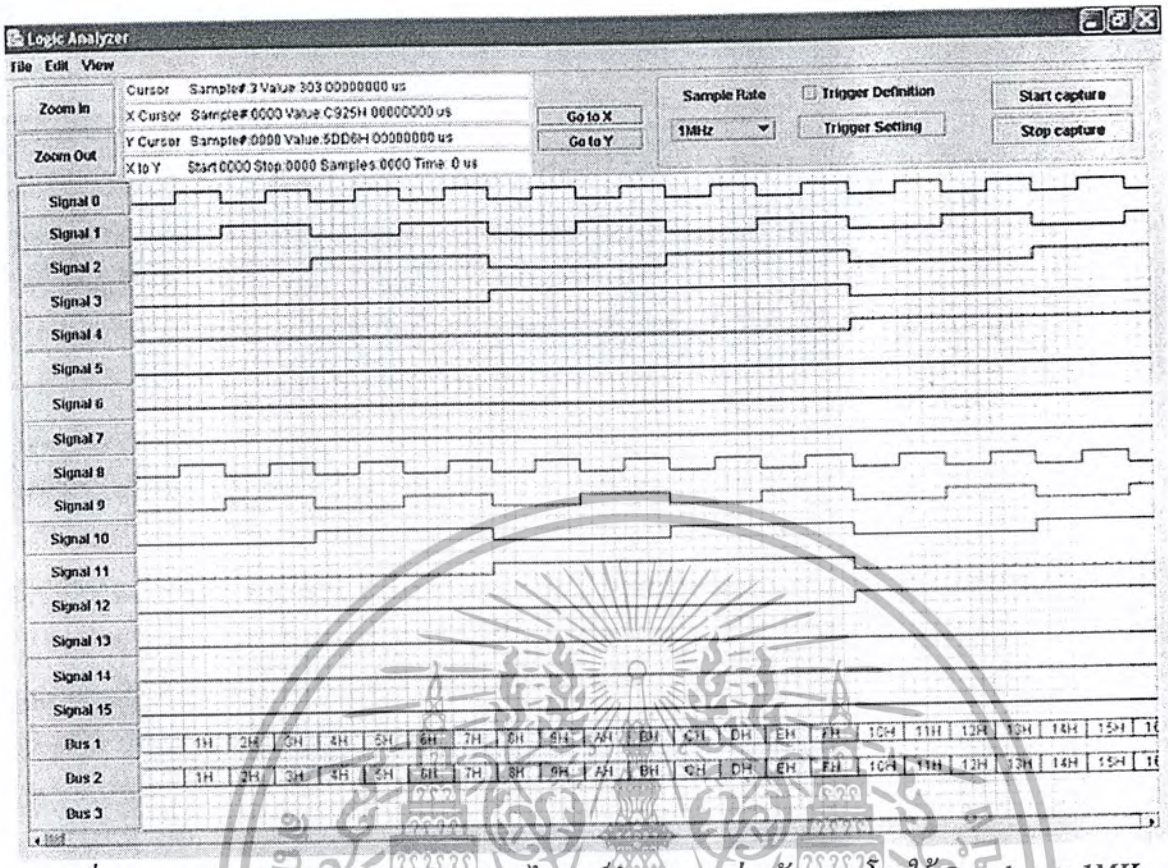
4.3 ผลการทดลอง

ผลการทดลองที่ได้แสดงบนหน้าจอจากโปรแกรมดังรูปที่ 4-2 และ 4-3 ได้ผลลัพธ์ตรงกับโปรแกรมที่เขียนโดยไมโครคอนโทรเลอร์ MCS-51 และใช้ ลอจิกแอนนาไลเซอร์วัดสัญญาณที่ได้ก็ได้ตรงกับที่วัดโดยเครื่องลอจิกแอนนาไลเซอร์ชุดนี้

สรุปได้ว่าเครื่องลอจิกแอนนาไลเซอร์ชุดนี้ สามารถตรวจรับข้อมูลได้ถูกต้องตามทฤษฎีที่ได้ออกแบบไว้โดยสามารถวัดที่ความถี่ของไมโครคอนโทรเลอร์ได้อย่างดีคือมีความถี่ไม่เกิน 20 MHz



รูปที่4-2 แสดงผลการทดลองของลอจิกแอนนาไลเซอร์จำนวน 16 ช่องสัญญาณโดยใช้ Sample rate 12MHz



รูปที่4-3 แสดงผลการทดลองของลอจิกแอนนาไลเซอร์จำนวน 16 ช่องสัญญาณโดยใช้ Sample rate 1MHz

บทที่ 5
บทวิจารณ์และสรุป

5.1 บทวิจารณ์

จากการทดลอง เมื่อนำลอจิกแอนนาไลเซอร์วัดสัญญาณดิจิตอลใดๆ ที่มีความถี่ไม่เกิน 12 เมกกะเฮิทซ์ ลอจิกแอนนาไลเซอร์สามารถวัดและแสดงผลบนจอภาพได้อย่างถูกต้องตรงกับเครื่องลอจิกแอนนาไลเซอร์จริง เหมือนสัญญาณที่แท้จริง จึงสามารถจะนำลอจิกแอนนาไลเซอร์ไปใช้งานได้จริง

โครงงานนี้สามารถทำงานได้ตามหลักและ ทฤษฎีที่ได้กล่าวมาในบทที่ 2 มีปัญหาและข้อจำกัดต่างๆ ดังต่อไปนี้

5.1.1 การเขียน VHDL ทำได้ไม่ดีนัก เนื่องจากไม่ตรงตามทฤษฎีที่ได้กล่าวมาข้างต้นจึงไม่สามารถ ให้การทำงานทุกส่วนอยู่บน FPGA ทั้งหมดต้องใช้ MCS-51 ช่วยการทำงานบางส่วนเพราะมีฟังก์ชั่นรองรับไว้แล้ว

5.1.2 ในช่วงที่เครื่องลอจิกแอนนาไลเซอร์ ทำการที่จะดึงข้อมูลที่ตรวจจับได้มาแสดงบนจอภาพนั้น จากซอฟแวร์ที่เขียนขึ้นมานั้น ต้องรอให้อ่านข้อมูลทั้ง 32 กิโลไบท์ทั้งหมดก่อน แล้วจึงจะแสดงผลทางจอภาพ ซึ่งใช้เวลาในการทำงานนานพอสมควรจากการใช้งานของเครื่องตามความเป็นจริงแล้ว ผู้ใช้งานไม่จำเป็นต้องการตรวจสอบข้อมูลทั้งหมด 32 กิโลไบท์ก็ได้ น่าจะมีการปรับปรุงซอฟแวร์ให้สามารถเลือกขนาดของข้อมูลที่ต้องการตรวจสอบได้ ทำให้ประหยัดเวลาในการทำงานของเครื่องได้อีกมาก

5.1.3 ซอร์ฟแวร์มีความเร็วในการแสดงสัญญาณไม่มานักทำให้ต้องส่งด้วยอัตราที่ต่ำกว่าที่เครื่องลอจิกแอนนาไลเซอร์ทำได้ และข้อจำกัดของพอร์ตใช้งานของคอมพิวเตอร์ยังมีอัตราที่ต่ำ

5.1.4 FPGA เป็น อุปกรณ์ที่มีใช้งาน ในไทยยังไม่มากเท่าที่ควรทำให้หาข้อมูล และอุปกรณ์ต่างๆ ได้ยากลำบาก และยังมีราคาสูงถ้าเทียบกับไมโครคอนโทรเลอร์

แต่อย่างไรก็ตามเครื่องลอจิกแอนนาในโครงงานนี้ก็ได้รวมฟังก์ชั่นและข้อดีของแต่ละยี่ห้อ และพัฒนาโดยใช้ FPGA ทำให้สามารถขยายผลและแก้ไขดัดแปลงต่อไปได้โดยมีฟังก์ชั่นดังนี้

โดยมีความสามารถและฟังชั่นพิเศษต่างๆดังนี้

- สามารถตั้งชื่อและสีของสัญญาณนั้นๆได้
- สามารถเลือกอัตราการสุ่มข้อมูล (Sample Rate)ได้
- สามารถจัดการแสดงเป็นกลุ่มข้อมูลได้
- สามารถ ตั้งการ Trigger โดยเลือกเป็นแบบ Edge Trigger หรือเป็น Pattern Trigger
- สามารถค้นหาตาม Pattern ที่ต้องการได้
- สามารถย่อขยายขนาดของรูปสัญญาณได้
- สามารถกำหนดช่วงเวลาโดยตั้งจุดเริ่มต้น จุดสิ้นสุดโดยจะแสดงค่าของชุดข้อมูลนั้นและระยะเวลาที่ใช้
- สามารถเลือกสิ่งแวดล้อมตามต้องการได้

5.2 สรุป

ผลที่ได้จากการทดลองใช้งานนั้น เครื่องลอจิกแอนนาไลเซอร์ชุดนี้ สามารถตรวจรับข้อมูลได้ถูกต้องตามทฤษฎีที่ได้ออกแบบไว้ดังที่กล่าวผ่านมาแล้วในบทที่ 3 ซึ่งผลจากการทดลองกล่าวสรุปได้ดังต่อไปนี้

5.2.1 สามารถตรวจจับสัญญาณได้ถึง 16 ช่องสัญญาณ แตกละช่องแยกออกจากกันโดยเด็ดขาด

5.2.2 อัตราสุ่มสัญญาณสามารถจัดตั้งได้ลดลงจากเดิม อัตราสูงสุดที่กำหนดไว้คือ 24 เมกกะเฮิทซ์ ลดลงเป็น 12 เมกกะเฮิทซ์ เพราะต้องรอการทำงานในส่วนของไมโครคอนโทรเลอร์ MCS-51 และสามารถลดทอนอัตราสุ่มสัญญาณได้ตามที่กำหนด

5.2.3 การกำหนดข้อมูลเริ่มต้นการทำงาน (Word Triggering) สามารถกำหนดได้ทั้ง 16 ช่องสัญญาณ โดยสามารถกำหนดสถานะทางตรรกะเป็นได้ทั้ง 1,0 หรือไม่กำหนด(Don't care) ก็ได้ ซึ่งทั้งนี้ขึ้นอยู่กับความต้องการของผู้ใช้งาน โดยในส่วนนี้ย้ายการทำงานมาในส่วนซอร์ฟแวร์

5.2.4 การควบคุมการทำงานและการแสดงผล กระทำผ่านทางเครื่องคอมพิวเตอร์ส่วนบุคคล โดยผ่านทางพอร์ทอนุกรม (RS-232)

ซึ่งจากผลการทดลองพบว่าผลที่ได้เป็นที่น่าพอใจพอสมควรเมื่อเทียบกับเครื่องชนิดเดียวกันของบริษัทอื่นๆที่สร้างขึ้นมา

5.3 ข้อเสนอแนะสำหรับการพัฒนาในอนาคต

ดังที่ทราบกันดีอยู่แล้วว่า ในสภาวะปัจจุบันเทคโนโลยีทางด้านไมโครคอนโทรเลอร์ได้ก้าวหน้าไปอย่างรวดเร็ว เพราะฉะนั้นอุปกรณ์ที่ใช้ในการตรวจสอบ และพัฒนาลอจิกแอนนาไลเซอร์ ชุดนี้ ผู้ที่สนใจควรจะพัฒนาอุปกรณ์ชิ้นนี้เพิ่มเติม เพื่อให้เครื่องมีประสิทธิภาพสูงขึ้นดังนี้

5.3.1 โดยอาศัยคุณสมบัติของ FPGA ซึ่งมีข้อได้เปรียบไมโครโปรเซสเซอร์เบอร์อื่นได้แก่

5.3.1.1 สามารถทำงานด้วยสัญญาณนาฬิกาสูงถึง 24 เมกกะเฮิทซ์หรือมากกว่านั้น ทำให้การจัดการข้อมูลเป็นไปได้อย่างรวดเร็ว เพื่อให้สอดคล้องกับการพัฒนาด้านไมโครคอนโทรเลอร์ในสภาวะปัจจุบัน

5.3.1.2 สามารถอ้างอิงหน่วยความจำได้สูง และมีส่วนควบคุมหน่วยความจำโดยตรงทำให้การใช้งานเป็นไปอย่างมีประสิทธิภาพ

5.3.1.3 ทำการวิเคราะห์สัญญาณแบบอนุกรมโดยสามารถกำหนดเองได้

5.3.1.4 สามารถบรรจุเกตได้สูงถึง 50,000 หรือมากกว่านั้นเพื่อการพัฒนาวงจรที่มีประสิทธิภาพมาก

จากคุณสมบัติดังกล่าวข้างต้น และจากการออกแบบไว้เมื่อต้องการเก็บข้อมูลให้ตัวคอมพิวเตอร์ส่วนบุคคลทำการเก็บข้อมูลไว้ ถ้าออกแบบเพิ่มเติมให้เครื่องลอจิกแอนนาไลเซอร์ สามารถทำงานบน FPGA ทั้งหมดก็จะทำให้เครื่องลอจิกแอนนาไลเซอร์ มีความเร็วในการทำงานสูง

5.3.2เมื่อใช้เครื่องลอจิกแอนนาไลเซอร์ทำการจับสัญญาณที่ได้จากบัสข้อมูลของไมโครคอนโทรเลอร์เบอร์อื่นๆ ควรจะมีโปรแกรมสนับสนุนช่วยเปลี่ยน การแสดงผลจากรูปสภาวะทางตรรกะให้เป็นภาษาแอสแซมบลี้ ของไมโครคอนโทรเลอร์เบอร์นั้นๆ ด้วย ในกรณีนี้ จะเห็นว่าหน่วยความจำของไมโครคอนโทรเลอร์

เบอร์นั้นๆหรือ มีเหลือเฟือที่จะบรรจุโปรแกรมเหล่านี้ลงไปได้ โดยผ่านทางจานแม่เหล็กที่เชื่อมต่ออยู่ หรือบันทึกลงไปในอีพรอมก็ได้

5.3.3 อัตราการสุ่มสัญญาณควรจะเพิ่มขึ้นให้ได้ถึง 100 เมกกะเฮิทซ์ หรือมากกว่านั้น เพื่อรองรับการนำไปใช้วิเคราะห์สัญญาณของบอร์ดใหม่ๆ ของไมโครคอนโทรเลอร์

5.3.4 ในช่วงที่เครื่องลอจิกแอนนาไลเซอร์ ทำการที่จะดึงข้อมูลที่ตรวจจับได้มาแสดงบนจอภาพนั้น จากซอฟแวร์ที่เขียนขึ้นมานั้น ต้องรอให้อ่านข้อมูลทั้ง 32 กิโลไบท์ทั้งหมดก่อน แล้วจึงจะแสดงผลทางจอภาพ ซึ่งใช้เวลาในการทำงานนานพอสมควรจากการใช้งานของเครื่องตามความเป็นจริงแล้ว ผู้ใช้งานไม่จำเป็นต้องการตรวจสอบข้อมูลทั้งหมด 32 กิโลไบท์ก็ได้ น่าจะมีการปรับปรุงซอฟแวร์ให้สามารถเลือกขนาดของข้อมูลที่ต้องการตรวจสอบได้ ทำให้ประหยัดเวลาในการทำงานของเครื่องได้อีกมาก

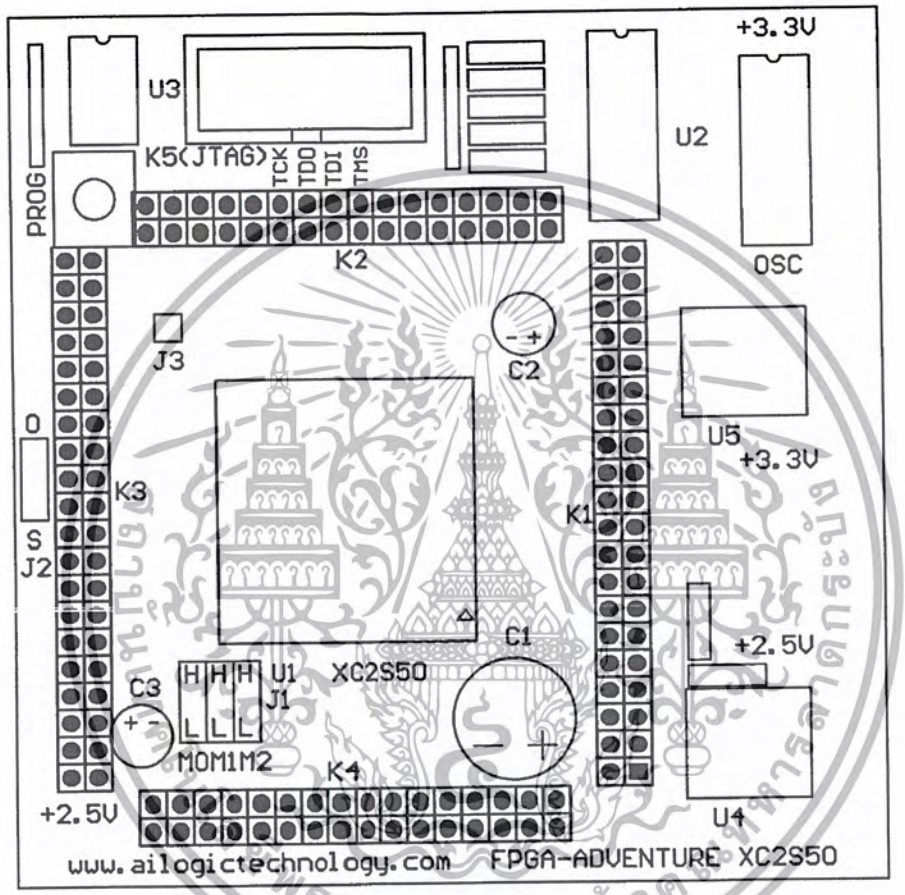
5.3.5 เพิ่มเติมในส่วนของซอฟท์แวร์ให้มีฟังก์ชั่นการทำงานเพื่ออำนวยความสะดวกในการใช้งาน

จากที่กล่าวมาแล้วข้างต้น จะเห็นว่าการเลือกใช้ FPGA นั้น เป็นการเปิดให้มีการพัฒนาต่อไปในอนาคตได้ด้วย รวมทั้งชุดคำสั่งต่างๆ ง่ายและมีโปรแกรมช่วยเหลือมากขึ้น ผู้พัฒนาสามารถหาโปรแกรมช่วยเหลือที่มีประสิทธิภาพในการพัฒนาวงจรต่อไป

