ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ การสร้างวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิต-ศาสตร์การกระจาย

The Implementation of 6TH Order Butterworth Digital Filter

ชื่อนักศึกษา

1. นายณรงค์	คงสมเพ็ชร	รหัสประจำตัว	41031310
2. นายธนพล	แก้วคำแจ้ง	รหัสประจำตัว	41031313
3. นายสุรเกียรติ	โสภณพุทธพร	รหัสประจำตัว	41031330

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** อิเล็กทรอนิกส์และคอมพิวเตอร์

อาจารย์ที่ปรึกษา อาจารย์กิติพงค์ มะโน

อาจารย์ที่ปรึกษาร่วม อาจารย์สุระชัย พิมพ์สาลี

คณะกรรมการสอบปริญญานิพนธ์		ลายมือชื่อ
1. อาจารย์กิติพงค์	มะโน	.....
2. อาจารย์สุระชัย	พิมพ์สาลี	.....
3. อาจารย์สุชิน	อาจหาญ	.....
4. อาจารย์อำพล	ทองระอา	.....

วัน/เดือน/ปีที่สอบ วันเสาร์ที่ 13 พฤษภาคม พ.ศ. 2543 เวลา 16.00 น.

สถานที่สอบ ห้อง ค.301 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.วิสุทธิ์ อธิพรธรรม)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 5 เดือน ธ.ค. พ.ศ. 2543

ปริญญานิพนธ์

การสร้างวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิตศาสตร์การกระจาย

THE IMPLEMENTATION OF 6TH ORDER BUTTERWORTH DIGITAL FILTER

นายณรงค์ คงสมเพ็ชร

นายธนพล แก้วคำแจ้ง

นายสุรเกียรติ โสภณพุทธพร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์

เรื่อง การสร้างวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิตศาสตร์การกระจาย

The Implementation of 6^{TH} Order Butterworth Digital Filter

วัตถุประสงค์

1. เพื่อศึกษาทฤษฎีของการกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยการใช้คณิตศาสตร์การกระจาย, ภาษา VHDL และอุปกรณ์ FPGAs
2. เพื่อออกแบบวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6โดยใช้คณิตศาสตร์การกระจาย, ภาษา VHDL และอุปกรณ์ FPGAs
3. เพื่อสร้างวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิตศาสตร์การกระจาย, ภาษา VHDL และอุปกรณ์ FPGAs
4. เพื่อทดลองวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิตศาสตร์การกระจาย, ภาษา VHDL และอุปกรณ์ FPGAs
5. เพื่อนำไปใช้กรองสัญญาณที่ความถี่คัทออฟ 20 kHz

ประโยชน์ที่คาดว่าจะได้รับ

1. มีความเข้าใจทฤษฎีของวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิตศาสตร์การกระจาย, ภาษา VHDL และอุปกรณ์ FPGAs
2. สามารถออกแบบวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิตศาสตร์การกระจาย, ภาษา VHDL และอุปกรณ์ FPGAs ได้
3. สามารถสร้างวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิตศาสตร์การกระจาย, ภาษา VHDL และอุปกรณ์ FPGAs ได้
4. สามารถทดลองวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิตศาสตร์การกระจาย, ภาษา VHDL และอุปกรณ์ FPGAs ได้
5. สามารถนำไปใช้กรองสัญญาณที่ความถี่คัทออฟ 20 kHz ได้

ชื่อหัวข้อ	การสร้างวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิตศาสตร์การกระจาย
นักศึกษา	นายณรงค์ คงสมเพ็ชร
	นายธนพล แก้วคำแจ้ง
	นายสุรเกียรติ โสภณพุทธพร
อาจารย์ที่ปรึกษา	อาจารย์กิติพงค์ มะโน
อาจารย์ที่ปรึกษาร่วม	อาจารย์สุระชัย พิมพ์สาลี
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชา	อิเล็กทรอนิกส์และคอมพิวเตอร์
ปีการศึกษา	2542

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอวิธีการออกแบบ และการสร้างวงจรกรองความถี่เชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 ด้วยอุปกรณ์ FPGAs โดยใช้โครงสร้างตัวประมวลผลเชิงเลขแบบคณิตศาสตร์กระจาย การออกแบบโดยการบรรยายด้วยภาษา VHDL และสร้างใช้งานจริงโดยอุปกรณ์ FPGAs ในการทดลองได้ออกแบบตัวกรองให้มีความถี่คัทออฟ 20 KHz โดยเก็บค่าสัมประสิทธิ์ของวงจรกรองที่คำนวณได้ไว้ในหน่วยความจำภายนอก จากผลการทดลองเลียนแบบการทำงานปรากฏให้ผลตรงตามหลักการที่นำเสนอ

Thesis Title	The Implementation of 6^{TH} Order Butterworth Digital Filter	
Students	Mr.Narong	Khongsompetch
	Mr.Thanapon	Keokumcheng
	Mr.Surakeat	Sophonputhaporn
Advisor	Mr.Kitipong	Mano
Co-Advisor	Mr.Surachai	Pimsalee
Education Level	Bachelor of Science in Industrial Education	
Program in	Electronics and Computer	
Academic Year	1999	

ABSTRACT

This thesis presents a design and implementation of 6^{th} order Butterworth digital filter with FPGAs device. It uses structure of digital signal processing of distributed arithmetic. The design describe by VHDL language and Implementation with FPGAs device. In experimental filter can use cutoff frequency at 20 kHz. The coefficients of filter circuit keep in external ROM. The result from simulation show that this method is correct.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ลุล่วงไปด้วยดี ด้วยเนื่องมาจากความร่วมมือของสมาชิกภายในกลุ่มทุกท่าน ขอขอบคุณอาจารย์กิติพงค์ มะโน สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุสาหกรรม และคณาจารย์ภาควิชาครุศาสตร์วิศวกรรมทุกท่านที่ให้ความอนุเคราะห์เครื่องมือ และอุปกรณ์ รวมทั้งยังให้คำแนะนำ แนวความคิด ความรู้ต่างๆ แนวทางแก้ไขปัญหา ในการจัดทำปริญญานิพนธ์ ขอขอบคุณห้องสมุดคณะครุศาสตร์อุตสาหกรรม ที่ช่วยอำนวยความสะดวกและเอื้อเฟื้อสถานที่ในการค้นคว้าข้อมูล สุดท้ายที่ควรระลึกถึงอย่างยิ่ง บิดา และมารดาที่เป็นผู้ให้ความสนับสนุนด้านการศึกษา และเป็นผู้ให้กำลังใจด้วยดีตลอดมา ตั้งแต่อดีตจนถึงปัจจุบัน

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

ในปัจจุบันเทคโนโลยีสมัยใหม่ได้มีการพัฒนาไปอย่างกว้างขวาง ทั้งในด้านของคอมพิวเตอร์และในด้านของอิเล็กทรอนิกส์ ซึ่งเมื่อมองโดยส่วนรวมแล้วทั้งสองต้องควบคู่กันไปไม่ว่าในทางอุตสาหกรรมหรือในทางการแพทย์ ล้วนแล้วจะต้องประกอบไปด้วยวิทยาการทางด้านอิเล็กทรอนิกส์และคอมพิวเตอร์ควบคู่กันไป ทางด้านอุตสาหกรรมก็ได้มีการนำเอาคอมพิวเตอร์เข้าไปช่วยซึ่งมีบทบาทมากทั้งทางตรงและทางอ้อม ถือได้ว่าวิทยาการทางด้านอิเล็กทรอนิกส์และคอมพิวเตอร์ก็มีบทบาทที่สำคัญในการพัฒนาเทคโนโลยีของอุตสาหกรรมและวิทยาการที่ก้าวหน้าทางอิเล็กทรอนิกส์เป็นอย่างมาก โดยเฉพาะการพัฒนาในรูปแบบของโปรแกรมที่นำมาสู่การเป็นอุปกรณ์ทางด้านอิเล็กทรอนิกส์ที่นำมาใช้เป็นวงจรในการทำงานที่มีคุณภาพสูงไม่ว่าจะเป็นการออกแบบหรือการวิเคราะห์ต่างๆ ซึ่งทั้งหมดล้วนแล้วก็ใช้แต่คอมพิวเตอร์เข้ามามีบทบาทและมีส่วนร่วมในการประยุกต์ใช้งานที่ทำให้เกิดประโยชน์สูงสุด ซึ่งโปรแกรมที่นำมาใช้ในการพัฒนาต่าง ๆ เหล่านี้ก็มีอยู่ด้วยกันหลายโปรแกรมไม่ว่าจะเป็นโปรแกรมที่ใช้ในการออกแบบวงจร โปรแกรมจัดระบบเป็นต้น และในการจัดทำโครงงานครั้งนี้ได้นำโปรแกรมชนิดหนึ่งมาใช้ในการออกแบบวงจรที่สามารถที่สร้างวงจรอิเล็กทรอนิกส์ขึ้นมาและให้สามารถใช้งานได้ โดยเป็นการเขียนโปรแกรมและนำตัวโปรแกรมไปสร้างเป็นวงจรแล้วนำไปบรรจุไว้ในอุปกรณ์อิเล็กทรอนิกส์ชนิดหนึ่งที่มีชื่อว่า FPGAs (Field Programmable Gate Arrays) อุปกรณ์ชนิดนี้สามารถสร้างเป็นวงจรอะไรก็ได้ที่มีขนาดใหญ่ให้สามารถรวมและ ยุบวงจรที่มีขนาดใหญ่ ๆ ให้เข้าไปอยู่ภายในอุปกรณ์ชนิดนี้โดยใช้โปรแกรม XACT Step ช่วยในการดาวน์โหลดลงในตัว FPGAs

ดังนั้นในการสร้างโครงงานนี้ได้มีการนำเอาโปรแกรม VHDL มาใช้ในการออกแบบวงจรอิเล็กทรอนิกส์ที่เป็นวงจรกรองความถี่ ตามชื่อหัวข้อโครงงาน การสร้างวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิตศาสตร์การกระจาย (The Implementation of 6th Order Butterworth Digital Filter) ซึ่งทั้งหมดของวงจรถูกสร้างไว้ในอุปกรณ์อิเล็กทรอนิกส์ที่เรียกว่า FPGAs โดยสามารถทำงานเหมือนกับการใช้ไอซีต่างๆ มาต่อเป็นวงจรแต่ความสามารถในการทำงานสามารถทำงานได้เหมือนกัน ซึ่งสะดวกและง่ายในการออกแบบวงจรให้สามารถใช้งานได้มีประสิทธิภาพเท่าเทียมกัน

1.2 ขอบเขตของปริญญานิพนธ์

1) ความถี่คัทออฟประมาณ 20 kHz

2) รับสัญญาณอินพุทขนาดประมาณ 5 Vp-p

3) เปลี่ยนค่าความถี่คัทออฟ 10-20 kHz โดยการเปลี่ยนแปลงค่าสัมประสิทธิ์ใน ROM

4) ตัวกรองป้อนกลับเชิงเลขใช้ บัตเตอร์เวอร์ธอันดับที่ 6

5) อัตราความถี่ในการสุ่มสัญญาณ 125 kHz

1.3 เนื้อหาโดยสังเขป

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้แบ่งออกเป็นบทต่างๆ เพื่อความสะดวกต่อการศึกษาและทำความเข้าใจในแต่ละบทประกอบด้วยเนื้อหาที่สำคัญดังนี้

บทที่ 2 ทฤษฎีและหลักการ ซึ่งกล่าวถึง VHDL กับการออกแบบอิเล็กทรอนิกส์, ส่วนประกอบหลักของภาษา VHDL, องค์ประกอบในการเขียนภาษา VHDL, สัญญาณและชนิดของข้อมูลในภาษา VHDL, ตัวดำเนินการของภาษา VHDL, Logic cell array family, โครงสร้างภายในของ FPGAs, รายละเอียดการใช้งานของอุปกรณ์ FPGAs, ข้อควรระวังในการใช้อุปกรณ์ FPGAs, ขั้นตอนของการออกแบบและสังเคราะห์โดยใช้ซอฟต์แวร์ของบริษัทไซลิงค์, ขั้นตอนการจำลองการทำงานโดยใช้ซอฟต์แวร์ของบริษัทไซลิงค์ และ ขั้นตอนการลงโปรแกรมลงบนอุปกรณ์ FPGAs โดยใช้ซอฟต์แวร์ XDM ของบริษัทไซลิงค์ (Xilinx)

บทที่ 3 การออกแบบ และการสร้างซึ่งจะกล่าวถึง ความคิดเบื้องต้นในการทำดิจิตอลฟิลเตอร์, ดิจิตอลฟิลเตอร์แบบคณิตศาสตร์แจกแจง, หลักการของโครงสร้างแบบเลขคณิตศาสตร์แจกแจง, การสร้างตารางเปิดดูจากฟังก์ชั่น Fi{.} สำหรับตัวกรองอันดับสอง, การคำนวนหาY(n) ในกรณีตัวกรองอันดับสอง, การคำนวนหา Y(n) ในกรณีตัวกรองอันดับหก, การออกแบบสร้างวงจรกรองสัญญาณเชิงเลขอันดับหก, ลักษณะการออกแบบ, วิธีการออกแบบ, การออกแบบวงจรกรองสัญญาณเชิงเลขด้วยภาษา VHDL และขั้นตอนการออกแบบผังการทำงานภายใน

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึงผลการทดลองโปรแกรม VHDL, การดาวน์โหลดโปรแกรมลงบนบอร์ดตัวอย่างของ FPGAs และผลการทดลอง

บทที่ 5 สรุปอภิปรายและข้อเสนอแนะ กล่าวถึงปัญหา, แนวทางการแก้ไข, และแนวทางในการพัฒนาเพื่อสามารถนำไปประยุกต์ใช้ได้อย่างกว้างขวาง

ในภาคผนวกแสดงรายละเอียดของโปรแกรม แผ่นวงจรพิมพ์ และรายการอุปกรณ์ต่างๆ ที่ใช้จัดทำโครงงานดังนี้

ภาคผนวก ก โปรแกรมการทำงานของวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์อันดับที่ 6 โดยใช้คณิตศาสตร์การกระจาย

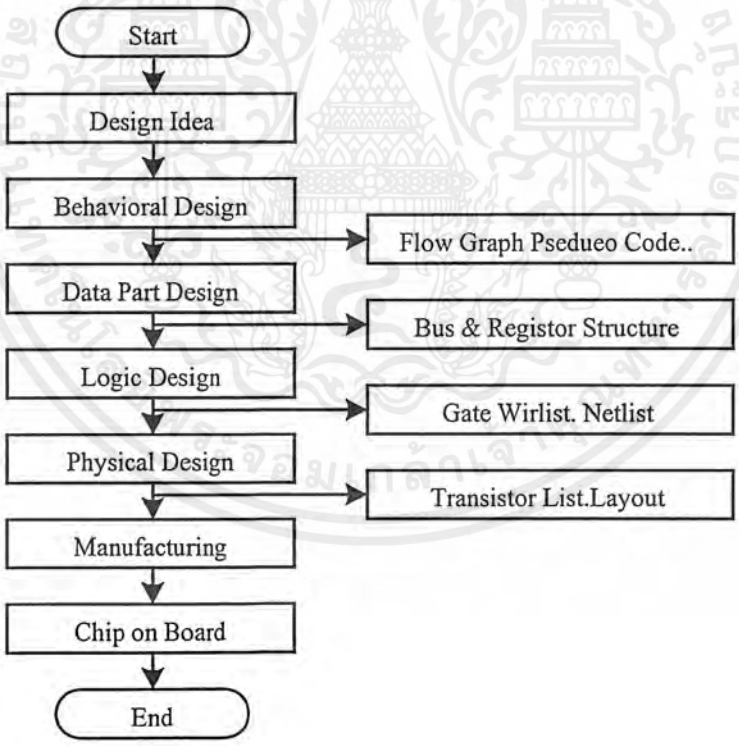
ภาคผนวก ข แผ่นวงจรพิมพ์

ภาคผนวก ค รายละเอียดข้อมูล และคุณสมบัติของอุปกรณ์

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

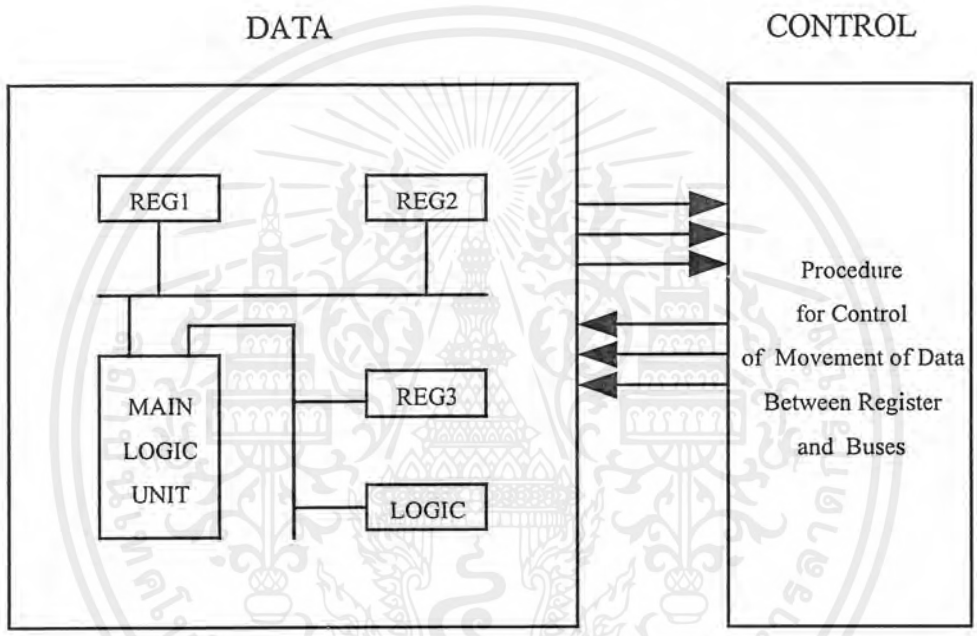
เพื่อเป็นการแก้ปัญหาที่ยุ่งยากในการออกแบบเชิงเลขจึงได้นำเอาระบบคอมพิวเตอร์ช่วยออกแบบทางวิศวกรรมเข้ามาช่วยในการออกแบบ ทั้งยังช่วยลดความยุ่งยากและระยะเวลาในการออกแบบให้รวดเร็วขึ้น ภาษาสำหรับบรรยายอุปกรณ์ฮาร์ดแวร์เป็นภาษาหนึ่งที่ได้รับการพัฒนาและนำมาใช้ในการออกแบบระบบเชิงเลข โดยที่มีขั้นตอนตั้งแต่การกำหนดแนวความคิดในการออกแบบจนกระทั้งได้มาเป็นอุปกรณ์ ที่ถูกโปรแกรมการทำงานตามการวางแนวความคิดให้อุปกรณ์ทำงานได้ตามต้องการ



รูปที่ 2.1 ขั้นตอนการออกแบบระบบเชิงเลข

ขั้นตอนดังกล่าวนั้นสามารถอธิบายได้ดังรูปที่ 2.1 โดยที่ขั้นแรกผู้ออกแบบเริ่มกำหนดแนวความคิดในการออกแบบหลังจากนั้นออกแบบระบบในเชิงพฤติกรรมขึ้นมาเพื่อตรวจสอบการทำงาน ซึ่งอาจจะเป็นผังงาน, ผังแสดงแบบ, หรือคำสั่งเทียม (Pseudo Code) ก็ได้

ขั้นต่อมาคือ การออกแบบเส้นทางเดินของข้อมูลให้เชื่อมต่อกันอาจจะเชื่อมต่อกันด้วยระบบบัสหนึ่งหรือสองทิศทาง (Unidirection or Bidirectional Bus) เพื่อเชื่อมต่อรีจิสเตอร์ (Registor) และวงจรตรรก (Logic) ให้เป็นระบบที่สมบูรณ์ดังรูปที่ 2.2



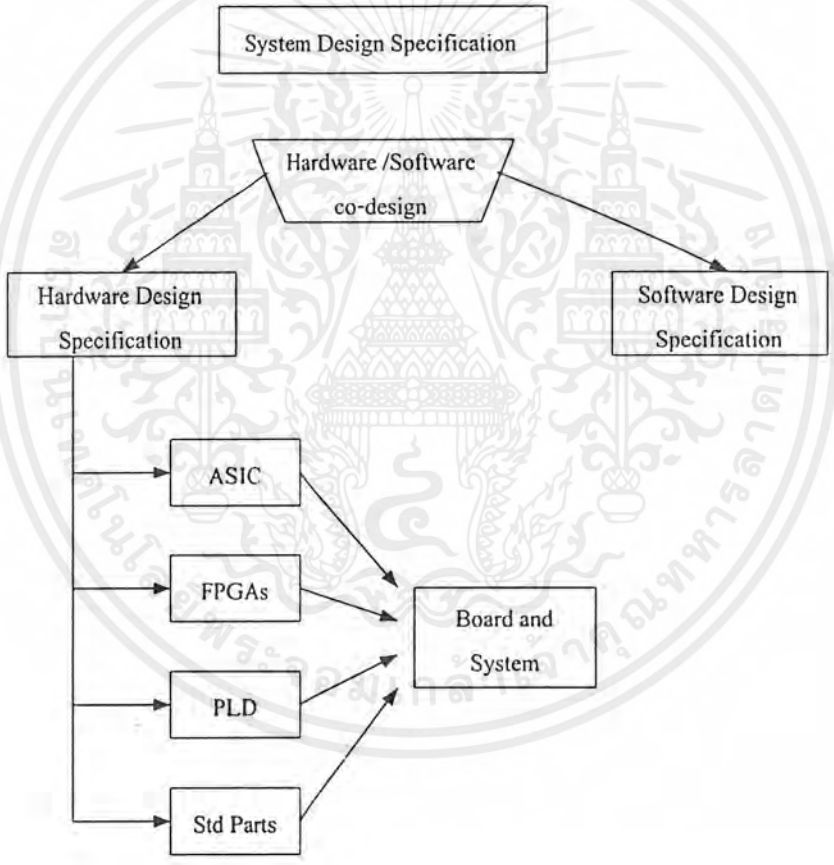
รูปที่ 2.2 การออกแบบระบบเส้นทางของข้อมูล

จากนั้นต่อไปจะเป็นขั้นตอนของการเชื่อมโยงเกตพื้นฐานและฟลิปฟลอป จนมาถึงขั้นตอนการออกแบบทางกายภาพ คือเป็นการทำทรานซิสเตอร์ลิส และการเลย์เอาท์ (Transistor List and Laout) จนกระทั่งเป็นขั้นตอนการผลิตลงบน Chip ซึ่งอาจเป็นอุปกรณ์ FPGAs, PLD หรือ Std Port หรือเป็นการส่งระบบที่ออกแบบสมบูรณ์และไปทำการเจือสารที่โรงงานผลิตออกมาเป็นวงจรรวม

2.2 แนะนำภาษา VHDL

VHDL เป็นภาษาบรรยายทางฮาร์ดแวร์สำหรับการออกแบบในระดับสูงทางอิเล็กทรอนิกส์ (Electronics) มีความสามารถในการเลียนแบบ (Simulation), การสังเคราะห์ (Synthesis), และ การทดสอบ (Testing) ในการจำลองระบบอิเล็กทรอนิกส์ดิจิตอลลักษณะการทำงานของ VHDL สามารถทำงานได้ทั้งแบบขนาน (Concurrent) และลำดับ (Sequential Statements)

2.3 VHDL กับการออกแบบอิเล็กทรอนิกส์

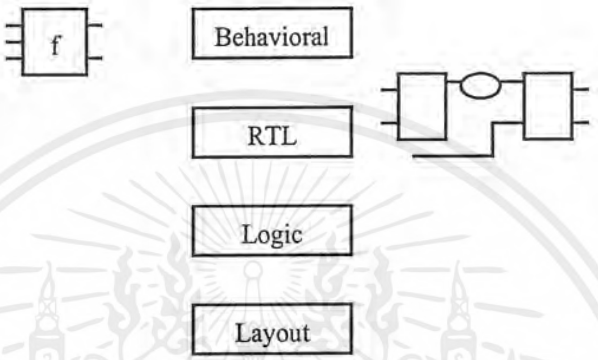


รูปที่ 2.3 การออกแบบระบบอิเล็กทรอนิกส์

จากรูปที่ 2.3 แสดงกระบวนการทำงานของภาษา VHDL จากการออกแบบทางฮาร์ดแวร์ผ่านกระบวนการแปลภาษา (Compile) การจำลองการทำงาน (Simulate) เพื่อตรวจสอบดูว่าผล

การทำงานเป็นไปตามจุดประสงค์หรือไม่ หลังจากนั้นนำไปสังเคราะห์ เพื่อให้ได้เป็นวงจรแสดงการเชื่อมต่อซึ่งประกอบด้วยเกต (Gate), ฟลิปฟลอป (Flip-Flop), และอุปกรณ์พื้นฐานต่างๆ แล้วนำไปโปรแกรมลงบนอุปกรณ์ประเภท ASIC, FPGAs, PLD หรือ Std part จนเป็นวงจรรวมแบบสมบูรณ์

2.3.1 ขั้นตอนในการออกแบบ



รูปที่ 2.4 ขั้นตอนการออกแบบ

1) Behavioral เป็นขั้นตอนในการสร้างแบบจำลองการบรรยายพฤติกรรมในขั้นตอนแรก

2) RTL (Registor Transfer Logic) การบรรยายการถ่ายโอน Logic

3) Logic เป็นขั้นตอนการสังเคราะห์ให้ได้เป็นวงจรแสดงการเชื่อมต่อ ซึ่งประกอบด้วยเกตต่างๆ

4) Layout เป็นขั้นตอนของการเจือสาร

ในส่วนของภาษา VHDL ครอบคลุมในขั้นตอน Behavioral, RTL และ Logic

วัตถุประสงค์ในขั้นตอนของ Behavioral มาถึงขั้น RTL ก็เพื่อ

1) ช่วยในการกำหนดการทำงานและการแก้ไขจุดผิดพลาดของโปรแกรม

2) ช่วยในการวิเคราะห์ปัญหาของโปรแกรม

3) เป็นวิธีในการระบุรายละเอียดและการแยกส่วนการทำงานของโปรแกรม

4) เป็นการตรวจสอบและเปรียบเทียบผลลัพธ์ของโปรแกรม

2.3.2 ข้อได้เปรียบของการใช้ VHDL

1) การออกแบบมีคุณภาพที่สูง

2) ออกแบบวงจรที่มีความซับซ้อนมากๆ ได้

3)ใช้เวลาในการออกแบบสั้นกว่า

4) ค้นหาข้อผิดพลาดและเปลี่ยนแปลงแก้ไขได้ง่าย

2.4 ลักษณะการเขียนภาษา VHDL

ในเชิงพฤติกรรมเป็นการบรรยายลักษณะการทำงานของอุปกรณ์ฮาร์ดแวร์ในเชิงพฤติกรรม เป็นการบรรยายเป็นลักษณะการเปลี่ยนแปลงของข้อมูลในรูปแบบของข้อมูลที่เข้ามาโดยไม่คำนึงถึงว่าลักษณะของโครงสร้างหรือความสัมพันธ์ของอุปกรณ์หรือความสัมพันธ์ของอุปกรณ์ที่อยู่ภายใน

ในเชิงข้อมูลเป็นการบรรยายถึงการเลื่อนไหลของข้อมูลผ่านรีจิสเตอร์ และบัสของระบบ เป็นระดับขั้นของการบรรยายที่อยู่ตรงกลางระหว่างการบรรยายเชิงพฤติกรรม และการบรรยายเชิงโครงสร้างเครื่องมือที่ใช้ในการควบคุมการไหลเลื่อน ได้แก่ Conditional, Selected และ Guarded

ในโครงสร้างเป็นการบรรยายการทำงานของระบบเชิงโครงสร้าง ต้องแสดงรายการของอุปกรณ์ทั้งหมดที่ใช้ในระบบ และต้องกำหนดการเชื่อมต่อระหว่างอุปกรณ์ต่างๆ ด้วยเพราะว่าการบรรยายในระดับนี้เป็นการบรรยายที่ใกล้เคียงลักษณะของฮาร์ดแวร์มากที่สุด ภาษา VHDL ได้จัดเตรียมเครื่องมือ และลักษณะโครงสร้างของการบรรยายในลักษณะนี้ไว้ที่สำคัญ 4 ลักษณะคือ

1) ความสามารถในการเลือกหรือกำหนดอุปกรณ์ที่ต้องการได้จากไลบราลี

2) การสร้างไลบราลีเพื่อเก็บอุปกรณ์ที่ผู้ใช้ออกแบบไว้เองได้

3) กลไกในการเชื่อมต่อระหว่างอุปกรณ์

4) โครงสร้างของการกำหนดอุปกรณ์ชนิดเดียวกันซ้ำๆ กัน

2.5 ส่วนประกอบหลักของภาษา VHDL

2.5.1 โครงสร้างของภาษา VHDL

จากเอกสารของ DOD (The United State Department of Defense) ได้กล่าวไว้ว่าภาษาบรรยายทางฮาร์ดแวร์ซึ่งเป็นภาษาที่ต้องการการกำหนดรูปแบบทางโครงสร้างเพราะในการกำหนดโครงสร้างนี้จะช่วยในการออกแบบ ทั้งระบบที่ง่ายไปจนถึงระบบที่ซับซ้อน ซึ่งระบบที่ว่านี้ถือเป็นมาตรฐานของภาษาบรรยายทางฮาร์ดแวร์

2.5.2 การทำงานที่พร้อมกัน

ในวงจรอิเล็กทรอนิกส์ อุปกรณ์ทุกๆ ตัวอยู่ในสภาพเตรียมพร้อมเสมอและมีเรื่องของเวลาเข้ามาควบคุมการทำงานเสมอ ภาษา VHDL ได้รับการออกแบบมาเพื่อบรรยายรูปแบบทางดิจิตอล

และการป้องกันของเวลาสำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วนของสถาปัตยกรรม (Architecture) มีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซสซึ่งมีการทำงานภายในเป็นลำดับคำสั่งก็ตาม หากมีหลายโปรเซสอยู่ภายในโครงสร้างเดียวกันทุกโปรเซสจะทำงานไปพร้อมกันด้วย

2.5.3 ลำดับคำสั่ง

ถึงแม้ว่าลักษณะเฉพาะของภาษาบรรยายทางฮาร์ดแวร์ จะสนับสนุนการปฏิบัติตามคำสั่งอย่างเป็นลำดับเป็นกระบวนการ ตัวภาษา VHDL ก็ยังจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งเอาไว้ เมื่อนักออกแบบได้บรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดในของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรม ที่ประกอบด้วยโครงสร้างแบบ if-then-else และ loop การบรรยายแบบลำดับคำสั่ง ทำให้การออกแบบหน้าที่การทำงานของภาษา VHDL ก็ยังเป็นการทำงานแบบพร้อมเพรียงกันอยู่

2.5.4 เวลาที่ใช้ในการควบคุม

ภาษา VHDL เป็นภาษาที่อนุญาตให้ผู้ออกแบบ สามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ การตรวจสอบการออกแบบเกต หรือการหน่วงเวลาสามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอยเหตุการณ์ นอกจากนี้ก็ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้

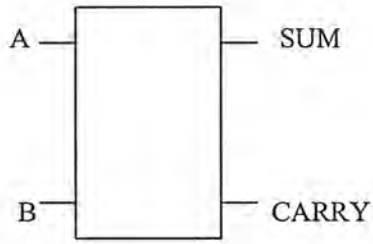
2.6 องค์ประกอบในการเขียนภาษา VHDL

สามารถแบ่งองค์ประกอบในการเขียนภาษา VHDL ได้ Entity, Architecture, Process, Configuration, Package และ Library

2.6.1 Entity

เป็นส่วนที่ใช้ในการประกาศ อธิบายการเชื่อมต่อกับองค์ประกอบภายนอกแต่ยังมิได้กำหนดการกระทำใดๆทั้งสิ้น ซึ่งในส่วนประกอบนี้ต้องกำหนดชื่อ (ห้ามซ้ำกับชื่อสงวนของภาษา VHDL) แล้วตามด้วย is ตัวอักษรในภาษา VHDL ทั้งตัวใหญ่ตัวเล็กมีค่าเท่ากันดังตัวอย่างแสดงในรูปที่ 2.5

```
entity HALFADD is
    port (A,B:in bit
        SUM,CARRY:out bit);
    End HALFADD;
```



รูปที่ 2.5 โปรแกรมการใช้งานและลักษณะการติดต่อใช้งานของ entity

จากรูปที่ 2.5 มีการประกาศชื่อ entity HALFADD ตาม is แล้วมีการกำหนดสัญญาณอินพุตและเอาต์พุต โดยมีวงเล็บเปิดและวงเล็บปิด ในแต่ละบรรทัดถูกปิดท้ายด้วย (;) และใช้ในการกำหนดสัญญาณให้กับตัวแปรและจบ entity ด้วยคำสั่ง end ตามด้วยชื่อของ entity

2.6.2 Architecture

เป็นส่วนที่ใช้การกำหนดการทำงานภายในของ entity ต้องเป็นการเชื่อมโยงกันของ entity จำเพาะซึ่ง entity เดียวสามารถมีได้หลาย Architecture ต้องมีการกำหนดชื่อของ Architecture นั้นๆ ด้วยว่าเป็นของ entity ไหน การเขียน Architecture จะตามหลัง begin เสมอและจบด้วยคำสั่ง end ตามด้วยชื่อของ Architecture นั้น ดังตัวอย่างแสดงในรูปที่ 2.6

```
architecture BEHAVE of HALFADD is
begin
    SUM <= A xor B;
    CARRY<= A and B;
end BEHAVE;
```

รูปที่ 2.6 โปรแกรมการใช้งานและลักษณะการติดต่อใช้งานของ Architecture

2.6.3 Configuration

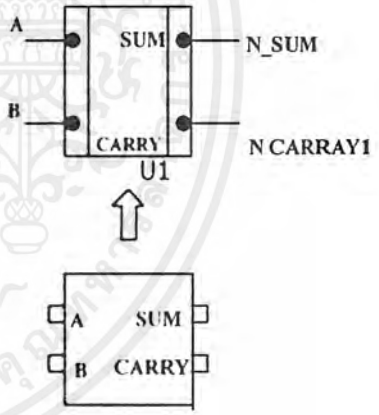
ในภาษา VHDL สามารถสร้างอุปกรณ์เก็บเอาไว้ได้ และสามารถจำลอง Socket ของอุปกรณ์แต่ละตัว เพื่อความสะดวกเหมือนกับการใช้งานจริงในการเชื่อมต่ออุปกรณ์ดังรูปที่ 2.7 u1 ถูกสร้างขึ้นเป็น Socket ให้กับอุปกรณ์ โดยในการนำไปใช้งานนั้นอาศัย Component ตามด้วยชื่อของอุปกรณ์มีการกำหนดสัญญาณอินพุตเป็นแบบบิตให้กับตัวแปร A และ B ส่วนเอาต์พุตก็เป็น

สัญญาณแบบแบบบิตที่กำหนดค่าให้กับตัวแปร SUM และ CARRY เหมือนกับ entity HALFADD ทุกประการ และจบด้วยคำสั่ง end component; สำหรับการเชื่อมต่อใช้คำสั่ง port map โดยเป็นการให้กับทางขวามือ คือ u1: HALFADD port map (A,B,N_SUM,N_CARRY); โดยการเขียนนี้ต้องอยู่หลังจาก begin ดังตัวอย่างแสดงในรูปที่ 2.8 และรูปที่ 2.9

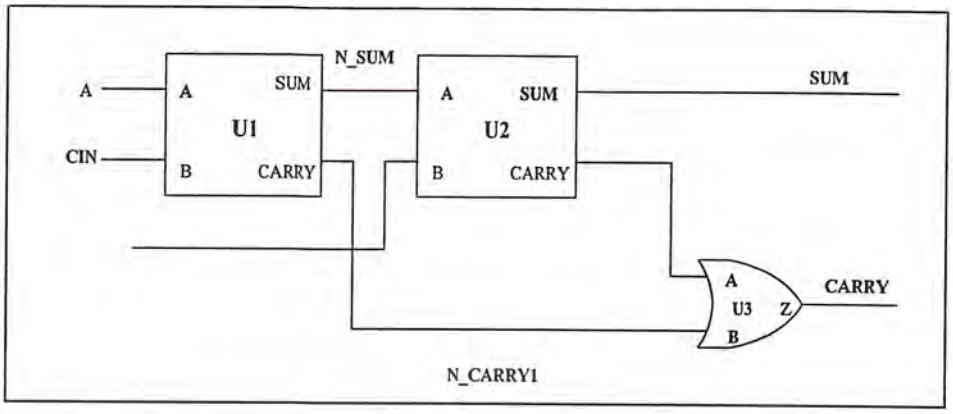
```
component HALFADD
 port (A,B : in bit);
       SUM, CARRY : out bit);
 end component
 ...
begin
 u1:HALFADD port map (A,B,N_SUM,N_CARRY);
 ...
```

```
DEFAULT
CONFIGURATION
```

```
entity HALFADD is
Port (A,B : in bit;
      SUM, CARRY : out bit
end HALFADD;
```



รูปที่ 2.7 โปรแกรมและการติดต่อใช้งานของ Configuration



รูปที่ 2.8 การเชื่อมต่อของอุปกรณ์ภายใน

```
entity FULLADD is
port (A,B,CIN : in bit);
      SUM, CARRY : out bit);
end FULLADD;
architecture STRUCTURAL of FULLADD is
      signal N_SUM, N_CARRY1, N_CARRY2 : bit;
      component HALFADD
      port (A,B : in bit;
            SUM,CARRY: out bit);
      end component;
begin
      u1 : HALFADD port map (A,B,N_SUM, N_CARRY1);
      u2 : HALFADD port map (N_SUM, CIN,SUMN_CARRY2);
      u3 : ORGATE port map (N_CARRY2, N_CARRY1, N_CARRY);
end STRUCTURAL;
```

รูปที่ 2.9 โปรแกรมการเชื่อมต่ออุปกรณ์ภายในและการประกาศสายสัญญาณ

จากรูปที่ 2.9 มีการประกาศสัญญาณภายในอุปกรณ์คือ Signal N_CARRY1, N_CARRY2 ให้มีสัญญาณเป็นแบบบิต จะสังเกตเห็นว่าวงจรมีอุปกรณ์ภายใน 3 ตัว คือ HALFADD2 ตัวและ

ORGATE 1 ตัวจึงประกอบด้วย u1, u2, u3 และสัญญาณที่เขียนใน port map จะเรียงสัญญาณเข้าจนถึงสัญญาณออกตามลำดับและจบโปรแกรมด้วยคำสั่ง ตามชื่อของ Architecture เสมอส่วนของ component เป็นส่วนหนึ่งของ Architecture

2.6.4 Process

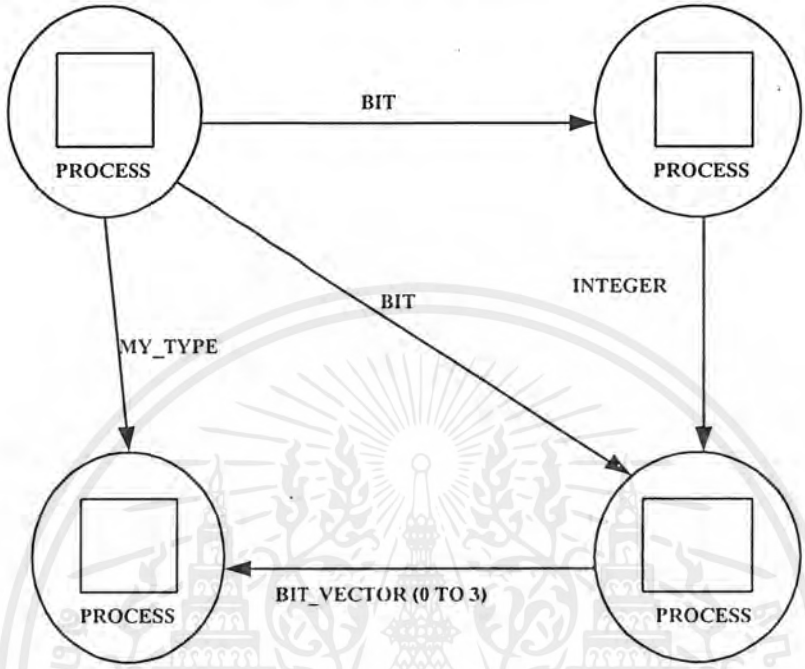
ส่วนนี้เป็นการบรรจุคำสั่งลำดับที่ต้องเขียนอยู่ภายใน Architecture ซึ่งโปรเซสหลายๆ โปรเซสจะมีการทำงานที่พร้อมกัน (Concurrent) ตัวแปรทางขวามือจะถูกกำหนดค่าให้จากสัญญาณทางซ้ายมือ (<=) ในการเริ่มโปรแกรมใช้คำสั่ง PROCESS และจบด้วยคำสั่ง END PROCESS ส่วนประกอบของการบรรยายโปรเซสประกอบด้วยส่วนของการประกาศตัวแปรที่ต้องใช้และส่วนของการปฏิบัติตามคำสั่งเพื่อให้ได้ผลลัพธ์ตามที่ต้องการ ดังตัวอย่างที่ต้องการ ดังรูปที่ 2.10

```
entity AND_OR is
        port (A,B : in bit ;
                Z_OR, Z_AND : out bit);
end AND_OR
architecture BEHAVE of AND_OR is
begin
        AND_OR_FUNC : process (A,B)
        begin
                if (A='1' or B='1') then  Z_OR <= '1';
                  else    Z_OR <= '0';   end if
               if (A='1' and B="1") then  Z_AND <= '1' ;
                  else    Z_AND ,= '0' ;  end if  ;
          end process AND_OR_FUNC ;
end BEHAVE
```

รูปที่ 2.10 โปรแกรมการทำงานของโปรเซส

จากรูปที่ 2.10 ชื่อของ Process ชื่อของโปรเซสคือ AND_OR_FUNC มีสัญญาณอินพุตเป็น A และ B โดยใช้คำสั่ง if-then-else ในการลำดับการทำงานดังนี้ ถ้า A=1 หรือ B=1 จะให้ Z_OR มีค่าเป็น 1 เงื่อนไขนอกจากนั้น Z_OR จะมีค่าเป็น 0 จบการทำงานด้วย end if ; และถ้า A=1 และ B=1

ดังนั้น Z_AND จะมีค่าเป็น 1 เงื่อนไขนอกจากนั้น Z_AND จะมีค่าเป็น 0 และจบ end process AND_OR_FUNC ; และจบ Architecture ด้วยคำสั่ง end BEHAVE;



รูปที่ 2.11 การเชื่อมต่อหลายๆ Process ที่สามารถเชื่อมต่อกันได้ด้วยสัญญาณที่ต่างกัน

2.6.5 Package

เป็นส่วนของโปรแกรมที่แปล (Compile) เรียบร้อยแล้ว และดึงออกมาใช้เท่านั้นหรืออาจกล่าวได้ว่า Package คือชนิดของข้อมูล โปรแกรมย่อยหรืออุปกรณ์ต่างๆ ที่ได้ออกแบบไว้แล้วนำมารวบรวมไว้เป็นกลุ่มๆ อยู่ภายในเพื่อให้ผู้ออกแบบเรียกใช้ได้สะดวกดังรูปที่ 2.12

```
Package  DEMO_PACK is
  -  contants
  -  data types
  -  components
  -  subprogram
end DEMO_PACK
```

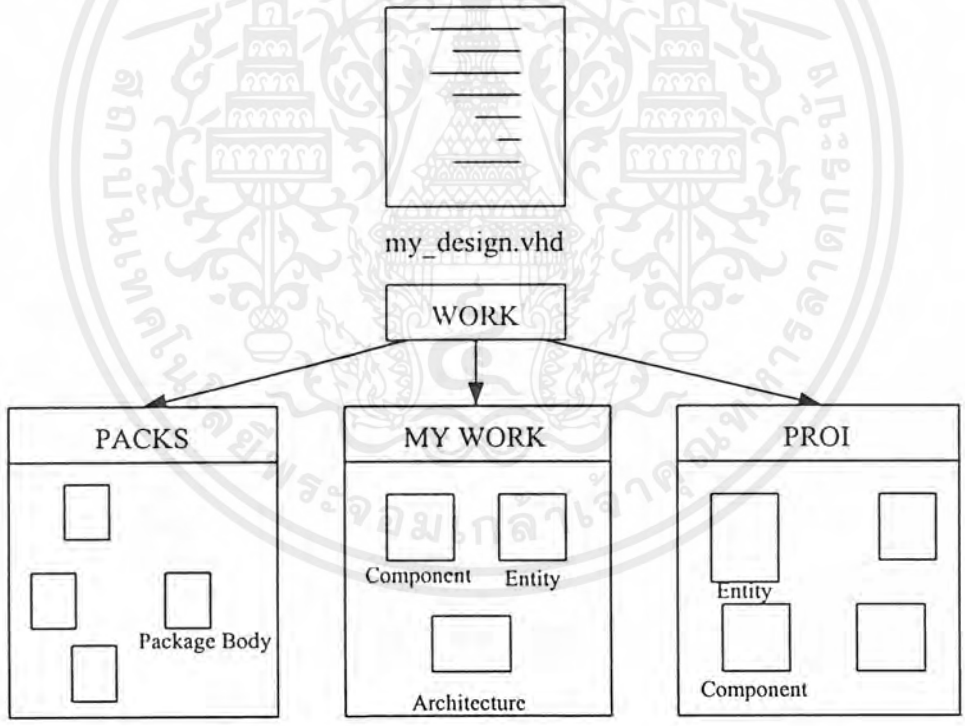
รูปที่ 2.12 โครงสร้างของ Package

```
Package body DEMO_PACK is
  - contant  values
  - subprogram definition
end DEMO_PACK;
```

