ใบรับรองปริญญานิพนธ์

ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อปริญญานิพนธ์ การสร้างวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลข แบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้ VHDL และอุปกรณ์ FPGAs

IMPLEMENTATION OF 6th ORDER BUTTERWORTH LOW PASS RECURSIVE DIGITAL FILTER USING VHDL AND FPGAs DEVICE

นักศึกษา	1. นายทองปาน	ปริวัตร	รหัสประจำตัว 39031310
	2. นายธงชัย	ขำเทศเจริญ	รหัสประจำตัว 39031311
	3. นายสุระชัย	พิมพ์สาลี	รหัสประจำตัว 39031336
	4. นางสาวเสาวลักษณ์	แสงแก	รหัสประจำตัว 39031337

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต สาขาวิชา อิเล็กทรอนิกส์และคอมพิวเตอร์

อาจารย์ผู้ควบคุมปริญญานิพนธ์

1. อาจารย์กิติพงค์	มะโน
2. ศ.ดร. วัลลภ	สุระกำพลธร
2. อาจารย์วรวิทย์	สมหา
3. อาจารย์อำพล	ทองระอา

คณะกรรมการสอบปริญญานิพนธ์	ลายมือชื่อ
1. อาจารย์กิติพงค์ มะโน	.....
2. อาจารย์วรวิทย์ สมหา	.....
3. อาจารย์ปิยะ จิตธรรมมาภิรมย์	.....
4. อาจารย์อำพล ทองระอา	.....
5. อาจารย์ไพบูลย์ พวงวงศ์ตระกูล	.....

วันเดือนปีที่สอบ 14 ธันวาคม 2540 เวลา 22.00 ถึง 22.30

สถานที่สอบ ห้อง ค.310 คณะครุศาสตร์อุตสาหกรรม

ภาควิชารับรองแล้ว

.....

(รองศาสตราจารย์ ระพล เทพหัสดิน ณ อยุธยา)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่.....เดือน.....พ.ศ. .....

ปริญญานิพนธ์

การสร้างวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลขแบบบัตเตอร์เวอร์ธ อันดับที่ 6 โดยใช้ VHDL และอุปกรณ์ FPGAs

IMPLEMENTATION OF 6th ORDER BUTTERWORTH LOW PASS RECURSIVE DIGITAL FILTER USING VHDL AND FPGAs DEVICE

นายทองปาน ปริวัตร

นายธงชัย ขำเทศเจริญ

นายสุระชัย พิมพ์สาลี

นางสาวเสาวลักษณ์ แสงแก

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์

เรื่อง การสร้างวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลขแบบบัตเตอร์เวอร์ธ อันดับที่ 6 โดยใช้ VHDL และอุปกรณ์ FPGAs

IMPLEMENTATION OF 6th ORDER BUTTERWORTH LOW PASS RECURSIVE DIGITAL FILTER USING VHDL AND FPGAs DEVICE

ผู้จัดทำ

1. นายทองปาน ปริวัตร
2. นายธงชัย ขำเทศเจริญ
3. นายสุระชัย พิมพ์สาลี
4. นางสาวเสาวลักษณ์ แสงแก

อาจารย์ที่ปรึกษา

ลงนาม.....

(อาจารย์กิติพงศ์ มะโน)

ลงนาม.....

(ศ.ดร.วัลลภ สุระกำพลธร)

ลงนาม.....

(อาจารย์วรวิทย์ สมหา)

ลงนาม.....

(อาจารย์อำพล ทองระอา)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

ลงนาม.....

(ผศ.ดร.ธีระพล เทพหัสดิน ณ อยุธยา)

การสร้างวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลขแบบบัตเตอร์เวอร์ธ อันดับที่ 6 โดยใช้ VHDL และอุปกรณ์ FPGAs

นายทองปาน ปริวัตร
นายธงชัย ขำเทศเจริญ
นายสุระชัย พิมพ์สาลี
นางสาวเสาวลักษณ์ แสงแก

อาจารย์ที่ปรึกษา
อาจารย์กิติพงศ์ มะโน
ศ.ดร.วัลลภ สุระกำพลธร
อาจารย์วรวิทย์ สมหา
อาจารย์อำพล ทองระอา

ปีการศึกษา 2540

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ เสนอการออกแบบวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้หลักการของคณิตศาสตร์การกระจาย เพื่อหลีกเลี่ยงหลักการคูณกันของสัญญาณอินพุทกับค่าสัมประสิทธิ์ของวงจรกรองมาเป็นหลักการบวกกัน โดยเก็บค่าสัมประสิทธิ์ของวงจรกรองไว้ในหน่วยความจำ แล้วใช้สัญญาณอินพุทเลื่อนเข้าไปชี้ตำแหน่งของหน่วยความจำ เพื่ออ่านค่าสัมประสิทธิ์ออกมา ซึ่งจะทำให้การทำงานของวงจรเร็วขึ้น หลังจากได้ใช้ภาษา VHDL ออกแบบการทำงานและทำการสร้างวงจรด้วยอุปกรณ์ FPGAs แล้วทำการทดลองปรากฏว่า ได้ผลการตอบสนองทางเอาท์พุทได้ถูกต้อง

IMPLEMENTATION OF 6th ORDER BUTTERWORTH LOW PASS RECURSIVE DIGITAL FILTER USING VHDL AND FPGAs DEVICE

MR.THONGPAN PARIWAT
MR.THONGCHAI KHAMTHETCHAROEN
MR.SURACHAI PIMSALEE
MISS.SAOWALUK SAENGGEA

ADVISORS

MR.KITIPONG MANO
MR.WANLOP SURAKOMPHONTRON
MR.WORAWIT SOMHA
MR.AMPHON THONGRA-AR

1997

ABSTRACT

This thesis presents the 6th Order Butterworth Lowpass Recursive Digital Filter was designed from the principle of arithmatic distribution to prevent from multiplication theory of input signal with coefficient value of filter circuit to be addition theory by collecting coefficient value of filter circuit in memory unit. Then, the input signal was shifted to post the memory unit and read out the coefficient value. It enabled the circuit to operate faster. The results of testing this design were relevant to the output respond.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์นี้สามารถสำเร็จลุล่วงไปได้ด้วยดี จากความร่วมมือของสมาชิกภายในกลุ่มทุกท่าน นอกจากนี้ยังได้รับความกรุณาจากอาจารย์ที่ปรึกษาประจำโครงงาน และอาจารย์ประจำภาควิชาครุศาสตร์วิศวกรรมทุกท่านในการให้คำปรึกษาแนะนำ และให้ความช่วยเหลือต่างๆ ตลอดจนให้โอกาสในการทำโครงงานอย่างเต็มที่ ทั้งเวลา สถานที่ เครื่องมือ และอุปกรณ์ต่างๆ และขอขอบพระคุณบุพการีผู้ให้กำเนิดที่ให้โอกาสในการศึกษา เพื่อน ตลอดจนผู้ที่เกี่ยวข้องทุกคนที่ให้คำแนะนำต่างๆ และเป็นกำลังใจในการทำงานจนทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี

สารบัญ

เรื่อง หน้า

เรื่อง หน้า

สารบัญตาราง

สารบัญภาพ

รูปภาพ หน้า

รูปภาพ หน้า

รูปภาพ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

ด้วยในปัจจุบันได้มีการนำคอมพิวเตอร์มาช่วยในการออกแบบวงจรอิเล็กทรอนิกส์ และนำผลการออกแบบด้วยคอมพิวเตอร์ไปสร้างเป็นวงจรด้วยอุปกรณ์ประเภทวงจรรวมเฉพาะงานกันเป็นจำนวนมาก โดยโปรแกรมสำหรับการออกแบบมีหลายระดับด้วยกัน เช่น การออกแบบระดับเกท ระดับทรานซิสเตอร์ ระดับ Layout และระดับฟังก์ชัน โดยแต่ละระดับจะมีข้อดีข้อด้อยแตกต่างกัน แต่อย่างไรก็ตามในโครงงานนี้สนใจการออกแบบวงจรอิเล็กทรอนิกส์เชิงเลขซึ่งมีหลายภาษาด้วยกัน และที่กำลังได้รับความนิยมมากขณะนี้คือภาษา VHDL (Very High Speed Integrated Circuit Hardware Description Language : VHSIC-HDL) เนื่องจากเป็นภาษาที่มีความสามารถในการทำงานสูงคือ ง่ายในการออกแบบ มีความเร็วสูง สามารถทำการปรับปรุงแก้ไขได้ง่าย สามารถออกแบบระบบที่มีความซับซ้อนได้ดี โดยการบรรยายพฤติกรรมของฮาร์ดแวร์ (Hardware) เพื่อให้การออกแบบฮาร์ดแวร์เป็นไปอย่างมีระบบและสะดวกมากยิ่งขึ้น

ดังนั้นโครงงานนี้จึงได้เลือกออกแบบวงจรเชิงเลขโดยใช้ภาษา VHDL ด้วยเช่นกัน โดยในโครงงานได้ออกแบบเป็นส่วนประมวลผลของวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 และนำผลของการออกแบบดังกล่าวมาสร้างเป็นอุปกรณ์ใช้งานจริงโดยใช้อุปกรณ์ประเภทวงจรรวมเฉพาะงาน (ASIC) ที่มีชื่อว่า FPGAs (Field Programable Gate Array) เพื่อทดสอบการทำงานของอุปกรณ์และระบบการทำงานทั้งหมดของวงจรกรองเชิงเลข เพื่อนำไปประยุกต์ใช้ภายในระบบต่างๆ ที่เกี่ยวข้องกับการใช้วงจรกรองสัญญาณเชิงเลขและเพื่อให้สามารถนำภาษา VHDL ไปใช้ในการออกแบบวงจรอื่นๆ ต่อไปได้

1.2 ขีดความสามารถของโครงงาน

1. สามารถสร้างวงจรกรองสัญญาณโดยใช้อุปกรณ์ FPGAs ได้

2. สามารถเปลี่ยนแปลงความถี่คัทออฟ (โดยการเปลี่ยนข้อมูลของหน่วยความจำ) ในพิสัย 20 Hz-20 Hz

3. สามารถประยุกต์ใช้เป็นวงจรกรองสัญญาณในการประมวลผลสัญญาณเชิงเลขอื่นได้

1.3 เนื้อหาโดยสังเขป

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้แบ่งออกเป็นบทต่างๆ เพื่อความสะดวกต่อการศึกษาและทำความเข้าใจในแต่ละบทจะประกอบด้วยเนื้อหาที่สำคัญดังนี้

บทที่ 2 ทฤษฎีและหลักการ ซึ่งจะกล่าวถึง VHDL กับการออกแบบอิเล็กทรอนิกส์, ส่วนประกอบหลักของภาษา VHDL, องค์ประกอบในการเขียนภาษา VHDL, สัญญาณและชนิดของข้อมูลในภาษา VHDL, ตัวดำเนินการของภาษา VHDL, Logic cell array family, โครงสร้างภายในของ FPGAs, รายละเอียดการใช้งานของอุปกรณ์ FPGAs, ข้อควรระวังในการใช้อุปกรณ์ FPGAs, ขั้นตอนการออกแบบและสังเคราะห์โดยใช้ซอฟแวร์ของบริษัทไซลิงค์, ขั้นตอนการจำลองการทำงานโดยใช้ซอฟแวร์ของบริษัทไซลิงค์, และขั้นตอนในการโปรแกรมลงบนอุปกรณ์ FPGAs โดยใช้ซอฟแวร์ XDM ของบริษัทไซลิงค์ (Xilinx)

บทที่ 3 การออกแบบ และการสร้างซึ่งจะกล่าวถึง ความคิดเบื้องต้นในการทำดิจิตอลฟิลเตอร์, ดิจิตอลฟิลเตอร์แบบคณิตแจกแจง, หลักการของโครงสร้างแบบเลขคณิตแจกแจง, การสร้างตารางเปิดดูจากฟังก์ชัน $F_i\{.\}$ สำหรับตัวกรองอันดับสอง, การคำนวณหา Y(n) ในกรณีตัวกรองอันดับสอง, การคำนวณหา Y(n) ในกรณีตัวกรองอันดับหก, การออกแบบสร้างวงจรกรองสัญญาณเชิงเลขอันดับหก, ลักษณะการออกแบบ, วิธีการออกแบบ, การออกแบบวงจรกรองสัญญาณเชิงเลขด้วยภาษา VHDL และขั้นตอนการออกแบบบล็อกการทำงานภายใน

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึงผลการทดลองโปรแกรมโปรแกรม VHDL, การดาวน์โหลดโปรแกรมลงบนบอร์ดตัวอย่างของ FPGAs, และผลการทดลอง

บทที่ 5 สรุปอภิปรายและข้อเสนอแนะ กล่าวถึงปัญหา, แนวทางแก้ไข, และแนวทางในการพัฒนาเพื่อสามารถนำไปประยุกต์ใช้ได้อย่างกว้างขวาง

ในภาคผนวกแสดงรายละเอียดของโปรแกรม ผังงาน และรายการอุปกรณ์ต่างๆ ที่ใช้จัดทำโครงงานดังนี้

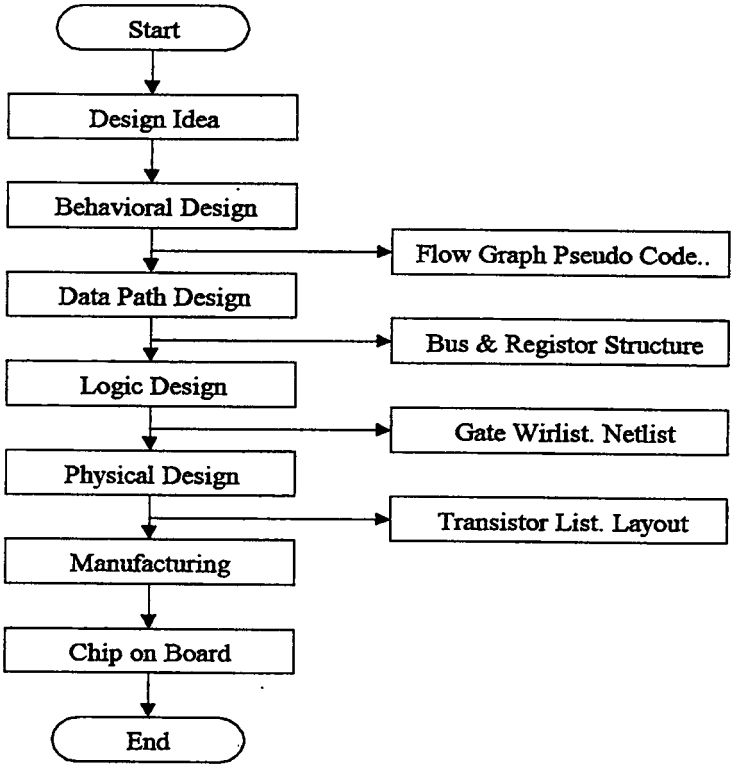
ภาคผนวก ก ผังการทำงานและโปรแกรมการทำงานของวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6

ภาคผนวก ข รายละเอียดข้อมูล และคุณสมบัติของอุปกรณ์

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

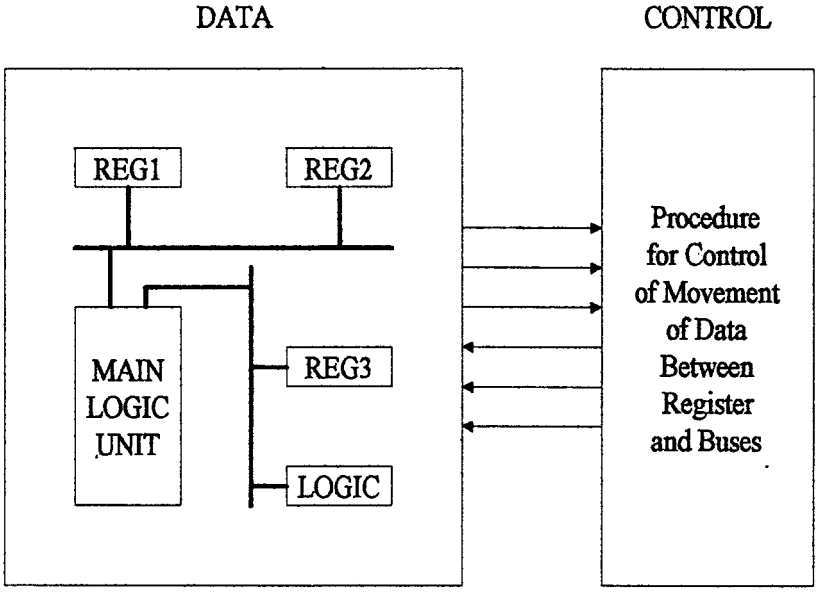
เพื่อเป็นการแก้ปัญหาที่ยุ่งยากในการออกแบบวงจรเชิงเลขจึงได้นำเอาระบบคอมพิวเตอร์ช่วยออกแบบทางวิศวกรรมเข้ามาช่วยในการออกแบบ ทั้งยังช่วยลดความยุ่งยากและระยะเวลาในการออกแบบให้รวดเร็วขึ้น ภาษาสำหรับบรรยายอุปกรณ์ฮาร์ดแวร์เป็นภาษาหนึ่งที่ได้รับการพัฒนา และนำมาใช้ในการออกแบบระบบเชิงเลข โดยที่มีขั้นตอนตั้งแต่การกำหนดแนวความคิดในการออกแบบจนกระทั่งได้มาเป็นอุปกรณ์ ที่ถูกโปรแกรมการทำงานตามการวางแนวความคิดให้อุปกรณ์ทำงานได้ตามต้องการ



รูปที่ 2.1 ขั้นตอนการออกแบบระบบเชิงเลข

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

ขั้นต่อมาคือ การออกแบบเส้นทางเดินของข้อมูลให้เชื่อมต่อกันอาจจะเชื่อมต่อกันด้วยระบบบัสหนึ่งหรือสองทิศทาง (Unidirectional or Bidirrectional Bus) เพื่อเชื่อมต่อรีจิสเตอร์ (Registor) และวงจรตรรก (Logic) ให้เป็นระบบที่สมบูรณ์ดังรูปที่ 2.2



รูปที่ 2.2 การออกแบบระบบเส้นทางของข้อมูล

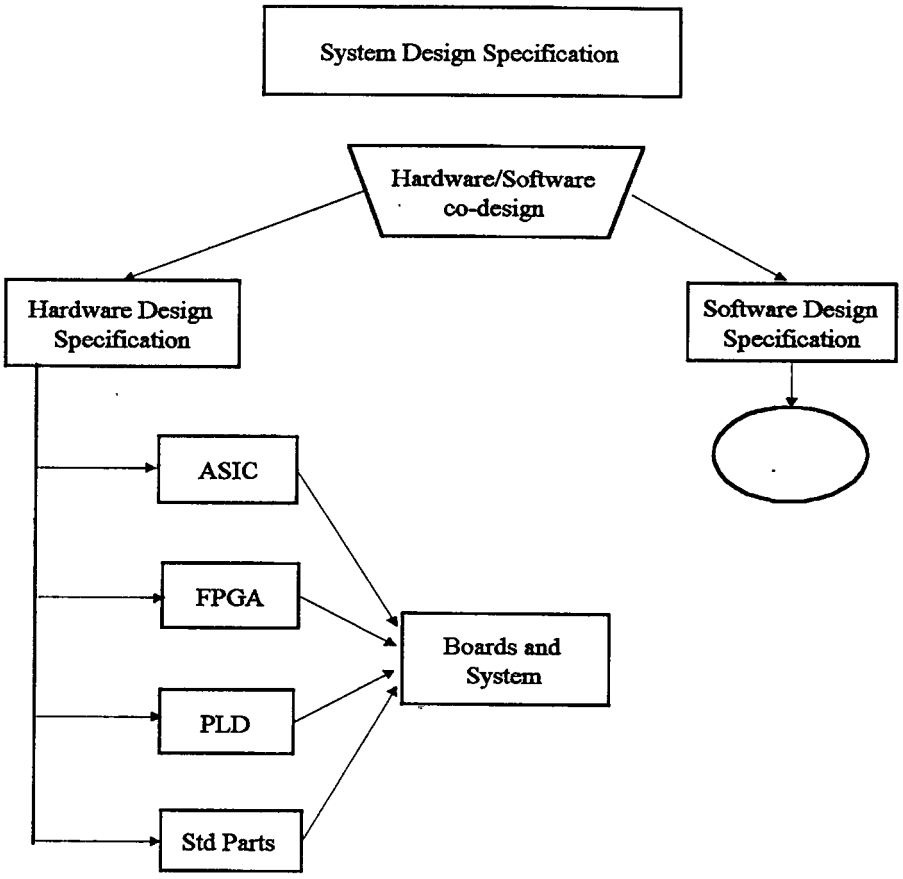
จากนั้นจะเป็นขั้นตอนของการเชื่อมโยงเกทพื้นฐานและฟลิปฟลอป จนมาถึงขั้นตอนการออกแบบทางกายภาพ คือเป็นการทำทรานซิสเตอร์ลิส และการเลเอาท์ (Transistor List and Latout) จนกระทั่งเป็นขั้นตอนการผลิตลงบน Chip ซึ่งอาจเป็นอุปกรณ์ FPGAs, PLD, หรือ Std Port หรืออาจจะเป็นการส่งระบบที่ออกแบบสมบูรณ์และไปทำการเจือสารที่โรงงานผลิตออกมาเป็นวงจรรวม

2.2 แนะนำภาษา VHDL

VHDL เป็นภาษาบรรยายทางฮาร์ดแวร์สำหรับการออกแบบในระดับสูงทางอิเล็กทรอนิกส์ (Electronics) มีความสามารถในการเลียนแบบ (Simulation), การสังเคราะห์

(Synthesis), และการทดสอบ (Testing) ในการจำลองระบบอิเล็กทรอนิกส์ดิจิตอลลักษณะการทำงานของ VHDL จะทำงานไปพร้อมๆ กัน (Concurrent) และเป็นลำดับ (Sequential Statements) ซึ่งหมายถึงทุกๆ คำสั่ง องค์ประกอบเกท หรือวงจรตรรกจะนำมาปฏิบัติทั้งหมด จึงดูเหมือนว่าได้มีการปฏิบัติไปพร้อมๆ กัน

2.3 VHDL กับการออกแบบอิเล็กทรอนิกส์

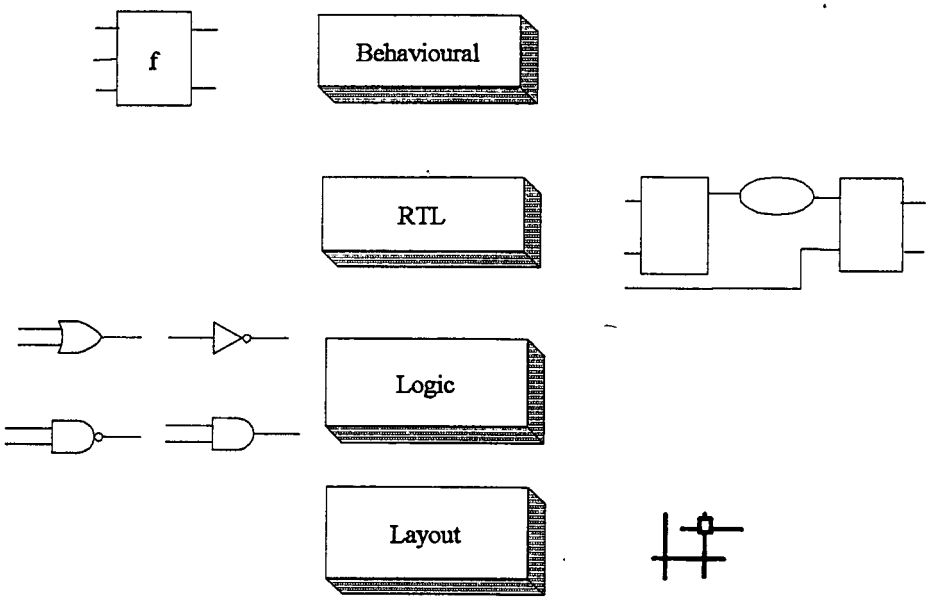


รูปที่ 2.3 การออกแบบระบบอิเล็กทรอนิกส์

จากรูปที่ 2.3 แสดงกระบวนการการทำงานของภาษา VHDL จากการออกแบบทางฮาร์ดแวร์ผ่านกระบวนการแปลภาษา (Compile) การจำลองการทำงาน (Simulate) เพื่อตรวจสอบดูว่าผลการทำงานเป็นไปตามจุดประสงค์หรือไม่ จากนั้นนำไปสังเคราะห์ เพื่อให้ได้เป็นวงจรแสดงการเชื่อมต่อซึ่งประกอบด้วยเกท (Gate), ฟลิปฟลอป (Flip-Flop), และ

อุปกรณ์พื้นฐานต่างๆ แล้วนำไปโปรแกรมลงบนอุปกรณ์ประเภท ASIC, FPGA, PLD หรือ Std part จนเป็นวงจรรวมแบบสมบูรณ์

2.3.1 ขั้นตอนในการออกแบบ



รูปที่ 2.4 ขั้นตอนการออกแบบ

1. Behavioral เป็นขั้นตอนในการสร้างแบบจำลองการบรรยายพฤติกรรมในขั้นตอนแรก
2. RTL (Register Transfer Logic) การบรรยายการถ่ายโอน Logic
3. Logic เป็นขั้นตอนการสังเคราะห์ให้ได้เป็นวงจรแสดงการเชื่อมต่อซึ่งประกอบด้วยเกทต่างๆ
4. Layout เป็นขั้นตอนของการเจือสาร

ในส่วนของภาษา VHDL ครอบคลุมในขั้นตอน Behavioral,RTL และ Logic

วัตถุประสงค์ในขั้นตอนของ Behavioural มาถึงขั้น RTL ก็เพื่อ

1. ช่วยในการกำหนดการทำงานและการแก้ไขจุดผิดพลาดของโปรแกรม
2. ช่วยในการวิเคราะห์ปัญหาของโปรแกรม
3. เป็นวิธีในการระบุรายละเอียดและการแยกส่วนการทำงานของโปรแกรม
4. เป็นการตรวจสอบและเปรียบเทียบผลลัพธ์ของโปรแกรม

2.3.2 ข้อได้เปรียบของการใช้ VHDL

1. การออกแบบมีคุณภาพที่สูงกว่า
2. ออกแบบวงจรที่มีความซับซ้อนมากๆ ได้
3. ใช้เวลาในการออกแบบสั้นกว่า
4. ค้นหาข้อผิดพลาดและเปลี่ยนแปลงแก้ไขได้ง่าย

ภาษา VHDL สามารถใช้ในการบรรยายได้ 3 ลักษณะคือ

2.4 การบรรยายเชิงพฤติกรรม

เป็นการบรรยายลักษณะการทำงานของอุปกรณ์ฮาร์ดแวร์ในเชิงพฤติกรรม เป็นการบรรยายลักษณะการเปลี่ยนแปลงของข้อมูลในรูปแบบของอัลกอริธึม สำหรับการคำนวณผลลัพธ์ที่เกิดขึ้นสืบเนื่องมาจากการเปลี่ยนแปลงสภาวะของข้อมูลที่เข้ามา โดยไม่คำนึงถึงว่าลักษณะโครงสร้าง หรือความสัมพันธ์ของอุปกรณ์ที่อยู่ภายใน

2.5 การบรรยายเชิงข้อมูล

เป็นการบรรยายถึงการเลื่อนไหลของข้อมูลผ่านรีจีสเตอร์ และบัสของระบบเป็นระดับขั้นของการบรรยายที่อยู่ตรงกลางระหว่างการบรรยายเชิงพฤติกรรม และการบรรยายเชิงโครงสร้างเครื่องมือที่ใช้ในการควบคุมการไหลเลื่อน ได้แก่ Conditional, Selected และ Guarded