ภาคผนวก ก.วงจรและอุปกรณ์ที่เกี่ยวข้อง

FPGA Adventure Discovery XC2S50 Board Manual

รายละเอียดบอร์ดพัฒนา FPGA Adventure Discovery XC2S50 มีรายละเอียดแสดงดังรูปที่ 1

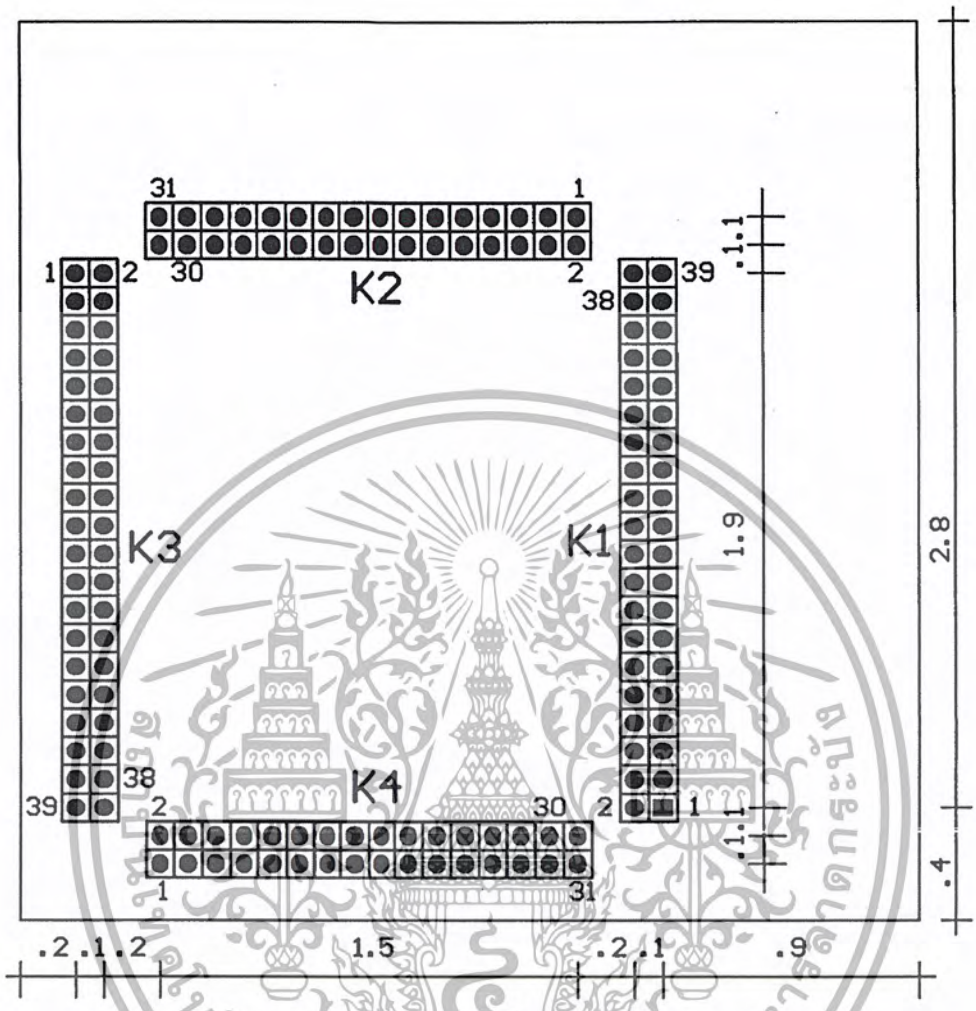


รูปที่ 1 บอร์ดพัฒนา FPGA Adventure Discovery XC2S50

1. **Connector and Jumper**

1.1 Expansion Connector (K1 – K4)

เป็นหัวต่อแบบ Dual Row Pin Header โดย K1 และ K3 เป็นแบบ 2x20 Pin ส่วน K2 และ K4 เป็นแบบ 2x16 Pin ที่ใช้เชื่อมสัญญาณอินพุทเอ้าท์พุท (I/O) ของ FPGA ไปยังบอร์ดอื่นๆหรืออุปกรณ์ภายนอก มีรายละเอียดตามตารางที่ 1 การติดตั้งบอร์ดพัฒนาเข้ากับบอร์ดอื่นทำได้โดยการบัดกรีโดยตรงหรือเสียบเข้ากับ Female Header โดยที่ตำแหน่งรูของ PCB แสดงดังรูปที่ 2 อุปกรณ์ (Component) ที่อยู่บนบอร์ดอื่นที่มีความสูงควรติดตั้งให้เลยออกไปจากขอบของบอร์ดพัฒนา



รูปที่ 2 ตำแหน่งรูของ PCB (Top Layer)

1.2 JTAG Connector (K5)

เป็น Connector 10 Pin ใช้ต่อกับ JTAG Cable ในการโปรแกรมข้อมูลจากคอมพิวเตอร์ผ่านทาง Parallel Port ลงสู่ชิพ FPGA

1.3 J1(M0 ,M1,M2)

จัมเปอร์ที่ใช้เป็น Configuration Mode ของ FPGA รายละเอียดแสดงตามตารางที่ 2 โดยปกติหากเป็นการ Download โดยใช้ JTAG Cable จะสามารถทำได้ทันทีโดยไม่ต้องสนใจจัมเปอร์ หรือ เซทตำแหน่งจัมเปอร์ตามตารางที่ 2 ก็ได้ ตัวอย่างเช่น ถ้าเซทจัมเปอร์ M0 , M1 และ M2 ไว้ด้านล่าง (L= Logic 0)ทั้งหมดดังรูป  แสดงว่าเป็น Master Serial Mode เป็นต้น

1.4 J2

เป็นจัมเปอร์ที่ใช้เลือกสัญญาณนาฬิกาจากออสซิลเลเตอร์บนบอร์ดพัฒนาหรือจากภายนอกที่ต่อมาบอร์ดอื่นเข้ากับขา GCK0 ของ FPGA หากต้องการเลือกสัญญาณนาฬิกาจากออสซิลเลเตอร์บนบอร์ดพัฒนาให้เซทจัมเปอร์ไปที่ตำแหน่ง S (Short) หากใช้สัญญาณนาฬิกาจากภายนอกให้เซทจัมเปอร์ไปที่ตำแหน่ง O (Open)

1.5 J3 และ Push Button

ใช้เมื่อต้องการโปรแกรม FPGA ใหม่โดยไม่ต้องตัดไฟเลี้ยงออก เมื่อ Short J3 ลงกราวด์แล้วปล่อยให้ Open จะให้ผลเหมือนกับการกด Push Button 1 ครั้ง เพื่อรีเซตให้ FPGA พร้อมที่จะทำการโปรแกรมใหม่อีกครั้ง J3 จะใช้ในกรณีที่โปรแกรมลง FPGA ใน Slave Serial Mode

1.6 Power Supply

บอร์ดพัฒนาในตระกูล Adventure นี้บอร์ดถูกออกแบบในลักษณะ Daughter Board Style ดังนั้นบอร์ดอื่นหรือเมนบอร์ดที่ใช้งานต้องมีไฟเลี้ยง +5 โวลต์(+/- 5%) ไปจ่ายให้กับบอร์ด Adventure ด้วย โดยจ่ายผ่านทางขาที่ 3 และ 4 หรือ 37 และ 38 ของ Connector K1 ซึ่งการติดตั้งอาจใช้วิธีการบัดกรีโดยตรงหรือเสียบบน Female Header ของบอร์ดอื่นก็ได้ ในขณะเดียวกันที่ขาของ K3 บอร์ด Adventure จะจ่ายไฟเลี้ยง +2.5 โวลต์ออกมาที่ขา 37 และ 38 และจ่ายไฟ +3.3 โวลต์ ออกมาที่ขา 39 และ 40 เพื่อให้บอร์ดอื่นสามารถใช้ไฟเลี้ยงดังกล่าวได้โดยไม่ต้องหาแหล่งจ่ายจากภายนอกเพิ่มเติม แต่และจุดจ่ายกระแสได้สูงสุดไม่เกิน 25 mA (หรือรวมกันได้ 50 mA) ซึ่งเพียงพอต่อการใช้งานไอซีเล็กๆ ประมาณ 2-3 ตัวเช่น บัฟเฟอร์ เกท ฟลิปฟลอบ เป็นต้น

ในการใช้งานบอร์ด ส่วน Core ของ FPGA ไม่ควรกินกระแสไฟเกิน 500 mA และส่วน I/O ไม่ควรกินกระแสไฟเกิน 200 mA และรวมกันไม่ควรกินกระแสไฟเกิน 600 mA

2. Input

2.1 Oscillator (OSC)

เป็นตัวกำเนิดสัญญาณนาฬิกาที่ให้ตั้งบน IC Socket เพื่อให้ถอดสามารถเปลี่ยนได้ ทำให้สามารถใช้กับออสซิลเลเตอร์ความถี่อื่นๆ ได้ โดยที่ออสซิลเลเตอร์จะต่ออยู่กับขา Global clock (GCK0) ของ FPGA ซึ่งเป็นขาที่เหมาะสำหรับวงจรที่ต้องการทำงานที่ความถี่ในสูงๆ ในกรณีที่ใช้วงจร DLL ของ FPGA สัญญาณนาฬิกาที่เข้าต้องไม่น้อยกว่า 25 MHz (ในทางปฏิบัติ 24 MHz ก็ยังสามารถทำงานได้)

2.2 Serial PROM Socket (U3)

เป็นช่องสำหรับใส่ IC Serial PROM เบอร์ XC17S50A ที่ได้รับการโปรแกรมวงจรเรียบร้อยแล้ว และต้องเซทจัมเปอร์ J1 เพื่อให้ FPGA อยู่ใน Master Serial Mode (M0 M1 และ M2 เป็น Logic = 0) ในโหมดนี้ FPGA จะดาวน์โหลดข้อมูลจาก Serial PROM ลงสู่ชิพ FPGA อย่างอัตโนมัติทุกครั้งที่มีการจ่ายไฟเลี้ยง

ตารางที่ 1 ตำแหน่ง Pin ที่ Connector (K1 – K4) และ FPGA

FPGA Adventure Discovery

I/O	Pin
OSC	P88

Connector	Pin																			
	1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39
	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
K1	GND	5V	GND	P3	P4	P6	P7	P11	P12	P15	P18	P20	P21	P23	P26	P28	P29	P31	5V	3.3V
	GND	5V	GND	GND	P5	GND	P10	GND	P13	GND	P19	GND	P22	GND	P27	GND	P30	GND	5V	3.3V
K2	P38	P40	P42	P43	P46	P47	P49	P50	P54	P56	P58	P59	P62	P63	P65	P66				
	GND	P41	GND	P44	GND	P48	GND	P51	GND	P57	GND	P60	GND	P64	GND	P67				
K3	GND	GND	P74	P76	P77	P79	P80	P84	P85	P87	P88	P93	P94	P96	P99	P101	P102	GND	2.5V	GND
	GND	GND	P75	GND	P78	GND	P83	GND	P86	GND	P91	GND	P95	GND	P100	GND	P103	GND	2.5V	GND
K4	P112	P114	P115	P117	P118	P121	P122	P124	P126	P130	P131	P133	P134	P137	P138	P140				
	P113	GND	P116	GND	P120	GND	P123	GND	P129	GND	P132	GND	P136	GND	P139	P141				

หมายเหตุ

- Pin 21 ของ Connector K3 ต่ออยู่กับ FPGA Pin 88 เป็น GCK0 และต่ออยู่กับออสซิลเลเตอร์บนบอร์ด
- Pin 22 ของ Connector K3 ต่ออยู่กับ FPGA Pin 91 เป็น GCK1
- Pin 21 ของ Connector K1 ต่ออยู่กับ FPGA Pin 18 เป็น GCK2
- Pin 19 ของ Connector K1 ต่ออยู่กับ FPGA Pin 15 เป็น GCK3
- ***Pin GCK0 – GCK3 เป็น Input ได้อย่างเดียว "ไม่" สามารถทำเป็น Output ได้***
- I/O ทุกขาของ FPGA สามารถรองรับสัญญาณที่ 5 โวล์ตและ 3.3 โวล์ตได้

ตารางที่ 2 การเซทจัมเปอร์ J1 เพื่อให้FPGA ทำงานในโหมดที่ต้องการ

Configuration Mode	M0	M1	M2	CCLK Direction	Data width	Serial Dout
Master Serial mode	0	0	0	Out	1	Yes
Slave Parallel mode	0	1	1	In	8	No
Boundary-Scan mode	1	0	1	N/A	1	No
Slave Serial mode	1	1	1	In	1	Yes

Pin Definitions

Pin Name	Dedicated Pin	Direction	Description
GCK0, GCK1, GCK2, GCK3	No	Input	Clock input pins that connect to Global Clock Buffers. These pins become user inputs when not needed for clocks.
M0, M1, M2	Yes	Input	Mode pins are used to specify the configuration mode.
CCLK	Yes	Input or Output	The configuration Clock I/O pin. It is an input for slave-parallel and slave-serial modes, and output in master-serial mode.
$\overline{\text{PROGRAM}}$	Yes	Input	Initiates a configuration sequence when asserted Low.
DONE	Yes	Bidirectional	Indicates that configuration loading is complete, and that the start-up sequence is in progress. The output may be open drain.
$\overline{\text{INIT}}$	No	Bidirectional (Open-drain)	When Low, indicates that the configuration memory is being cleared. This pin becomes a user I/O after configuration.
BUSY/DOUT	No	Output	In Slave Parallel mode, BUSY controls the rate at which configuration data is loaded. This pin becomes a user I/O after configuration unless the Slave Parallel port is retained. In serial modes, DOUT provides configuration data to downstream devices in a daisy-chain. This pin becomes a user I/O after configuration.
D0/DIN, D1, D2, D3, D4, D5, D6, D7	No	Input or Output	In Slave Parallel mode, D0-D7 are configuration data input pins. During readback, D0-D7 are output pins. These pins become user I/Os after configuration unless the Slave Parallel port is retained. In serial modes, DIN is the single data input. This pin becomes a user I/O after configuration.
$\overline{\text{WRITE}}$	No	Input	In Slave Parallel mode, the active-low Write Enable signal. This pin becomes a user I/O after configuration unless the Slave Parallel port is retained.
$\overline{\text{CS}}$	No	Input	In Slave Parallel mode, the active-low Chip Select signal. This pin becomes a user I/O after configuration unless the Slave Parallel port is retained.
TDI, TDO, TMS, TCK	Yes	Mixed	Boundary Scan Test Access Port pins (IEEE 1149.1).
V_{CCINT}	Yes	Input	Power supply pins for the internal core logic.
V_{CCO}	Yes	Input	Power supply pins for output drivers (subject to banking rules)
V_{REF}	No	Input	Input threshold voltage pins. Become user I/Os when an external threshold voltage is not needed (subject to banking rules).
GND	Yes	Input	Ground.
IRDY, TRDY	No	See PCI core documentation	These signals can only be accessed when using Xilinx PCI cores. If the cores are not used, these pins are available as user I/Os.

Pinout Tables

The following device-specific pinout tables include all packages available for each Spartan-II device. They follow the pad locations around the die, and include Boundary Scan register locations.

XC2S15 Device Pinouts

XC2S15 Pad Name					
Function	Bank	VQ100	TQ144	CS144	Bndry Scan
GND	-	P1	P143	A1	-
TMS	-	P2	P142	B1	-
I/O	7	P3	P141	C2	77
I/O	7	-	P140	C1	80
I/O, V_{REF}	7	P4	P139	D4	83
I/O	7	P5	P137	D2	86
I/O	7	P6	P136	D1	89
GND	-	-	P135	E4	-
I/O	7	P7	P134	E3	92
I/O	7	-	P133	E2	95
I/O, V_{REF}	7	P8	P132	E1	98
I/O	7	P9	P131	F4	101
I/O	7	-	P130	F3	104
I/O, IRDY(1)	7	P10	P129	F2	107
GND	-	P11	P128	F1	-
V_{CCO}	7	P12	P127	G2	-
V_{CCO}	6	P12	P127	G2	-
I/O, TRDY(1)	6	P13	P126	G1	110
V_{CCINT}	-	P14	P125	G3	-
I/O	6	-	P124	G4	113
I/O	6	P15	P123	H1	116
I/O, V_{REF}	6	P16	P122	H2	119
I/O	6	-	P121	H3	122
I/O	6	P17	P120	H4	125
GND	-	-	P119	J1	-
I/O	6	P18	P118	J2	128
I/O	6	P19	P117	J3	131
I/O, V_{REF}	6	P20	P115	K1	134
I/O	6	-	P114	K2	137
I/O	6	P21	P113	K3	140
I/O	6	P22	P112	L1	143
M1	-	P23	P111	L2	146
GND	-	P24	P110	L3	-
M0	-	P25	P109	M1	147
V_{CCO}	6	P26	P108	M2	-
V_{CCO}	5	P26	P107	N1	-
M2	-	P27	P106	N2	148
I/O	5	-	P103	K4	155

XC2S15 Device Pinouts *(Continued)*

XC2S15 Pad Name					
Function	Bank	VQ100	TQ144	CS144	Bndry Scan
I/O, V_{REF}	5	P30	P102	L4	158
I/O	5	P31	P100	N4	161
I/O	5	P32	P99	K5	164
GND	-	-	P98	L5	-
V_{CCINT}	-	P33	P97	M5	-
I/O	5	-	P96	N5	167
I/O	5	-	P95	K6	170
I/O, V_{REF}	5	P34	P94	L6	173
I/O	5	-	P93	M6	176
V_{CCINT}	-	P35	P92	N6	-
I, GCK1	5	P36	P91	M7	185
V_{CCO}	5	P37	P90	N7	-
V_{CCO}	4	P37	P90	N7	-
GND	-	P38	P89	L7	-
I, GCK0	4	P39	P88	K7	186
I/O	4	P40	P87	N8	190
I/O	4	-	P86	M8	193
I/O, V_{REF}	4	P41	P85	L8	196
I/O	4	-	P84	K8	199
I/O	4	-	P83	N9	202
V_{CCINT}	-	P42	P82	M9	-
GND	-	-	P81	L9	-
I/O	4	P43	P80	K9	205
I/O	4	P44	P79	N10	208
I/O, V_{REF}	4	P45	P77	L10	211
I/O	4	-	P76	N11	214
I/O	4	P46	P75	M11	217
I/O	4	P47	P74	L11	220
GND	-	P48	P73	N12	-
DONE	3	P49	P72	M12	223
V_{CCO}	4	P50	P71	N13	-
V_{CCO}	3	P50	P70	M13	-
$\overline{PROGRAM}$	-	P51	P69	L12	226
I/O ($\overline{INIT}$)	3	P52	P68	L13	227
I/O (D7)	3	P53	P67	K10	230
I/O	3	-	P66	K11	233
I/O, V_{REF}	3	P54	P65	K12	236
I/O	3	P55	P63	J10	239

XC2S15 Device Pinouts *(Continued)*

XC2S15 Pad Name					
Function	Bank	VQ100	TQ144	CS144	Bndry Scan
I/O (D6)	3	P56	P62	J11	242
GND	-	-	P61	J12	-
I/O (D5)	3	P57	P60	J13	245
I/O	3	P58	P59	H10	248
I/O, V_{REF}	3	P59	P58	H11	251
I/O (D4)	3	P60	P57	H12	254
I/O	3	-	P56	H13	257
V_{CCINT}	-	P61	P55	G12	-
I/O, TRDY(1)	3	P62	P54	G13	260
V_{CCO}	3	P63	P53	G11	-
V_{CCO}	2	P63	P53	G11	-
GND	-	P64	P52	G10	-
I/O, IRDY(1)	2	P65	P51	F13	263
I/O	2	-	P50	F12	266
I/O (D3)	2	P66	P49	F11	269
I/O, V_{REF}	2	P67	P48	F10	272
I/O	2	P68	P47	E13	275
I/O (D2)	2	P69	P46	E12	278
GND	-	-	P45	E11	-
I/O (D1)	2	P70	P44	E10	281
I/O	2	P71	P43	D13	284
I/O, V_{REF}	2	P72	P41	D11	287
I/O	2	-	P40	C13	290
I/O (DIN, D0)	2	P73	P39	C12	293
I/O (DOUT, BUSY)	2	P74	P38	C11	296
CCLK	2	P75	P37	B13	299
V_{CCO}	2	P76	P36	B12	-
V_{CCO}	1	P76	P35	A13	-
TDO	2	P77	P34	A12	-
GND	-	P78	P33	B11	-
TDI	-	P79	P32	A11	-
I/O ($\overline{CS}$)	1	P80	P31	D10	0
I/O ($\overline{WRITE}$)	1	P81	P30	C10	3
I/O	1	-	P29	B10	6
I/O, V_{REF}	1	P82	P28	A10	9
I/O	1	P83	P27	D9	12
I/O	1	P84	P26	C9	15
GND	-	-	P25	B9	-
V_{CCINT}	-	P85	P24	A9	-
I/O	1	-	P23	D8	18

XC2S15 Device Pinouts *(Continued)*

XC2S15 Pad Name					
Function	Bank	VQ100	TQ144	CS144	Bndry Scan
I/O	1	-	P22	C8	21
I/O, V_{REF}	1	P86	P21	B8	24
I/O	1	-	P20	A8	27
I/O	1	P87	P19	B7	30
I, GCK2	1	P88	P18	A7	36
GND	-	P89	P17	C7	-
V_{CCO}	1	P90	P16	D7	-
V_{CCO}	0	P90	P16	D7	-
I, GCK3	0	P91	P15	A6	37
V_{CCINT}	-	P92	P14	B6	-
I/O	0	-	P13	C6	44
I/O, V_{REF}	0	P93	P12	D6	47
I/O	0	-	P11	A5	50
I/O	0	-	P10	B5	53
V_{CCINT}	-	P94	P9	C5	-
GND	-	-	P8	D5	-
I/O	0	P95	P7	A4	56
I/O	0	P96	P6	B4	59
I/O, V_{REF}	0	P97	P5	C4	62
I/O	0	-	P4	A3	65
I/O	0	P98	P3	B3	68
TCK	-	P99	P2	C3	-
V_{CCO}	0	P100	P1	A2	-
V_{CCO}	7	P100	P144	B2	-

04/18/01

Notes:
1. IRDY and TRDY can only be accessed when using Xilinx PCI cores.