รูปที่ 2.12 (ต่อ)โครงสร้างของ Package

2.6.6 Library

เป็นส่วนของการเก็บโปรแกรมต่างๆ ที่สมบูรณ์ซึ่งภายใน Library ประกอบด้วย Entity, Architecture, Package, Configuration มีลักษณะการใช้งานดังรูปที่ 2.13



รูปที่ 2.13 ส่วนของการเก็บโปรแกรมต่างๆที่สมบูรณ์

จากรูที่ 2.13 ก่อนการเรียกใช้ Work จะต้องทำการแปล (Complie) ที่ Package, Mywork และ Proi ก่อนแล้วจึงสามารถเรียกใช้ได้

2.7 Logic cell array family (XC4000)

1) คุณลักษณะ

1.1) เป็น Generation ที่ 3 ของ FPGAs

1.1.1) มี Flip-Flop เป็นจำนวนมาก

1.1.2) ในการผลิตฟังก์ชันของการทำงานมีการยืดหยุ่นสูง

1.1.3) มี Ultra-Fast RAM ในตัว

1.1.4) ใช้กับงานที่ต้องการความเร็วสูง

1.1.5) มี Wide Edge Decoder

1.1.6) Interconnect Line เป็นแบบ Hiearchy

1.1.7) สามารถใช้ 3-State Bus ภายในได้

1.1.8) มีการกระจายกำลังงานของสัญญาณต่ำ

1.2) มีสถาปัตยกรรมภายในที่ยืดหยุ่น

1.2.1) มี Logic Blocks และ I/O Blocks ที่โปรแกรมได้

1.2.2) มี Interconnect และ Wide Decoder ที่โปรแกรมได้

1.3) ทำกระบวนการ Sub-Micron ชนิด CMOS ได้

1.3.1) มี Logic และ Interconnect ที่มีความเร็วสูง

1.3.2) ใช้กำลังงานต่ำ

1.4) คุณลักษณะทาง System-Oriented

1.4.1) รองรับมาตรฐาน IEEE 1149.1 ในการทำ Boundary-Scan Logic

1.4.2) สามารถโปรแกรมค่า Output Slew Rate ได้

1.4.3) สามารถโปรแกรมให้ Input มี Pull-Up หรือ Pull-Down Resister ได้

1.4.4) ให้กระแส Output ได้ตั้งแต่ 12-24 mA (แล้วแต่รุ่น)

1.5) ทำการ Config โดยการโหลดเอาแฟ้มข้อมูล Binary

1.5.1) ไม่จำกัดจำนวนครั้งในการโปรแกรมซ้ำ

1.5.2) มี Mode ในการโปรแกรมให้เลือก 6 Modes

1.6) มีโปรแกรม XACT Development System ที่ทำงานบน PC 386, 486, NEC PC, Apollo, Sun-4 และ Hewlett-Packard 700 Series

1.6.1) สามารถติดต่อกับโปรแกรมอื่นๆ ได้เช่น VIEW Logic, Mentor Graphics และ OrCAD

1.6.2) มี Automatic Place and Routing ที่ครบ

1.6.3) มี Interactive Design Editor ที่ใช้สำหรับการทำ Optimization

1.6.4) มี 288 Macros, 34 Hard Macros, RAM/ROM Complier

2) รายละเอียด

LCA ตระกูล XC4000 ประกอบไปด้วยวงจร Logic ที่มีความหนาแน่นสูง, มีความยืดหยุ่น, การโปรแกรมสถาปัตยกรรมของ CLBs ที่ต่อกันภายในมีประสิทธิภาพมากในวิธีการแบบ Hierachy , และการต่อ IOBs ที่อยู่รอบๆก็ใช้วิธีการ Perimeter

2.8 โครงสร้างภายในของอุปกรณ์ FPGAs

FPGAs จัดเป็นวงจรรวมเฉพาะกิจชนิดหนึ่งที่สามารถโปรแกรมเป็นวงจรเชิงเลขใดๆ ก็ได้ เช่นเดียวกับ EPLD ต่างกันที่ EPLD โปรแกรมลงบน EPROM ภายในและสามารถโปรแกรมใหม่ได้หลังจากนำไปลบด้วยแสง UV แต่ FPGAs โปรแกรมลงบนสแตติกแรมภายในด้วย ข้อมูลที่อยู่ภายนอก และสามารถโปรแกรมใหม่ได้โดยการรีเซตด้วยสัญญาณไฟฟ้านอกจากนั้น FPGAs ยังประหยัดและมีความจุวงจรสูง (จำนวนเกตมาก) ได้อีกด้วย

วงจรรวมชนิดที่ใช้ในโครงงานนี้ผลิตโดยบริษัทไซลิงค์ (Xilinx) ซึ่งเป็นบริษัทที่ทำการค้นคว้าร่วมกับบริษัทเอ็มเอ็มไอ (MMI) สร้างเป็นกลุ่มของเกตจำนวน 600-25,000 เกตดังแสดงในตารางที่ 2.1 การที่ต้องการบอกขนาดของวงจรรวมเป็นจำนวนเกตเพราะจะได้รู้ว่าขนาดของวงจรที่ได้ออกแบบไว้สามารถโปรแกรมลงบนวงจรรวม FPGAs ได้หรือไม่

FPGAs มีโครงสร้างภายในใกล้เคียงกับสถาปัตยกรรมของเกตอะเรย์ (GAL,Gate Array Logic) มาก สามารถโปรแกรมและลบคอนฟิกกูเรชั่น (Configuration) ภายในสแตติกแรม (Static RAM) ได้ โดยใช้กระแสไฟฟ้าซึ่งทำการโปรแกรมได้ โดยดึงข้อมูลฐานสิบหกมาจากภายนอก เช่น Parallel EPROM หรือ PROM ต่างกับ EPLD, PAL ที่มี EPROM อยู่ในตัวภายใน FPGAs จะจัดเรียงเป็นลอจิกเซลล์ล้อมรอบภายนอกด้วยอินพุตเอาต์พุตเซล FPGAs ตัวแรกที่ผลิตโดยบริษัทไซลิงค์คือเบอร์ XC2064 (2000 Family) ประกอบด้วยเซลเรียงกันเป็นเมตริกซ์ (Matrix) เป็นจำนวน 64 เซล หลังจากนั้นผลิต FPGAs ตระกูล 3000 และ 4000 ซึ่งมีโครงสร้าง ซับซ้อนขึ้น สามารถเพิ่มจำนวนเกตได้มากขึ้นและดีขึ้น แต่ละเซลเรียกว่า CLB (Configurable Logic Block)

ตารางที่ 2.1 คุณสมบัติของ FPGAs ตระกูลต่างๆ

FPGAs	App.Gate Count	Max I/Os	Flip-Flops	RAM bits	Available CLBs
XC2064	1,000	58	122	0	64
XC2018	1,500	74	174	0	100
XC3020/3120	1,800	64	256	0	64
XC3030/3130	2,700	80	360	0	100
XC3042/3142	3,700	96	480	0	144
XC3064/3164	5,500	120	688	0	244
XC3090/3190	7,500	144	928	0	320
XC3195	9,000	176	1,320	0	484
XC4002A	2,000	64	256	2,048	64
XC3003/4003A	3,000	80	360	3,200	100
XC4000H	3,000	160	200	3,200	100
XC4004A	4,000	960	480	4,608	144
XC4005/4005A	5,000	122	616	6,072	196
XC4005H	5,000	192	392	6,272	196
XC4006	6,000	128	768	8,192	256
XC4008	8,000	144	936	10,368	324
XC4010	10,000	160	1,120	12,800	400
XC4013	13,000	192	1,536	18,432	576
XC4025	25,000	256	2,560	32,768	1,024

ตารางที่ 2.2 ประมาณการนับเกตของเกตพื้นฐาน

Gate	Equivalent gate count	Gate	Equivalent gate count
INV	1	RS Latch	3
NAIINR2	1	D Latch	4
NAIINR3	2	D Latch with CLR	5
NAIINR4	2	D Latch with PRE	5
NAIINR6	5	D Latch with PRE/CLR	6
NAIINR8	6	D F/F	6
NAIINR9	7	D F/F with CLR	7
NAIINR12	8	D F/F with PRE	7
NAIINR16	11	D F/F with PRE/CL	8
BUFF	2	JK F/F with CLR	9
ANIIOR2	2	JK F/F with PRE	12
ANIIOR3	2	JK F/F with PRE/CL	13
ANIIOR4	3	T F/F with CLR	8
XOR2	3	T F/F with PRE	8
XNOR2	3	T F/F with PRE/CL	9

หมายเหตุ : NAIINR2 หมายถึงเกต NAND2 หรือ เกต NOR2

2.8.1 ส่วนที่เป็นองค์ประกอบของลอจิก (Configurable Logic Block)

CLB จะจัดเรียงกันเป็นแบบเมตริกต์แบบอะเรย์ขนาด M*N การออกแบบนั้นสามารถทำได้โดยการจัดวาง CLB และเชื่อมต่อขาของ CLB ให้ต่อถึงกัน เราสามารถจัด CLB ให้ต่อถึงกันได้โดยการทำด้วยมือหรือให้โปรแกรมที่สนับสนุน FPGAs ทำให้โดยอัตโนมัติโดยวิธีของมันเอง สำหรับไฟล์ที่ได้จากโปรแกรมเหล่านี้เราเรียกว่าไฟล์ที่กำหนดการวางอุปกรณ์ (Configuration File) ซึ่งจะบรรจุโครงร่างภายในของ CLB ตามความเหมาะสมอีกด้านหนึ่งไฟล์กำหนดการวางอุปกรณ์นั้นจะเป็นไฟล์กระแสข้อมูล (Bit Stream) ซึ่งสามารถใช้โปรแกรมหน่วยความจำภายในของ FPGAs ได้

2.8.2 ส่วนอินพุตและเอาต์พุตของอุปกรณ์ FPGAs

รอบนอกของ FPGAs จะประกอบด้วย IOBs ประมาณ 64 ถึง 144 ตัว ซึ่งขึ้นอยู่กับตระกูลของ FPGAs ซึ่ง IOBs จะเป็นตัวเชื่อมต่อระหว่างภายในกับภายนอกของวงจรลอจิกของ FPGAs ลักษณะของ IOBs จะมีลักษณะ 2 ทิศทาง สามารถโปรแกรมให้เป็นอินพุตหรือเอาต์พุตก็ได้

2.9 รายละเอียดการใช้งานของอุปกรณ์ FPGAs

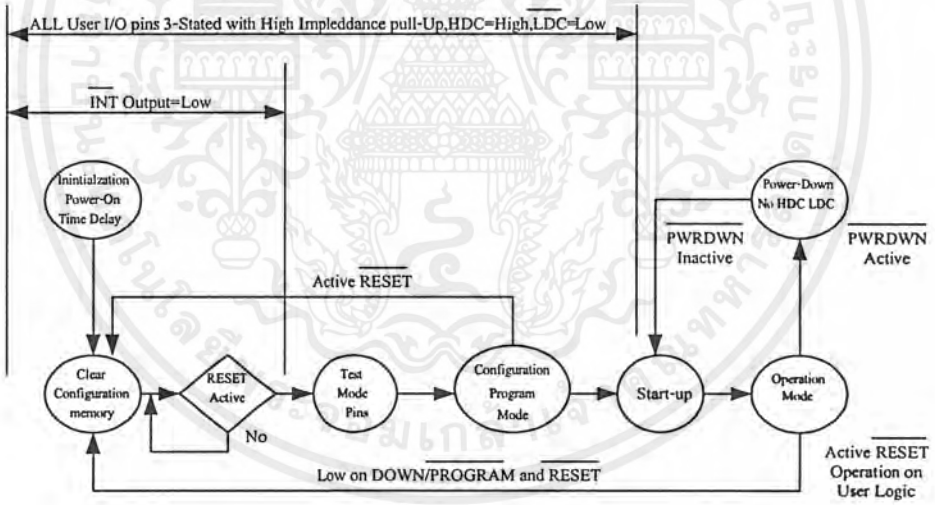
FPGAs สามารถทำงานได้หลายลักษณะ โดยกำหนดได้ที่ขาสัญญาณ M0, M1, และ M2 ดังแสดงในตารางที่ 2.3 ในลักษณะมาสเตอร์พาราแรล (Master Parallel Mode) รับโปรแกรม คอนฟิกทีละ 1 ไบท์ (Byte) จากหน่วยความจำภายนอกที่เป็นแบบขนาน โดยสามารถรับโปรแกรมคอนฟิก (Config) จากแอดเดรส (Address) ต่ำหรือสูงก่อนก็ได้ การต่อลักษณะเพอริเฟอรัล (Peripheral) จะรับโปรแกรมคอนฟิกทีละ 1 ไบท์จากไมโครโปรเซสเซอร์ โดยสามารถโต้ตอบกันได้ว่าพร้อมหรือไม่ที่จะรับข้อมูล การต่อลักษณะสเลฟซีเรียล (Slave Serial) จะรับโปรแกรมคอนฟิกทีละ 1 บิต (Bit) จากไมโครโปรเซสเซอร์ตามสัญญาณอินพุต CCLK ส่วนการต่อลักษณะมาสเตอร์ซีเรียล (Master Serial) จะรับโปรแกรมคอนฟิกทีละ 1 บิตจากหน่วยความจำภายนอกที่เป็นแบบอนุกรม

ตารางที่ 2.3 โหมดต่าง ๆ ของการคอนฟิกกูเรชัน

Mode	M2	M1	M0	CCLK	Data
Master Serial	0	0	0	Output	Bit-Serial
Slave Serial	1	1	1	Input	Bit-Serial
Master Parallel up	1	0	0	Output	Byte-Wide,00000up
Master Parallel down	1	1	0	Output	Byte-Wide,3FFF down
Peripheral Synchr	0	1	1	Input	Byte-Wide
Peripheral Asynchr	1	0	1	Output	Byte-Wide
Reserved	0	1	0	-	-
Reserved	0	0	1	-	-

จากความต้องการสร้างให้ใช้กระแสไฟฟ้าต่ำจากลักษณะการต่อใช้งานทั้ง 5 แบบ จึงมีเพียง 2 แบบเท่านั้นที่เหมาะสม คือ แบบมาสเตอร์ซีเรียลและแบบสเลฟซีเรียล ส่วนแบบมาสเตอร์พาแรลต้องใช้ EPROM 27CXXX ซึ่งกินกระแสมากกว่า PROM XC17XXX เหมาะในการทดสอบต้นแบบก่อน เมื่อวงจรต้นแบบทำงานได้ถูกต้องแล้วจึงทำการอัดโปรแกรมลง PROM อีกทีหนึ่งเพราะว่าในแบบพาราแรลนั้น EPROM สามารถโปรแกรมได้ใหม่ต่างกับ PROM ที่โปรแกรมได้เพียงครั้งเดียว

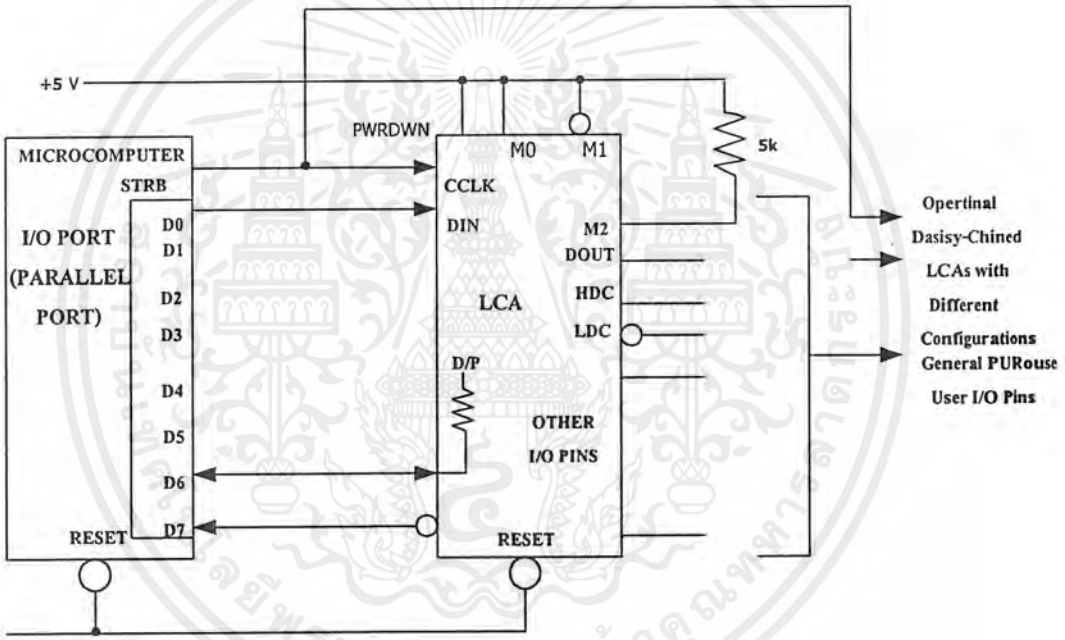
การใช้งาน FPGAs ในการต่อลักษณะสเลฟซีเรียลและมาสเตอร์ซีเรียล เมื่อเริ่มจ่ายไฟเข้าตัว FPGAs จะทำการลบข้อมูลหน่วยความจำที่ใช้ในคอนฟิก (Configuration Memory) ตรวจสอบลักษณะการต่อคอนฟิกว่าเป็นลักษณะใดในตารางที่ 2.3 ว่าเป็นแบบอนุกรมหรือขนาน หลังจากนั้นจะเริ่มทำการโปรแกรมคอนฟิกสัญญาณ DONE/PROGRAM เป็น '0' ซึ่งอยู่ในระหว่างโปรแกรมและเมื่อข้อมูลในคอนฟิก ที่รับมาจากภายนอกเต็มหน่วยความจำที่ใช้ในคอนฟิก และความยาวของข้อมูลตรงกับที่ส่วนหัวของข้อมูลคอนฟิก สัญญาณ DONE/PROGRAM จะเป็น '1' ซึ่งหมายถึงโปรแกรมทำการคอนฟิกเสร็จสิ้น ดูรูปที่ 2.14 ประกอบ



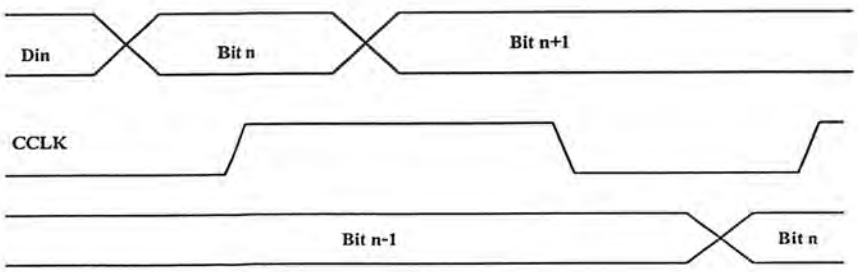
รูปที่ 2.14 ลำดับผังการไหลของข้อมูลในการคอนฟิกเมื่อเริ่มป้อนแหล่งจ่ายไฟเข้าไอซี และการโปรแกรมใหม่

2.9.1 การใช้งานในลักษณะสเลฟซีเรียล

การต่อใช้งานในลักษณะนี้ เหมาะสมกับวงจรที่ออกแบบมาเพื่อทำงานร่วมกับไมโครคอมพิวเตอร์อยู่แล้ว ทั้งนี้เพราะ FPGAs ได้ใช้ความสามารถของไมโครโปรเซสเซอร์ในการเก็บและส่งข้อมูลคอนฟิกให้เพียงแต่ต้องเขียนโปรแกรม เพื่อส่งโปรแกรมคอนฟิกให้เพิ่มลักษณะการต่อในลักษณะนี้เป็นดังรูปที่ 2.15 ซึ่งไมโครคอมพิวเตอร์จะสร้างสัญญาณเพื่อทำการคอนฟิกให้กับอุปกรณ์ FPGAs การป้อนโปรแกรมคอนฟิกให้ FPGAs ทำได้โดยต่อสัญญาณ Strobe เข้ากับขา CCLK และพอร์ต D0 เข้ากับขา DIN สร้างสัญญาณนาฬิกาป้อนที่ขา CCLK และป้อนโปรแกรมคอนฟิกแบบอนุกรมเข้าที่ขา DIN ดังแผนภูมิในรูปที่ 2.16



รูปที่ 2.15 การต่อใช้งานในลักษณะสเลฟซีเรียล



รูปที่ 2.16 แผนภูมิเวลาการป้อนข้อมูลโปรแกรมคอนฟิกในลักษณะสเลฟซีเรียล

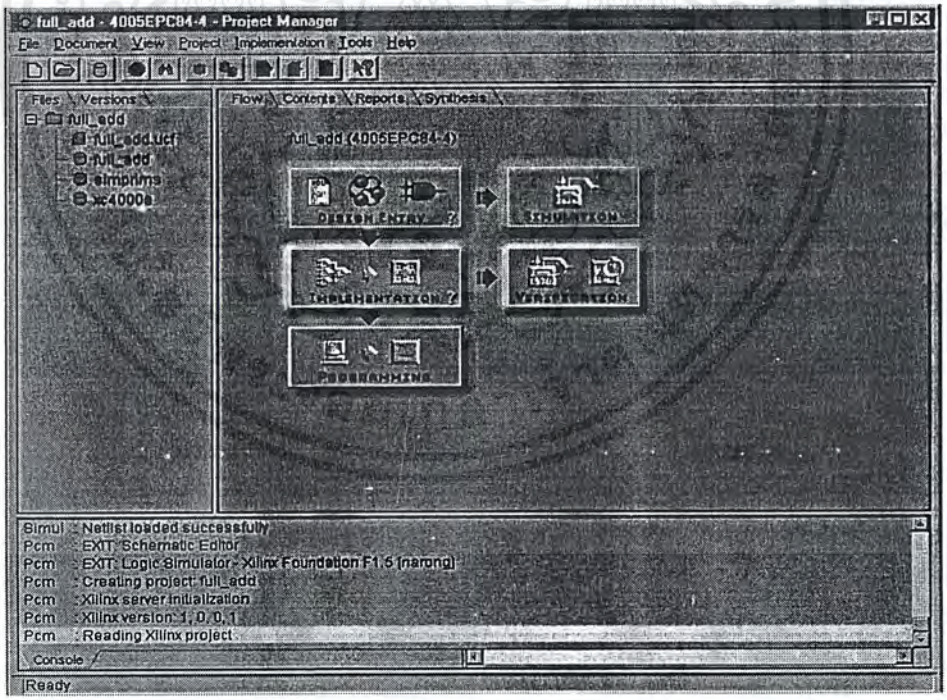
2.10 ข้อควรระวังในการใช้อุปกรณ์ FPGAs

สิ่งที่สำคัญ คือ อุปกรณ์ FPGAs ไวต่อความร้อนมาก การบัดกรีโดยหัวแร้งกำลังสูงหรือบัดกรีโดยจี้หัวแร้งที่ขาไอซีเป็นเวลานานจะทำให้ไอซีเสียหายได้ ระยะเวลาในการบัดกรีหนึ่งจุดไม่ควรเกิน 5-10 วินาที ควรใช้ซ็อกเกต (Socket) ไอซีในการประกอบวงจรลงแผ่นวงจรพิมพ์

การป้องกันไอซีจากแรงดันไฟฟ้าไม่ควรต่อสลับขั้วบวกกับลบ จะทำให้ไอซีเสียได้ นอกจากนั้นแรงดันของแหล่งจ่ายไฟต้องอยู่ในช่วงที่โรงงานกำหนดมา สำหรับ FPGAs ค่าที่ใช้งานอยู่ในช่วง Vcc = 4.75-5.25 V และแรงดันที่ทนได้อยู่ในช่วง –0.5 – 7 V ดังนั้นก่อนป้อนแรงดันควรเช็คให้แน่ใจก่อน

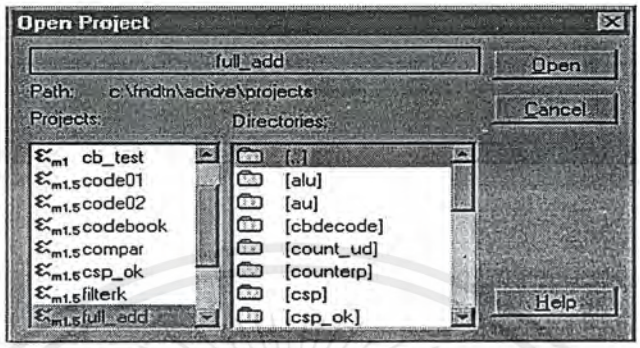
2.11 ขั้นตอนการออกแบบและสังเคราะห์โดยใช้ซอฟต์แวร์ของบริษัทไซลิงค์

ซอฟต์แวร์ของบริษัทไซลิงค์ (Xilinx) เวอร์ชัน 6.01 เป็นเวอร์ชันที่ทำงานได้ทั้งบน Windows 3.11 และ Windows 95 เมื่อเริ่มต้นใช้โปรแกรมที่หน้าจอจะปรากฏดังรูปที่ 2.17



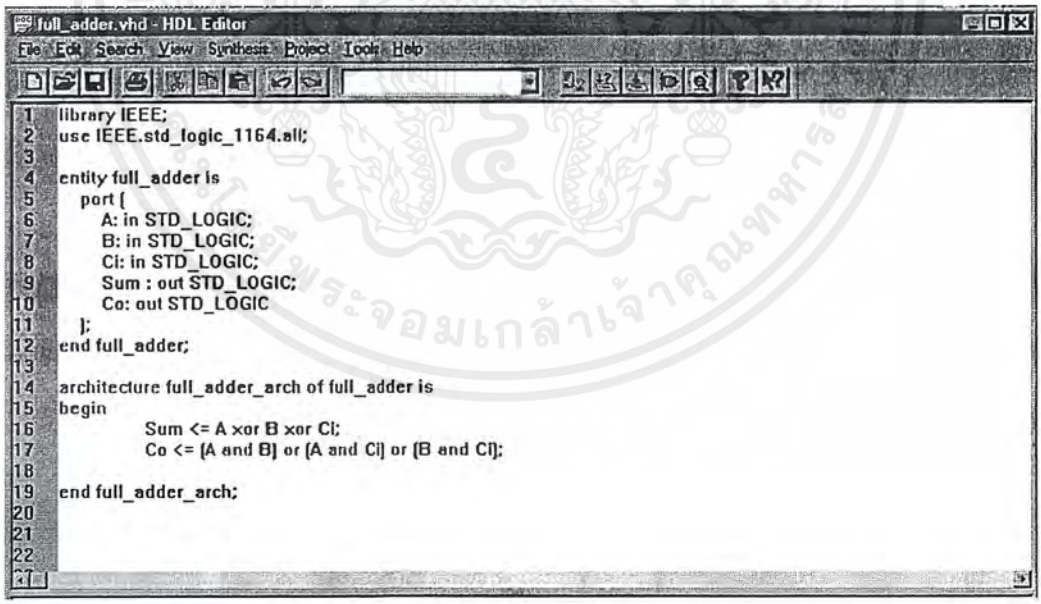
รูปที่ 2.17 หน้าต่าง Project Manager

คลิกเมาส์ที่เมนู File แล้วเลือก New Project เพื่อสร้าง Project ใหม่ขึ้นมาหรือเลือกที่ Open Project เพื่อเรียกใช้โปรเจ็ค Full_add ที่ได้เขียนไว้แล้ว จะแสดงหน้าต่างดังรูปที่ 2.18



รูปที่ 2.18 หน้าต่าง Open Project

ดับเบิ้ลคลิกที่โปรเจ็ค Full_add หลังจากนั้นคลิกเมาส์ที่ HDL Entity เพื่อที่จะนำไฟล์ Full_add.vhd ที่ได้เขียนไว้แล้วมาทำการสังเคราะห์ จะปรากฏโปรแกรมที่เขียนไว้ดังรูปที่ 2.19



รูปที่ 2.19 หน้าต่าง Full_add-HDL Editor

จากนั้นทำการสังเคราะห์ (Synthesis) โปรแกรม โดยคลิกเมาส์ที่เมนู Synthesis และเลือกที่ Synthesis ถ้าหากการ Synthesis ไม่มีข้อผิดพลาดจะปรากฏหน้าต่างดังรูปที่ 2.20

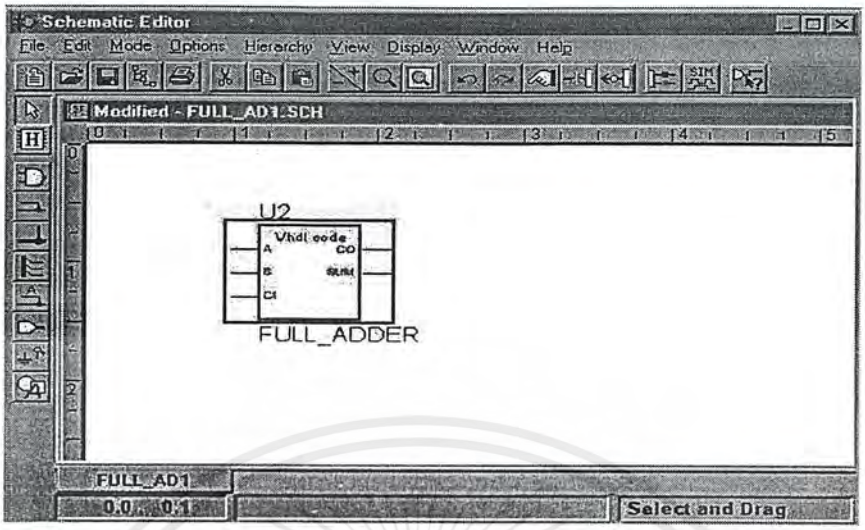
รูปที่ 2.20 หน้าต่าง HDL Editor หลังจาก Synthesis

ถ้าหากเกิดการผิดพลาดให้กลับไปแก้ไขที่โปรแกรม จากนั้นคลิกเมาส์ไปที่ Project เลือกที่ Create Macro เพื่อทำการสร้าง Symbol ขึ้นมา ถ้าไม่มีข้อผิดพลาดจะปรากฏหน้าต่างดังรูป 2.21

รูปที่ 2.21 หน้าต่าง HDL Editor หลังจาก Create Macro

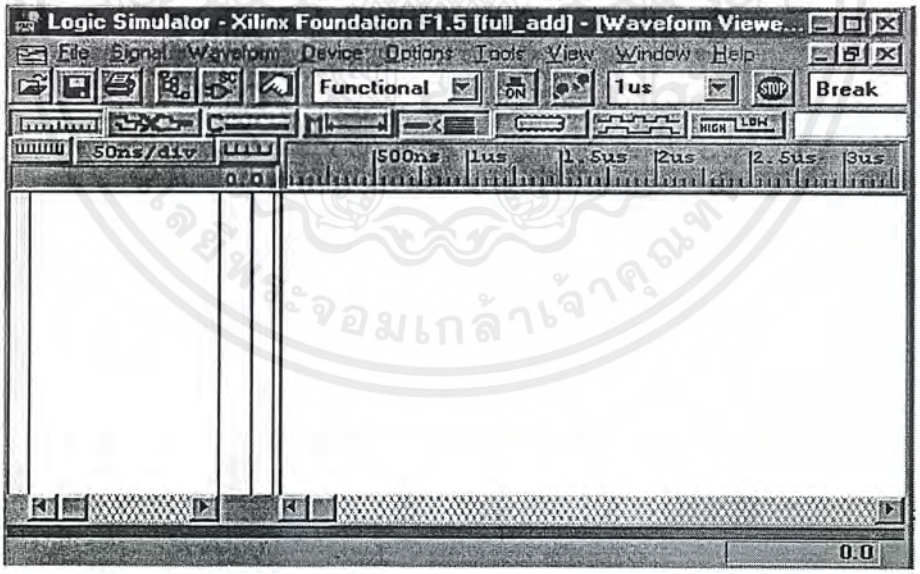
2.12 ขั้นตอนการจำลองการทำงานโดยใช้ซอฟต์แวร์ของบริษัทไซลิงค์ (Xilinx)

กลับไปที่หน้าต่าง Project Manager แล้วคลิกเมาส์ที่ Schematic Editor จะแสดงหน้าต่างของ Schematic ดังรูปที่ 2.22



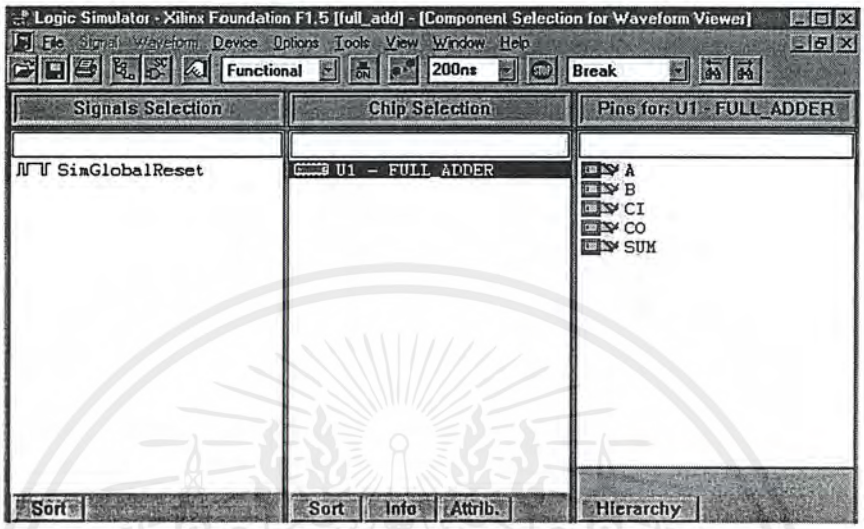
รูปที่ 2.22 หน้าต่าง Schematic Editor

จากนั้นคลิกเมาส์ที่ Simulate เพื่อทำการ Simulate สัญญาณ ซึ่งจะแสดงหน้าต่างดังรูปที่ 2.23



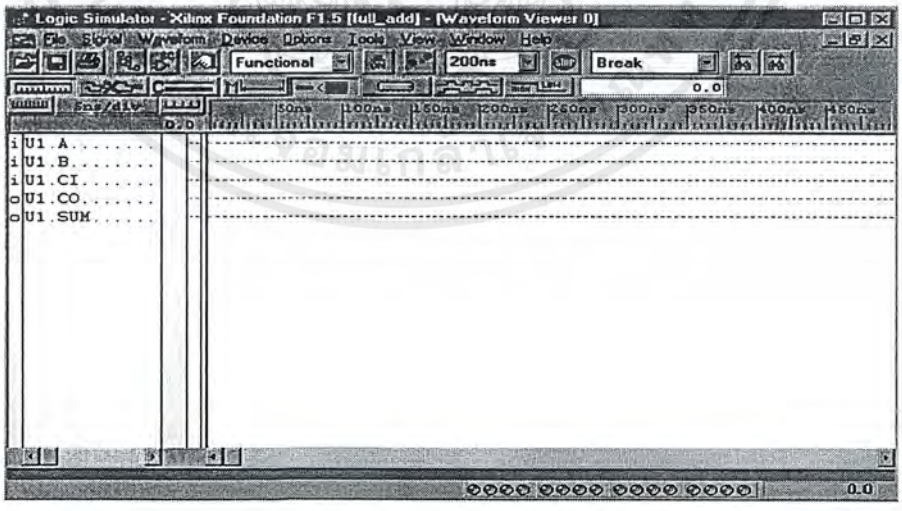
รูปที่ 2.23 หน้าต่าง Logic Simulator

จากนั้นคลิกเมาส์ที่เมนู Signal เลือก Add Signals เพื่อใช้ในการเลือกสัญญาณที่เราต้องการให้แสดงผล ซึ่งแสดงหน้าต่างดังรูปที่ 2.24



รูปที่ 2.24 หน้าต่าง Component Selector-From Waveform Viewer

จากนั้นเลือกสัญญาณที่ต้องการนำไปแสดงผล ซึ่งจะแสดงสัญญาณที่ต้องการบนหน้าต่าง Logic Simulator ดังรูปที่ 2.25



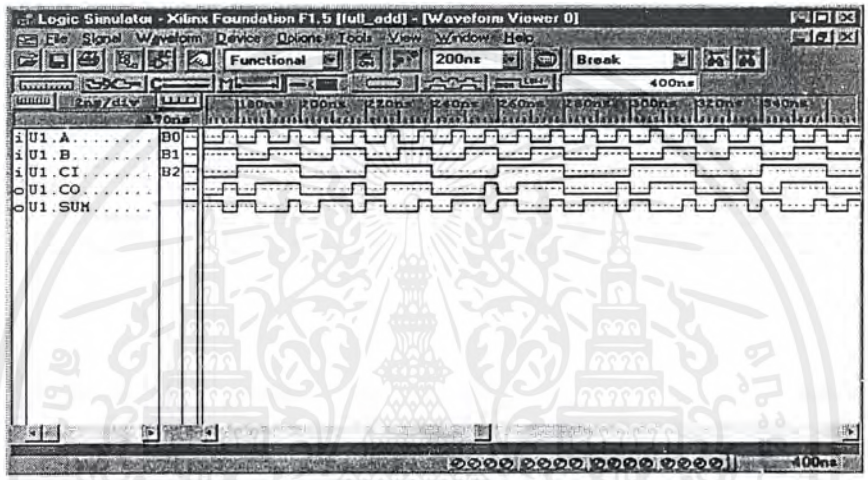
รูปที่ 2.25 หน้าต่าง Logic Simulator

ทำการกำหนดสัญญาณโดยคลิกเมาส์ที่เมนู Simulator แล้วเลือกที่ Add Simulator โดยเลือกให้สัญญาณ A = B0

สัญญาณ B = B1

สัญญาณ CI = B2

จากนั้นทำการ Run โดยคลิกเมาส์ที่ปุ่ม Step จะแสดงหน้าต่างดังรูปที่ 2.26

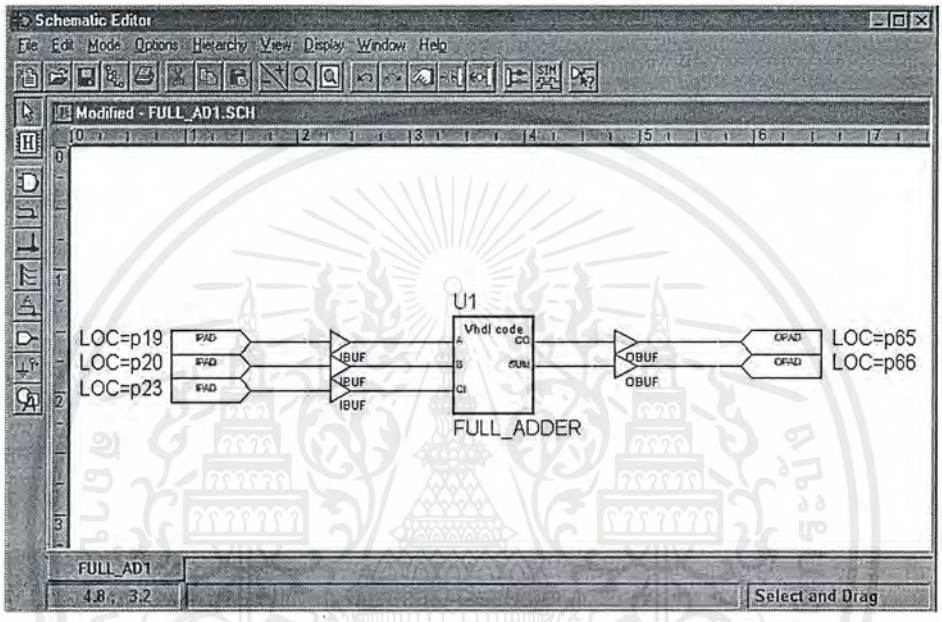


รูปที่ 2.26 ผลการ Run

เมื่อได้ผลการ Run ออกมาแล้วให้ทำการตรวจสอบผลการ Run โปรแกรมว่าถูกต้องหรือไม่ ถ้าหากผลการทดลองไม่ถูกต้อง ต้องกลับไปแก้ไขที่ Source Program ใหม่ แล้วทำการสังเคราะห์โปรแกรมใหม่ แล้วจึงเริ่มขั้นตอนการจำลองการทำงานอีกครั้ง

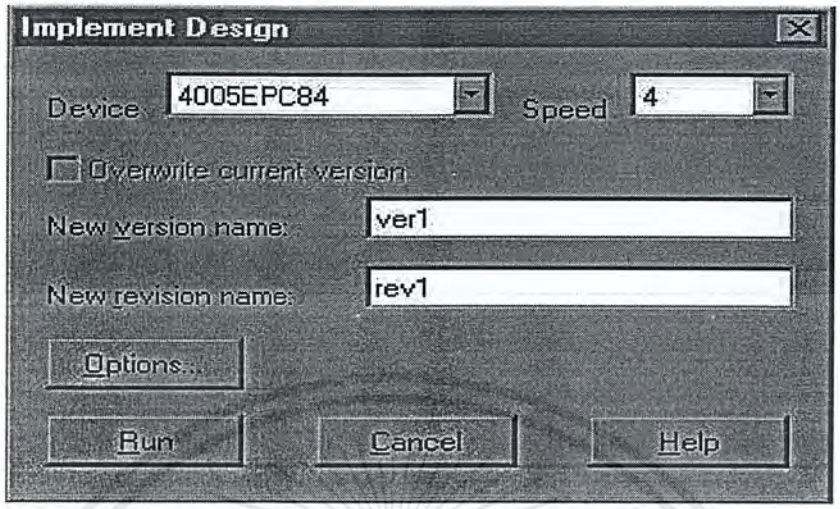
2.13 ขั้นตอนการโปรแกรมลงบนอุปกรณ์ FPGAs โดยใช้ซอฟต์แวร์ XDM ของบริษัทไซลิงค์ (Xilinx)

กลับไปที่หน้าต่าง Schematic Editor แล้วทำการใส่ Buffer ทั้งด้านอินพุตและเอาต์พุตและทำการใส่ตำแหน่งขาที่สัมพันธ์กับ FPGAs ดังรูปที่ 2.27



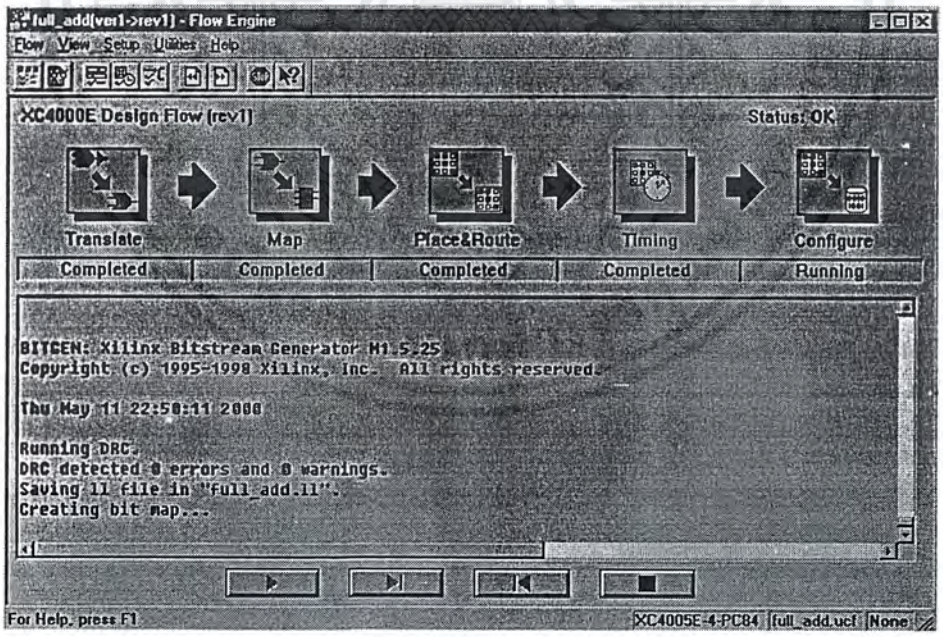
รูปที่ 2.27 หน้าต่าง Schematic Editor ภายหลังจากใส่ตำแหน่งขา

จากนั้นกลับไปที่หน้าต่าง Project Manager คลิกเมาส์ไปที่ Implementation จะมีหน้าต่างขึ้นมาบอกว่าเป็นเวอร์ชันที่เท่าไร ถ้ามีการแก้ไขเวอร์ชันจะเพิ่มมากขึ้น จากนั้นคลิกเมาส์ไปที่ Run ดังรูปที่ 2.28