2.6 การบรรยายเชิงโครงสร้าง

เป็นการบรรยายการทำงานของระบบในเชิงโครงสร้างจะต้องแสดงรายการของอุปกรณ์ทั้งหมดที่ใช้ในระบบ และต้องกำหนดการเชื่อมต่อระหว่างอุปกรณ์ต่างๆ ด้วยเพราะว่าการบรรยายในระดับนี้เป็นการบรรยายที่ใกล้เคียงลักษณะของฮาร์ดแวร์มากที่สุด ภาษา VHDL ได้จัดเตรียมเครื่องมือ และลักษณะโครงสร้างของการบรรยายในลักษณะนี้ไว้ที่สำคัญ 4 ลักษณะคือ

1. ความสามารถในการเลือกหรือกำหนดอุปกรณ์ที่ต้องการได้จากไลบรารี
2. การสร้างไลบรารี่เพื่อเก็บอุปกรณ์ที่ผู้ใช้ออกแบบไว้เองได้
3. กลไกในการเชื่อมต่อระหว่างอุปกรณ์

4. โครงสร้างของการกำหนดอุปกรณ์ชนิดเดียวกันซ้ำๆกัน

2.7 ส่วนประกอบหลักของภาษา VHDL

2.7.1 โครงสร้าง

จากเอกสารของ DOD (The United State Department of Defense) ได้กล่าวไว้ว่าภาษาบรรยายทางฮาร์ดแวร์ เป็นภาษาที่ต้องการการกำหนดรูปแบบทางโครงสร้างเพราะการกำหนดโครงสร้างนี้จะช่วยในการออกแบบ ทั้งระบบที่ง่ายไปจนถึงระบบที่ซับซ้อน ซึ่งระบบที่ว่านี้ถือเป็นมาตราฐานของภาษาบรรยายทางฮาร์แวร์

2.7.2 การทำงานที่พร้อมกัน

ในวงจรอิเล็กทรอนิกส์ อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอและจะมีเรื่องของเวลาเข้ามาควบคุมการทำงานเสมอ ภาษา VHDL ได้รับการออกแบบมาเพื่อบรรยายรูปแบบทางดิจิตอล และการป้องกันของเวลา สำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วนของสถาปัตยกรรม (Architecture) จะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซสซึ่งมีการทำงานภายในเป็นแบบลำดับคำสั่งก็ตาม หากมีหลายโปรเซสอยู่ภายในโครงสร้างเดียวกันทุกโปรเซสจะทำงานไปพร้อมกันด้วย

2.7.3 ลำดับคำสั่ง

ถึงแม้ว่าลักษณะเฉพาะของภาษาบรรยายทางฮาร์ดแวร์ จะสนับสนุนการปฏิบัติตามคำสั่งอย่างเป็นลำดับเป็นกระบวนการ ตัวภาษา VHDL ก็ยังได้จัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งเอาไว้ เมื่อนักออกแบบได้บรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดในของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรม ที่ประกอบด้วยโครงสร้างแบบ if-then-else และ loop การบรรยายแบบลำดับคำสั่ง ทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้สะดวกและง่ายขึ้น แต่อย่างไรก็ตามการทำงานของภาษา VHDL ก็ยังเป็นการทำงานแบบพร้อมเพรียงกันอยู่

2.7.4 เวลาที่ใช้ในการควบคุม

ภาษา VHDL เป็นภาษาที่อนุญาตให้ผู้ออกแบบ สามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ การตรวจสอบการออกแบบเกท หรือการหน่วงเวลาสามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอยเหตุการณ์ นอกจากนี้ก็ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้

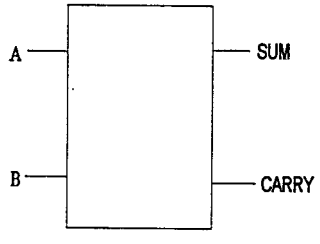
2.8 องค์ประกอบในการเขียนภาษา VHDL

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

2.8.1 Entity

เป็นส่วนที่ใช้ในการประกาศ อธิบายการเชื่อมต่อการติดต่อกับองค์ประกอบภายนอก แต่ยังมิได้กำหนดการกระทำใดๆ ทั้งสิ้น ซึ่งในส่วนประกาศนี้จะต้องกำหนดชื่อ (ห้ามซ้ำกับชื่อสงวนของภาษา VHDL) แล้วตามด้วย is ตัวอักษรในภาษา VHDL ทั้งตัวใหญ่ตัวเล็กมีค่าเท่ากันดังตัวอย่างแสดงในรูปที่ 2.5

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



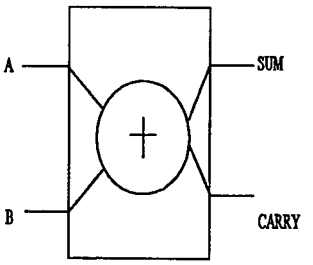
รูปที่ 2.5 โปรแกรมการใช้งานและลักษณะการติดต่อใช้งานของ entity

จากรูปที่ 2.5 จะเห็นว่ามีการประกาศชื่อ entity HALFADD ตาม is แล้วมีการกำหนดสัญญาณอินพุทและเอาท์พุท โดยมีวงเล็บเปิดและวงเล็บปิด ในแต่ละบรรทัดจะถูกปิดท้ายด้วย (;) และใช้ในการกำหนดสัญญาณให้กับตัวแปรและจบ entity ด้วยคำสั่ง end ตามด้วยชื่อของ entity

2.8.2 Architecture

เป็นส่วนที่ใช้การกำหนดการทำงานภายในของ entity ต้องเป็นการเชื่อมโยงกันของ entity จำเพาะซึ่ง entity เดียวสามารถมีได้หลาย Architecture จะต้องมีการกำหนดชื่อของ Architecture นั้นๆ ด้วยว่าเป็นของ entity ไหน การเขียน Architecture จะตามหลัง begin เสมอ และจบด้วยคำสั่ง 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

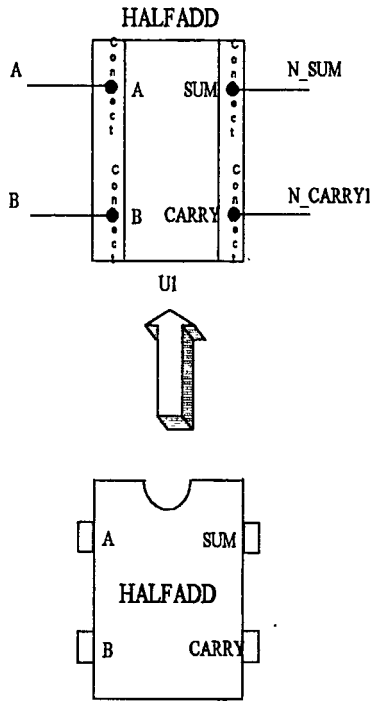
2.8.3 Configuration

ในภาษา VHDL สามารถสร้างอุปกรณ์เก็บเอาไว้ได้ และสามารถจำลอง Socket ของอุปกรณ์แต่ละตัว เพื่อความสะดวกเหมือนกับการใช้งานจริงในการเชื่อมต่ออุปกรณ์ดังรูปที่ 2.7 u1 ถูกสร้างขึ้นเป็น Socket ให้กับอุปกรณ์ โดยในการนำไปใช้งานนั้นจะอาศัย Component ตามด้วยชื่ออุปกรณ์มีการกำหนดสัญญาณอินพุทเป็นแบบบิทให้กับตัวแปร A และ B ส่วนเอาท์พุทก็เป็นสัญญาณแบบบิทที่กำหนดค่าให้กับตัวแปร SUM และ CARRY เหมือนกับ entity HALFADD ทุกประการ และจบด้วยคำสั่ง end component; สำหรับการเชื่อมต่อจะใช้คำสั่ง port map โดยเป็นการให้กับทางขวามือ คือ u1: HALFADD port map (A,B,N_SUM,N_CARRY);โดยการเขียนนี้ต้องอยู่หลังจาก begin ดังตัวอย่างแสดงในรูปที่ 2.8 และรูปที่ 2.9

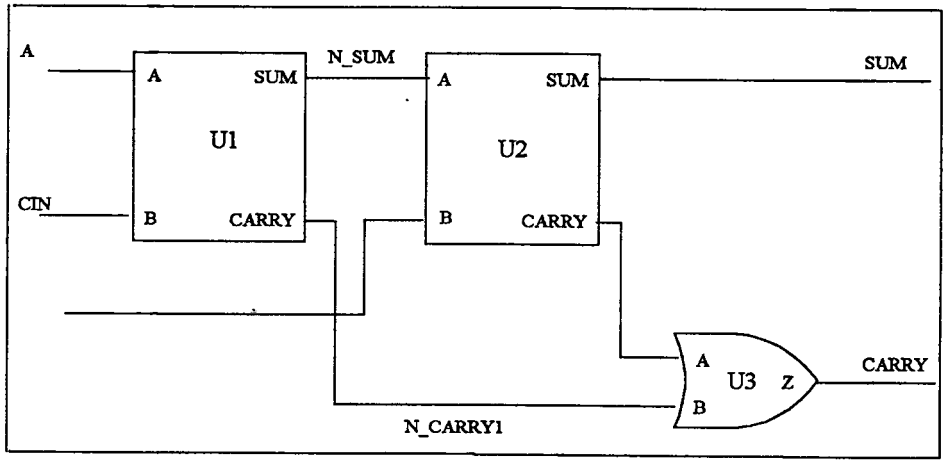
```
component HALFADD
port (A,B : in bit;
      SUM,CARRY : out bit);
end component;
...
begin
u1:HALFADD port map  (A,B,N_SUM,N_CARRY);
...
```

```
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_CRRY1, 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,SUM,N_CARRY2);
      u3 : OSGATE port map (N_CARRY2, N_CARRY1 ,CARRY);
end STURCTURAL;
```

รูปที่ 2.9 โปรแกรมการเชื่อมต่ออุปกรณ์ภายในและการประกาศสายสัญญาณ

จากรูปที่ 2.9 มีการประกาศสัญญาณภายในอุปกรณ์คือ Signal N_CARRY1, N_CARRY2 ให้มีสัญญาณเป็นแบบบิท จะสังเกตเห็นว่าวงจรมีอุปกรณ์ภายใน 3 ตัว คือ HALFADD 2 ตัวและ ORGATE 1 ตัวจึงประกอบด้วย u1, u2, u3 และสัญญาณที่เขียนใน port map จะเรียงสัญญาณเข้าจนถึงสัญญาณออกตามลำดับและจบโปรแกรมด้วยคำสั่ง end STRUCTURAL ตามชื่อของ Architecture เสมอส่วนของ component เป็นส่วนหนึ่งของ Architecture

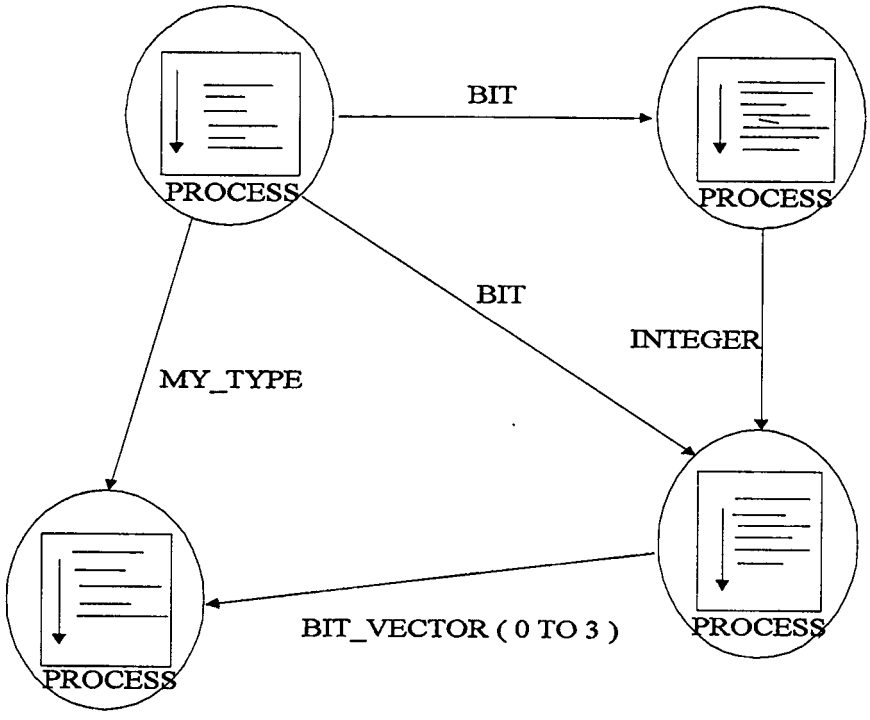
2.8.4 Process

ส่วนนี้เป็นการบรรจุคำสั่งลำดับที่ต้องเขียนอยู่ภายใน Architecture ซึ่งโปรเซสหลายๆ โปรเซสจะมีการทำงานที่พร้อมกัน (concurrent) ตัวแปรทางขวามือจะถูกกำหนดค่าให้จากสัญญาณทางซ้ายมือ (<=) ในการเริ่มโปรแกรมใช้คำสั่ง PROCESS และจบด้วยคำสั่ง END PROCESS ส่วนประกอบของการบรรยายโปรเซสประกอบด้วยส่วนของการประกาศตัวแปรที่ต้องใช้และส่วนของการปฏิบัติตามคำสั่งเพื่อให้ได้ผลลัพธ์ตามที่ต้องการ ดังตัวอย่างที่ต้องการในรูปที่ 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 มีสัญญาณอินพุทเป็น 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 Process ด้วย end process AND_OR_FUNC ; และจบ Architecture ด้วยคำสั่ง end BEHAVE;



รูปที่ 2.11 การเชื่อมต่อหลายๆ process ที่สามารถเชื่อมต่อกันได้ด้วยสัญญาณที่ต่างกัน

2.8.5 Package

เป็นส่วนของโปรแกรมที่แปล (compile) เรียบร้อยแล้ว และดึงเอามาใช้เท่านั้นหรืออาจกล่าวได้ว่า package คือกลุ่มชนิดของข้อมูล โปรแกรมย่อยหรืออุปกรณ์ต่างๆ ที่ได้ออกแบบไว้แล้วนำมารวบรวมไว้เป็นกลุ่มๆ อยู่ภายในเพื่อให้ผู้ออกแบบเรียกใช้ได้สะดวกดังรูปที่ 2.12

```
package DEMO_PACK is
    -- constants
    -- data types
    -- components
    -- subprogram
end DEMO_PACK
```

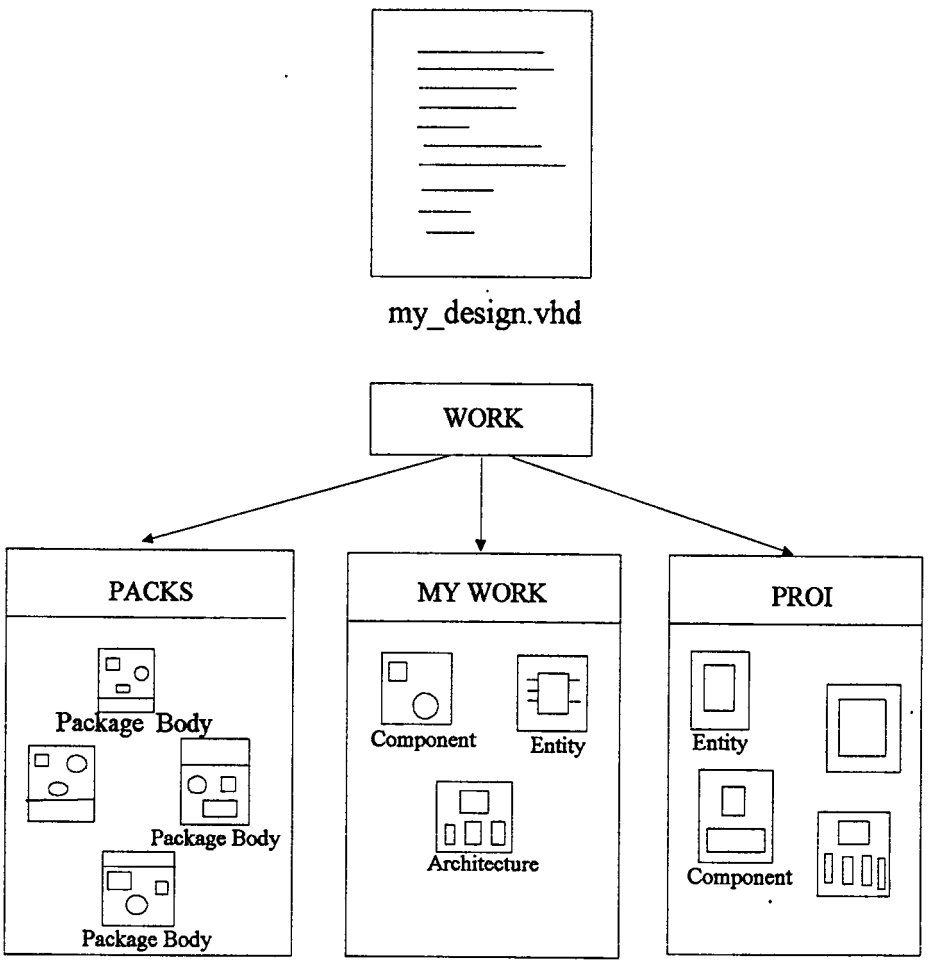
รูปที่ 2.12 โครงสร้างของ Package

```
package body DEMO_PACK is
    --constant values
    --subprogram definition
end DEMO_PACK;
```

รูปที่ 2.12 (ต่อ) โครงสร้างของ package

2.8.6 Library

เป็นส่วนของการเก็บโปรแกรมต่างๆ ที่สมบูรณ์ซึ่งภายใน library จะประกอบด้วย entity, architecture, package, configulation จะมีลักษณะการใช้งานดังรูปที่ 2.13



รูปที่ 2.13 ส่วนของการเก็บโปรแกรมต่างๆ ที่สมบูรณ์

จากรูปที่ 2.13 ก่อนการเรียกใช้ work จะต้องทำการแปล (complie) ที่ packs, mywork และ porj ก่อนแล้วจึงสามารถเรียกใช้ได้

2.9 สัญญาณและชนิดของข้อมูลในภาษา VHDL

สัญญาณมีลักษณะเป็นเสมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูลและมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วย การกำหนดค่าให้สัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่า และสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการส่งผ่านค่าของสัญญาณเช่น w <=a AFTER 12 nS หมายถึงการกำหนดค่าของสัญญาณ a ให้กับ w หลังจากเวลาผ่านไป 12 nS ในทางตรงกันข้ามตัวแปรมีลักษณะเป็นเสมือนตัวกลางซอฟแวร์ที่ใช้ในการส่งผ่านข้อมูลและไม่มีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยการกำหนดค่าให้กับตัวแปรจะใช้สัญลักษณ์ := ตัวแปรจะถูกใช้ในส่วนที่มีการทำงานเป็นแบบลำดับคำสั่งในฟังก์ชัน โพรซีเจอร์ และโพรเซส

2.9.1 ลักษณะข้อมูลมาตราฐาน

1. กลุ่มของข้อมูลชนิดต่างๆต้องสามารถนำมาประเมินค่าได้
2. กลุ่มของข้อมูลมาตราฐานสามารถให้คำจำกัดความได้
3. ผู้ใช้สามารถที่จะนำข้อมูลเหล่านี้ไปใช้ได้ถูกต้อง ต้องดูที่คำจำกัดความของข้อมูลชนิดนั้นเพื่อไม่ให้เกิดความผิดพลาด

2.9.2 ชนิดของข้อมูลมาตราฐาน

1. FALSE,TRUE นำไปใช้ในสมการ BOOLEAN
2. BIT เช่น “0”, “1”
3. BITVECTOR เช่น “10001”, “10100101”
4. CHARACTER เช่น “A”, “C”, “R”, “2”, “$”
5. ทศนิยม เช่น “1.02”, “0.436”, “1.0e-10”, “-4.356e45”
6. กำหนดเป็นข้อความ เช่น “% Error 56.3”, “messag string”
7. แบบจำนวนเต็ม เช่น “1”, “254”, “-456”
8. แบบกำหนดค่าเวลา เช่น “40 nS”, “5.5 pS”

ข้อมูลเหล่านี้จะถูกเก็บอยู่ใน package ดังรูปที่ 2.14

```
package STANDARD is
        type BOOLEAN is (FALSE,TRUE);
        type BIT is ('0', '1');
        type CHARACTER is (-- ascii set);
        type INTEGER is range
                        implementation_defined;
        type REAL is range
                        implementation_defined;
        BIT_VECTOR, STRING,TIME
end STANDARD;
```

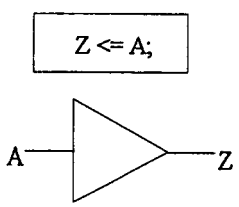
รูปที่ 2.14 การเขียนโปรแกรมของข้อมูลภายใน package

2.10 สัญญาณและการกระตุ้น (signal and driver)

```
signal A,B,Z : bit;
signal  X_INT : integer;
```

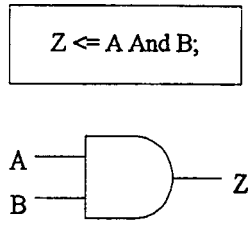
รูปที่ 2.15 การกระตุ้นสัญญาณให้ A, B, Z

จากรูปที่ 2.15 เป็นการกระตุ้นสัญญาณให้ A, B, Z เป็นสัญญาณแบบ Bit, X_INT เป็น interger



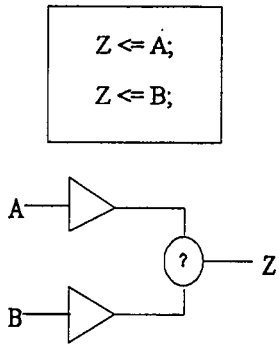
รูปที่ 2.16 การส่งค่า A ให้ Z

จากรูปที่ 2.16 ตัวกระทำทางด้านขวาคือ A ส่งค่าให้ Z



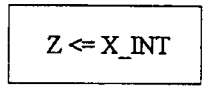
รูปที่ 2.17 การให้ค่า A And B แก่ Z

จากรูปที่ 2.17 เป็นการให้ค่าทางขวามือคือ A And B แก่ Z และยังมีรูปแบบใช้ไม่ได้ เช่น



รูปที่ 2.18 การส่งค่า A และ B ที่ไม่สามารถส่งให้แก่ Z ได้

จากรูปที่ 2.18 ค่า A และ B ไม่สามารถส่งให้แก่ Z ได้ เพราะอยู่ในสภาวะที่ไม่สามารถตัดสินใจได้ หรือรูปแบบในการกำหนดข้อมูลคนละชนิดให้แก่ตัวแปรก็ไม่สามารถทำได้



รูปที่ 2.19 การส่งค่า X_INT ให้แก่ Z

จากรูปที่ 2.19 Z ถูกกำหนดให้มีค่าเป็น Bit ส่วน X_INT ถูกกำหนด integer

2.11 ข้อมูลแบบเป็นชุด

เป็นข้อมูลเหมือนกับข้อมูลทั่วไปเพียงแต่มีลักษณะของข้อมูลเป็นชุด ซึ่งข้อมูลจะถูกกำหนดค่าข้อมูลเรียงตามลำดับ ดังตัวอย่างในรูปที่ 2.20

```
signal A_BUS, B_BUS, Z_BUS : bit_vector (3 downto 0)
```

รูปที่ 2.20 การกำหนดค่าข้อมูลแบบเป็นชุด

จากรูปที่ 2.20 เป็นการกำหนดค่า bit_vector 4 ค่า คือจาก 0-3 โดยค่าถูกกำหนดจาก 3, 2, 1, 0

```
Z_BUS <= A_BUS;
```

```
Z_BUS(3) <= A_BUS(3)
Z_BUS(2) <= A_BUS(2)
Z_BUS(1) <= A_BUS(1)
Z_BUS(0) <= A_BUS(0)
```

รูปที่ 2.21 การให้ค่า A_BUS แก่ Z_BUS

จากรูปที่ 2.21 ค่าจะเรียงตามลำดับ Z_BUS(3) ถูกกำหนดค่าให้จาก A_BUS(3) ตามลำดับ

```
Z_BUS(3) <= A_BUS(0)
Z_BUS(2) <= A_BUS(1)
Z_BUS(1) <= A_BUS(2)
Z_BUS(0) <= A_BUS(3)
```

รูปที่ 2.22 การกำหนดค่า A_BUS(0) ให้แก่ Z_BUS(3)

จากรูปที่ 2.22 เป็นการกำหนดค่า Z_BUS(3) ถูกกำหนดค่าให้จาก A_BUS(0)

สิ่งที่ต้องคำนึงเมื่อให้ข้อมูลแบบเป็นชุดคือ ขนาดของ Array ข้างซ้ายและข้างขวาจะต้องเท่ากัน และสมาชิกแต่ละตัวจะถูกกำหนดไว้โดยตำแหน่ง ดังแสดงในรูปที่ 2.23

```
signal A_BUS, B_BUS, Z_BUS : bit_vector (3 downto 0);
signal A_BIT, B_BIT, C_BIT, D_BIT : bit;
signal BYTE : bit_vector (7 downto 0);

Z_BUS <= (A_BIT, B_BIT, C_BIT, D_BIT);

BYTE <= (7 => '1', 5 downto 1 => '1', 6 => B_BIT, others => '0');
```

รูปที่ 2.23 แสดงข้อมูลเป็นกลุ่ม (Aggregates)

2.12 ตัวดำเนินการภาษา VHDL (Operators)

การบรรยายเชิงพฤติกรรมในภาษา VHDL ก็มีตัวกระทำทางลอจิกและคณิตศาสตร์เช่นเดียวกับภาษาซอฟแวร์ทั่วไปดังตารางที่ 2.1

ตารางที่ 2.1 ตัวดำเนินการภาษา VHDL

PREDEFINED OPERATORS
LOGICAL OPERATORS : NOT AND OR NAND NOR XOR
OPERAND TYPE : BIT BOOLEAN
RESULT TYPE : BIT BOOLEAN
RETLATIONAL OPERATORS : = /= < <= > >=
OPERAND TYPE : any type
RESULT TYPE : Boolean
ARITMETIC OPERATORS : + - * / MOD REN ABS
OPERAND TYPE : INTEGER REAL Physical
RESULT TYPE : INTEGER REAL Physical
CONCENTENATION OPERATORS : &
OPERAND TYPE : array of any type
RESULT TYPE : array of any type

2.13 คำสั่งการทำซ้ำ (Sequential Statements)

2.13.1 คำสั่ง If-then-else

```
if CONDITION then
    -- sequential statements
end if;
```

```
if CONDITION then
    -- sequential statements
else
    -- sequential statements
end if;
```

```
if CONDITION then
    -- sequential statements
elsif CONDITION then
    -- sequential statements
elsif CONDITION then
    -- sequential statements
else
    -- sequential statements
end if;
```

รูปที่ 2.24 รูปแบบของคำสั่ง if-then-else

```
brary IEEE;
use IEEE.Std_Logic_1164.all;
entity IF_EXANPLE is
port (A, B, C, X : in   std_ulogic_vector(3 downto 0);
      Z          : out std_ulogic_vector(3 downto 0);
end IF_EXAMPLE;
Architecture A of IF_EXAMPLE is
begin
   process (A, B, C, X)
   begin
         if (x = "0000") then
            Z <= A;
         elsif (X <= "0101") then
            Z <= B;
         else
            Z <= C;
         end if
   end process;
end A;
```