Additional XC2S15 Package Pins

VQ100

Not Connected Pins					
P28	P29	-	-	-	-

11/02/00

TQ144

Not Connected Pins					
P42	P64	P78	P101	P104	P105
P116	P138	-	-	-	-

11/02/00

CS144

Not Connected Pins					
D3	D12	J4	K13	M3	M4
M10	N3	-	-	-	-

11/02/00

XC2S30 Device Pinouts

XC2S30 Pad Name						
Function	Bank	VQ100	TQ144	CS144	PQ208	Bndry Scan
GND	-	P1	P143	A1	P1	-
TMS	-	P2	P142	B1	P2	-
I/O	7	P3	P141	C2	P3	113
I/O	7	-	P140	C1	P4	116
I/O	7	-	-	-	P5	119
I/O, V_{REF}	7	P4	P139	D4	P6	122
I/O	7	-	P138	D3	P8	125
I/O	7	P5	P137	D2	P9	128
I/O	7	P6	P136	D1	P10	131
GND	-	-	P135	E4	P11	-
V_{CCO}	7	-	-	-	P12	-
I/O	7	P7	P134	E3	P14	134
I/O	7	-	P133	E2	P15	137
I/O	7	-	-	-	P16	140
I/O	7	-	-	-	P17	143
I/O	7	-	-	-	P18	146
GND	-	-	-	-	P19	-
I/O, V_{REF}	7	P8	P132	E1	P20	149
I/O	7	P9	P131	F4	P21	152
I/O	7	-	P130	F3	P22	155
I/O	7	-	-	-	P23	158
I/O, IRDY(1)	7	P10	P129	F2	P24	161
GND	-	P11	P128	F1	P25	-
V_{CCO}	7	P12	P127	G2	P26	-
V_{CCO}	6	P12	P127	G2	P26	-
I/O, TRDY(1)	6	P13	P126	G1	P27	164
V_{CCINT}	-	P14	P125	G3	P28	-
I/O	6	-	P124	G4	P29	170
I/O	6	P15	P123	H1	P30	173
I/O, V_{REF}	6	P16	P122	H2	P31	176
GND	-	-	-	-	P32	-
I/O	6	-	-	-	P33	179
I/O	6	-	-	-	P34	182
I/O	6	-	-	-	P35	185
I/O	6	-	P121	H3	P36	188
I/O	6	P17	P120	H4	P37	191
V_{CCO}	6	-	-	-	P39	-
GND	-	-	P119	J1	P40	-
I/O	6	P18	P118	J2	P41	194
I/O	6	P19	P117	J3	P42	197
I/O	6	-	P116	J4	P43	200

XC2S30 Device Pinouts *(Continued)*

XC2S30 Pad Name						
Function	Bank	VQ100	TQ144	CS144	PQ208	Bndry Scan
I/O, V_{REF}	6	P20	P115	K1	P45	203
I/O	6	-	-	-	P46	206
I/O	6	-	P114	K2	P47	209
I/O	6	P21	P113	K3	P48	212
I/O	6	P22	P112	L1	P49	215
M1	-	P23	P111	L2	P50	218
GND	-	P24	P110	L3	P51	-
M0	-	P25	P109	M1	P52	219
V_{CCO}	6	P26	P108	M2	P53	-
V_{CCO}	5	P26	P107	N1	P53	-
M2	-	P27	P106	N2	P54	220
I/O	5	-	P103	K4	P57	227
I/O	5	-	-	-	P58	230
I/O, V_{REF}	5	P30	P102	L4	P59	233
I/O	5	-	P101	M4	P61	236
I/O	5	P31	P100	N4	P62	239
I/O	5	P32	P99	K5	P63	242
GND	-	-	P98	L5	P64	-
V_{CCO}	5	-	-	-	P65	-
V_{CCINT}	-	P33	P97	M5	P66	-
I/O	5	-	P96	N5	P67	245
I/O	5	-	P95	K6	P68	248
I/O	5	-	-	-	P69	251
I/O	5	-	-	-	P70	254
I/O	5	-	-	-	P71	257
GND	-	-	-	-	P72	-
I/O, V_{REF}	5	P34	P94	L6	P73	260
I/O	5	-	-	-	P74	263
I/O	5	-	P93	M6	P75	266
V_{CCINT}	-	P35	P92	N6	P76	-
I, GCK1	5	P36	P91	M7	P77	275
V_{CCO}	5	P37	P90	N7	P78	-
V_{CCO}	4	P37	P90	N7	P78	-
GND	-	P38	P89	L7	P79	-
I, GCK0	4	P39	P88	K7	P80	276
I/O	4	P40	P87	N8	P81	280
I/O	4	-	P86	M8	P82	283
I/O	4	-	-	-	P83	286
I/O, V_{REF}	4	P41	P85	L8	P84	289
GND	-	-	-	-	P85	-
I/O	4	-	-	-	P86	292

XC2S30 Device Pinouts *(Continued)*

XC2S30 Pad Name						
Function	Bank	VQ100	TQ144	CS144	PQ208	Bndry Scan
I/O	4	-	-	-	P87	295
I/O	4	-	-	-	P88	298
I/O	4	-	P84	K8	P89	301
I/O	4	-	P83	N9	P90	304
V_{CCINT}	-	P42	P82	M9	P91	-
V_{CCO}	4	-	-	-	P92	-
GND	-	-	P81	L9	P93	-
I/O	4	P43	P80	K9	P94	307
I/O	4	P44	P79	N10	P95	310
I/O	4	-	P78	M10	P96	313
I/O, V_{REF}	4	P45	P77	L10	P98	316
I/O	4	-	-	-	P99	319
I/O	4	-	P76	N11	P100	322
I/O	4	P46	P75	M11	P101	325
I/O	4	P47	P74	L11	P102	328
GND	-	P48	P73	N12	P103	-
DONE	3	P49	P72	M12	P104	331
V_{CCO}	4	P50	P71	N13	P105	-
V_{CCO}	3	P50	P70	M13	P105	-
$\overline{PROGRAM}$	-	P51	P69	L12	P106	334
I/O ($\overline{INIT}$)	3	P52	P68	L13	P107	335
I/O (D7)	3	P53	P67	K10	P108	338
I/O	3	-	P66	K11	P109	341
I/O	3	-	-	-	P110	344
I/O, V_{REF}	3	P54	P65	K12	P111	347
I/O	3	-	P64	K13	P113	350
I/O	3	P55	P63	J10	P114	353
I/O (D6)	3	P56	P62	J11	P115	356
GND	-	-	P61	J12	P116	-
V_{CCO}	3	-	-	-	P117	-
I/O (D5)	3	P57	P60	J13	P119	359
I/O	3	P58	P59	H10	P120	362
I/O	3	-	-	-	P121	365
I/O	3	-	-	-	P122	368
I/O	3	-	-	-	P123	371
GND	-	-	-	-	P124	-
I/O, V_{REF}	3	P59	P58	H11	P125	374
I/O (D4)	3	P60	P57	H12	P126	377
I/O	3	-	P56	H13	P127	380
V_{CCINT}	-	P61	P55	G12	P128	-
I/O, TRDY(1)	3	P62	P54	G13	P129	386

XC2S30 Device Pinouts *(Continued)*

XC2S30 Pad Name						
Function	Bank	VQ100	TQ144	CS144	PQ208	Bndry Scan
V_{CCO}	3	P63	P53	G11	P130	-
V_{CCO}	2	P63	P53	G11	P130	-
GND	-	P64	P52	G10	P131	-
I/O, IRDY(1)	2	P65	P51	F13	P132	389
I/O	2	-	-	-	P133	392
I/O	2	-	P50	F12	P134	395
I/O (D3)	2	P66	P49	F11	P135	398
I/O, V_{REF}	2	P67	P48	F10	P136	401
GND	-	-	-	-	P137	-
I/O	2	-	-	-	P138	404
I/O	2	-	-	-	P139	407
I/O	2	-	-	-	P140	410
I/O	2	P68	P47	E13	P141	413
I/O (D2)	2	P69	P46	E12	P142	416
V_{CCO}	2	-	-	-	P144	-
GND	-	-	P45	E11	P145	-
I/O (D1)	2	P70	P44	E10	P146	419
I/O	2	P71	P43	D13	P147	422
I/O	2	-	P42	D12	P148	425
I/O, V_{REF}	2	P72	P41	D11	P150	428
I/O	2	-	-	-	P151	431
I/O	2	-	P40	C13	P152	434
I/O (DIN, D0)	2	P73	P39	C12	P153	437
I/O (DOUT, BUSY)	2	P74	P38	C11	P154	440
CCLK	2	P75	P37	B13	P155	443
V_{CCO}	2	P76	P36	B12	P156	-
V_{CCO}	1	P76	P35	A13	P156	-
TDO	2	P77	P34	A12	P157	-
GND	-	P78	P33	B11	P158	-
TDI	-	P79	P32	A11	P159	-
I/O ($\overline{CS}$)	1	P80	P31	D10	P160	0
I/O ($\overline{WRITE}$)	1	P81	P30	C10	P161	3
I/O	1	-	P29	B10	P162	6
I/O	1	-	-	-	P163	9
I/O, V_{REF}	1	P82	P28	A10	P164	12
I/O	1	-	-	-	P166	15
I/O	1	P83	P27	D9	P167	18
I/O	1	P84	P26	C9	P168	21
GND	-	-	P25	B9	P169	-
V_{CCO}	1	-	-	-	P170	-

XC2S30 Device Pinouts *(Continued)*

XC2S30 Pad Name						
Function	Bank	VQ100	TQ144	CS144	PQ208	Bndry Scan
V_{CCINT}	-	P85	P24	A9	P171	-
I/O	1	-	P23	D8	P172	24
I/O	1	-	P22	C8	P173	27
I/O	1	-	-	-	P174	30
I/O	1	-	-	-	P175	33
I/O	1	-	-	-	P176	36
GND	-	-	-	-	P177	-
I/O, V_{REF}	1	P86	P21	B8	P178	39
I/O	1	-	-	-	P179	42
I/O	1	-	P20	A8	P180	45
I/O	1	P87	P19	B7	P181	48
I, GCK2	1	P88	P18	A7	P182	54
GND	-	P89	P17	C7	P183	-
V_{CCO}	1	P90	P16	D7	P184	-
V_{CCO}	0	P90	P16	D7	P184	-
I, GCK3	0	P91	P15	A6	P185	55
V_{CCINT}	-	P92	P14	B6	P186	-
I/O	0	-	P13	C6	P187	62
I/O	0	-	-	-	P188	65
I/O, V_{REF}	0	P93	P12	D6	P189	68
GND	-	-	-	-	P190	-
I/O	0	-	-	-	P191	71
I/O	0	-	-	-	P192	74
I/O	0	-	-	-	P193	77
I/O	0	-	P11	A5	P194	80
I/O	0	-	P10	B5	P195	83
V_{CCINT}	-	P94	P9	C5	P196	-
V_{CCO}	0	-	-	-	P197	-
GND	-	-	P8	D5	P198	-
I/O	0	P95	P7	A4	P199	86
I/O	0	P96	P6	B4	P200	89

XC2S30 Device Pinouts *(Continued)*

XC2S30 Pad Name						
Function	Bank	VQ100	TQ144	CS144	PQ208	Bndry Scan
I/O	0	-	-	-	P201	92
I/O, V_{REF}	0	P97	P5	C4	P203	95
I/O	0	-	-	-	P204	98
I/O	0	-	P4	A3	P205	101
I/O	0	P98	P3	B3	P206	104
TCK	-	P99	P2	C3	P207	-
V_{CCO}	0	P100	P1	A2	P208	-
V_{CCO}	7	P100	P144	B2	P208	-

04/18/01

Notes:
1. IRDY and TRDY can only be accessed when using Xilinx PCI cores.

Additional XC2S30 Package Pins

VQ100

Not Connected Pins					
P28	P29	-	-	-	-

11/02/00

TQ144

Not Connected Pins					
P104	P105	-	-	-	-

11/02/00

CS144

Not Connected Pins					
M3	N3	-	-	-	-

11/02/00

PQ208

Not Connected Pins					
P7	P13	P38	P44	P55	P56
P60	P97	P112	P118	P143	P149
P165	P202	-	-	-	-

11/02/00

Notes:
1. For the PQ208 package, P13, P38, P118, and P143, which are Not Connected Pins on the XC2S30, are assigned to V_{CCINT} on larger devices.

XC2S50 Device Pinouts

XC2S50 Pad Name					
Function	Bank	TQ144	PQ208	FG256	Bndry Scan
GND	-	P143	P1	GND*	-
TMS	-	P142	P2	D3	-
I/O	7	P141	P3	C2	149

XC2S50 Device Pinouts *(Continued)*

XC2S50 Pad Name					
Function	Bank	TQ144	PQ208	FG256	Bndry Scan
I/O	7	-	-	A2	152
I/O	7	P140	P4	B1	155
I/O	7	-	-	E3	158

XC2S50 Device Pinouts *(Continued)*

XC2S50 Pad Name					
Function	Bank	TQ144	PQ208	FG256	Bndry Scan
I/O	7	-	P5	D2	161
GND	-	-	-	GND*	-
I/O, V_{REF}	7	P139	P6	C1	164
I/O	7	-	P7	F3	167
I/O	7	-	-	E2	170
I/O	7	P138	P8	E4	173
I/O	7	P137	P9	D1	176
I/O	7	P136	P10	E1	179
GND	-	P135	P11	GND*	-
V_{CCO}	7	-	P12	V_{CCO} Bank 7*	-
V_{CCINT}	-	-	P13	V_{CCINT}*	-
I/O	7	P134	P14	F2	182
I/O	7	P133	P15	G3	185
I/O	7	-	-	F1	188
I/O	7	-	P16	F4	191
I/O	7	-	P17	F5	194
I/O	7	-	P18	G2	197
GND	-	-	P19	GND*	-
I/O, V_{REF}	7	P132	P20	H3	200
I/O	7	P131	P21	G4	203
I/O	7	-	-	H2	206
I/O	7	P130	P22	G5	209
I/O	7	-	P23	H4	212
I/O, IRDY(1)	7	P129	P24	G1	215
GND	-	P128	P25	GND*	-
V_{CCO}	7	P127	P26	V_{CCO} Bank 7*	-
V_{CCO}	6	P127	P26	V_{CCO} Bank 6*	-
I/O, TRDY(1)	6	P126	P27	J2	218
V_{CCINT}	-	P125	P28	V_{CCINT}*	-
I/O	6	P124	P29	H1	224
I/O	6	-	-	J4	227
I/O	6	P123	P30	J1	230
I/O, V_{REF}	6	P122	P31	J3	233
GND	-	-	P32	GND*	-
I/O	6	-	P33	K5	236
I/O	6	-	P34	K2	239
I/O	6	-	P35	K1	242
I/O	6	-	-	K3	245
I/O	6	P121	P36	L1	248

XC2S50 Device Pinouts *(Continued)*

XC2S50 Pad Name					
Function	Bank	TQ144	PQ208	FG256	Bndry Scan
I/O	6	P120	P37	L2	251
V_{CCINT}	-	-	P38	V_{CCINT}*	-
V_{CCO}	6	-	P39	V_{CCO} Bank 6*	-
GND	-	P119	P40	GND*	-
I/O	6	P118	P41	K4	254
I/O	6	P117	P42	M1	257
I/O	6	P116	P43	L4	260
I/O	6	-	-	M2	263
I/O	6	-	P44	L3	266
I/O, V_{REF}	6	P115	P45	N1	269
GND	-	-	-	GND*	-
I/O	6	-	P46	P1	272
I/O	6	-	-	L5	275
I/O	6	P114	P47	N2	278
I/O	6	-	-	M4	281
I/O	6	P113	P48	R1	284
I/O	6	P112	P49	M3	287
M1	-	P111	P50	P2	290
GND	-	P110	P51	GND*	-
M0	-	P109	P52	N3	291
V_{CCO}	6	P108	P53	V_{CCO} Bank 6*	-
V_{CCO}	5	P107	P53	V_{CCO} Bank 5*	-
M2	-	P106	P54	R3	292
I/O	5	-	-	N5	299
I/O	5	P103	P57	T2	302
I/O	5	-	-	P5	305
I/O	5	-	P58	T3	308
GND	-	-	-	GND*	-
I/O, V_{REF}	5	P102	P59	T4	311
I/O	5	-	P60	M6	314
I/O	5	-	-	T5	317
I/O	5	P101	P61	N6	320
I/O	5	P100	P62	R5	323
I/O	5	P99	P63	P6	326
GND	-	P98	P64	GND*	-
V_{CCO}	5	-	P65	V_{CCO} Bank 5*	-
V_{CCINT}	-	P97	P66	V_{CCINT}*	-
I/O	5	P96	P67	R6	329