รูปที่ 2.28 หน้าต่าง Implement Design

หลังจากนั้นจะปรากฏหน้าต่างการ Implementation ดังรูป2.29



รูปที่ 2.29 หน้าต่างการ Implementation

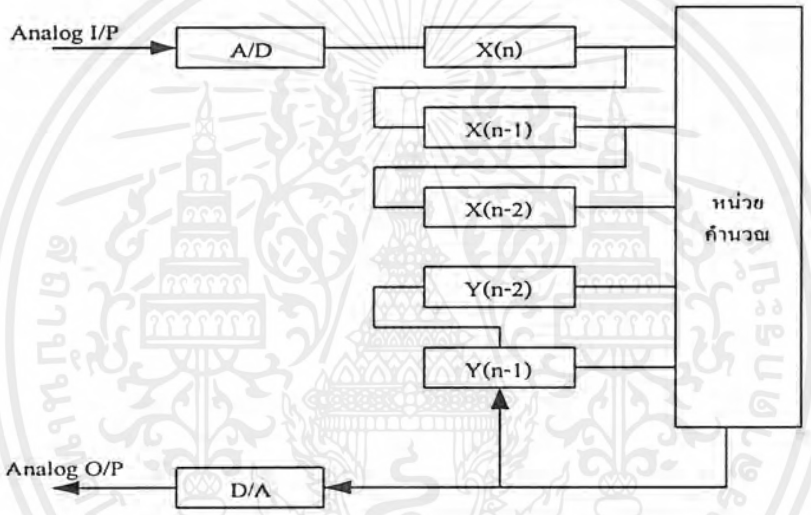
เมื่อเสร็จสิ้นกระบวนการ Implementation แล้วจะได้ไฟล์ที่มีนามสกุล .Bit เพื่อดาวน์โหลดลงบนบอร์ดตัวอย่างของ FPGAs โดยในการดาวน์โหลดไฟล์บิตสตรีมลงบนบอร์ดตัวอย่างของ FPGAs นั้น ให้คลิกเมาส์ที่เมนู Tools และเลือกที่ Hardware Debugger ซึ่งแสดงหน้าต่าง Communication Setup

เมื่อทำการกำหนดค่า Baud Rate และเลือก Port ที่ส่งข้อมูลออกไปแล้วให้คลิกที่ OK เมื่อเสร็จแล้วให้ไปที่หน้าต่าง Hardware Debugger คลิกที่เมนู Download และเลือกที่ Download Design ซึ่งโปรแกรม Debugger จะทำการ Download Design ไฟล์ Fulladd.Bit ลงบนบอร์ดตัวอย่างของ FPGAs ผ่านทางสายดาวน์โหลดเคเบิล เมื่อดาวน์โหลดเสร็จเรียบร้อยแล้ว จึงนำลงบนบอร์ดตัวอย่างของ FPGAs ไปทดสอบการทำงานของวงจร Fulladder

บทที่ 3
การออกแบบ การสร้าง และการทำงาน

3.1 ความคิดเบื้องต้นในการสร้างดิจิตอลฟิลเตอร์

ดิจิตอลฟิลเตอร์สามารถสร้างได้ทั้งแบบฮาร์ดแวร์เพียงอย่างเดียวหรือสร้างจากฮาร์ดแวร์ร่วมกับซอฟต์แวร์หรือสร้างจากซอฟต์แวร์เพียงอย่างเดียวได้ ซึ่งทั้งสามแบบมีโครงสร้างดังรูป 3.1



รูปที่ 3.1 ผังการทำงานของ Second-Order Digital Filter

สัญญาณแอนะลอกอินพุตจะถูกแปลงเป็นสัญญาณดิจิตอลโดย A/D (Analog to Digital Converter) สัญญาณดิจิตอลที่ได้จะถูกชิฟท์แบบขนานเข้าไปในชิฟท์รีจิสเตอร์ X(n) โดยค่าเริ่มต้นของ X(n), X(n-1), X(n-2), Y(n-1), Y(n-2) ควรมีค่าเป็นศูนย์และโน้มเข้าสู่ค่าจริงๆ ของมัน ในเวลาต่อมา บิตขวาสุดของ X(n), X(n-1), X(n-2), Y(n-1), Y(n-2) จะเป็นอินพุตของหน่วยคำนวณ หน่วยคำนวณก็จะคำนวณค่า Y(n) ได้จากสมการ (3.1)

$$Y(n) = a_0X(n) + a_1X(n-1) + a_2X(n-2) - b_1Y(n-1) - b_2Y(n-2) \qquad (3.1)$$

ค่าสัมประสิทธิ์ a_0, a_1, a_2, $-b_1$, $-b_2$ จะถูกเก็บไว้ในหน่วยความจำในรูปของเลขฐานสอง พร้อมกันดังนี้

X(n) ก็จะถูกชิฟท์ออกทางขวา 1 บิต เข้าสู่ X(n-1)

X(n-1) ก็จะถูกชิฟท์ออกทางขวา 1 บิต เข้าสู่ X(n-2)

X(n-2) ก็จะถูกชิฟท์ออกทางขวาทิ้ง 1 บิต

เป็นผลให้บิตขวาสุดของชิฟท์รีจิสเตอร์ถูกเลื่อนมาอีก 1 ตำแหน่ง ตำแหน่งใหม่ที่เลื่อนมาจะเป็นอินพุตเข้าสู่หน่วยคำนวณต่อไป ทำเช่นนี้เรื่อยไปจนครบทุกบิตของชิฟท์รีจิสเตอร์ เมื่อครบแล้วหน่วยความจำก็จะให้ค่าเอาต์พุต Y(n) ออกมา Y(n) ที่คำนวณได้จะถูกแปลงเป็นสัญญาณแอนะลอก โดยวงจร D/A (Digital to Analog converter) และในขณะเดียวกัน Y(n) ที่ได้พร้อมกับ X(n) ค่าใหม่ก็จะถูกชิฟท์แบบขนานเข้าไปในชิฟท์รีจิสเตอร์ X(n) กับ Y(n) พร้อมกัน จากนั้นก็เริ่มทำการคำนวณค่า Y(n) ค่าใหม่ต่อไปด้วยวิธีการดังกล่าว

3.2 ดิจิตอลฟิลเตอร์แบบเลขคณิตแจกแจง

3.2.1 การสร้าง Distributed Arithmetic Digital Filter

จากสมการ (3.1) Second-Order Digital Filter

$$Y(n)=a_0X(n)+a_1X(n-1)+a_2X(n-2)-b_1Y(n-1)-b_2Y(n-2)$$

จะเห็นว่าในการคำนวณ Y(n) จะต้องทำการคูณและบวก

จากที่ได้ทราบกันแล้วว่า การคูณในคอมพิวเตอร์นี้ใช้เวลามาก ทำให้การทำงานไม่สามารถเป็นเวลาจริง (Real time) ได้ ซึ่งโครงสร้างแบบเลขคณิตแจกแจง (Distributed Arithmetic Structure) หรือโครงสร้างแบบ Rom/Acc (ROM-Accumulator) สามารถเปลี่ยนการคูณเป็นการบวก (Adding) และการเลื่อน (Shifting) โดยนำหน่วยความจำประเภท ROM หรือ RAM มาประยุกต์ใช้

3.2.2 หลักการของโครงสร้างแบบเลขคณิตแจกแจง

พิจารณาตัวกรองอันดับสองที่มีฟังก์ชันถ่ายโอนเป็น

$$H_{(z)} = \frac{a_0 + a_{1(z-1)} + a_{2(z-2)}}{1 + b_{1(z-1)} + b_{2(z-2)}} \tag{3.2}$$

เขียนแบบสมการผลต่างสืบเนื่องได้คือ

$$Y(n) = a_0X(n) + a_1X(n-1) + a_2X(n-2) - b_1Y(n-1) - b_2Y(n-2)$$

โดยให้ X(n), X(n-1) และ X(n-2) เป็นลำดับสัญญาณเข้า

Y(n), Y(n-1) และ Y(n-2) เป็นลำดับสัญญาณออก

a_0, a_1, a_2, $-b_1$ และ $-b_2$ เป็นค่าสัมประสิทธิ์ของตัวกรอง

วิธีการของโครงสร้างเลขคณิตแจกแจงทำโดยการเขียนแทนลำดับสัญญาณเข้าและลำดับสัญญาณออกด้วยตัวเลขแบบส่วนเติมเต็มสอง (2's Complement) ที่มีจำนวนบิตรวมทั้งเครื่องหมายด้วยเป็น B บิต หรือเขียนกระจายเป็นเลขฐานสองคือ

$$X(n) = X0(n)X1(n)X2(n)...Xb-1(n) \tag{3.3}$$

$$Y(n) = Y0(n)Y1(n)Y2(n)...Yb-1(n) \tag{3.4}$$

โดยที่ X0(n) และ Y0(n) เป็นบิตที่แสดงเครื่องหมายของตัวเลข ส่วน X1(n) และ Y1(n) [I=1,2,...B-1] เป็นบิตที่ i ของลำดับสัญญาณ และมีค่าเป็น 0 หรือ 1 เท่านั้น

จากสมการ (3.3) อาจเขียนรวมได้ในรูปของ 2's Complement คือ

$$X(n) = X_0(n) + \sum_{i=1}^{B-1} X(n)2^{-i} \tag{3.5}$$

$$Y(n) = -Y_0(n) + \sum_{i=1}^{B-1} Y(n)2^{-i} \tag{3.6}$$

เมื่อแทนสมการ (3.5) และ (3.6) ลงในสมการผลต่างสืบเนื่อง

$$Y(n)=a_0X(n)+a_1X(n-1)+a_2X(n-2)-b_1Y(n-1)-b_2Y(n-2)$$

จะได้

$$Y(n)=a_0\left[-X_0(n)+\sum_{i=1}^{B-1}X(n)2^{-i}\right]+a_1\left[-X_0(n-1)+\sum_{i=1}^{B-1}Xi(n-1)2^{-i}\right]$$
$$+a_2\left[-X_0(n-2)+\sum_{i=1}^{B-1}Xi(n-2)2^{-i}\right]-b_1\left[-Y_0(n-1)+\sum_{i=1}^{B-1}Yi(n-1)2^{-i}\right]$$
$$-b_2\left[-Y_0(n-2)+\sum_{i=1}^{B-1}Yi(n-2)2^{-i}\right] \qquad (3.7)$$

เมื่อทำการจัดพจน์ใหม่โดยเอาบิตที่สมนัยกันมาเขียนรวมกันจะได้

$$Y(n)=-\left[a_0x_0(n)+a_1x_0(n-1)+a_2x_0(n-2)-b_1y_0(n-1)-b_2y_0(n-2)\right]$$
$$+\sum_{i=1}^{B-1}2^{-i}\left[a_0x_i(n)+a_1x_i(n-1)+a_2x_i(n-2)-b_1y_i(n-1)-b_2y_i(n-2)\right] \qquad (3.8)$$

ให้ $$a_0x_i(n)+a_1x_i(n-1)+a_2x_i(n-2)-b_1y_i(n-1)-b_2y_i(n-2)=F_i\{.\} \qquad (3.9)$$

จะได้ $$Y(n)=\sum_{i=1}^{B-1}2^{-i}F_i\{.\}-F_0\{.\} \qquad (3.10)$$

ผลจากสมการ (3.9) และ (3.10) จะได้โครงสร้างแบบเลขคณิตแจกแจง โดยนำค่าฟังก์ชันทั้งหมดของ Fi{.} มาทำเป็นตารางเปิดดู (Look up Table) โดยค่าในตารางเปิดดูคิดคำนวณจากสมการ (3.9) และเนื่องจาก Fi{.} มีตัวแปรอยู่ 5 ตัว คือ Xi(n), Xi(n-1), Xi(n-2), Yi(n-1), Yi(n-2) ทำให้ตารางเปิดดูมีค่าฟังก์ชันของ Fi{.} อยู่ 2^5 ค่า ค่าของฟังก์ชัน Fi{.} ทั้ง 32 ค่านี้จะถูกคำนวณแล้วทำการปัดเศษให้เหลือ B บิต แล้วเก็บไว้ในรอมหรือแรม เพื่อนำไปสร้างตัวกรองต่อไป

3.3 การสร้างตารางเปิดดูจากฟังก์ชัน Fi{.} สำหรับตัวกรองอันดับสอง

เมื่อได้ผลต่างสืบเนื่องอันดับสองแล้ว นำค่าสัมประสิทธิ์คูณกับบิตที่ 1 ของข้อมูลแต่ละตัว แล้วบวกกันเก็บไว้ที่แอดเดรสที่ชี้โดยค่า Xi(n), Xi(n-1), Xi(n-2), Yi(n-1), Yi(n-2) ดังแสดงในตารางที่ 3.1

ตารางที่ 3.1 การแปลงค่าสัมประสิทธิ์

ตำแหน่งที่	แอดเดรสของรอม					ค่าของ Fi{.} ภายในรอม
	Xi(n)	Xi(n-1)	Xi(n-2)	Yi(n-1)	Yi(n-2)	
0	0	0	0	0	0	0
1	0	0	0	0	1	$-b_2$
2	0	0	0	1	0	$-b_1$
3	0	0	0	1	1	$-b_1-b_2$
4	0	0	1	0	0	a_2
5	0	0	1	0	1	a_2-b_2
6	0	0	1	1	0	a_2-b_1
7	0	0	1	1	1	$a_2-b_1-b_2$
8	0	1	0	0	0	a_1
9	0	1	0	0	1	a_1-b_2
10	0	1	0	1	0	a_1-b_1
11	0	1	0	1	1	$a_1-b_1-b_2$
12	0	1	1	0	0	a_1+b_2
13	0	1	1	0	1	$a_1+a_2-b_2$
16	1	0	0	0	0	a_0
17	1	0	0	0	1	$a_0- b_2$
18	1	0	0	1	0	$a_0- b_1$
19	1	0	0	1	1	$a_0-b_1-b_2$
20	1	0	1	0	0	a_0+a_2

ตารางที่ 3.1 (ต่อ) การแปลงค่าสัมประสิทธิ์

ตำแหน่งที่	แอดเดรสของรอม					ค่าของ Fi{.} ภายในรอม
	Xi(n)	Xi(n-1)	Xi(n-2)	Yi(n-1)	Yi(n-2)	
21	1	0	1	0	1	$a_0+a_2-b_2$
22	1	0	1	1	0	$a_0+a_2-b_1$
23	1	0	1	1	1	$a_0+a_2-b_1-b_2$
24	1	1	0	0	0	a_0+a_1
25	1	1	0	0	1	$a_0+a_1-b_2$
26	1	1	0	1	0	$a_0+a_1-b_1$
27	1	1	0	1	1	$a_0+a_2-b_1-b_2$
28	1	1	1	0	0	$a_0+a_1-a_2$
29	1	1	1	0	1	$a_0+a_1+a_2-b_2$
30	1	1	1	1	0	$a_0+a_1+a_2-b_1$
31	1	1	1	1	1	$a_0+a_1+a_2-b_1-b_2$

3.4 การคำนวณหา Y(n) ในกรณีตัวกรองอันดับสอง

จากสมการ (3.10)

$$Y(n)=\sum_{i=1}^{n-1}2^{-i}Fi\{.\}-F_0\{.\}$$

จะเห็นว่าในการคำนวณหา Y(n) จะต้องทำการบวกค่าฟังก์ชัน Fi{.} และพร้อมกับการเลื่อน B-1 ครั้ง (B=จำนวนบิต) แล้วทำการลบอีก 1 ครั้ง รวมแล้วทำการคำนวณหาค่าจะต้องกระทำ B ครั้ง

แต่จากการทดลองนั้นได้ทำการกรองอันดับหก โดยใช้ตัวกรองอันดับสอง 3 ชุดมาคาสเคด (Cascade) กัน ดังนั้นจึงมีการคำนวณเอาต์พุต 3 ขั้นตอน เป็นไปตามสมการ

$$V(n)=a01X(n)+a11X(n-1)+a21X(n-2)-b11V(n-1)-b21V(n-2) \quad (3.11)$$

$$W(n)=a02V(n)+a12V(n-1)+a22V(n-2)-b12W(n-1)-b22W(n-2) \quad (3.12)$$

$$Y(n)=a03W(n)+a13W(n-1)+a23W(n-2)-b13Y(n-1)-b23Y(n-2) \quad (3.13)$$

โดย a01, a11, a21, b11, b21 เป็นค่าสัมประสิทธิ์ของตัวกรองในขั้นตอนที่ 1

a02, a12, a22, b12, b22 เป็นค่าสัมประสิทธิ์ของตัวกรองในขั้นตอนที่ 2

a03, a13, a23, b13, b23 เป็นค่าสัมประสิทธิ์ของตัวกรองในขั้นตอนที่ 3

ในส่วนของตารางเมื่อทำการเปิดดูจะมี 3 ตาราง โดยค่าฟังก์ชัน Fi{.} ที่เก็บในตารางที่ 3 เป็นดังนี้

TABLE1	Fi1 { X(n), X(n-1), X(n-2), V(n-1), V(n-2) }
TABLE2	Fi2 { V(n), V(n-1), V(n-2), W(n-1), W(n-2) }
TABLE3	Fi3 { W(n), W(n-1), W(n-2), Y(n-1), Y(n-2) }

3.5 การคำนวณหา Y(n) ของตัวกรองอันดับที่หก

$$V(n)=\sum_{i=1}^{B-1}2^{-i}Fi_1\{.\}-F_{01}\{.\} \quad (3.14)$$

$$W(n)=\sum_{i=1}^{B-1}2^{-i}Fi_1\{.\}-F_{02}\{.\} \quad (3.15)$$

$$Y(n)=\sum_{i=1}^{B-1}2^{-i}Fi_3\{.\}-F_{03}\{.\} \quad (3.16)$$

โดย Fi1{.} = Fi1 { Xi(n), Xi(n-1), Xi(n-2), Vi(n-1), Vi(n-2) }

Fo1{.} = Fo1 { Xo(n), Xo(n-1), Xo(n-2), Vo(n-1), Vo(n-2) }

Fi2{.} = Fi2 { Vi(n), Vi(n-1), Vi(n-2), Wi(n-1), Wi(n-2) }

Fo2{.} = Fo2 { Vo(n), V(n-1), Vo(n-2), Wo(n-1), Wo(n-2) }

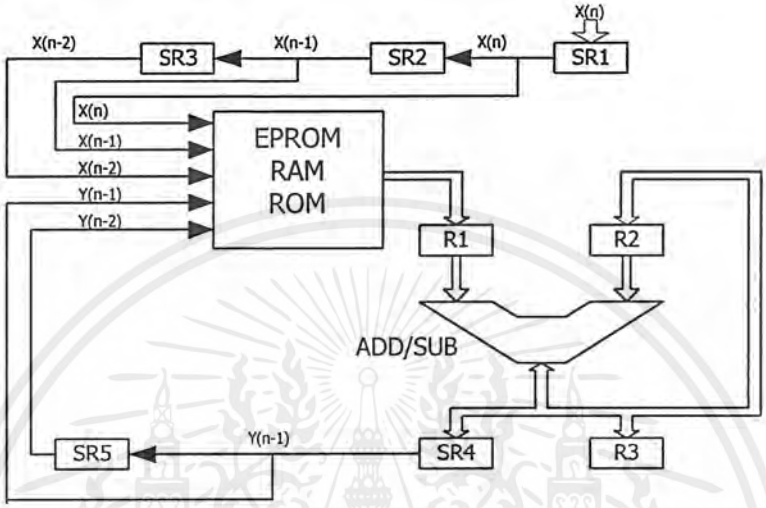
Fi3{.} = Fi3 { Wi(n), Wi(n-1), Wi(n-2), Yi(n-1), Yi(n-2) }

Fo3{.} = Fo3 { Wo(n), Wo(n-1), Wo(n-2), Yo(n-1), Yo(n-2) }

และทำการคำนวณเช่นเดียวกับในกรณีตัวกรองอันดับสอง แต่ทำวิธีการซ้ำเดิม 3 ครั้ง

โครงสร้างของตัวกรองสัญญาณแบบป้อนกลับเชิงเลขอันดับสองโดยใช้สมการ (3.14), (3.15) และ (3.16) นำมาสร้างเป็นวงจรทางอิเล็กทรอนิกส์ หรือฮาร์ดแวร์ดังแสดงในรูปที่ 3.2 SR1 และ SR4 เป็นรีจิสเตอร์ที่เลื่อนข้อมูลขนาด L บิต เข้าแบบขนานออกแบบอนุกรม (Parallel in Serial

out) SR2, SR3 และ SR5 เป็นรีจิสเตอร์เลื่อนข้อมูลเข้าออกแบบอนุกรม (Serial in Serial out) R1, R2, R3 เป็นรีจิสเตอร์เก็บข้อมูลแบบชั่วคราว (ข้อมูลเข้าออกแบบขนาน) ส่วน ADD/SUB เป็นวงจรบวกเลขแบบส่วนเติมเต็มสอง มีขั้นตอนการทำงานดังนี้



รูปที่ 3.2 โครงสร้างของตัวกรองป้อนกลับเชิงเลขอันดับที่สอง

1) ทำการลบข้อมูลภายในรีจิสเตอร์ R1, R2 และ R3 จากนั้นทำการโหลดข้อมูล X(n) เข้าไปเก็บในรีจิสเตอร์ SR1

2) เอาต์พุตของรีจิสเตอร์ SR1, SR2, SR3, SR4, SR5 คือลำดับสัญญาณ X(n), X(n-1), X(n-2), Y(n-1), Y(n-2) ตามลำดับ ถูกใช้เป็นแอดเดรสของ ROM เพื่อหาค่าของ Fi เมื่อ I=L (หมายถึงบิตที่ L) นำค่าของ Fi ที่ได้เก็บไว้ในรีจิสเตอร์ R1 จากนั้นนำค่าใน R1 บวกกับค่าใน R2 ด้วยวงจรบวก ADD/SUB ผลลัพธ์ที่ได้ถูกเก็บไว้ใน R2 พร้อมกับเลื่อนค่าหรือข้อมูลใน R2 ไปทางขวา 1 บิต

3) ทำการเลื่อนข้อมูลในรีจิสเตอร์ SR1, SR2, SR3, SR4, SR5 ไป 1 บิต เพื่อกำหนดค่าแอดเดรสของ ROM ใหม่ ได้ค่าของ Fi เมื่อ I=บิตที่ L-1 นำค่าของ Fi ที่เป็นไว้ในรีจิสเตอร์ R1 จากนั้นนำค่าใน R1 บวกกับค่าใน R2 ด้วยวงจร ADD/SUB ผลลัพธ์ที่ได้ถูกเก็บไว้ใน R2 พร้อมกับเลื่อนค่าหรือข้อมูลใน R2 ไปทางขวา 1 บิต

4) ในทำนองเดียวกันกระทำซ้ำตามขั้นตอนที่ 3 สำหรับ I=บิตที่ L-2, L-3,...,1 ตามลำดับ

5) ขั้นตอนนี้จะแตกต่างกับขั้นตอนที่ 3 และ 4 คือเมื่อเลื่อนข้อมูลในรีจิสเตอร์ SR1, SR2, SR3, SR4, SR5 ต่อจากขั้นตอนที่ 4 ไปอีก 1 บิต สำหรับหาค่า Fo และนำค่าไปเก็บไว้ใน R1 จากนั้น

นำค่าหรือข้อมูลใน R1 ไปลบออกจากค่าใน R2 ผลลัพธ์ที่ได้คือลำดับสัญญาณ Y(n) และถูกเก็บไว้ในรีจิสเตอร์ R3 เพื่อรอการแปลงเป็นสัญญาณเชิงอุปมานต่อไป

6) กระทำซ้ำตามขั้นตอน 1-5 ใหม่อีกเรื่อยๆ สำหรับหาค่า Y(n) ในลำดับถัดไป

3.6 การออกแบบสร้างวงจรกรองสัญญาณเชิงเลขอันดับ 6

เพื่อเป็นการประหยัดฮาร์ดแวร์ และลดความยุ่งยากของวงจรและง่ายต่อการสร้างอาศัยโครงสร้างของตัวกรองสัญญาณเชิงเลขอันดับสองดังที่กล่าวมาแล้ว มาต่อ Cascade กัน 3 วงจร ใช้ภาคคำนวณหรือประมวลผลชนิด D/A เพียงตัวเดียวสลับใช้งาน ดังแสดงในรูปที่ 3.3 เพื่อให้ง่ายจะแยกอธิบายการทำงานเป็น 3 ขั้นตอนดังนี้

1) ขั้นตอนที่ 1

ตำแหน่งของ MUX1, MUX2 อยู่ในตำแหน่ง A โดยมีวงจร A/D, RX0, RX1, RX2, RV01, RV02, ROM1 ประกอบกันเป็นโครงสร้างตัวกรองเชิงเลขอันดับ 2 ส่วนแรก โดยใช้ภาคประมวลผล D/A คือวงจรบวก ลบ (ADD/SUB) และรีจิสเตอร์ ACC (Accumulator) ร่วมกัน สัญญาณที่ใช้ควบคุมคือ SC, LRX0, CKX0, A/S, CLACC, CK และ M1 มีขั้นตอนทำงานดังนี้

1.1) A/D (Analog to Digital Converter) ถูกควบคุมด้วยสัญญาณ SC ทำการแปลงสัญญาณได้จากการสุ่มให้เป็นลำดับสัญญาณเชิงเลข X(n) ขนาด 8 บิต

1.2) สัญญาณควบคุม LRX0 ทำหน้าที่โหลดลำดับสัญญาณหรือข้อมูล X(n) เข้าไปเก็บไว้ในรีจิสเตอร์ RX0

1.3) สัญญาณนาฬิกา CKX0 จะทำการเลื่อนข้อมูล X(n) ในรีจิสเตอร์ คือ RX0, RX1, RX2, RV01, RV02 ไปครั้งละ 1 บิต เอาต์พุตของแต่ละรีจิสเตอร์คือ X(n), X(n-1), X(n-2), V0(n-1), V0(n-2) ที่ถูกเลื่อนแต่ละครั้งจะเป็นแอดเดรสของ ROM1 (ROM1 จะถูกเลือกด้วยสัญญาณ M1) เอาต์พุตของ ROM1 จะถูกส่งไปบวกกับค่าที่อยู่ใน ACC ด้วย

1.4) ADD/SUB (ควบคุมการบวกด้วยสัญญาณ A/S) ผลลัพธ์ที่ได้จะถูกโหลดเข้าไปเก็บไว้ใน ACC ด้วยสัญญาณ CK

1.5) CKX0 จะเลื่อนข้อมูล X(n) ไปอีก 1 บิต แล้วกระทำซ้ำข้อ 1.4 จนกระทั่ง CKX0 เลื่อนข้อมูลไปถึงบิตที่ 8 จึงนำค่าที่ได้จากเอาต์พุตของ ROM1 ไปลบออกจากค่าที่อยู่ใน ACC (ที่ได้จากการบวกของการเลื่อนข้อมูล 7 ครั้ง) ผลลัพธ์ที่ได้จะถูกโหลดเข้าไปในรีจิสเตอร์ RV0 สิ้นสุดการทำงานขั้นตอนที่ 1

2) ขั้นตอนที่ 2

ตำแหน่งของ MUX1 อยู่ในตำแหน่ง B ส่วน MUX2 อยู่ตำแหน่งเดิมคือ A โดยรีจิสเตอร์ RV0, RV1, RV2, RV11, RV12, ROM2 ประกอบกันเป็นโครงสร้างของตัวกรองเชิงเลขอันดับที่ 2 ส่วนที่สองใช้ภาคประมวลแบบ D/A ร่วมกันเหมือนขั้นตอนที่ 1 จะเห็นได้ว่าขั้นนี้ คือ ประมวลผลของตัวกรองในอันดับที่ 4 นั่นเอง ส่วนสัญญาณควบคุมคือ LRV0, CKV0, M2, A/S, CK, CLACC, LRV1 เหมือนกับข้อ 1.3-1.5 โดยเปลี่ยนรีจิสเตอร์และสัญญาณควบคุมตามลำดับดังนี้

รีจิสเตอร์ RX0, RX1, RX2, RV01, RV02 $\Longrightarrow$ RV0, RV01, RV02, RV11, RV12

แอดเดรส X(n), X(n-1), X(n-2), V0(n-1), V0(n-2)

$\Longrightarrow$ V0(n), V0(n-1), V0(n-2), V1(n-1), V1(n-2)

สัญญาณควบคุม M1, LRX0, CKX0, LRV0 $\Longrightarrow$ M2, LRV0, CKV0, LRV1

3) ขั้นตอนที่ 3

ตำแหน่ง MUX1 อยู่ในตำแหน่ง A (เพื่อรอทำงานในลำดับสัญญาณ X(n) ใหม่) MUX2 อยู่ในตำแหน่ง B โดยมี RV1, RV11, RV12, RY1, RY2, ROM3 ประกอบกันเป็นโครงสร้างตัวกรองอันดับสองส่วนที่ 3 ขั้นตอนนี้เป็นการประมวลผลของตัวกรองอันดับที่ 6 นั่นเอง สัญญาณควบคุมคือ LRV1, CKY, M3, A_S, CLACC, CK, LRY ขั้นตอนการทำงานเหมือนกับข้อ 1.3-1.5 โดยเปลี่ยนรีจิสเตอร์ และสัญญาณควบคุมตามลำดับดังนี้

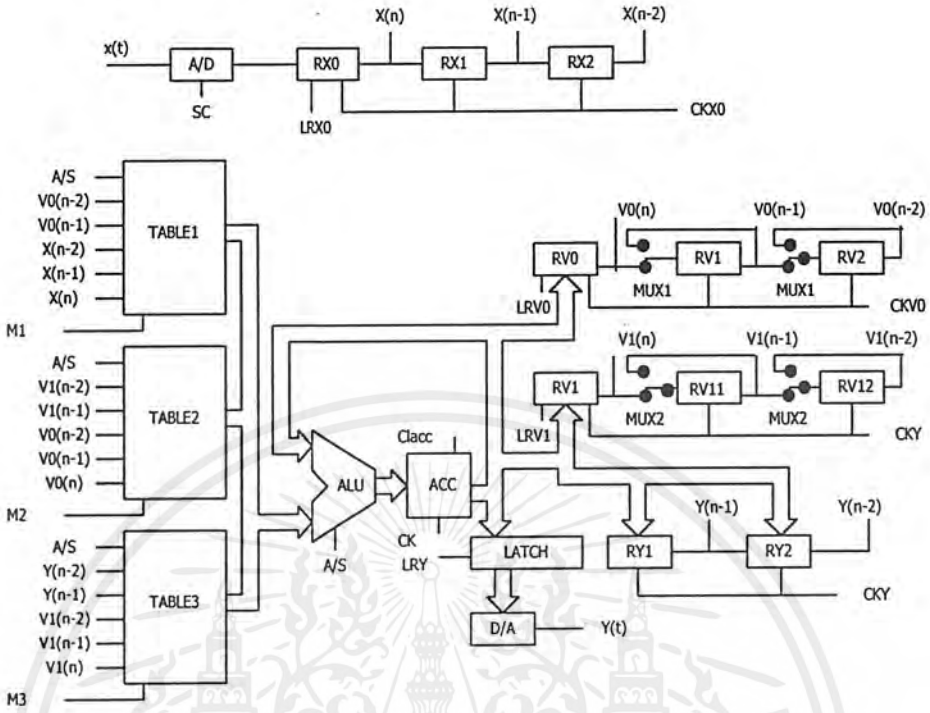
รีจิสเตอร์ RX0, RX1, RX2, RV01, RV02 $\Longrightarrow$ RV1, RV11, RV12, RY1, RY2

แอดเดรส X(n), X(n-1), X(n-2), V0(n-1), V0(n-2)

$\Longrightarrow$ V1(n), V1(n-1), V1(n-2), Y(n-1), Y(n-2)

สัญญาณควบคุม M1, LRX0, CKX0, LRV0 $\Longrightarrow$ M3, LRV1, CKY, LRY

ขั้นตอนนี้จะแตกต่างกับขั้นตอนที่ 2,3 คือ เมื่อ CKY เลื่อนข้อมูลในรีจิสเตอร์มาถึงบิตที่ 8 แล้ว สัญญาณ LRY จะทำการโหลดผลลัพธ์เก็บไว้ใน Buffer เพื่อทำการแปลงสัญญาณเชิงเลขเป็นสัญญาณเชิงอุปมานด้วยวงจร D/A จากนั้นจะวนกลับไปทำงานซ้ำในขั้นตอน 1,2,3 ตามลำดับ จึงจะได้ตัวกรองสัญญาณเชิงเลขอันดับ 6 ดังแสดงในรูปที่ 3.3



รูปที่ 3.3 โครงสร้างการทำงานของวงจรกรองสัญญาณเชิงเลขอันดับที่ 6

3.7 การออกแบบวงจรกรองสัญญาณเชิงเลขด้วยภาษา VHDL

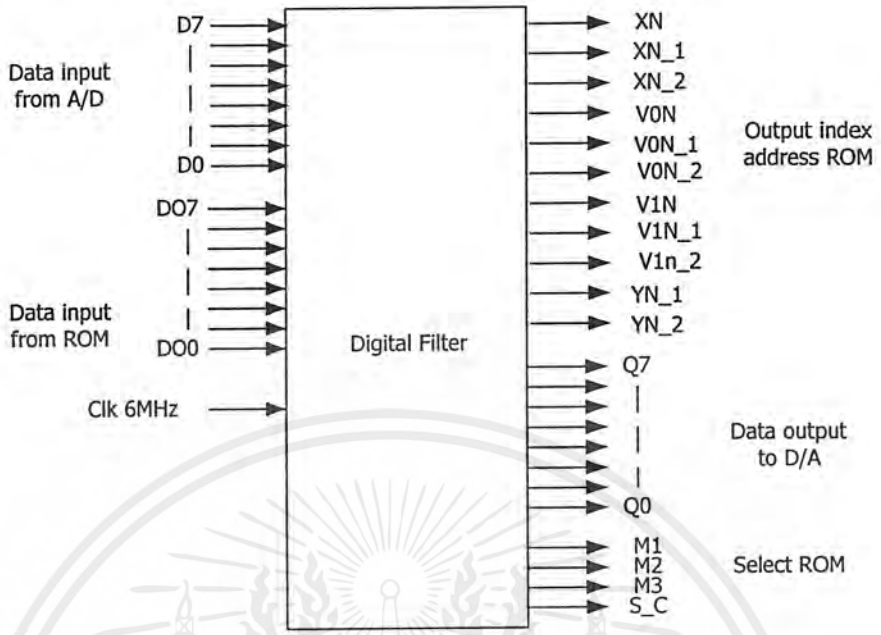
จากโครงสร้างการทำงานของวงจรกรองสัญญาณเชิงเลขอันดับที่ 6 เราสามารถที่จะใช้ภาษาVHDL ออกแบบโดยการแบ่งผังการทำงานออกเป็น 3 ส่วนคือ

1) ส่วนการควบคุม (Control Unit)

2) ส่วนการประมวลผล (ALU Unit)

3) ส่วนการเลื่อนข้อมูล (Shift Data Unit)

ซึ่งขั้นตอนการใช้ภาษา VHDL ออกแบบนั้นเริ่มจากการกำหนดผังการทำงานรวม แล้วกำหนดสัญญาณอินพุตและเอาต์พุตที่ต้องการดังรูปที่ 3.4



รูปที่ 3.4 ผังการทำงานรวมของวงจรกรองสัญญาณเชิงเลขอันดับที่ 6

จากรูปที่ 3.4 ผังการทำงานรวมการทำงานของวงจรกรองสัญญาณเชิงเลข ประกอบด้วย

1) ข้อมูลอินพุตจาก A/D ซึ่งมีขนาด 8 บิต โดยบิต D7 เป็นบิตที่มีนัยสำคัญสูงสุด

2) ข้อมูลอินพุตค่าสัมประสิทธิ์ของวงจรกรองสัญญาณได้จาก EPROM ซึ่งได้ทำการบันทึกไว้มีขนาด 8 บิตเช่นเดียวกัน โดยบิต D7 เป็นบิตที่มีนัยสำคัญสูงสุด

3) สัญญาณนาฬิกา (Clock) ขนาด 6 MHz

4) สัญญาณชี้ตำแหน่งค่าสัมประสิทธิ์ของตัวกรองสัญญาณเชิงเลข ประกอบด้วยสัญญาณ XN, XN_1, XN_2, V0N, V0N_1, VN_2, V1N, V1N_1, V1N_2, YN_1, และ YN_2

5) ข้อมูลเอาต์พุตเป็นสัญญาณเชิงเลข เพื่อนำไปแปลงเป็นสัญญาณเชิงอุปมาน โดยใช้วงจร D/A

6) สัญญาณ M1, M2 และ M3 เพื่อใช้ในการเลือก EPROM ตัวที่ต้องการ

7) สัญญาณ S_C เป็นสัญญาณกำหนดความถี่ในการ Sampling ให้กับวงจร A/D

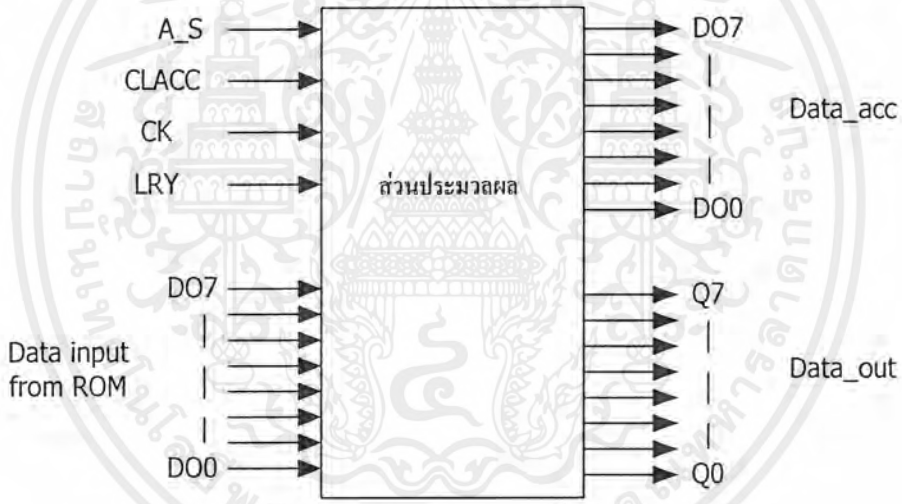
3.8 ขั้นตอนการออกแบบผังการทำงานภายใน

จากที่กล่าวมาแล้วในหัวข้อที่ 3.9 เราสามารถแบ่งผังการทำงานออกเป็น 4 ส่วน ซึ่งมีอยู่ 3 ส่วนที่อยู่ภายในผังการทำงานรวมของวงจรกรองสัญญาณเชิงเลขคือ ส่วนการควบคุม (Control Unit), ส่วนการประมวลผล (Process Unit), และการเลื่อนข้อมูล (Shift Data Unit) สำหรับส่วนของตารางค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขนั้นถูกต่อไว้ภายนอก

ดังนั้นในการออกแบบด้วยภาษา VHDL จะทำการออกแบบเพียง 3 ส่วนที่อยู่ในผังการทำงานรวม ดังนี้

3.8.1 ส่วนประมวลผล

ทำหน้าที่ในการรับข้อมูลจากตารางค่าสัมประสิทธิ์มาทำการประมวลผลโดยใช้วิธีการบวกสะสม และจะเป็นการลบกันเมื่อสัญญาณ A_S มีค่าเป็น '1' ดังแสดงในรูปที่ 3.5



รูปที่ 3.5 ผังการทำงานของส่วนประมวลผล

จากรูปที่ 3.5 ผังการทำงานของส่วนประมวลผล ประกอบด้วยสัญญาณต่างๆ ดังนี้

1) สัญญาณ A_S เป็นสัญญาณที่ใช้กำหนดการบวกหรือลบข้อมูล

2) สัญญาณ CK เป็นสัญญาณนาฬิกาของส่วนประมวลผล

3) ข้อมูลจาก ROM เป็นค่าสัมประสิทธิ์ของวงจรกรองเชิงเลขที่เก็บใน EPROM

4) ข้อมูล Data_acc เป็นข้อมูลผลลัพธ์ของการบวกกันภายในส่วนประมวลผล

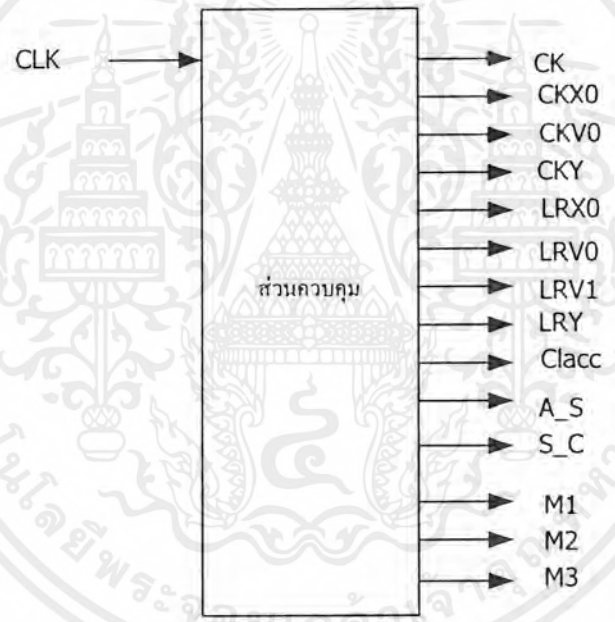
5) ข้อมูล Data_out เป็นข้อมูลที่ส่งออกจากเอาต์พุต โดยจะใช้สัญญาณ LRY เป็นสัญญาณนาฬิกาที่จะกำหนดการส่งข้อมูล

6) สัญญาณ Clacc เป็นสัญญาณที่ใช้เคลียร์ข้อมูลในส่วนประมวลผล

ลักษณะการทำงานของส่วนประมวลผลคือ จะทำการรับข้อมูลค่าสัมประสิทธิ์ของวงจรตัวกรองสัญญาณเชิงเลขมาจาก EPROM แล้วจะทำการบวกหรือลบกับค่าข้อมูลที่อยู่ภายในส่วนประมวลผล ซึ่งขึ้นอยู่กับว่าสัญญาณ A_S จะเป็น '1' หรือ '0' โดยการทำงานจะใช้ CK ควบคุมการทำงาน ข้อมูลที่ได้ถูกเก็บค่าไว้ภายในและถูกส่งออกมาทาง Data_acc และตรวจสอบค่า LRY ถ้ามีค่าเป็น '0' ก็จะส่งข้อมูลออกมาที่ Data_out

3.8.2 ส่วนควบคุม

ทำหน้าที่ให้กำเนิดสัญญาณควบคุมการทำงานทั้งหมดของวงจรกรองสัญญาณเชิงเลข ซึ่งเราสามารถกำหนดผังการทำงานดังรูปที่ 3.6 เพื่อใช้ภาษา VHDL ในการบรรยายการทำงานต่อไป



รูปที่ 3.6 ผังการทำงานของส่วนควบคุม

จากรูปที่ 3.6 ผังการทำงานของส่วนควบคุม โดยดูจากรูปที่ 3.3 ประกอบ มีสัญญาณของส่วนต่างๆ ดังนี้

1) สัญญาณ CLK เป็นอินพุตของวงจร โดยรับโดยตรงจากภายนอกมีค่าความถี่ 6 MHz

2) สัญญาณ CK เป็นสัญญาณนาฬิกาสำหรับส่วนประมวลผลทำการคำนวณทุก ๆ ขอบขาขึ้น

3) สัญญาณ CKX0 เป็นสัญญาณในการเลื่อนข้อมูลของรีจิสเตอร์ RX0, RX1, RX2, RV1 และ RV2 ทำงานเป็นตัวกรองเชิงเลขอันดับ 2 ส่วนแรก

4) สัญญาณ CKV0 เป็นสัญญาณในการเลื่อนข้อมูลของรีจิสเตอร์ RV0, RV1, RV2, RV11 และ RV12 ทำงานเป็นตัวกรองเชิงเลขอันดับที่ 4 ส่วนที่สอง

5) สัญญาณ CKY เป็นสัญญาณในการเลื่อนข้อมูลของรีจิสเตอร์ RV1, RV11, RV12, RY1 และ RY2 ทำงานเป็นตัวกรองเชิงเลขอันดับที่ 6 ส่วนที่สาม

6) สัญญาณ LRX0 เป็นสัญญาณในการโหลดข้อมูลจากแอคคิวมูเลเตอร์สู่รีจิสเตอร์ RX0 ในวงจรเลื่อนข้อมูล ซึ่งจะทำงานที่ลอจิก '0'

7) สัญญาณ LRV0 เป็นสัญญาณในการโหลดข้อมูลจากแอคคิวมูเลเตอร์สู่รีจิสเตอร์ RV0 ในวงจรเลื่อนข้อมูล ซึ่งจะทำงานที่ลอจิก '0'

8) สัญญาณ LRV1 เป็นสัญญาณในการโหลดข้อมูลจากแอคคิวมูเลเตอร์สู่รีจิสเตอร์ RV1 ในวงจรเลื่อนข้อมูล ซึ่งจะทำงานที่ลอจิก '0'