รูปที่ 2.25 ตัวอย่างการใช้คำสั่ง If-then-else

จากรูปแบบของคำสั่ง If-then-else ในรูปที่ 2.24 นำมาเขียนตัวอย่างดังรูปที่ 2.25 คำสั่ง if จะกระทำภายใน Architecture และอยู่ภายใน Process โดยที่คำสั่ง if 2 สเตจ โดยเริ่มจากคำสั่ง if ตามด้วย else if และ else จบคำสั่ง if ที่ end if

2.13.2 คำสั่ง case

การใช้คำสั่ง case ต้องใช้ให้ครบของความเป็นไปได้ของ expresstion และต้องเรียงจากน้อยสุดไปหามากสุด

```
case expression is
    when VALUE_2 | VALUE_4 =>
            sequential statements
    when VALUE_M to VALUE_N =>
            sequential statements
    when others =>
            sequential statements
end case;
```

รูปที่ 2.26 รูปแบบของคำสั่ง Case

```
entity CASE_EXANPLE is
port (A, B, X : in   integer range 0 to 15;
      Z       : out integer range 0 to 15;
end CASE_EXAMPLE;
Architecture A of CASE_EXAMPLE is
begin
   process (A, B, X)
   begin
          case X is
            when 0 to 4 =>
              Z <= B;
            when 7 | 9 =>
              Z <= A;
            when others =>
              Z <= 0;
          end case;
   end process
end A;
```

รูปที่ 2.27 ตัวอย่างการใช้คำสั่ง Case

จากรูปแบบคำสั่ง Case ในรูปที่ 2.26 นำมาเขียนตัวอย่างได้ดังรูปที่ 2.27 case จะกำหนดค่าเป็น x ให้ตัวแปร B เป็นค่า 0-4 ให้ตัวแปร C เป็นค่า 5 ตัวแปร A เป็นค่า 7,9 และถ้าเป็นค่าอื่น Z จะมีค่าเป็น 0 จบด้วย end case;

2.13.3 คำสั่ง For

```
for loop_parameter in discrete_range loop
    -- sequential_statements
end loop;
```

รูปที่ 2.28 รูปแบบของคำสั่ง For

```
entity FOR_LOOP is
port (A : in integer range 0 to 3;
      Z : out std_ulogic_vector (3 downto 0));
end FOR_LOOP;
architecture A of FOR_LOOP is
begin
   process (A)
   begin
      Z <= "0000";
      for I in 0 to 3 loop
         if (A = I) then
            Z(I) <= '1';
         end if;
      end loop;
   end process;
end A;
```

รูปที่ 2.29 ตัวอย่างการใช้คำสั่ง For

จากรูปแบบของคำสั่ง For ดังรูปที่ 2.28 นำมาเขียนตัวอย่างได้ดังรูปที่ 2.29 คำสั่ง for จะถูกกำหนดให้วนการทำงานใน loop 3 ครั้งและจบด้วย end loop ซึ่งคำสั่ง for loop นี้ก็อยู่ในกระบวนการของ process

2.14 Logic cell array family

2.14.1 XC3000

1. คุณลักษณะ (Features)

1.1 ประสิทธิภาพสูงใช้งานได้ที่ความถี่ 70-, 100-125 MHz ขึ้นไป

1.2 เป็น Geration ที่ 2 ของ FPGAs

1.2.1 I/O function

1.2.2 Digital logic function

1.2.3 Interconnection

1.3 มีสถาปัตยกรรมภายในที่ยืดหยุ่น

1.3.1 มีจำนวน Gate ภายในจำนวน 2,000 ถึง 9,000 Gate

1.3.2. เพิ่มความสามารถพิเศษของ Register และ I/O

1.3.3. มีค่า fan-out สูง

1.3.4 มี 3 State bus ภายใน

1.3.5 ทำงานกับสัญญาณ TTL และ CMOS

1.3.6 มี Oscillator amplifier ในตัว

1.4 เป็นผลิตภัณฑ์มาตรฐาน

1.4.1 ใช้พลังงานต่ำ, เป็น CMOS, ใช้เทคโนโลยี Static Memory

1.4.2 ประสิทธิภาพเท่าเทียมกับการใช้ TTL SSI/MIS

1.4.3 ผ่านการทดสอบจากโรงงานแล้ว 100%

1.4.4 สามารถเลือก Configuration Mode ได้

1.4.5 มี Development software ช่วยในการพัฒนา(XACTdevelopment software) สมบูรณ์แบบ

1.4.6 สามารถ capture ผังวงจรได้

1.4.7 มี Automatic Place/Route

1.4.8 ทำการจำลองวงจร และหาฐานเวลาได้

1.4.9 มีตัวที่ใช้แก้ไขในการออกแบบได้

1.4.10 มี library และ user macros

1.4.11 ทำการคำนวณหาค่าของฐานเวลาได้

1.4.12 มี XACTOR In-Circuit Vertfier

1.4.13 มีมาตรฐานของแฟ้มข้อมูลในตัว PROM

2. รายละเอียด

LCA ตระกูล XC3000 Logic cell array (LCA) family ประกอบไปด้วยวงจรลอจิกที่มีความหนาแน่นและประสิทธิภาพสูง ความยืดหยุ่น และการโปรแกรมได้ user array architecture เกิดขึ้นมาจาก configuration program ที่อยู่ภายในและส่วนประกอบที่สามารถปรับเปลี่ยนเพื่อให้ได้วงจรตามต้องการได้ 3 แบบ คือ IOB, array of CLB, เชื่อมต่อภายในตามความต้องการของผู้ใช้ logic function และการเชื่อมต่อภายในถูกกำหนดด้วยโปรแกรมที่อยู่ในหน่วยความจำภายในตัว LCA โปรแกรมนี้ Load เข้าสู่ Memory ได้ในหลายๆ รูปแบบตามความเหมาะสมที่ใช้งานโปรแกรมถูกบรรจุใน EPROM หรือ ROM ภายนอกหรืออาจอยู่บน floppy disk, Hardisk ก็ได้ภานใน Chip มีส่วนพิเศษที่ช่วยในการ load โปรแกรมแบบอัตโนมัติเมื่อจ่ายไฟเข้าระบบ (power-up) LCA ตระกูล XC3000 นี้ มีหลายแบบให้เลือกตามความเหมาะสมตามขนาด, อุณหภูมิ, รูปแบบของตัวถัง, อุณหภูมิใช้งาน เป็นต้น ดังตารางที่ 2.2

ตารางที่ 2.2 เปรียบเทียบขนาดของ XC3000 LCA family

Basic Array	Logic capacity (gates)	Configurable Logic Blocks	Max User I/Os	No. of PADs	Program Data (Bits)
XC3020	2000	64	64	74	14,779
XC3020	3000	100	80	98	22,176
XC3042	4200	144	96	118	30,784
XC3064	6400	224	120	140	46,064
XC3090	9000	320	144	166	64,160

2.14.2 XC4000

1. คุณลักษณะ

1.1. เป็น Generation ที่ 3 ของ FPGAs

1.1.1 มี Flip-flop เป็นจำนวนมาก

1.1.2 ในการผลิตฟังก์ชันของการทำงานมีความยืดหยุ่นสูง

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 Hierachy

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 สามารถโปรแกรมต่ำ 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 Hewlatt-Packard 700 series

1.6.1 สามารถติดต่อกับโปรแกรมอื่น ๆ ได้ เช่น VIEWlogic, Mentor Graphics และ OrCAD

1.6.2 มี automatic Place and Rounting ที่ครบ

1.6.3 มี Interactive Design Editor ที่ใช้สำหรับการทำ optimization

1.6.4 มี 288 macros, 34 hard macros, RAM/ROM compiler

2. รายละเอียด

LCA ตระกูล XC4000 ประกอบไปด้วยวงจร logic ที่มีความหนาแน่นสูง, มีความยืดหยุ่น, การโปรแกรมสถาปัตยกรรมของ CLBs ที่ต่อกันภายในมีประสิทธิภาพมากในวิธีการแบบ Hierachy, และการต่อ IOBs ที่อยู่รอบ ๆ ก็ใช้วิธีการ perimeter

2.15 โครงสร้างภายในของอุปกรณ์ FPGAs

FPGAs จัดเป็นวงจรรวมเฉพาะกิจชนิดหนึ่งที่สามารถโปรแกรมเป็นวงจรเชิงเลขใด ๆ ก็ได้ เช่นเดียวกับ EPLD ต่างกันที่ EPLD โปรแกรมลงบน EPROM ภายใน และสามารถโปรแกรมใหม่ได้หลังจากนำไปลบด้วยแสง UV แต่ FPGAs โปรแกรมลงบนสแตติกแรมภายในด้วยข้อมูลที่อยู่ภายนอก และสามารถโปรแกรมใหม่ได้โดยการรีเซตด้วยสัญญาณไฟฟ้า นอกจากนั้น FPGAs ยังประหยัดไฟและมีความจุวงจรสูง (จำนวนเกตมาก) ได้อีกด้วย

วงจรรวมชนิดที่ใช้ในโครงงานนี้ผลิตโดยบริษัทไซลิงค์ (Xilinx) ซึ่งเป็นบริษัทที่ทำการค้นคว้าร่วมกับบริษัทเอ็มเอ็มไอ (MMI) สร้างเป็นกลุ่มของเกทจำนวน 600-25,000 เกท ดังแสดงในตารางที่ 2.3 การที่ต้องการบอกขนาดของวงจรรวมเป็นจำนวนเกทเพราะจะได้รู้ว่าขนาดของวงจรที่ได้ออกแบบไว้สามารถโปรแกรมลงบนวงจรรวม FPGAs ได้หรือไม่

FPGAs มีโครงสร้างภายในใกล้เคียงกับสถาปัตยกรรมของเกทอาเรย์ (GAL,Gate Array Logic) มาก สามารถโปรแกรมและลบคอนฟิกกูเรชัน (Configuration) ภายในสแตติก-แรม (Static RAM) ได้ โดยใช้กระแสไฟฟ้าซึ่งทำการโปรแกรมได้โดยดึงข้อมูลฐานสิบหกมาจากภายนอก เช่น Parallel EPROM หรือ Serial PROM ต่างกับ EPLD,PAL ที่มี EPROM อยู่ในตัว ภายใน FPGAs จะจัดเรียงเป็นลอจิกเซลล์ล้อมรอบภายนอกด้วยอินพุทเอาท์พุทเซล FPGAs ตัวแรกที่ผลิตโดยบริษัทไซลิงค์คือเบอร์ XC2064 (2000 Family) ประกอบด้วยเซลเรียงกันเป็นเมตริกซ์ (Matrix) เป็นจำนวน 64 เซล หลังจากนั้นผลิต FPGAs ตระกูล 3000 และ 4000 ซึ่งมีโครงสร้างซับซ้อนขึ้นสามารถเพิ่มจำนวนเกทได้มากขึ้นและดีขึ้น แต่ละเซลเรียกว่า CLB (Configurable Logic Block)

ตารางที่ 2.3 คุณสมบัติของ FPGAs ตระกูลต่างๆ

FPGAs	Appr.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
XC4003/4003A	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

NAIINR2 หมายถึงเกท NAND2 หรือ เกท NOR2

ตารางที่ 2.4 ตารางประมาณการนับเกทของเกทพื้นฐาน

Gate	Equipvalent gate count	Gate	Equipvalent 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

2.15.1 ส่วนที่เป็นองค์ประกอบของลอจิก (Configurable Logic Block)

CLB จะจัดเรียงกันเป็นแบบเมตริกซ์แบบอาเรย์ขนาด M*N การออกแบบนั้นสามารถทำได้โดยการจัดวาง CLB และเชื่อมต่อขาของ CLB ให้ต่อถึงกัน เราสามารถจัด CLB ให้ต่อถึงกันได้โดยการทำด้วยมือหรือให้โปรแกรมที่สนับสนุน FPGAs ทำให้โดยอัตโนมัติโดยวิธีของมันเอง สำหรับไฟล์ที่ได้จากโปรแกรมเหล่านี้ เราเรียกว่า ไฟล์ที่กำหนดการวางอุปกรณ์ (configuration File) ซึ่งจะบรรจุโครงร่างภายในของ CLB ตามความเหมาะสม อีกด้านหนึ่งไฟล์กำหนดการวางอุปกรณ์นั้นจะเป็นไฟล์กระแสข้อมูล (Bit Stream) ซึ่งสามารถใช้โปรแกรมหน่วยความจำภายในของ FPGAs ได้

2.15.2 ส่วนอินพุทและเอาท์พุทของอุปกรณ์ FPGAs

รอบนอกของ FPGAs จะประกอบด้วย IOBs ประมาณ 64 ถึง 144 ตัว ซึ่งขึ้นอยู่กับตระกูลของ FPGAs ซึ่ง IOBs จะเป็นตัวเชื่อมต่อระหว่างภายในกับภายนอกของวงจรลอจิก

ของ FPGAs ลักษณะของ IOBs จะมีลักษณะ 2 ทิศทาง สามารถโปรแกรมให้เป็นอินพุทหรือเอาท์พุทก็ได้

2.16 รายละเอียดการใช้งานของอุปกรณ์ FPGAs

FPGAs สามารถทำงานได้หลายลักษณะ โดยกำหนดได้ที่ขาสัญญาณ M0, M1, และ M2 ดังแสดงในตารางที่ 2.4 ในลักษณะมาสเตอร์พาราเรล (Master Parallel Mode) รับโปรแกรมคอนฟิกทีละ 1 ไบท์ (Byte) จากหน่วยความจำภายนอกที่เป็นแบบขนานโดยสามารถรับโปรแกรมคอนฟิก (Config) จากแอดเดรส (Address) ต่ำหรือสูงก่อนก็ได้การต่อลักษณะเพอริเฟอรัล (Peripheral) จะรับโปรแกรมคอนฟิกทีละ 1 ไบท์จากไมโครโปรเซสเซอร์โดยสามารถโต้ตอบกันได้ว่าพร้อมหรือไม่ที่จะรับข้อมูลต่อไป การต่อลักษณะสเลฟซีเรียล (Slave Serial) จะรับโปรแกรมคอนฟิกทีละ 1 บิท (Bit) จากไมโครโปรเซส-เซอร์ตามสัญญาณอินพุท CCLK ส่วนการต่อลักษณะมาสเตอร์ซีเรียล (Master Serial) จะรับโปรแกรมคอนฟิกทีละ 1 บิทจากหน่วยความจำภายนอกที่เป็นแบบอนุกรม

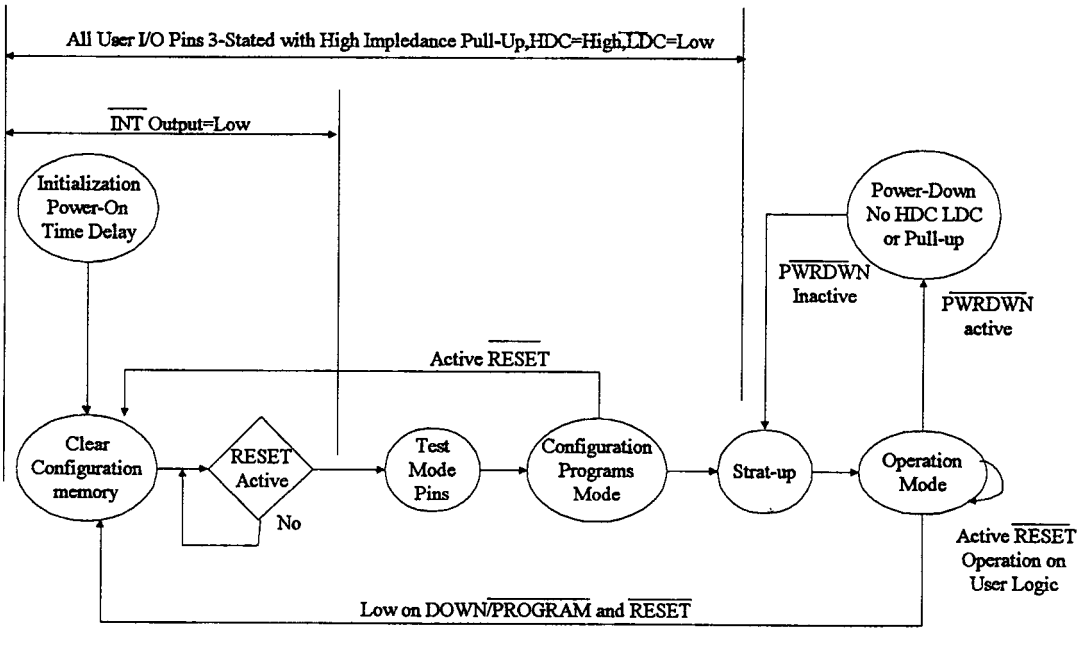
ตารางที่ 2.5 โหมดต่างๆ ของการคอนฟิกกูเรชัน

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,00000 up
Master Parallel down	1	1	0	output	Byte-Wide,3FFFF 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 แบบจึงมีเพียง 2 แบบเท่านั้นที่เหมาะสม คือ แบบมาสเตอร์ซีเรียลและแบบสเลฟซีเรียล ส่วนแบบมาส-เตอร์พาราแรลต้องใช้ EPROM 27CXXX ซึ่งกินกระแสมากกว่า PROM XC17XXX เหมาะใน

การทดสอบต้นแบบก่อน เมื่อวงจรต้นแบบทำงานได้ถูกต้องแล้วจึงทำการอัดโปรแกรมลง PROM อีกทีหนึ่งเพราะว่าในแบบพาราแรลนั้น EPROM สามารถโปรแกรมได้ใหม่ต่างกับ PROM ที่โปรแกรมได้เพียงครั้งเดียว

การใช้งาน FPGAs ในการต่อลักษณะสเลฟซีเรียลและมาสเตอร์ซีเรียล เมื่อเริ่มจ่ายไฟเข้าตัว FPGAs จะทำการลบข้อมูลหน่วยความจำที่ใช้ในคอนฟิก (Configuration Memory) ตรวจสอบลักษณะการต่อคอนฟิกว่าเป็นลักษณะใดในตารางที่ 2.5 ว่าเป็นแบบอนุกรมหรือขนาน หลังจากนั้นจะเริ่มทำการโปรแกรมคอนฟิกสัญญาณ DONE/ PROGRAM เป็น "0" ซึ่งอยู่ในระหว่างโปรแกรม และเมื่อข้อมูลในคอนฟิก ที่รับมาจากภายนอกเต็มหน่วยความจำที่ใช้ในคอนฟิก และความยาวของข้อมูลตรงกับที่ส่วนหัวของข้อมูลคอนฟิก สัญญาณ DONE/PROGRAMจะเป็น "1" ซึ่งหมายถึงโปรแกรมทำการคอนฟิกเสร็จสิ้น ดูรูปที่ 2.30 ประกอบ

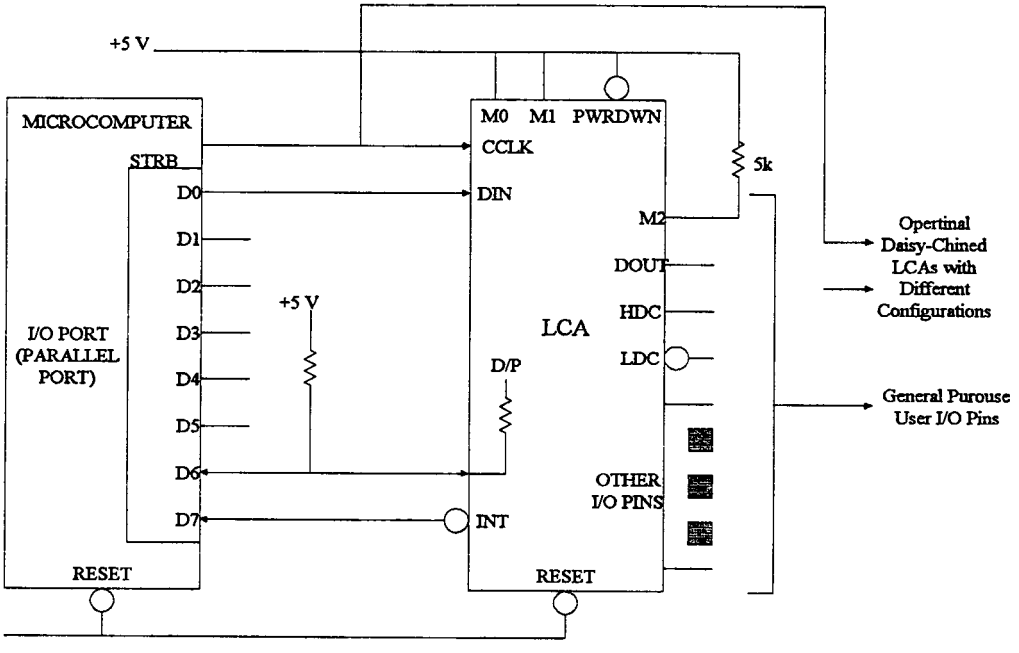


รูปที่ 2.30 ลำดับไดอะแกรมในการคอนฟิก

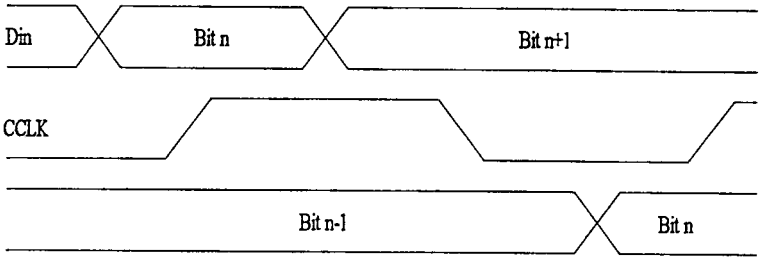
เมื่อเริ่มป้อนแหล่งจ่ายไฟเข้าไอซีและการโปรแกรมใหม่

2.16.1 การใช้งานในลักษณะสเลฟซีเรียล

การต่อใช้งานในลักษณะนี้ เหมาะสมกับวงจรที่ออกแบบมาเพื่อทำงานร่วมกับไมโครคอมพิวเตอร์อยู่แล้ว ทั้งนี้เพราะ FPGAs ได้ใช้ความสามารถของไมโครโปรเซสเซอร์ในการเก็บและส่งข้อมูลคอนฟิกให้เพียงแต่ต้องเขียนโปรแกรม เพื่อส่งโปรแกรมคอนฟิกให้เพิ่ม ลักษณะการต่อในลักษณะนี้เป็นดังรูปที่ 2.31 ซึ่งไมโครคอมพิวเตอร์จะสร้างสัญญาณเพื่อทำการคอนฟิกให้กับอุปกรณ์ FPGAs การป้อนโปรแกรมคอนฟิกให้ FPGAs ทำได้โดยต่อสัญญาณ Strobe เข้ากับขา CCLK และพอร์ต D0 เข้ากับขา DIN สร้างสัญญาณคล็อกป้อนที่ขา CCLK และป้อนโปรแกมคอนฟิกแบบอนุกรมเข้าที่ขา DIN ดังแผนภูมิในรูปที่ 2.32



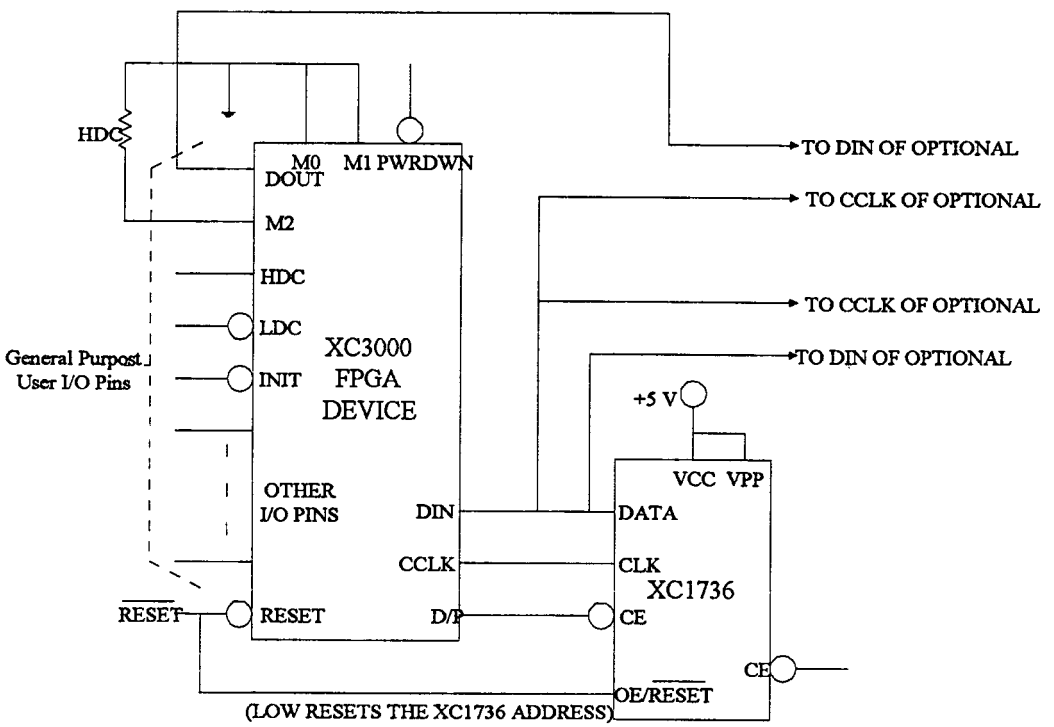
รูปที่ 2.31 การต่อใช้งานในลักษณะสเลฟซีเรียล



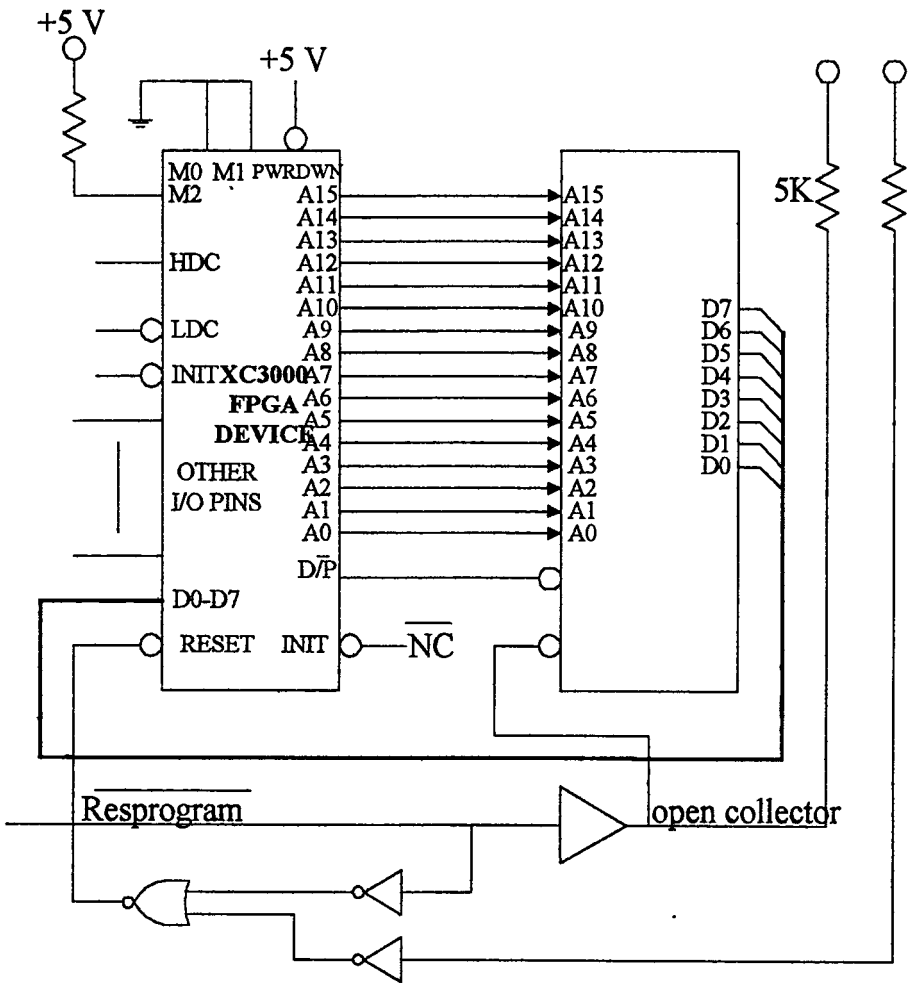
รูปที่ 2.32 แผนภูมิเวลาการป้อนข้อมูลโปรแกรมคอนฟิกในลักษณะสเลฟซีเรียล