XC2S50 Device Pinouts *(Continued)*

XC2S50 Pad Name					
Function	Bank	TQ144	PQ208	FG256	Bndry Scan
I/O	5	P95	P68	M7	332
I/O	5	-	P69	N7	338
I/O	5	-	P70	T6	341
I/O	5	-	P71	P7	344
GND	-	-	P72	GND*	-
I/O, V_{REF}	5	P94	P73	P8	347
I/O	5	-	P74	R7	350
I/O	5	-	-	T7	353
I/O	5	P93	P75	T8	356
V_{CCINT}	-	P92	P76	V_{CCINT}*	-
I, GCK1	5	P91	P77	R8	365
V_{CCO}	5	P90	P78	V_{CCO} Bank 5*	-
V_{CCO}	4	P90	P78	V_{CCO} Bank 4*	-
GND	-	P89	P79	GND*	-
I, GCK0	4	P88	P80	N8	366
I/O	4	P87	P81	N9	370
I/O	4	P86	P82	R9	373
I/O	4	-	-	N10	376
I/O	4	-	P83	T9	379
I/O, V_{REF}	4	P85	P84	P9	382
GND	-	-	P85	GND*	-
I/O	4	-	P86	M10	385
I/O	4	-	P87	R10	388
I/O	4	-	P88	P10	391
I/O	4	P84	P89	T10	397
I/O	4	P83	P90	R11	400
V_{CCINT}	-	P82	P91	V_{CCINT}*	-
V_{CCO}	4	-	P92	V_{CCO} Bank 4*	-
GND	-	P81	P93	GND*	-
I/O	4	P80	P94	M11	403
I/O	4	P79	P95	T11	406
I/O	4	P78	P96	N11	409
I/O	4	-	-	R12	412
I/O	4	-	P97	P11	415
I/O, V_{REF}	4	P77	P98	T12	418
GND	-	-	-	GND*	-
I/O	4	-	P99	T13	421
I/O	4	-	-	N12	424
I/O	4	P76	P100	R13	427

XC2S50 Device Pinouts *(Continued)*

XC2S50 Pad Name					
Function	Bank	TQ144	PQ208	FG256	Bndry Scan
I/O	4	-	-	P12	430
I/O	4	P75	P101	P13	433
I/O	4	P74	P102	T14	436
GND	-	P73	P103	GND*	-
DONE	3	P72	P104	R14	439
V_{CCO}	4	P71	P105	V_{CCO} Bank 4*	-
V_{CCO}	3	P70	P105	V_{CCO} Bank 3*	-
$\overline{PROGRAM}$	-	P69	P106	P15	442
I/O ($\overline{INIT}$)	3	P68	P107	N15	443
I/O (D7)	3	P67	P108	N14	446
I/O	3	-	-	T15	449
I/O	3	P66	P109	M13	452
I/O	3	-	-	R16	455
I/O	3	-	P110	M14	458
GND	-	-	-	GND*	-
I/O, V_{REF}	3	P65	P111	L14	461
I/O	3	-	P112	M15	464
I/O	3	-	-	L12	467
I/O	3	P64	P113	P16	470
I/O	3	P63	P114	L13	473
I/O (D6)	3	P62	P115	N16	476
GND	-	P61	P116	GND*	-
V_{CCO}	3	-	P117	V_{CCO} Bank 3*	-
V_{CCINT}	-	-	P118	V_{CCINT}*	-
I/O (D5)	3	P60	P119	M16	479
I/O	3	P59	P120	K14	482
I/O	3	-	-	L16	485
I/O	3	-	P121	K13	488
I/O	3	-	P122	L15	491
I/O	3	-	P123	K12	494
GND	-	-	P124	GND*	-
I/O, V_{REF}	3	P58	P125	K16	497
I/O (D4)	3	P57	P126	J16	500
I/O	3	-	-	J14	503
I/O	3	P56	P127	K15	506
V_{CCINT}	-	P55	P128	V_{CCINT}*	-
I/O, TRDY(1)	3	P54	P129	J15	512
V_{CCO}	3	P53	P130	V_{CCO} Bank 3*	-

XC2S50 Device Pinouts *(Continued)*

XC2S50 Pad Name					
Function	Bank	TQ144	PQ208	FG256	Bndry Scan
V_{CCO}	2	P53	P130	V_{CCO} Bank 2*	-
GND	-	P52	P131	GND*	-
I/O, IRDY(1)	2	P51	P132	H16	515
I/O	2	-	P133	H14	518
I/O	2	P50	P134	H15	521
I/O	2	-	-	J13	524
I/O (D3)	2	P49	P135	G16	527
I/O, V_{REF}	2	P48	P136	H13	530
GND	-	-	P137	GND*	-
I/O	2	-	P138	G14	533
I/O	2	-	P139	G15	536
I/O	2	-	P140	G12	539
I/O	2	-	-	F16	542
I/O	2	P47	P141	G13	545
I/O (D2)	2	P46	P142	F15	548
V_{CCINT}	-	-	P143	V_{CCINT}*	-
V_{CCO}	2	-	P144	V_{CCO} Bank 2*	-
GND	-	P45	P145	GND*	-
I/O (D1)	2	P44	P146	E16	551
I/O	2	P43	P147	F14	554
I/O	2	P42	P148	D16	557
I/O	2	-	-	F12	560
I/O	2	-	P149	E15	563
I/O, V_{REF}	2	P41	P150	F13	566
GND	-	-	-	GND*	-
I/O	2	-	P151	E14	569
I/O	2	-	-	C16	572
I/O	2	P40	P152	E13	575
I/O	2	-	-	B16	578
I/O (DIN, D0)	2	P39	P153	D14	581
I/O (DOUT, BUSY)	2	P38	P154	C15	584
CCLK	2	P37	P155	D15	587
V_{CCO}	2	P36	P156	V_{CCO} Bank 2*	-
V_{CCO}	1	P35	P156	V_{CCO} Bank 1*	-
TDO	2	P34	P157	B14	-
GND	-	P33	P158	GND*	-
TDI	-	P32	P159	A15	-
I/O ($\overline{CS}$)	1	P31	P160	B13	0

XC2S50 Device Pinouts *(Continued)*

XC2S50 Pad Name					
Function	Bank	TQ144	PQ208	FG256	Bndry Scan
I/O ($\overline{WRITE}$)	1	P30	P161	C13	3
I/O	1	-	-	C12	6
I/O	1	P29	P162	A14	9
I/O	1	-	-	D12	12
I/O	1	-	P163	B12	15
GND	-	-	-	GND*	-
I/O, V_{REF}	1	P28	P164	C11	18
I/O	1	-	P165	A13	21
I/O	1	-	-	D11	24
I/O	1	-	P166	A12	27
I/O	1	P27	P167	E11	30
I/O	1	P26	P168	B11	33
GND	-	P25	P169	GND*	-
V_{CCO}	1	-	P170	V_{CCO} Bank 1*	-
V_{CCINT}	-	P24	P171	V_{CCINT}*	-
I/O	1	P23	P172	A11	36
I/O	1	P22	P173	C10	39
I/O	1	-	P174	B10	45
I/O	1	-	P175	D10	48
I/O	1	-	P176	A10	51
GND	-	-	P177	GND*	-
I/O, V_{REF}	1	P21	P178	B9	54
I/O	1	-	P179	E10	57
I/O	1	-	-	A9	60
I/O	1	P20	P180	D9	63
I/O	1	P19	P181	A8	66
I, GCK2	1	P18	P182	C9	72
GND	-	P17	P183	GND*	-
V_{CCO}	1	P16	P184	V_{CCO} Bank 1*	-
V_{CCO}	0	P16	P184	V_{CCO} Bank 0*	-
I, GCK3	0	P15	P185	B8	73
V_{CCINT}	-	P14	P186	V_{CCINT}*	-
I/O	0	P13	P187	A7	80
I/O	0	-	-	D8	83
I/O	0	-	P188	A6	86
I/O, V_{REF}	0	P12	P189	B7	89
GND	-	-	P190	GND*	-
I/O	0	-	P191	C8	92
I/O	0	-	P192	D7	95

XC2S50 Device Pinouts *(Continued)*

XC2S50 Pad Name					
Function	Bank	TQ144	PQ208	FG256	Bndry Scan
I/O	0	-	P193	E7	98
I/O	0	P11	P194	C7	104
I/O	0	P10	P195	B6	107
V_{CCINT}	-	P9	P196	V_{CCINT}*	-
V_{CCO}	0	-	P197	V_{CCO} Bank 0*	-
GND	-	P8	P198	GND*	-
I/O	0	P7	P199	A5	110
I/O	0	P6	P200	C6	113
I/O	0	-	P201	B5	116
I/O	0	-	-	D6	119
I/O	0	-	P202	A4	122
I/O, V_{REF}	0	P5	P203	B4	125
GND	-	-	-	GND*	-
I/O	0	-	P204	E6	128
I/O	0	-	-	D5	131
I/O	0	P4	P205	A3	134
I/O	0	-	-	C5	137
I/O	0	P3	P206	B3	140
TCK	-	P2	P207	C4	-
V_{CCO}	0	P1	P208	V_{CCO} Bank 0*	-
V_{CCO}	7	P144	P208	V_{CCO} Bank 7*	-

04/18/01

Notes:

1. IRDY and TRDY can only be accessed when using Xilinx PCI cores.
2. Pads labelled GND*, V_{CCINT}*, V_{CCO} Bank 0*, V_{CCO} Bank 1*, V_{CCO} Bank 2*, V_{CCO} Bank 3*, V_{CCO} Bank 4*, V_{CCO} Bank 5*, V_{CCO} Bank 6*, V_{CCO} Bank 7* are internally bonded to independent ground or power planes within the package.

Additional XC2S50 Package Pins

TQ144

Not Connected Pins					
P104	P105	-	-	-	-

11/02/00

PQ208

Not Connected Pins					
P55	P56	-	-	-	-

11/02/00

FG256

V_{CCINT} Pins					
C3	C14	D4	D13	E5	E12
M5	M12	N4	N13	P3	P14
V_{CCO} Bank 0 Pins					
E8	F8	-	-	-	-
V_{CCO} Bank 1 Pins					
E9	F9	-	-	-	-
V_{CCO} Bank 2 Pins					
H11	H12	-	-	-	-
V_{CCO} Bank 3 Pins					
J11	J12	-	-	-	-
V_{CCO} Bank 4 Pins					
L9	M9	-	-	-	-
V_{CCO} Bank 5 Pins					
L8	M8	-	-	-	-
V_{CCO} Bank 6 Pins					
J5	J6	-	-	-	-
V_{CCO} Bank 7 Pins					
H5	H6	-	-	-	-
GND Pins					
A1	A16	B2	B15	F6	F7
F10	F11	G6	G7	G8	G9
G10	G11	H7	H8	H9	H10
J7	J8	J9	J10	K6	K7
K8	K9	K10	K11	L6	L7
L10	L11	R2	R15	T1	T16
Not Connected Pins					
P4	R4	-	-	-	-

11/02/00

XC2S100 Device Pinouts

XC2S100 Pad Name						
Function	Bank	TQ144	PQ208	FG256	FG456	Bndry Scan
GND	-	P143	P1	GND*	GND*	-
TMS	-	P142	P2	D3	D3	-
I/O	7	P141	P3	C2	B1	185
I/O	7	-	-	A2	F5	191
I/O	7	P140	P4	B1	D2	194
I/O	7	-	-	-	E3	197
I/O	7	-	-	E3	G5	200
I/O	7	-	P5	D2	F3	203
GND	-	-	-	GND*	GND*	-
V_{CCO}	7	-	-	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-
I/O, V_{REF}	7	P139	P6	C1	E2	206
I/O	7	-	P7	F3	E1	209
I/O	7	-	-	E2	H5	215
I/O	7	P138	P8	E4	F2	218
I/O	7	-	-	-	F1	221
I/O, V_{REF}	7	P137	P9	D1	H4	224
I/O	7	P136	P10	E1	G1	227
GND	-	P135	P11	GND*	GND*	-
V_{CCO}	7	-	P12	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-
V_{CCINT}	-	-	P13	V_{CCINT}*	V_{CCINT}*	-
I/O	7	P134	P14	F2	H3	230
I/O	7	P133	P15	G3	H2	233
I/O	7	-	-	F1	J5	236
I/O	7	-	P16	F4	J2	239
I/O	7	-	P17	F5	K5	245
I/O	7	-	P18	G2	K1	248
GND	-	-	P19	GND*	GND*	-
I/O, V_{REF}	7	P132	P20	H3	K3	251
I/O	7	P131	P21	G4	K4	254
I/O	7	-	-	H2	L6	257
I/O	7	P130	P22	G5	L1	260
I/O	7	-	P23	H4	L4	266
I/O, IRDY(1)	7	P129	P24	G1	L3	269
GND	-	P128	P25	GND*	GND*	-
V_{CCO}	7	P127	P26	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-
V_{CCO}	6	P127	P26	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
I/O, TRDY(1)	6	P126	P27	J2	M1	272

XC2S100 Device Pinouts *(Continued)*

XC2S100 Pad Name						
Function	Bank	TQ144	PQ208	FG256	FG456	Bndry Scan
V_{CCINT}	-	P125	P28	V_{CCINT}*	V_{CCINT}*	-
I/O	6	P124	P29	H1	M3	281
I/O	6	-	-	J4	M4	284
I/O	6	P123	P30	J1	M5	287
I/O, V_{REF}	6	P122	P31	J3	N2	290
GND	-	-	P32	GND*	GND*	-
I/O	6	-	P33	K5	N3	293
I/O	6	-	P34	K2	N4	296
I/O	6	-	P35	K1	P2	302
I/O	6	-	-	K3	P4	305
I/O	6	P121	P36	L1	P3	308
I/O	6	P120	P37	L2	R2	311
V_{CCINT}	-	-	P38	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	6	-	P39	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
GND	-	P119	P40	GND*	GND*	-
I/O	6	P118	P41	K4	T1	314
I/O, V_{REF}	6	P117	P42	M1	R4	317
I/O	6	-	-	-	T2	320
I/O	6	P116	P43	L4	U1	323
I/O	6	-	-	M2	R5	326
I/O	6	-	P44	L3	U2	332
I/O, V_{REF}	6	P115	P45	N1	T3	335
V_{CCO}	6	-	-	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
GND	-	-	-	GND*	GND*	-
I/O	6	-	P46	P1	T4	338
I/O	6	-	-	L5	W1	341
I/O	6	-	-	-	U4	344
I/O	6	P114	P47	N2	Y1	347
I/O	6	-	-	M4	W2	350
I/O	6	P113	P48	R1	Y2	356
I/O	6	P112	P49	M3	W3	359
M1	-	P111	P50	P2	U5	362
GND	-	P110	P51	GND*	GND*	-
M0	-	P109	P52	N3	AB2	363
V_{CCO}	6	P108	P53	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
V_{CCO}	5	P107	P53	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
M2	-	P106	P54	R3	Y4	364
I/O	5	-	-	N5	V7	374

XC2S100 Device Pinouts *(Continued)*

XC2S100 Pad Name						
Function	Bank	TQ144	PQ208	FG256	FG456	Bndry Scan
I/O	5	P103	P57	T2	Y6	377
I/O	5	-	-	-	AA4	380
I/O	5	-	-	P5	W6	383
I/O	5	-	P58	T3	Y7	386
GND	-	-	-	GND*	GND*	-
V_{CCO}	5	-	-	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
I/O, V_{REF}	5	P102	P59	T4	AA5	389
I/O	5	-	P60	M6	AB5	392
I/O	5	-	-	T5	AB6	398
I/O	5	P101	P61	N6	AA7	401
I/O	5	-	-	-	W7	404
I/O, V_{REF}	5	P100	P62	R5	W8	407
I/O	5	P99	P63	P6	Y8	410
GND	-	P98	P64	GND*	GND*	-
V_{CCO}	5	-	P65	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
V_{CCINT}	-	P97	P66	V_{CCINT}*	V_{CCINT}*	-
I/O	5	P96	P67	R6	AA8	413
I/O	5	P95	P68	M7	V9	416
I/O	5	-	-	-	AB9	419
I/O	5	-	P69	N7	Y9	422
I/O	5	-	P70	T6	W10	428
I/O	5	-	P71	P7	AB10	431
GND	-	-	P72	GND*	GND*	-
I/O, V_{REF}	5	P94	P73	P8	Y10	434
I/O	5	-	P74	R7	V11	437
I/O	5	-	-	T7	W11	440
I/O	5	P93	P75	T8	AB11	443
V_{CCINT}	-	P92	P76	V_{CCINT}*	V_{CCINT}*	-
I, GCK1	5	P91	P77	R8	Y11	455
V_{CCO}	5	P90	P78	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
V_{CCO}	4	P90	P78	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
GND	-	P89	P79	GND*	GND*	-
I, GCK0	4	P88	P80	N8	W12	456
I/O	4	P87	P81	N9	U12	460
I/O	4	P86	P82	R9	Y12	466
I/O	4	-	-	N10	AA12	469
I/O	4	-	P83	T9	AB13	472
I/O, V_{REF}	4	P85	P84	P9	AA13	475

XC2S100 Device Pinouts *(Continued)*

XC2S100 Pad Name						
Function	Bank	TQ144	PQ208	FG256	FG456	Bndry Scan
GND	-	-	P85	GND*	GND*	-
I/O	4	-	P86	M10	Y13	478
I/O	4	-	P87	R10	V13	481
I/O	4	-	P88	P10	AA14	487
I/O	4	-	-	-	V14	490
I/O	4	P84	P89	T10	AB15	493
I/O	4	P83	P90	R11	AA15	496
V_{CCINT}	-	P82	P91	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	4	-	P92	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
GND	-	P81	P93	GND*	GND*	-
I/O	4	P80	P94	M11	Y15	499
I/O, V_{REF}	4	P79	P95	T11	AB16	502
I/O	4	-	-	-	AB17	505
I/O	4	P78	P96	N11	V15	508
I/O	4	-	-	R12	Y16	511
I/O	4	-	P97	P11	AB18	517
I/O, V_{REF}	4	P77	P98	T12	AB19	520
V_{CCO}	4	-	-	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
GND	-	-	-	GND*	GND*	-
I/O	4	-	P99	T13	Y17	523
I/O	4	-	-	N12	V16	526
I/O	4	-	-	-	W17	529
I/O	4	P76	P100	R13	AB20	532
I/O	4	-	-	P12	AA19	535
I/O	4	P75	P101	P13	AA20	541
I/O	4	P74	P102	T14	W18	544
GND	-	P73	P103	GND*	GND*	-
DONE	3	P72	P104	R14	Y19	547
V_{CCO}	4	P71	P105	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
V_{CCO}	3	P70	P105	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
$\overline{PROGRAM}$	-	P69	P106	P15	W20	550
I/O ($\overline{INIT}$)	3	P68	P107	N15	V19	551
I/O (D7)	3	P67	P108	N14	Y21	554
I/O	3	-	-	T15	W21	560
I/O	3	P66	P109	M13	U20	563
I/O	3	-	-	-	U19	566
I/O	3	-	-	R16	T18	569
I/O	3	-	P110	M14	W22	572