9) สัญญาณ LRY เป็นสัญญาณในการโหลดข้อมูลจาก LATCH สู่วงจร D/A ซึ่งจะทำงานที่ลอจิก '0'

10) สัญญาณ A_S เป็นสัญญาณที่ใช้กำหนดการบวกลบ ในภาค ALU โดยถ้า A_S เป็น '1' ที่ส่วนประมวลผลจะเป็นการลบข้อมูล ถ้าเป็น '0' จะเป็นการบวกกันของข้อมูล

11) สัญญาณ S_C เป็นสัญญาณ Sampling ที่จะส่งออกไปให้ A/D

12) สัญญาณ M1 ควบคุมการอ่านข้อมูลของ EPROM ตัวที่หนึ่ง

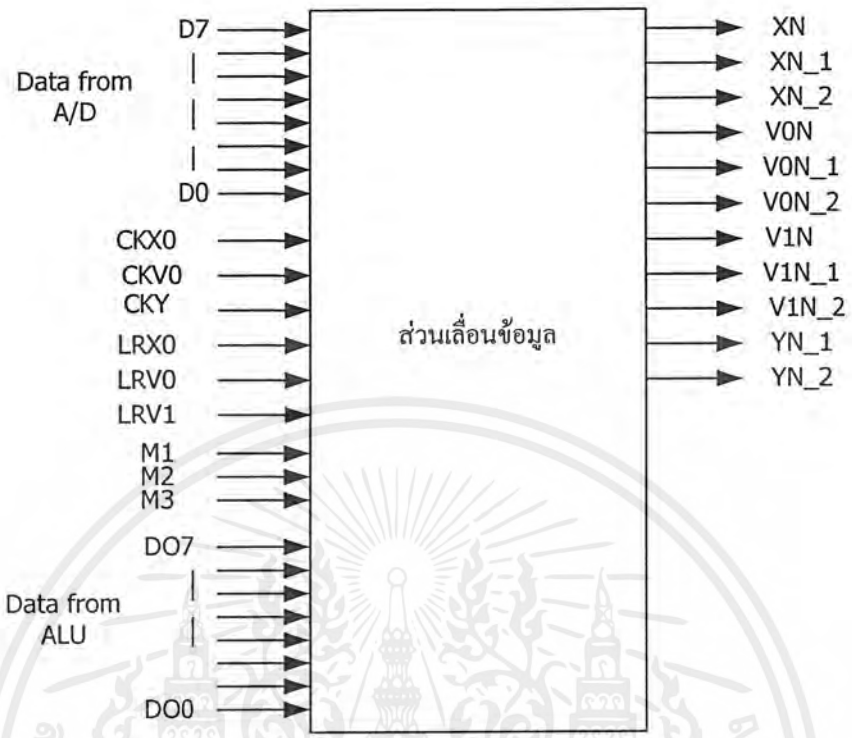
13) สัญญาณ M2 ควบคุมการอ่านข้อมูลของ EPROM ตัวที่สอง

14) สัญญาณ M3 ควบคุมการอ่านข้อมูลของ EPROM ตัวที่สาม

จากนั้นจึงทำการเขียนโปรแกรม VHDL ได้ดังโปรแกรมที่ 2 ในภาคผนวก และทำการจำลองการทำงานส่วนควบคุม ได้ผลการจำลองการทำงานดังรูปที่ 4.1 ในบทที่ 4

3.8.3 ส่วนเลื่อนข้อมูล

ทำหน้าที่ในการเลื่อนข้อมูล ซึ่งการทำงานภายในจะประกอบด้วยชุดเลื่อนข้อมูลเข้าแบบขนาน ออกแบบอนุกรม และชุดเลื่อนข้อมูลเข้าแบบอนุกรม ออกแบบอนุกรม ซึ่งผลจากการเลื่อนข้อมูลอนุกรมนั้นจะถูกส่งออกมาเพื่อนำไปชี้ตำแหน่งของ EPROM เพื่ออ่านค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขซึ่งสามารถกำหนดผังการทำงานได้ดังรูปที่ 3.7



รูปที่ 3.7 ผังการทำงานของส่วนเลื่อนข้อมูล

จากรูปที่ 3.7 จะประกอบด้วยสัญญาณต่างๆ ที่ใช้ในการเลื่อนข้อมูลดังนี้

1) Data Input จาก D/A ขนาด 8 บิต

2) สัญญาณนาฬิกา CKX0, CKV0 และ CKY ซึ่งเป็นสัญญาณที่ใช้ควบคุมการเลื่อนข้อมูลภายในทั้งแบบเข้าขนานออกอนุกรม และแบบเข้าอนุกรมออกอนุกรม

3) สัญญาณที่ใช้ในการโหลดข้อมูลประกอบด้วย LRX0, LRV0, LRV1 และ LRY ซึ่งจะใช้สำหรับโหลดข้อมูลจาก Data Input หรือจาก Data_acc

4) สัญญาณควบคุม M1, M2 และ M3 เป็นสัญญาณควบคุมการทำงานของการเลื่อนข้อมูลว่าจะให้ทำการเลื่อนข้อมูลชุดใด ซึ่งภายในจะมีชุดการเลื่อนข้อมูลอยู่ 3 ชุดใหญ่ๆ

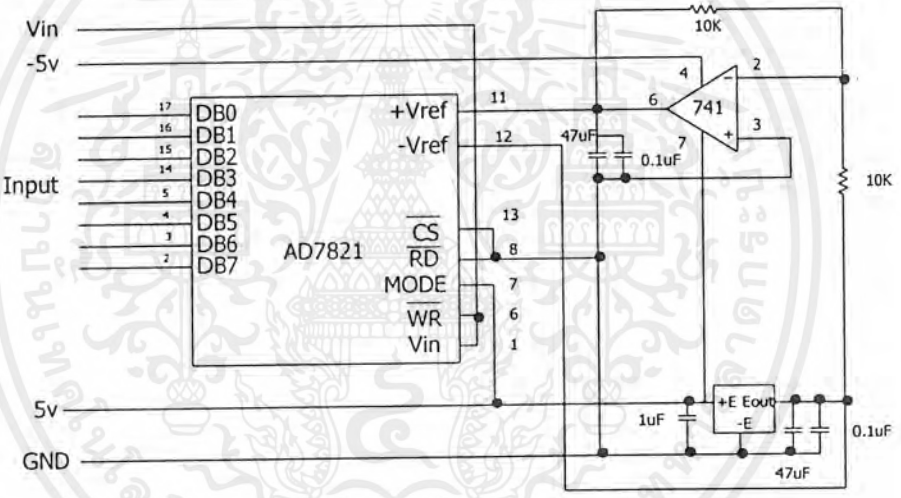
5) สัญญาณที่ใช้ชี้ตำแหน่งค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลข ซึ่งจะถูกส่งไปยัง EPROM ประกอบด้วยสัญญาณ XN, XN_1, XN_2, V0N, V0N_1, V0N_2, V1N, V1N_1, V1N_2, YN_1 และ YN_2

3.9 การออกแบบวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 ด้วยอุปกรณ์ FPGAs

วงจรที่ใช้ในการทดสอบวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 เป็นแบบบอร์ดตัวอย่างของ FPGAs ซึ่งจะทำการโปรแกรมผ่านทางดาวน์โหลดเคเบิล (Download Cable) โดยวงจรที่ใช้ทดสอบวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 แบ่งการทำงานออกเป็น 4 ส่วนด้วยกัน คือ

3.9.1 ส่วนอินพุต

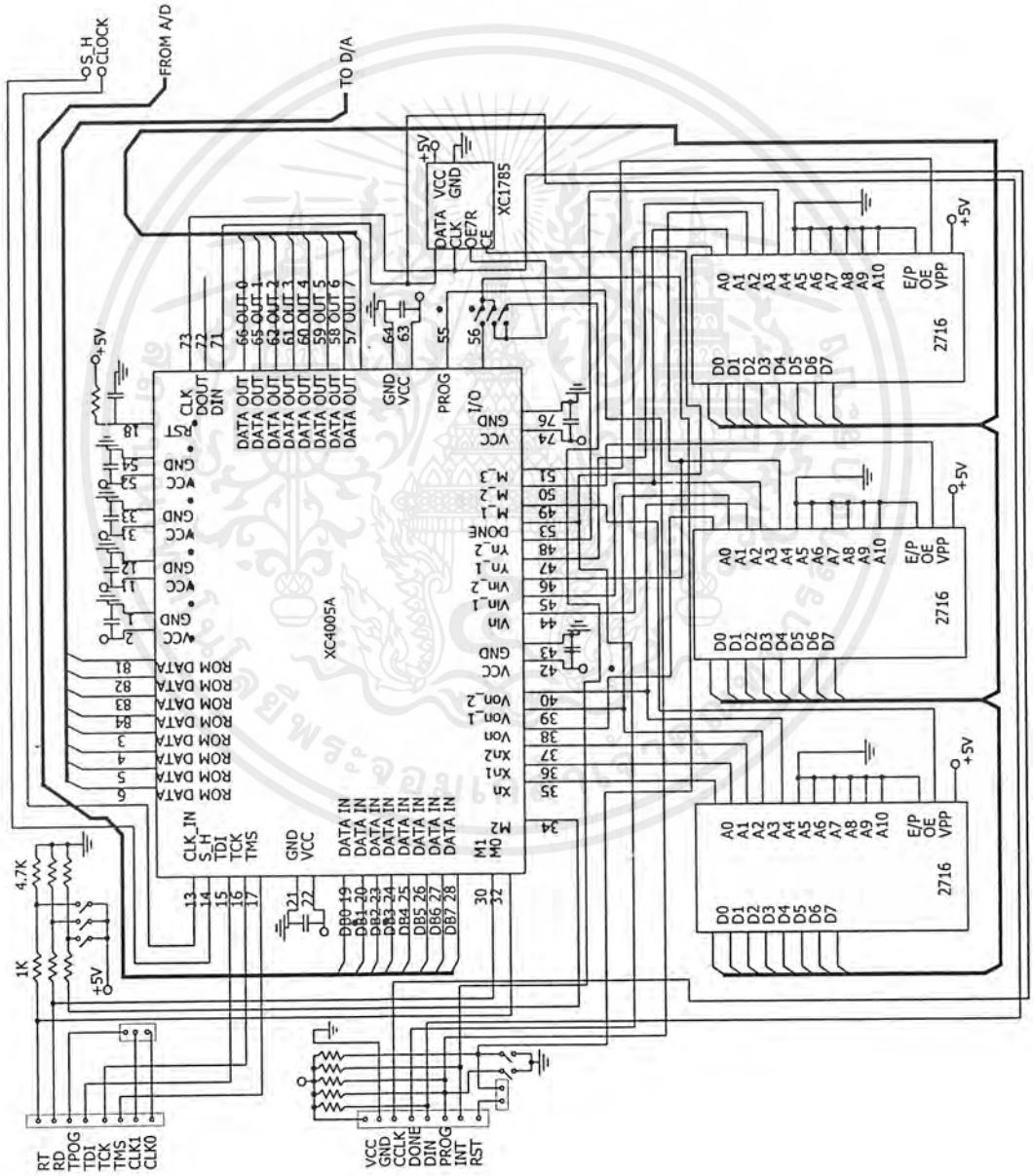
คือภาควงจร A/D ซึ่งจะรับอินพุตที่เป็นสัญญาณเชิงอุปมานเข้ามา แล้วทำการแปลงให้เป็นสัญญาณเชิงเลขขนาด 8 บิต ซึ่งจะเป็นอินพุตให้กับอุปกรณ์ FPGAs ซึ่งออกแบบไว้ดังรูปที่ 3.8



รูปที่ 3.8 การออกแบบส่วนอินพุต

3.9.2 ส่วนประมวลผล

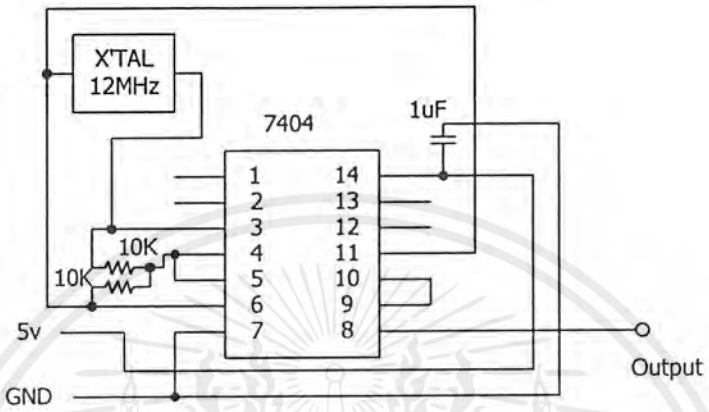
คือชุดอุปกรณ์ FPGAs ซึ่งเป็นการออกแบบการต่อใช้งาน FPGAs เบอร์ XC4005E รวมกับค่าตารางสัมประสิทธิ์ของวงจรที่เก็บไว้ใน EPROM ด้วย ซึ่งในส่วนนี้จะเป็นการนำสัญญาณเชิงเลขที่รับมาจากอินพุตมาทำการประมวลผลและส่งออกเอาต์พุตไปให้กับภาคเอาต์พุตต่อไป ซึ่งทำการออกแบบได้ดังรูป 3.9



รูปที่ 3.9 การออกแบบในส่วนประมวลผล

3.9.3 ส่วนสร้างสัญญาณนาฬิกา

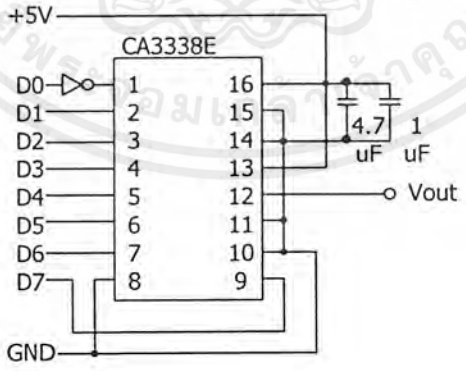
เราสามารถสร้างสัญญาณนาฬิกาได้โดยการใช้คริสตัลโมดูลในการกำเนิดสัญญาณนาฬิกาความถี่ 12 MHz แล้วทำการหารความถี่ลงเหลือ 6 MHz ดังแสดงในรูปที่ 3.10



รูปที่ 3.10 การออกแบบส่วนสร้างสัญญาณนาฬิกา

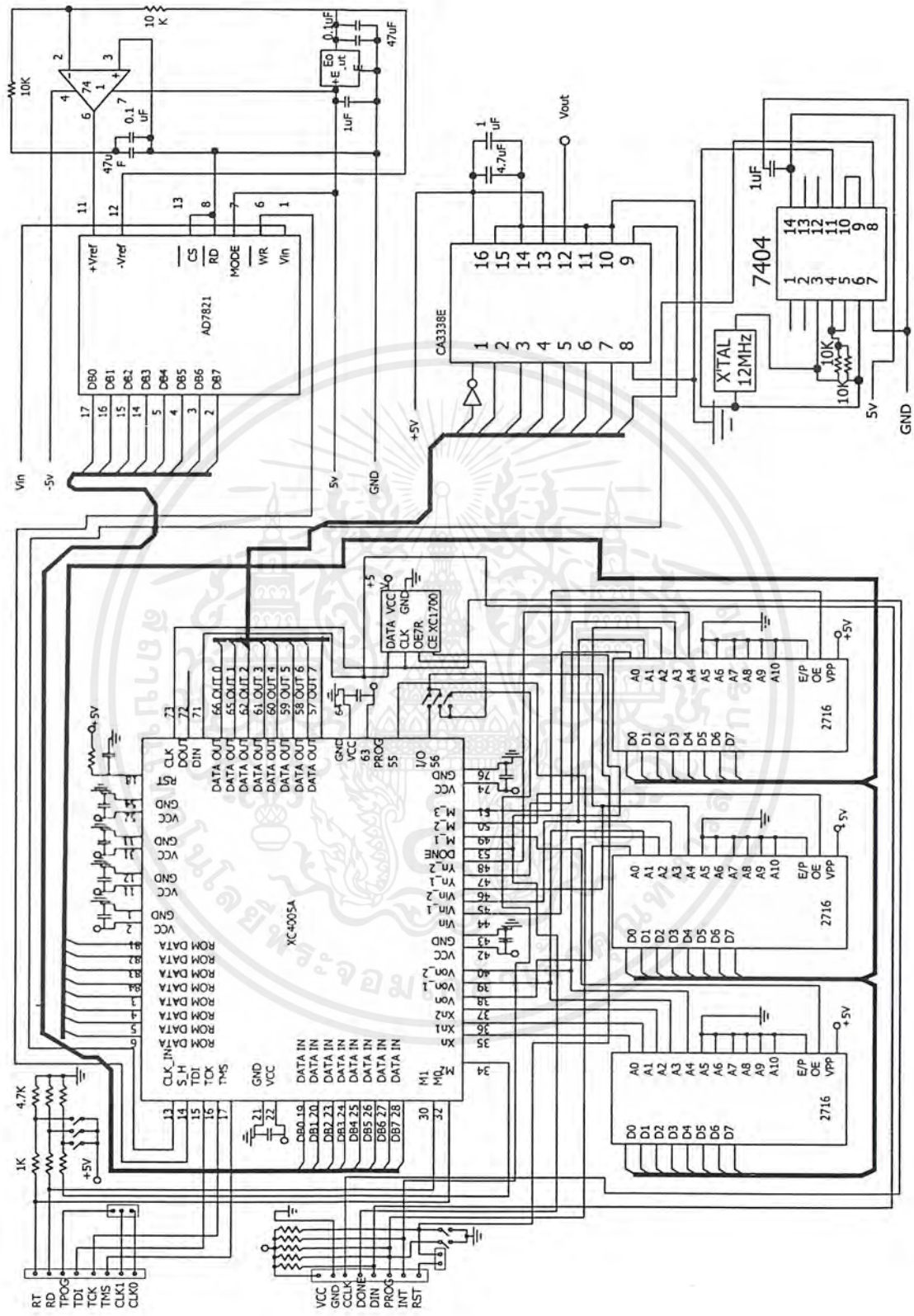
3.9.4 ส่วนเอาต์พุต

เป็นวงจร D/A ซึ่งทำการแปลงสัญญาณเชิงเลขที่ได้จากภาคประมวลผลให้กลับเป็นสัญญาณเชิงอุปมานเหมือนเดิม โดยรับสัญญาณดิจิตอลขนาด 8 บิตเข้ามาซึ่งทำการออกแบบวงจรได้ดังรูป 3.11



รูปที่ 3.11 การออกแบบส่วนเอาต์พุต

หลังจากทำการออกแบบวงจรทุกส่วนแล้วนำวงจรมารวมกันเป็นวงจรรวมดังรูปที่ 3.12



รูปที่ 3.12 วงจรรวม

บทที่ 4

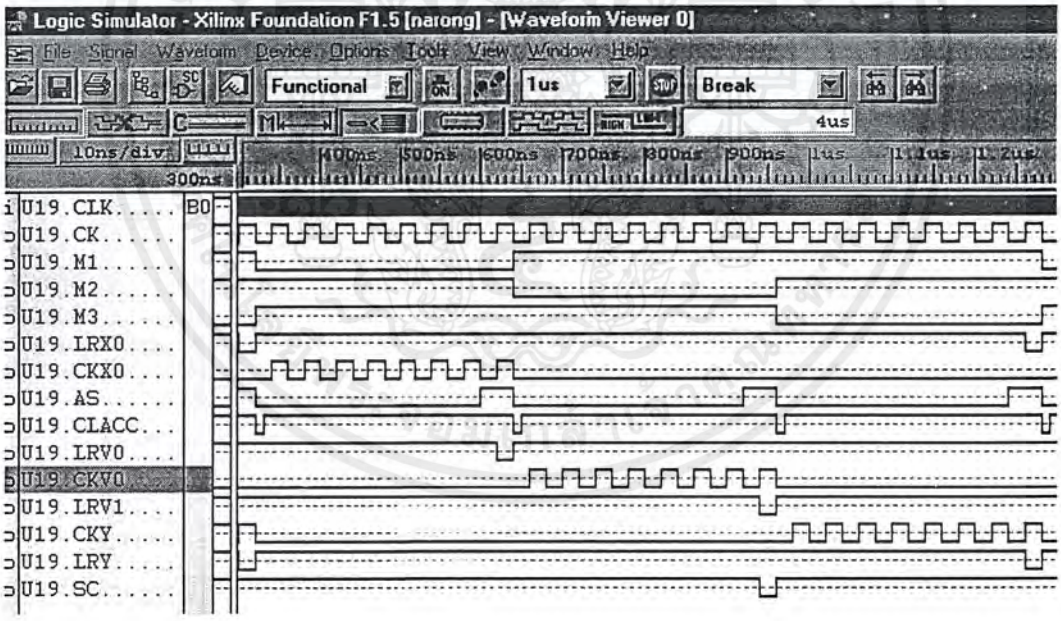
การทดลองและผลการทดลอง

4.1 ผลการทดลอง VHDL

จากบทที่ 3 ซึ่งได้ทำการออกแบบการทำงานโดยแบ่งออกเป็น 4 ส่วน และใช้ภาษา VHDL ในการออกแบบส่วนควบคุม ส่วนประมวลผล และส่วนการเลื่อนข้อมูลนั้น ในบทนี้ได้จำลองการทำงานของแต่ละส่วน ซึ่งได้ผลการทำงานดังนี้

4.1.1 ส่วนควบคุม

จากขั้นตอนการออกแบบผังการทำงานส่วนควบคุมในหัวข้อ 3.8.2 แล้วนำมาเขียนเป็นภาษา VHDL ได้ดังโปรแกรมที่ 2 ในภาคผนวก ก จากนั้นนำโปรแกรม VHDL มาทำการสังเคราะห์และจำลองการทำงาน ได้ผลดังรูปที่ 4.1

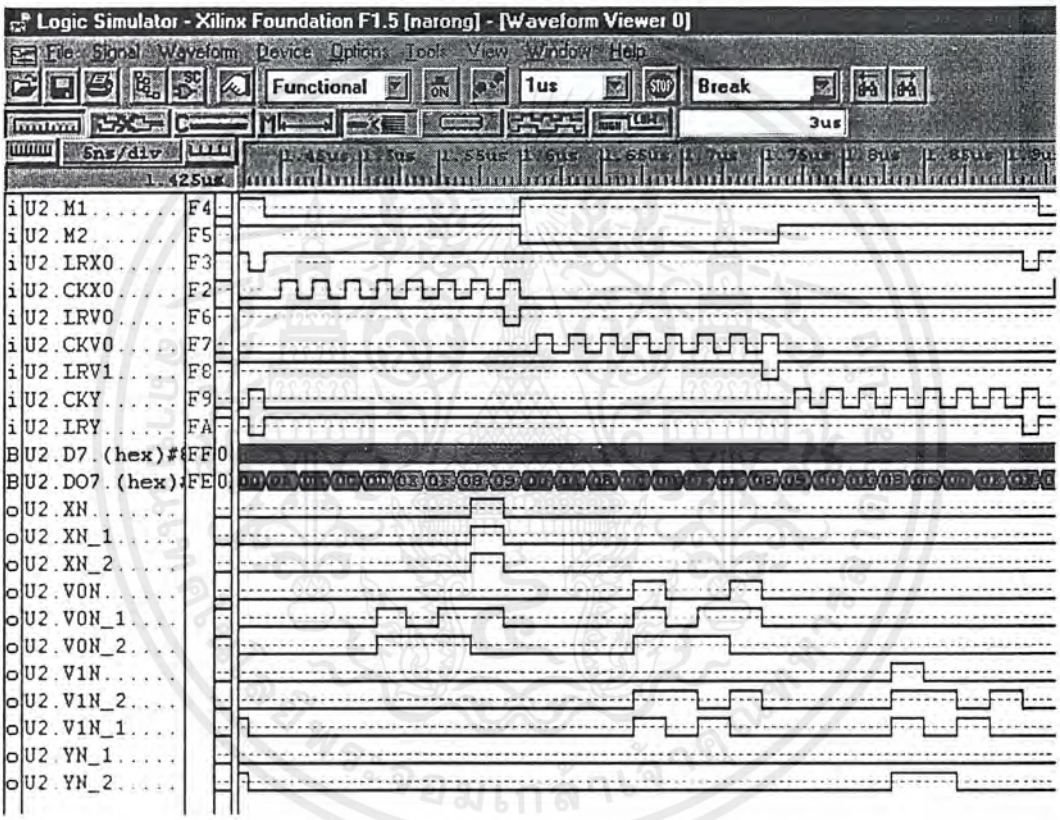


รูปที่ 4.1 ผลการจำลองการทำงานของส่วนควบคุม

สัญญาณที่ได้จากส่วนควบคุมซึ่งจำลองการทำงานโดยทดลองป้อนสัญญาณนาฬิกาเข้าที่อินพุต แล้วจะได้สัญญาณเอาต์พุตซึ่งเป็นสัญญาณการควบคุมการทำงานทั้งหมดของวงจรกรองสัญญาณเชิงเลข

4.1.2 ส่วนเลื่อนข้อมูล

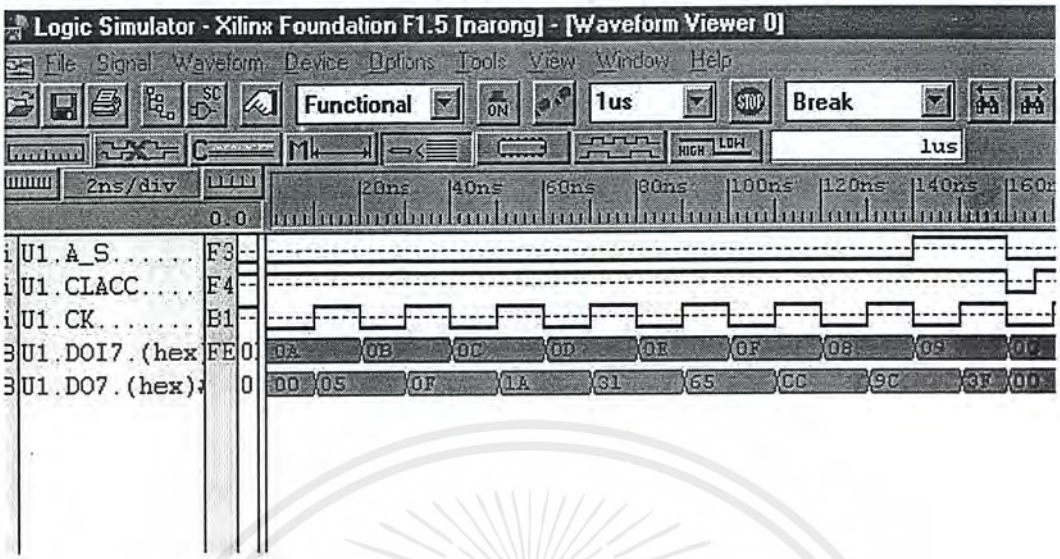
จากขั้นตอนการออกแบบผังการทำงานส่วนเลื่อนข้อมูลในหัวข้อ 3.8.3 แล้วนำมาเขียนเป็นภาษา VHDL ได้ดังโปรแกรมที่ 3 ในภาคผนวก ก จากนั้นนำโปรแกรม VHDL มาทำการสังเคราะห์และจำลองการทำงานโดยทำการป้อนสัญญาณอินพุตโดยกำหนดให้เหมือนกับสัญญาณควบคุมจากส่วนควบคุม ส่วนข้อมูลอินพุตจาก A/D กำหนดเป็น 01H จะได้ผลการจำลองการทำงาน ดังรูปที่ 4.2



รูปที่ 4.2 ผลการจำลองการทำงานของส่วนเลื่อนข้อมูล

4.1.3 ส่วนประมวลผล

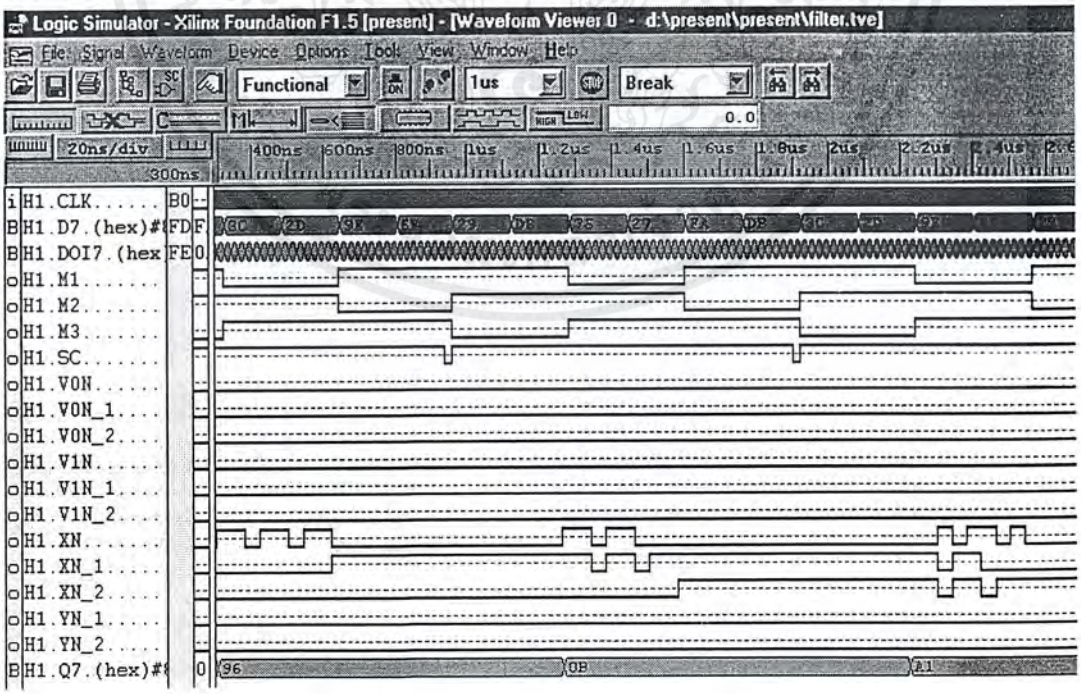
จากขั้นตอนการออกแบบผังการทำงานส่วนประมวลผลในหัวข้อ 3.8.1 แล้วนำมาเขียนเป็นภาษา VHDL ได้ดังโปรแกรมที่ 1 ในภาคผนวก ก จากนั้นนำโปรแกรม VHDL มาทำการสังเคราะห์และจำลองการทำงาน โดยทำการป้อนสัญญาณอินพุตที่ส่วนประมวลผลกำหนดให้ข้อมูลจาก ROM เท่ากับ 01H และกำหนดควบคุมสัญญาณต่างๆ ให้เหมือนกับสัญญาณจากส่วนควบคุม ซึ่งจะได้ผลการจำลองการทำงานดังรูปที่ 4.3



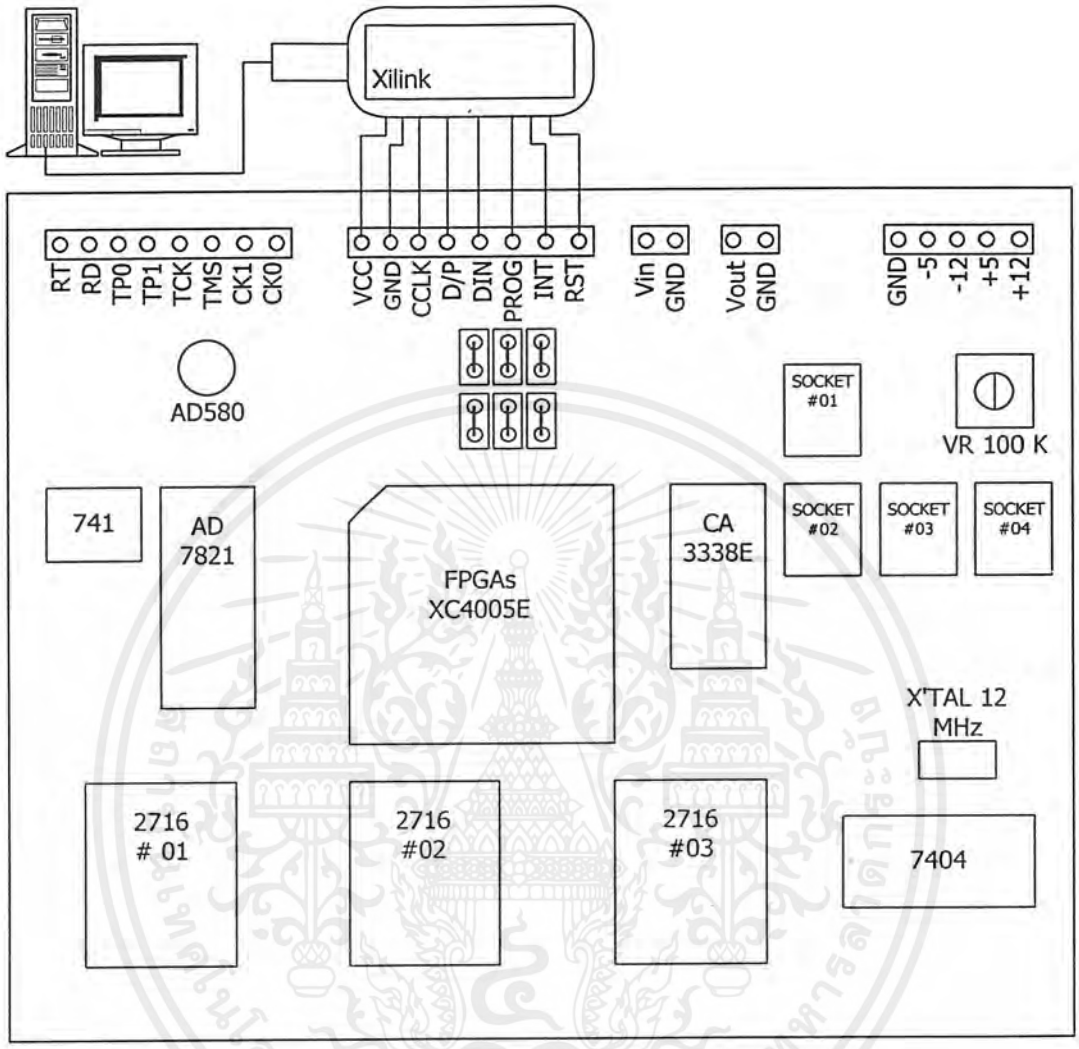
รูปที่ 4.3 ผลการจำลองการทำงานของส่วนประมวลผล

4.1.4 รวมการทำงานส่วนควบคุม, ส่วนเลื่อนข้อมูล, และส่วนประมวลผล

หลังจากรวมการทำงานทั้งหมดแล้ว ทำการป้อนสัญญาณจำลองการทำงาน ซึ่งได้ผลการจำลองการทำงานรวมทั้งหมดดังรูปที่ 4.4



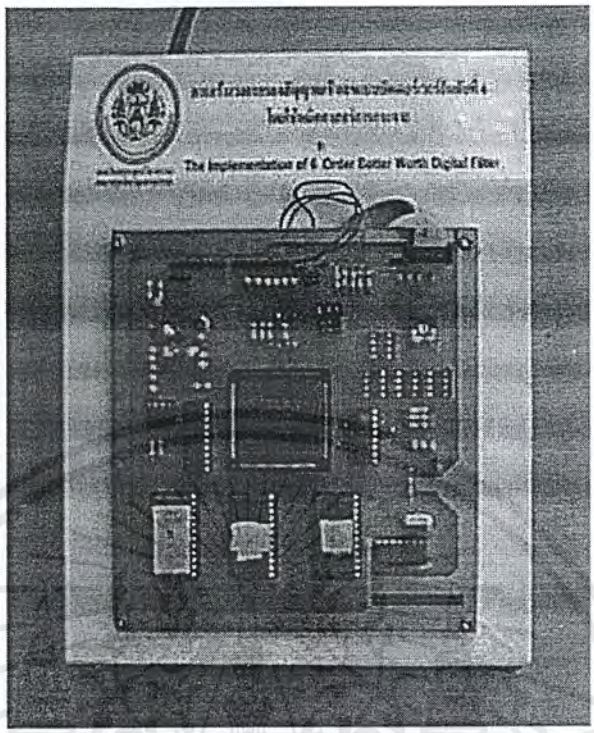
รูปที่ 4.4 ผลการจำลองการทำรวมทั้งหมด



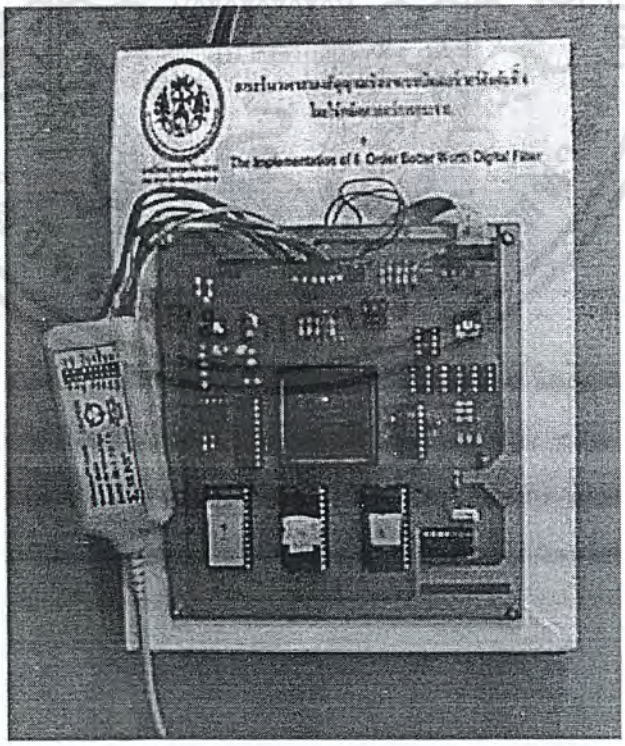
รูปที่ 4.5 การเชื่อมต่อกันระหว่างคอมพิวเตอร์กับบอร์ดตัวอย่างของ FPGAs

4.2 การดาวน์โหลดโปรแกรมลงบนตัวอย่างของ FPGAs

ผังการทำงานของบอร์ดตัวอย่างของ FPGAs แสดงดังรูปที่ 4.5 ใช้ไอซี FPGAs เบอร์ XC4005E ควบคุมการทำงาน โดยสามารถทำงานเป็นวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 ซึ่งบอร์ดตัวอย่างของ FPGAs ของจริงแสดงดังรูปที่ 4.6 และ 4.7



รูปที่ 4.6 ภาพถ่ายบอร์ดตัวอย่างของ FPGAs



รูปที่ 4.7 การทดสอบบอร์ดตัวอย่างของ FPGAs

เมื่อดำเนินการออกแบบตามขั้นตอนในบทที่ 3 จนได้ไฟล์ Filter.bit ของวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 จึงทำการทดสอบการทำงานของโปรแกรมและบอร์ดของตัวอย่างของ FPGAs โดยการดาวน์โหลดไฟล์ Filterbit ลงบนบอร์ดตัวอย่างของ FPGAs เมื่อต่อสายดาวน์โหลดเคเบิลจากพอร์ท COM2 เข้ากับบอร์ดตัวอย่างของ FPGAs ดังแสดงในรูปที่ 4.7 แล้ว สามารถทำการดาวน์โหลดไฟล์บิทสตรีมได้ด้วยขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 1 เข้าสู่โปรแกรม XDM แล้วคลิกเมาส์ที่เมนู Tools และเลือกที่คำสั่ง Hardware Debugger ซึ่งจะแสดงหน้าต่าง Communication Setup

ขั้นตอนที่ 2 กำหนดค่าอัตราความเร็วในการส่งข้อมูล (Baud Rate) และเลือกพอร์ทที่จะส่งข้อมูลออกไป ในโครงงานนี้เลือกอัตราความเร็วในการส่งข้อมูลที่ 9600 Bit/Sec และเลือกส่งข้อมูลออกที่พอร์ท COM1

ขั้นตอนที่ 3 ตั้งดิพสวิตซ์บอร์ดตัวอย่างของ FPGAs ให้ใช้งานลักษณะสเลฟซีเรียลซึ่งสามารถตั้งดิพสวิตซ์ได้ดังนี้คือ

ตารางที่ 4.1 การตั้งดิพสวิตซ์ต่างๆ ในบอร์ดตัวอย่างของ FPGAs

Dip SW	LABLE	SETTING
1	M0	ON
2	M1	ON
3	M2	ON
4	INIT	ON

ขั้นตอนที่ 4 เมื่อกำหนดอัตราความเร็วในการส่งข้อมูล และเลือกพอร์ตที่จะส่งข้อมูลออกไปแล้วให้ตอบตกลง จะกลับเข้าสู่หน้าต่าง Hardware Debugger ให้เลือกที่เมนู Download และเลือกที่ Down Design ซึ่งโปรแกรม Debugger จะทำการดาวน์โหลดไฟล์ Filter.bit ลงสู่บอร์ดตัวอย่างของ FPGAs เมื่อทำการดาวน์โหลดเสร็จจึงนำบอร์ดตัวอย่างของ FPGAs ไปทดลองได้

4.3 ผลการทดลอง

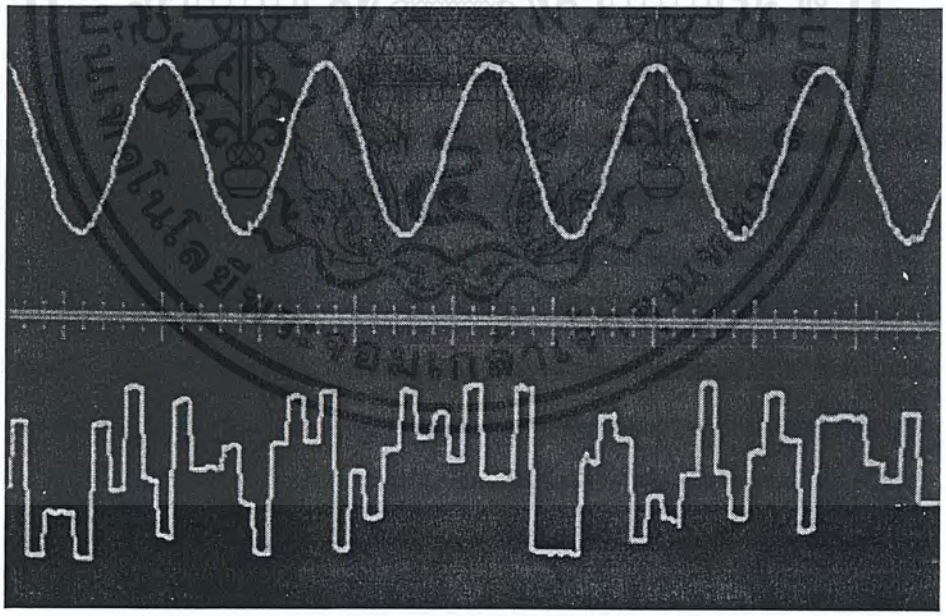
การทดลองทำได้โดยป้อนความถี่ต่างๆ เข้าไปและให้ค่า Vin = 5 $V_{(p-p)}$ จากนั้นจึงทำการอ่านค่า Vo(p-p) ได้ดังตารางที่ 4.2 และรูปสัญญาณอินพุตเปรียบเทียบกับเอาต์พุต ดังรูปที่ 4.8

ตารางที่ 4.2 ผลการทดลอง

F_{in}(Hz)	$V_{o(p-p)}$(VOLTS)	f_{in}(Hz)	$V_{o(p-p)}$(VOLTS)
100	5.27	2300	5.32
200	5.23	2400	5.33
300	5.32	2500	5.31
400	5.33	2600	5.29
500	5.31	2700	5.32
600	5.22	2800	5.31
700	5.22	2900	5.31
800	5.21	3000	5.32
900	5.27	3100	5.31
1000	5.25	3200	5.33
1100	5.25	3300	5.31
1200	5.29	3400	5.33
1300	5.32	3500	5.31
1400	5.32	4000	5.31
1500	5.32	4500	5.32
1600	5.31	5000	5.30
1700	5.31	5500	5.31
1800	5.31	6000	5.32
1900	5.31	6500	5.33
2000	5.30	7000	5.32
2100	5.32	7500	5.32
2200	5.32	8000	5.31

ตารางที่ 4.2 (ต่อ) ผลการทดลอง

F_{in}(Hz)	$V_{o(p-p)}$(VOLTS)	f_{in}(Hz)	$V_{o(p-p)}$(VOLTS)
8500	5.31	13000	5.30
9000	5.31	193500	5.30
9500	5.31	14000	5.25
10000	5.31	15000	5.27
10500	5.32	16000	5.35
11000	5.32	17000	5.33
11500	5.32	18000	5.32
12000	5.32	19000	5.25
12500	5.33	20000	5.26