2.16.2 การใช้งานในลักษณะมาสเตอร์ซีเรียล

การต่อใช้งานในลักษณะนี้ส่วนที่เก็บโปรแกรมคอนฟิกจะต่างจากการต่อในลักษณะแรกคือ ใช้ PROM เบอร์ XC17XXX เป็นตัวเก็บโปรแกรม ทำให้ไม่ต้องเสียเวลาเขียนโปรแกรมเพื่อทำการคอนฟิก ซึ่งวิธีการอัดโปรแกรมคอนฟิกลง PROM ทำตามขั้นตอนดังนี้คือ เมคบิท (Make Bits) สร้างไฟล์ .BIT จากวงจรที่ออกแบบ และใช้โปรแกรม MakePROM สร้าง Hex ไฟล์แล้วทำการอัดโปรแกรมลง PROM ด้วยอุปกรณ์อัด PROM ที่มาพร้อมกับตัวโปรแกรมของไซลิงค์ PROM XC17XXX จะส่งสัญญาณเพื่อทำการคอนฟิกให้กับอุปกรณ์ FPGAs ดังแสดงในรูปที่ 2.33 D0-D7 เป็นขารับข้อมูลที่ใช้ในการคอนฟิกแบบขนาน A0-A15 เป็นแอดเดรสที่ FPGAs สร้างให้กับ EPROM เพื่ออ่านข้อมูลจากหน่วยความจำมาเก็บไว้ในสแตติกแรม (Static RAM) แอดเดรสทั้ง 16 เส้น ไม่จำเป็นต้องต่อให้ครบก็ได้ขึ้นอยู่กับขนาดของหน่วยความจำ EPROMที่ใช้ และสามารถกำหนดให้นับขึ้นหรือลงได้



รูปที่ 2.33 การต่อใช้งานในลักษณะมาสเตอร์ซีเรียล



รูปที่ 2.34 การต่อใช้งานในลักษณะมาสเตอร์พาราแรล

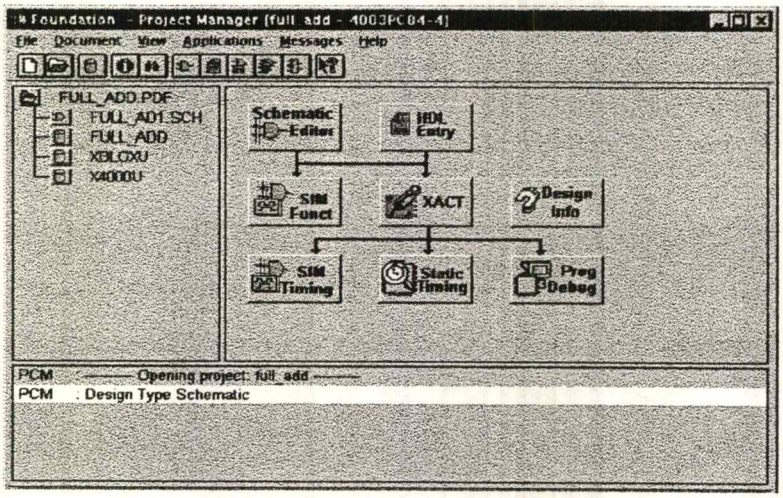
2.17 ข้อควรระวังในการใช้อุปกรณ์ FPGAs

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

การป้องกันไอซีจากแรงดันไฟฟ้าไม่ควรต่อสลับขั้วบวกกับลบจะทำให้ไอซีเสียได้ นอกจากนั้นแรงดันของแหล่งไฟต้องอยู่ในช่วงที่โรงงานกำหนดมา สำหรับ FPGAs ค่าที่ใช้งานอยู่ในช่วง Vcc = 4.75-5.25 V และแรงดันที่ทนได้อยู่ในช่วง -0.5 - 7 V ดังนั้นก่อนป้อนแรงดันควรเช็คให้แน่ใจก่อน

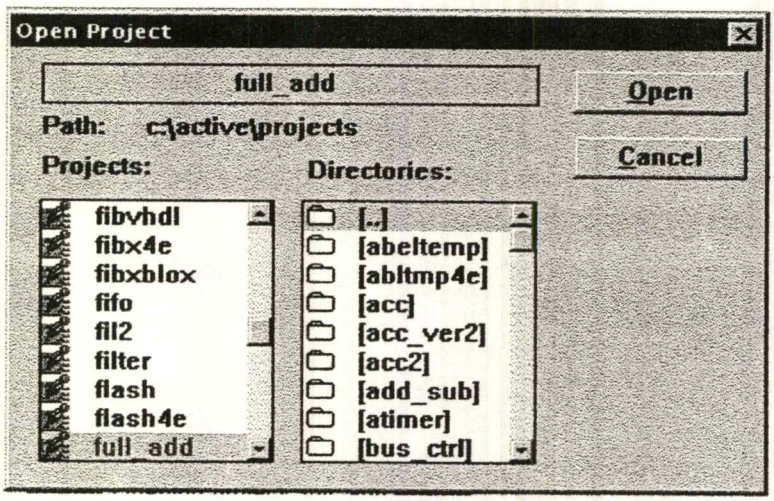
2.18 ขั้นตอนการออกแบบและสังเคราะห์โดยใช้ซอฟแวร์ของบริษัทไซลิงค์

ซอฟแวร์ของบริษัทไซลิงค์ (Xilinx) เวอร์ชัน 6.01 เป็นเวอร์ชันที่ทำงานได้ทั้งบน Windows 3.11 และ Windows 95 เมื่อเริ่มต้นใช้โปรแกรมที่หน้าจอจะปรากฏดังรูปที่ 2.35



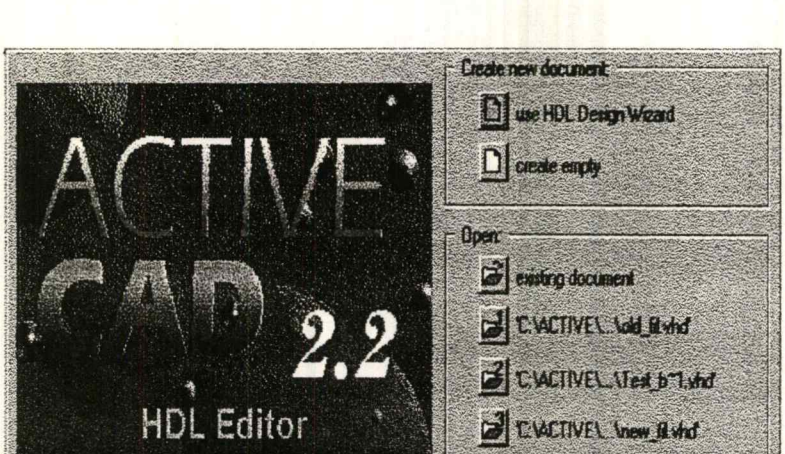
รูปที่ 2.35 หน้าต่าง Project Manager

คลิกเมาส์ที่เมนู File แล้วเลือก New Project เพื่อสร้าง Project ใหม่ขึ้นมาหรือเลือกที่ Open Project เพื่อเรียกใช้โปรเจ็ค Full_add ที่ได้เขียนไว้แล้ว จะแสดงหน้าต่างดังรูปที่ 2.36



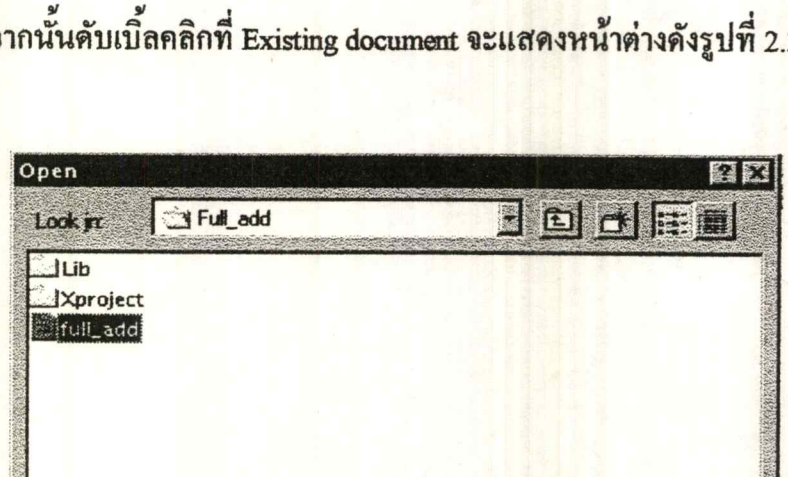
รูปที่ 2.36 หน้าต่าง Open Project

ดับเบิ้ลคลิกที่โปรเจ็ค Full_add หลังจากนั้นคลิกเมาส์ที่ HDL Entity เพื่อที่จะนำไฟล์ Full_add.vhd ที่ได้เขียนไว้แล้วมาทำการสังเคราะห์ จะแสดงหน้าต่างดังรูปที่ 2.37



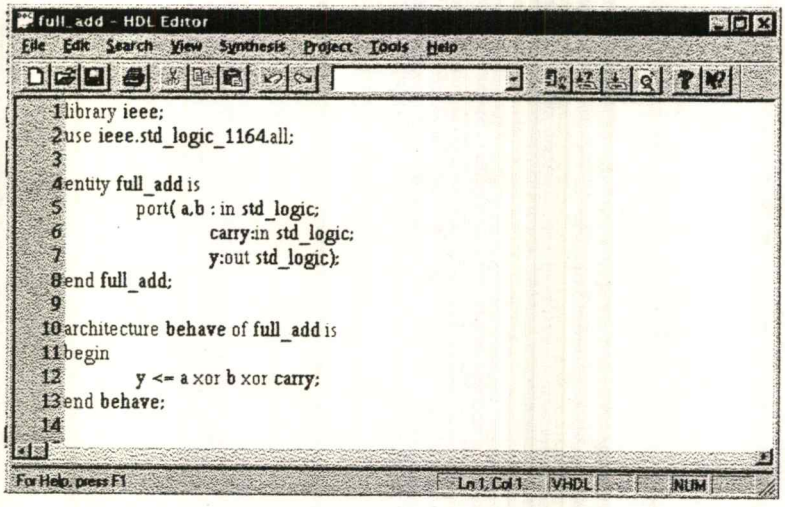
รูปที่ 2.37 หน้าต่าง ACTIVE CAD 2.2 HDL Editor

หลังจากนั้นดับเบิ้ลคลิกที่ Existing document จะแสดงหน้าต่างดังรูปที่ 2.38



รูปที่ 2.38 หน้าต่าง Open

จากนั้นดับเบิ้ลคลิกที่ไฟล์ Full_add จะปรากฏโปรแกรมที่เขียนไว้ดังรูปที่ 2.39



รูปที่ 2.39 หน้าต่าง full_add-HDL Editor

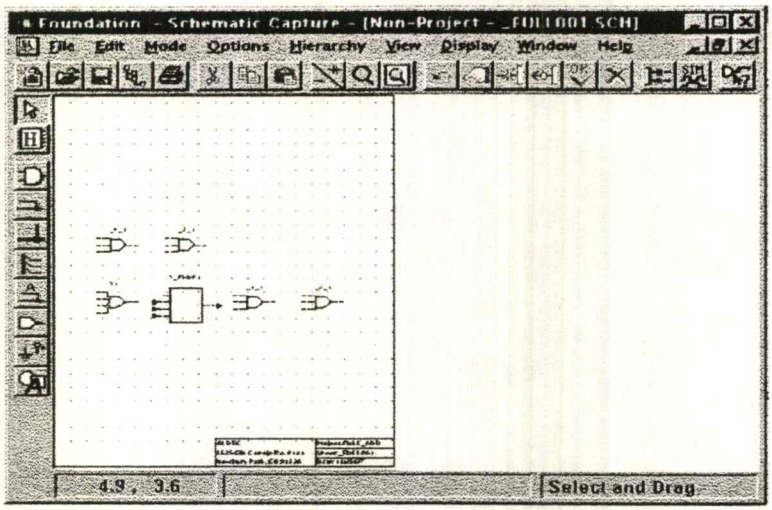
จากนั้นทำการสังเคราะห์ (Synthesis) โปรแกรม โดยคลิกเมาส์ที่เมนู Systhesis และเลือกที่ Systhesize ถ้าหากการ Systhesis ไม่มีข้อผิดพลาดจะปรากฏหน้าต่างดังรูปที่ 2.40

รูปที่ 2.40 หน้าต่าง HDL Editer

ถ้าหากเกิดการผิดพลาดให้กลับไปแก้ไขที่โปรแกรม

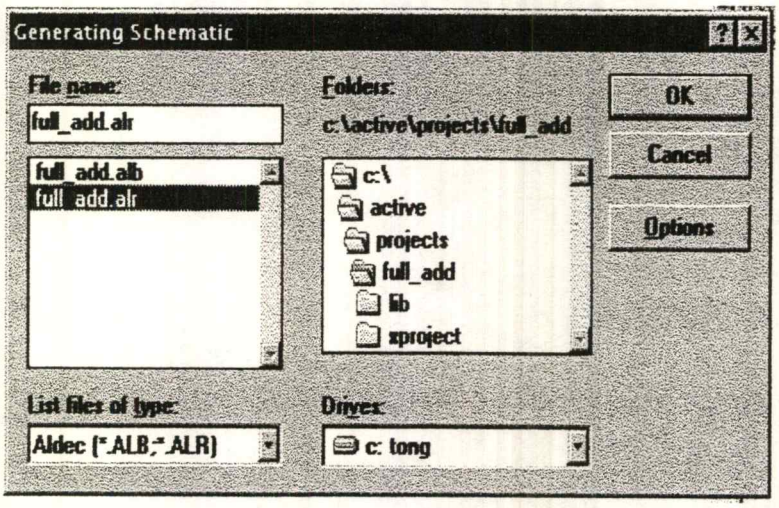
2.19 ขั้นตอนการจำลองการทำงานโดยใช้ซอฟแวร์ของบริษัทไซลิงค์ (Xilinx)

กลับไปที่หน้าต่าง Project Manager แล้วคลิกเมาส์ที่ Schematic Editor จะแสดงหน้าต่างของ Schematic ดังรูปที่ 2.41



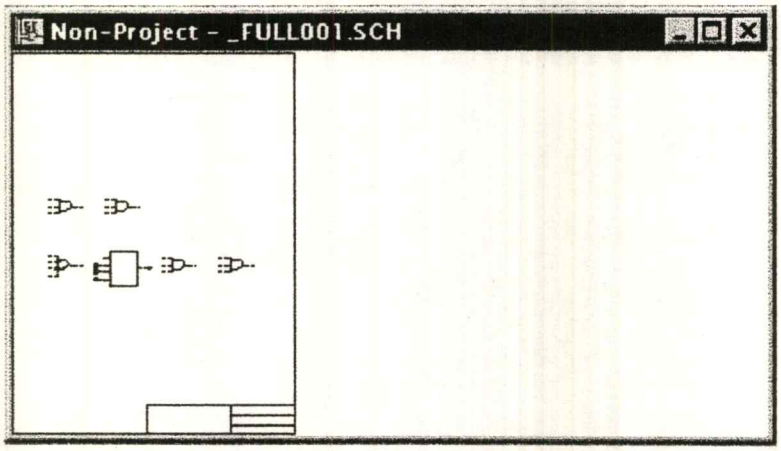
รูปที่ 2.41 หน้าต่าง Schematic Editor

จากนั้นดับเบิ้ลคลิกที่ full_add.xnf เพื่อโหลดไฟล์ full_add.xnf แล้วคลิกเมาส์ที่เมนู File แล้วเลือก Generate Schmatic From Netlist เพื่อทำการสร้างวงจรจาก Netlist ที่ได้และจะแสดงหน้าต่างดังรูปที่ 2.42



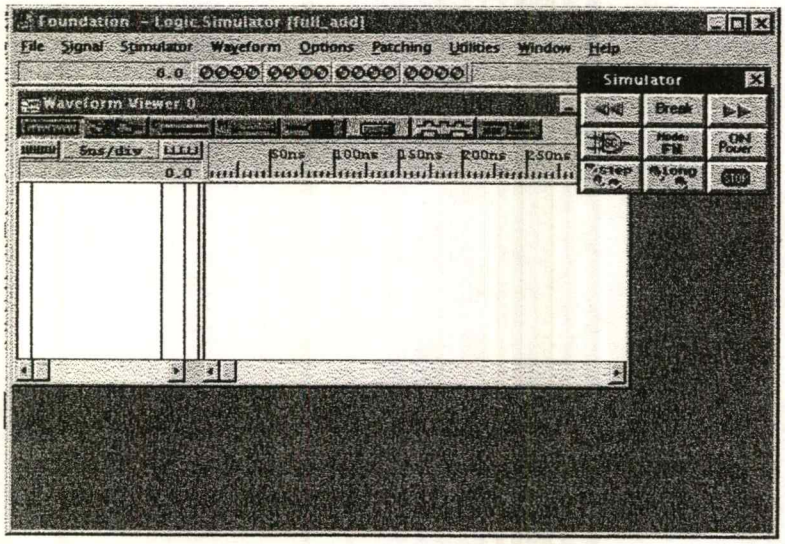
รูปที่ 2.42 หน้าต่าง Generating Schematic

ดับเบิ้ลคลิกที่ไฟล์ Full_add.air เพื่อทำการสร้างวงจรจาก Netlist ที่ได้ หลังจากนั้นทำการโหลดไฟล์ _full001.sch จะแสดงหน้าต่างดังรูปที่ 2.43



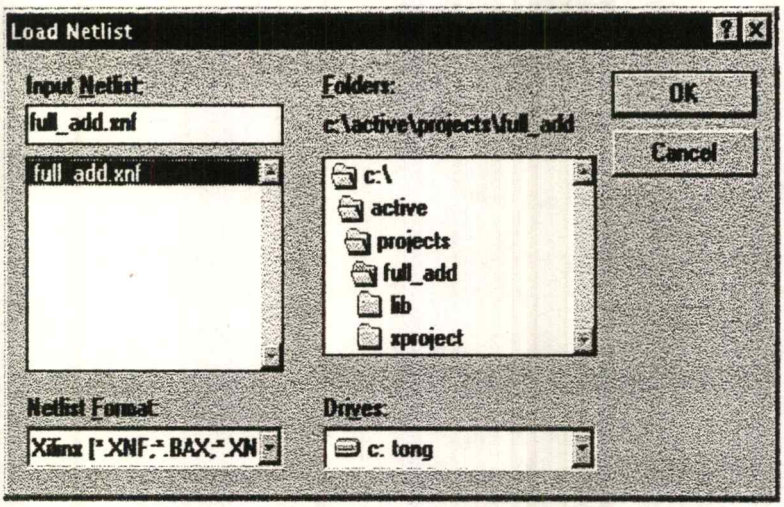
รูปที่ 2.43 หน้าต่าง Non-Project-_Full_add.air

จากนั้นคลิกเมาส์ที่ Simulate เพื่อทำการ Simulate สัญญาณ ซึ่งจะแสดงหน้าต่างดังรูปที่ 2.44



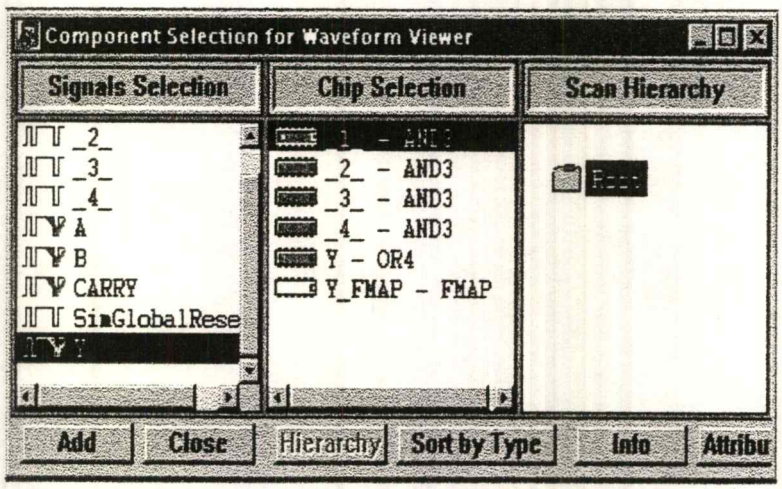
รูปที่ 2.44 หน้าต่าง Logic Simulator

จากนั้นคลิกเมาส์ไปที่เมนู File แล้วเลือกคำสั่ง Load Netlist... จะแสดงหน้าต่าง Load Netlist ดังรูปที่ 2.45



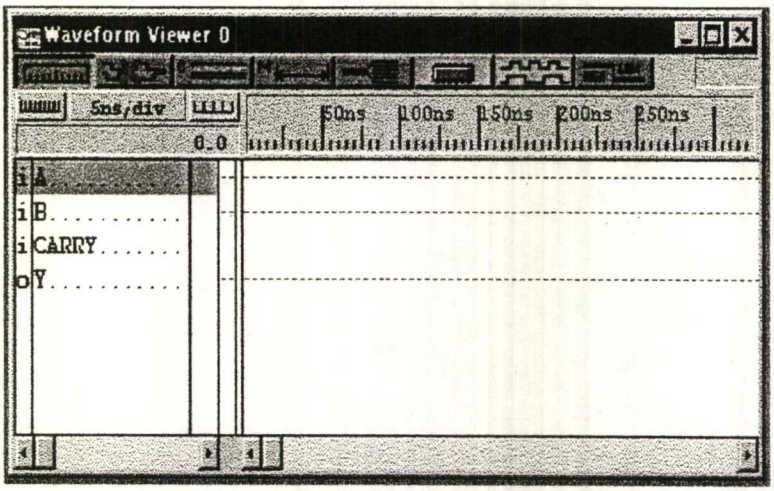
รูปที่ 2.45 หน้าต่าง Load Netlist

แล้วเลือกไฟล์ .xnf (Xilinx Netlist File) ที่ต้องการ แล้วคลิกเมาส์ที่เมนู Signal เลือก Add Signals เพื่อใช้ในการเลือกสัญญาณที่เราต้องการให้แสดงผล ซึ่งจะแสดงหน้าต่างดังรูปที่ 2.46



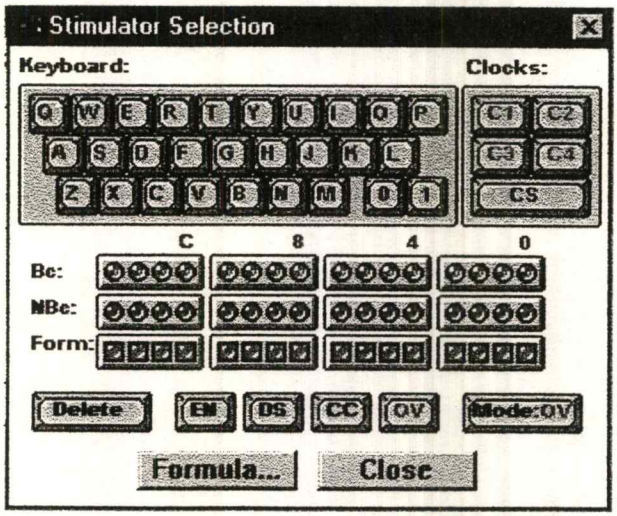
รูปที่ 2.46 หน้าต่าง Component Selectot-From Wavefrom Viewer

จากนั้นเลือกสัญญาณที่ต้องการจะนำไปแสดงผล ซึ่งจะแสดงสัญญาณที่ต้องการบนหน้าต่าง Logic Simulator ดังรูปที่ 2.47



รูปที่ 2.47 หน้าต่าง Logic Simulator

ทำการกำหนดสัญญาณโดยคลิกเมาส์ที่เมนู Simulator แล้วเลือกที่ Add Simulator... จะแสดงหน้าต่างดังรูปที่ 2.48



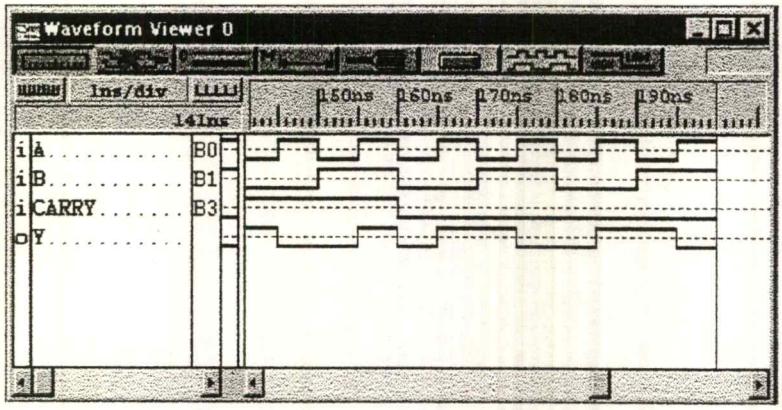
รูปที่ 2.48 หน้าต่าง Simulator Selection

โดยเลือกให้สัญญาณ A = B0

สัญญาณ B = B1

สัญญาณ Carry = B4

จากนั้นทำการ Run โดยคลิกเมาส์ที่ปุ่ม Step จะแสดงหน้าต่างดังรูปที่ 2.49



รูปที่ 2.49 ผลการ Run

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

2.20 ขั้นตอนการโปรแกรมลงบนอุปกรณ์ FPGAs โดยใช้ซอฟแวร์ XDM ของบริษัทไซลิงค์ (Xilinx)

เมื่อเริ่มเข้าสู่โปรแกรม XACT Design Manager (XDM) ที่หน้าจอจะแสดงรูปภาพดังรูปที่ 2.51 หลังจากนั้นจะเข้าสู่หน้าต่าง Design Manager ดังรูปที่ 2.52 ให้คลิกเมาส์ที่เมนู File และเลือกที่ New Project โดยจะแสดงหน้าต่างดังรูปที่ 2.53

คลิกเมาส์ที่ Browse บริเวณช่อง Input Design โดยให้ไปที่ไดเร็กทอรีที่มีไฟล์ Fulladd.1 ซึ่งเป็นวงจรระดับเกทของโปรแกรม Fulladd.vhd

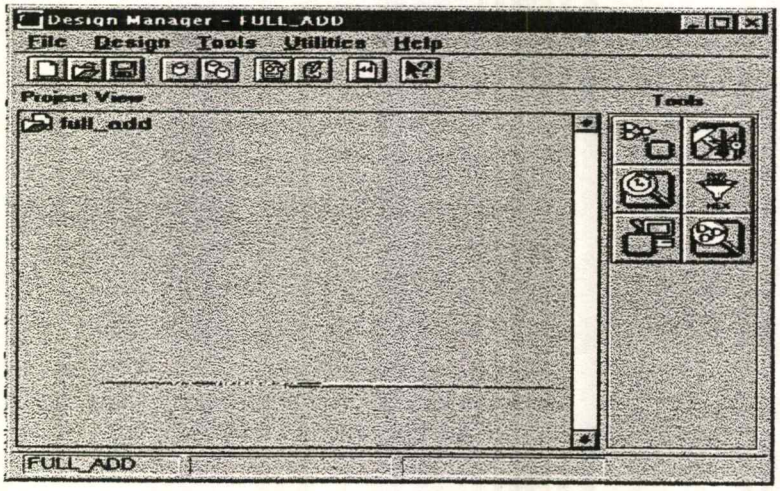
เมื่อได้ผลการ Run ออกมาแล้วให้ทำการตรวจสอบผลการ Run โปรแกรมว่าถูกต้องหรือไม่ ถ้าหากผลการทดลองไม่ถูกต้อง ต้องกลับไปแก้ไขที่ Source Progran ใหม่ แล้วทำการสังเคราะห์โปรแกรมใหม่ แล้วจึงเริ่มขั้นตอนการจำลองการทำงานอีกครั้ง