XC2S100 Device Pinouts *(Continued)*

XC2S100 Pad Name		TQ144	PQ208	FG256	FG456	Bndry Scan
Function	Bank					
GND	-	-	-	GND*	GND*	-
V_{CCO}	3	-	-	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
I/O, V_{REF}	3	P65	P111	L14	U21	575
I/O	3	-	P112	M15	T20	578
I/O	3	-	-	L12	T21	584
I/O	3	P64	P113	P16	R18	587
I/O	3	-	-	-	U22	590
I/O, V_{REF}	3	P63	P114	L13	R19	593
I/O (D6)	3	P62	P115	N16	T22	596
GND	-	P61	P116	GND*	GND*	-
V_{CCO}	3	-	P117	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
V_{CCINT}	-	-	P118	V_{CCINT}*	V_{CCINT}*	-
I/O (D5)	3	P60	P119	M16	R21	599
I/O	3	P59	P120	K14	P18	602
I/O	3	-	-	L16	P20	605
I/O	3	-	P121	K13	P21	608
I/O	3	-	P122	L15	N18	614
I/O	3	-	P123	K12	N20	617
GND	-	-	P124	GND*	GND*	-
I/O, V_{REF}	3	P58	P125	K16	N21	620
I/O (D4)	3	P57	P126	J16	N22	623
I/O	3	-	-	J14	M19	626
I/O	3	P56	P127	K15	M20	629
V_{CCINT}	-	P55	P128	E5	V_{CCINT}*	-
I/O, TRDY(1)	3	P54	P129	J15	M22	638
V_{CCO}	3	P53	P130	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
V_{CCO}	2	P53	P130	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
GND	-	P52	P131	GND*	GND*	-
I/O, IRDY(1)	2	P51	P132	H16	L20	641
I/O	2	-	P133	H14	L17	644
I/O	2	P50	P134	H15	L21	650
I/O	2	-	-	J13	L22	653
I/O (D3)	2	P49	P135	G16	K20	656
I/O, V_{REF}	2	P48	P136	H13	K21	659
GND	-	-	P137	GND*	GND*	-
I/O	2	-	P138	G14	K22	662
I/O	2	-	P139	G15	J21	665
I/O	2	-	P140	G12	J18	671

XC2S100 Device Pinouts *(Continued)*

XC2S100 Pad Name		TQ144	PQ208	FG256	FG456	Bndry Scan
Function	Bank					
I/O	2	-	-	F16	J22	674
I/O	2	P47	P141	G13	H19	677
I/O (D2)	2	P46	P142	F15	H20	680
V_{CCINT}	-	-	P143	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	2	-	P144	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
GND	-	P45	P145	GND*	GND*	-
I/O (D1)	2	P44	P146	E16	H22	683
I/O, V_{REF}	2	P43	P147	F14	H18	686
I/O	2	-	-	-	G21	689
I/O	2	P42	P148	D16	G18	692
I/O	2	-	-	F12	G20	695
I/O	2	-	P149	E15	F19	701
I/O, V_{REF}	2	P41	P150	F13	F21	704
V_{CCO}	2	-	-	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
GND	-	-	-	GND*	GND*	-
I/O	2	-	P151	E14	F20	707
I/O	2	-	-	C16	F18	710
I/O	2	-	-	-	E21	713
I/O	2	P40	P152	E13	D22	716
I/O	2	-	-	B16	E20	719
I/O (DIN, D0)	2	P39	P153	D14	D20	725
I/O (DOUT, BUSY)	2	P38	P154	C15	C21	728
CCLK	2	P37	P155	D15	B22	731
V_{CCO}	2	P36	P156	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
V_{CCO}	1	P35	P156	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
TDO	2	P34	P157	B14	A21	-
GND	-	P33	P158	GND*	GND*	-
TDI	-	P32	P159	A15	B20	-
I/O ($\overline{CS}$)	1	P31	P160	B13	C19	0
I/O ($\overline{WRITE}$)	1	P30	P161	C13	A20	3
I/O	1	-	-	C12	D17	9
I/O	1	P29	P162	A14	A19	12
I/O	1	-	-	-	B18	15
I/O	1	-	-	D12	C17	18
I/O	1	-	P163	B12	D16	21
GND	-	-	-	GND*	GND*	-

XC2S100 Device Pinouts *(Continued)*

XC2S100 Pad Name						
Function	Bank	TQ144	PQ208	FG256	FG456	Bndry Scan
V_{CCO}	1	-	-	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
I/O, V_{REF}	1	P28	P164	C11	A18	24
I/O	1	-	P165	A13	B17	27
I/O	1	-	-	D11	D15	33
I/O	1	-	P166	A12	C16	36
I/O	1	-	-	-	D14	39
I/O, V_{REF}	1	P27	P167	E11	E14	42
I/O	1	P26	P168	B11	A16	45
GND	-	P25	P169	GND*	GND*	-
V_{CCO}	1	-	P170	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
V_{CCINT}	-	P24	P171	V_{CCINT}*	V_{CCINT}*	-
I/O	1	P23	P172	A11	C15	48
I/O	1	P22	P173	C10	B15	51
I/O	1	-	-	-	F12	54
I/O	1	-	P174	B10	C14	57
I/O	1	-	P175	D10	D13	63
I/O	1	-	P176	A10	C13	66
GND	-	-	P177	GND*	GND*	-
I/O, V_{REF}	1	P21	P178	B9	B13	69
I/O	1	-	P179	E10	E12	72
I/O	1	-	-	A9	B12	75
I/O	1	P20	P180	D9	D12	78
I/O	1	P19	P181	A8	D11	84
I, GCK2	1	P18	P182	C9	A11	90
GND	-	P17	P183	GND*	GND*	-
V_{CCO}	1	P16	P184	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
V_{CCO}	0	P16	P184	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
I, GCK3	0	P15	P185	B8	C11	91
V_{CCINT}	-	P14	P186	V_{CCINT}*	V_{CCINT}*	-
I/O	0	P13	P187	A7	A10	101
I/O	0	-	-	D8	B10	104

XC2S100 Device Pinouts *(Continued)*

XC2S100 Pad Name						
Function	Bank	TQ144	PQ208	FG256	FG456	Bndry Scan
I/O	0	-	P188	A6	C10	107
I/O, V_{REF}	0	P12	P189	B7	A9	110
GND	-	-	P190	GND*	GND*	-
I/O	0	-	P191	C8	B9	113
I/O	0	-	P192	D7	E10	116
I/O	0	-	P193	E7	A8	122
I/O	0	-	-	-	D9	125
I/O	0	P11	P194	C7	E9	128
I/O	0	P10	P195	B6	A7	131
V_{CCINT}	-	P9	P196	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	0	-	P197	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
GND	-	P8	P198	GND*	GND*	-
I/O	0	P7	P199	A5	B7	134
I/O, V_{REF}	0	P6	P200	C6	E8	137
I/O	0	-	-	-	D8	140
I/O	0	-	P201	B5	C7	143
I/O	0	-	-	D6	D7	146
I/O	0	-	P202	A4	D6	152
I/O, V_{REF}	0	P5	P203	B4	C6	155
V_{CCO}	0	-	-	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
GND	-	-	-	GND*	GND*	-
I/O	0	-	P204	E6	B5	158
I/O	0	-	-	D5	E7	161
I/O	0	-	-	-	E6	164
I/O	0	P4	P205	A3	B4	167
I/O	0	-	-	C5	A3	170
I/O	0	P3	P206	B3	C5	176
TCK	-	P2	P207	C4	C4	-
V_{CCO}	0	P1	P208	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
V_{CCO}	7	P144	P208	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-

04/18/01

Notes:

1. IRDY and TRDY can only be accessed when using Xilinx PCI cores.
2. Pads labelled GND*, V_{CCINT}*, V_{CCO} Bank 0*, V_{CCO} Bank 1*, V_{CCO} Bank 2*, V_{CCO} Bank 3*, V_{CCO} Bank 4*, V_{CCO} Bank 5*, V_{CCO} Bank 6*, V_{CCO} Bank 7* are internally bonded to independent ground or power planes within the package.

Additional XC2S100 Package Pins

TQ144

Not Connected Pins					
P104	P105	-	-	-	-

11/02/00

PQ208

Not Connected Pins					
P55	P56	-	-	-	-

11/02/00

FG256

V_{CCINT} Pins					
C3	C14	D4	D13	E5	E12
M5	M12	N4	N13	P3	P14
V_{CCO} Bank 0 Pins					
E8	F8	-	-	-	-
V_{CCO} Bank 1 Pins					
E9	F9	-	-	-	-
V_{CCO} Bank 2 Pins					
H11	H12	-	-	-	-
V_{CCO} Bank 3 Pins					
J11	J12	-	-	-	-
V_{CCO} Bank 4 Pins					
L9	M9	-	-	-	-
V_{CCO} Bank 5 Pins					
L8	M8	-	-	-	-
V_{CCO} Bank 6 Pins					
J5	J6	-	-	-	-
V_{CCO} Bank 7 Pins					
H5	H6	-	-	-	-
GND Pins					
A1	A16	B2	B15	F6	F7
F10	F11	G6	G7	G8	G9
G10	G11	H7	H8	H9	H10
J7	J8	J9	J10	K6	K7
K8	K9	K10	K11	L6	L7
L10	L11	R2	R15	T1	T16
Not Connected Pins					
P4	R4	-	-	-	-

11/02/00

FG456

V_{CCINT} Pins					
E5	E18	F6	F17	G7	G8
G9	G14	G15	G16	H7	H16
J7	J16	P7	P16	R7	R16
T7	T8	T9	T14	T15	T16
U6	U17	V5	V18	-	-
V_{CCO} Bank 0 Pins					
F10	F7	F8	F9	G10	G11

Additional XC2S100 Package Pins *(Continued)*

V_{CCO} Bank 1 Pins					
F13	F14	F15	F16	G12	G13
V_{CCO} Bank 2 Pins					
G17	H17	J17	K16	K17	L16
V_{CCO} Bank 3 Pins					
M16	N16	N17	P17	R17	T17
V_{CCO} Bank 4 Pins					
T12	T13	U13	U14	U15	U16
V_{CCO} Bank 5 Pins					
T10	T11	U10	U7	U8	U9
V_{CCO} Bank 6 Pins					
M7	N6	N7	P6	R6	T6
V_{CCO} Bank 7 Pins					
G6	H6	J6	K6	K7	L7
GND Pins					
A1	A22	B2	B21	C3	C20
J9	J10	J11	J12	J13	J14
K9	K10	K11	K12	K13	K14
L9	L10	L11	L12	L13	L14
M9	M10	M11	M12	M13	M14
N9	N10	N11	N12	N13	N14
P9	P10	P11	P12	P13	P14
Y3	Y20	AA2	AA21	AB1	AB22
Not Connected Pins					
A2	A4	A5	A6	A12	A13
A14	A15	A17	B3	B6	B8
B11	B14	B16	B19	C1	C2
C8	C9	C12	C18	C22	D1
D4	D5	D10	D18	D19	D21
E4	E11	E13	E15	E16	E17
E19	E22	F4	F11	F22	G2
G3	G4	G19	G22	H1	H21
J1	J3	J4	J19	J20	K2
K18	K19	L2	L5	L18	L19
M2	M6	M17	M18	M21	N1
N5	N19	P1	P5	P19	P22
R1	R3	R20	R22	T5	T19
U3	U11	U18	V1	V2	V10
V12	V17	V3	V4	V6	V8
V20	V21	V22	W4	W5	W9
W13	W14	W15	W16	W19	Y5
Y14	Y18	Y22	AA1	AA3	AA6
AA9	AA10	AA11	AA16	AA17	AA18
AA22	AB3	AB4	AB7	AB8	AB12
AB14	AB21	-	-	-	-

11/02/00

XC2S150 Device Pinouts

XC2S150 Pad Name					Bndry Scan
Function	Bank	PQ208	FG256	FG456	
GND	-	P1	GND*	GND*	-
TMS	-	P2	D3	D3	-
I/O	7	P3	C2	B1	221
I/O	7	-	-	E4	224
I/O	7	-	-	C1	227
I/O	7	-	A2	F5	230
GND	-	-	GND*	GND*	-
I/O	7	P4	B1	D2	233
I/O	7	-	-	E3	236
I/O	7	-	-	F4	239
I/O	7	-	E3	G5	242
I/O	7	P5	D2	F3	245
GND	-	-	GND*	GND*	-
V_{CCO}	7	-	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-
I/O, V_{REF}	7	P6	C1	E2	248
I/O	7	P7	F3	E1	251
I/O	7	-	-	G4	254
I/O	7	-	-	G3	257
I/O	7	-	E2	H5	260
I/O	7	P8	E4	F2	263
I/O	7	-	-	F1	266
I/O, V_{REF}	7	P9	D1	H4	269
I/O	7	P10	E1	G1	272
GND	-	P11	GND*	GND*	-
V_{CCO}	7	P12	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-
V_{CCINT}	-	P13	V_{CCINT}*	V_{CCINT}*	-
I/O	7	P14	F2	H3	275
I/O	7	P15	G3	H2	278
I/O	7	-	-	H1	284
I/O	7	-	F1	J5	287
I/O	7	P16	F4	J2	290
I/O	7	-	-	J3	293
I/O	7	P17	F5	K5	299
I/O	7	P18	G2	K1	302
GND	-	P19	GND*	GND*	-
V_{CCO}	7	-	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-
I/O, V_{REF}	7	P20	H3	K3	305
I/O	7	P21	G4	K4	308
I/O	7	-	H2	L6	311

XC2S150 Device Pinouts *(Continued)*

XC2S150 Pad Name					Bndry Scan
Function	Bank	PQ208	FG256	FG456	
I/O	7	P22	G5	L1	314
I/O	7	-	-	L5	317
I/O	7	P23	H4	L4	320
I/O, IRDY(1)	7	P24	G1	L3	323
GND	-	P25	GND*	GND*	-
V_{CCO}	7	P26	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-
V_{CCO}	6	P26	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
I/O, TRDY(1)	6	P27	J2	M1	326
V_{CCINT}	-	P28	V_{CCINT}*	V_{CCINT}*	-
I/O	6	-	-	M6	332
I/O	6	P29	H1	M3	335
I/O	6	-	J4	M4	338
I/O	6	P30	J1	M5	341
I/O, V_{REF}	6	P31	J3	N2	344
V_{CCO}	6	-	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
GND	-	P32	GND*	GND*	-
I/O	6	P33	K5	N3	347
I/O	6	P34	K2	N4	350
I/O	6	-	-	N5	356
I/O	6	P35	K1	P2	359
I/O	6	-	K3	P4	362
I/O	6	-	-	R1	365
I/O	6	P36	L1	P3	371
I/O	6	P37	L2	R2	374
V_{CCINT}	-	P38	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	6	P39	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
GND	-	P40	GND*	GND*	-
I/O	6	P41	K4	T1	377
I/O, V_{REF}	6	P42	M1	R4	380
I/O	6	-	-	T2	383
I/O	6	P43	L4	U1	386
I/O	6	-	M2	R5	389
I/O	6	-	-	V1	392
I/O	6	-	-	T5	395
I/O	6	P44	L3	U2	398
I/O, V_{REF}	6	P45	N1	T3	401
V_{CCO}	6	-	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
GND	-	-	GND*	GND*	-

XC2S150 Device Pinouts *(Continued)*

XC2S150 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
I/O	6	P46	P1	T4	404
I/O	6	-	L5	W1	407
I/O	6	-	-	V2	410
I/O	6	-	-	U4	413
I/O	6	P47	N2	Y1	416
GND	-	-	GND*	GND*	-
I/O	6	-	M4	W2	419
I/O	6	-	-	V3	422
I/O	6	-	-	V4	425
I/O	6	P48	R1	Y2	428
I/O	6	P49	M3	W3	431
M1	-	P50	P2	U5	434
GND	-	P51	GND*	GND*	-
M0	-	P52	N3	AB2	435
V_{CCO}	6	P53	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
V_{CCO}	5	P53	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
M2	-	P54	R3	Y4	436
I/O	5	-	-	W5	443
I/O	5	-	-	AB3	446
I/O	5	-	N5	V7	449
GND	-	-	GND*	GND*	-
I/O	5	P57	T2	Y6	452
I/O	5	-	-	AA4	455
I/O	5	-	-	AB4	458
I/O	5	-	P5	W6	461
I/O	5	P58	T3	Y7	464
GND	-	-	GND*	GND*	-
V_{CCO}	5	-	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
I/O, V_{REF}	5	P59	T4	AA5	467
I/O	5	P60	M6	AB5	470
I/O	5	-	-	V8	473
I/O	5	-	-	AA6	476
I/O	5	-	T5	AB6	479
I/O	5	P61	N6	AA7	482
I/O	5	-	-	W7	485
I/O, V_{REF}	5	P62	R5	W8	488
I/O	5	P63	P6	Y8	491
GND	-	P64	GND*	GND*	-

XC2S150 Device Pinouts *(Continued)*

XC2S150 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
V_{CCO}	5	P65	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
V_{CCINT}	-	P66	V_{CCINT}*	V_{CCINT}*	-
I/O	5	P67	R6	AA8	494
I/O	5	P68	M7	V9	497
I/O	5	-	-	W9	503
I/O	5	-	-	AB9	506
I/O	5	P69	N7	Y9	509
I/O	5	-	-	V10	512
I/O	5	P70	T6	W10	518
I/O	5	P71	P7	AB10	521
GND	-	P72	GND*	GND*	-
V_{CCO}	5	-	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
I/O, V_{REF}	5	P73	P8	Y10	524
I/O	5	P74	R7	V11	527
I/O	5	-	T7	W11	530
I/O	5	P75	T8	AB11	533
I/O	5	-	-	U11	536
V_{CCINT}	-	P76	V_{CCINT}*	V_{CCINT}*	-
I, GCK1	5	P77	R8	Y11	545
V_{CCO}	5	P78	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
V_{CCO}	4	P78	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
GND	-	P79	GND*	GND*	-
I, GCK0	4	P80	N8	W12	546
I/O	4	P81	N9	U12	550
I/O	4	-	-	V12	553
I/O	4	P82	R9	Y12	556
I/O	4	-	N10	AA12	559
I/O	4	P83	T9	AB13	562
I/O, V_{REF}	4	P84	P9	AA13	565
V_{CCO}	4	-	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
GND	-	P85	GND*	GND*	-
I/O	4	P86	M10	Y13	568
I/O	4	P87	R10	V13	571
I/O	4	-	-	W14	577
I/O	4	P88	P10	AA14	580
I/O	4	-	-	V14	583
I/O	4	-	-	Y14	586
I/O	4	P89	T10	AB15	592