รูปที่ 4.8 ผลการทดลอง

บทที่ 5
บทสรุป ปัญหา แนวทางแก้ไขและพัฒนา

5.1 บทสรุป

การออกแบบสร้างวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้คณิตศาสตร์การกระจายนั้นได้มีการออกแบบวงจรควบคุมทั้งหมดโดยใช้ภาษา VHDL เข้ามาช่วยในการออกแบบวงจรควบคุมแล้วจึงนำไปดาวน์โหลดลงในอุปกรณ์ FPGAs เพื่อทำการลดขนาดของวงจรลง ในส่วนของวงจรกรองความถี่แล้วจะใช้วงจรที่อยู่ในตัวของ FPGAs เท่านั้น โดยจะอาศัยค่าสัมประสิทธิ์ที่คำนวณได้ไว้ในหน่วยความจำ (EPROM) ภายนอก ในส่วนของการทำงานจะแบ่งออกเป็นสามส่วน อันประกอบไปด้วยส่วนแรก คือ ส่วนของวงจรที่เปลี่ยนสัญญาณจากแอนะ-ลอกเป็นสัญญาณดิจิตอลขนาดแปดบิต จากนั้นก็จะทำการส่งสัญญาณดิจิตอลที่เป็นสัญญาณอินพุตส่งเข้าไปยังส่วนของภาคชิฟท์รีจิสเตอร์ถูกควบคุมโดยภาคควบคุม ซึ่งทั้งหมดจะเป็นวงจรที่ถูกเขียนโดยภาษา VHDL บรรจุอยู่ภายในอุปกรณ์ FPGAs เมื่อสัญญาณที่เข้ามาจะถูกส่งออกมาเป็นแอดเดรสค่าต่างๆ กันเพื่อนำไปเปิดค่าสัมประสิทธิ์ที่ถูกเก็บไว้ภายในหน่วยความจำ (EPROM) จากนั้นก็จะทำการส่งสัญญาณออกมาในรูปแบบของสัญญาณดิจิตอลขนาดแปดบิตส่งกลับเข่าไปใน FPGAs ที่ภาคประมวล ทำการบวกสะสมครบทั้งเจ็ดครั้ง ครั้งที่แปดให้นำมาลบ ทำเช่นนี้ไปจนครบสามรอบ จึงนำเอาสัญญาณนี้ไปผ่านวงจรแปลงสัญญาณจากดิจิตอลเป็นสัญญาณแอนะลอกส่งเป็นสัญญาณเอาท์พุตต่อไป

ในส่วนของการดาว์นโหลดโปรแกรมที่ออกแบบโดยใช้ภาษา VHDL นั้นจะใช้โปรแกรม XACT STEP มาใช้เป็นส่วนช่วยในการดาว์นโหลดโปรแกรมผ่านทางสาย Xchecker ลงสู่อุปกรณ์ FPGAs ซึ่งจากการทดลองจะแบ่งเป็นสองส่วน คือ การทดลองในส่วนของซอฟต์แวร์ และส่วนของฮาร์ดแวร์ เมื่อทำการทดลองแล้วปรากฏว่าในส่วนของซอฟต์แวร์นั้นสามารถทำงานได้ตามปกติ เมื่อนำมารวมเข้าเป็นวงจรฮาร์ดแวร์สัญญาณที่ได้มีความผิดเพี้ยนจากเดิมไปเล็กน้อย ทั้งนี้อันเนื่องมาจากค่าความผิดพลาดของอุปกรณ์ทางด้านฮาร์ดแวร์นั้นเอง

5.2 ปัญหาและแนวทางแก้ไข

1) เนื่องจากว่าในการออกแบบวงจรด้วยภาษา VHDL ในบางครั้งในการสร้างวงจรนั้นการเดินสายสัญญาณในการเชื่อมต่ออุปกรณ์แต่ละตัวนั้นจะมีปัญหาที่อาจทำให้วงจรไม่ได้ต่อถึงกัน

แนวทางการแก้ไข

ควรจะมีการตรวจสอบหรือแก้ไขสายสัญญาณหรือสายบัสที่ทำหน้าที่ในการเชื่อมต่อระหว่างอุปกรณ์ว่าเชื่อมต่อกันดีหรือไม่ มิเช่นนั้นอาจจะทำให้วงจรทำงานไม่ครบวงจร

2) อุปกรณ์ FPGAs มีความร้อนเกิดขึ้นในขณะที่ใช้งานผิดปกติ

แนวทางการแก้ไข

เมื่อมีการกำหนดให้ภายในอุปกรณ์ FPGAs มีการทำงานของวงจรที่มีขนาดใหญ่ หรือใช้อุปกรณ์มากก็อาจจะเกิดการใช้กระแสมากจึงเป็นสาเหตุหนึ่งที่ทำให้ FPGAs เกิดความร้อน ควรกำหนดวงจรและอุปกรณ์ใช้งานให้มีขนาดเหมาะสม

3) ในส่วนของการออกแบบวงจรแปลงสัญญาณจากแอนะลอกเป็นสัญญาณดิจิตอลนั้นเกิดปัญหาในส่วนของการตอบสนองของสัญญาณเอาต์พุต

แนวทางการแก้ไข

ให้ตรวจสอบสัญญาณอินพุตที่ป้อนให้ และตรวจสอบระดับของแรงดันไฟเลี้ยงที่ VREF+ และ VREF- ให้อยู่ในระดับที่กำหนดและมีระดับแรงดันเท่ากัน

4) ในส่วนของการออกแบบวงจรแปลงสัญญาณจากดิจิตอลเป็นสัญญาณแอนะลอกนั้นจะเกิดปัญหาในส่วนของเอาท์พุตที่มีระดับของสัญญาณที่ต่ำ

แนวทางการแก้ไข

ให้ใช้ออปแอมป์มาทำการขยายระดับของสัญญาณให้มีขนาดของสัญญาณที่สูงขึ้น โดยกำหนดให้มีอัตราการขยายของสัญญาณประมาณ 10 เท่า

5) เกิดสัญญาณรบกวนที่สัญญาณเอาต์พุต อันส่งผลให้เกิดการไม่เสถียรของสัญญาณ

แนวทางในการแก้ไข

ให้นำตัวเก็บประจุค่า 1uF ชนิดใดก็ได้มาต่อคร่อมจุดที่เป็นแหล่งจ่ายไฟเลี้ยงกับกราวด์ให้ใกล้กับจุดที่เป็นไฟเลี้ยงของอุปกรณ์และกราวด์มากที่สุด

6) ในการสร้างสัญญาณนาฬิกานั้นอาจเกิดการทำงานที่ไม่พร้อมกัน ของขาอินพุตของไอซีเบอร์ 7400 จึงทำให้เกิดสัญญาณนาฬิกาที่ผิดเพี้ยน

แนวทางในการแก้ไข

ให้เปลี่ยนไอซีเกตที่มีขาเข้าของสัญญาณอินพุตที่มีทางเข้าเพียงขาเดียวเพื่อตัดปัญหาของการทำงานที่ไม่พร้อมกัน เช่นใช้ ไอซี 7404 แทน

7) ในการทำงานของวงจรรวมแล้ว เมื่อทำงานเกิดความผิดเพี้ยนอยู่เล็กน้อยทั้งนี้สาเหตุอาจจะเกิดจากสัญญาณนาฬิกาที่ป้อนเข้าสู่วงจรภายในอุปกรณ์ FPGAs

แนวทางในการแก้ไข

ตรวจสอบการทำงานของวงจรกำเนิดสัญญาณนาฬิกาให้มีสัญญาณนาฬิกาที่ 12 MHz ทั้งนี้เนื่องจากวงจรเป็นการทำงานที่มีความถี่สูงจึงต้องอาศัยระดับสัญญาณนาฬิกาที่สูงและคงที่

8) เนื่องจากในวงจรมีการใช้ไฟเลี้ยงที่แหล่งจ่ายขนาด $\pm$ 5 โวลท์ เป็นจำนวนมากโดยใช้ไอซีเบอร์ 7805 มาใช้ เกิดความร้อนที่ตัวไอซีสูง

แนวทางในการแก้ไข

ให้แบ่งส่วนของแหล่งจ่ายขนาด $\pm$ 5 โวลท์ ออกเป็น 2 ถึง 3 ชุดโดยแยกแหล่งจ่ายออกไปสู่แต่ละภาคทั้งนี้เพื่อแบ่งเบาภาระการโหลดของกระแสภายในตัวไอซี 7805

9) จากการนำเอาวงจรในภาคต่าง ๆ มาต่อรวมกันลงในอุปกรณ์ FPGAs แล้วปรากฏว่าสัญญาณในการเลื่อนข้อมูลในภาครีจิสเตอร์ไม่ทำงาน

แนวทางในการแก้ไข

เนื่องจากว่าในการออกแบบการทำงานโดยการจำลองด้วยภาษา VHDL จะต้องอาศัยความถูกต้องและแม่นยำสูง ดังนั้นเมื่อมีการออกแบบในส่วนของการทำงานของสัญญาณซ้ำ ๆ กัน อาจทำให้เกิดการทำงานที่ผิดเพี้ยนอาจส่งผลให้ได้เอาต์พุทไม่ตรงตามทฤษฎี ดังนั้นควรออกแบบวงจรให้มีความซับซ้อนน้อยที่สุด

5.3 แนวทางการพัฒนา

1) ในส่วนของการออกแบบหน่วยความจำ สามารถใช้ภาษา VHDL ออกแบบให้สามารถที่จะเป็นวงจรที่อยู่ภายในอุปกรณ์ FPGAs ได้

2) เพิ่มขนาดของอันดับในการกรองความถี่มากขึ้นเพื่อให้เกิดความถูกต้องตามทฤษฎี

3) เพิ่มขนาดของบิตที่ทำการแปลงสัญญาณจากแอนะลอกเป็นดิจิตอลให้สูงขึ้นแล้วคำนวณค่าสัมประสิทธิ์ใหม่เพื่อให้ได้สัญญาณที่ไม่ผิดเพี้ยนไปจากทฤษฎี

4) พัฒนาวงจรกรองสัญญาณที่อยู่ภายในอุปกรณ์ FPGAs ให้น้อยลงแต่ยังคงประสิทธิภาพในการทำงานคงเดิมหรือดีขึ้น

5) ลดจำนวนของอุปกรณ์ทางด้านฮาร์ดแวร์ เช่น การใช้สัญญาณนาฬิกาจากภายในอุปกรณ์ FPGAs

6) พัฒนาและออกแบบวงจรให้สามารถเลือกโหมดการกรองความถี่ในระดับต่าง ๆ กันได้เพื่อการศึกษาในเรื่องของวงจรกรองสัญญาณความถี่ในระดับต่าง ๆ

ภาคผนวก ก

โปรแกรม

โปรแกรมที่ ก.1 ส่วนประมวลผล

```
library IEEE;
use IEEE.std_logic_1164.all;

entity ALU_UNIT is
port (
      Doi: in STD_LOGIC_vector(7 downto 0);
      CK: in STD_LOGIC;
      CLACC: in STD_LOGIC;
      A_S: in STD_LOGIC;
     Do: out STD_LOGIC_vector(7 downto 0)
           );
end ALU_UNIT;

architecture ALU_UNIT_arch of ALU_UNIT is
component exor_gate
port (
 a: in STD_LOGIC;
 b: in STD_LOGIC;
 c: out STD_LOGIC);
end component;
component ttl_74ls83
   port (
      A: in STD_LOGIC_VECTOR(3 downto 0);
      B: in STD_LOGIC_VECTOR(3 downto 0);
      Ci: in STD_LOGIC;
      Co: out STD_LOGIC;
      S: out STD_LOGIC_VECTOR(3 downto 0));
end component;
component ttl_74174
   port (
      D5: in STD_LOGIC_VECTOR(4 downto 0);
```

```
        CLK: in STD_LOGIC;
        CLR: in STD_LOGIC;
        Q5: out STD_LOGIC_VECTOR(4 downto 0));
    end component;
    signal Dooi : std_logic_vector(7 downto 0);
    signal ff_acc2: std_logic;
    signal compA2 : std_logic_vector(3 downto 0);
    signal compA1 : std_logic_vector(7 downto 4);
    signal acc1 : std_logic_vector(4 downto 0);
    signal acc2 : std_logic_vector(4 downto 0);
    signal co_acc2 : std_logic;
    signal co_acc1 : std_logic;
    signal sum_acc2 : std_logic_vector(3 downto 0);
    signal sum_acc1 : std_logic_vector(3 downto 0);
    signal fb_add2 : std_logic_vector(3 downto 0);
    signal fb_add1 : std_logic_vector(3 downto 0);
    signal in_acc2 : std_logic_vector(4 downto 0);
    signal in_acc1 : std_logic_vector(4 downto 0);
    signal ADD1A : std_logic;
    signal ADD1B : std_logic;
    begin
    stages : for i in 3 downto 0 generate
    ex_or  :  exor_gate  port map (Doi(i),A_S,compA2(i));
              end generate;
    stages1 : for i in 7 downto 4 generate
    ex_or  : exor_gate port map (Doi(i),A_S,compA1(i));
              end generate;
              fb_add2(3) <= acc1(4);
              fb_add2(2) <= acc2(0);
              fb_add2(1)  <= acc2(1);
              fb_add2(0)  <= acc2(2);
              fb_add1(3) <= acc1(0);
              fb_add1(2) <= acc1(1);
              fb_add1(1) <= acc1(2);
```

```
        fb_add1(0) <= acc1(3);
add_2 : ttl_74ls83 port map (fb_add2,compA2,A_S,co_acc2,sum_acc2);
add_1 : ttl_74ls83 port map (fb_add1,compA1,co_acc2,co_acc1,sum_acc1);
        in_acc2(0) <= sum_acc2(3);
        in_acc2(1) <= sum_acc2(2);
        in_acc2(2) <= sum_acc2(1);
        in_acc2(3) <= sum_acc2(0);
        ff_acc2 <= acc2(3);
        in_acc2(4) <= ff_acc2;
        in_acc1(1) <= sum_acc1(3);
        in_acc1(2) <= sum_acc1(2);
        in_acc1(3) <= sum_acc1(1);
        in_acc1(4) <= sum_acc1(0);
        ADD1A <= compA1(7) xor co_acc1;
        ADD1B <= ADD1A xor ACC1(0);
        in_acc1(0) <= ADD1B;
acc_1 : ttl_74174 port map (in_acc1,CK,CLACC,ACC1);
acc_2 : ttl_74174 port map (in_acc2,CK,CLACC,ACC2);
        Dooi(7) <= ACC1(2);
        Dooi(6) <= ACC1(3);
        Dooi(5) <= ACC1(4);
        Dooi(4) <= ACC2(0);
        Dooi(3) <= ACC2(1);
        Dooi(2) <= ACC2(2);
        Dooi(1) <= ACC2(3);
        Dooi(0) <= ACC2(4);
        Do <= Dooi;
end ALU_UNIT_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity exor_gate is
 port (
```

```
        a: in STD_LOGIC;
        b: in STD_LOGIC;
        c: out STD_LOGIC
        );
        end exor_gate;

        architecture exor_gate_arch of exor_gate is
        begin
                c <= a xor b;
        end exor_gate_arch;

        library IEEE;
        use IEEE.std_logic_1164.all;

        entity ttl_74ls83 is
          port (
            A: in STD_LOGIC_VECTOR(3 downto 0);
            B: in STD_LOGIC_VECTOR(3 downto 0);
            Ci: in STD_LOGIC;
            Co: out STD_LOGIC;
            S: out STD_LOGIC_VECTOR(3 downto 0)
          );
        end ttl_74ls83;

        architecture ttl_74ls83_arch of ttl_74ls83 is
        component Full_adder
          port (
            X : in STD_LOGIC;
            Y : in STD_LOGIC;
            Cin : in STD_LOGIC;
            Cout : out STD_LOGIC;
            Sum : out STD_LOGIC);
        end component;
signal C : std_logic_vector(3 downto 0);
```

```
begin
stages : for i in 3 downto 0 generate
Lowbit : if i = 0 generate
FA     : Full_adder port map (A(0),B(0),Ci,C(0),S(0));
            end generate;
Otherbit : if i /= 0 generate
FA     : Full_adder port map (A(i),B(i),C(i-1),C(i),S(i));
            end generate;
            end generate;
            Co <= C(3);
end ttl_74ls83_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity Full_adder is
  port (
    X: in STD_LOGIC;
    Y: in STD_LOGIC;
    Cin: in STD_LOGIC;
    Cout: out STD_LOGIC;
    Sum: out STD_LOGIC
  );
end Full_adder;

architecture Full_adder_arch of Full_adder is
begin
            Sum <= X xor Y Xor Cin;
            Cout <= (x and Y) or (x and Cin) or (Y and Cin);
end Full_adder_arch;

library IEEE;
use IEEE.std_logic_1164.all;
```

```
entity ttl_74174 is
  port (
    D5: in STD_LOGIC_VECTOR(4 downto 0);
    CLK: in STD_LOGIC;
    CLR: in STD_LOGIC;
    Q5: out STD_LOGIC_VECTOR(4 downto 0)
  );
end ttl_74174;

architecture ttl_74174_arch of ttl_74174 is
component dff_rst_74174
  port (
    D: in STD_LOGIC;
    CLK: in STD_LOGIC;
    CLR: in STD_LOGIC;
    Q: out STD_LOGIC);
end component;
signal CLR_bar : STD_LOGIC;
signal CLK_bar : STD_LOGIC;
begin
        CLK_bar <= not CLK;
stages : for i in 4 downto 0 generate
DFF : dff_rst_74174 port map (D5(i),CLK_bar,CLR,Q5(i));
          end generate;
end ttl_74174_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity dff_rst_74174 is
  port (
    D: in STD_LOGIC;
    CLK: in STD_LOGIC;
    CLR: in STD_LOGIC;
```

```
Q: out STD_LOGIC
  );
end dff_rst_74174;

architecture dff_rst_74174_arch of dff_rst_74174 is
begin
        process(CLK,CLR,D)
        begin
        if (CLK = '0' and CLK'event) then
                Q <= D;
        end if;
        if (CLR = '0') then
                Q <= '0';
        end if;
        end process;
end dff_rst_74174_arch;
```

โปรแกรมที่ ก.2 ส่วนภาคควบคุม

```
library IEEE;
use IEEE.std_logic_1164.all;

entity CON_UNIT is
        port (
          clk: in STD_LOGIC;
          d1: out STD_LOGIC;
          d2: out STD_LOGIC;
          d1b: out STD_LOGIC;
          d2b: out STD_LOGIC;
          CLACC: out STD_LOGIC;
          AS: out STD_LOGIC;
          CKV0: out STD_LOGIC;
          CKX0: out STD_LOGIC;
          LRV0: out STD_LOGIC;
          M1: out STD_LOGIC;
          M2: out STD_LOGIC;
          M3: out STD_LOGIC;
          LRX0: out STD_LOGIC;
          LRV1: out STD_LOGIC;
          SC: out STD_LOGIC;
          CK: out STD_LOGIC;
          CKY: out STD_LOGIC;
          LRY: out STD_LOGIC
        );
end CON_UNIT;

architecture CON_UNIT_arch of CON_UNIT is
component not_1
        port (
            ai: in STD_LOGIC;
            co: out STD_LOGIC
                );
```

```
end component;
component JK_FF
  port (
    J: in STD_LOGIC;
    K: in STD_LOGIC;
    CLK: in STD_LOGIC;
    Q: out STD_LOGIC;
    QB: out STD_LOGIC
        );
end component;

component d_ff_pr
  port (
    clk: in STD_LOGIC;
    pre: in STD_LOGIC;
    q: out STD_LOGIC;
    qb: out STD_LOGIC
  );
end component;

component d_ff_rst_l
  port (
    d: in STD_LOGIC;
    clk: in STD_LOGIC;
    rst: in STD_LOGIC;
    q: out STD_LOGIC;
    qb: out STD_LOGIC
  );
end component;

component and_g
  port (
    a: in STD_LOGIC;
    b: in STD_LOGIC;
```

```
c: out STD_LOGIC
  );
end component;
component nor_g
  port (
    a: in STD_LOGIC;
    b: in STD_LOGIC;
    c: out STD_LOGIC
  );
end component;
component counter_74393
  port (
    CLR: in STD_LOGIC;
    CLK: in STD_LOGIC;
    Qa: out STD_LOGIC;
    Qb: out STD_LOGIC;
    Qc: out STD_LOGIC;
    Qd: out STD_LOGIC;
    Qe: out STD_LOGIC
  );
end component;
signal rst : std_logic;
signal a11 : std_logic;
signal a12 : std_logic;
signal b11 : std_logic;
signal c11 : std_logic;
signal d11 : std_logic;
signal e11 : std_logic;
signal m_1 : std_logic;
signal m_2 : std_logic;
signal m_3 : std_logic;
signal m_2b : std_logic;
signal m_3b : std_logic;
signal Qd1b : std_logic;
```

```
signal Qd1 : std_logic;
signal Di : std_logic;
signal Ki : std_logic;
signal a_s : std_logic;
signal Qd2 : std_logic;
signal Qd2b : std_logic;
signal b11b : std_logic;
signal c_and_d : std_logic;
signal lr_v1 : std_logic;
signal m_3_b : std_logic;
signal CKV0_S : std_logic;
signal CKX0_S : std_logic;
signal LRV0_S : std_logic;
signal LRX0_S : std_logic;
signal LRV1_S : std_logic;
signal SC_S  : std_logic;
begin
        Ki <= '1';
        rst <= '0';
u1 : counter_74393 port map (rst,clk,a11,b11,c11,d11,e11);
        CK <= b11;
u2 : not_1     port map (a11,a12);
u3a : d_ff_pr  port map (Qd2b,a12,Qd1,Qd1b);
        CLACC <= not Qd1b ;
u3b : d_ff_rst_1  port map (a_s,b11,c11,Qd2,Qd2b);
u4a : JK_FF    port map (m_3b,Ki,e11,m_2,m_2b);
u4b : JK_FF    port map (m_2,m_2b,e11,m_3,m_3b);
u5 : nor_g     port map (m_2,m_3,m_1);
u6a : not_1    port map (b11,b11b);
u6b : not_1    port map (m_3,m_3_b);
u7a : and_g    port map (c11,d11,c_and_d);
u7b : and_g    port map (c_and_d,e11,a_s);  AS <= a_s;
u7c : and_g    port map (m_2,b11,CKV0);
u7d : and_g          port map (b11,m_1,CKX0);
```

```
u7e : and_g    port map (m_1,Qd2,LRV0_S);  LRV0 <= not LRV0_S;
u7f : and_g    port map (Qd2,m_3,LRX0_S);  LRX0 <= not LRX0_S;
          LRY <= notLRX0_S;
u7g : and_g    port map (Qd2,m_2,lr_v1);  LRV1 <= not lr_v1;  SC <= not lr_v1;
u8  : nor_g    port map (b11b,m_3_b,CKY);
u9a : not_1    port map (m_2,M2);
u9b : not_1    port map (m_1,M1);
u9c : not_1    port map (m_3,M3);
end CON_UNIT_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity JK_FF is
  port (
    J: in STD_LOGIC;
    K: in STD_LOGIC;
    CLK: in STD_LOGIC;
    Q: out STD_LOGIC;
    QB: out STD_LOGIC
  );
end JK_FF;

architecture JK_FF_arch of JK_FF is
signal state : STD_LOGIC;
signal ST : STD_LOGIC;
begin
          Q <= state;
          QB <= not state;
          ST <= state;
          process(J,K,CLK)
          begin
                    if(CLK = '0' and CLK'event) then
                    if (J = '1' and K = '0') then
```

```
            state <= '1';
                        end if;
                        if (J = '0' and K = '1') then
                        state <= '0';
                        end if;
                        if (J = '1' and K = '1') then
                        state <= not ST;
                        end if;
                        end if;
end process;
end JK_FF_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity d_ff_pr is
  port (
    clk: in STD_LOGIC;
    pre: in STD_LOGIC;
    q: out STD_LOGIC;
    qb: out STD_LOGIC
  );
end d_ff_pr;

architecture d_ff_pr_arch of d_ff_pr is
signal state : std_logic;
begin
            q <= state;
            qb <= not state;
            process(clk,pre)
            begin
            if ( clk = '1' and clk'event) then
            state <= '0';
            end if;
```

```
        if ( pre = '0') then
                state <= '1';
                end if;
                end process;
        end d_ff_pr_arch;

        library IEEE;
        use IEEE.std_logic_1164.all;

        entity d_ff_rst_1 is
          port (
            d: in STD_LOGIC;
            clk: in STD_LOGIC;
            rst: in STD_LOGIC;
            q: out STD_LOGIC;
            qb: out STD_LOGIC
          );
        end d_ff_rst_1;

        architecture d_ff_rst_1_arch of d_ff_rst_1 is
        signal state : std_logic;
        begin
                q <= state;
                qb <= not state;
                process(d,clk,rst)
                begin
                        if ( clk = '1' and clk'event) then
                        state <= d;
                        end if;
                        if ( rst = '0') then
                        state <= '0';
                        end if;
                end process;
end d_ff_rst_1_arch;
```

```
library IEEE;
use IEEE.std_logic_1164.all;

entity and_g is
  port (
    a: in STD_LOGIC;
    b: in STD_LOGIC;
    c: out STD_LOGIC
  );
end and_g;

architecture and_g_arch of and_g is
begin
        c <= a and b;
end and_g_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity nor_g is
  port (
    a: in STD_LOGIC;
    b: in STD_LOGIC;
    c: out STD_LOGIC
  );
end nor_g;

architecture nor_g_arch of nor_g is
begin
        c <= a nor b;
end nor_g_arch;

library IEEE;
use IEEE.std_logic_1164.all;
```

```
entity counter_74393 is
 port (
 CLR: in STD_LOGIC;
 CLK: in STD_LOGIC;
 Qa: out STD_LOGIC;
 Qb: out STD_LOGIC;
 Qc: out STD_LOGIC;
 Qd: out STD_LOGIC;
 Qe: out STD_LOGIC
 );
 end counter_74393;

 architecture counter_74393_arch of counter_74393 is
 component not_1
 port (
 ai: in STD_LOGIC;
 co: out STD_LOGIC
 );
end component;
component T_CL
port (
      T: in STD_LOGIC;
      CL: in STD_LOGIC;
      Q: out STD_LOGIC
   );
end component;
signal n_g : STD_LOGIC;
signal a,b,c,d,e : STD_LOGIC;
begin
            u1 : not_1 port map (CLR,n_g);
            u2 : T_CL  port map (CLK,n_g,a);
            u3 : T_CL port map (a,n_g,b);
            u4 : T_CL port map (b,n_g,c);
            u5 : T_CL port map (c,n_g,d);
```

```
u6 : T_CL port map (d,n_g,e);
        Qa <= a;
        Qb <= b;
        Qc <= c;
        Qd <= d;
        Qe <= e;
end counter_74393_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity not_1 is
  port (
    ai: in STD_LOGIC;
    co: out STD_LOGIC
  );
end not_1;

architecture not_1_arch of not_1 is
begin
        co <= not ai;
end not_1_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity T_CL is
  port (
    T: in STD_LOGIC;
    CL: in STD_LOGIC;
    Q: out STD_LOGIC
  );
end T_CL;
```

```
architecture T_CL_arch of T_CL is
signal state : std_logic;
signal st : std_logic;
begin
        Q <= state;
        st <= state;
        process(T,CL)
        begin
        if(T = '0' and T'event) then
        state <= not st;
        end if;
        if(CL = '0') then
        state <= '0';
        end if;
        end process;
end T_CL_arch;
```

โปรแกรมที่ ก.3 ส่วนเลื่อนข้อมูล

```
library IEEE;
use IEEE.std_logic_1164.all;

entity REG_UNIT_1 is
  port (
        M1: in STD_LOGIC;
        M2: in STD_LOGIC;
        D: in STD_LOGIC_vector(7 downto 0);
        DO: in STD_LOGIC_vector(7 downto 0);
        CKX0: in STD_LOGIC;
        LRX0: in STD_LOGIC;
        CKV0: in STD_LOGIC;
        LRV0: in STD_LOGIC;
        LRV1: in STD_LOGIC;
        CKY: in STD_LOGIC;
        LRY: in STD_LOGIC;
        Xn: out STD_LOGIC;
        Xn_1: out STD_LOGIC;
        Xn_2: out STD_LOGIC;
        V0n: out STD_LOGIC;
        V0n_1: out STD_LOGIC;
        V0n_2: out STD_LOGIC;
        V1n: out STD_LOGIC;
        V1n_1: out STD_LOGIC;
        V1n_2: out STD_LOGIC;
        Yn_1: out STD_LOGIC;
        Yn_2: out STD_LOGIC
        );
end REG_UNIT_1;

architecture REG_UNIT_1_arch of REG_UNIT_1 is
component IC_74ls165
        port (
```

```
            S_L: in STD_LOGIC;
            CK: in STD_LOGIC;
            data: in STD_LOGIC_vector(7 DOWNTO 0);
            Q: out STD_LOGIC
            );
end component;
component ttl_74ls157
            port (
            A: in STD_LOGIC_VECTOR(4 downto 1);
            B: in STD_LOGIC_VECTOR(4 downto 1);
            G: in STD_LOGIC;
            SL: in STD_LOGIC;
            Y: out STD_LOGIC_VECTOR(4 downto 1)
            );
end component;
component ttl_74ls91
            port (
            A: in STD_LOGIC;
            CLK: in STD_LOGIC;
            QH: out STD_LOGIC
            );
end component;
signal Q_RX0 : STD_LOGIC;
signal Q_RX1 : STD_LOGIC;
signal Q_RX1_BAR : STD_LOGIC;
signal Q_RX2 : STD_LOGIC;
signal Q_RX2_BAR : STD_LOGIC;
signal SER : STD_LOGIC;
signal NH : STD_LOGIC;
signal Q_RV0 : STD_LOGIC;
signal IN_MUX1_A : STD_LOGIC_VECTOR(4 downto 1);
signal IN_MUX1_B : STD_LOGIC_VECTOR(4 downto 1);
signal OUT_MUX1 : STD_LOGIC_VECTOR(4 downto 1);
signal Q_RV1 : STD_LOGIC;
```

```
signal Q_RV2 : STD_LOGIC;
signal Q_RV1_BAR : STD_LOGIC;
signal Q_RV2_BAR : STD_LOGIC;
signal Q_RV10 : STD_LOGIC;
signal IN_MUX2_A : STD_LOGIC_VECTOR(4 downto 1);
signal IN_MUX2_B : STD_LOGIC_VECTOR(4 downto 1);
signal OUT_MUX2 : STD_LOGIC_VECTOR(4 downto 1);
signal Q_RV11 : STD_LOGIC;
signal Q_RV11_BAR : STD_LOGIC;
signal Q_RV12 : STD_LOGIC;
signal Q_RV12_BAR : STD_LOGIC;
signal Q_RY : STD_LOGIC;
signal Q_RY1 : STD_LOGIC;
signal Q_RY1_BAR : STD_LOGIC;
signal OUT_MUX2_2 : STD_LOGIC;
signal OUT_MUX1_2 : STD_LOGIC;
begin
        SER <= '1';
        NH  <= '0';
RX0 : IC_74ls165 port map (LRX0,CKX0,D,Q_RX0);
RX1 : ttl_74ls91 port map (Q_RX0,CKX0,Q_RX1);
RX2 : ttl_74ls91 port map (Q_RX1,CKX0,Q_RX2);
        Xn <= Q_RX0;  Xn_1 <= Q_RX1;  Xn_2 <= Q_RX2;
RV0 : IC_74ls165 port map (LRV0,OUT_MUX1(1),DO,Q_RV0);
        IN_MUX1_A(1) <= CKX0;
        IN_MUX1_A(2) <= '1';
        IN_MUX1_A(3) <= Q_RV1;
        IN_MUX1_A(4) <= Q_RV2;
        IN_MUX1_B(1) <= CKV0;
        IN_MUX1_B(2) <= '1';
        IN_MUX1_B(3) <= Q_RV0;
        IN_MUX1_B(4) <= Q_RV1;
MUX1 : ttl_74ls157 port map (IN_MUX1_A,IN_MUX1_B,NH,M1,OUT_MUX1);
        OUT_MUX1_2 <= OUT_MUX1(2);
```

```
RV1 : ttl_74ls91 port map (OUT_MUX1(3),OUT_MUX1(1),Q_RV1);
        RV2 : ttl_74ls91 port map (OUT_MUX1(4),OUT_MUX1(1),Q_RV2);
                V0n <= Q_RV0;  V0n_1 <= Q_RV1 ; V0n_2 <= Q_RV2;

        RV10 : IC_74ls165 port map (LRV1,OUT_MUX2(1),DO,Q_RV10);
                IN_MUX2_A(1) <= CKV0;
                IN_MUX2_A(2) <= '1';
                IN_MUX2_A(3) <= Q_RV11;
                IN_MUX2_A(4) <= Q_RV12;
                IN_MUX2_B(1) <= CKY;
                IN_MUX2_B(2) <= '1';
                IN_MUX2_B(3) <= Q_RV10;
                IN_MUX2_B(4) <= Q_RV11;
        MUX2 : ttl_74ls157 port map (IN_MUX2_A,IN_MUX2_B,NH,M2,OUT_MUX2);
        V11 : ttl_74ls91 port map (OUT_MUX2(3),OUT_MUX2(1),Q_RV11);
                OUT_MUX2_2 <= OUT_MUX2(2);
        V12 : ttl_74ls91 port map (OUT_MUX2(4),OUT_MUX2(1),Q_RV12);
                V1n <= Q_RV10; V1n_1 <= Q_RV11; V1n_2 <= Q_RV12;
        RY : IC_74ls165 port map (LRY,CKY,DO,Q_RY);
        RY1 : ttl_74ls91 port map (Q_RY,CKY,Q_RY1);
                Yn_1 <= Q_RY;Yn_2 <= Q_RY1;
        end REG_UNIT_1_arch;

        library IEEE;
        use IEEE.std_logic_1164.all;

        entity IC_74ls165 is
                port (
                 S_L: in STD_LOGIC;
                 CK: in STD_LOGIC;
                data: in STD_LOGIC_vector(7 DOWNTO 0);
                Q: out STD_LOGIC
                );
end IC_74ls165;
```

```
architecture IC_74ls165_arch of IC_74ls165 is
component df_clr_ce_1
        port (
        d: in STD_LOGIC;
        clk: in STD_LOGIC;
        clr: in STD_LOGIC;
        q: out STD_LOGIC
          );
end component;
component mul2_1E
         port (
        d0: in STD_LOGIC;
        d1: in STD_LOGIC;
        s0: in STD_LOGIC;
        o: out STD_LOGIC
        );
end component;
component or2_b1
         port (
        a: in STD_LOGIC;
        b: in STD_LOGIC;
        f: out STD_LOGIC
        );
end component;
signal CK_SL : std_logic;
signal CLR : std_logic;
signal MD     : std_logic_vector(7 downto 0);
signal QS     : std_logic_vector(7 downto 0);
begin
         CLR <= '0';
stages : for i in 7 downto 0 generate
lowbit : if i = 7 generate
MUL    : mul2_1E  port map (data(7),CLR,S_L,MD(7));
         end generate;
```

```
other  : if i /= 7 generate
MUL    : mul2_1E  port map (data(i),QS(i+1),S_L,MD(i));
           end generate;
           end generate;
stages1 : for i in 7 downto 0 generate
DF     : df_clr_ce_1  port map (MD(i),CK_SL,CLR,QS(i));
           end generate;
u1:or2_b1 port map(CK,S_L,CK_SL);
           Q <= QS(0);
end IC_74ls165_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity df_clr_ce_1 is
           port (
           d: in STD_LOGIC;
           clk: in STD_LOGIC;
           clr: in STD_LOGIC;
           q: out STD_LOGIC
           );
end df_clr_ce_1;

architecture df_clr_ce_1_arch of df_clr_ce_1  is
begin
            process (clk,clr,d)
            begin
           if (clr = '1') then
           q <= '0';
           elsif (clk = '1' and clk'event) then
           q <= d;
           end if;
            end process;
end df_clr_ce_1_arch;
```

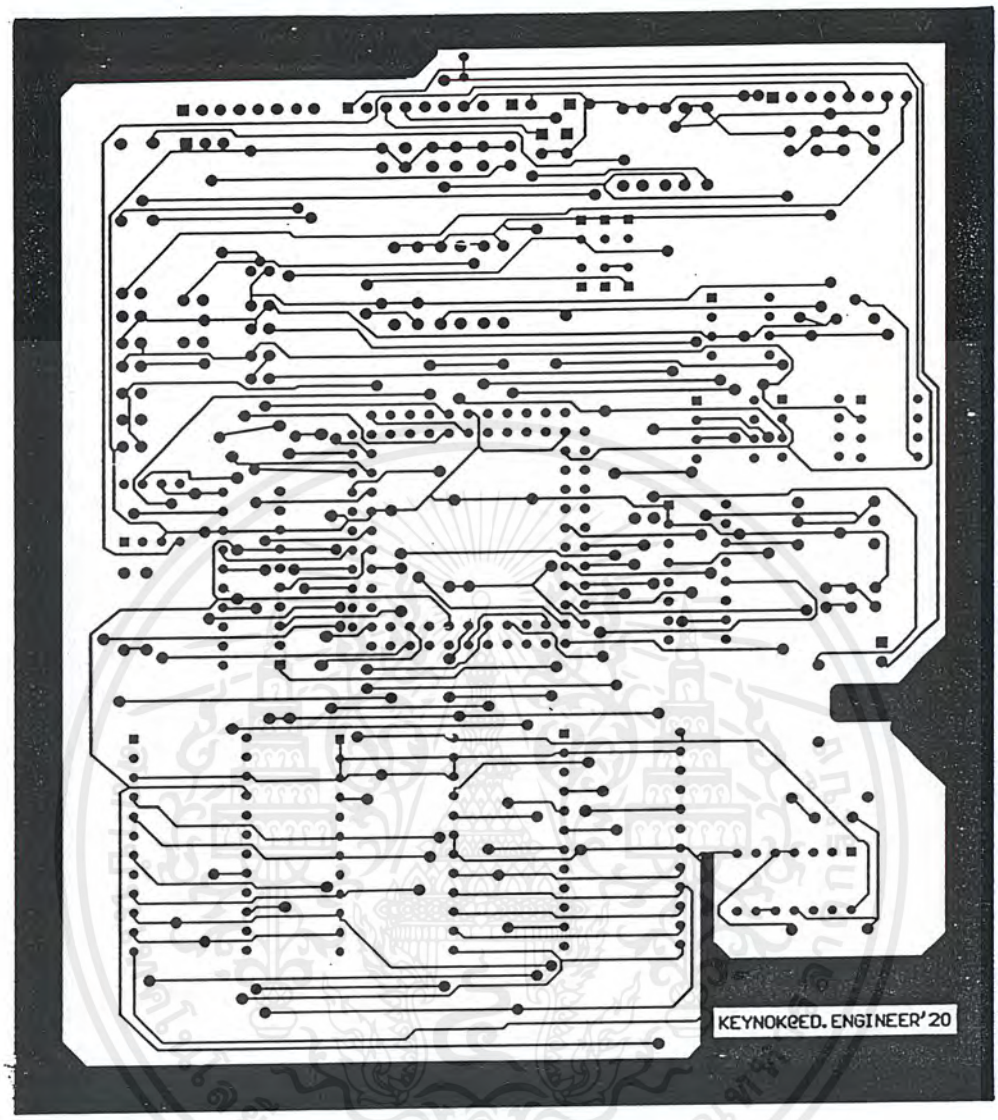
```
library IEEE;
use IEEE.std_logic_1164.all;

entity mul2_1E is
        port (
        d0: in STD_LOGIC;
        d1: in STD_LOGIC;
        s0: in STD_LOGIC;
        o: out STD_LOGIC
        );
end mul2_1E;

architecture mul2_1E_arch of mul2_1E is
signal o1 : std_logic;
signal o2 : std_logic;
begin
        o1 <= d0 and (not s0);
        o2 <= d1 and s0;
        o <= o1 or o2;
end mul2_1E_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity or2_b1 is
        port (
        a: in STD_LOGIC;
        b: in STD_LOGIC;
        f: out STD_LOGIC
        );
end or2_b1;

architecture or2_b1_arch of or2_b1 is
begin
```

```
        f <= a or (not b);
end or2_b1_arch;


library IEEE;
use IEEE.std_logic_1164.all;

entity ttl_74ls157 is
        port (
        A: in STD_LOGIC_VECTOR(4 downto 1);
        B: in STD_LOGIC_VECTOR(4 downto 1);
        G: in STD_LOGIC;
        SL: in STD_LOGIC;
        Y: out STD_LOGIC_VECTOR(4 downto 1)
        );
end ttl_74ls157;

architecture ttl_74ls157_arch of ttl_74ls157 is
begin
        process(G,SL,A,B)
        begin
                if (G = '1') then
                for i in 4 downto 1 Loop Y(i) <= '0';
                end loop;
                else if (G = '0') then
                if (SL = '0') then Y <= A ;
                else if (SL = '1') then Y <= B;
                end if;
                end if;
                end if;
                end if;
        end process;
end ttl_74ls157_arch;
```

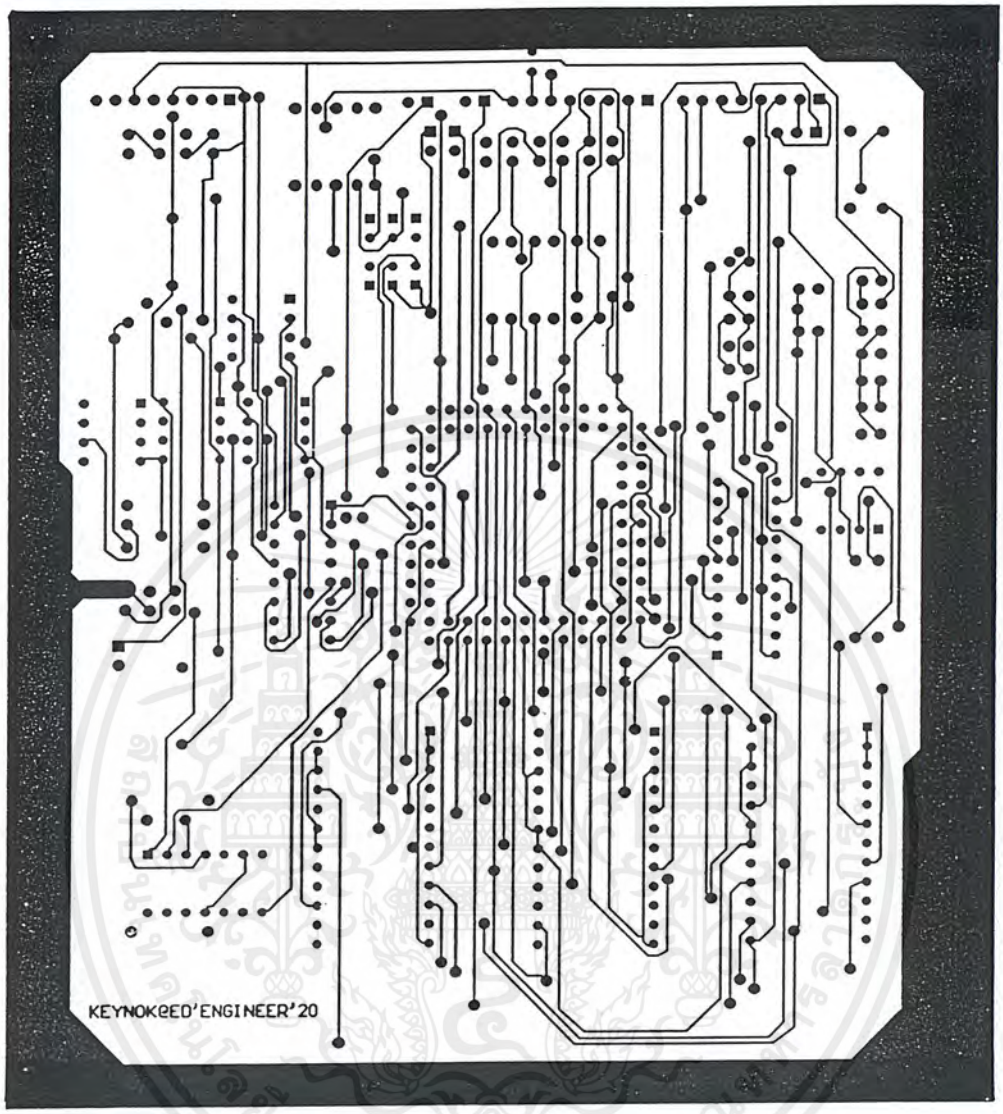
```
library IEEE;
use IEEE.std_logic_1164.all;

entity ttl_74ls91 is
        port (
        A: in STD_LOGIC;
        CLK: in STD_LOGIC;
        QH: out STD_LOGIC
);
end ttl_74ls91;

architecture ttl_74ls91_arch of ttl_74ls91 is
component RS_FF_2
        port (
        R: in STD_LOGIC;
        ck: in STD_LOGIC;
        S: in STD_LOGIC;
        Q: out STD_LOGIC;
        Qnot: out STD_LOGIC);
end component;
signal ck_bar : std_logic;
signal a_bar  : std_logic;
signal qs     : std_logic_vector(7 downto 0);
signal qs_bar : std_logic_vector(7 downto 0);
begin
        ck_bar <= not CLK;
        a_bar  <= not A;
stages : for i in 7 downto 0 generate
lowbit : if i = 0 generate
RS     : RS_FF_2  port map (a_bar,ck_bar,A,qs(0),qs_bar(0));
        end generate;
other  : if i /= 0 generate
RS     : RS_FF_2  port map (qs_bar(i-1),ck_bar,qs(i-1),qs(i),qs_bar(i));
        end generate;
```

```
end generate;
        QH <= qs(7);
end ttl_74ls91_arch;

library IEEE;
use IEEE.std_logic_1164.all;

entity RS_FF_2 is
  port (
    R: in STD_LOGIC;
    ck: in STD_LOGIC;
    S: in STD_LOGIC;
    Q: out STD_LOGIC;
    Qnot: out STD_LOGIC
  );
end RS_FF_2;

architecture RS_FF_2_arch of RS_FF_2 is
signal state : std_logic;
begin
        Q <= state;
        Qnot <= not state;
        process(ck,R,S)
        begin
                if (ck ='0' and ck'event)then
                if(R ='0' and S ='1') then
                 state <= '1';
                end if;
                if(R ='1' and S ='0') then
                state <= '0';
                end if;
                end if;
        end process;
end RS_FF_2_arch;
```

ภาคผนวก ข
แผ่นวงจรพิมพ์



รูปที่ ข.1 แผ่นวงจรพิมพ์ด้านบน



รูปที่ ข.2 แผ่นวงจรพิมพ์ด้านล่าง

ภาคผนวก ค
รายละเอียดข้อมูลและคุณสมบัติของอุปกรณ์

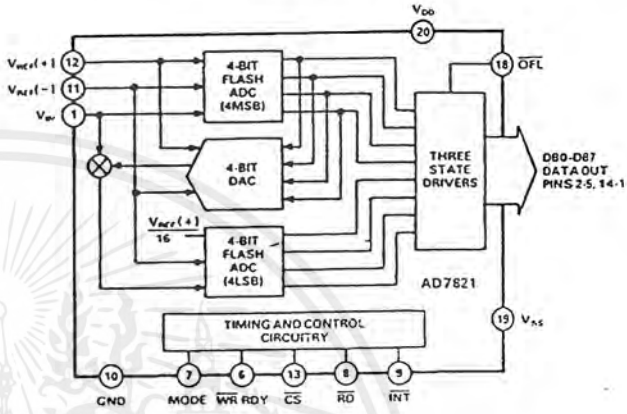
ANALOG DEVICES

LC²MOS High Speed, µP-Compatible 8-Bit ADC with Track/Hold Function

AD7821

FEATURES
Fast Conversion Time: 660 ns max
100 kHz Track-and-Hold Function
1 MHz Sample Rate
Unipolar and Bipolar Input Ranges
Ratiometric Reference Inputs
No External Clock
Extended Temperature Range Operation
Skinny 20-Pin DIPs, SOIC and 20-Terminal Surface Mount Packages

FUNCTIONAL BLOCK DIAGRAM



GENERAL DESCRIPTION

The AD7821 is a high speed, 8-bit, sampling, analog-to-digital converter that offers improved performance over the popular AD7820. It offers a conversion time of 660 ns (vs. 1.36 µs for the AD7820) and 100 kHz signal bandwidth (vs. 6.4 kHz). The sampling instant is better defined and occurs on the falling edge of $\overline{WR}$ or $\overline{RD}$. The provision of a V_{SS} pin (Pin 19) allows the part to operate from ±5 V supplies and to digitize bipolar input signals. Alternatively, for unipolar inputs, the V_{SS} pin can be grounded and the AD7821 will operate from a single +5 V supply, like the AD7820.