2.20 ขั้นตอนการโปรแกรมลงบนอุปกรณ์ FPGAs โดยใช้ซอฟแวร์ XDM ของบริษัทไซลิงค์

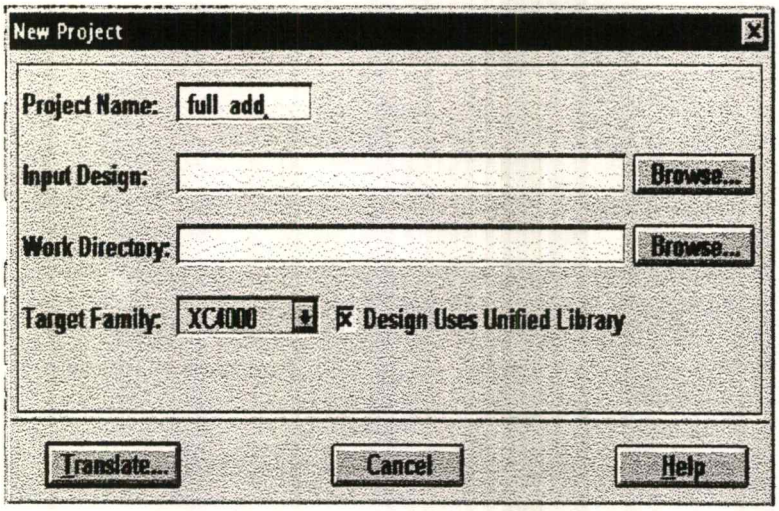
เมื่อเริ่มเข้าสู่โปรแกรม XACT Design Manager (XDM) ที่หน้าจอจะแสดงรูปภาพดังรูปที่ 2.51 หลังจากนั้นจะเข้าสู่หน้าต่าง Design Manager ดังรูปที่ 2.52 ให้คลิกเมาส์ที่เมนู File และเลือกที่ New Project โดยจะแสดงหน้าต่างดังรูปที่ 2.53

คลิกเมาส์ที่ Browse บริเวณช่อง Input Design โดยให้ไปที่ไดเร็กทอรีที่มีไฟล์ Fulladd.1 ซึ่งเป็นวงจรระดับเกทของโปรแกรม Fulladd.vhd

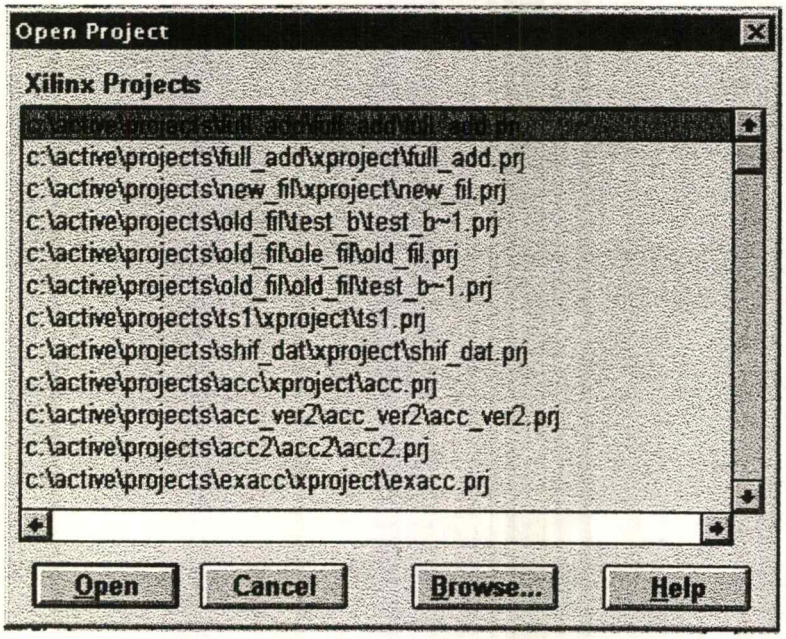
รูปที่ 2.50 หน้าจอเริ่มต้นของโปรแกรม XACT Design Manager



รูปที่ 2.51 หน้าต่าง Design Manager

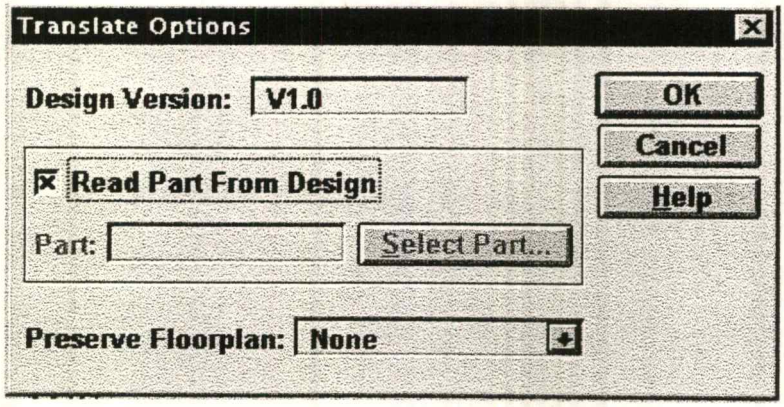


รูปที่ 2.52 หน้าต่าง New Project



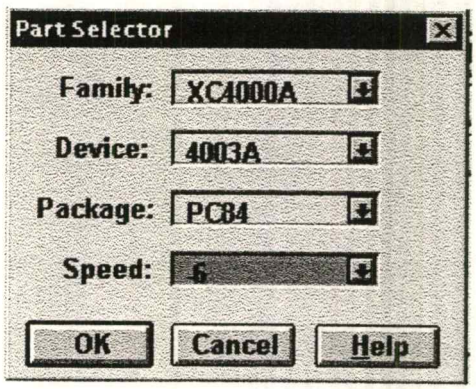
รูปที่ 2.53 หน้าต่าง Input Design

คลิกเมาส์ที่ Translate จะแสดงหน้าต่าง Translate ดังรูปที่ 2.54



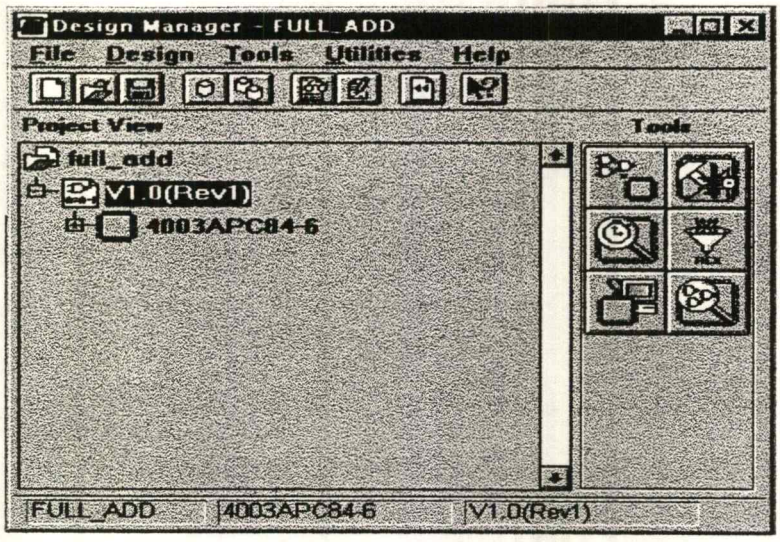
รูปที่ 2.54 หน้าต่าง Translate

คลิกเมาส์ที่ช่อง Real Part Design และที่ Select Part ซึ่งจะแสดงหน้าต่างของ Part Selector และเปลี่ยน Speed จาก 4 เป็น 6 ซึ่งเป็นเบอร์ของไอซี FPGAs เบอร์ XC4005APC84C-6 ที่ใช้ในบอร์ดตัวอย่างของ FPGAs ดังรูปที่ 2.55

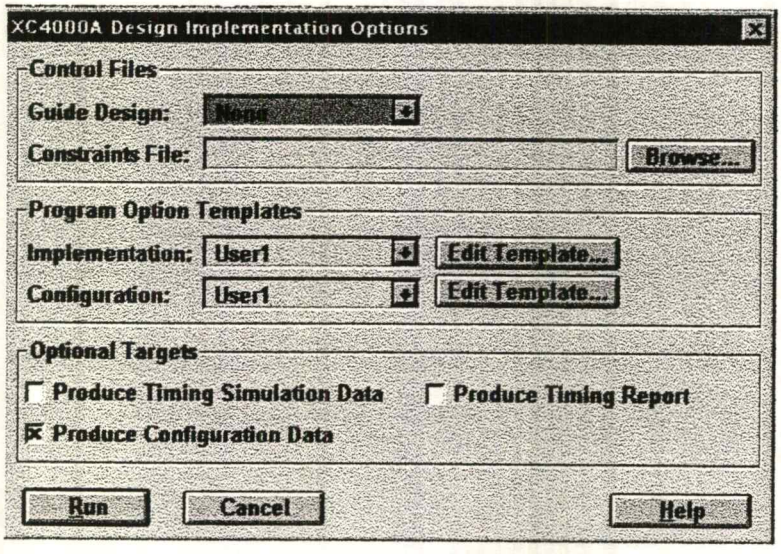


รูปที่ 2.55 หน้าต่าง Part Selector

เมื่อทำการกำหนดค่าต่างๆ โปรแกรมจะทำการ Translate ให้ เมื่อเสร็จการ Translate แล้ว จะแสดงหน้าต่างดังรูปที่ 2.56 หลังจากนั้นให้คลิกที่เมนู Design เสร็จแล้วเลือกคำสั่ง Implement จะแสดงหน้าต่าง XC 4000 Design Implementation Option ดังรูปที่ 2.57 ที่ช่อง Output Targets จะให้เลือกการรายงานผลการ Run โดยเลือกให้รายงานผลทุกอย่าง เสร็จแล้ว คลิกเมาส์ที่ Run ซึ่งจะเป็นกระบวนการ Design จากวงจรระดับเกทของไฟล์ Fulladd.1 ให้เป็นไฟล์บิทสตรีม (Bit Stream) เพื่อใช้ดาวน์โหลดลงบนอุปกรณ์ FPGAs ดังรูปที่ 2.58

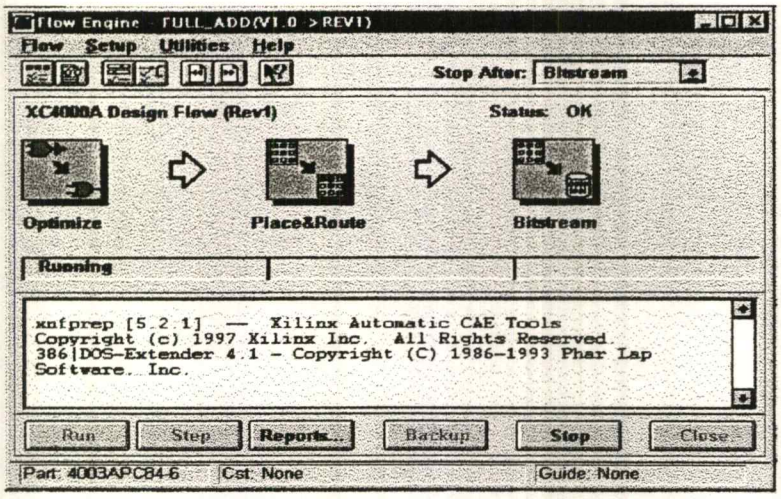


รูปที่ 2.56 ผลการ Translate

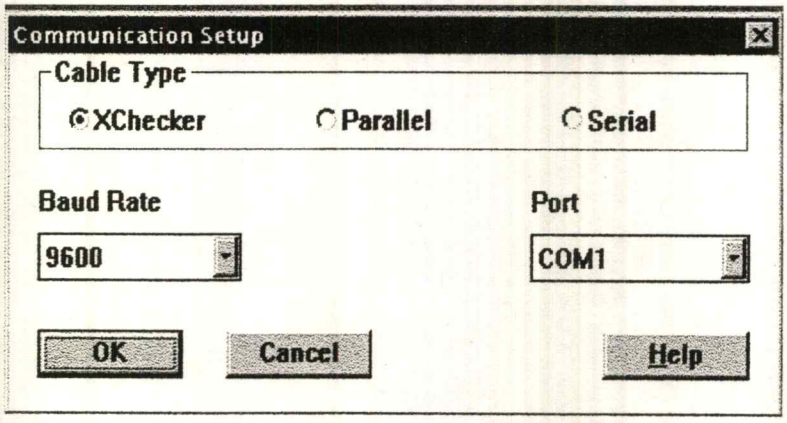


รูปที่ 2.57 หน้าต่าง XC 4000 Design Implementation Output

เมื่อเสร็จกระบวนการแล้วจะได้ไฟล์ที่มีนามสกุล .Bit เพื่อดาวน์โหลดลงบนบอร์ดตัวอย่างของ FPGAsโดยในการดาวน์โหลดไฟล์บิทสตรีมลงบนบอร์ดตัวอย่างของ FPGAs นั้นให้คลิกเมาส์ที่เมนู Tools และเลือกที่ Hardware Debugger ซึ่งจะแสดงหน้าต่าง Communication Setup ดังรูปที่ 2.59



รูปที่ 2.58 หน้าต่างกระบวนการสร้างไฟล์บิตสตรีม



รูปที่ 2.59 หน้าต่าง Communication Setup

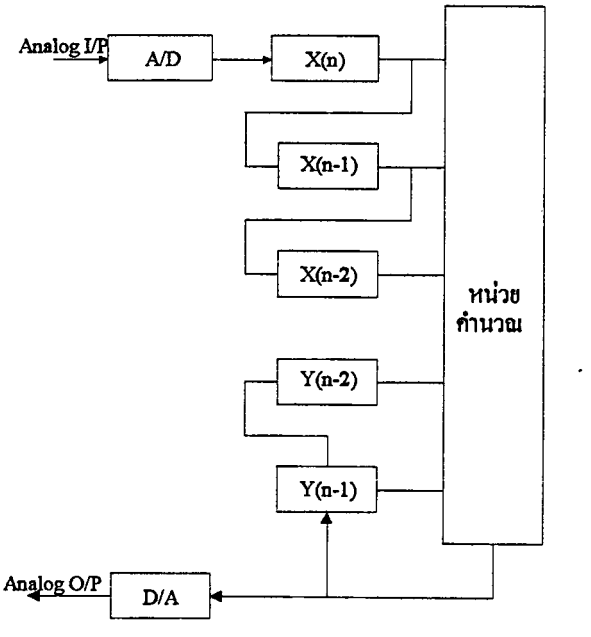
เมื่อทำการกำหนดค่า Baud Rate และเลือก Port ที่จะส่งข้อมูลออกไปแล้วให้คลิกที่ OK เมื่อเสร็จแล้วให้ไปที่หน้าต่าง Hardware Debugger คลิกที่เมนู Download และเลือกที่ Download Design ซึ่งโปรแกรม Debugger จะทำการ Download Design ไฟล์ Fulladd.Bit ลงบนบอร์ดตัวอย่างของ FPGAs ผ่านทางสายดาวน์โหลดเคเบิล เมื่อดาวน์โหลดเสร็จเรียบร้อยแล้วจึงนำลงบนบอร์ดตัวอย่างของ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

บทที่ 3

การออกแบบและการสร้าง

3.1 ความคิดเบื้องต้นในการสร้างดิจิตอลฟิลเตอร์

ดิจิตอลฟิลเตอร์สามารถสร้างได้ทั้งแบบฮาร์ดแวร์เพียงอย่างเดียวหรือสร้างจากฮาร์ดแวร์ร่วมกับซอฟแวร์ หรือสร้างจากซอฟแวร์เพียงอย่างเดียวได้ ซึ่งทั้งสามแบบจะมีโครงสร้างดังรูป 3.1



รูปที่ 3.1 บล็อกไดอะแกรมของ Second-Order Digital Filter

สัญญาณอนาลอกอินพุทจะถูกแปลงเป็นสัญญาณดิจิตอลโดย A/D (Analog to Digital Converter) สัญญาณดิจิตอลที่ได้จะถูกชิฟท์แบบขนานเข้าไปในชิฟรีจิสเตอร์ X(n) โดยค่าเริ่มต้นของ X(n), X(n-1), X(n-2), Y(n-1), Y(n-2) ควรมีค่าเป็นศูนย์และโน้มเข้าสู่ค่าจริงๆ ของมันในเวลาต่อมา X(n) แล้ว บิทขวาสุดของ X(n), X(n-1), X(n-2), Y(n-2), Y(n-2) จะเป็นอินพุทของหน่วยคำนวณ 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, $-b_1$, $-b_2$ จะถูกเก็บไว้ในหน่วยความจำในรูปของเลขฐานสองพร้อมกันดังนี้

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

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

X(n-2) ก็จะถูกชิฟท์ออกทางขวาทิ้ง 1 บิท

เป็นผลให้บิทขวาสุดของชิฟท์รีจิสเตอร์ถูกเลื่อนมาอีก 1 ตำแหน่ง ตำแหน่งใหม่ที่เลื่อนมาจะเป็นอินพุทเข้าสู่หน่วยคำนวณต่อไป ทำเช่นนี้เรื่อยไปจนครบทุกบิทของชิฟท์รีจิสเตอร์ เมื่อครบแล้วหน่วยความจำก็จะให้ค่าเอาท์พุท Y(n) ออกมา Y(n) ที่คำนวณได้จะถูกแปลงเป็นสัญญาณอนาลอกโดยวงจร 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 Distribited 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

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

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_1และ-b_2 เป็นค่าสัมประสิทธิ์ของตัวกรอง

วิธีการของโครงสร้างเลขคณิตแจกแจงทำโดยการเขียนแทนลำดับสัญญาณเข้าและลำดับสัญญาณออกด้วยตัวเลขแบบส่วนเติมเต็มสอง (2'complement) ที่มีจำนวนบิทรวมทั้งเครื่องหมายด้วยเป็น B บิท หรือเขียนกระจายเป็นเลขฐานสองคือ

$$X(n) = X_0(n)X_1(n)X_2(n)\ldots X_{b-1}(n) \quad (3.3)$$

$$Y(n) = Y_0(n)Y_1(n)Y_2(n)\ldots Y_{b-1}(n) \quad (3.4)$$

โดยที่ $X_0(n)$ และ $Y_0(n)$ เป็นบิทที่แสดงเครื่องหมายของตัวเลข ส่วน $X_1(n)$ และ $Y_1(n)$ [i=1,2,...B-1] เป็นบิทที่ i ของลำดับสัญญาณ และมีค่าเป็น 0 หรือ 1 เท่านั้น

จากสมการ (3.3) อาจเขียนรวมได้ในรูปแบบของ 2's Complement คือ

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

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

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

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

เมื่อทำการจัดพจน์ใหม่โดยเอาบิทที่สมนัยกันมาเขียนรวมกันจะได้

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

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

ผลจากสมการ (3.9) และ (3.10) จะได้โครงสร้างแบบเลขคณิตแจกแจง โดยนำค่าฟังก์ชันทั้งหมดของ $F_i\{.\}$ มาทำเป็นตารางเปิดดู (Look up table) โดยค่าในตารางเปิดดูคิดคำนวณจากสมการ (3.9) และเนื่องจาก $F_i\{.\}$ มีตัวแปรอยู่ 5 ตัว คือ $X_i(n)$, $X_i(n-1)$, $X_i(n-2)$, $Y_i(n-1)$, $Y_i(n-2)$ ทำให้ตารางเปิดดูมีค่าฟังก์ชันของ $F_i\{.\}$ อยู่ 2^5 ค่า ค่าของฟังก์ชัน $F_i\{.\}$ ทั้ง 32 ค่านี้จะถูกคำนวณแล้วทำการปัดเศษให้เหลือ B บิท แล้วเก็บไว้ในรอมหรือแรม เพื่อนำไปสร้างตัวกรองต่อไป

3.3 การสร้างตารางเปิดดูจากฟังก์ชัน $F_i\{.\}$ สำหรับตัวกรองอันดับสอง

เมื่อได้สมการผลต่างสืบเนื่องอันดับสองแล้ว นำค่าสัมประสิทธิ์คูณกับบิทที่ 1 ของข้อมูลแต่ละตัวแล้วบวกกันเก็บไว้ที่แอดเดรสที่ชี้โดยค่า $X_i(n)$, $X_i(n-1)$, $X_i(n-2)$, $Y_i(n-1)$, $Y_i(n-2)$ ดังแสดงในตารางที่ 3.1

ตารางที่ 3.1 ค่าสัมประสิทธิ์

ตำแหน่งที่	แอดเดรสของรอม					ค่าของ $F_i\{.\}$ ภายในรอม
	$X_i(n)$	$X_i(n-1)$	$X_i(n-2)$	$Y_i(n-1)$	$Y_i(n-2)$	
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	$a_1+a_2-b_2$
14	0	1	1	1	0	$a_1+a_2-b_1$
15	0	1	1	1	1	$a_1+a_2-b_1-b_2$
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
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	$a_0+a_1+a_2$

ตารางที่ 3.1 (ต่อ) ค่าสัมประสิทธิ์

ตำแหน่งที่	แอดเดรสของรอม					ค่าของ $F_i\{.\}$ ภายในรอม
	$X_i(n)$	$X_i(n-1)$	$X_i(n-2)$	$Y_i(n-1)$	$Y_i(n-2)$	
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)

$$Y(n) = \sum_{i=1}^{B-1} 2^{-i}F_i\{.\}-F_o\{.\}$$

จะเห็นว่าในการคำนวณหา Y(n) จะต้องทำการบวกค่าฟังก์ชัน $F_i\{.\}$ และพร้อมกับการเลื่อน B-1 ครั้ง (B=จำนวนบิท) แล้วทำการลบอีก 1 ครั้ง รวมแล้วทำการคำนวณหาค่าจะต้องกระทำ B ครั้ง

แต่จากการทดลองนั้นได้ทำตัวกรองอันดับหก โดยใช้ตัวกรองอันดับสอง 3 ชุดมาคาสเคด (Cascade) กัน ดังนั้นจึงมีการคำนวณเอาท์พุท 3 ขั้นตอน เป็นไปตามสมการ

$$V(n) = a_{01}X(n)+a_{11}X(n-1)+a_{21}(n-2)-b_{11}V(n-1)-b_{21}V(n-2) \quad (3.11)$$

$$W(n) = a_{02}V(n)+a_{12}V(n-1)+a_{22}V(n-2)-b_{12}W(n-1)-b_{22}W(n-2) \quad (3.12)$$

$$Y(n) = a_{03}W(n)+a_{13}W(n-1)+a_{23}W(n-2)-b_{13}Y(n-1)-b_{23}Y(n-2) \quad (3.13)$$

โดย $a_{01}, a_{11}, a_{21}, b_{11}, b_{21}$ เป็นค่าสัมประสิทธิ์ของตัวกรองในขั้นตอนที่ 1

$a_{02}, a_{12}, a_{22}, b_{12}, b_{22}$ เป็นค่าสัมประสิทธิ์ของตัวกรองในขั้นตอนที่ 2

$a_{03}, a_{13}, a_{23}, b_{13}, b_{23}$ เป็นค่าสัมประสิทธิ์ของตัวกรองในขั้นตอนที่ 3

ในส่วนของตารางเปิดดูจะมี 3 ตาราง โดยค่าฟังก์ชัน $F_i\{.\}$ ที่เก็บในตารางที่ 3 เป็นดังนี้

TABLE 1 F_{i1} { X(n), X(n-1), X(n-2), V(n-1), V(n-2)}

TABLE 2 F_{i2} { V(n), V(n-1), V(n-2), W(n-1), W(n-2)}

TABLE 3 F_{i3} { W(n), W(n-1), W(n-2), Y(n-1), Y(n-2)}

3.5 การคำนวณหา Y(n) ของตัวกรองอันดับที่หก

$$\mathbf{V(n)} = \sum_{i=1}^{B-1} 2^{-i}\mathbf{F_{i1}\{.\}}-\mathbf{F_{o1}\{.\}} \tag{3.14}$$

$$\mathbf{W(n)} = \sum_{i=1}^{B-1} 2^{-i}\mathbf{F_{i2}\{.\}}-\mathbf{F_{o2}\{.\}} \tag{3.15}$$

$$\mathbf{Y(n)} = \sum_{i=1}^{B-1} 2^{-i}\mathbf{F_{i3}\{.\}}-\mathbf{F_{o3}\{.\}} \tag{3.16}$$

โดย $F_{i1}\{.\} = F_{i1}$ { $X_i(n), X_i(n-1), X_i(n-2), V_i(n-1), V_i(n-2)$}

$F_{o1}\{.\} = F_{o1}$ { $X_o(n), X_o(n-1), X_o(n-2), V_o(n-1), V_o(n-2)$}

$F_{i2}\{.\} = F_{i2}$ { $V_i(n), V_i(n-1), V_i(n-2), W_i(n-1), W_i(n-2)$}

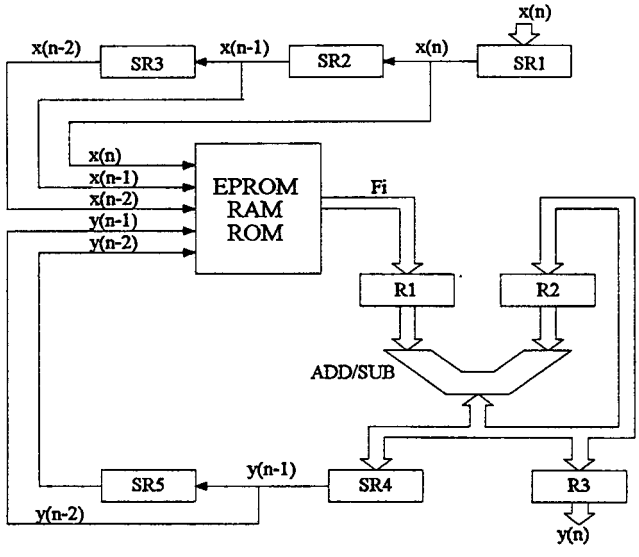
$F_{o2}\{.\} = F_{o2}$ { $V_o(n), V_o(n-1), V_o(n-2), W_o(n-1), W_o(n-2)$}

$F_{i3}\{.\} = F_{i3}$ { $W_i(n), W_i(n-1), W_i(n-2), Y_i(n-1), Y_i(n-2)$}

$F_{o3}\{.\} = F_{o3}$ { $W_o(n), W_o(n-1), W_o(n-2), Y_o(n-1), Y_o(n-2)$}

และทำการคำนวณเช่นเดียวกับในกรณีตัวกรองอันดับสอง แต่ทำวิธีการซ้ำเดิม 3 ครั้ง

โครงสร้างของตัวกรองสัญญาณแบบป้อนกลับเชิงเลขอันดับสองโดยใช้สมการ (3.14), (3.15) และ (3.16) นำมาสร้างเป็นวงจรทางอิเล็กทรอนิกส์ หรือฮาร์ดแวร์ดังแสดงในรูปที่ 3.2 SR1 และ SR4 เป็นรีจิสเตอร์ที่เลื่อนข้อมูลขนาด L บิท 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. เอาท์พุทของรีจิสเตอร์ SR1, SR2, SR3, SR4, SR5 คือลำดับสัญญาณ x(n), x(n-1), x(n-2), y(n-1), y(n-2) ตามลำดับ ถูกใช้เป็นแอดเดรสของ ROM เพื่อหาค่าของ F_i เมื่อ i=L (หมายถึงบิทที่ L) นำค่าของ F_i ที่ได้เก็บไว้ในรีจิสเตอร์ R1 จากนั้นนำค่าใน R1 บวกกับค่าใน R2 ด้วยวงจรบวก ADD/SUB ผลลัพธ์ที่ได้ถูกเก็บไว้ใน R2 พร้อมกับเลื่อนค่าหรือข้อมูลใน R2 ไปทางขวา 1 บิท