XC2S150 Device Pinouts *(Continued)*

XC2S150 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
I/O	4	P90	R11	AA15	595
V_{CCINT}	-	P91	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	4	P92	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
GND	-	P93	GND*	GND*	-
I/O	4	P94	M11	Y15	598
I/O, V_{REF}	4	P95	T11	AB16	601
I/O	4	-	-	AB17	604
I/O	4	P96	N11	V15	607
I/O	4	-	R12	Y16	610
I/O	4	-	-	AA17	613
I/O	4	-	-	W16	616
I/O	4	P97	P11	AB18	619
I/O, V_{REF}	4	P98	T12	AB19	622
V_{CCO}	4	-	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
GND	-	-	GND*	GND*	-
I/O	4	P99	T13	Y17	625
I/O	4	-	N12	V16	628
I/O	4	-	-	AA18	631
I/O	4	-	-	W17	634
I/O	4	P100	R13	AB20	637
GND	-	-	GND*	GND*	-
I/O	4	-	P12	AA19	640
I/O	4	-	-	V17	643
I/O	4	-	-	Y18	646
I/O	4	P101	P13	AA20	649
I/O	4	P102	T14	W18	652
GND	-	P103	GND*	GND*	-
DONE	3	P104	R14	Y19	655
V_{CCO}	4	P105	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
V_{CCO}	3	P105	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
$\overline{PROGRAM}$	-	P106	P15	W20	658
I/O ($\overline{INIT}$)	3	P107	N15	V19	659
I/O (D7)	3	P108	N14	Y21	662
I/O	3	-	-	V20	665
I/O	3	-	-	AA22	668
I/O	3	-	T15	W21	671
GND	-	-	GND*	GND*	-
I/O	3	P109	M13	U20	674

XC2S150 Device Pinouts *(Continued)*

XC2S150 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
I/O	3	-	-	U19	677
I/O	3	-	-	V21	680
I/O	3	-	R16	T18	683
I/O	3	P110	M14	W22	686
GND	-	-	GND*	GND*	-
V_{CCO}	3	-	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
I/O, V_{REF}	3	P111	L14	U21	689
I/O	3	P112	M15	T20	692
I/O	3	-	-	T19	695
I/O	3	-	-	V22	698
I/O	3	-	L12	T21	701
I/O	3	P113	P16	R18	704
I/O	3	-	-	U22	707
I/O, V_{REF}	3	P114	L13	R19	710
I/O (D6)	3	P115	N16	T22	713
GND	-	P116	GND*	GND*	-
V_{CCO}	3	P117	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
V_{CCINT}	-	P118	V_{CCINT}*	V_{CCINT}*	-
I/O (D5)	3	P119	M16	R21	716
I/O	3	P120	K14	P18	719
I/O	3	-	-	P19	725
I/O	3	-	L16	P20	728
I/O	3	P121	K13	P21	731
I/O	3	-	-	N19	734
I/O	3	P122	L15	N18	740
I/O	3	P123	K12	N20	743
GND	-	P124	GND*	GND*	-
V_{CCO}	3	-	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
I/O, V_{REF}	3	P125	K16	N21	746
I/O (D4)	3	P126	J16	N22	749
I/O	3	-	J14	M19	752
I/O	3	P127	K15	M20	755
I/O	3	-	-	M18	758
V_{CCINT}	-	P128	V_{CCINT}*	V_{CCINT}*	-
I/O, TRDY(1)	3	P129	J15	M22	764
V_{CCO}	3	P130	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
V_{CCO}	2	P130	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
GND	-	P131	GND*	GND*	-

XC2S150 Device Pinouts *(Continued)*

XC2S150 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
I/O, IRDY(1)	2	P132	H16	L20	767
I/O	2	P133	H14	L17	770
I/O	2	-	-	L18	773
I/O	2	P134	H15	L21	776
I/O	2	-	J13	L22	779
I/O (D3)	2	P135	G16	K20	782
I/O, V_{REF}	2	P136	H13	K21	785
V_{CCO}	2	-	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
GND	-	P137	GND*	GND*	-
I/O	2	P138	G14	K22	788
I/O	2	P139	G15	J21	791
I/O	2	-	-	J20	797
I/O	2	P140	G12	J18	800
I/O	2	-	F16	J22	803
I/O	2	-	-	J19	806
I/O	2	P141	G13	H19	812
I/O (D2)	2	P142	F15	H20	815
V_{CCINT}	-	P143	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	2	P144	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
GND	-	P145	GND*	GND*	-
I/O (D1)	2	P146	E16	H22	818
I/O, V_{REF}	2	P147	F14	H18	821
I/O	2	-	-	G21	824
I/O	2	P148	D16	G18	827
I/O	2	-	F12	G20	830
I/O	2	-	-	G19	833
I/O	2	-	-	F22	836
I/O	2	P149	E15	F19	839
I/O, V_{REF}	2	P150	F13	F21	842
V_{CCO}	2	-	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
GND	-	-	GND*	GND*	-
I/O	2	P151	E14	F20	845
I/O	2	-	C16	F18	848
I/O	2	-	-	E22	851
I/O	2	-	-	E21	854
I/O	2	P152	E13	D22	857
GND	-	-	GND*	GND*	-
I/O	2	-	B16	E20	860
I/O	2	-	-	D21	863

XC2S150 Device Pinouts *(Continued)*

XC2S150 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
I/O	2	-	-	C22	866
I/O (DIN, D0)	2	P153	D14	D20	869
I/O (DOUT, BUSY)	2	P154	C15	C21	872
CCLK	2	P155	D15	B22	875
V_{CCO}	2	P156	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
V_{CCO}	1	P156	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
TDO	2	P157	B14	A21	-
GND	-	P158	GND*	GND*	-
TDI	-	P159	A15	B20	-
I/O (CS)	1	P160	B13	C19	0
I/O (WRITE)	1	P161	C13	A20	3
I/O	1	-	-	B19	6
I/O	1	-	-	C18	9
I/O	1	-	C12	D17	12
GND	-	-	GND*	GND*	-
I/O	1	P162	A14	A19	15
I/O	1	-	-	B18	18
I/O	1	-	-	E16	21
I/O	1	-	D12	C17	24
I/O	1	P163	B12	D16	27
GND	-	-	GND*	GND*	-
V_{CCO}	1	-	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
I/O, V_{REF}	1	P164	C11	A18	30
I/O	1	P165	A13	B17	33
I/O	1	-	-	E15	36
I/O	1	-	-	A17	39
I/O	1	-	D11	D15	42
I/O	1	P166	A12	C16	45
I/O	1	-	-	D14	48
I/O, V_{REF}	1	P167	E11	E14	51
I/O	1	P168	B11	A16	54
GND	-	P169	GND*	GND*	-
V_{CCO}	1	P170	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
V_{CCINT}	-	P171	V_{CCINT}*	V_{CCINT}*	-
I/O	1	P172	A11	C15	57
I/O	1	P173	C10	B15	60
I/O	1	-	-	A15	66
I/O	1	-	-	F12	69

XC2S150 Device Pinouts *(Continued)*

XC2S150 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
I/O	1	P174	B10	C14	72
I/O	1	-	-	B14	75
I/O	1	P175	D10	D13	81
I/O	1	P176	A10	C13	84
GND	-	P177	GND*	GND*	-
V_{CCO}	1	-	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
I/O, V_{REF}	1	P178	B9	B13	87
I/O	1	P179	E10	E12	90
I/O	1	-	A9	B12	93
I/O	1	P180	D9	D12	96
I/O	1	-	-	C12	99
I/O	1	P181	A8	D11	102
I, GCK2	1	P182	C9	A11	108
GND	-	P183	GND*	GND*	-
V_{CCO}	1	P184	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
V_{CCO}	0	P184	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
I, GCK3	0	P185	B8	C11	109
V_{CCINT}	-	P186	V_{CCINT}*	V_{CCINT}*	-
I/O	0	-	-	E11	116
I/O	0	P187	A7	A10	119
I/O	0	-	D8	B10	122
I/O	0	P188	A6	C10	125
I/O, V_{REF}	0	P189	B7	A9	128
V_{CCO}	0	-	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
GND	-	P190	GND*	GND*	-
I/O	0	P191	C8	B9	131
I/O	0	P192	D7	E10	134
I/O	0	-	-	D10	140
I/O	0	P193	E7	A8	143
I/O	0	-	-	D9	146
I/O	0	-	-	B8	149
I/O	0	P194	C7	E9	155
I/O	0	P195	B6	A7	158

XC2S150 Device Pinouts *(Continued)*

XC2S150 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
V_{CCINT}	-	P196	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	0	P197	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
GND	-	P198	GND*	GND*	-
I/O	0	P199	A5	B7	161
I/O, V_{REF}	0	P200	C6	E8	164
I/O	0	-	-	D8	167
I/O	0	P201	B5	C7	170
I/O	0	-	D6	D7	173
I/O	0	-	-	B6	176
I/O	0	-	-	A5	179
I/O	0	P202	A4	D6	182
I/O, V_{REF}	0	P203	B4	C6	185
V_{CCO}	0	-	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
GND	-	-	GND*	GND*	-
I/O	0	P204	E6	B5	188
I/O	0	-	D5	E7	191
I/O	0	-	-	A4	194
I/O	0	-	-	E6	197
I/O	0	P205	A3	B4	200
GND	-	-	GND*	GND*	-
I/O	0	-	C5	A3	203
I/O	0	-	-	B3	206
I/O	0	-	-	D5	209
I/O	0	P206	B3	C5	212
TCK	-	P207	C4	C4	-
V_{CCO}	0	P208	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
V_{CCO}	7	P208	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-

04/18/01

Notes:

1. IRDY and TRDY can only be accessed when using Xilinx PCI cores.
2. Pads labelled GND*, V_{CCINT}*, V_{CCO} Bank 0*, V_{CCO} Bank 1*, V_{CCO} Bank 2*, V_{CCO} Bank 3*, V_{CCO} Bank 4*, V_{CCO} Bank 5*, V_{CCO} Bank 6*, V_{CCO} Bank 7* are internally bonded to independent ground or power planes within the package.

Additional XC2S150 Package Pins

PQ208

Not Connected Pins					
P55	P56	-	-	-	-

11/02/00

FG256

V_{CCINT} Pins					
C3	C14	D4	D13	E5	E12
M5	M12	N4	N13	P3	P14
V_{CCO} Bank 0 Pins					
E8	F8	-	-	-	-
V_{CCO} Bank 1 Pins					
E9	F9	-	-	-	-
V_{CCO} Bank 2 Pins					
H11	H12	-	-	-	-
V_{CCO} Bank 3 Pins					
J11	J12	-	-	-	-
V_{CCO} Bank 4 Pins					
L9	M9	-	-	-	-
V_{CCO} Bank 5 Pins					
L8	M8	-	-	-	-
V_{CCO} Bank 6 Pins					
J5	J6	-	-	-	-
V_{CCO} Bank 7 Pins					
H5	H6	-	-	-	-
GND Pins					
A1	A16	B2	B15	F6	F7
F10	F11	G6	G7	G8	G9
G10	G11	H7	H8	H9	H10
J7	J8	J9	J10	K6	K7
K8	K9	K10	K11	L6	L7
L10	L11	R2	R15	T1	T16
Not Connected Pins					
P4	R4	-	-	-	-

11/02/00

Additional XC2S150 Package Pins *(Continued)*

FG456

V_{CCINT} Pins					
E5	E18	F6	F17	G7	G8
G9	G14	G15	G16	H7	H16
J7	J16	P7	P16	R7	R16
T7	T8	T9	T14	T15	T16
U6	U17	V5	V18	-	-
V_{CCO} BANK 0 Pins					
F7	F8	F9	F10	G10	G11
V_{CCO} Bank 1 Pins					
F13	F14	F15	F16	G12	G13
V_{CCO} Bank 2 Pins					
G17	H17	J17	K16	K17	L16
V_{CCO} Bank 3 Pins					
M16	N16	N17	P17	R17	T17
V_{CCO} Bank 4 Pins					
T12	T13	U13	U14	U15	U16
V_{CCO} Bank 5 Pins					
T10	T11	U7	U8	U9	U10
V_{CCO} Bank 6 Pins					
M7	N6	N7	P6	R6	T6
V_{CCO} Bank 7 Pins					
G6	H6	J6	K6	K7	L7
GND Pins					
A1	A22	B2	B21	C3	C20
J9	J10	J11	J12	J13	J14
K9	K10	K11	K12	K13	K14
L9	L10	L11	L12	L13	L14
M9	M10	M11	M12	M13	M14
N9	N10	N11	N12	N13	N14
P9	P10	P11	P12	P13	P14
Y3	Y20	AA2	AA21	AB1	AB22
Not Connected Pins					
A2	A6	A12	A13	A14	B11
B16	C2	C8	C9	D1	D4
D18	D19	E13	E17	E19	F11
G2	G22	H21	J1	J4	K2
K18	K19	L2	L19	M2	M17
M21	N1	P1	P5	P22	R3
R20	R22	U3	U18	V6	W4
W13	W15	W19	Y5	Y22	AA1
AA3	AA9	AA10	AA11	AA16	AB7
AB8	AB12	AB14	AB21	-	-

11/02/00

XC2S200 Device Pinouts

XC2S200 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
GND	-	P1	GND*	GND*	-
TMS	-	P2	D3	D3	-
I/O	7	P3	C2	B1	257
I/O	7	-	-	E4	263
I/O	7	-	-	C1	266
I/O	7	-	A2	F5	269
GND	-	-	GND*	GND*	-
I/O, V_{REF}	7	P4	B1	D2	272
I/O	7	-	-	E3	275
I/O	7	-	-	F4	281
GND	-	-	GND*	GND*	-
I/O	7	-	E3	G5	284
I/O	7	P5	D2	F3	287
GND	-	-	GND*	GND*	-
V_{CCO}	7	-	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-
I/O, V_{REF}	7	P6	C1	E2	290
I/O	7	P7	F3	E1	293
I/O	7	-	-	G4	296
I/O	7	-	-	G3	299
I/O	7	-	E2	H5	302
GND	-	-	GND*	GND*	-
I/O	7	P8	E4	F2	305
I/O	7	-	-	F1	308
I/O, V_{REF}	7	P9	D1	H4	314
I/O	7	P10	E1	G1	317
GND	-	P11	GND*	GND*	-
V_{CCO}	7	P12	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-
V_{CCINT}	-	P13	V_{CCINT}*	V_{CCINT}*	-
I/O	7	P14	F2	H3	320
I/O	7	P15	G3	H2	323
I/O	7	-	-	J4	326
I/O	7	-	-	H1	329
I/O	7	-	F1	J5	332
GND	-	-	GND*	GND*	-
I/O	7	P16	F4	J2	335
I/O	7	-	-	J3	338
I/O	7	-	-	J1	341
I/O	7	P17	F5	K5	344
I/O	7	P18	G2	K1	347
GND	-	P19	GND*	GND*	-

XC2S200 Device Pinouts *(Continued)*

XC2S200 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
V_{CCO}	7	-	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-
I/O, V_{REF}	7	P20	H3	K3	350
I/O	7	P21	G4	K4	353
I/O	7	-	-	K2	359
I/O	7	-	H2	L6	362
I/O	7	P22	G5	L1	365
I/O	7	-	-	L5	368
I/O	7	P23	H4	L4	374
I/O, IRDY(1)	7	P24	G1	L3	377
GND	-	P25	GND*	GND*	-
V_{CCO}	7	P26	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-
V_{CCO}	6	P26	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
I/O, TRDY(1)	6	P27	J2	M1	380
V_{CCINT}	-	P28	V_{CCINT}*	V_{CCINT}*	-
I/O	6	-	-	M6	389
I/O	6	P29	H1	M3	392
I/O	6	-	J4	M4	395
I/O	6	-	-	N1	398
I/O	6	P30	J1	M5	404
I/O, V_{REF}	6	P31	J3	N2	407
V_{CCO}	6	-	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
GND	-	P32	GND*	GND*	-
I/O	6	P33	K5	N3	410
I/O	6	P34	K2	N4	413
I/O	6	-	-	P1	416
I/O	6	-	-	N5	419
I/O	6	P35	K1	P2	422
GND	-	-	GND*	GND*	-
I/O	6	-	K3	P4	425
I/O	6	-	-	R1	428
I/O	6	-	-	P5	431
I/O	6	P36	L1	P3	434
I/O	6	P37	L2	R2	437
V_{CCINT}	-	P38	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	6	P39	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
GND	-	P40	GND*	GND*	-
I/O	6	P41	K4	T1	440
I/O, V_{REF}	6	P42	M1	R4	443

XC2S200 Device Pinouts *(Continued)*

XC2S200 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
I/O	6	-	-	T2	449
I/O	6	P43	L4	U1	452
GND	-	-	GND*	GND*	-
I/O	6	-	M2	R5	455
I/O	6	-	-	V1	458
I/O	6	-	-	T5	461
I/O	6	P44	L3	U2	464
I/O, V_{REF}	6	P45	N1	T3	467
V_{CCO}	6	-	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
GND	-	-	GND*	GND*	-
I/O	6	P46	P1	T4	470
I/O	6	-	L5	W1	473
GND	-	-	GND*	GND*	-
I/O	6	-	-	V2	476
I/O	6	-	-	U4	482
I/O, V_{REF}	6	P47	N2	Y1	485
GND	-	-	GND*	GND*	-
I/O	6	-	M4	W2	488
I/O	6	-	-	V3	491
I/O	6	-	-	Y4	494
I/O	6	P48	R1	Y2	500
I/O	6	P49	M3	W3	503
M1	-	P50	P2	U5	506
GND	-	P51	GND*	GND*	-
M0	-	P52	N3	AB2	507
V_{CCO}	6	P53	V_{CCO} Bank 6*	V_{CCO} Bank 6*	-
V_{CCO}	5	P53	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
M2	-	P54	R3	Y4	508
I/O	5	-	-	W5	518
I/O	5	-	-	AB3	521
I/O	5	-	N5	V7	524
GND	-	-	GND*	GND*	-
I/O, V_{REF}	5	P57	T2	Y6	527
I/O	5	-	-	AA4	530
I/O	5	-	-	AB4	536
I/O	5	-	P5	W6	539
I/O	5	P58	T3	Y7	542
GND	-	-	GND*	GND*	-

XC2S200 Device Pinouts *(Continued)*

XC2S200 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
V_{CCO}	5	-	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
I/O, V_{REF}	5	P59	T4	AA5	545
I/O	5	P60	M6	AB5	548
I/O	5	-	-	V8	551
I/O	5	-	-	AA6	554
I/O	5	-	T5	AB6	557
GND	-	-	GND*	GND*	-
I/O	5	P61	N6	AA7	560
I/O	5	-	-	W7	563
I/O, V_{REF}	5	P62	R5	W8	569
I/O	5	P63	P6	Y8	572
GND	-	P64	GND*	GND*	-
V_{CCO}	5	P65	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
V_{CCINT}	-	P66	V_{CCINT}*	V_{CCINT}*	-
I/O	5	P67	R6	AA8	575
I/O	5	P68	M7	V9	578
I/O	5	-	-	AB8	581
I/O	5	-	-	W9	584
I/O	5	-	-	AB9	587
GND	-	-	GND*	GND*	-
I/O	5	P69	N7	Y9	590
I/O	5	-	-	V10	593
I/O	5	-	-	AA9	596
I/O	5	P70	T6	W10	599
I/O	5	P71	P7	AB10	602
GND	-	P72	GND*	GND*	-
V_{CCO}	5	-	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
I/O, V_{REF}	5	P73	P8	Y10	605
I/O	5	P74	R7	V11	608
I/O	5	-	-	AA10	614
I/O	5	-	T7	W11	617
I/O	5	P75	T8	AB11	620
I/O	5	-	-	U11	623
V_{CCINT}	-	P76	V_{CCINT}*	V_{CCINT}*	-
I, GCK1	5	P77	R8	Y11	635
V_{CCO}	5	P78	V_{CCO} Bank 5*	V_{CCO} Bank 5*	-
V_{CCO}	4	P78	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
GND	-	P79	GND*	GND*	-