The AD7821 has a built-in track-and-hold function capable of digitizing full-scale signals up to 100 kHz max. It also uses a half-flash conversion technique that eliminates the need to generate a CLK signal for the ADC.

The AD7821 is designed with standard microprocessor control signals ($\overline{CS}$, $\overline{RD}$, $\overline{WR}$, RDY, $\overline{INT}$) and latched, three-state data outputs capable of interfacing to high speed data buses. An overflow output ($\overline{OFL}$) is also provided for cascading devices to achieve higher resolution.

The AD7821 is fabricated in Linear-Compatible CMOS (LC²MOS), an advanced, mixed technology process combining precision bipolar circuits with low power CMOS logic. The part features a low power dissipation of 50 mW.

PRODUCT HIGHLIGHTS

1. Fast Conversion Time
   The half-flash conversion technique, coupled with fabrication on Analog Devices' LC²MOS process, enables a very fast conversion time. The conversion time for the WR-RD mode is 660 ns, with 700 ns for the RD mode.
2. Built-In Track-and-Hold
   This allows input signals with slew rates up to 1.6 V/µs to be converted to 8-bits without an external track-and-hold. This corresponds to a 5 V peak-to-peak, 100 kHz sine wave signal.
3. Total Unadjusted Error
   The AD7821 features an excellent total unadjusted error figure of less than ±1 LSB over the full operating temperature range.
4. Unipolar/Bipolar Input Ranges
   The AD7821 is specified for single supply (+5 V) operation with a unipolar full-scale range of 0 to +5 V, and for dual supply (±5 V) operation with a bipolar input range of ±2.5 V. Typical performance characteristics are given for other input ranges.
5. Dynamic Specifications for DSP Users
   In addition to the traditional ADC specifications, the AD7821 is specified for ac parameters, including signal-to-noise ratio, distortion and slew rate.

AD7821–SPECIFICATIONS

V_{DD} = +5 V ± 5%, GND = 0 V. Unipolar Input Range: V_{SS} = GND, $V_{REF}(+)$ = 5 V, $V_{REF}(-)$ = GND. Bipolar Input Range: V_{SS} = –5 V ± 5%, $V_{REF}(+)$ = 2.5 V, $V_{REF}(-)$ = –2.5 V. These test conditions apply unless otherwise stated. All specifications T_{MIN} to T_{MAX} unless otherwise noted. Specifications apply for RD Mode (Pin 7 = 0 V).

Parameter	K Version[1]	B, T Versions	Units	Comments
UNIPOLAR INPUT RANGE				
Resolution[2]	8	8	Bits	
Total Unadjusted Error[3]	±1	±1	LSB max	
Minimum Resolution for which No Missing Codes are Guaranteed	8	8	Bits	
BIPOLAR INPUT RANGE				
Resolution[2]	8	8	Bits	
Zero Code Error	±1	±1	LSB max	
Full Scale Error	±1	±1	LSB max	
Signal-to-Noise Ratio (SNR)[3]	45	45	dB min	V_{IN} = 99.85 kHz Full-Scale Sine Wave with $f_{SAMPLING}$ = 500 kHz
Total Harmonic Distortion (THD)[3]	-50	-50	dB max	V_{IN} = 99.85 kHz Full-Scale Sine Wave with $f_{SAMPLING}$ = 500 kHz
Peak Harmonic or Spurious Noise[3]	-50	-50	dB max	V_{IN} = 99.85 kHz Full-Scale Sine Wave with $f_{SAMPLING}$ = 500 kHz
Intermodulation Distortion (IMD)[3]				fa (84.72 kHz) and fb (94.97 kHz) Full-Scale Sine Waves with $f_{SAMPLING}$ = 500 kHz
	-50	-50	dB max	Second Order Terms
	-50	-50	dB max	Third Order Terms
Slew Rate, Tracking[3]	1.6	1.6	V/µs max	
	2.36	2.36	V/µs typ	
REFERENCE INPUT				
Input Resistance	1.0/4.0	1.0/4.0	kΩ min/kΩ max	
$V_{REF}(+)$ Input Voltage Range	$V_{REF}(-)/V_{DD}$	$V_{REF}(-)/V_{DD}$	V min/V max	
$V_{REF}(-)$ Input Voltage Range	$V_{SS}/V_{REF}(+)$	$V_{SS}/V_{REF}(+)$	V min/V max	
ANALOG INPUT				
Input Voltage Range	$V_{REF}(-)/V_{REF}(+)$	$V_{REF}(-)/V_{REF}(+)$	V min/ max	
Input Leakage Current	±3	±3	µA max	–5 V ≤ V_{IN} ≤ +5 V
Input Capacitance	55	55	pF typ	
LOGIC INPUTS				
$\overline{CS}$, $\overline{WR}$, $\overline{RD}$				
V_{INH}	2.4	2.4	V min	
V_{INL}	0.8	0.8	V max	
I_{INH} ($\overline{CS}$, $\overline{RD}$)	1	1	µA max	
I_{INH} ($\overline{WR}$)	3	3	µA max	
I_{INL}	-1	-1	µA max	
Input Capacitance[4]	8	8	pF max	Typically 5 pF
MODE				
V_{INH}	3.5	3.5	V min	
V_{INL}	1.5	1.5	V max	
I_{INH}	200	200	µA max	50 µA typ
I_{INL}	-1	-1	µA max	
Input Capacitance[4]	8	8	pF max	Typically 5 pF
LOGIC OUTPUTS				
DB0-DB7, $\overline{OFL}$, $\overline{INT}$				
V_{OH}	4.0	4.0	V min	I_{SOURCE} = 360 µA
V_{OL}	0.4	0.4	V max	I_{SINK} = 1.6 mA
I_{OUT} (DB0-DB7)	±3	±3	µA max	Floating State Leakage
Output Capacitance[4] (DB0-DB7)	8	8	pF max	Typically 5 pF
RDY				
V_{OL}	0.4	0.4	V max	I_{SINK} = 2.6 mA
I_{OUT}	±3	±3	µA max	Floating State Leakage
Output Capacitance[4]	8	8	pF max	Typically 5 pF
POWER SUPPLY				
I_{DD}[5]	15	20	mA max	$\overline{CS}$ = $\overline{RD}$ = 0 V
I_{SS}	100	100	µA max	$\overline{CS}$ = $\overline{RD}$ = 0 V
Power Dissipation	50	50	mW typ	
Power Supply Sensitivity	±1/4	±1/4	LSB max	±1/16 LSB typ. V_{DD} = 4.75 V to 5.25 V. ($V_{REF}(+)$ = 4.75 V max for Unipolar Mode)

NOTES
[1]Temperature Ranges are as follows: K Version = –40°C to +85°C; B Version = –40°C to +85°C; T Version = –55°C to +125°C.
[2]1 LSB = 19.53 mV for both the unipolar (0 V to +5 V) and bipolar (–2.5 V to +2.5 V) input ranges.
[3]See Terminology.
[4]Sample tested at +25°C to ensure compliance.
[5]See Typical Performance Characteristics.

Specifications subject to change without notice.

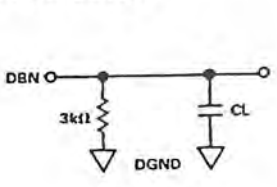
TIMING CHARACTERISTICS[1] (V_{DD} = +5 V ± 5%, V_{SS} = 0 V or –5 V ± 5%; Unipolar or Bipolar Input Range)

Parameter	Limit at +25°C (All Versions)	Limit at T_{MIN}, T_{MAX} (K, B Versions)	Limit at T_{MIN}, T_{MAX} (T Version)	Units	Conditions/Comments
t_{CSS}	0	0	0	ns min	$\overline{CS}$ to $\overline{RD}$/$\overline{WR}$ Setup Time
t_{CSH}	0	0	0	ns min	$\overline{CS}$ to $\overline{RD}$/$\overline{WR}$ Hold Time
t_{RDY}[2]	70	85	100	ns max	$\overline{CS}$ to RDY Delay. Pull-Up Resistor 5 kΩ.
t_{CRD}	700	875	975	ns max	Conversion Time (RD Mode)
t_{ACC0}[3]					Data Access Time (RD Mode)
	t_{CRD} + 25	t_{CRD} + 30	t_{CRD} + 35	ns max	C_L = 20 pF
	t_{CRD} + 50	t_{CRD} + 65	t_{CRD} + 75	ns max	C_L = 100 pF
t_{INTH}[2]	50	-	-	ns typ	$\overline{RD}$ to $\overline{INT}$ Delay (RD Mode)
	80	85	90	ns max	
t_{DH}[4]	15	15	15	ns min	Data Hold Time
	60	70	80	ns max	
t_P	350	425	500	ns min	Delay Time Between Conversions
t_{WR}	250	325	400	ns min	Write Pulse Width
	10	10	10	µs max	
t_{RD}	250	350	450	ns min	Delay Time between $\overline{WR}$ and $\overline{RD}$ Pulses
t_{READ1}	160	205	240	ns min	$\overline{RD}$ Pulse Width (WR-RD Mode, see Figure 12b) Determined by t_{ACC1}
t_{ACC1}[3]					Data Access Time (WR-RD Mode, see Figure 12b)
	160	205	240	ns max	C_L = 20 pF
	185	235	275	ns max	C_L = 100 pF
t_{RI}	150	185	220	ns max	$\overline{RD}$ to $\overline{INT}$ Delay
t_{INTL}[2]	380	-	-	ns typ	$\overline{WR}$ to $\overline{INT}$ Delay
	500	610	700	ns max	
t_{READ2}	65	75	85	ns min	$\overline{RD}$ Pulse Width (WR-RD Mode, see Figure 12a) Determined by t_{ACC2}
t_{ACC2}[3]					Data Access Time (WR-RD Mode, see Figure 12a)
	65	75	85	ns max	C_L = 20 pF
	90	110	130	ns max	C_L = 100 pF
t_{IHWR}[2]	80	100	120	ns max	$\overline{WR}$ to $\overline{INT}$ Delay (Stand-Alone Operation)
t_{ID}[3]					Data Access Time after $\overline{INT}$ (Stand-Alone Operation)
	30	35	40	ns max	C_L = 20 pF
	45	60	70	ns max	C_L = 100 pF

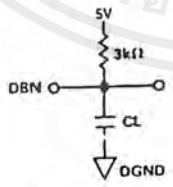
NOTES
[1]Sample tested at +25°C to ensure compliance. All input control signals are specified with tr = tf = 5 ns (10% to 90% of +5 V) and timed from a voltage level of 1.6 V.
[2]C_L = 50 pF.
[3]Measured with load circuits of Figure 1 and defined as the time required for an output to cross 0.8 V or 2.4 V.
[4]Defined as the time required for the data lines to change 0.5 V when loaded with the circuits of Figure 2.

Specifications subject to change without notice.

Test Circuits

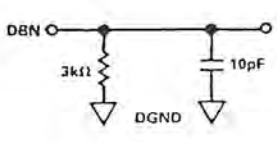


a. High Z to V_{OH}

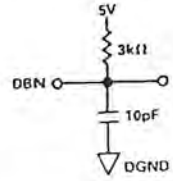


b. High Z to V_{OL}

Figure 1. Load Circuits for Data Access Time Test



a. V_{OH} to High Z



b. V_{OL} to High Z

Figure 2. Load Circuits for Data Hold Time Test

ORDERING GUIDE

Model[1]	Temperature Range	Total Unadjusted Error (LSB)	Package Option[2]
AD7821KN	–40°C to +85°C	±1 max	N-20
AD7821KP	–40°C to +85°C	±1 max	P-20A
AD7821KR	–40°C to +85°C	±1 max	R-20
AD7821BQ	–40°C to +85°C	±1 max	Q-20
AD7821TQ	–55°C to +125°C	±1 max	Q-20
AD7821TE	–55°C to +125°C	±1 max	E-20A

NOTES
[1]To order MIL-STD-883, Class B processed parts, add /883B to part number. Contact local sales office for military data sheet.
[2]E = Leadless Ceramic Chip Carrier; N = Plastic DIP; P = Plastic Leaded Chip Carrier; Q = Cerdip; R = SOIC.

AD7821

ABSOLUTE MAXIMUM RATINGS*

V_{DD} to GND	-0.3 V, + 7 V
V_{SS} to GND	+0.3 V, + 7 V
Digital Input Voltage to GND (Pins 6–8, 13)	-0.3 V, V_{DD} + 0.3 V
Digital Output Voltage to GND (Pins 2–5, 9, 14–18)	-0.3 V, V_{DD} + 0.3 V
V_{REF}(+) to GND	V_{SS} - 0.3 V, V_{DD} + 0.3 V
V_{REF}(–) to GND	V_{SS} - 0.3 V, V_{DD} + 0.3 V
V_{IN} to GND	V_{SS} - 0.3 V, V_{DD} + 0.3 V
Operating Temperature Range	
Commercial (K Version)	-40°C to +85°C
Industrial (B Version)	-40°C to +85°C
Extended (T Version)	-55°C to +125°C
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 secs)	+300°C
Power Dissipation (Any Package) to +75°C	450 mW
Derates above +75°C by	6 mW/°C

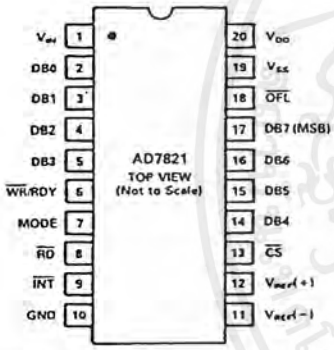
*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

CAUTION

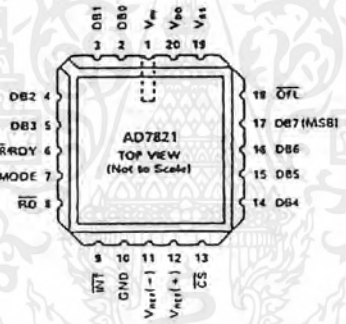
ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although the AD7821 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.

PIN CONFIGURATIONS

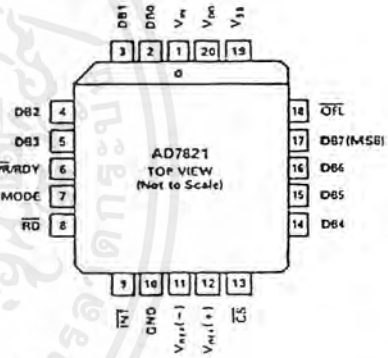
DIP AND SOIC



LCCC



PLCC



TERMINOLOGY

LEAST SIGNIFICANT BIT (LSB)

An ADC with 8-bit resolution can resolve one part in 2^8 (1/256 of full scale). For the AD7821 operating in either the unipolar or bipolar input range with 5 V full scale, one LSB is 19.53 mV.

TOTAL UNADJUSTED ERROR

This is a comprehensive specification which includes relative accuracy, offset error and full-scale error.

SLEW RATE

Slew Rate is the maximum allowable rate of change of input signal such that the digital sample values are not in error.

TOTAL HARMONIC DISTORTION

Total harmonic distortion is the ratio of the square root of the sum of the squares of the rms value of the harmonics to the rms value of the fundamental. For the AD7821, total harmonic distortion (THD) is defined as

$$20\ log\left[\frac{\sqrt{\left(V_2^2+V_3^2+V_5^2+V_6^2\right)}}{V_1}\right] dB$$

where V_1 is the rms amplitude of the fundamental and V_2, V_3, V_4, V_5, V_6, are the rms amplitudes of the individual harmonics.

INTERMODULATION DISTORTION

With inputs consisting of sine waves at two frequencies, fa and fb, any active device with nonlinearities will create distortion products, of order (m+n), at sum and difference frequencies of mfa+nfb, where m, n = 0, 1, 2, 3,- - - -. Intermodulation terms are those for which m or n is not equal to zero. For example, the second order terms include (fa + fb) and (fa – fb), and the third order terms include (2fa + fb), (2fa – fb), (fa + 2fb) and (fa - 2fb). For the AD7821 intermodulation distortion is calculated separately for both the second and third order terms.

SIGNAL-TO-NOISE RATIO

Signal-to-noise ratio (SNR) is measured signal-to-noise at the output of the ADC. The signal is the rms magnitude of the fundamental. Noise is the rms sum of all nonfundamental signals (excluding dc) up to half the sampling frequency. SNR is dependent on the number of quantization levels used in the digitization process. The theoretical SNR for a sine wave input is given by:

$$SNR = (6.02\ N + 1.76)\ dB$$

where N is the number of bits in the ADC. Thus, for an ideal 8-bit ADC, SNR = 50 dB.

PEAK HARMONIC OR SPURIOUS NOISE

Peak harmonic or spurious noise is the rms value of the largest nonfundamental frequency (excluding dc) up to half the sampling frequency to the rms value of the fundamental.

AD7821

ABSOLUTE MAXIMUM RATINGS*

V_{DD} to GND	-0.3 V, + 7 V
V_{SS} to GND	+0.3 V, + 7 V
Digital Input Voltage to GND (Pins 6-8, 13)	-0.3 V, V_{DD} + 0.3 V
Digital Output Voltage to GND (Pins 2-5, 9, 14-18)	-0.3 V, V_{DD} + 0.3 V
$V_{REF}(+)$ to GND	V_{SS} - 0.3 V, V_{DD} + 0.3 V
$V_{REF}(-)$ to GND	V_{SS} - 0.3 V, V_{DD} + 0.3 V
V_{IN} to GND	V_{SS} - 0.3 V, V_{DD} + 0.3 V
Operating Temperature Range	
Commercial (K Version)	-40°C to +85°C
Industrial (B Version)	-40°C to +85°C
Extended (T Version)	-55°C to +125°C
Storage Temperature Range	-65°C to +150°C
Lead Temperature (Soldering, 10 secs)	+300°C
Power Dissipation (Any Package) to +75°C	450 mW
Derates above +75°C by	6 mW/°C

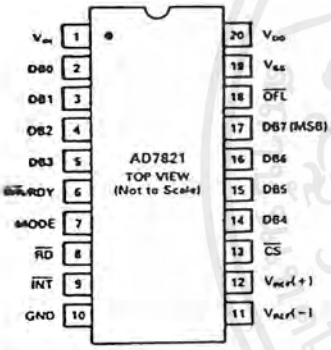
*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

CAUTION

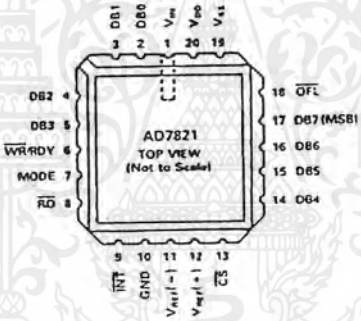
ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although the AD7821 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high energy electrostatic discharges. Therefore, proper ESD precautions are recommended to avoid performance degradation or loss of functionality.

PIN CONFIGURATIONS

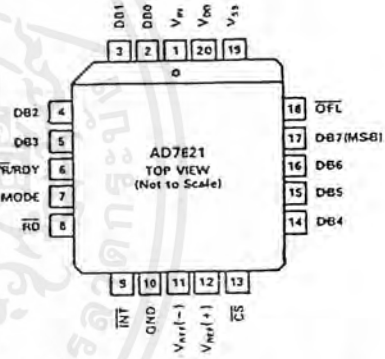
DIP AND SOIC



LCCC



PLCC



TERMINOLOGY

LEAST SIGNIFICANT BIT (LSB)

An ADC with 8-bit resolution can resolve one part in 2^8 (1/256 of full scale). For the AD7821 operating in either the unipolar or bipolar input range with 5 V full scale, one LSB is 19.53 mV.

TOTAL UNADJUSTED ERROR

This is a comprehensive specification which includes relative accuracy, offset error and full-scale error.

SLEW RATE

Slew Rate is the maximum allowable rate of change of input signal such that the digital sample values are not in error.

TOTAL HARMONIC DISTORTION

Total harmonic distortion is the ratio of the square root of the sum of the squares of the rms value of the harmonics to the rms value of the fundamental. For the AD7821, total harmonic distortion (THD) is defined as

$$20\ log\left[\frac{\sqrt{\left(V_2^2+V_3^2+V_5^2+V_6^2\right)}}{V_1}\right] dB$$

where V_1 is the rms amplitude of the fundamental and V_2, V_3, V_4, V_5, V_6, are the rms amplitudes of the individual harmonics.

INTERMODULATION DISTORTION

With inputs consisting of sine waves at two frequencies, fa and fb, any active device with nonlinearities will create distortion products, of order (m+n), at sum and difference frequencies of mfa+nfb, where m, n = 0, 1, 2, 3,- - - -. Intermodulation terms are those for which m or n is not equal to zero. For example, the second order terms include (fa + fb) and (fa - fb), and the third order terms include (2fa + fb), (2fa - fb), (fa + 2fb) and (fa - 2fb). For the AD7821 intermodulation distortion is calculated separately for both the second and third order terms.

SIGNAL-TO-NOISE RATIO

Signal-to-noise ratio (SNR) is measured signal-to-noise at the output of the ADC. The signal is the rms magnitude of the fundamental. Noise is the rms sum of all nonfundamental signals (excluding dc) up to half the sampling frequency. SNR is dependent on the number of quantization levels used in the digitization process. The theoretical SNR for a sine wave input is given by:

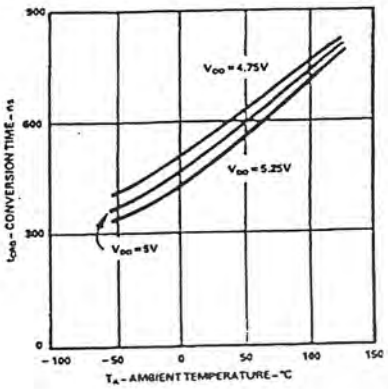
$$SNR = (6.02\ N + 1.76)\ dB$$

where N is the number of bits in the ADC. Thus, for an ideal 8-bit ADC, SNR = 50 dB.

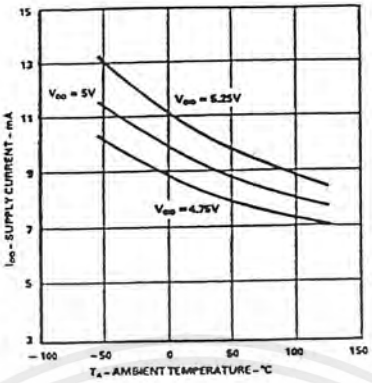
PEAK HARMONIC OR SPURIOUS NOISE

Peak harmonic or spurious noise is the rms value of the largest nonfundamental frequency (excluding dc) up to half the sampling frequency to the rms value of the fundamental.

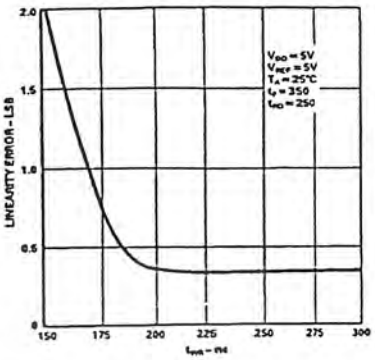
Typical Performance Curves–AD7821



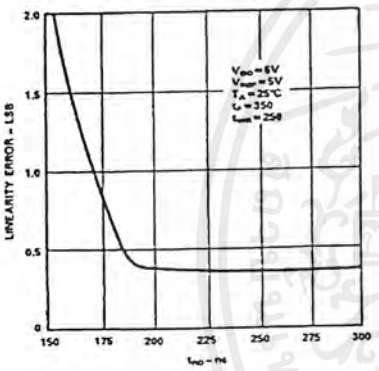
Conversion Time (RD Mode) vs. Temperature



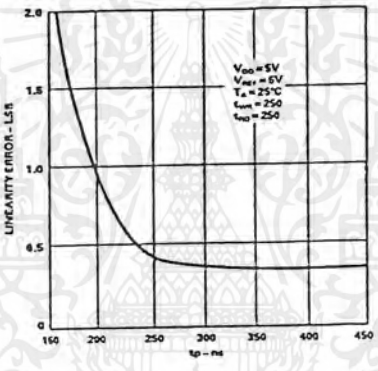
Power Supply Current vs. Temperature (Not Including Reference Ladder)



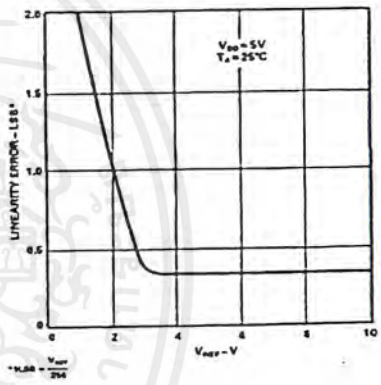
Accuracy vs. t_{WR}



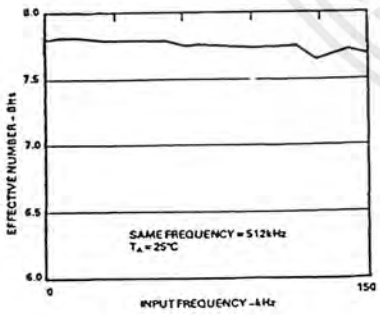
Accuracy vs. t_{RD}



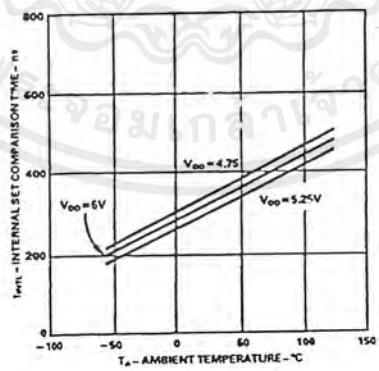
Accuracy vs. t_P



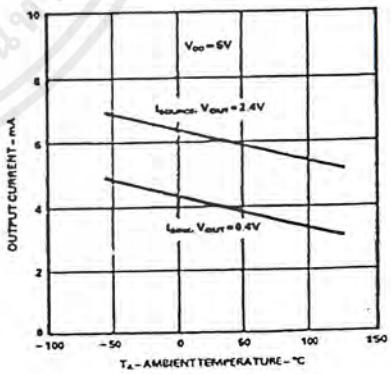
Accuracy vs. V_{REF} [$V_{REF} = V_{REF}(+) - V_{REF}(-)$]



Effective Number of Bits vs. Input Signal (±2.5 V) Frequency



t_{INTL}, Internal Time Delay vs. Temperature



Output Current vs. Temperature

PIN FUNCTION DESCRIPTION

Pin	Mnemonic	Description
1	V_{IN}	Analog Input: Range $V_{REF}(-) \le V_{IN} \le V_{REF}(+)$
2	DB0	Three-State Data Output (LSB).
3-5	DB1-DB3	Three-State Data Outputs.
6	$\overline{WR}$/RDY	WRITE control input/READY status output. See Digital Interface section.
7	MODE	Mode Selection Input. It determines whether the device operates in the WR-RD or RD mode. This input is internally pulled low through a 50 µA current source. See Digital Interface section.
8	$\overline{RD}$	READ Input. $\overline{RD}$ must be low to access data from the part. See Digital Interface section.
9	$\overline{INT}$	INTERRUPT Output. $\overline{INT}$ going low indicates that the conversion is complete. $\overline{INT}$ returns high on the rising edge of $\overline{CS}$ or $\overline{RD}$. See Digital Interface section.
10	GND	Ground.
11	$V_{REF}(-)$	Lower limit of reference span. Range: $V_{SS} \le V_{REF}(-) \le V_{REF}(+)$.
12	$V_{REF}(+)$	Upper limit of reference span. Range: $V_{REF}(-) < V_{REF}(+) \le V_{DD}$.
13	$\overline{CS}$	Chip Select Input. The device is selected when this input is low.
14-16	DB4-DB6	Three-State Data Outputs.
17	DB7	Three-State Data Output (MSB).
18	$\overline{OFL}$	Overflow Output. If the analog input is higher than ($V_{REF}(+)$ - 1/2 LSB), $\overline{OFL}$ will be low at the end of conversion. It is a non-three-state output which can be used to cascade 2 or more devices to increase resolution.
19	V_{SS}	Negative supply voltage. V_{SS} = 0 V; Unipolar Operation. V_{SS} = -5 V; Bipolar Operation.
20	V_{DD}	Positive supply voltage, +5 V.

CIRCUIT INFORMATION

BASIC DESCRIPTION

The AD7821 uses a half flash conversion technique (see Functional Block Diagram), whereby two 4-bit flash ADCs are used to achieve an 8-bit result. Each 4-bit flash ADC contains 15 comparators, which compare an unknown input voltage to the reference ladder, to achieve a 4-bit result. The MS (most significant) flash ADC converts an unknown analog input voltage (V_{IN}) to provide the 4 MS data bits. An internal DAC, driven by the 4 MS data bits, then recreates an analog approximation of the input voltage. The DAC output voltage is subtracted from the analog input, and the difference is converted by the LS (least significant) ADC to provide the 4 LS data bits. The MS flash ADC also has one additional comparator to detect overrange on the analog input.

OPERATING SEQUENCE

The AD7821 has two operating modes. The RD mode allows a conversion to be started and data to be read with a single, extended, READ operation (i.e., $\overline{CS}$ and $\overline{RD}$ are taken low). The conversion process is timed out by internal one-shots. The WR-RD mode uses $\overline{WR}$ to start a conversion and $\overline{RD}$ to read the data and allows the conversion timing to be externally controlled. The operating sequence for the WR-RD mode is shown in Figure 3.

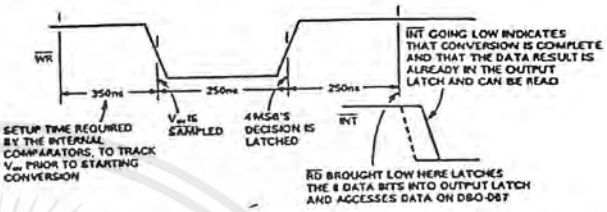


Figure 3. Operating Sequence (WR-RD Mode)

A conversion is initiated and the analog input signal (V_{IN}) sampled on the falling edge of $\overline{WR}$ (falling edge of $\overline{RD}$, RD mode). A setup time (t_P, delay time between conversions) of 350 ns is required prior to this falling edge. See Digital Interface section for more details. When $\overline{WR}$ is low, the internal MS (most significant) ADC compares the sampled analog input with the reference ladder to provide the 4 MS data bits. A minimum of 250 ns is required for this comparison. On the rising edge of $\overline{WR}$, the MS data result is latched internally and the LS (least significant) conversion begins, to yield the 4 LS data bits. $\overline{INT}$ goes low typically 380 ns after the rising edge of $\overline{WR}$. This indicates the LS conversion is complete and that both the LS and MS data results are latched into the output buffer. $\overline{RD}$ going low then enables the output data. If a faster conversion time is required, the $\overline{RD}$ line can be brought low 250 ns after $\overline{WR}$ goes high. This latches both the LS and MS data bits and outputs the conversion result on DB0-DB7.

REFERENCE AND INPUT

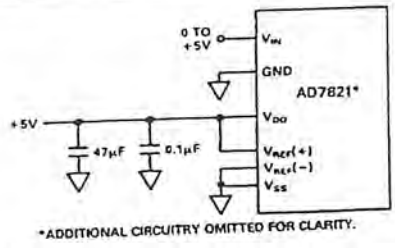
The $V_{REF}(-)$ and $V_{REF}(+)$ reference inputs on the AD7821 are fully differential and define the zero and full-scale input range of the ADC. The transfer characteristic of the part is defined by the integer value of the following expression:

$$Data\ (LSBs) = 256\left[\frac{V_{IN} - V_{REF}(-)}{V_{REF}(+) - V_{REF}(-)}\right] + 0.5$$

As a result, the analog input (V_{IN}) Of the device can easily be set up to provide both unipolar and bipolar operation. The data output code for unipolar and bipolar operation is Natural Binary and Offset Binary, respectively.

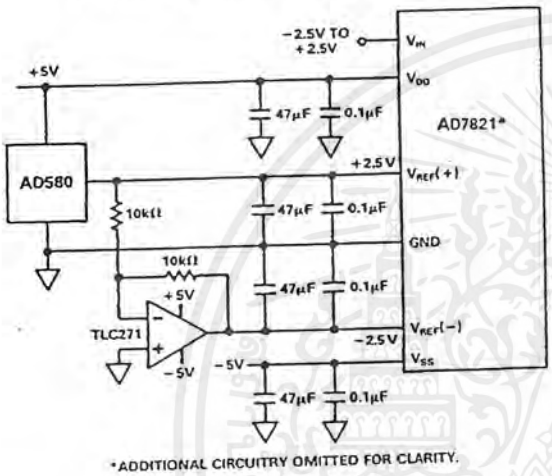
The span of the analog input voltage can easily be varied. By reducing the reference span, $V_{REF}(+) - V_{REF}(-)$, to less than 5 V the sensitivity of the converter can be increased (i.e., if V_{REF} = 2 V then 1 LSB = 7.8 mV). The reference flexibility also allows the input span for unipolar operation to be offset from zero ($V_{REF}(-)$ > GND). Additionally, the input/reference arrangement facilitates ratiometric operation.

Figures 4 and 5 show some configurations which are possible. For minimum noise a 47 µF capacitor in parallel with a 0.1 µF capacitor should be connected between the reference inputs and GND.



*ADDITIONAL CIRCUITRY OMITTED FOR CLARITY.

Figure 4. Power Supply as Reference. Unipolar Operation (0 to + 5 V)



*ADDITIONAL CIRCUITRY OMITTED FOR CLARITY.

Figure 5. External Reference. Bipolar Operation (–2.5 V to +2.5 V)

INPUT CURRENT

The analog input of the AD7821 behaves somewhat differently to conventional A/D converters. This is due to the ADC's sampled data comparators, which take varying amounts of input current depending on the cycle of the converter.

The equivalent input circuit of the AD7821 is shown in Figure 6. When a conversion ends (e.g., falling edge of $\overline{INT}$, WR-RD mode, $t_{RD} > t_{INTL}$) all the input switches are closed and V_{IN} is connected to the comparators of the internal LS and MS ADCs. Therefore, V_{IN} is connected to 31 one-pF input capacitors simultaneously .

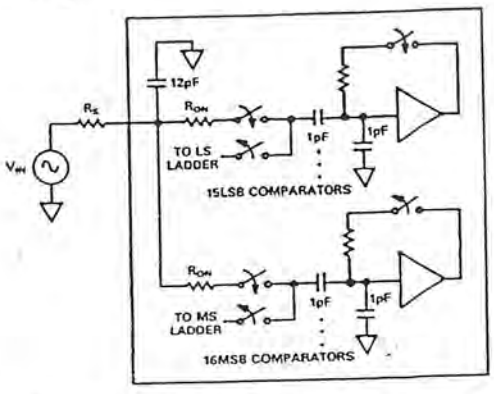


Figure 6. AD7821 Equivalent Input Circuit

The input capacitors must charge to the input voltage through the on resistance of the analog switches (about 2 kΩ to 5 kΩ). In addition, about 12 pF of input stray capacitance must be charged.

The analog input can be modeled as an equivalent RC network as shown in Figure 7. As R_S (source impedance) increases, the input capacitance takes longer to charge.

The comparators track the analog input between conversions. A minimum delay time (t_P) of 350 ns is required between conversions to allow for voltage source settling and comparator tracking time. This allows input time constants of 50 ns without settling time problems. Typical total input capacitance values of 55 pF allow R_S to be 0.9 kΩ without lengthening t_P to give V_{IN} more time to settle.

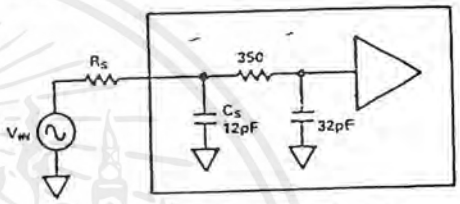


Figure 7. RC Network Model

INPUT TRANSIENTS

Transients on the analog input signal caused by charging current flowing into V_{IN} will not normally degrade the ADC's performance. In effect, the AD7821 does not "look" at the input when these transients occur. The comparators' inputs track V_{IN} and are not sampled until the falling edge of $\overline{WR}$ (WR-RD Mode) or $\overline{RD}$ (RD Mode), so at least 350 ns (t_P) is provided to charge the ADC's input capacitance. It is, therefore, not necessary to filter out these transients with an external capacitor at the V_{IN} terminal.

INHERENT TRACK-AND-HOLD

A major benefit of the AD7821's input structure is its ability to measure a variety of high-speed signals without the help of an external track-and-hold. Any ADC which does not have a built-in track-and-hold, regardless of its speed, requires the analog input to remain stable to at least 1/2 LSB for the duration of the conversion to maintain full accuracy. This requires the use of a track-and-hold whenever the input is a high-speed signal. The AD7821's sampled-data comparators, by nature of their input switching, inherently accomplish this track-and-hold function. Although the conversion time for the AD7821 is 660 ns (WR-RD mode, $t_{WR} + t_{RD} + t_{ACC1}$), the time for which V_{IN} must be stable to 1/2 LSB is much smaller. The AD7821 tracks V_{IN} between conversions only, and its value on the falling edge of $\overline{WR}$ or $\overline{RD}$ in the WR-RD or RD modes, respectively, is the measured value.

SINUSOIDAL INPUTS

The bandwidth of the built-in track-and-hold is 100 kHz max (150 kHz typ, 5 V p-p). This is limited by the analog bandwidth of the comparators and timing skew between the comparator switches. This means that the analog input frequency can be up to 100 kHz without the aid of an external track-and-hold. The Nyquist criterion requires that the sampling rate be at least twice the input frequency (i.e., ≥2 × 100 kHz). This requires an ideal antialiasing filter with an infinite roll-off. To ease the prob-

lem of antialiasing filter design, the sampling rate is usually set much greater than the Nyquist criterion. The maximum sampling rate (f_{MAX}) for the AD7821 in the WR-RD mode, ($t_{RD} < t_{INTL}$) can be calculated as follows:

$$f_{MAX} = \frac{1}{t_{WR} + t_{RD} + t_{RI} + t_P}$$

$$f_{MAX} = \frac{1}{0.25E-6 + 0.25E-6 + 0.15E-6 + 0.35E-6}$$

t_{WR} = *Write Pulse Width*

t_{RD} = *Delay Time between* $\overline{WR}$ *and* $\overline{RD}$ *Pulses*

t_{RI} = $\overline{RD}$ *to* $\overline{INT}$ *Delay*

t_P = *Delay Time between Conversions*

This permits a maximum sampling rate for the AD7821 of 1 MHz, which is much greater than the Nyquist criterion for sampling a 100 kHz analog input signal.

DIGITAL SIGNAL PROCESSING APPLICATIONS

In Digital Signal Processing (DSP) application areas like voice recognition, echo cancellation and adaptive filtering, the dynamic characteristics (Signal-to-Noise Ratio, Harmonic Distortion, Intermodulation Distortion) of an ADC are critical. Since the AD7821 is a very fast ADC with a built-in track-and-hold function, it is specified dynamically as well as with standard dc specifications (Total Unadjusted Error, etc.).

SIGNAL-TO-NOISE RATIO AND DISTORTION

The dynamic performance of the AD7821 is evaluated by applying a very low distortion sine wave signal to the analog input (V_{IN}) which is then sampled at a 512 kHz sampling rate. A Fast Fourier Transform (FFT) plot is then generated from which Signal-to-Noise Ratio (SNR) and harmonic distortion data are obtained.

Figure 8 shows a 2048 point FFT plot of the AD7821 with an input signal of 100.25 kHz. The SNR is 49.1 dB. It should be noted that the harmonics are taken into account when calculating the SNR. The theoretical relationship between SNR and resolution (N) is expressed by the following equation:

$$SNR = (6.02\ N + 1.76)\ dB \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

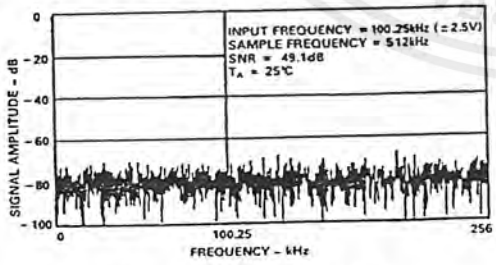


Figure 8. AD7821 FFT Plot

EFFECTIVE NUMBER OF BITS

By working backwards from Equation (1) it is possible to get a measure of ADC performance expressed in effective number of bits (N). A plot of the effective number of bits versus input frequency is given in the Typical Performance Characteristics section. The effective number of bits typically falls between 7.7 and 7.9, corresponding to SNR figures of 48.1 and 49.7 dB.

INTERMODULATION DISTORTION

For intermodulation distortion (IMD), an FFT plot consisting of very low distortion sine waves at two frequencies is generated by sampling an analog input applied to the ADC. Figure 9 shows a 2048 point plot for IMD.

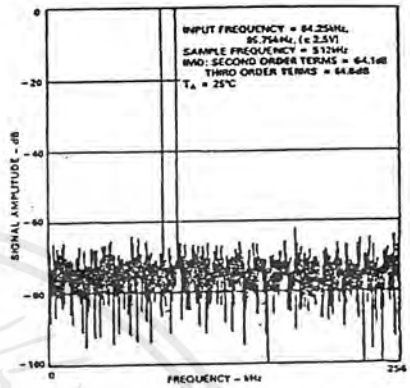


Figure 9. FFT Plot for IMD

HISTOGRAM PLOT

When a sine wave of specified frequency is applied to the V_{IN} input of the AD7821 and several thousand samples are taken, it is possible to plot a histogram showing the frequency of occurrence of each of the 256 ADC codes. A perfect ADC produces a probability density function described by the equation:

$$P(V) = \frac{1}{\pi(A^2 - V^2)^{1/2}}$$

where A is the peak amplitude of the sine wave and $P(V)$ is the probability of occurrence at a voltage V

If a particular step is wider than the ideal 1 LSB width, then the code associated with that step will accumulate more counts than for the code for an ideal step. Likewise, a step narrower than the ideal width will have fewer counts. Missing codes are easily seen because a missing code means zero counts for a particular code. The absence of large spikes in the plot indicates small differential nonlinearity.