3. ทำการเลื่อนข้อมูลในรีจิสเตอร์ SR1, SR2, SR3, SR4, SR5 ไป 1 บิท เพื่อกำหนดค่าแอดเดรสของ ROM ใหม่ ได้ค่าของ F_i เมื่อ i=บิทที่ L-1 นำค่าของ F_i ที่เป็นไว้ในรีจิสเตอร์ R1 จากนั้นนำค่าใน R1 บวกกับค่าใน R2 ด้วยวงจร ADD/SUB ผลลัพธ์ที่ได้ถูกเก็บไว้ใน R2 พร้อมกับเลื่อนค่าหรือข้อมูลใน R2 ไปทางขวา 1 บิท

4. ในทำนองเดียวกันกระทำซ้ำตามขั้นตอนที่ 3 สำหรับ i=บิทที่ L-2, L-3,...,1 ตามลำดับ

5. ขั้นตอนนี้จะแตกต่างกับขั้นตอนที่ 3 และ 4 คือเมื่อเลื่อนข้อมูลในรีจิสเตอร์ SR1, SR2, SR3, SR4, SR5 ต่อจากขั้นตอนที่ 4 ไปอีก 1 บิท สำหรับหาค่า F_0 และนำค่าไปเก็บไว้ใน

R1 จากนั้นนำค่าหรือข้อมูลใน R1 ไปลบออกจากค่าใน R2 ผลลัพธ์ที่ได้คือลำดับสัญญาณ y(n) และถูกเก็บไว้ในรีจิสเตอร์ R3 เพื่อรอการแปลงเป็นสัญญาณเชิงอุปมานต่อไป

6. กระทำซ้ำตามขั้นตอน 1-5 ใหม่อีกเรื่อย ๆ สำหรับหาค่า y(n) ในลำดับถัดไป

3.6 การออกแบบสร้างวงจรกรองสัญญาณเชิงเลขอันดับ 6

เพื่อเป็นการประหยัดฮาร์ดแวร์ และลดความยุ่งยากของวงจรและง่ายต่อการสร้างอาศัยโครงสร้างของตัวกรองสัญญาณเชิงเลขอันดับสองดังที่กล่าวมาแล้ว มาต่อ Cascade กัน 3 วงจร ใช้ภาคคำนวณหรือประมวลผลชนิด DA เพียงตัวเดียวสลับใช้งาน ดังแสดงในรูปที่ 3.3 เพื่อให้ง่ายจะแยกอธิบายการทำงานเป็น 3 ขั้นตอนดังนี้

1. ขั้นตอนที่ 1

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

1.1 วงจร S/H (Sampling and Hold) (วงจรจริงรวมอยู่ในชิป A/D) ทำการสุ่มสัญญาณเชิงอุปมาน x(t)

1.2 A/D (Analog to digital converter) ถูกควบคุมด้วยสัญญาณ SC ให้ทำการแปลงสัญญาณที่ได้จากการสุ่มให้เป็นลำดับสัญญาณเชิงเลข x(n) ขนาด 8 บิท

1.3 สัญญาณควบคุม LRX0 ทำหน้าที่โหลดลำดับสัญญาณหรือข้อมูล X(n) เข้าไปเก็บไว้ในรีจิสเตอร์ RX0

1.4 สัญญาณนาฬิกา CKX0 จะทำการเลื่อนข้อมูล x(n) ในรีจิสเตอร์ RX0, RX1, RX2, RV01, RV02 ไปครั้งละ 1 บิท เอาท์พุทของแต่ละรีจิสเตอร์ คือ x(n), x(n-1), x(n-2), v_0(n-1), v_0(n-2) ที่ถูกเลื่อนแต่ละครั้งจะเป็นแอดเดรสของ ROM1 (ROM1 จะถูกเลือกด้วยสัญญาณ M1) เอาท์พุทของ ROM1 จะถูกส่งไปบวกกับค่าที่อยู่ใน ACC ด้วยวงจรบวก ADD/SUB (ควบคุมการบวกด้วยสัญญาณ A/S) ผลลัพธ์ที่ได้จะถูกโหลดเข้าไปเก็บไว้ใน ACC ด้วยสัญญาณ LACC

1.5 CKX0 จะเลื่อนข้อมูล x(n) ไปอีก 1 บิท แล้วกระทำซ้ำข้อ 1.4 จนกระทั่ง CLX0

เลื่อนข้อมูลไปถึงบิทที่ 8 จึงนำค่าที่ได้จากเอาท์พุทของ ROM1 ไปลบออกจากค่าที่อยู่ใน ACC (ที่ได้จากการบวกของการเลื่อนข้อมูล 7 ครั้ง) ผลลัพธ์ที่ได้จะถูกโหลดเข้าไปในรีจิสเตอร์ RV0, สิ้นสุดการทำงานขั้นตอนที่ 1

2. ขั้นตอนที่ 2

ตำแหน่งของ MUX1 อยู่ในตำแหน่ง B ส่วน MUX2 อยู่ตำแหน่งเดิมคือ A โดยมีรีจิสเตอร์ RV0, RV01, RV02, RV11, RV12, ROM2 ประกอบกันเป็นโครงสร้างของตัวกรองเชิงเลขอันดับที่ 2 ส่วนที่สองใช้ภาคประมวลผลแบบ DA ร่วมกันเหมือนขั้นตอนที่ 1 จะเห็นได้ว่าขั้นนี้ คือ ประมวลผลของตัวกรองในอันดับที่ 4 นั่นเอง ส่วนสัญญาณควบคุม คือ LRV0, CKV0, M2, A/S, LACC, CLACC, LRV1 เหมือนกับข้อ 1.3-1.5 โดยเปลี่ยนรีจิสเตอร์และสัญญาณควบคุมตามลำดับดังนี้

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

แอดเดรส x(n), x(n-1), x(n-2), v_0(n-1), v_0(n-2)

$\Rightarrow$ v_0(n), v_0(n-1), v_0(n-2), v_1(n-1), v_1(n-2)

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

3. ขั้นตอนที่ 3

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

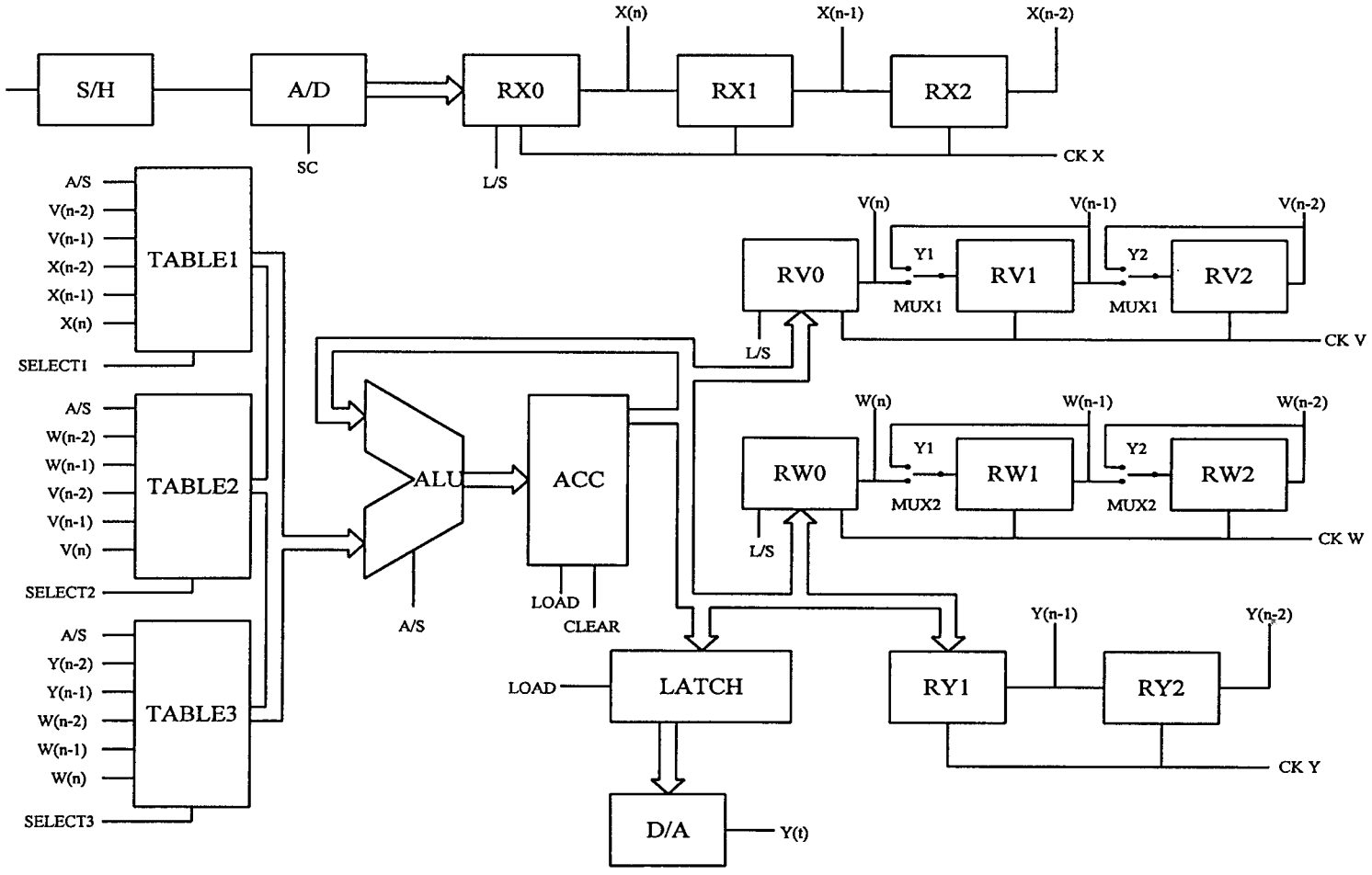
รีจิสเตอร์ RX0, RX1, RX2, RV01, RV02 $\Rightarrow$ RV1, RV11, RV12, RY, RY1

แอดเดรส x(n), x(n-1), x(n-2), v_0(n-1), v_0(n-2)

$\Rightarrow$ v_1(n), v_1(n-1), v_1(n-2), y(n-1), y(n-2)

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

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



รูปที่ 3.3 บล็อกไดอะแกรมของวงจรเชิงเลขอันดับที่ 6

3.7 ลักษณะการออกแบบ

3.7.1 การออกแบบจากล่างขึ้นบน (Bottom Up Design)

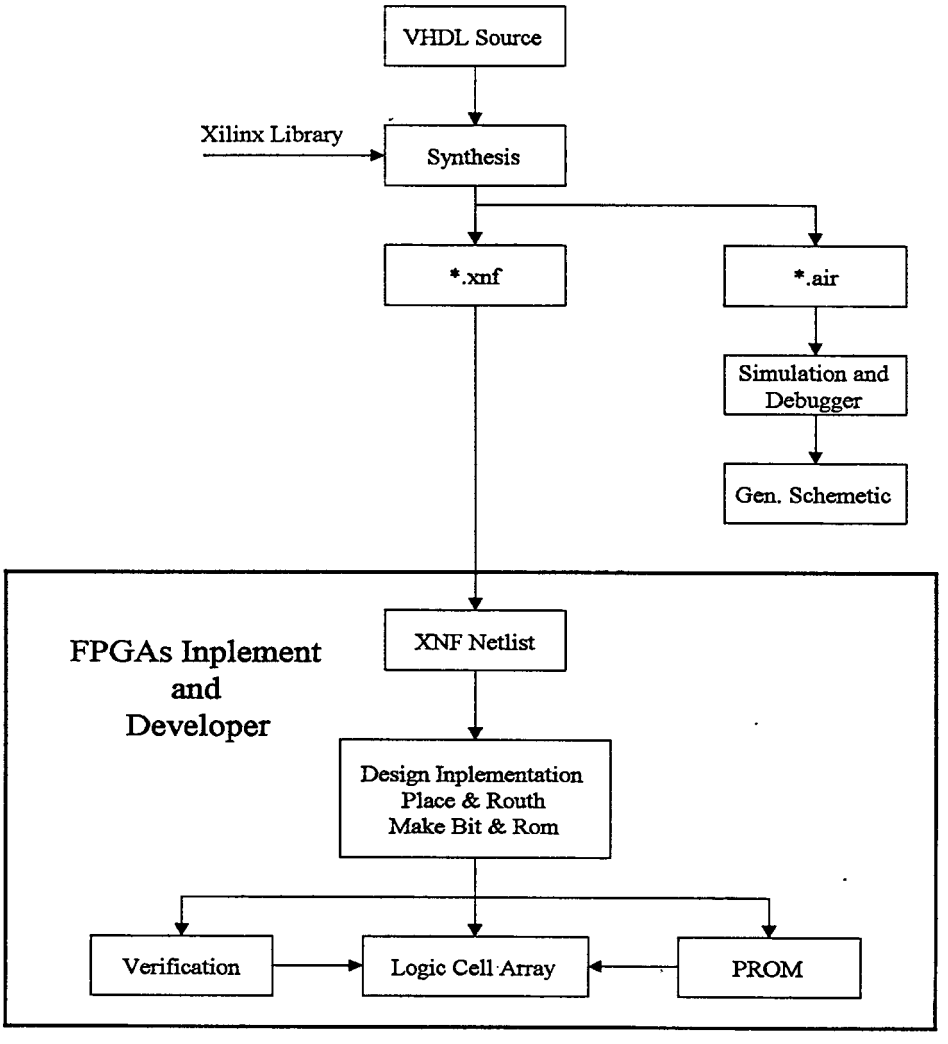
ในอดีตถึงปัจจุบันการออกแบบในระบบดิจิตอลจะเป็นลักษณะที่เรียกว่า "การออกแบบจากล่างขึ้นบน" (Bottom Up Design) คือผู้ออกแบบจะเริ่มต้นกำหนดหัวข้องาน แล้วใช้หลักการทางทฤษฎีแบ่งออกเป็นฟังก์ชันการทำงานต่างๆ แล้วเริ่มต้นออกแบบ เมื่อได้วงจรตามที่ต้องการแล้วจะต้องหาอุปกรณ์มาตรฐานต่างๆ เช่น IC 74LSXX เป็นต้น เพื่อนำมารองรับฟังก์ชันการทำงานต่างๆ ที่ได้จากการออกแบบ ถ้าไม่สามารถหาอุปกรณ์มารองรับได้จะต้องออกแบบหรือดัดแปลงวงจรใหม่ ในขั้นตอนสุดท้าย คือการจำลองการทำงานโดยทดลองจากวงจรต้นแบบ

3.7.2 การออกแบบจากบนลงล่าง (Top Down Design)

จากการออกแบบในหัวข้อ 3.7.1 นั้น ในกรณีการทำงานของวงจรไม่ตรงตามข้อกำหนดต้องทำการออกแบบ และประกอบวงจรใหม่อีกครั้ง ซึ่งจะเห็นว่าขั้นตอนและกระบวนการดังกล่าวยุ่งยากและใช้เวลามาก ดังนั้นในการออกแบบวงจรสมัยใหม่สามารถออกแบบระบบดิจิตอลได้จากแนวคิดโดยสังเขป โดยวิธีการเขียนรูปแบบแล้วทดลองการทำงานของรูปแบบนั้นจนเป็นที่น่าพอใจแล้วค่อยๆ เพิ่มเติมรายละเอียดของระบบไปทีละขั้น ซึ่งในแต่ละขั้นตอนสามารถที่จะจำลองการทำงานภายใต้สภาวะเดิมได้ ทำให้ไม่มีโอกาสที่รูปแบบการทำงานจะผิดไปจากวัตถุประสงค์เดิม เมื่อเกิดข้อผิดพลาดก็สามารถแก้ไขได้ทันที หลังจากนั้นจึงนำไปลงอุปกรณ์และทดสอบการทำงานของวงจรซ้ำอีกครั้ง การออกแบบในลักษณะนี้เรียกว่า "การออกแบบจากบนลงล่าง" (Top Down Design)

3.8 วิธีการออกแบบ

จากข้อดีของการออกแบบจากบนลงล่างนั้น ทำให้โครงงานนี้เลือกใช้วิธีการออกแบบจากบนลงล่างในการสร้างวงจรกรองสัญญาณเชิงเลข โดยทำการออกแบบวงจรกรองความถี่ป้อนกลับความถี่ต่ำเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้ภาษา VHDL ซึ่งเป็นซอฟแวร์ของบริษัทไซลิงค์ (Xilinx) และทำการสร้างเป็นวงจรตามที่ได้ออกแบบไว้แล้วด้วยอุปกรณ์ FPGAs ของบริษัทไซลิงค์ (Xilinx) ขั้นตอนการออกแบบโปรแกรมและอุปกรณ์ดังกล่าวแสดงดังรูปที่ 3.4



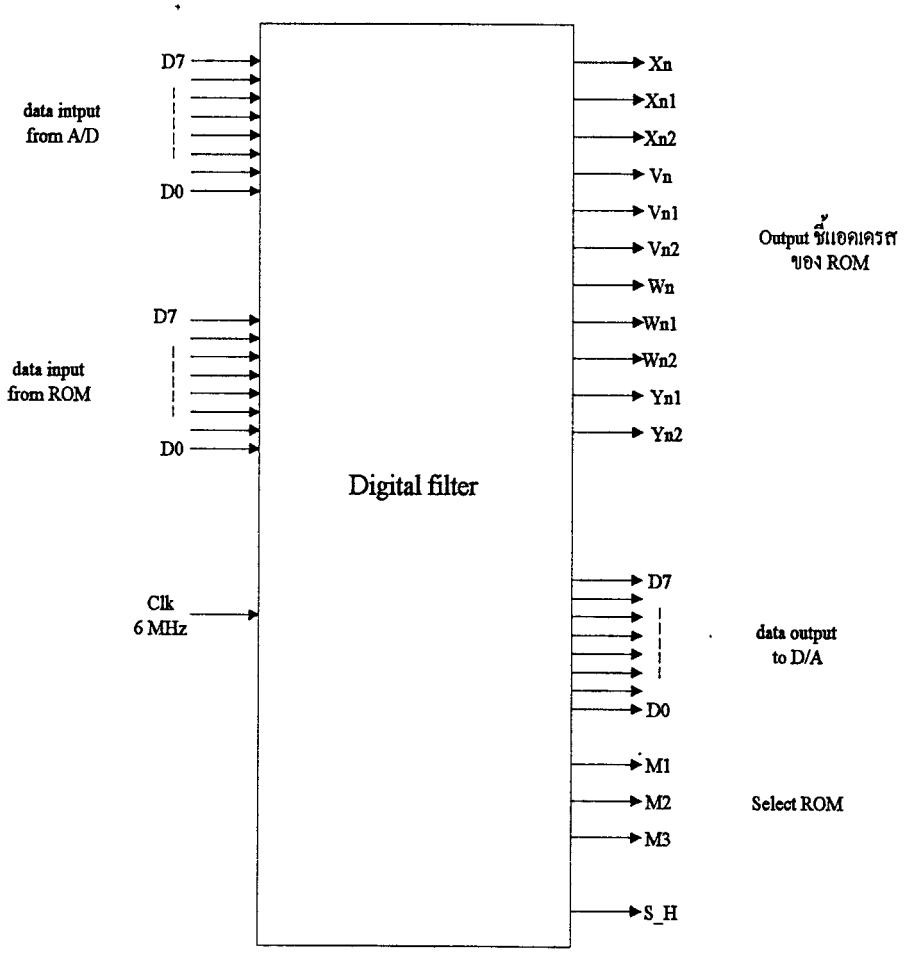
รูปที่ 3.4 ผังการทำงานของการออกแบบวงจรโดยใช้ซอฟแวร์และอุปกรณ์ FPGAsของบริษัทไซลิงค์ (Xilinx)

3.9 การออกแบบวงจรกรองสัญญาณเชิงเลขด้วยภาษา VHDL

จากบล็อกไดอะแกรมของวงจรกรองสัญญาณเชิงเลขอันดับที่ 6 เราสามารถที่จะใช้ภาษา VHDL ออกแบบโดยการแบ่งบล็อกการทำงานออกเป็น 4 ส่วน คือ

1. ส่วนการควบคุม (Control Unit)
2. ส่วนการประมวลผล (Process Unit)
3. ส่วนการเลื่อนข้อมูล (Shift Data Unit)
4. ส่วนตารางเก็บค่าสัมประสิทธิ์ (Efficiency Table Unit)

ซึ่งขั้นตอนการใช้ภาษา VHDL ออกแบบนั้นเริ่มจากการกำหนดบล็อกรวมของการทำงาน แล้วกำหนดสัญญาณอินพุทและเอาท์พุทที่ต้องการดังรูปที่ 3.5



รูปที่ 3.5 บล็อกรวมการทำงานของวงจรกรองสัญญาณเชิงเลข

จากรูปที่ 3.5 บล็อกรวมการทำงานของวงจรกรองสัญญาณเชิงเลข ประกอบด้วย

1. ข้อมูลอินพุทจาก A/D ซึ่งมีขนาด 8 บิท โดยบิท D7 เป็นบิทที่มีนัยสำคัญสูงสุด

2. ข้อมูลอินพุท ค่าสัมประสิทธิ์ของวงจรกรองสัญญาณจะได้จาก EPROM ซึ่งได้ทำการบันทึกไว้มีขนาด 8 บิทเช่นเดียวกัน โดยบิท D7 เป็นบิทที่มีนัยสำคัญสูงสุด

3. สัญญาณคล็อก (Clock) ขนาด 6 Mhz

4. สัญญาณชี้ตำแหน่งค่าสัมประสิทธิ์ของตัวกรองสัญญาณเชิงเลข ประกอบด้วยสัญญาณ Xn, Xn1, Xn2, Vn, Vn1, Vn2, Wn, Wn1, Wn2, Yn1, และ Yn2

5. ข้อมูลเอาท์พุทเป็นสัญญาณเชิงเลข เพื่อนำไปแปลงเป็นสัญญาณเชิงอุปมาน โดยใช้วงจร D/A

6. สัญญาณ M1, M2 และ M3 เพื่อใช้ในการเลือก EPROM ตัวที่ต้องการ

7. สัญญาณ S_H เป็นสัญญาณกำหนดความถี่ในการ Sampling ให้กับวงจร A/D

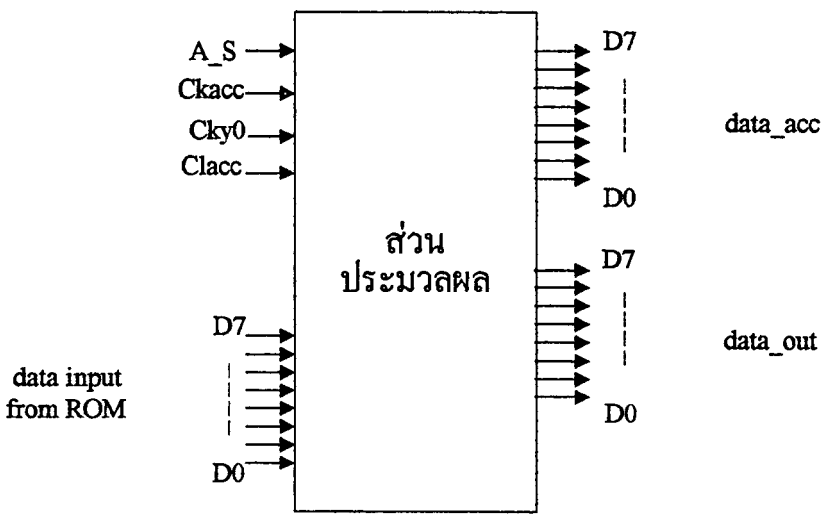
3.10 ขั้นตอนการออกแบบบล็อกการทำงานภายใน

จากที่กล่าวมาแล้วในหัวข้อที่ 3.9 เราสามารถแบ่งบล็อกการทำงานออกเป็น 4 ส่วน ซึ่งจะมีอยู่ 3 ส่วนที่อยู่ภายในบล็อกการทำงานรวมของวงจรกรองสัญญาณเชิงเลขคือ ส่วนการควบคุม (Control Unit), ส่วนการประมวลผล (Process Unit), และส่วนการเลื่อนข้อมูล (Shift Data Unit) สำหรับส่วนของตารางค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขนั้นจะต่ออยู่ภายนอก

ดังนั้นในการออกแบบด้วยภาษา VHDL จะทำการออกแบบเพียง 3 ส่วนที่อยู่ในบล็อกการทำงานรวม ดังนี้

3.10.1 ส่วนประมวลผล

ทำหน้าที่ในการรับข้อมูลจากตารางค่าสัมประสิทธิ์มาทำการประมวลผลโดยใช้วิธีการบวกสะสม และจะเป็นการลบกันเมื่อสัญญาณ A_S มีค่าเป็น "1" ดังแสดงในรูปที่ 3.6



รูปที่ 3.6 บล็อกการทำงานของส่วนประมวลผล

จากรูปที่ 3.6 บล็อกการทำงานของส่วนประมวลผล ประกอบด้วยสัญญาณต่างๆ ดังนี้

1. สัญญาณ A_S เป็นสัญญาณที่ใช้กำหนดการบวกหรือลบข้อมูล

2. สัญญาณ Ckacc เป็นสัญญาณนาฬิกาของส่วนประมวล

3. ข้อมูลจาก ROM เป็นค่าสัมประสิทธิ์ของวงจรกรองเชิงเลขที่เก็บใน EPROM

4. ข้อมูล data_acc จะเป็นข้อมูลผลลัพธ์ของการบวกกันภายในส่วนประมวลผล

5. ข้อมูล data_out จะเป็นข้อมูลที่ส่งออกเอาท์พุท โดยจะใช้สัญญาณ Cky0 เป็นสัญญาณนาฬิกาที่จะกำหนดการส่งข้อมูล

6. สัญญาณ Clacc จะเป็นสัญญาณที่ใช้เคลียร์ข้อมูลในส่วนประมวลผล

ลักษณะการทำงานของส่วนประมวลผลคือ จะทำการรับข้อมูลค่าสัมประสิทธิ์ของวงจรตัวกรองสัญญาณเชิงเลขมาจาก EPROM แล้วจะทำการบวกหรือลบกับค่าข้อมูลที่อยู่ภายในส่วนประมวลผล ซึ่งขึ้นอยู่กับว่าสัญญาณ A_S จะเป็น "1" หรือ "0" โดยการทำงานจะใช้ Ckacc ควบคุมการทำงาน ข้อมูลที่ได้จะถูกเก็บค่าไว้ภายในและถูกส่งออกมาทาง data_acc และจะตรวจสอบค่า Cky0 ถ้ามีค่าเป็น "1" ก็จะส่งข้อมูลออกมาที่ data_out

ขอยกตัวอย่างให้เห็นถึงโปรแกรมส่วนประมวลผล ซึ่งจะทำการบวกลบโดยการนำเอาค่าสัมประสิทธิ์ที่เก็บไว้ใน EPROM มาทำการบวกกัน 7 ครั้ง ในลักษณะการบวกสะสม และในครั้งที่ 8 จะนำมาลบ ดังแสดงในรูปที่ 3.7