XC2S200 Device Pinouts *(Continued)*

XC2S200 Pad Name					Bndry
Function	Bank	PQ208	FG256	FG456	Scan
I, GCK0	4	P80	N8	W12	636
I/O	4	P81	N9	U12	640
I/O	4	-	-	V12	646
I/O	4	P82	R9	Y12	649
I/O	4	-	N10	AA12	652
I/O	4	-	-	W13	655
I/O	4	P83	T9	AB13	661
I/O, V_{REF}	4	P84	P9	AA13	664
V_{CCO}	4	-	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
GND	-	P85	GND*	GND*	-
I/O	4	P86	M10	Y13	667
I/O	4	P87	R10	V13	670
I/O	4	-	-	AB14	673
I/O	4	-	-	W14	676
I/O	4	P88	P10	AA14	679
GND	-	-	GND*	GND*	-
I/O	4	-	-	V14	682
I/O	4	-	-	Y14	685
I/O	4	-	-	W15	688
I/O	4	P89	T10	AB15	691
I/O	4	P90	R11	AA15	694
V_{CCINT}	-	P91	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	4	P92	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
GND	-	P93	GND*	GND*	-
I/O	4	P94	M11	Y15	697
I/O, V_{REF}	4	P95	T11	AB16	700
I/O	4	-	-	AB17	706
I/O	4	P96	N11	V15	709
GND	-	-	GND*	GND*	-
I/O	4	-	R12	Y16	712
I/O	4	-	-	AA17	715
I/O	4	-	-	W16	718
I/O	4	P97	P11	AB18	721
I/O, V_{REF}	4	P98	T12	AB19	724
V_{CCO}	4	-	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
GND	-	-	GND*	GND*	-
I/O	4	P99	T13	Y17	727
I/O	4	-	N12	V16	730
I/O	4	-	-	AA18	733

XC2S200 Device Pinouts *(Continued)*

XC2S200 Pad Name					Bndry
Function	Bank	PQ208	FG256	FG456	Scan
I/O	4	-	-	W17	739
I/O, V_{REF}	4	P100	R13	AB20	742
GND	-	-	GND*	GND*	-
I/O	4	-	P12	AA19	745
I/O	4	-	-	V17	748
I/O	4	-	-	Y18	751
I/O	4	P101	P13	AA20	757
I/O	4	P102	T14	W18	760
GND	-	P103	GND*	GND*	-
DONE	3	P104	R14	Y19	763
V_{CCO}	4	P105	V_{CCO} Bank 4*	V_{CCO} Bank 4*	-
V_{CCO}	3	P105	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
PROGRAM	-	P106	P15	W20	766
I/O (INIT)	3	P107	N15	V19	767
I/O (D7)	3	P108	N14	Y21	770
I/O	3	-	-	V20	776
I/O	3	-	-	AA22	779
I/O	3	-	T15	W21	782
GND	-	-	GND*	GND*	-
I/O, V_{REF}	3	P109	M13	U20	785
I/O	3	-	-	U19	788
I/O	3	-	-	V21	794
GND	-	-	GND*	GND*	-
I/O	3	-	R16	T18	797
I/O	3	P110	M14	W22	800
GND	-	-	GND*	GND*	-
V_{CCO}	3	-	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
I/O, V_{REF}	3	P111	L14	U21	803
I/O	3	P112	M15	T20	806
I/O	3	-	-	T19	809
I/O	3	-	-	V22	812
I/O	3	-	L12	T21	815
GND	-	-	GND*	GND*	-
I/O	3	P113	P16	R18	818
I/O	3	-	-	U22	821
I/O, V_{REF}	3	P114	L13	R19	827
I/O (D6)	3	P115	N16	T22	830
GND	-	P116	GND*	GND*	-

XC2S200 Device Pinouts *(Continued)*

XC2S200 Pad Name					Bndry
Function	Bank	PQ208	FG256	FG456	Scan
V_{CCO}	3	P117	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
V_{CCINT}	-	P118	V_{CCINT}*	V_{CCINT}*	-
I/O (D5)	3	P119	M16	R21	833
I/O	3	P120	K14	P18	836
I/O	3	-	-	R22	839
I/O	3	-	-	P19	842
I/O	3	-	L16	P20	845
GND	-	-	GND*	GND*	-
I/O	3	P121	K13	P21	848
I/O	3	-	-	N19	851
I/O	3	-	-	P22	854
I/O	3	P122	L15	N18	857
I/O	3	P123	K12	N20	860
GND	-	P124	GND*	GND*	-
V_{CCO}	3	-	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
I/O, V_{REF}	3	P125	K16	N21	863
I/O (D4)	3	P126	J16	N22	866
I/O	3	-	-	M17	872
I/O	3	-	J14	M19	875
I/O	3	P127	K15	M20	878
I/O	3	-	-	M18	881
V_{CCINT}	-	P128	V_{CCINT}*	V_{CCINT}*	-
I/O, TRDY(1)	3	P129	J15	M22	890
V_{CCO}	3	P130	V_{CCO} Bank 3*	V_{CCO} Bank 3*	-
V_{CCO}	2	P130	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
GND	-	P131	GND*	GND*	-
I/O, IRDY(1)	2	P132	H16	L20	893
I/O	2	P133	H14	L17	896
I/O	2	-	-	L18	902
I/O	2	P134	H15	L21	905
I/O	2	-	J13	L22	908
I/O	2	-	-	K19	911
I/O (D3)	2	P135	G16	K20	917
I/O, V_{REF}	2	P136	H13	K21	920
V_{CCO}	2	-	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
GND	-	P137	GND*	GND*	-
I/O	2	P138	G14	K22	923
I/O	2	P139	G15	J21	926

XC2S200 Device Pinouts *(Continued)*

XC2S200 Pad Name					Bndry
Function	Bank	PQ208	FG256	FG456	Scan
I/O	2	-	-	K18	929
I/O	2	-	-	J20	932
I/O	2	P140	G12	J18	935
GND	-	-	GND*	GND*	-
I/O	2	-	F16	J22	938
I/O	2	-	-	J19	941
I/O	2	-	-	H21	944
I/O	2	P141	G13	H19	947
I/O (D2)	2	P142	F15	H20	950
V_{CCINT}	-	P143	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	2	P144	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
GND	-	P145	GND*	GND*	-
I/O (D1)	2	P146	E16	H22	953
I/O, V_{REF}	2	P147	F14	H18	956
I/O	2	-	-	G21	962
I/O	2	P148	D16	G18	965
GND	-	-	GND*	GND*	-
I/O	2	-	F12	G20	968
I/O	2	-	-	G19	971
I/O	2	-	-	F22	974
I/O	2	P149	E15	F19	977
I/O, V_{REF}	2	P150	F13	F21	980
V_{CCO}	2	-	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-
GND	-	-	GND*	GND*	-
I/O	2	P151	E14	F20	983
I/O	2	-	C16	F18	986
GND	-	-	GND*	GND*	-
I/O	2	-	-	E22	989
I/O	2	-	-	E21	995
I/O, V_{REF}	2	P152	E13	D22	998
GND	-	-	GND*	GND*	-
I/O	2	-	B16	E20	1001
I/O	2	-	-	D21	1004
I/O	2	-	-	C22	1007
I/O (DIN, D0)	2	P153	D14	D20	1013
I/O (DOUT, BUSY)	2	P154	C15	C21	1016
CCLK	2	P155	D15	B22	1019
V_{CCO}	2	P156	V_{CCO} Bank 2*	V_{CCO} Bank 2*	-

XC2S200 Device Pinouts *(Continued)*

XC2S200 Pad Name		PQ208	FG256	FG456	Bndry Scan
Function	Bank				
V_{CCO}	1	P156	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
TDO	2	P157	B14	A21	-
GND	-	P158	GND*	GND*	-
TDI	-	P159	A15	B20	-
I/O ($\overline{CS}$)	1	P160	B13	C19	0
I/O ($\overline{WRITE}$)	1	P161	C13	A20	3
I/O	1	-	-	B19	9
I/O	1	-	-	C18	12
I/O	1	-	C12	D17	15
GND	-	-	GND*	GND*	-
I/O, V_{REF}	1	P162	A14	A19	18
I/O	1	-	-	B18	21
I/O	1	-	-	E16	27
I/O	1	-	D12	C17	30
I/O	1	P163	B12	D16	33
GND	-	-	GND*	GND*	-
V_{CCO}	1	-	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
I/O, V_{REF}	1	P164	C11	A18	36
I/O	1	P165	A13	B17	39
I/O	1	-	-	E15	42
I/O	1	-	-	A17	45
I/O	1	-	D11	D15	48
GND	-	-	GND*	GND*	-
I/O	1	P166	A12	C16	51
I/O	1	-	-	D14	54
I/O, V_{REF}	1	P167	E11	E14	60
I/O	1	P168	B11	A16	63
GND	-	P169	GND*	GND*	-
V_{CCO}	1	P170	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
V_{CCINT}	-	P171	V_{CCINT}*	V_{CCINT}*	-
I/O	1	P172	A11	C15	66
I/O	1	P173	C10	B15	69
I/O	1	-	-	E13	72
I/O	1	-	-	A15	75
I/O	1	-	-	F12	78
GND	-	-	GND*	GND*	-
I/O	1	P174	B10	C14	81
I/O	1	-	-	B14	84
I/O	1	-	-	A14	87

XC2S200 Device Pinouts *(Continued)*

XC2S200 Pad Name		PQ208	FG256	FG456	Bndry Scan
Function	Bank				
I/O	1	P175	D10	D13	90
I/O	1	P176	A10	C13	93
GND	-	P177	GND*	GND*	-
V_{CCO}	1	-	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
I/O, V_{REF}	1	P178	B9	B13	96
I/O	1	P179	E10	E12	99
I/O	1	-	-	A13	105
I/O	1	-	A9	B12	108
I/O	1	P180	D9	D12	111
I/O	1	-	-	C12	114
I/O	1	P181	A8	D11	120
I, GCK2	1	P182	C9	A11	126
GND	-	P183	GND*	GND*	-
V_{CCO}	1	P184	V_{CCO} Bank 1*	V_{CCO} Bank 1*	-
V_{CCO}	0	P184	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
I, GCK3	0	P185	B8	C11	127
V_{CCINT}	-	P186	V_{CCINT}*	V_{CCINT}*	-
I/O	0	-	-	E11	137
I/O	0	P187	A7	A10	140
I/O	0	-	D8	B10	143
I/O	0	-	-	F11	146
I/O	0	P188	A6	C10	152
I/O, V_{REF}	0	P189	B7	A9	155
V_{CCO}	0	-	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
GND	-	P190	GND*	GND*	-
I/O	0	P191	C8	B9	158
I/O	0	P192	D7	E10	161
I/O	0	-	-	C9	164
I/O	0	-	-	D10	167
I/O	0	P193	E7	A8	170
GND	-	-	GND*	GND*	-
I/O	0	-	-	D9	173
I/O	0	-	-	B8	176
I/O	0	-	-	C8	179
I/O	0	P194	C7	E9	182
I/O	0	P195	B6	A7	185
V_{CCINT}	-	P196	V_{CCINT}*	V_{CCINT}*	-
V_{CCO}	0	P197	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-

XC2S200 Device Pinouts *(Continued)*

XC2S200 Pad Name					
Function	Bank	PQ208	FG256	FG456	Bndry Scan
GND	-	P198	GND*	GND*	-
VO	0	P199	A5	B7	188
VO, V_{REF}	0	P200	C6	E8	191
VO	0	-	-	D8	197
VO	0	P201	B5	C7	200
GND	-	-	GND*	GND*	-
VO	0	-	D6	D7	203
VO	0	-	-	B6	206
VO	0	-	-	A5	209
VO	0	P202	A4	D6	212
VO, V_{REF}	0	P203	B4	C6	215
V_{CCO}	0	-	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
GND	-	-	GND*	GND*	-
VO	0	P204	E6	B5	218
VO	0	-	D5	E7	221
VO	0	-	-	A4	224
VO	0	-	-	E6	230
VO, V_{REF}	0	P205	A3	B4	233
GND	-	-	GND*	GND*	-
VO	0	-	C5	A3	236
VO	0	-	-	B3	239
VO	0	-	-	D5	242
VO	0	P206	B3	C5	248
TCK	-	P207	C4	C4	-
V_{CCO}	0	P208	V_{CCO} Bank 0*	V_{CCO} Bank 0*	-
V_{CCO}	7	P208	V_{CCO} Bank 7*	V_{CCO} Bank 7*	-

04/18/01

Notes:

1. IRDY and TRDY can only be accessed when using Xilinx PCI cores.
2. Pads labelled GND*, V_{CCINT}*, V_{CCO} Bank 0*, V_{CCO} Bank 1*, V_{CCO} Bank 2*, V_{CCO} Bank 3*, V_{CCO} Bank 4*, V_{CCO} Bank 5*, V_{CCO} Bank 6*, V_{CCO} Bank 7* are internally bonded to independent ground or power planes within the package.

Additional XC2S200 Package Pins

PQ208

Not Connected Pins					
P55	P56	-	-	-	-

11/02/00

FG256

V_{CCINT} Pins					
C3	C14	D4	D13	E5	E12
M5	M12	N4	N13	P3	P14
V_{CCO} Bank 0 Pins					
E8	F8	-	-	-	-
V_{CCO} Bank 1 Pins					
E9	F9	-	-	-	-
V_{CCO} Bank 2 Pins					
H11	H12	-	-	-	-
V_{CCO} Bank 3 Pins					
J11	J12	-	-	-	-
V_{CCO} Bank 4 Pins					
L9	M9	-	-	-	-
V_{CCO} Bank 5 Pins					
L8	M8	-	-	-	-
V_{CCO} Bank 6 Pins					
J5	J6	-	-	-	-
V_{CCO} Bank 7 Pins					
H5	H6	-	-	-	-
GND Pins					
A1	A16	B2	B15	F6	F7
F10	F11	G6	G7	G8	G9
G10	G11	H7	H8	H9	H10
J7	J8	J9	J10	K6	K7
K8	K9	K10	K11	L6	L7
L10	L11	R2	R15	T1	T16
Not Connected Pins					
P4	R4	-	-	-	-

11/02/00

dditional XC2S200 Package Pins *(Continued)*

G456

V_{CCINT} Pins					
E5	E18	F6	F17	G7	G8
G9	G14	G15	G16	H7	H16
J7	J16	P7	P16	R7	R16
T7	T8	T9	T14	T15	T16
U6	U17	V5	V18	-	-
V_{CCO} BANK 0 Pins					
F7	F8	F9	F10	G10	G11
V_{CCO} Bank 1 Pins					
F13	F14	F15	F16	G12	G13
V_{CCO} Bank 2 Pins					
G17	H17	J17	K16	K17	L16
V_{CCO} Bank 3 Pins					
M16	N16	N17	P17	R17	T17
V_{CCO} Bank 4 Pins					
T12	T13	U13	U14	U15	U16
V_{CCO} Bank 5 Pins					
T10	T11	U7	U8	U9	U10
V_{CCO} Bank 6 Pins					
M7	N6	N7	P6	R6	T6
V_{CCO} Bank 7 Pins					

Additional XC2S200 Package Pins *(Continued)*

G6	H6	J6	K6	K7	L7
GND Pins					
A1	A22	B2	B21	C3	C20
J9	J10	J11	J12	J13	J14
K9	K10	K11	K12	K13	K14
L9	L10	L11	L12	L13	L14
M9	M10	M11	M12	M13	M14
N9	N10	N11	N12	N13	N14
P9	P10	P11	P12	P13	P14
Y3	Y20	AA2	AA21	AB1	AB22
Not Connected Pins					
A2	A6	A12	B11	B16	C2
D1	D4	D18	D19	E17	E19
G2	G22	L2	L19	M2	M21
R3	R20	U3	U18	V6	W4
W19	Y5	Y22	AA1	AA3	AA11
AA16	AB7	AB12	AB21	-	-

11/02/00

Revision History

Version No.	Date	Description
2.0	09/18/00	Sectioned the Spartan-II Family data sheet into four modules. Corrected all known errors in the pinout tables.
2.1	10/04/00	Added notes requiring PWDN to be tied to V_{CCINT} when unused.
2.2	11/02/00	Removed the Power Down feature.
2.3	03/05/01	Added notes on pinout tables for IRDY and TRDY.
2.4	04/30/01	Reinstated XC2S50 V_{CCO} Bank 7, GND, and "not connected" pins missing in version 2.3.
2.5	09/03/03	Added caution about Not Connected Pins to XC2S30 pinout tables on page 6.