Figure 10 shows a histogram plot for the AD7821, which corresponds very well with the ideal shape. The plot indicates very small differential nonlinearity and no missing codes for an input frequency of 100.25 kHz.

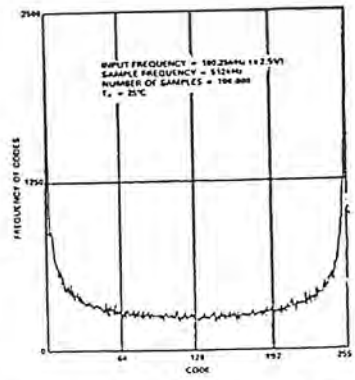


Figure 10. AD7821 Histogram Plot

In digital signal processing applications, where the AD7821 is used to sample ac signals, it is essential that the signal sampling occurs at exactly equal intervals. This minimizes errors due to sampling uncertainty or jitter. A precise timer or clock source, to start the ADC conversion process, is the best method of generating equidistant sampling intervals.

The two modes of operation given in the data sheet are suitable for DSP applications because the sampling instant of the AD7821 is well defined. V_{IN} is sampled on the falling edge of $\overline{WR}$ or $\overline{RD}$ in the WR-RD or RD modes, respectively.

DIGITAL INTERFACE

The AD7821 has two basic interface modes which are determined by the status of the MODE pin. When this pin is low, the converter is in the RD mode, with this pin high, the AD7821 is set up for the WR-RD mode.

The RD mode is designed for microprocessors that can be driven into a WAIT state. A READ operation (i.e., $\overline{CS}$ and $\overline{RD}$ are taken low) starts a conversion and data is read when the conversion is complete. The WR-RD mode does not require microprocessor WAIT states. A WRITE operation (i.e., $\overline{CS}$ and $\overline{WR}$ are taken low) initiates a conversion, and a READ operation reads the result when the conversion is complete.

RD Mode (MODE = 0)

The timing diagram for the RD mode is shown in Figure 11. This mode is intended for use with microprocessors which have a WAIT state facility, whereby a READ instruction cycle can be extended to accommodate slow memory devices. A conversion is started by taking $\overline{CS}$ and $\overline{RD}$ low (READ operation). Both $\overline{CS}$ and $\overline{RD}$ are then kept low until output data appears.

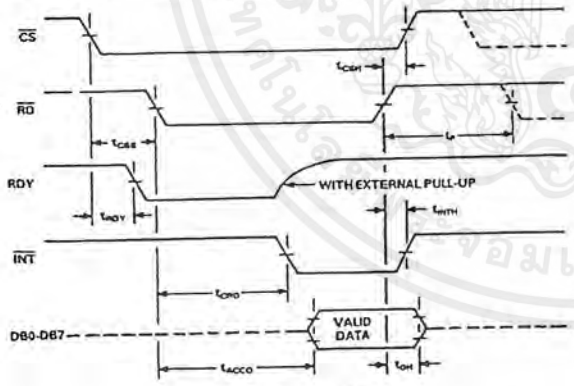


Figure 11. RD Mode

In this mode, Pin 6 of the AD7821 is configured as a status output, RDY. This RDY output can be used to drive the processor READY or WAIT input. It is an open drain output (no internal pull-up device) which goes low after the falling edge of $\overline{CS}$ and goes high impedance at the end of conversion. An $\overline{INT}$ line is also provided which goes low when a conversion is complete. $\overline{INT}$ returns high on the rising edge of $\overline{CS}$ or $\overline{RD}$.

WR-RD Mode (MODE = 1)

In the WR-RD mode, Pin 6 is configured as a WRITE ($\overline{WR}$) input for the AD7821. With $\overline{CS}$ low, conversion is initiated on the falling edge of $\overline{WR}$. Two options exist for reading data from the converter.

In the first of these options the processor waits for the $\overline{INT}$ status line to go low before reading the data (see Figure 12a). $\overline{INT}$ typically goes low within 380 ns after the rising edge of $\overline{WR}$. It indicates that conversion is complete and that the data result is in the output latch. With $\overline{CS}$ low, the data outputs (DB0–DB7) are activated when $\overline{RD}$ goes low. $\overline{INT}$ is reset by the rising edge of $\overline{RD}$ or $\overline{CS}$.

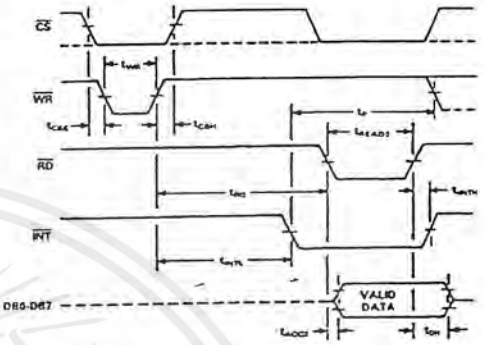


Figure 12a. WR-RD Mode ($t_{RD} > t_{INTL}$)

The alternative option can be used to shorten the conversion time. This is a method for bypassing the internal time-out circuit. The $\overline{INT}$ line is ignored and $\overline{RD}$ can be brought low 250 ns after the rising edge of $\overline{WR}$. In this case $\overline{RD}$ going low transfers the data result into the output latch and activates the data output (DB0–DB7). $\overline{INT}$ is driven low on the falling edge of $\overline{RD}$ and is reset on the rising edge of $\overline{RD}$ or $\overline{CS}$. The timing for this interface is shown in Figure 12b.

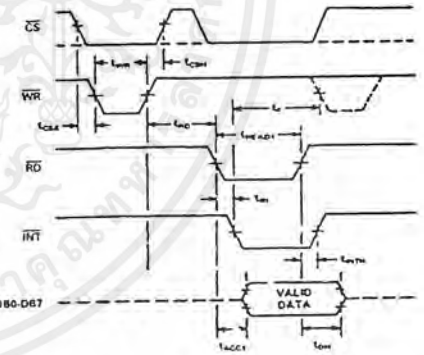


Figure 12b. WR-RD Mode ($t_{RD} < t_{INTL}$)

The AD7821 can also be used in stand-alone operation in the WR-RD mode. $\overline{CS}$ and $\overline{RD}$ are tied low, and a conversion is initiated by bringing $\overline{WR}$ low. Output data is valid 530 ns ($t_{INTL} + t_{ID}$) after the rising edge of $\overline{WR}$. The timing diagram for this mode is shown in Figure 13.

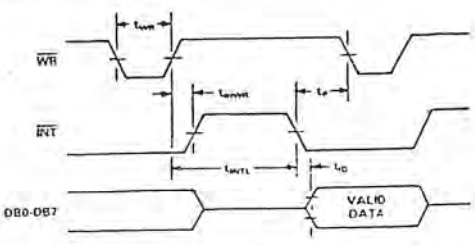


Figure 13. WR-RD Mode Stand-Alone Operation, $\overline{CS} = \overline{RD} = 0$

AD7821

MICROPROCESSOR INTERFACING

The AD7821 is designed for easy interfacing to microprocessors as a memory mapped peripheral or an I/O device. This reduces to a minimum the amount of external logic required for interfacing.

AD7821 - 68008 INTERFACE

Figure 14 shows an AD7821 interface to the 68008 microprocessor. The ADC is configured for the RD interface mode. This means that one read instruction starts a conversion and reads the result when the conversion is completed. The read cycle is stretched out over the entire conversion period by taking the $\overline{INT}$ line back to the $\overline{DTACK}$ input of the 68008. Starting a conversion and reading the relevant data consists of a <MOVE B Dn, addr> instruction, where addr is the decoded ADC address and Dn is the data register into which the result is placed.

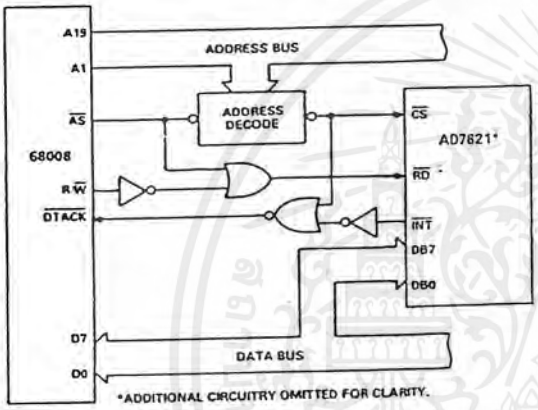


Figure 14. AD7821 to 68008 Interface

AD7821 - 8088 INTERFACE

A typical interface to the 8088 is shown in Figure 15. The AD7821 is configured for the RD interface mode. One read instruction starts a conversion and reads the result. The read cycle is stretched out over the entire conversion period by taking the RDY line back to the READY input of the 8088. Starting a conversion and reading the result consists of a <MOV AX, (addr)> instruction, where addr is the decoded ADC address and AX is the 8088 data register into which the conversion result is placed.

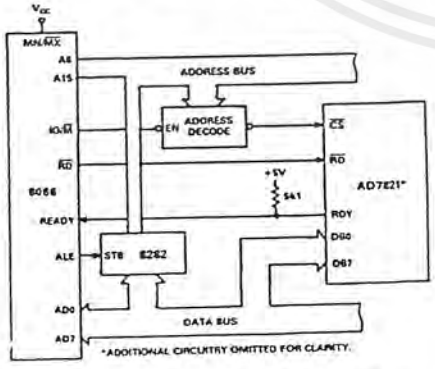


Figure 15. AD7821 to 8088 Interface

AD7821 - TMS32010 INTERFACE

A typical interface to the TMS32010 is shown in Figure 16. The AD7821 is mapped at a port address and the interface is designed for the maximum TMS32010 clock frequency of 20 MHz. In this case, the AD7821 is configured in the WR-RD interface mode. This means that a write instruction starts a conversion and a read instruction reads the result when the conversion is completed. A precise timer or clock source is used to start a conversion in applications requiring equidistant sampling intervals. The scheme used, whereby the AD7821 generates an interrupt to the TMS32010, is limited in that it does not allow the AD7821 to be sampled at its maximum rate. This is because the time between samples has to be long enough to allow the TMS32010 to service its interrupt and read data from the AD7821. Constant interruption of the TMS32010 by the AD7821, every time the ADC completes a conversion, is not a very efficient use of the processor time. To overcome these problems, some buffer memory or FIFO could be placed between the AD7821 and the TMS32010. The $\overline{INT}$ line of the AD7821 could be used to trigger a pulse which drives its $\overline{CS}$ and $\overline{RD}$ lines and places the AD7821 data into a FIFO or buffer memory. The microprocessor can then read a batch of data from the FIFO or buffer memory at some convenient time. Reading data from the AD7821, after an $\overline{INT}$ has been received, consists of <IN A, PA> instruction (PA is the decoded ADC address).

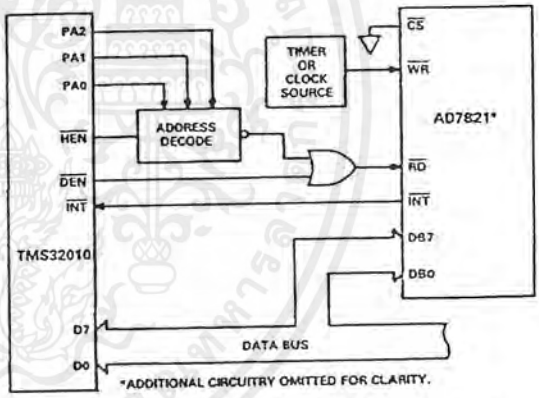


Figure 16. AD7821 to TMS32010 Interface

AD7821 - 8051 INTERFACE

Figure 17 shows the AD7821 interface to the 8051 microcomputer. The AD7821 is configured in the WR-RD interface mode and is connected to the 8051 ports. The processor starts conversion and then polls $\overline{INT}$, until it goes low, before reading the conversion result. Data is read from the AD7821 by using the <MOV A, 90H> instruction (90H is the address for Port 1).

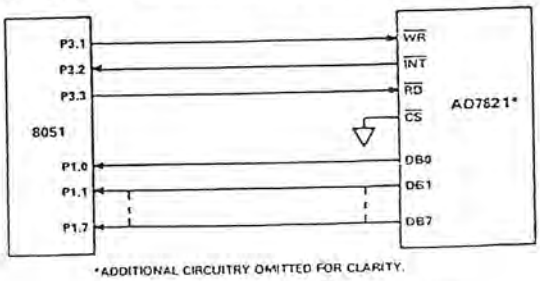


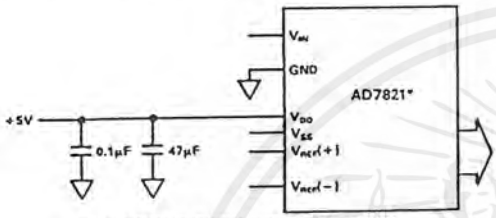
Figure 17. AD7821 to 8051 Interface

APPLYING THE AD7821

The AD7821 is specified for a unipolar input range of 0 to +5 V and a bipolar input range of -2.5 V to +2.5 V. The $V_{REF}(-)$ and $V_{REF}(+)$ voltages required for these input ranges are outlined below. See the Typical Performance Characteristics section for operation with unspecified input voltage ranges.

UNIPOLAR OPERATION

Figure 18 gives the configuration and reference voltages required for 0 V to +5 V operation. The nominal transfer characteristic for this input range is shown in Figure 19. The output code is Natural Binary with 1 LSB = (5/256) V = 19.5 mV.



$V_{REF}(+)$	$V_{REF}(-)$	V_{SS}	V_{IN}	RANGE
+5V	GND	GND	UNIPOLAR	0 to +5V
+2.5V	−2.5V	−5V	BIPOLAR	−2.5V to +2.5V

Figure 18. AD7821 Unipolar/Bipolar Operation

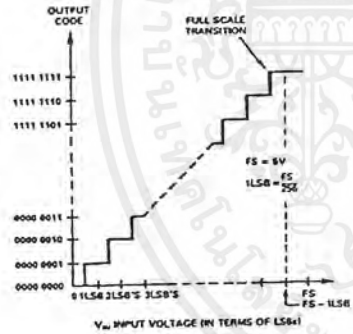


Figure 19. Nominal Transfer Characteristic for Unipolar (0 V to +5 V) Operation

BIPOLAR OPERATION

Figure 18 gives the configuration and reference voltages required for -2.5 V to +2.5 V operation. The nominal transfer characteristic for this input range is shown in Figure 20. The output code is Offset Binary with 1 LSB = ([+2.5 - (-2.5)]/256) V = 19.5 mV.

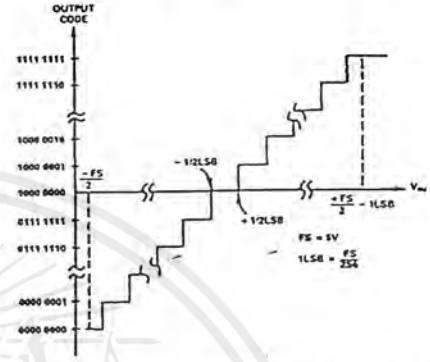


Figure 20. Nominal Transfer Characteristic for Bipolar (-2.5 V to +2.5 V) Operation

16-CHANNEL TELECOM A/D CONVERTER

The fast sampling rate (1 MHz) and bipolar operation of the AD7821 makes it useful in Telecom applications for sampling a number of input channels using a multiplexer. Figure 21 shows a circuit for such an application.

The maximum signal frequency required for acceptable quality in Telecom applications is 3 kHz. The circuit given in Figure 21 permits each of the 16-input channels to be sampled at a rate of 16 kHz maximum. The sampling rate takes account of such multiplexer parameters as t_{ON}, settling time etc. The circuit also eases the problem of the antialiasing filter design by sampling at a rate much greater than that required by the Nyquist criterion.

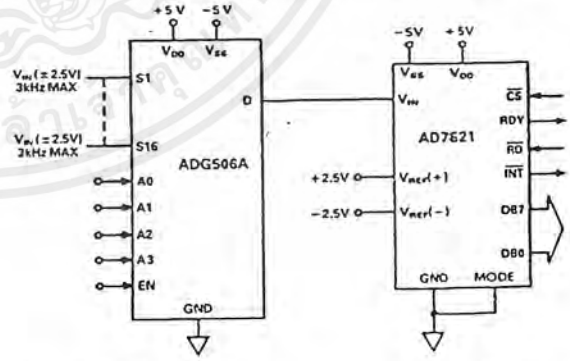


Figure 21. 16-Channel Telecom A/D Converter System

AD7821

SIMULTANEOUS SAMPLING A/D CONVERTERS

The AD7821's inherent track-and-hold and well-defined sampling instant makes it useful, in such applications as sonar, where a number of input channels are required to be sampled simultaneously. Figure 22 shows a circuit for such an application.

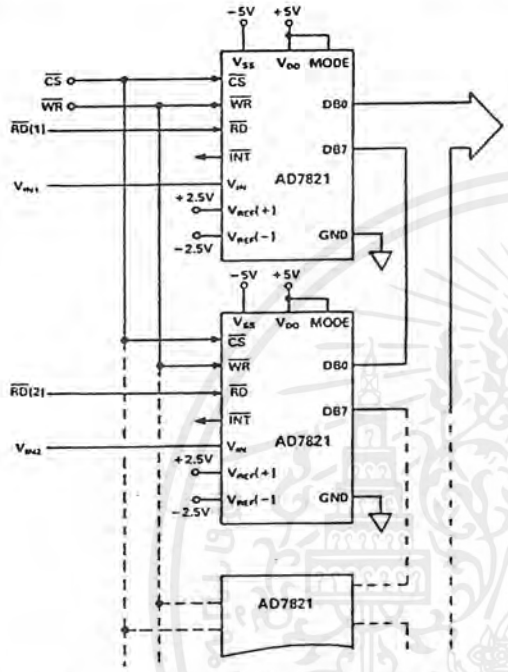


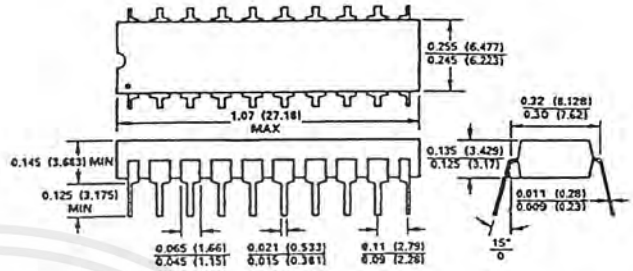
Figure 22. Simultaneous Sampling A/D Converters

The actual sampling instant, which is the instant at which V_{IN} is measured, occurs approximately 50 ns after the falling edge of $\overline{WR}$ or $\overline{RD}$ in the WR-RD or RD modes, respectively, due to internal logic delays. However, the internal logic delay and, therefore, the sampling instant can vary from device to device, but is typically within ±5 ns. This means that a maximum common input sine wave of ±2.5 V at 32 kHz, applied to any number of AD7821s in the circuit of Figure 22, will yield a maximum difference between the converter outputs of typically ±1/4 LSB.

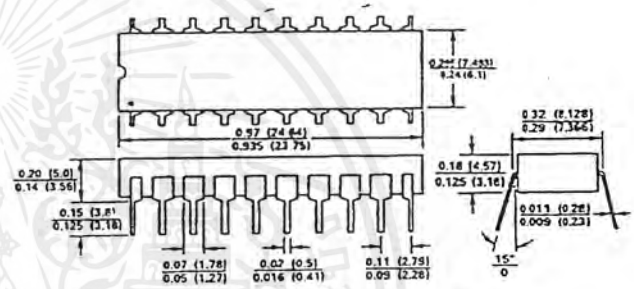
OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).

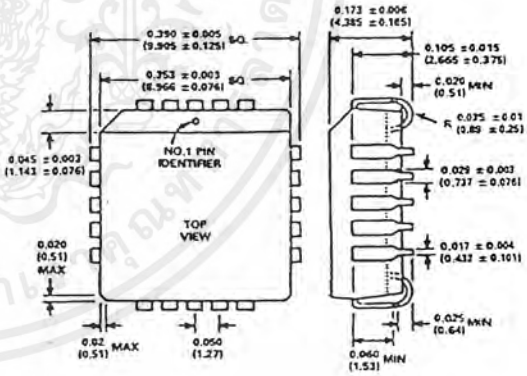
20-Pin Plastic DIP (N-20)



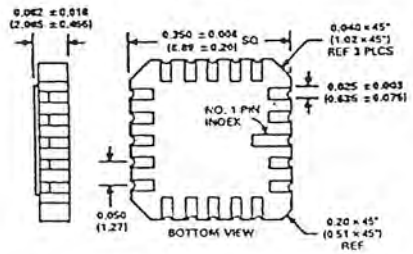
20-Pin Cerdip (Q-20)



20-Terminal Plastic Leaded Chip Carrier (P-20A)



20-Terminal Leadless Ceramic Chip Carrier (E-20A)



intersil

CA3338, CA3338A

CMOS Video Speed, 8-Bit, 50 MSPS, R2R D/A Converters

August 1997

Features

- CMOS/SOS Low Power
- R2R Output, Segmented for Low "Glitch"
- CMOS/TTL Compatible Inputs
- Fast Settling: (Typ) to $^1/_2$ LSB 20ns
- Feedthrough Latch for Clocked or Unclocked Use
- Accuracy (Typ) ..... ±0.5 LSB
- Data Complement Control
- High Update Rate (Typ) 50MHz
- Unipolar or Bipolar Operation

Applications

- TV/Video Display
- High Speed Oscilloscope Display
- Digital Waveform Generator
- Direct Digital Synthesis

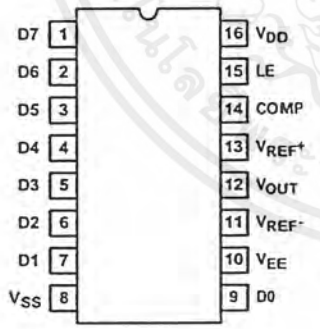
Description

The CA3338 family are CMOS/SOS high speed R2R voltage output digital-to-analog converters. They can operate from a single +5V supply, at video speeds, and can produce "rail-to-rail" output swings. Internal level shifters and a pin for an optional second supply provide for an output range below digital ground. The data complement control allows the inversion of input data while the latch enable control provides either feedthrough or latched operation. Both ends of the R2R ladder network are available externally and may be modulated for gain or offset adjustments. In addition, "glitch" energy has been kept very low by segmenting and thermometer encoding of the upper 3 bits.

The CA3338 is manufactured on a sapphire substrate to give low dynamic power dissipation, low output capacitance, and inherent latch-up resistance.

Pinout

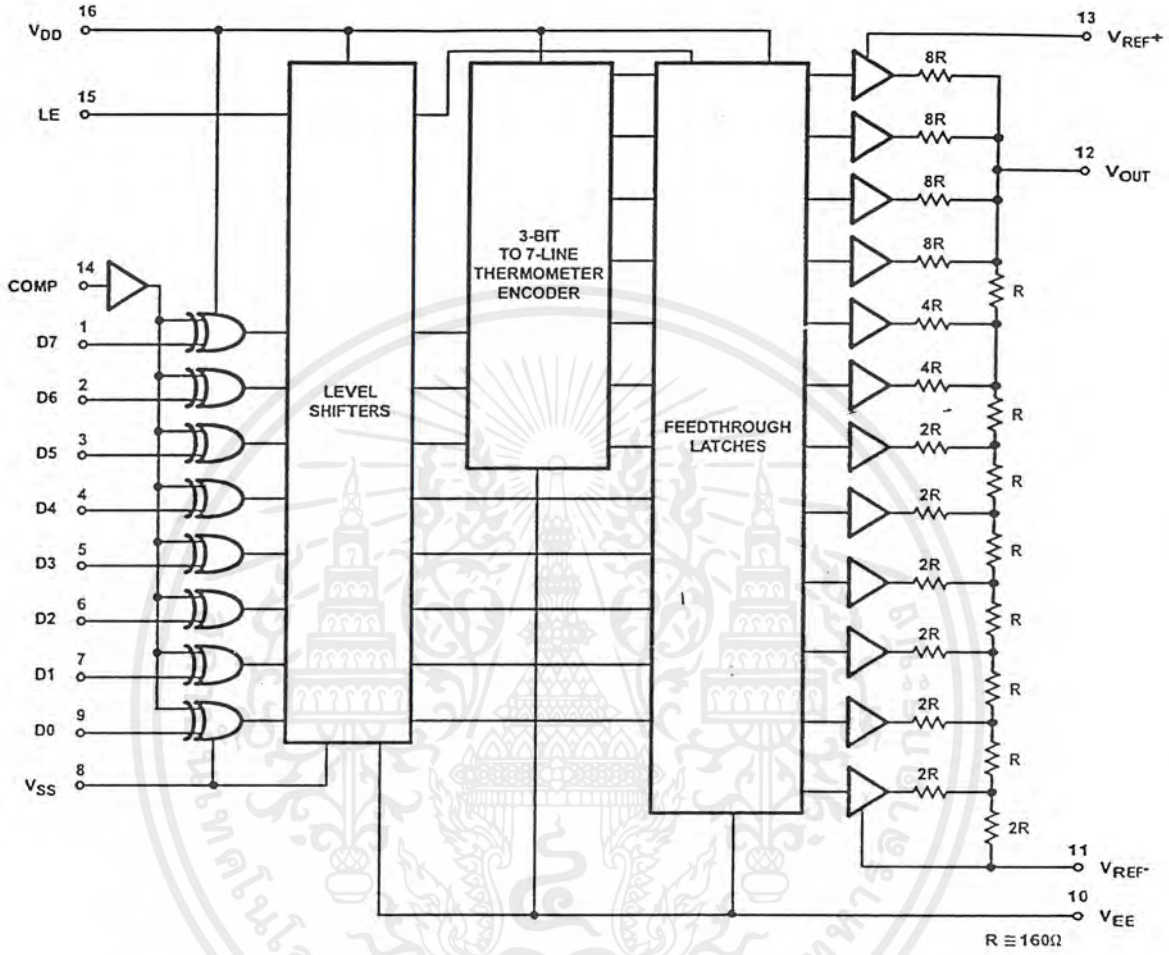
CA3338, CA3338A
(PDIP, SBDIP, SOIC)
TOP VIEW



Ordering Information

PART NUMBER	LINEARITY (INL, DNL)	TEMP. RANGE (°C)	PACKAGE	PKG. NO.
CA3338E	±1.0 LSB	-40 to 85	16 Ld PDIP	E16.3
CA3338AE	±0.75 LSB	-40 to 85	16 Ld PDIP	E16.3
CA3338D	±1.0 LSB	-55 to 125	16 Ld SBDIP	D16.3
CA3338AD	±0.75 LSB	-55 to 125	16 Ld SBDIP	D16.3
CA3338M	±1.0 LSB	-40 to 85	16 Ld SOIC	M16.3
CA3338AM	±0.75 LSB	-40 to 85	16 Ld SOIC	M16.3

Functional Diagram



Absolute Maximum Ratings

DC Supply-Voltage Range . . . . . -0.5V to +8V
(V_{DD} - V_{SS} or V_{DD} - V_{EE}, Whichever is Greater)
Input Voltage Range
Digital Inputs (LE, COMP D0 - D7) . . . . V_{SS} - 0.5V to V_{DD} + 0.5V
Analog Pins (V_{REF}+, V_{REF}-, V_{OUT}) . . . . V_{DD} - 8V to V_{DD} + 0.5V
DC Input Current
Digital Inputs (LE, COMP, D0 - D7) . . . . . ±20mA
Recommended Supply Voltage Range. . . . . .4.5V to 7.5V

Operating Conditions

Temperature Range (T_A)
Ceramic Package, D suffix . . . . . -55 °C to 125°C
Plastic Package, E suffix, M suffix . . . . . -40 °C to 85°C

Thermal Information

Thermal Resistance (Typical, Note 1)	θ_{JA} (°C/W)	θ_{JC} (°C/W)
SBDIP Package	75	24
PDIP Package	100	N/A
SOIC Package	100	N/A

Maximum Junction Temperature
Ceramic Package . . . . . 175 °C
Plastic Packages . . . . . 150 °C
Maximum Storage Temperature Range, T_{STG} . . . . -65°C to 150°C
Maximum Lead Temperature (Soldering 10s) . . . . . 300 °C
(SOIC - Lead Tips Only)

CAUTION: Stresses above those listed in "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress only rating and operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.

NOTE:

1. θ_{JA} is measured with the component mounted on an evaluation PC board in free air.

Electrical Specifications

T_A = 25°C, V_{DD} = 5V, V_{REF}+ = 4.608V, V_{SS} = V_{EE} = V_{REF}- = GND, LE Clocked at 20MHz, $R_L \geq 1$ MΩ, Unless Otherwise Specified

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNITS
ACCURACY					
Resolution		8	-	-	Bits
Integral Linearity Error	See Figure 4				
CA3338		-	-	±1	LSB
CA3338A		-	-	±0.75	LSB
Differential Linearity Error	See Figure 4				
CA3338		-	-	±0.75	LSB
CA3338A		-	-	±0.5	LSB
Gain Error	Input Code = FF_{HEX}, See Figure 3				
CA3338		-	-	±0.75	LSB
CA3338A		-	-	±0.5	LSB
Offset Error	Input Code = 00_{HEX}; See Figure 3	-	-	±0.25	LSB
DIGITAL INPUT TIMING					
Update Rate	To Maintain $^1/_2$ LSB Settling	DC	50	-	MHz
Update Rate	V_{REF}- = V_{EE} = -2.5V, V_{REF}+ = +2.5V	DC	20	-	MHz
Set Up Time t_{SU1}	For Low Glitch	-	-2	-	ns
Set Up Time t_{SU2}	For Data Store	-	8	-	ns
Hold Time t_H	For Data Store	-	5	-	ns
Latch Pulse Width t_W	For Data Store	-	5	-	ns
Latch Pulse Width t_W	V_{REF}- = V_{EE} = -2.5V, V_{REF}+ = +2.5V	-	25	-	ns
OUTPUT PARAMETERS	R_L Adjusted for $1V_{P-P}$ Output				
Output Delay t_{D1}	From LE Edge	-	25	-	ns
Output Delay t_{D2}	From Data Changing	-	22	-	ns
Rise Time t_r	10% to 90% of Output	-	4	-	ns
Settling Time t_S	10% to Settling to $^1/_2$ LSB	-	20	-	ns
Output Impedance	V_{REF}+ = 6V, V_{DD} = 6V	120	160	200	Ω
Glitch Area		-	150	-	pV/s
Glitch Area	V_{REF}- = V_{EE} = -2.5V, V_{REF}+ = +2.5V	-	250	-	pV/s

Electrical Specifications $T_A = 25^oC$, $V_{DD} = 5V$, $V_{REF}+ = 4.608V$, $V_{SS} = V_{EE} = V_{REF}- = GND$, LE Clocked at 20MHz, $R_L \geq 1 M\Omega$, Unless Otherwise Specified **(Continued)**

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNITS
REFERENCE VOLTAGE					
$V_{REF}+$ Range	(+) Full Scale, Note 1	$V_{REF}- + 3$	-	V_{DD}	V
$V_{REF}-$ Range	(-) Full Scale, Note 1	V_{EE}	-	$V_{REF}+ - 3$	V
$V_{REF}+$ Input Current	$V_{REF}+ = 6V$, $V_{DD} = 6V$	-	40	50	mA
SUPPLY VOLTAGE					
Static I_{DD} or I_{EE}	LE = Low, D0 - D7 = High	-	100	220	µA
	LE = Low, D0 - D7 = Low	-	-	100	µA
Dynamic I_{DD} or I_{EE}	V_{OUT} = 10MHz, 0V to 5V Square Wave	-	20	-	mA
Dynamic I_{DD} or I_{EE}	V_{OUT} = 10MHz, ±2.5V Square Wave	-	25	-	mA
V_{DD} Rejection	50kHz Sine Wave Applied	-	3	-	mV/V
V_{EE} Rejection	50kHz Sine Wave Applied	-	1	-	mV/V
DIGITAL INPUTS D0 - D7, LE, COMP					
High Level Input Voltage	Note 1	2	-	-	V
Low Level Input Voltage	Note 1	-	-	0.8	V
Leakage Current		-	±1	±5	µA
Capacitance		-	5	-	pF
TEMPERATURE COEFFICIENTS					
Output Impedance		-	200	-	ppm/°C

NOTE:

1. Parameter not tested, but guaranteed by design or characterization.

Pin Descriptions

PIN	NAME	DESCRIPTION
1	D7	Most Significant Bit
2	D6	Input
3	D5	Data
4	D4	Bits
5	D3	(High = True)
6	D2	
7	D1	
8	V_{SS}	Digital Ground
9	D_0	Least Significant Bit. Input Data Bit
10	V_{EE}	Analog Ground
11	$V_{REF}-$	Reference Voltage Negative Input
12	V_{OUT}	Analog Output
13	$V_{REF}+$	Reference Voltage Positive Input
14	COMP	Data Complement Control input. Active High
15	LE	Latch Enable Input. Active Low
16	V_{DD}	Digital Power Supply, +5V

Digital Signal Path

The digital inputs (LE, COMP, and D0 - D7) are of TTL compatible HCT High Speed CMOS design: the loading is essentially capacitive and the logic threshold is typically 1.5V.

The 8 data bits, D0 (weighted 2^0) through D7 (weighted 2^7), are applied to Exclusive OR gates (see Functional Diagram). The COMP (data complement) control provides the second input to the gates: if COMP is high, the data bits will be inverted as they pass through.

The input data and the LE (latch enable) signals are next applied to a level shifter. The inputs, operating between the levels of V_{DD} and V_{SS}, are shifted to operate between V_{DD} and V_{EE}. V_{EE} optionally at ground or at a negative voltage, will be discussed under bipolar operation. All further logic elements except the output drivers operate from the V_{DD} and V_{EE} supplies.

The upper 3 bits of data, D5 through D7, are input to a 3-to-7 line bar graph encoder. The encoder outputs and D0 through D4 are applied to a feedthrough latch, which is controlled by LE (latch enable).

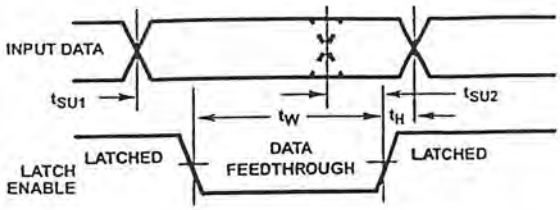


FIGURE 1. DATA TO LATCH ENABLE TIMING

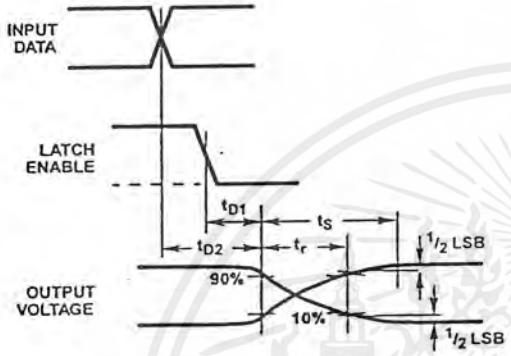


FIGURE 2. DATA AND LATCH ENABLE TO OUTPUT TIMING

Latch Operation

Data is fed from input to output while LE is low: LE should be tied low for non-clocked operation.

Non-clocked operation or changing data while LE is low is not recommended for applications requiring low output "glitch" energy: there is no guarantee of the simultaneous changing of input data or the equal propagation delay of all bits through the converter. Several parameters are given if the converter is to be used in either of these modes: t_{D2} gives the delay from the input changing to the output changing (10%), while t_{SU2} and t_H give the set up and hold times (referred to LE rising edge) needed to latch data. See Figures 1 and 2.

Clocked operation is needed for low "glitch" energy use. Data must meet the given t_{SU1} set up time to the LE falling edge, and the t_H hold time from the LE rising edge. The delay to the output changing, t_{D1}, is now referred to the LE falling edge.

There is no need for a square wave LE clock; LE must only meet the minimum t_W pulse width for successful latch operation. Generally, output timing (desired accuracy of settling) sets the upper limit of usable clock frequency.

Output Structure

The latches feed data to a row of high current CMOS drivers, which in turn feed a modified R2R ladder network.

The "N" channel (pull down) transistor of each driver plus the bottom "2R" resistor are returned to V_{REF}- this is the (-) full-scale reference. The "P" channel (pull up) transistor of each driver is returned to V_{REF}+, the (+) full-scale reference.

In unipolar operation, V_{REF}- would typically be returned to analog ground, but may be raised above ground (see specifications). There is substantial code dependent current that flows from V_{REF}+ to V_{REF}- (see V_{REF}+ input current in specifications), so V_{REF}- should have a low impedance path to ground.

In bipolar operation, V_{REF}- would be returned to a negative voltage (the maximum voltage rating to V_{DD} must be observed). V_{EE}, which supplies the gate potential for the output drivers, must be returned to a point at least as negative as V_{REF}-. Note that the maximum clocking speed decreases when the bipolar mode is used.

Static Characteristics

The ideal 8-bit D/A would have an output equal to V_{REF}- with an input code of 00_{HEX} (zero scale output), and an output equal to 255/256 of V_{REF}+ (referred to V_{REF}-) with an input code of FF_{HEX} (full scale output). The difference between the ideal and actual values of these two parameters are the OFFSET and GAIN errors, respectively; see Figure 3.

If the code into an 8-bit D/A is changed by 1 count, the output should change by 1/255 (full scale output - zero scale output). A deviation from this step size is a differential linearity error, see Figure 4. Note that the error is expressed in fractions of the ideal step size (usually called an LSB). Also note that if the (-) differential linearity error is less (in absolute numbers) than 1 LSB, the device is monotonic. (The output will always increase for increasing code or decrease for decreasing code).

If the code into an 8-bit D/A is at any value, say "N", the output voltage should be N/255 of the full scale output (referred to the zero scale output). Any deviation from that output is an integral linearity error, usually expressed in LSBs. See Figure 4.

Note that OFFSET and GAIN errors do not affect integral linearity, as the linearity is referenced to actual zero and full scale outputs, not ideal. Absolute accuracy would have to also take these errors into account.

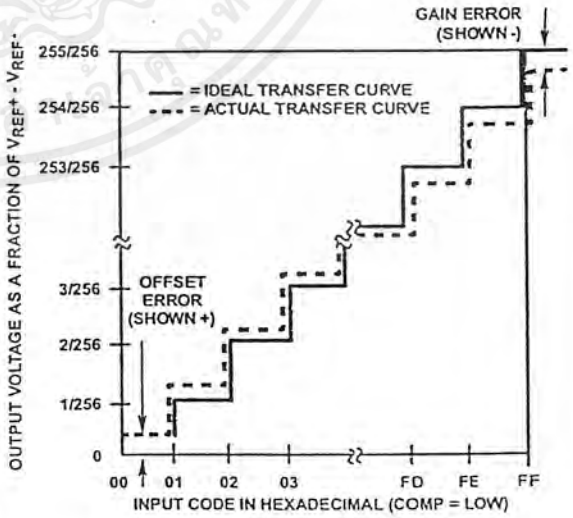


FIGURE 3. D/A OFFSET AND GAIN ERROR

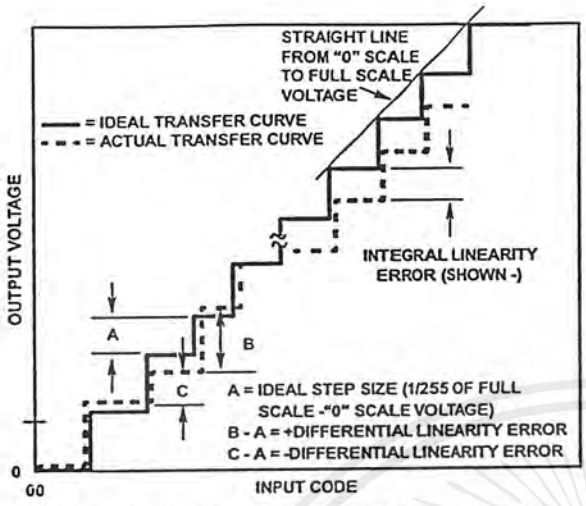


FIGURE 4. D/A INTEGRAL AND DIFFERENTIAL LINEARITY ERROR

Dynamic Characteristics

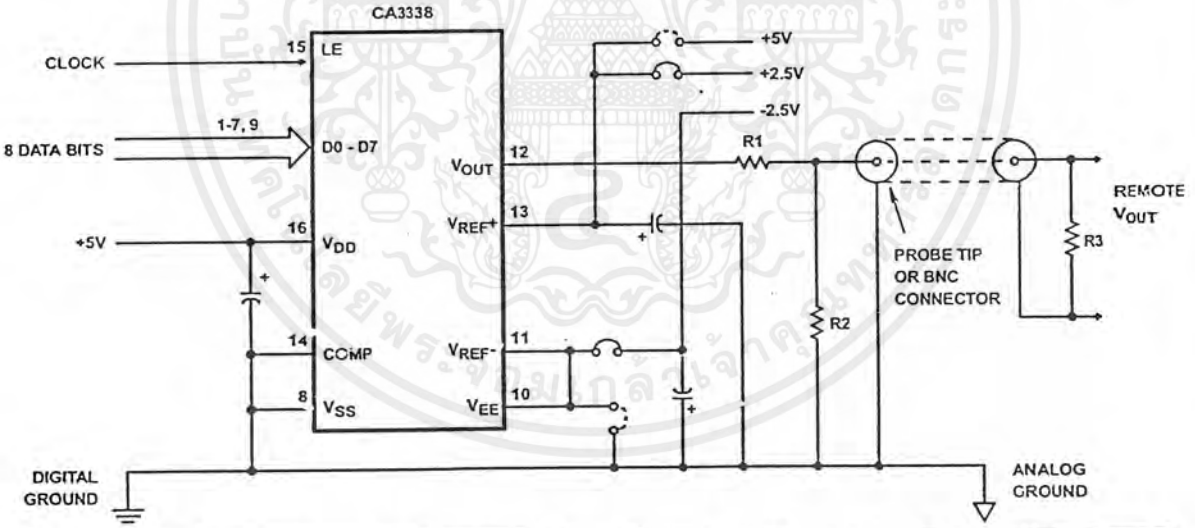
Keeping the full-scale range ($V_{REF}+ - V_{REF}-$) as high as possible gives the best linearity and lowest "glitch" energy (referred to 1V). This provides the best "P" and "N" channel gate drives (hence saturation resistance) and propagation delays. The $V_{REF}+$ (and $V_{REF}-$ if bipolar) terminal should be well bypassed as near the chip as possible.

"Glitch" energy is defined as a spurious voltage that occurs as the output is changed from one voltage to another. In a binary input converter, it is usually highest at the most significant bit transition ($7F_{HEX}$ to 80_{HEX} for an 8 bit device), and can be measured by displaying the output as the input code alternates around that point. The "glitch" energy is the area between the actual output display and an ideal one LSB step voltage (subtracting negative area from positive), at either the positive or negative-going step. It is usually expressed in pV/s.

The CA3338 uses a modified R2R ladder, where the 3 most significant bits drive a bar graph decoder and 7 equally weighted resistors. This makes the "glitch" energy at each $^1/_8$ scale transition ($1F_{HEX}$ to 20_{HEX}, $3F_{HEX}$ to 40_{HEX}, etc.) essentially equal, and far less than the MSB transition would otherwise display.

For the purpose of comparison to other converters, the output should be resistively divided to 1V full scale. Figure 5 shows a typical hook-up for checking "glitch" energy or settling time.

The settling time of the A/D is mainly a function of the output resistance (approximately 160Ω in parallel with the load resistance) and the load plus internal chip capacitance. Both "glitch" energy and settling time measurements require very good circuit and probe grounding: a probe tip connector such as Tektronix part number 131-0258-00 is recommended.

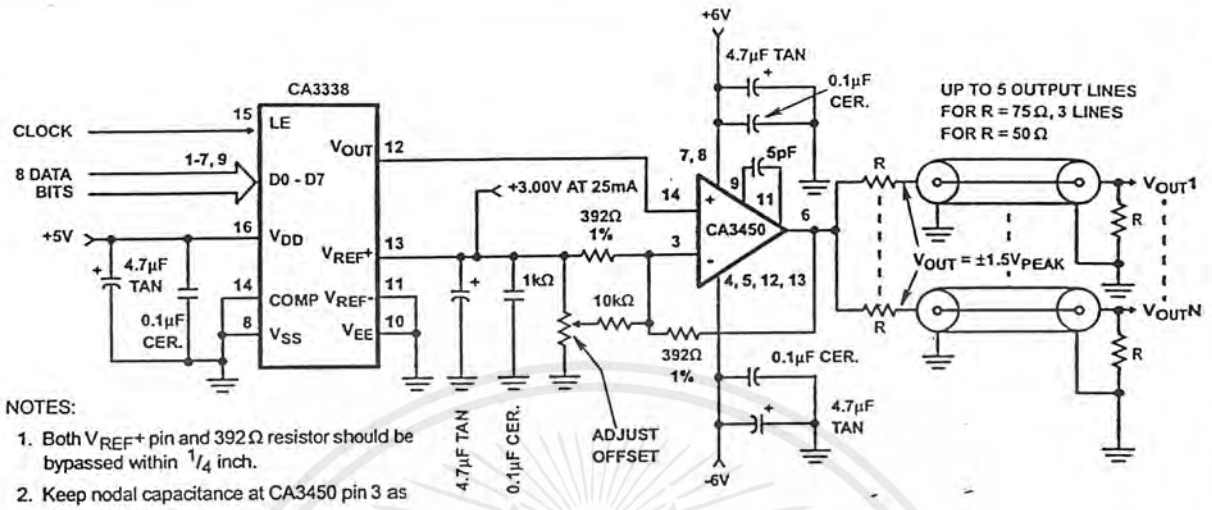


FUNCTION	CONNECTOR	R1	R2	R3	$V_{OUT\ (P-P)}$
Oscilloscope Display	Probe Tip	82 Ω	62Ω	N/C	1V
Match 93Ω Cable	BNC	75	160	93	1V
Match 75Ω Cable	BNC	18	130	75	1V
Match 50Ω Cable	BNC	Short	75	50	0 79V

NOTES:

2. $V_{OUT(P-P)}$ is approximate, and will vary as R_{OUT} of D/A varies.
3. All drawn capacitors are 0.1 µF multilayer ceramic/4.7 µF tantalum.
4. Dashed connections are for unipolar operation. Solid connection are for bipolar operation.