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY acc IS
  PORT (
      ckacc          :IN STD_LOGIC;
      clacc          :IN STD_LOGIC;
      rom_data       :IN STD_LOGIC_VECTOR (7 downto 0);
      a_s            :IN STD_LOGIC;
      cky            :IN STD_LOGIC;
```

รูปที่ 3.7 โปรแกรมของส่วนประมวลผล

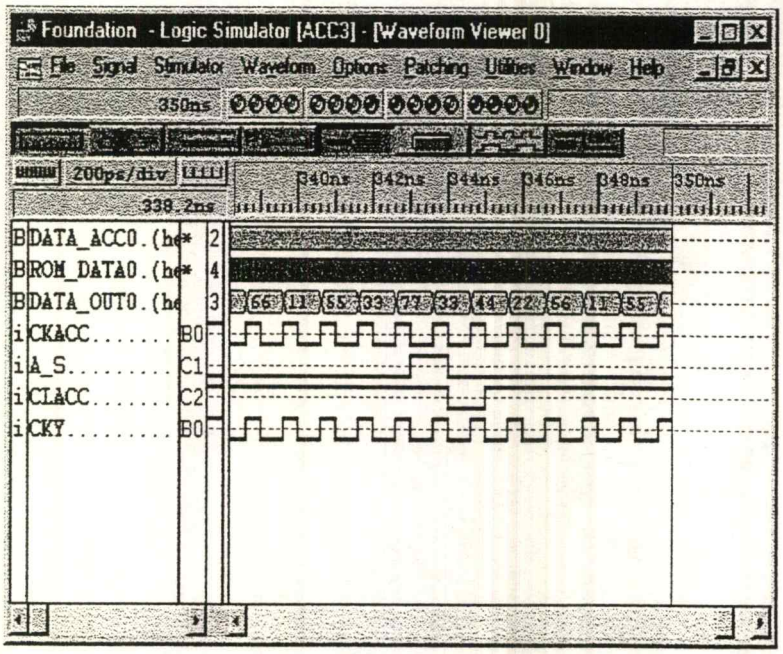
```
        data_acc        :BUFFER STD_LOGIC_VECTOR (7 downto 0);
        data_out        :BUFFER STD_LOGIC_VECTOR (7 downto 0)
    );
END ACC;

ARCHITECTURE acc_arch of acc is
BEGIN
        PROCESS(ckacc,clacc,rom_data,a_s,cky)
        variable base,oper,result : std_logic_vector(7 downto 0):="00000000";
        variable carry : std_logic;
        BEGIN
                IF(ckacc='1' and ckacc'EVENT ) THEN
                        IF(clacc = '0' ) THEN
                                result:="00000000";
                                base:="00000000";
                        END IF;
                                base:=result;
                                oper:=rom_data;
                        IF (a_s='1') THEN
                                carry:='1';
                        ELSE carry:='0';
                        END IF;
                        FOR lp in 0 to 7 LOOP
                                IF(a_s='1')THEN
                                        oper(lp):=not oper(lp);
                                END IF;
                result(lp) := (base(lp) xor oper(lp)) xor carry;
                carry:=(base(lp) and oper(lp)) or (carry and (base(lp) or oper(lp)));
                        END LOOP;
```

รูปที่ 3.7 (ต่อ) โปรแกรมของส่วนประมวลผล