The Spartan-II Family Data Sheet

DS001-1, ***Spartan-II 2.5V FPGA Family: Introduction and Ordering Information*** (Module 1)

DS001-2, ***Spartan-II 2.5V FPGA Family: Functional Description*** (Module 2)

DS001-3, ***Spartan-II 2.5V FPGA Family: DC and Switching Characteristics*** (Module 3)

DS001-4, ***Spartan-II 2.5V FPGA Family: Pinout Tables*** (Module 4)

ภาคผนวก ข.โปรแกรมในส่วนของฮาร์ดแวร์

โปรแกรมในส่วนของการสุ่มและคงค่าข้อมูล

```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
-- synopsys translate_off
library UNISIM;
use UNISIM.Vcomponents.ALL;
-- synopsys translate_on

entity IFD_MXILINX_samp is
  port ( C : in    std_logic;
         D : in    std_logic;
         Q : out   std_logic);
end IFD_MXILINX_samp;

architecture BEHAVIORAL of IFD_MXILINX_samp is
  attribute INIT     : STRING ;
  attribute BOX_TYPE : STRING ;
  attribute IOB      : STRING ;
  signal D_IN   : std_logic;
  signal XLXN_1 : std_logic;
  signal XLXN_2 : std_logic;
  component FDCE
    -- synopsys translate_off
    generic( INIT : bit :=  '0');
    -- synopsys translate_on
    port ( C   : in    std_logic;
           CE  : in    std_logic;
           CLR : in    std_logic;
           D   : in    std_logic;
           Q   : out   std_logic);
  end component;
```

```
attribute INIT of FDCE : COMPONENT is "0";
attribute BOX_TYPE of FDCE : COMPONENT is "BLACK_BOX";

component IBUF
  port ( I : in    std_logic;
         O : out   std_logic);
end component;
attribute BOX_TYPE of IBUF : COMPONENT is "BLACK_BOX";

component VCC
  port ( P : out   std_logic);
end component;
attribute BOX_TYPE of VCC : COMPONENT is "BLACK_BOX";

component GND
  port ( G : out   std_logic);
end component;
attribute BOX_TYPE of GND : COMPONENT is "BLACK_BOX";

attribute IOB of I_36_15 : LABEL is "TRUE";
begin
I_36_15 : FDCE
  port map (C=>C, CE=>XLXN_2, CLR=>XLXN_1, D=>D_IN, Q=>Q);
I_36_24 : IBUF
  port map (I=>D, O=>D_IN)
I_36_26 : VCC
  port map (P=>XLXN_2)
I_36_29 : GND
  port map (G=>XLXN_1);

end BEHAVIORAL;
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
```

```
-- synopsys translate_off
library UNISIM;
use UNISIM.Vcomponents.ALL;
-- synopsys translate_on

entity IFD16_MXILINX_samp is
  port ( C : in    std_logic;
         D : in    std_logic_vector (15 downto 0);
         Q : out   std_logic_vector (15 downto 0));
end IFD16_MXILINX_samp;

architecture BEHAVIORAL of IFD16_MXILINX_samp is
  attribute HU_SET     : STRING ;
  component IFD_MXILINX_samp
    port ( C : in    std_logic;
           D : in    std_logic;
           Q : out   std_logic);
  end component;

  attribute HU_SET of I_Q0 : LABEL is "I_Q0_0";
  attribute HU_SET of I_Q1 : LABEL is "I_Q1_1";
  attribute HU_SET of I_Q2 : LABEL is "I_Q2_2";
  attribute HU_SET of I_Q3 : LABEL is "I_Q3_3";
  attribute HU_SET of I_Q4 : LABEL is "I_Q4_4";
  attribute HU_SET of I_Q5 : LABEL is "I_Q5_5";
  attribute HU_SET of I_Q6 : LABEL is "I_Q6_6";
  attribute HU_SET of I_Q7 : LABEL is "I_Q7_7";
  attribute HU_SET of I_Q8 : LABEL is "I_Q8_8";
  attribute HU_SET of I_Q9 : LABEL is "I_Q9_9";
  attribute HU_SET of I_Q10 : LABEL is "I_Q10_10";
  attribute HU_SET of I_Q11 : LABEL is "I_Q11_11";
  attribute HU_SET of I_Q12 : LABEL is "I_Q12_12";
  attribute HU_SET of I_Q13 : LABEL is "I_Q13_13";
  attribute HU_SET of I_Q14 : LABEL is "I_Q14_14";
```

```
  attribute HU_SET of I_Q15 : LABEL is "I_Q15_15";
begin
  I_Q0 : IFD_MXILINX_samp
    port map (C=>C, D=>D(0), Q=>Q(0));
  I_Q1 : IFD_MXILINX_samp
    port map (C=>C, D=>D(1), Q=>Q(1));
  I_Q2 : IFD_MXILINX_samp
    port map (C=>C, D=>D(2), Q=>Q(2));
  I_Q3 : IFD_MXILINX_samp
    port map (C=>C, D=>D(3), Q=>Q(3));
  I_Q4 : IFD_MXILINX_samp
    port map (C=>C, D=>D(4), Q=>Q(4));
  I_Q5 : IFD_MXILINX_samp
    port map (C=>C, D=>D(5), Q=>Q(5));
  I_Q6 : IFD_MXILINX_samp
    port map (C=>C, D=>D(6), Q=>Q(6));
  I_Q7 : IFD_MXILINX_samp
    port map (C=>C, D=>D(7), Q=>Q(7));
  I_Q8 : IFD_MXILINX_samp
    port map (C=>C, D=>D(8), Q=>Q(8));
  I_Q9 : IFD_MXILINX_samp
    port map (C=>C, D=>D(9), Q=>Q(9));
  I_Q10 : IFD_MXILINX_samp
    port map (C=>C, D=>D(10), Q=>Q(10));
  I_Q11 : IFD_MXILINX_samp
    port map (C=>C, D=>D(11), Q=>Q(11));
  I_Q12 : IFD_MXILINX_samp
    port map (C=>C, D=>D(12), Q=>Q(12));
  I_Q13 : IFD_MXILINX_samp
    port map (C=>C, D=>D(13), Q=>Q(13));
  I_Q14 : IFD_MXILINX_samp
    port map (C=>C, D=>D(14), Q=>Q(14));
  I_Q15 : IFD_MXILINX_samp
    port map (C=>C, D=>D(15), Q=>Q(15));
```

```
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
-- synopsys translate_off
library UNISIM;
use UNISIM.Vcomponents.ALL;
-- synopsys translate_on

entity IBUF16_MXILINX_samp is
  port ( I : in    std_logic_vector (15 downto 0);
         O : out   std_logic_vector (15 downto 0));
end IBUF16_MXILINX_samp;


architecture BEHAVIORAL of IBUF16_MXILINX_samp is
  attribute BOX_TYPE   : STRING ;
  component IBUF
    port ( I : in    std_logic;
           O : out   std_logic);
  end component;
  attribute BOX_TYPE of IBUF : COMPONENT is "BLACK_BOX";

begin
  I_36_30 : IBUF
    port map (I=>I(8), O=>O(8));
  I_36_31 : IBUF
    port map (I=>I(9), O=>O(9));
  I_36_32 : IBUF
    port map (I=>I(10), O=>O(10));
  I_36_33 : IBUF
    port map (I=>I(11), O=>O(11));
  I_36_34 : IBUF
```

```
    port map (I=>I(15), O=>O(15));
  I_36_35 : IBUF
    port map (I=>I(14), O=>O(14));
  I_36_36 : IBUF
    port map (I=>I(13), O=>O(13));
  I_36_37 : IBUF
    port map (I=>I(12), O=>O(12));
  I_36_38 : IBUF
    port map (I=>I(4), O=>O(4));
  I_36_39 : IBUF
    port map (I=>I(5), O=>O(5));
  I_36_40 : IBUF
    port map (I=>I(6), O=>O(6));
  I_36_41 : IBUF
    port map (I=>I(7), O=>O(7));
  I_36_42 : IBUF
    port map (I=>I(3), O=>O(3));
  I_36_43 : IBUF
    port map (I=>I(2), O=>O(2));
  I_36_44 : IBUF
    port map (I=>I(1), O=>O(1))
  I_36_45 : IBUF
    port map (I=>I(0), O=>O(0));

end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
-- synopsys translate_off
library UNISIM;
use UNISIM.Vcomponents.ALL;
-- synopsys translate_on
```

```
entity samp is
  port ( Data_in  : in    std_logic_vector (15 downto 0);
         Samp     : in    std_logic;
         Data_out : out   std_logic_vector (15 downto 0));
end samp;

architecture BEHAVIORAL of samp is
  attribute HU_SET     : STRING ;
  signal XLXN_2   : std_logic_vector (15 downto 0);
  component IBUF16_MXILINX_samp
    port ( I : in    std_logic_vector (15 downto 0);
           O : out   std_logic_vector (15 downto 0));
  end component;

  component IFD16_MXILINX_samp
    port ( C : in    std_logic;
           D : in    std_logic_vector (15 downto 0);
           Q : out   std_logic_vector (15 downto 0));
  end component;

  attribute HU_SET of XLXI_1 : LABEL is "XLXI_1_16";
  attribute HU_SET of XLXI_2 : LABEL is "XLXI_2_17";
begin
  XLXI_1 : IBUF16_MXILINX_samp
    port map (I(15 downto 0)=>Data_in(15 downto 0), O(15 downto 0)=>XLXN_2(15
        downto 0));

  XLXI_2 : IFD16_MXILINX_samp
    port map (C=>Samp, D(15 downto 0)=>XLXN_2(15 downto 0), Q(15 downto
        0)=>Data_out(15 downto 0));

end BEHAVIORAL;
```

โปรแกรมในส่วนของการสร้างสัญญาณนาฬิกา

```
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
-- synopsys translate_off
library UNISIM;
use UNISIM.Vcomponents.ALL;
-- synopsys translate_on

entity M2_1_MXILINX_clk_gen is
  port ( D0 : in    std_logic;
         D1 : in    std_logic;
         S0 : in    std_logic;
         O  : out   std_logic);
end M2_1_MXILINX_clk_gen;

architecture BEHAVIORAL of M2_1_MXILINX_clk_gen is
  attribute BOX_TYPE   : STRING ;
  signal M0 : std_logic;
  signal M1 : std_logic;
  component AND2B1
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           O  : out   std_logic);
  end component;
  attribute BOX_TYPE of AND2B1 : COMPONENT is "BLACK_BOX";

  component OR2
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           O  : out   std_logic);
  end component;
```

```
  attribute BOX_TYPE of OR2 : COMPONENT is "BLACK_BOX";

  component AND2
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           O  : out   std_logic);
  end component;
  attribute BOX_TYPE of AND2 : COMPONENT is "BLACK_BOX";

begin
  I_36_7 : AND2B1
    port map (I0=>S0, I1=>D0, O=>M0);
  I_36_8 : OR2
    port map (I0=>M1, I1=>M0, O=>O);
  I_36_9 : AND2
    port map (I0=>D1, I1=>S0, O=>M1);

end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
-- synopsys translate_off
library UNISIM;
use UNISIM.Vcomponents.ALL;
-- synopsys translate_on

entity FTCE_MXILINX_clk_gen is
  port ( C   : in    std_logic;
         CE  : in    std_logic;
         CLR : in    std_logic;
         T   : in    std_logic;
         Q   : out   std_logic);
end FTCE_MXILINX_clk_gen;
```

```
architecture BEHAVIORAL of FTCE_MXILINX_clk_gen is
  attribute BOX_TYPE   : STRING ;
  attribute INIT       : STRING ;
  attribute RLOC       : STRING ;
  signal TQ     : std_logic;
  signal Q_DUMMY : std_logic;
  component XOR2
    port ( I0 : in    std_logic;
           I1 : in    std_logic;
           O  : out   std_logic);
  end component;
  attribute BOX_TYPE of XOR2 : COMPONENT is "BLACK_BOX";

  component FDCE
    -- synopsys translate_off
    generic( INIT : bit := '0');
    -- synopsys translate_on
    port ( C   : in    std_logic;
           CE  : in    std_logic;
           CLR : in    std_logic;
           D   : in    std_logic;
           Q   : out   std_logic);
  end component;
  attribute INIT of FDCE : COMPONENT is "0";
  attribute BOX_TYPE of FDCE : COMPONENT is "BLACK_BOX";

  attribute RLOC of I_36_35 : LABEL is "R0C0.S0";
begin
  Q <= Q_DUMMY;
  I_36_32 : XOR2
    port map (I0=>T, I1=>Q_DUMMY, O=>TQ);

  I_36_35 : FDCE
```

```
    port map (C=>C, CE=>CE, CLR=>CLR, D=>TQ, Q=>Q_DUMMY);
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
-- synopsys translate_off
library UNISIM;
use UNISIM.Vcomponents.ALL;
-- synopsys translate_on

entity CB4CE_MXILINX_clk_gen is
  port ( C   : in   std_logic;
       CE  : in   std_logic;
       CLR : in   std_logic;
       CEO : out  std_logic;
       Q0  : out  std_logic;
       Q1  : out  std_logic;
       Q2  : out  std_logic;
       Q3  : out  std_logic;
       TC  : out  std_logic);
end CB4CE_MXILINX_clk_gen;

architecture BEHAVIORAL of CB4CE_MXILINX_clk_gen is
  attribute HU_SET    : STRING ;
  attribute BOX_TYPE  : STRING ;
  signal T2      : std_logic;
  signal T3      : std_logic;
  signal XLXN_1  : std_logic;
  signal Q0_DUMMY : std_logic;
  signal Q1_DUMMY : std_logic;
  signal Q2_DUMMY : std_logic;
  signal Q3_DUMMY : std_logic;
  signal TC_DUMMY : std_logic;
```

```
component FTCE_MXILINX_clk_gen
  port ( C   : in   std_logic;
         CE  : in   std_logic;
         CLR : in   std_logic;
         T   : in   std_logic;
         Q   : out  std_logic);
end component;

component AND4
  port ( I0 : in   std_logic;
         I1 : in   std_logic;
         I2 : in   std_logic;
         I3 : in   std_logic;
         O  : out  std_logic);
end component;
attribute BOX_TYPE of AND4 : COMPONENT is "BLACK_BOX";

component AND3
  port ( I0 : in   std_logic;
         I1 : in   std_logic;
         I2 : in   std_logic;
         O  : out  std_logic);
end component;
attribute BOX_TYPE of AND3 : COMPONENT is "BLACK_BOX";

component AND2
  port ( I0 : in   std_logic;
         I1 : in   std_logic;
         O  : out  std_logic);
end component;
attribute BOX_TYPE of AND2 : COMPONENT is "BLACK_BOX";

component VCC
  port ( P : out  std_logic);
```

```
  end component;
  attribute BOX_TYPE of VCC : COMPONENT is "BLACK_BOX";

  attribute HU_SET of I_Q0 : LABEL is "I_Q0_0";
  attribute HU_SET of I_Q1 : LABEL is "I_Q1_1";
  attribute HU_SET of I_Q2 : LABEL is "I_Q2_2";
  attribute HU_SET of I_Q3 : LABEL is "I_Q3_3";
begin
  Q0 <= Q0_DUMMY;
  Q1 <= Q1_DUMMY;
  Q2 <= Q2_DUMMY;
  Q3 <= Q3_DUMMY;
  TC <= TC_DUMMY;
  I_Q0 : FTCE_MXILINX_clk_gen
    port map (C=>C, CE=>CE, CLR=>CLR, T=>XLXN_1, Q=>Q0_DUMMY);
  I_Q1 : FTCE_MXILINX_clk_gen
    port map (C=>C, CE=>CE, CLR=>CLR, T=>Q0_DUMMY, Q=>Q1_DUMMY);
  I_Q2 : FTCE_MXILINX_clk_gen
    port map (C=>C, CE=>CE, CLR=>CLR, T=>T2, Q=>Q2_DUMMY);
  I_Q3 : FTCE_MXILINX_clk_gen
    port map (C=>C, CE=>CE, CLR=>CLR, T=>T3, Q=>Q3_DUMMY);
  I_36_31 : AND4
    port map (I0=>Q3_DUMMY, I1=>Q2_DUMMY, I2=>Q1_DUMMY, I3=>Q0_DUMMY,
        O=>TC_DUMMY);
  I_36_32 : AND3
    port map (I0=>Q2_DUMMY, I1=>Q1_DUMMY, I2=>Q0_DUMMY, O=>T3);

  I_36_33 : AND2
    port map (I0=>Q1_DUMMY, I1=>Q0_DUMMY, O=>T2);
  I_36_58 : VCC
    port map (P=>XLXN_1);
  I_36_67 : AND2
    port map (I0=>CE, I1=>TC_DUMMY, O=>CEO);
```

```
end BEHAVIORAL;

library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;
-- synopsys translate_off
library UNISIM;
use UNISIM.Vcomponents.ALL;
-- synopsys translate_on

entity clk_gen is
  port ( Clk      : in   std_logic;
         Clk_clr  : in   std_logic;
         Clk_En   : in   std_logic;
         Clk_En_Mux : in  std_logic;
         Sump     : out  std_logic);
end clk_gen;

architecture BEHAVIORAL of clk_gen is
  attribute HU_SET    : STRING ;
  attribute BOX_TYPE  : STRING ;
  signal XLXN_1    : std_logic;
  signal XLXN_2    : std_logic;
  signal XLXN_3    : std_logic;
  signal XLXN_4    : std_logic;
  signal XLXN_5    : std_logic;
  signal XLXN_7    : std_logic;
  signal XLXN_8    : std_logic;
  signal XLXN_9    : std_logic;
  signal XLXN_10   : std_logic;
  signal XLXN_11   : std_logic;
  signal XLXN_22   : std_logic;
  signal XLXN_23   : std_logic;
  component CB4CE_MXILINX_clk_gen
```

```
 port ( C   : in   std_logic;
        CE  : in   std_logic;
        CLR : in   std_logic;
        CEO : out  std_logic;
        Q0  : out  std_logic;
        Q1  : out  std_logic;
        Q2  : out  std_logic;
        Q3  : out  std_logic;
        TC  : out  std_logic);
end component;

component AND4
  port ( I0 : in   std_logic;
         I1 : in   std_logic;
         I2 : in   std_logic;
         I3 : in   std_logic;
         O  : out  std_logic);
end component;
attribute BOX_TYPE of AND4 : COMPONENT is "BLACK_BOX";

component INV
  port ( I : in   std_logic;
         O : out  std_logic);
end component;

attribute BOX_TYPE of INV : COMPONENT is "BLACK_BOX";

component M2_1_MXILINX_clk_gen
  port ( D0 : in   std_logic;
         D1 : in   std_logic;
         S0 : in   std_logic;
         O  : out  std_logic);
end component;
```

```
  attribute HU_SET of XLXI_2 : LABEL is "XLXI_2_4";
  attribute HU_SET of XLXI_6 : LABEL is "XLXI_6_5";
  attribute HU_SET of XLXI_10 : LABEL is "XLXI_10_6";
begin
  XLXI_2 : CB4CE_MXILINX_clk_gen
    port map (C=>Clk, CE=>Clk_En, CLR=>Clk_clr, CEO=>open, Q0=>XLXN_3,
        Q1=>XLXN_1, Q2=>XLXN_4, Q3=>XLXN_5, TC=>open);
  XLXI_3 : AND4
    port map (I0=>XLXN_5, I1=>XLXN_4, I2=>XLXN_2, I3=>XLXN_3, O=>XLXN_23)
  XLXI_4 : INV
    port map (I=>XLXN_1, O=>XLXN_2);
  XLXI_5 : INV
    port map (I=>XLXN_9, O=>XLXN_10);
  XLXI_6 : CB4CE_MXILINX_clk_gen
    port map (C=>Clk, CE=>Clk_En, CLR=>Clk_clr, CEO=>open, Q0=>XLXN_7,
        Q1=>XLXN_8, Q2=>XLXN_9, Q3=>XLXN_11, TC=>open);
  XLXI_7 : AND4
    port map (I0=>XLXN_11, I1=>XLXN_10, I2=>XLXN_8, I3=>XLXN_7, O=>XLXN_22);
  XLXI_10 : M2_1_MXILINX_clk_gen
    port map (D0=>XLXN_23, D1=>XLXN_22, S0=>Clk_En_Mux, O=>Sump);

end BEHAVIORAL;
```

บรรณานุกรม

[1] Muhammad Ail Mazidi , Janice Gillispie Mazidi : THE 80x86 IBM PC AND COMPATIBLE COMPUTERS VOLUME II Seccond Edition , Prentice-Hall International,Inc , 1995

[2] Lewis C. Eggbert : Interfacing to IBM Personal Computer. , Howard W. Same & Co.,Inc , 1983

[3] Logic Databook Volume 2 National Semiconductor , 1984

[4] Interface Bipolar LSI Biporlar Memory Programmable Logic Databook National Semiconductor , 1983

[5] วิชิต ตั้งสุขนิรันดร : การพัฒนาไมโครโปรเซสเซอร์ให้เป็นลอจิกแอนนาไลเซอร์ , 2534

[6] วรเทพ บุญธรรมจิต , วาสนาพงศ์นรากุล : ลอจิกแอนนาไลเซอร์ , 2532

[7] ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง : เอกสารประกอบการทดลอง วิชา 01072001 Advanced Digital System Design

[8] วรพจน์ กรแก้ววัฒนกุล , ชัยวัฒน์ ลิ้มพรวิจิตรวิไล : เรียนรู้และปฏิบัติการไมโครคอนโทรเลอร์ MCS-51 แบบแฟลช, ©Innovative Experiment Co.,Ltd. , 2540

[9] นอ.ชาติชาย ดิษฐกุล รน. : คู่มือ VHDL