FIGURE 5. CA3338 DYNAMIC TEST CIRCUIT



NOTES:

1. Both V_{REF}+ pin and 392Ω resistor should be bypassed within $^1/_4$ inch.
2. Keep nodal capacitance at CA3450 pin 3 as low as possible.
3. V_{OUT} Range = ±3V at CA3450.

FIGURE 6. CA3338 AND CA3450 FOR DRIVING MULTIPLE COAXIAL LINES

TABLE 1. OUTPUT VOLTAGE vs INPUT CODE AND V REF

V_{REF}+ V_{REF}- STEP SIZE	5.12V 0 0.0200V	5.00V 0 0.0195V	4.608V 0 0.0180V	2.56V -2.56V 0.0200V	2.50V -2.50V 0.0195V
Input Code					
11111112 = FF HEX	5.1000V	4.9805V	4.5900V	2.5400V	2.4805V
11111102 = FE-HEX	5.0800	4.9610	4.5720	2.5200	2.4610
•					
•					
•					
$10000001_2 = 81_{HEX}$	2.5800	2.5195	2.3220	0.0200	0.0195
$10000000_2 = 80_{HEX}$	2.5600	2.5000	2.3040	0.0000	0.0000
$01111111_2 = 7F_{HEX}$	2.5400	2.4805	2.2860	- 0.0200	-0.0195
•					
•					
•					
$00000001_2 = 01_{HEX}$	0.0200	0.0195	0.0180	-2.5400	-2.4805
$00000000_2 = 00_{HEX}$	0.0000	0.0000	0.0000	-2.5600	-2.5000

Applications

The output of the CA3338 can be resistively divided to match a doubly terminated 50Ω or 75Ω line, although peak-to-peak swings of less than 1V may result. The output magnitude will also vary with the converter's output impedance. Figure 5 shows such an application. Note that because of the HCT input structure, the CA3338 could be operated up to +7.5V V_{DD} and V_{REF}+ supplies and still accept 0V to 5V CMOS input voltages.

If larger voltage swings or better accuracy is desired, a high speed output buffer, such as the HA-5033, HA-2542, or CA3450, can be employed. Figure 6 shows a typical application, with the output capable of driving ±2V into multiple 50Ω terminated lines.

Operating and Handling Considerations

HANDLING

All inputs and outputs of CMOS devices have a network for electrostatic protection during handling. Recommended handling practices for CMOS devices are described in AN6525. "Guide to Better Handling and Operation of CMOS Integrated Circuits."

OPERATING

Operating Voltage

During operation near the maximum supply voltage limit, care should be taken to avoid or suppress power supply turn-on and turn-off transients, power supply ripple, or ground noise; any of these conditions must not cause the absolute maximum ratings to be exceeded.

Input Signals

To prevent damage to the input protection circuit, input signals should never be greater than V_{DD} nor less than V_{SS}. Input currents must not exceed 20mA even when the power supply is off.

Unused Inputs

A connection must be provided at every input terminal. All unused input terminals must be connected to either V_{CC} or GND, whichever is appropriate.

XILINX®

XC4000E and XC4000X Series Field Programmable Gate Arrays

May 14, 1999 (Version 1.6)

Product Specification

XC4000E and XC4000X Series Features

Note: Information in this data sheet covers the XC4000E, XC4000EX, and XC4000XL families. A separate data sheet covers the XC4000XLA and XC4000XV families. Electrical Specifications and package/pin information are covered in separate sections for each family to make the information easier to access, review, and print. For access to these sections, see the Xilinx WEBLINX web site at

http://www.xilinx.com/partinfo/databook.htm#xc4000.

- System featured Field-Programmable Gate Arrays
  - Select-RAM™ memory: on-chip ultra-fast RAM with
    - synchronous write option
    - dual-port RAM option
  - Fully PCI compliant (speed grades -2 and faster)
  - Abundant flip-flops
  - Flexible function generators
  - Dedicated high-speed carry logic
  - Wide edge decoders on each edge
  - Hierarchy of interconnect lines
  - Internal 3-state bus capability
  - Eight global low-skew clock or signal distribution networks
- System Performance beyond 80 MHz
- Flexible Array Architecture
- Low Power Segmented Routing Architecture
- Systems-Oriented Features
  - IEEE 1149.1-compatible boundary scan logic support
  - Individually programmable output slew rate
  - Programmable input pull-up or pull-down resistors
  - 12 mA sink current per XC4000E output
- Configured by Loading Binary File
  - Unlimited re-programmability
- Read Back Capability
  - Program verification
  - Internal node observability
- Backward Compatible with XC4000 Devices
- Development System runs on most common computer platforms
  - Interfaces to popular design environments
  - Fully automatic mapping, placement and routing
  - Interactive design editor for design optimization

Low-Voltage Versions Available

- Low-Voltage Devices Function at 3.0 - 3.6 Volts
- XC4000XL: High Performance Low-Voltage Versions of XC4000EX devices

Additional XC4000X Series Features

- Highest Performance — 3.3 V XC4000XL
- Highest Capacity — Over 180,000 Usable Gates
- 5 V tolerant I/Os on XC4000XL
- 0.35 µm SRAM process for XC4000XL
- Additional Routing Over XC4000E
  - almost twice the routing capacity for high-density designs
- Buffered Interconnect for Maximum Speed Blocks
- Improved VersaRing™ I/O Interconnect for Better Fixed Pinout Flexibility
- 12 mA Sink Current Per XC4000X Output
- Flexible New High-Speed Clock Network
  - Eight additional Early Buffers for shorter clock delays
  - Virtually unlimited number of clock signals
- Optional Multiplexer or 2-input Function Generator on Device Outputs
- Four Additional Address Bits in Master Parallel Configuration Mode
- XC4000XV Family offers the highest density with 0.25 µm 2.5 V technology

Introduction

XC4000 Series high-performance, high-capacity Field Programmable Gate Arrays (FPGAs) provide the benefits of custom CMOS VLSI, while avoiding the initial cost, long development cycle, and inherent risk of a conventional masked gate array.

The result of thirteen years of FPGA design experience and feedback from thousands of customers, these FPGAs combine architectural versatility, on-chip Select-RAM memory with edge-triggered and dual-port modes, increased speed, abundant routing resources, and new, sophisticated software to achieve fully automated implementation of complex, high-density, high-performance designs.

The XC4000E and XC4000X Series currently have 20 members, as shown in Table 1.

Table 1: XC4000E and XC4000X Series Field Programmable Gate Arrays

Device	Logic Cells	Max Logic Gates (No RAM)	Max. RAM Bits (No Logic)	Typical Gate Range (Logic and RAM)*	CLB Matrix	Total CLBs	Number of Flip-Flops	Max. User I/O
XC4002XL	152	1,600	2,048	1,000 - 3,000	8 x 8	64	256	64
XC4003E	238	3,000	3,200	2,000 - 5,000	10 x 10	100	360	80
XC4005E/XL	466	5,000	6,272	3,000 - 9,000	14 x 14	196	616	112
XC4006E	608	6,000	8,192	4,000 - 12,000	16 x 16	256	768	128
XC4008E	770	8,000	10,368	6,000 - 15,000	18 x 18	324	936	144
XC4010E/XL	950	10,000	12,800	7,000 - 20,000	20 x 20	400	1,120	160
XC4013E/XL	1368	13,000	18,432	10,000 - 30,000	24 x 24	576	1,536	192
XC4020E/XL	1862	20,000	25,088	13,000 - 40,000	28 x 28	784	2,016	224
XC4025E	2432	25,000	32,768	15,000 - 45,000	32 x 32	1,024	2,560	256
XC4028EX/XL	2432	28,000	32,768	18,000 - 50,000	32 x 32	1,024	2,560	256
XC4036EX/XL	3078	36,000	41,472	22,000 - 65,000	36 x 36	1,296	3,168	288
XC4044XL	3800	44,000	51,200	27,000 - 80,000	40 x 40	1,600	3,840	320
XC4052XL	4598	52,000	61,952	33,000 - 100,000	44 x 44	1,936	4,576	352
XC4062XL	5472	62,000	73,728	40,000 - 130,000	48 x 48	2,304	5,376	384
XC4085XL	7448	85,000	100,352	55,000 - 180,000	56 x 56	3,136	7,168	448

* Max values of Typical Gate Range include 20-30% of CLBs used as RAM.

Note: *All functionality in low-voltage families is the same as in the corresponding 5-Volt family, except where numerical references are made to timing or power.*

Description

XC4000 Series devices are implemented with a regular, flexible, programmable architecture of Configurable Logic Blocks (CLBs), interconnected by a powerful hierarchy of versatile routing resources, and surrounded by a perimeter of programmable Input/Output Blocks (IOBs). They have generous routing resources to accommodate the most complex interconnect patterns.

The devices are customized by loading configuration data into internal memory cells. The FPGA can either actively read its configuration data from an external serial or byte-parallel PROM (master modes), or the configuration data can be written into the FPGA from an external device (slave and peripheral modes).

XC4000 Series FPGAs are supported by powerful and sophisticated software, covering every aspect of design from schematic or behavioral entry, floor planning, simulation, automatic block placement and routing of interconnects, to the creation, downloading, and readback of the configuration bit stream.

Because Xilinx FPGAs can be reprogrammed an unlimited number of times, they can be used in innovative designs where hardware is changed dynamically, or where hardware must be adapted to different user applications. FPGAs are ideal for shortening design and development cycles, and also offer a cost-effective solution for production rates well beyond 5,000 systems per month. For lowest high-volume unit cost, a design can first be implemented in the XC4000E or XC4000X, then migrated to one of Xilinx' compatible HardWire mask-programmed devices.

Taking Advantage of Re-configuration

FPGA devices can be re-configured to change logic function while resident in the system. This capability gives the system designer a new degree of freedom not available with any other type of logic.

Hardware can be changed as easily as software. Design updates or modifications are easy, and can be made to products already in the field. An FPGA can even be re-configured dynamically to perform different functions at different times.

Re-configurable logic can be used to implement system self-diagnostics, create systems capable of being re-configured for different environments or operations, or implement multi-purpose hardware for a given application. As an added benefit, using re-configurable FPGA devices simplifies hardware design and debugging and shortens product time-to-market.

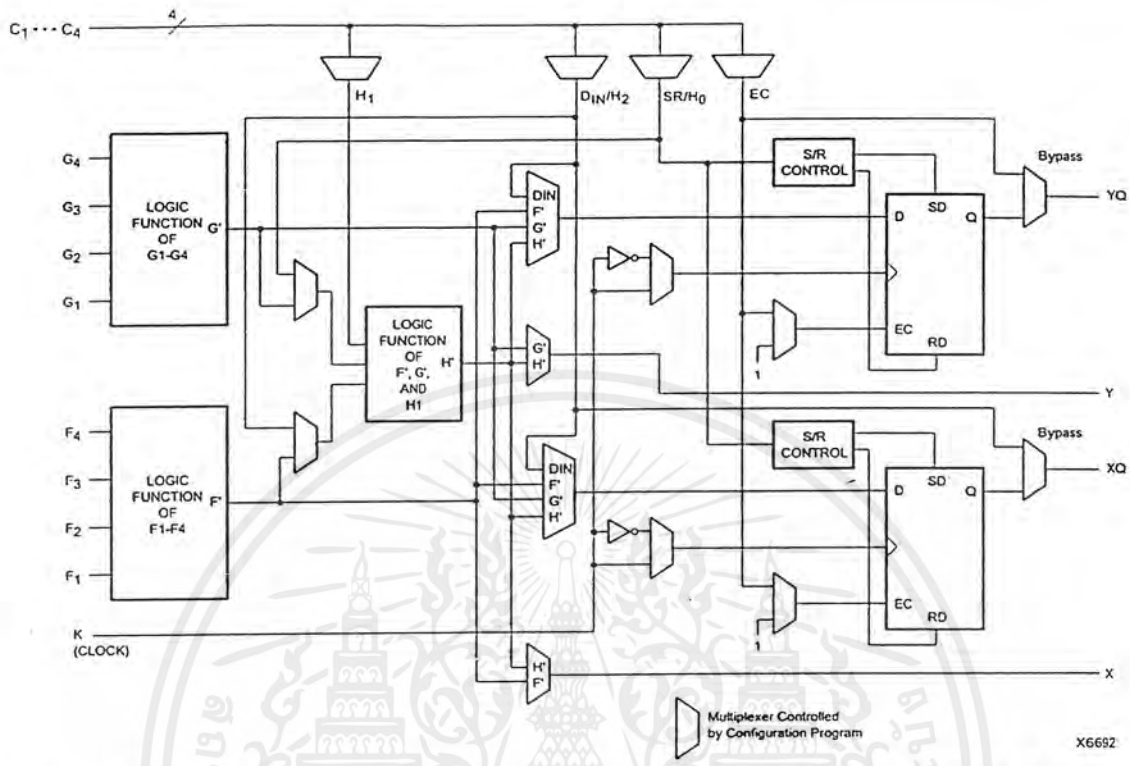


Figure 1: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

Flip-Flops

The CLB can pass the combinatorial output(s) to the interconnect network, but can also store the combinatorial results or other incoming data in one or two flip-flops, and connect their outputs to the interconnect network as well.

The two edge-triggered D-type flip-flops have common clock (K) and clock enable (EC) inputs. Either or both clock inputs can also be permanently enabled. Storage element functionality is described in Table 2.

Latches (XC4000X only)

The CLB storage elements can also be configured as latches. The two latches have common clock (K) and clock enable (EC) inputs. Storage element functionality is described in Table 2.

Clock Input

Each flip-flop can be triggered on either the rising or falling clock edge. The clock pin is shared by both storage elements. However, the clock is individually invertible for each storage element. Any inverter placed on the clock input is automatically absorbed into the CLB.

Clock Enable

The clock enable signal (EC) is active High. The EC pin is shared by both storage elements. If left unconnected for either, the clock enable for that storage element defaults to the active state. EC is not invertible within the CLB.

Table 2: CLB Storage Element Functionality (active rising edge is shown)

Mode	K	EC	SR	D	Q
Power-Up or GSR	X	X	X	X	SR
Flip-Flop	X	X	1	X	SR
	⎍/⎺	1*	0*	D	D
	0	X	0*	X	Q
Latch	1	1*	0*	X	Q
	0	1*	0*	D	D
Both	X	0	0*	X	Q

Legend:

X	Don't care
⎍/⎺	Rising edge
SR	Set or Reset value. Reset is default.
0*	Input is Low or unconnected (default value)
1*	Input is High or unconnected (default value)

Table 16: Pin Descriptions

Pin Name	I/O During Config.	I/O After Config.	Pin Description
Permanently Dedicated Pins			
VCC	I	I	Eight or more (depending on package) connections to the nominal +5 V supply voltage (+3.3 V for low-voltage devices). All must be connected, and each must be decoupled with a 0.01 - 0.1 µF capacitor to Ground.
GND	I	I	Eight or more (depending on package type) connections to Ground. All must be connected.
CCLK	I or O	I	During configuration, Configuration Clock (CCLK) is an output in Master modes or Asynchronous Peripheral mode, but is an input in Slave mode and Synchronous Peripheral mode. After configuration, CCLK has a weak pull-up resistor and can be selected as the Readback Clock. There is no CCLK High or Low time restriction on XC4000 Series devices, except during Readback. See "Violating the Maximum High and Low Time Specification for the Readback Clock" on page 56 for an explanation of this exception.
DONE	I/O	O	DONE is a bidirectional signal with an optional internal pull-up resistor. As an output, it indicates the completion of the configuration process. As an input, a Low level on DONE can be configured to delay the global logic initialization and the enabling of outputs. The optional pull-up resistor is selected as an option in the XACT*step* program that creates the configuration bitstream. The resistor is included by default.
$\overline{\text{PROGRAM}}$	I	I	$\overline{\text{PROGRAM}}$ is an active Low input that forces the FPGA to clear its configuration memory. It is used to initiate a configuration cycle. When $\overline{\text{PROGRAM}}$ goes High, the FPGA finishes the current clear cycle and executes another complete clear cycle, before it goes into a WAIT state and releases $\overline{\text{INIT}}$. The $\overline{\text{PROGRAM}}$ pin has a permanent weak pull-up, so it need not be externally pulled up to Vcc.
User I/O Pins That Can Have Special Functions			
RDY/$\overline{\text{BUSY}}$	O	I/O	During Peripheral mode configuration, this pin indicates when it is appropriate to write another byte of data into the FPGA. The same status is also available on D7 in Asynchronous Peripheral mode, if a read operation is performed when the device is selected. After configuration, RDY/$\overline{\text{BUSY}}$ is a user-programmable I/O pin. RDY/$\overline{\text{BUSY}}$ is pulled High with a high-impedance pull-up prior to $\overline{\text{INIT}}$ going High.
$\overline{\text{RCLK}}$	O	I/O	During Master Parallel configuration, each change on the A0-A17 outputs (A0 - A21 for XC4000X) is preceded by a rising edge on $\overline{\text{RCLK}}$, a redundant output signal. $\overline{\text{RCLK}}$ is useful for clocked PROMs. It is rarely used during configuration. After configuration, $\overline{\text{RCLK}}$ is a user-programmable I/O pin.
M0, M1, M2	I	I (M0), O (M1), I (M2)	As Mode inputs, these pins are sampled after $\overline{\text{INIT}}$ goes High to determine the configuration mode to be used. After configuration, M0 and M2 can be used as inputs, and M1 can be used as a 3-state output. These three pins have no associated input or output registers. During configuration, these pins have weak pull-up resistors. For the most popular configuration mode, Slave Serial, the mode pins can thus be left unconnected. The three mode inputs can be individually configured with or without weak pull-up or pull-down resistors. A pull-down resistor value of 4.7 kΩ is recommended. These pins can only be used as inputs or outputs when called out by special schematic definitions. To use these pins, place the library components MD0, MD1, and MD2 instead of the usual pad symbols. Input or output buffers must still be used.
TDO	O	O	If boundary scan is used, this pin is the Test Data Output. If boundary scan is not used, this pin is a 3-state output without a register, after configuration is completed. This pin can be user output only when called out by special schematic definitions. To use this pin, place the library component TDO instead of the usual pad symbol. An output buffer must still be used.

Table 16: Pin Descriptions (Continued)

Pin Name	I/O During Config.	I/O After Config.	Pin Description
TDI, TCK, TMS	I	I/O or I (JTAG)	If boundary scan is used, these pins are Test Data In, Test Clock, and Test Mode Select inputs respectively. They come directly from the pads, bypassing the IOBs. These pins can also be used as inputs to the CLB logic after configuration is completed. If the BSCAN symbol is not placed in the design, all boundary scan functions are inhibited once configuration is completed, and these pins become user-programmable I/O. In this case, they must be called out by special schematic definitions. To use these pins, place the library components TDI, TCK, and TMS instead of the usual pad symbols. Input or output buffers must still be used.
HDC	O	I/O	High During Configuration (HDC) is driven High until the I/O go active. It is available as a control output indicating that configuration is not yet completed. After configuration, HDC is a user-programmable I/O pin.
$\overline{\text{LDC}}$	O	I/O	Low During Configuration ($\overline{\text{LDC}}$) is driven Low until the I/O go active. It is available as a control output indicating that configuration is not yet completed. After configuration, $\overline{\text{LDC}}$ is a user-programmable I/O pin.
$\overline{\text{INIT}}$	I/O	I/O	Before and during configuration, $\overline{\text{INIT}}$ is a bidirectional signal. A 1 kΩ - 10 kΩ external pull-up resistor is recommended. As an active-Low open-drain output, $\overline{\text{INIT}}$ is held Low during the power stabilization and internal clearing of the configuration memory. As an active-Low input, it can be used to hold the FPGA in the internal WAIT state before the start of configuration. Master mode devices stay in a WAIT state an additional 30 to 300µs after $\overline{\text{INIT}}$ has gone High. During configuration, a Low on this output indicates that a configuration data error has occurred. After the I/O go active, $\overline{\text{INIT}}$ is a user-programmable I/O pin.
PGCK1 - PGCK4 (XC4000E only)	Weak Pull-up	I or I/O	Four Primary Global inputs each drive a dedicated internal global net with short delay and minimal skew. If not used to drive a global buffer, any of these pins is a user-programmable I/O. The PGCK1-PGCK4 pins drive the four Primary Global Buffers. Any input pad symbol connected directly to the input of a BUFGP symbol is automatically placed on one of these pins.
SGCK1 - SGCK4 (XC4000E only)	Weak Pull-up	I or I/O	Four Secondary Global inputs each drive a dedicated internal global net with short delay and minimal skew. These internal global nets can also be driven from internal logic. If not used to drive a global net, any of these pins is a user-programmable I/O pin. The SGCK1-SGCK4 pins provide the shortest path to the four Secondary Global Buffers. Any input pad symbol connected directly to the input of a BUFGS symbol is automatically placed on one of these pins.
GCK1 - GCK8 (XC4000X only)	Weak Pull-up	I or I/O	Eight inputs can each drive a Global Low-Skew buffer. In addition, each can drive a Global Early buffer. Each pair of global buffers can also be driven from internal logic, but must share an input signal. If not used to drive a global buffer, any of these pins is a user-programmable I/O. Any input pad symbol connected directly to the input of a BUFGLS or BUFGE symbol is automatically placed on one of these pins.
FCLK1 - FCLK4 (XC4000XLA and XC4000XV only)	Weak Pull-up	I or I/O	Four inputs can each drive a Fast Clock (FCLK) buffer which can deliver a clock signal to any IOB clock input in the octant of the die served by the Fast Clock buffer. Two Fast Clock buffers serve the two IOB octants on the left side of the die and the other two Fast Clock buffers serve the two IOB octants on the right side of the die. On each side of the die, one Fast Clock buffer serves the upper octant and the other serves the lower octant. If not used to drive a Fast Clock buffer, any of these pins is a user-programmable I/O.

Table 16: Pin Descriptions (Continued)

Pin Name	I/O During Config.	I/O After Config.	Pin Description
$\overline{CS0}$, CS1, $\overline{WS}$, $\overline{RS}$	I	I/O	These four inputs are used in Asynchronous Peripheral mode. The chip is selected when $\overline{CS0}$ is Low and CS1 is High. While the chip is selected, a Low on Write Strobe ($\overline{WS}$) loads the data present on the D0 - D7 inputs into the internal data buffer. A Low on Read Strobe ($\overline{RS}$) changes D7 into a status output — High if Ready, Low if Busy — and drives D0 - D6 High. In Express mode, CS1 is used as a serial-enable signal for daisy-chaining. $\overline{WS}$ and $\overline{RS}$ should be mutually exclusive, but if both are Low simultaneously, the Write Strobe overrides. After configuration, these are user-programmable I/O pins.
A0 - A17	O	I/O	During Master Parallel configuration, these 18 output pins address the configuration EPROM. After configuration, they are user-programmable I/O pins.
A18 - A21 (XC4003XL to XC4085XL)	O	I/O	During Master Parallel configuration with an XC4000X master, these 4 output pins add 4 more bits to address the configuration EPROM. After configuration, they are user-programmable I/O pins. (See Master Parallel Configuration section for additional details.)
D0 - D7	I	I/O	During Master Parallel and Peripheral configuration, these eight input pins receive configuration data. After configuration, they are user-programmable I/O pins.
DIN	I	I/O	During Slave Serial or Master Serial configuration, DIN is the serial configuration data input receiving data on the rising edge of CCLK. During Parallel configuration, DIN is the D0 input. After configuration, DIN is a user-programmable I/O pin.
DOUT	O	I/O	During configuration in any mode but Express mode, DOUT is the serial configuration data output that can drive the DIN of daisy-chained slave FPGAs. DOUT data changes on the falling edge of CCLK, one-and-a-half CCLK periods after it was received at the DIN input. In Express modefor XC4000E and XC4000X only, DOUT is the status output that can drive the CS1 of daisy-chained FPGAs, to enable and disable downstream devices. After configuration, DOUT is a user-programmable I/O pin.
Unrestricted User-Programmable I/O Pins			
I/O	Weak Pull-up	I/O	These pins can be configured to be input and/or output after configuration is completed. Before configuration is completed, these pins have an internal high-value pull-up resistor (25 kΩ - 100 kΩ) that defines the logic level as High.

Boundary Scan

The 'bed of nails' has been the traditional method of testing electronic assemblies. This approach has become less appropriate, due to closer pin spacing and more sophisticated assembly methods like surface-mount technology and multi-layer boards. The IEEE Boundary Scan Standard 1149.1 was developed to facilitate board-level testing of electronic assemblies. Design and test engineers can imbed a standard test logic structure in their device to achieve high fault coverage for I/O and internal logic. This structure is easily implemented with a four-pin interface on any boundary scan-compatible IC. IEEE 1149.1-compatible devices may be serial daisy-chained together, connected in parallel, or a combination of the two.

The XC4000 Series implements IEEE 1149.1-compatible BYPASS, PRELOAD/SAMPLE and EXTEST boundary scan instructions. When the boundary scan configuration option is selected, three normal user I/O pins become dedicated inputs for these functions. Another user output pin becomes the dedicated boundary scan output. The details of how to enable this circuitry are covered later in this section.

By exercising these input signals, the user can serially load commands and data into these devices to control the driving of their outputs and to examine their inputs. This method is an improvement over bed-of-nails testing. It avoids the need to over-drive device outputs, and it reduces the user interface to four pins. An optional fifth pin, a reset for the control logic, is described in the standard but is not implemented in Xilinx devices.

The dedicated on-chip logic implementing the IEEE 1149.1 functions includes a 16-state machine, an instruction register and a number of data registers. The functional details can be found in the IEEE 1149.1 specification and are also discussed in the Xilinx application note XAPP 017: *Boundary Scan in XC4000 Devices*."

Figure 40 on page 43 shows a simplified block diagram of the XC4000E Input/Output Block with boundary scan implemented. XC4000X boundary scan logic is identical.

XC4005E/XL Device Pnout Tables

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

XC4005E/XL Pad Name	PC 84	PQ 100	VQ 100††	TQ 144	PG 156†	PQ 160	PQ 208	Bndry Scan
VCC	P2	P92	P89	P128	H3	P142	P183	-
I/O (A8)	P3	P93	P90	P129	H1	P143	P184	44
I/O (A9)	P4	P94	P91	P130	G1	P144	P185	47
I/O (A19) ††	-	P95	P92	P131	G2	P145	P186	50
I/O (A18) ††	-	P96	P93	P132	G3	P146	P187	53
I/O (A10)	P5	P97	P94	P133	F1	P147	P190	56
I/O (A11)	P6	P98	P95	P134	F2	P148	P191	59
I/O	-	-	-	P135	E1	P149	P192	62
I/O	-	-	-	P136	E2	P150	P193	65
GND	-	-	-	P137	F3	P151	P194	-
I/O (A12)	P7	P99	P96	P138	E3	P154	P199	68
I/O (A13)	P8	P100	P97	P139	C1	P155	P200	71
I/O	-	-	-	P140	C2	P156	P201	74
I/O	-	-	-	P141	D3	P157	P202	77
I/O (A14)	P9	P1	P98	P142	B1	P158	P203	80
I/O, SGCK1 †, GCK8 †† (A15)	P10	P2	P99	P143	B2	P159	P204	83
VCC	P11	P3	P100	P144	C3	P160	P205	-
GND	P12	P4	P1	P1	C4	P1	P2	-
I/O, PGCK1†, GCK1†† (A16)	P13	P5	P2	P2	B3	P2	P4	86
I/O (A17)	P14	P6	P3	P3	A1	P3	P5	89
I/O	-	-	-	P4	A2	P4	P6	92
I/O	-	-	-	P5	C5	P5	P7	95
I/O, TDI	P15	P7	P4	P6	B4	P6	P8	98
I/O, TCK	P16	P8	P5	P7	A3	P7	P9	101
GND	-	-	-	P8	C6	P10	P14	-
I/O	-	-	-	P9	B5	P11	P15	104
I/O	-	-	-	P10	B6	P12	P16	107
I/O, TMS	P17	P9	P6	P11	A5	P13	P17	110
I/O	P18	P10	P7	P12	C7	P14	P18	113
I/O	-	-	-	P13	B7	P15	P21	116
I/O	-	P11	P8	P14	A6	P16	P22	119
I/O	P19	P12	P9	P15	A7	P17	P23	122
I/O	P20	P13	P10	P16	A8	P18	P24	125
GND	P21	P14	P11	P17	C8	P19	P25	-
VCC	P22	P15	P12	P18	B8	P20	P26	-
I/O	P23	P16	P13	P19	C9	P21	P27	128
I/O	P24	P17	P14	P20	B9	P22	P28	131
I/O	-	P18	P15	P21	A9	P23	P29	134
I/O	-	-	-	P22	B10	P24	P30	137
I/O	P25	P19	P16	P23	C10	P25	P33	140
I/O	P26	P20	P17	P24	A10	P26	P34	143
I/O	-	-	-	P25	A11	P27	P35	146
I/O	-	-	-	P26	B11	P28	P36	149
GND	-	-	-	P27	C11	P29	P37	-
I/O	P27	P21	P18	P28	B12	P32	P42	152
I/O	-	P22	P19	P29	A13	P33	P43	155
I/O	-	-	-	P30	A14	P34	P44	158
I/O	-	-	-	P31	C12	P35	P45	161
I/O	P28	P23	P20	P32	B13	P36	P46	164
I/O, SGCK2 †, GCK2 ††	P29	P24	P21	P33	B14	P37	P47	167
O (M1)	P30	P25	P22	P34	A15	P38	P48	170
GND	P31	P26	P23	P35	C13	P39	P49	-
I (M0)	P32	P27	P24	P36	A16	P40	P50	173
VCC	P33	P28	P25	P37	C14	P41	P55	-
I (M2)	P34	P29	P26	P38	B15	P42	P56	174
I/O, PGCK2 †, GCK3 ††	P35	P30	P27	P39	B16	P43	P57	175
I/O (HDC)	P36	P31	P28	P40	D14	P44	P58	178
I/O	-	-	-	P41	C15	P45	P59	181
I/O	-	-	-	P42	D15	P46	P60	184
I/O	-	P32	P29	P43	E14	P47	P61	187
I/O (LDC)	P37	P33	P30	P44	C16	P48	P62	190

XC4005E/XL Pad Name	PC 84	PQ 100	VQ 100††	TQ 144	PG 156†	PQ 160	PQ 208	Bndry Scan
GND	-	-	-	P45	F14	P51	P67	-
I/O	-	-	-	P46	F15	P52	P68	193
I/O	-	-	-	P47	E16	P53	P69	196
I/O	P38	P34	P31	P48	F16	P54	P70	199
I/O	P39	P35	P32	P49	G14	P55	P71	202
I/O	-	P36	P33	P50	G15	P56	P74	205
I/O	-	P37	P34	P51	G16	P57	P75	208
I/O	P40	P38	P35	P52	H16	P58	P76	211
I/O (INIT)	P41	P39	P36	P53	H15	P59	P77	214
VCC	P42	P40	P37	P54	H14	P60	P78	-
GND	P43	P41	P38	P55	J14	P61	P79	-
I/O	P44	P42	P39	P56	J15	P62	P80	217
I/O	P45	P43	P40	P57	J16	P63	P81	220
I/O	-	P44	P41	P58	K16	P64	P82	223
I/O	-	P45	P42	P59	K15	P65	P83	226
I/O	P46	P46	P43	P60	K14	P66	P86	229
I/O	P47	P47	P44	P61	L16	P67	P87	232
I/O	-	-	-	P62	M16	P68	P88	235
I/O	-	-	-	P63	L15	P69	P89	238
GND	-	-	-	P64	L14	P70	P90	-
I/O	P48	P48	P45	P65	P16	P73	P95	241
I/O	P49	P49	P46	P66	M14	P74	P96	244
I/O	-	-	-	P67	N15	P75	P97	247
I/O	-	-	-	P68	P15	P76	P98	250
I/O	P50	P50	P47	P69	N14	P77	P99	253
I/O, SGCK3 †, GCK4 ††	P51	P51	P48	P70	R16	P78	P100	256
GND	P52	P52	P49	P71	P14	P79	P101	-
DONE	P53	P53	P50	P72	R15	P80	P103	-
VCC	P54	P54	P51	P73	P13	P81	P106	-
PROGRAM	P55	P55	P52	P74	R14	P82	P108	-
I/O (D7)	P56	P56	P53	P75	T16	P83	P109	259
I/O, PGCK3†, GCK5††	P57	P57	P54	P76	T15	P84	P110	262
I/O	-	-	-	P77	R13	P85	P111	265
I/O	-	-	-	P78	P12	P86	P112	268
I/O (D6)	P58	P58	P55	P79	T14	P87	P113	271
I/O	-	P59	P56	P80	T13	P88	P114	274
GND	-	-	-	P81	P11	P91	P119	-
I/O	-	-	-	P82	R11	P92	P120	277
I/O	-	-	-	P83	T11	P93	P121	280
I/O (D5)	P59	P60	P57	P84	T10	P94	P122	283
I/O (CS0)	P60	P61	P58	P85	P10	P95	P123	286
I/O	-	P62	P59	P86	R10	P96	P126	289
I/O	-	P63	P60	P87	T9	P97	P127	292
I/O (D4)	P61	P64	P61	P88	R9	P98	P128	295
I/O	P62	P65	P62	P89	P9	P99	P129	298
VCC	P63	P66	P63	P90	R8	P100	P130	-
GND	P64	P67	P64	P91	P8	P101	P131	-
I/O (D3)	P65	P68	P65	P92	T8	P102	P132	301
I/O (RS)	P66	P69	P66	P93	T7	P103	P133	304
I/O	-	P70	P67	P94	T6	P104	P134	307
I/O	-	-	-	P95	R7	P105	P135	310
I/O (D2)	P67	P71	P68	P96	P7	P106	P138	313
I/O	P68	P72	P69	P97	T5	P107	P139	316
I/O	-	-	-	P98	R5	P108	P140	319
I/O	-	-	-	P99	T4	P109	P141	322
GND	-	-	-	P100	P6	P110	P142	-
I/O (D1)	P69	P73	P70	P101	T3	P113	P147	325
I/O (RCLK, RDY/BUSY)	P70	P74	P71	P102	P5	P114	P148	328
I/O	-	-	-	P103	R4	P115	P149	331
I/O	-	-	-	P104	R3	P116	P150	334
I/O (D0, DIN)	P71	P75	P72	P105	P4	P117	P151	337

XC4005E/XL Pad Name	PC 84	PQ 100	VQ 100††	TQ 144	PG 156†	PQ 160	PQ 208	Bndry Scan
I/O, SGCK4 †, GCK6 †† (DOUT)	P72	P76	P73	P106	T2	P118	P152	340
CCLK	P73	P77	P74	P107	R2	P119	P153	-
VCC	P74	P78	P75	P108	P3	P120	P154	-
O, TDO	P75	P79	P76	P109	T1	P121	P159	0
GND	P76	P80	P77	P110	N3	P122	P160	-
I/O (A0, WS)	P77	P81	P78	P111	R1	P123	P161	2
I/O, PGCK4 †, GCK7 †† (A1)	P78	P82	P79	P112	P2	P124	P162	5
I/O	-	-	-	P113	N2	P125	P163	8
I/O	-	-	-	P114	M3	P126	P164	11
I/O (CS1, A2)	P79	P83	P80	P115	P1	P127	P165	14
I/O (A3)	P80	P84	P81	P116	N1	P128	P166	17
GND	-	-	-	P118	L3	P131	P171	-
I/O	-	-	-	P119	L2	P132	P172	20
I/O	-	-	-	P120	L1	P133	P173	23
I/O (A4)	P81	P85	P82	P121	K3	P134	P174	26
I/O (A5)	P82	P86	P83	P122	K2	P135	P175	29
I/O (A21) ††	-	P87	P84	P123	K1	P137	P178	32
I/O (A20) ††	-	P88	P85	P124	J1	P138	P179	35
I/O (A6)	P83	P89	P86	P125	J2	P139	P180	38
I/O (A7)	P84	P90	P87	P126	J3	P140	P181	41
GND	P1	P91	P88	P127	H2	P141	P182	-

6/10/97

† = E only
†† = XL only

Additional XC4005E/XL Package Pins

TQ144

Not Connected Pins					
P117	-	-	-	-	-

5/5/97

PG156

Not Connected Pins					
A4	A12	D1	D2	D16	E15
M1	M2	M15	N16	R5	R12
T12	-	-	-	-	-

5/5/97

PQ160

Not Connected Pins					
P8	P9	P30	P31	P49	P50
P71	P72	P89	P90	P111	P112
P129	P130	P136	P152	P153	-

6/16/97

PQ208

Not Connected Pins					
P1	P3	P10	P11	P12	P13
P19	P20	P31	P32	P38	P39
P40	P41	P51	P52	P53	P54
P63	P64	P65	P66	P72	P73
P84	P85	P91	P92	P93	P94
P102	P104	P105	P107	P115	P116
P117	P118	P124	P125	P136	P137
P143	P144	P145	P146	P155	P156
P157	P158	P167	P168	P169	P170
P176	P177	P188	P189	P195	P196
P197	P198	P206	P207	P208	-

6/5/97

XC4006E Device Pinout Tables

XC4006E Pad Name	PC 84	TQ 144	PG 156	PQ 160	PQ 208	Bndry Scan
VCC	P2	P128	H3	P142	P183	-
I/O (A8)	P3	P129	H1	P143	P184	50
I/O (A9)	P4	P130	G1	P144	P185	53
I/O	-	P131	G2	P145	P186	56
I/O	-	P132	G3	P146	P187	59
I/O (A10)	P5	P133	F1	P147	P190	62
I/O (A11)	P6	P134	F2	P148	P191	65
I/O	-	P135	E1	P149	P192	68
I/O	-	P136	E2	P150	P193	71
GND	-	P137	F3	P151	P194	-
I/O	-	-	D1	P152	P197	74
I/O	-	-	D2	P153	P198	77
I/O (A12)	P7	P138	E3	P154	P199	80
I/O (A13)	P8	P139	C1	P155	P200	83
I/O	-	P140	C2	P156	P201	86
I/O	-	P141	C3	P157	P202	89
I/O (A14)	P9	P142	B1	P158	P203	92
I/O, SGCK1 (A15)	P10	P143	B2	P159	P204	95
VCC	P11	P144	C3	P160	P205	-
GND	P12	P1	C4	P1	P2	-
I/O, PGCK1 (A16)	P13	P2	B3	P2	P4	98
I/O (A17)	P14	P3	A1	P3	P5	101
I/O	-	P4	A2	P4	P6	104
I/O	-	P5	C5	P5	P7	107
I/O, TDI	P15	P6	B4	P6	P8	110
I/O, TCK	P16	P7	A3	P7	P9	113
I/O	-	-	A4	P8	P10	116
I/O	-	-	-	P9	P11	119
GND	-	P8	C6	P10	P14	-

XC4006E Pad Name	PC 84	TQ 144	PG 156	PQ 160	PQ 208	Bndry Scan
I/O	-	P9	B5	P11	P15	122
I/O	-	P10	B6	P12	P16	125
I/O, TMS	P17	P11	A5	P13	P17	128
I/O	P18	P12	C7	P14	P18	131
I/O	-	P13	B7	P15	P21	134
I/O	-	P14	A6	P16	P22	137
I/O	P19	P15	A7	P17	P23	140
I/O	P20	P16	A8	P18	P24	143
GND	P21	P17	C8	P19	P25	-
VCC	P22	P18	B8	P20	P26	-
I/O	P23	P19	C9	P21	P27	146
I/O	P24	P20	B9	P22	P28	149
I/O	-	P21	A9	P23	P29	152
I/O	-	P22	B10	P24	P30	155
I/O	P25	P23	C10	P25	P33	158
I/O	P26	P24	A10	P26	P34	161
I/O	-	P25	A11	P27	P35	164
I/O	-	P26	B11	P28	P36	167
GND	-	P27	C11	P29	P37	-
I/O	-	-	A12	P30	P40	170
I/O	-	-	-	P31	P41	173
I/O	P27	P28	B12	P32	P42	176
I/O	-	P29	A13	P33	P43	179
I/O	-	P30	A14	P34	P44	182
I/O	-	P31	C12	P35	P45	185
I/O	P28	P32	B13	P36	P46	188
I/O, SGCK2	P29	P33	B14	P37	P47	191
O (M1)	P30	P34	A15	P38	P48	194
GND	P31	P35	C13	P39	P49	-

บรรณานุกรม

วัลลภ สุถระกำพลธร, การประมวลผลสัญญาณเชิงเลข , สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณ - ทหารลาดกระบัง,2533.

วินัย ทองตัน และคณะ , การออกแบบและสร้างวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 , วิศวสารลาดกระบัง, ปีที่ 13 ฉบับที่ 1, กรกฎาคม 2539.

Xilink Inc, XCAT HARDWARE & PERIPHERALS GUIDE , U.S.A., April 1994 .

Xilink Inc, XCAT LIBRARIES GUIDE , U.S.A., April 1994.

Xilink Inc, THE PROGRAMMABLE LOGIC DATA BOOK , California U.S.A., April 1995.

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายณรงค์ คงสมเพ็ชร
วันเดือนปีเกิด	7 มิถุนายน พ.ศ. 2521
สถานที่เกิด	จังหวัดพระนครศรีอยุธยา
ภูมิลำเนาเดิม	50/2 หมู่ 4 ต.ภาชี อ.ภาชี จ.พระนครศรีอยุธยา 13140
ที่อยู่ปัจจุบัน	631/1 หมู่บ้านริมสวน ถ.อ่อนนุช-ลาดกระบัง แขวงลาด-กระบัง เขตลาดกระบัง กรุงเทพมหานคร 10520
โทรศัพท์	(02) 7392232
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนวัดภาชี “สุนทรอุปถัมภ์”
มัธยมศึกษาตอนต้น	โรงเรียนภาชี “สุนทรวิทยานุกูล”
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคพระนครศรีอยุธยา
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคพระนครศรีอยุธยา
ปริญญาตรี	สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	ทุนงบประมาณ
คติพจน์	คิดไกล ก้าวไกล มีวิสัยทัศน์

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายธนพล แก้วคำแจ้ง
วันเดือนปีเกิด	15 ธันวาคม พ.ศ. 2518
สถานที่เกิด	จังหวัดอุดรธานี
ภูมิลำเนาเดิม	86/2 ถ.บ้านเหล่า ต.หมากแข้ง อ.เมือง จ.อุดรธานี 41000
ที่อยู่ปัจจุบัน	631/1 หมู่บ้านริมสวน ถ.อ่อนนุช-ลาดกระบัง แขวงลาด-กระบัง เขตลาดกระบัง กรุงเทพมหานคร 10520
โทรศัพท์	(02) 7392232, (01) 6316065
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนดอนบอสโกวิทยา
มัธยมศึกษาตอนต้น	โรงเรียนอุดรพิทยานุกูล
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคอุดรธานี
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคอุดรธานี
ปริญญาตรี	สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	-
คติพจน์	ช่างมันเถอะ

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายสุรเกียรติ โสภณพุทธพร
วันเดือนปีเกิด	27 สิงหาคม พ.ศ. 2520
สถานที่เกิด	จังหวัดฉะเชิงเทรา
ภูมิลำเนาเดิม	1135/5 ม.1 ซ.แสงฟ้า ถ.พนมพัฒนา ต.พนมสารคาม อ.พนมสารคาม จ.ฉะเชิงเทรา 24120
ที่อยู่ปัจจุบัน	631/1 หมู่บ้านริมสวน ถ.อ่อนนุช-ลาดกระบัง แขวงลาดกระบัง เขตลาดกระบัง กรุงเทพมหานคร 10520
โทรศัพท์	(02) 7392232
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนวัดปัตุลาธิราชรังชสฤษฎ์
มัธยมศึกษาตอนต้น	โรงเรียนระยองวิทยาคม
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคระยอง
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคระยอง
ปริญญาตรี	สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	-
คติพจน์	คิดแล้วกระทำ กระทำแล้วนำไปคิด