```
                    data_acc<=result;
                IF(cky='1' ) THEN
                    data_out<=result;
                END IF;
            END IF;
        END PROCESS;
    END acc_arch;
```

รูปที่ 3.7 (ต่อ) โปรแกรมของส่วนประมวลผล

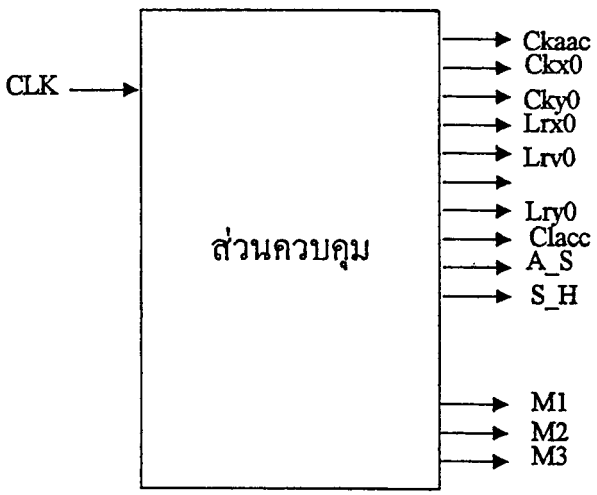
หลังจากที่เขียนโปรแกรมของส่วนประมวลผลเสร็จสมบูรณ์ และทำการแปลภาษาจนไม่พบข้อผิดพลาด ขั้นตอนต่อมาจะเป็นการทดสอบการทำงานของโปรแกรมที่เขียนขึ้นมา เพื่อดูว่าการทำงานเป็นไปตามจุดประสงค์ที่ออกแบบไว้ ซึ่งได้ผลการจำลองการทำงานของส่วนประมวลผลดังรูปที่ 3.8



รูปที่ 3.8 การจำลองการทำงานของส่วนประมวลผล

3.10.2 ส่วนควบคุม

ทำหน้าที่ให้กำเนิดสัญญาณควบคุมการทำงานทั้งหมดของวงจรกรองสัญญาณเชิงเลข ซึ่งเราจะทำการกำหนดบล็อกการทำงานดังรูปที่ 3.9 เพื่อใช้ภาษา VHDL ในการบรรยายการทำงานต่อไป



รูปที่ 3.9 บล็อกการทำงานของส่วนควบคุม

จากรูปที่ 3.9 บล็อกการทำงานของส่วนควบคุม ประกอบด้วยสัญญาณของส่วนต่างๆ ดังนี้

1. สัญญาณ CLK เป็นอินพุทของวงจร โดยรับโดยตรงจากภายนอกมีค่าเท่ากับ 6 Mhz
2. สัญญาณ Ckacc เป็นสัญญาณนาฬิกาสำหรับส่วนประมวลผล

   สัญญาณ Ckx0 เป็นสัญญาณนาฬิกาสำหรับส่วนเลื่อนข้อมูล

   สัญญาณ Cky0 เป็นสัญญาณนาฬิกาสำหรับโหลดข้อมูลมาเอาท์พุท
3. สัญญาณ Lrx0 เป็นสัญญาณในการโหลดข้อมูลจากรีจิสเตอร์ rx0 ในวงจรเลื่อนข้อมูล ซึ่งจะทำงานที่ลอจิก "1"

   สัญญาณ Lrv0 เป็นสัญญาณในการโหลดข้อมูลจากรีจิสเตอร์ rv0 ในวงจรเลื่อนข้อมูล ซึ่งจะทำงานที่ลอจิก "1"

   สัญญาณ Lrw0 เป็นสัญญาณในการโหลดข้อมูลจากรีจิสเตอร์ rw0 ในวงจรเลื่อนข้อมูล ซึ่งจะทำงานที่ลอจิก "1"

สัญญาณ Lry0 เป็นสัญญาณในการโหลดข้อมูลจากรีจิสเตอร์ ry0 ในวงจรเลื่อนข้อมูล ซึ่งจะทำงานที่ลอจิก "1"

4. สัญญาณ A_S เป็นสัญญาณที่ใช้กำหนดการบวกลบ โดยถ้า A_S เป็น "1" ที่ส่วนประมวลผลจะเป็นการลบข้อมูล ถ้าเป็น "0" จะเป็นการบวกกันของข้อมูล

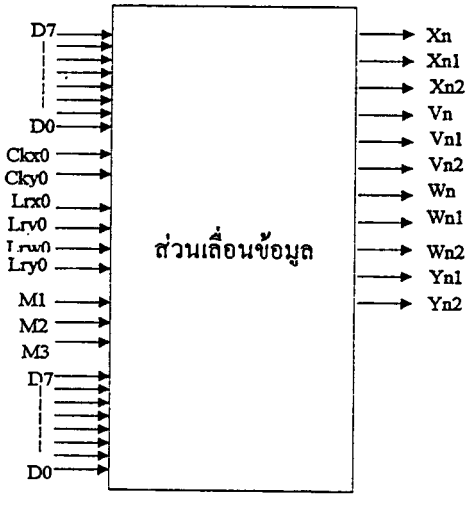
5. สัญญาณ S_H เป็นสัญญาณ Sampling ที่จะส่งออกไปให้ A/D

6. สัญญาณ M1, M2, และ M3 จะเป็นสัญญาณสำหรับเลือกใช้ EPROM และจะใช้ในการเลือกชุดในการเลื่อนข้อมูลด้วย

จากนั้นจึงทำการเขียนโปรแกรม VHDL ได้ดังโปรแกรมที่ 2 ในภาคผนวก และทำการจำลองการทำงานส่วนควบคุม ได้ผลการจำลองการทำงานดังรูปที่ 4.1 ในบทที่ 4

3.10.3 ส่วนเลื่อนข้อมูล

ทำหน้าที่ในการเลื่อนข้อมูล ซึ่งการทำงานภายในจะประกอบด้วยชุดเลื่อนข้อมูลเข้าแบบขนาน ออกแบบอนุกรม และชุดเลื่อนข้อมูลเข้าแบบอนุกรม ออกแบบอนุกรม ซึ่งผลจากการเลื่อนข้อมูลอนุกรมนั้นจะถูกส่งออกมาเพื่อนำไปชี้ตำแหน่งของ EPROM เพื่ออ่านค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลขซึ่งสามารถออกแบบได้ดังรูปที่ 3.10



รูปที่ 3.10 บล็อกการทำงานของส่วนเลื่อนข้อมูล

จากรูปที่ 3.10 จะประกอบด้วยสัญญาณต่างๆ ที่ใช้ในการเลื่อนข้อมูลดังนี้

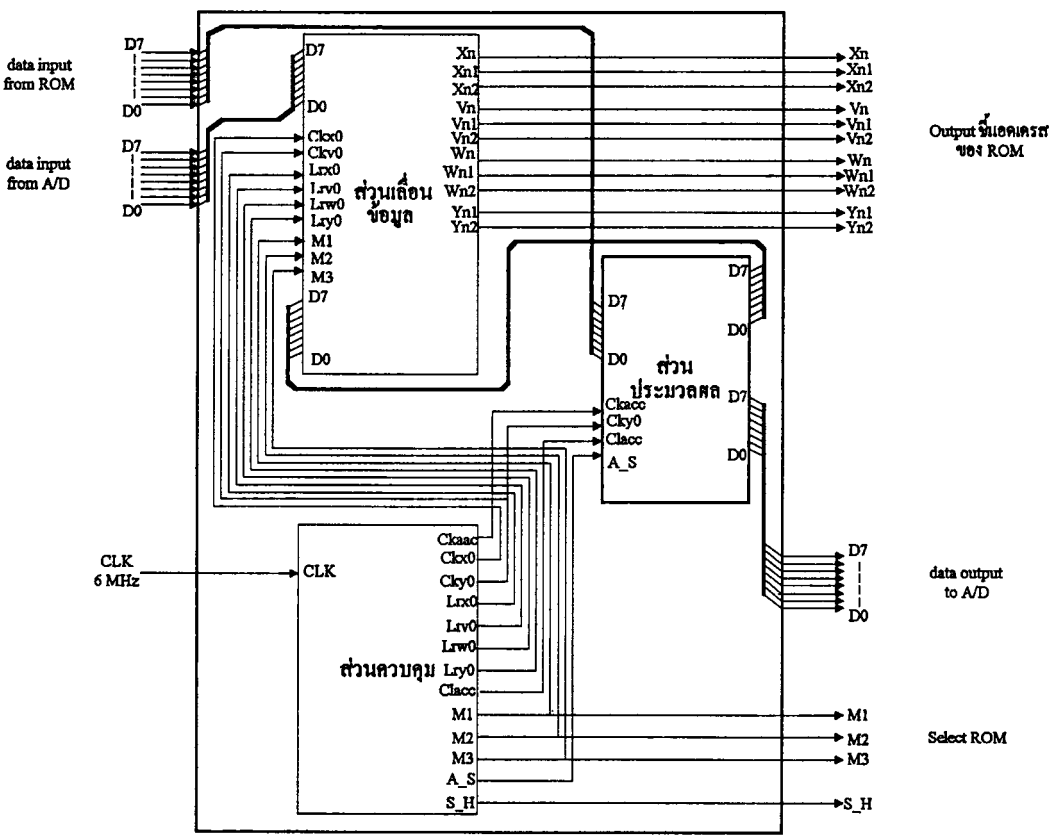
1. Data Input จาก D/A ขนาด 8 บิท

2. สัญญาณนาฬิกา Ckx0 และ Cky0 ซึ่งเป็นสัญญาณที่ใช้ควบคุมการเลื่อนข้อมูลภายในทั้งแบบเข้าขนานออกอนุกรม และแบบเข้าอนุกรมออกอนุกรม

3. สัญญาณที่ใช้ในการโหลดข้อมูลประกอบด้วย Lrx0, Lrv0, Lrw0, Lry0 ซึ่งจะใช้สำหรับโหลดข้อมูลจาก Data Input หรือจาก Data_acc

4. สัญญาณควบคุม M1, M2, M3 เป็นสัญญาณควบคุมการทำงานของการเลื่อนข้อมูลว่าจะให้ทำการเลื่อนข้อมูลชุดใด ซึ่งภายในจะมีชุดการเลื่อนข้อมูลอยู่ 3 ชุดใหญ่ๆ

5. สัญญาณที่ใช้ชี้ตำแหน่งค่าสัมประสิทธิ์ของวงจรกรองสัญญาณเชิงเลข ซึ่งจะถูกส่งไปยัง EPROM ประกอบด้วยสัญญาณ Xn, Xn1, Xn2, Vn, Vn1, Vn2, Wn, Wn1, Wn2, Yn1, และ Yn2



รูปที่ 3.11 บล็อกการทำงานรวม

จากนั้นจึงทำการเขียนโปรแกรม VHDL ได้ดังโปรแกรมที่ 2 ในภาคผนวก และทำการจำลองการทำงานส่วนควบคุม ได้ผลการจำลองการทำงานดังรูปที่ 4.1 ในบทที่ 4

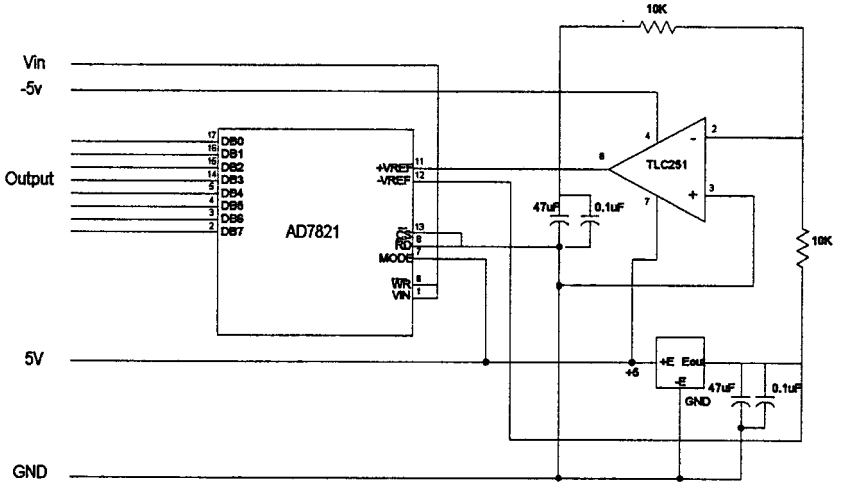
หลังจากออกแบบทั้ง 3 ส่วนแล้ว จะทำการเขียนโปรแกรม VHDL เพื่อจำลองการทำงานของทั้ง 3 ส่วน และนำมาต่อรวมกันให้เป็นวงจรรวมเดียวกัน โดยใช้ภาษา VHDL ในการบรรยายบล็อกการทำงานรวม เพื่อต่อเชื่อมกันดังรูปที่ 3.11

3.11 การออกแบบวงจรกรองป้อนกลับความถี่ต่ำเชิงเลขแบบบัตเตอร์เวอร์ธ อันดับที่ 6 ด้วยอุปกรณ์ FPGAs

วงจรที่ใช้ในการทดสอบวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลขอันดับ 6 เป็นแบบบอร์ดตัวอย่างของ FPGAs ซึ่งจะทำการโปรแกรมผ่านทางดาวน์โหลดเคเบิล (Download Cable) โดยวงจรที่ใช้ทดสอบวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลขอันดับ 6 แบ่งการทำงานออกเป็น 4 ส่วนด้วยกัน คือ

3.11.1 ส่วนอินพุท

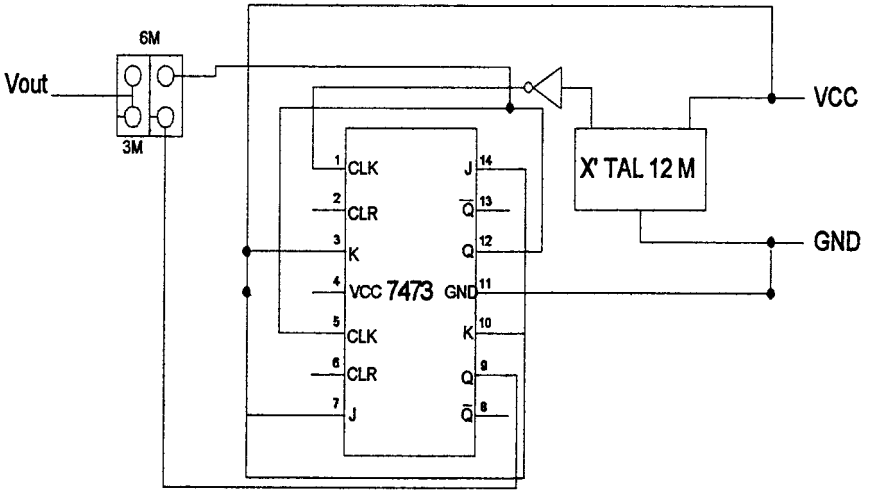
คือภาควงจร A/D ซึ่งจะรับอินพุทที่เป็นสัญญาณเชิงอุปมาณเข้ามา แล้วทำการแปลงให้เป็นสัญญาณเชิงเลขขนาด 8 บิท ซึ่งจะเป็นอินพุทให้กับอุปกรณ์ FPGAs ซึ่งการออกแบบแสดงไว้ดังรูปที่ 3.12



รูปที่ 3.12 การออกแบบส่วนอินพุท

3.11.2 ส่วนสร้างสัญญาณนาฬิกา

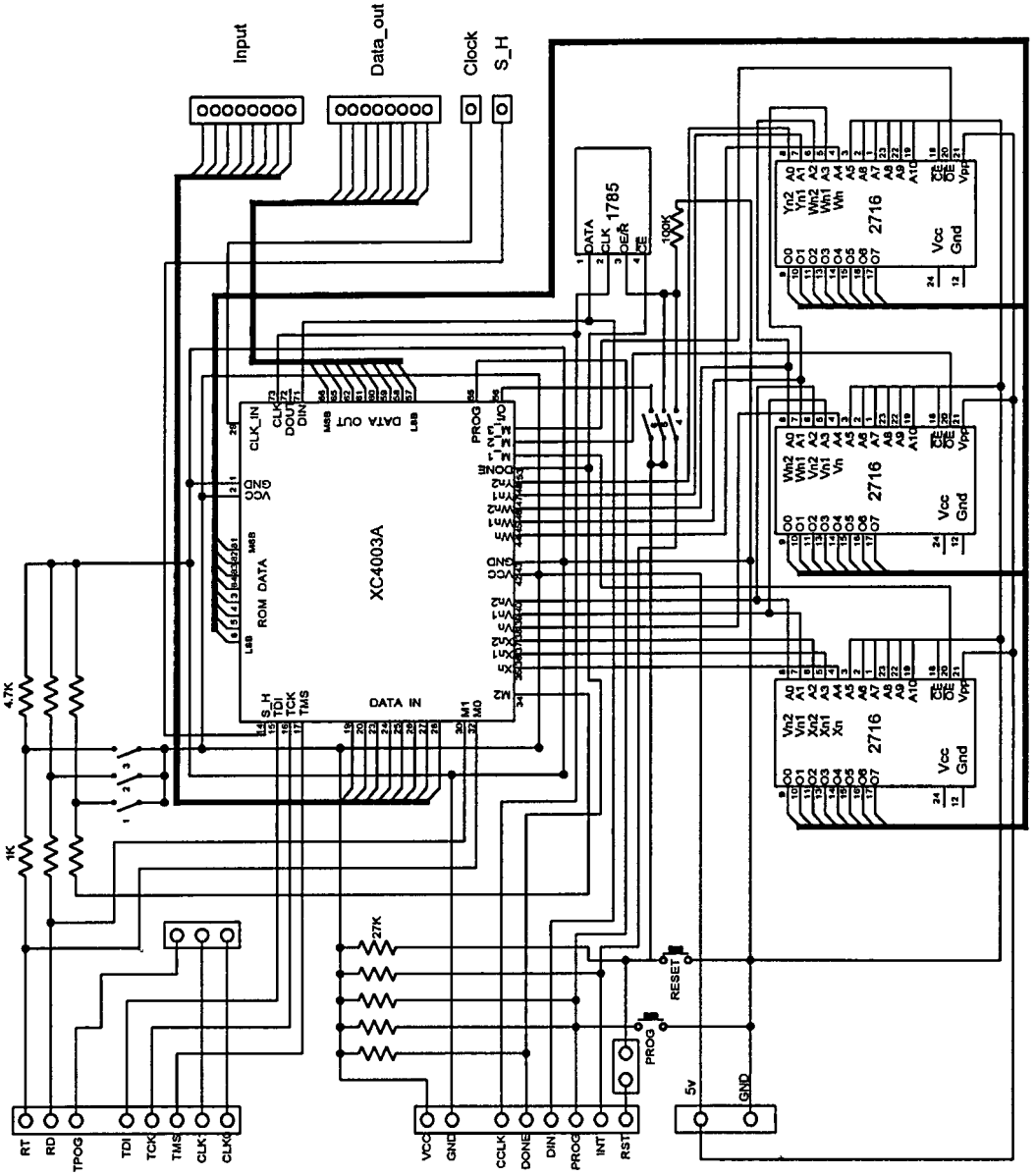
เราสามารถสร้างสัญญาณนาฬิกาได้โดยการใช้คริสตัลโมดูลในการกำเนิดสัญญาณนาฬิกาความถี่ 12 MHz แล้วทำการหารความถี่ลงเหลือ 6 MHz และ 3 MHz โดยจะใช้สวิทช์ในการเลือกใช้งานว่าจะใช้ความถี่ใด ดังแสดงในรูปที่ 3.13



รูปที่ 3.13 การออกแบบส่วนสร้างสัญญาณนาฬิกา

3.11.3 ส่วนประมวลผล

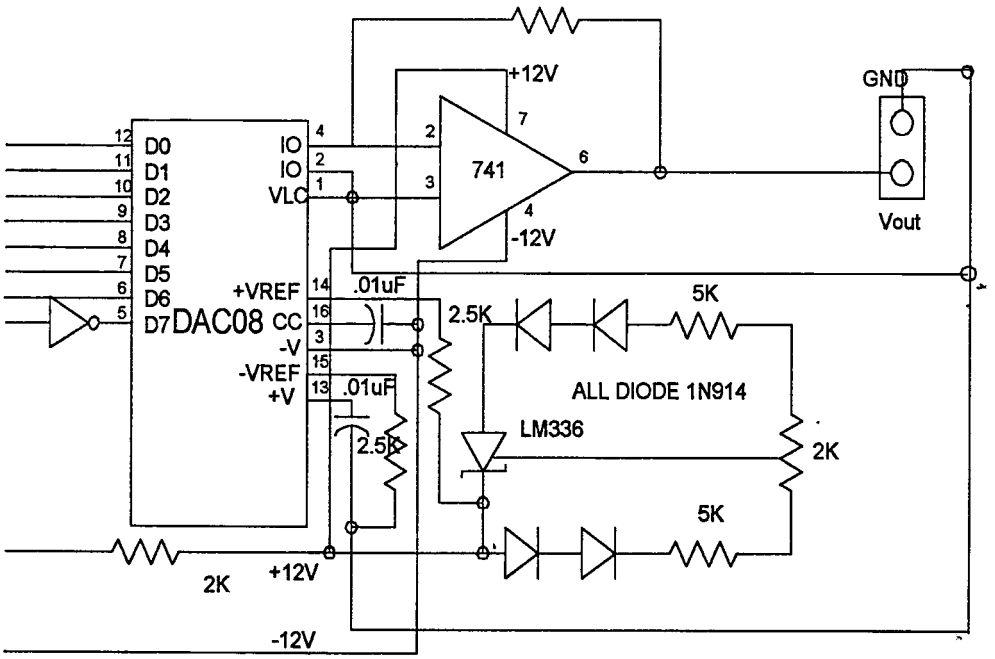
คือชุดอุปกรณ์ FPGAs ซึ่งเป็นการออกแบบการต่อใช้งาน FPGAs เบอร์ XC4005E รวมกับค่าตารางสัมประสิทธิ์ของวงจรที่เก็บไว้ใน EPROM ด้วย ซึ่งในส่วนนี้จะเป็นการนำสัญญาณเชิงเลขที่รับมาจากอินพุทมาทำการประมวลผลและส่งออกเอาท์พุทไปให้กับภาคเอาท์พุทต่อไป ซึ่งทำการออกแบบได้ดังรูปที่ 3.14



รูปที่ 3.14 การออกแบบส่วนประมวลผล

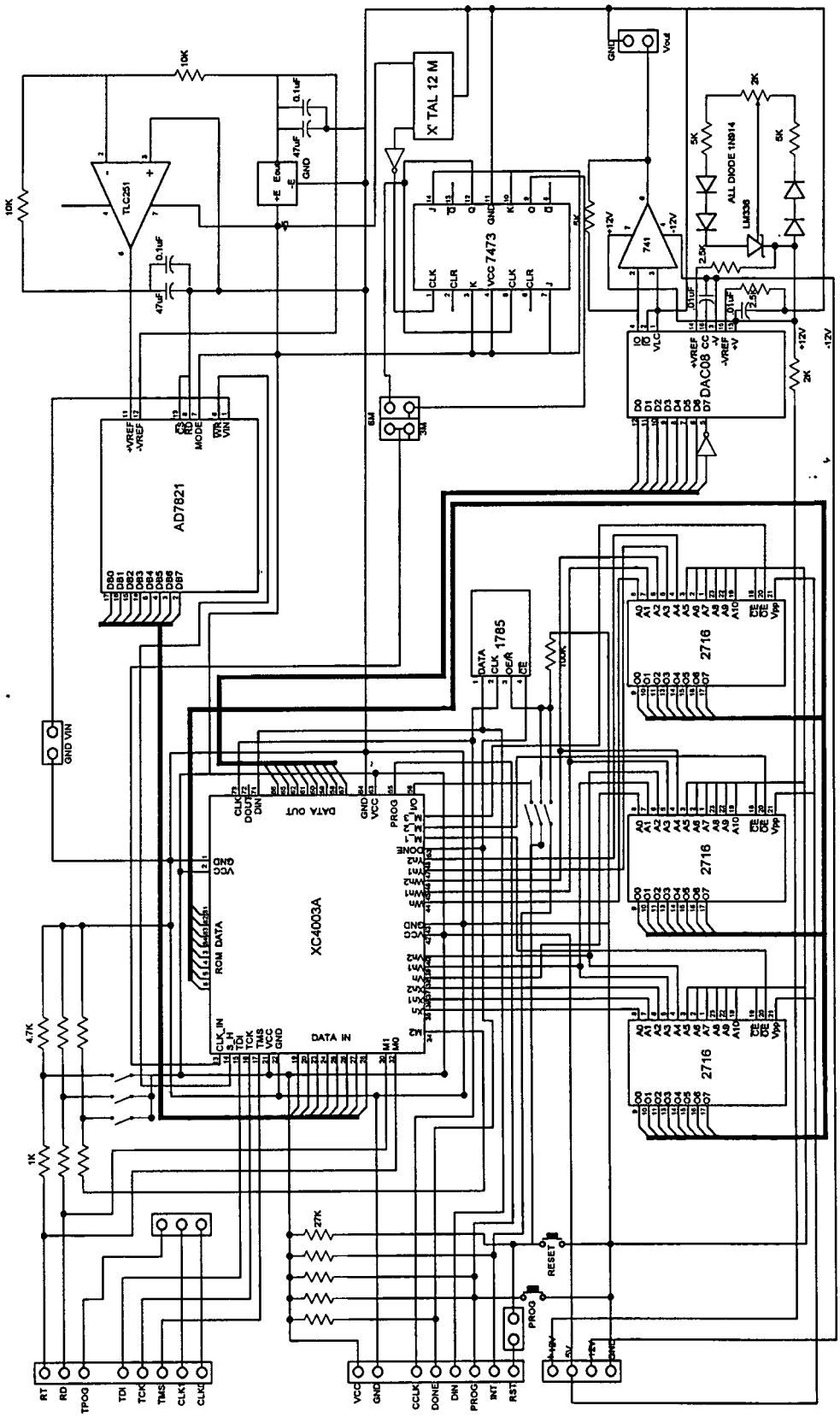
3.11.4 ส่วนเอาท์พุท

เป็นวงจร D/A ซึ่งจะทำการแปลงสัญญาณเชิงเลขที่ได้จากภาคประมวลผลให้กลับเป็นสัญญาณเชิงอุปมาณเหมือนเดิม โดยรับสัญญาณอินพุทขนาด 8 บิทเข้ามาซึ่งทำการออกแบบวงจรได้ดังรูปที่ 3.15



รูปที่ 3.15 การออกแบบส่วนเอาท์พุท

หลังจากทำการออกแบบวงจรทุกส่วนแล้วนำวงจรมารวมกันเป็นวงจรรวมดังรูปที่ 3.16



รูปที่ 3.16 การออกแบบวงจรรวม

บทที่ 4

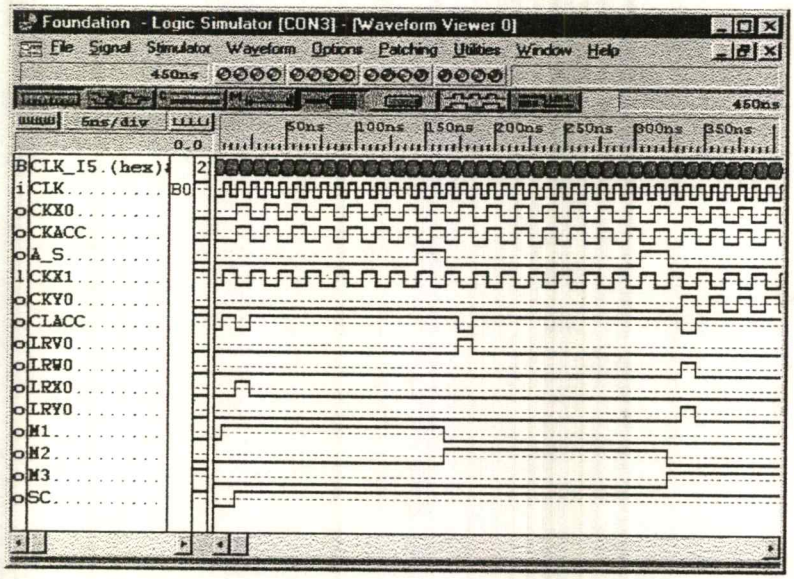
การทดลองและผลการทดลอง

4.1 ผลการทดลองโปรแกรม VHDL

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

4.1.1 ส่วนควบคุม

จากขั้นตอนการออกแบบบล็อกส่วนควบคุมในหัวข้อ 3.10.2 แล้วนำมาเขียนเป็นโปรแกรมภาษา VHDL ได้ดังโปรแกรมที่ 2 ในภาคผนวก ก. จากนั้นนำโปรแกรม VHDL มาทำการสังเคราะห์และจำลองการทำงาน ได้ผลดังรูปที่ 4.1

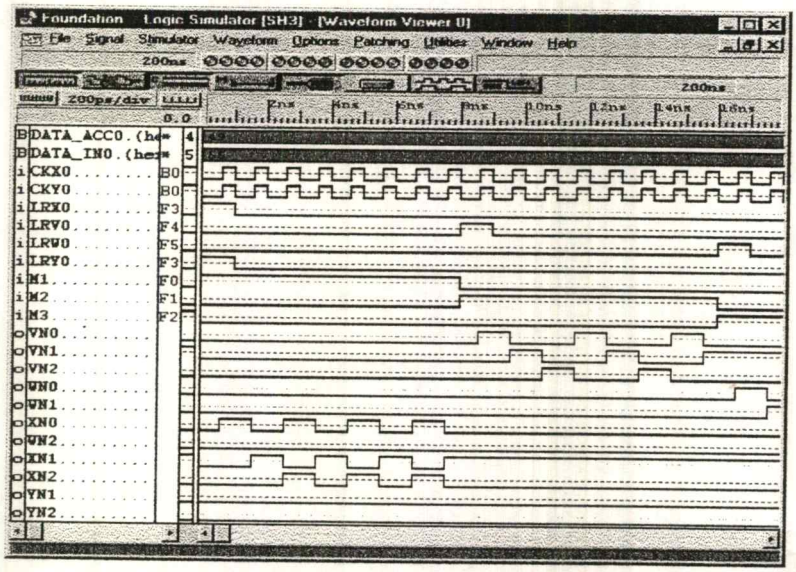


รูปที่ 4.1 ผลการจำลองการทำงานของส่วนควบคุม

สัญญาณที่ได้จากส่วนควบคุมซึ่งจำลองการทำงานโดยทดลองป้อนสัญญาณคล็อกเข้าที่อินพุท แล้วจะได้สัญญาณเอาท์พุทซึ่งเป็นสัญญาณการควบคุมการทำงานทั้งหมดของวงจรกรองสัญญาณเชิงเลข

4.1.2 ส่วนเลื่อนข้อมูล

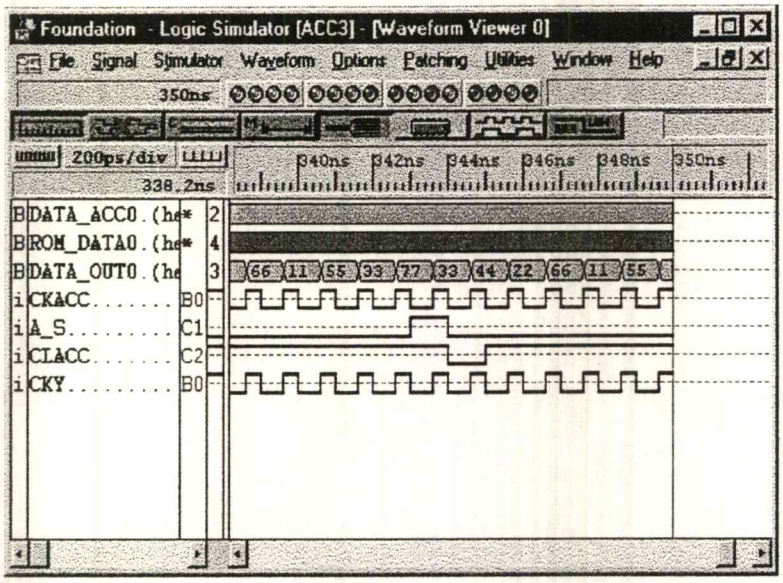
จากขั้นตอนการออกแบบบล็อกส่วนเลื่อนข้อมูลในหัวข้อ 3.10.3 แล้วนำมาเขียนเป็นโปรแกรมภาษา VHDL ได้ดังโปรแกรมที่ 3 ในภาคผนวก ก. จากนั้นนำโปรแกรม VHDL มาทำการสังเคราะห์และจำลองการทำงาน โดยทำการป้อนสัญญาณอินพุทโดยกำหนดให้เหมือนกับสัญญาณควบคุมจากส่วนควบคุม ส่วนข้อมูลอินพุทจาก A/D กำหนดเป็น 01H จะได้ผลการจำลองการทำงานดังรูปที่ 4.2



รูปที่ 4.2 ผลการจำลองการทำงานของส่วนเลื่อนข้อมูล

4.1.3 ส่วนประมวลผล

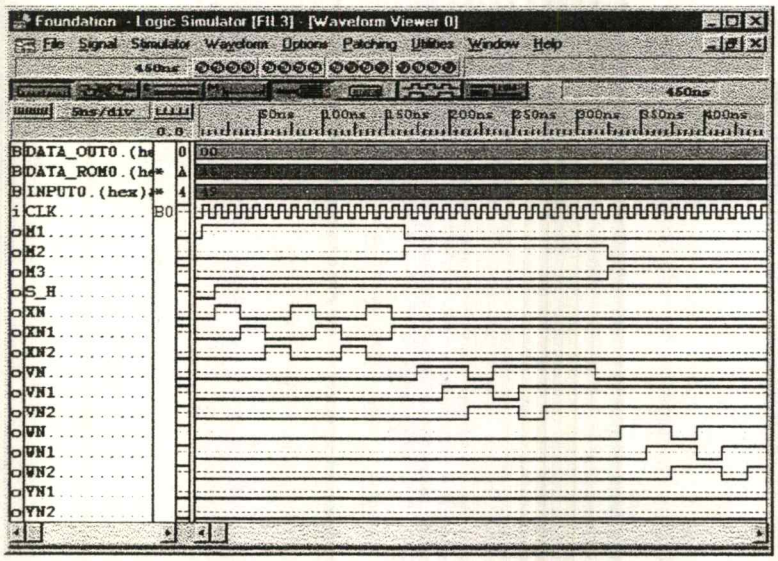
จากขั้นตอนการออกแบบบล็อกส่วนประมวลผลในหัวข้อ 3.10.1 แล้วนำมาเขียนเป็นโปรแกรมภาษา VHDL ได้ดังโปรแกรมที่ 1 ในภาคผนวก ก. จากนั้นนำโปรแกรม VHDL มาทำการสังเคราะห์และจำลองการทำงาน โดยทำการป้อนสัญญาณอินพุทที่ส่วนประมวลผลกำหนดให้ข้อมูลจาก ROM เท่ากับ 01H และกำหนดสัญญาณควบคุมต่างๆ ให้เหมือนกับสัญญาณจากส่วนควบคุม ซึ่งจะได้ผลการจำลองการทำงานดังรูปที่ 4.3



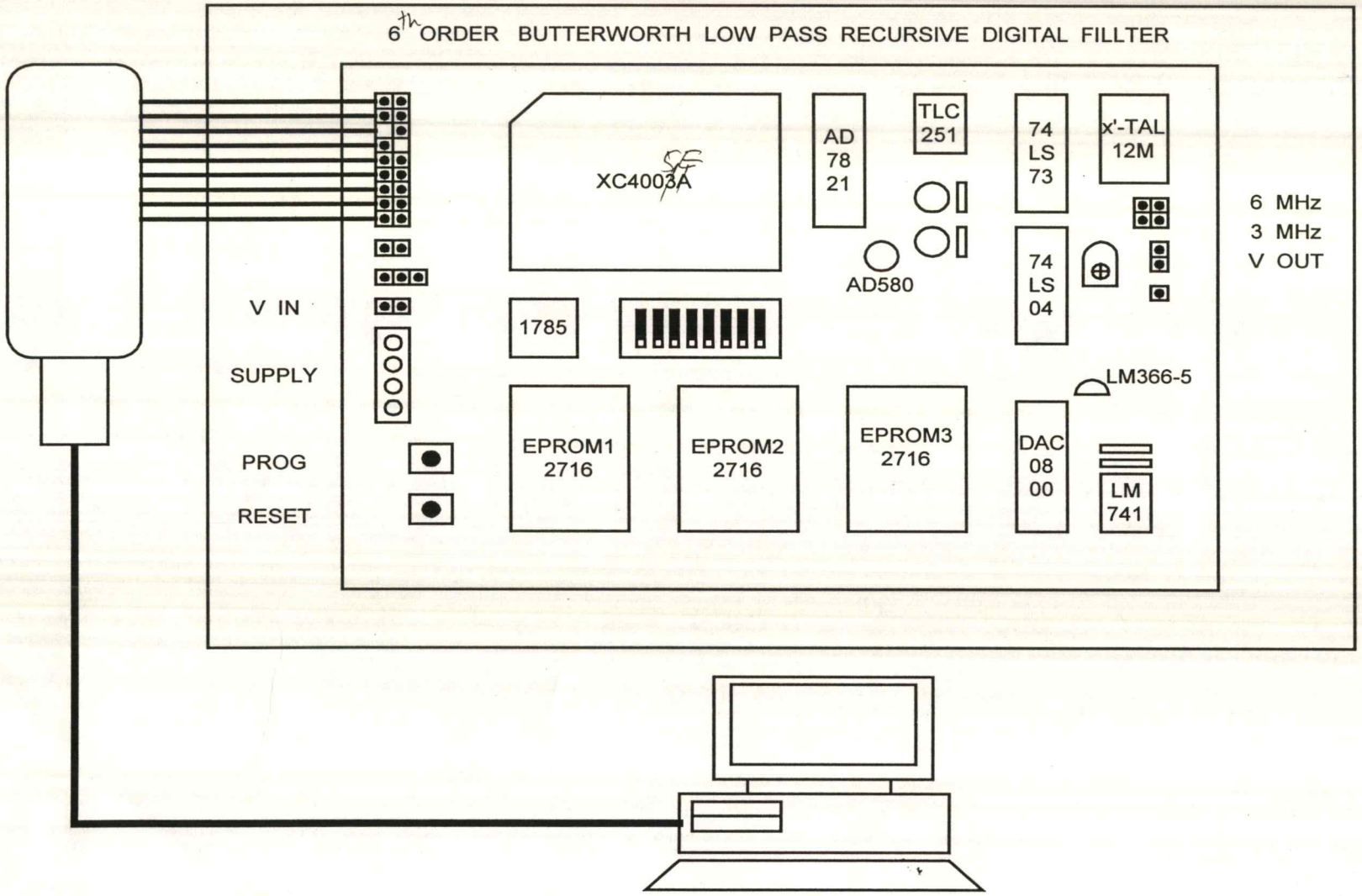
รูปที่ 4.3 ผลการจำลองการทำงานของส่วนประมวลผล

4.1.4 รวมการทำงานส่วนควบคุม, ส่วนเลื่อนข้อมูล, และส่วนประมวลผล

หลังจากรวมการทำงานทั้งหมดแล้ว ทำการป้อนสัญญาณจำลองการทำงาน ซึ่งได้ผลการจำลองการทำงานรวมทั้งหมดดังรูปที่ 4.4



รูปที่ 4.4 การจำลองการทำงานรวมทั้งหมด

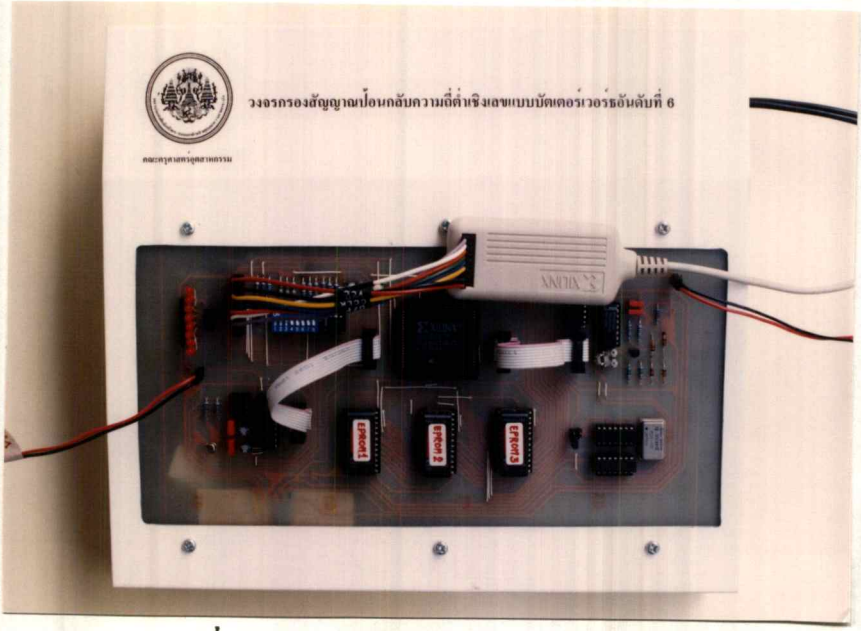


รูปที่ 4.5 การเชื่อมต่อกันระหว่างคอมพิวเตอร์กับบอร์ดตัวอย่างของ FPGAs

4.2 การดาวน์โหลดโปรแกรมลงบนบอร์ดตัวอย่างของ FPGAs

บล็อกไดอะแกรมของบอร์ดตัวอย่างของ FPGAs แสดงดังรูปที่ 4.5 ใช้ไอซี FPGAs เบอร์ XC4005E ควบคุมการทำงาน โดยสามารถใช้งานเป็นวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลขอันดับ 6 ซึ่งบอร์ดตัวอย่างของ FPGAs ของจริงแสดงดังรูปที่ 4.6 และ 4.7

รูปที่ 4.6 ภาพถ่ายบอร์ดตัวอย่างของ FPGAs



รูปที่ 4.7 การทดสอบบอร์ดตัวอย่างของ FPGAs

เมื่อดำเนินการออกแบบตามขั้นตอนในบทที่ 3 จนได้ไฟล์ filter.bit ของวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลขอันดับ 6 จึงทำการทดสอบการทำงานของโปรแกรมและบอร์ดตัวอย่างของ FPGAs โดยการดาวน์โหลดไฟล์ filter.bit ลงบนบอร์ดตัวอย่างของ FPGAs เมื่อต่อสายดาวน์โหลดเคเบิลจากพอร์ท COM2 เข้ากับบอร์ดตัวอย่างของ FPGAs ดังแสดงในรูปที่ 4.7 แล้ว สามารถทำการดาวน์โหลดไฟล์บิทสตรีมได้ด้วยขั้นตอนต่อไปนี้

ขั้นตอนที่ 1 เข้าสู่โปรแกรม XDM แล้วคลิกเมาส์ที่เมนู Tools และเลือกที่คำสั่ง Hardware Debugger ซึ่งจะแสดงหน้าต่าง Communication Setup ดังรูปที่ 2.59 ในบทที่ 2

ขั้นตอนที่ 2 กำหนดค่าอัตราความเร็วในการส่งข้อมูล (Baud Rate) และเลือกพอร์ทที่จะส่งข้อมูลออกไป ในโครงงานนี้เลือกอัตราความเร็วในการส่งข้อมูลที่ 9600 Bit/Sec และเลือกส่งข้อมูลออกที่พอร์ท COM2

ขั้นตอนที่ 3 ตั้งดิพสวิตช์บอร์ดตัวอย่างของ FPGAs ให้ใช้งานลักษณะสเลฟซีเรียลซึ่งสามารถตั้งดิพสวิตช์ได้ดังนี้คือ

ตารางที่ 4.1 การตั้งดิพสวิตช์ต่างๆ ในบอร์ดตัวอย่างของ FPGAs

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

ขั้นตอนที่ 4 เมื่อกำหนดอัตราความเร็วในการส่งข้อมูล และเลือกพอร์ทที่จะส่งข้อมูลออกไปแล้วให้ตอบตกลง จะกลับเข้าสู่หน้าต่าง Hardware Debugger ให้เลือกที่เมนู Download และเลือกที่ Download Design ซึ่งโปรแกรม Debugger จะทำการดาวน์โหลดไฟล์ filter.bit ลงสู่บอร์ดตัวอย่างของ FPGAs เมื่อทำการดาวน์โหลดเสร็จจึงนำบอร์ดตัวอย่างของ FPGAs ไปทดลองได้

4.3 ผลการทดลอง

การทดลองทำได้โดยการป้อนค่าความถี่ต่างๆ เข้าไปและสมมุติค่า Vin ด้วย จากนั้นจึงทำการอ่านค่า $V_{o(p-p)}$ และทำการคำนวณค่า Normalized value ได้ดังตารางที่ 4.1

ตารางที่ 4.1 ผลการทดลอง

f_{in}(Hz)	$V_{o(p-p)}$(Volts)	Normalized value	f_{in}(Hz)	$V_{o(p-p)}$(Volts)	Normalized value
100			2100		
200			2200		
300			2300		
400			2400		
500			2500		
600			2600		
700			2700		
800			2800		
900			2900		
1000			3000		
1100			3100		
1200			3200		
1300			3300		
1400			3400		
1500			3500		
1600			4000		
1700			4500		
1800			5000		
1900			5500		
2000			6000		

ตารางที่ 4.2 (ต่อ) ผลการทดลอง

f_{in}(Hz)	$V_{0(p-p)}$(Volts)	Normalized value	f_{in}(Hz)	$V_{0(p-p)}$(Volts)	Normalized value
6500			12000		
7000			12500		
7500			13000		
8000			13500		
8500			14000		
9000			15000		
9500			16000		
10000			17000		
10500			18000		
11000			19000		
11500			20000		

4.4 การทดลองโดยการใช้ Serial PROM

หลังจากทำการโปรแกรมลง Serial PROM แล้วทำการทดลองโดยใช้ Serial PROM ซึ่งได้ผลการทดลองดังตารางที่ 4.3

ตารางที่ 4.3 ผลการทดลองโดยการใช้ Serial PROM

f_{in}(Hz)	$V_{0(p-p)}$(Volts)	Normalized value	f_{in}(Hz)	$V_{0(p-p)}$(Volts)	Normalized value
100			2500		
200			2600		
300			2700		
400			2800		
500			2900		
600			3000		
700			3100		
800			3200		
900			3300		
1000			3400		
1100			3500		
1200			4000		
1300			4500		
1400			5000		
1500			5500		
1600			6000		
1700			6500		
1800			7000		
1900			7500		
2000			8000		
2100			8500		
2200			9000		
2300			9500		
2400			10000		

ตารางที่ 4.3 (ต่อ) ผลการทดลองโดยการใช้ Serial PROM

f_{in}(Hz)	$V_{0(p-p)}$(Volts)	Normalized value	f_{in}(Hz)	$V_{0(p-p)}$(Volts)	Normalized value
10500			14000		
11000			15000		
11500			16000		
12000			17000		
12500			18000		
13000			19000		
13500			20000		

รูปที่ 4.8 ผลการทดลอง

รูปที่ 4.9 ผลการทดลอง

บทที่ 5
สรุป ปัญหา แนวทางแก้ไขและพัฒนา

5.1 สรุป

วงจรกรองสัญญาณเชิงเลขที่ได้จากการออกแบบด้วยการเขียนภาษา VHDL และการสร้างด้วยอุปกรณ์ FPGAs นั้น ลักษณะวิธีการออกแบบ ซึ่งใช้วิธีการของเลขคณิตศาสตร์การกระจาย ทำให้การออกแบบด้วยการเขียนโปรแกรมภาษา VHDL ทำได้เพียงบรรยายการทำงานของวงจรกรองสัญญาณเชิงเลข และทำการจำลองขั้นตอนการทำงานได้เท่านั้นไม่สามารถที่จะดูผลการทดลองจากการออกแบบได้ วิธีการที่จะดูผลการทดลองได้ต้องทำการสังเคราะห์โปรแกรมที่ได้ให้เป็นวงจรระดับเกท แล้วทำการแปลงไฟล์ที่สังเคราะห์ได้ให้เป็นไฟล์ บิตสตรีม โดยใช้ซอร์ฟแวร์ XACT STEP เวอร์ชั่น 6.01 เพื่อทำการดาวน์โหลด ลงบนบอร์ดตัวอย่างของ FPGAs ผ่านทางสายดาวน์โหลด

การทดสอบการทำงานของวงจรกรอง โดยการดาวน์โหลดโปรแกรมลงอุปกรณ์ FPGAs แล้วทำการป้อนอินพุทความถี่ต่าง ๆ กัน ได้ผลการทดลองที่ใกล้เคียงกับการออกแบบแต่มีข้อผิดพลาดที่ความถี่สูงกว่าความถี่คัตออฟความชันของระบบ จะน้อยกว่าที่ได้ทำการออกแบบไว้ ซึ่งความถี่คัตออฟนั้น สามารถเปลี่ยนแปลงค่าได้โดยการเปลี่ยนค่าสัมประสิทธิ์ของตัวกรองสัญญาณเชิงเลขที่เก็บไว้ใน EPROM ซึ่งในการทดลองนี้ ได้ตั้งความถี่คัตออฟไว้ที่ 10 KHz

5.2 ปัญหาและแนวทางแก้ไข

ในการจัดทำโครงงานนี้สามารถสรุปปัญหาที่เกิดขึ้นระหว่างทำโครงงานนี้พร้อมกับแนวทางในการแก้ไขดังนี้

1. การใช้งานโปรแกรม ซึ่งช่วงแรกของการทำโครงงาน โปรแกรมสำหรับใช้งานมีไม่เพียงพอ เนื่องจากมีหลายกลุ่มที่ทำโครงงานลักษณะเดียวกันนี้

แนวทางแก้ไข ทำการจัดช่วงเวลาในการใช้งานโปรแกรมของแต่ละกลุ่ม เพื่อให้ทุกกลุ่มได้ใช้โปรแกรมเท่าๆ กัน

2. ปัญหาเกี่ยวกับโปรแกรมเกี่ยวกับการเซตระบบของโปรแกรม เพื่อใช้งานซึ่งค่อนข้างเซตยากทำให้การใช้งานมีปัญหาทำงานได้ไม่เต็มที่

แนวทางแก้ไข ให้อาจารย์ที่ปรึกษาช่วยเซตระบบการทำงานของโปรแกรมให้

3. ปัญหาการลงโปรแกรม FOUNDATION Siries ของบริษัท Xilinx ซึ่งเป็นโปรแกรมตัวใหม่ ที่นำมาใช้งานแทนโปรแกรมเดิม ไม่สามารถ Authorization ได้

แนวทางแก้ไข ต้องเข้าไปเรียกใช้โปรแกรม Xkey ที่ทำงานบน ระบบ DOS เพื่อทำการ Authorization โปรแกรม โดยใช้คำสั่ง Xkey - W หลังจากนั้นให้ทำการใส่หมายเลข Serial Number และ รหัสผ่านตามที่เครื่องกำหนด

4. ไม่สามารถทำการสังเคราะห์โปรแกรมที่เขียนขึ้นให้เป็นวงจรระดับเกทได้ เนื่องจากมีความซับซ้อนเกินไป

แนวทางแก้ไข เปลี่ยนรูปแบบในการเขียนโปรแกรม จากเดิมที่เขียนเป็นวงจรรวมทั้งหมด เพียงวงจรเดียว ก็ทำการแบ่งออกเป็นบล็อกการทำงาน 3 ส่วน เขียนเป็นอุปกรณ์แต่ละตัวเพื่อนำมาเชื่อมต่อกันภายหลัง

5. ปัญหาในการรวมโปรแกรมย่อยๆ ให้เป็นโปรแกรมตัวใหญ่ตัวเดียวเมื่อรวมแล้ว โปรแกรมตัวใหญ่ไม่สามารถมองเห็นโปรแกรมย่อยที่อยู่ภายในได้

แนวทางการแก้ไข ทำการทดลองเขียนโปรแกรม ทดสอบที่มีขนาดเล็ก ๆ โดยการ Add Library ของโปรแกรมย่อย พร้อมกับคัดลอกไฟล์โปรแกรมย่อยเข้ามาเก็บมาไว้โปรเจคเดียวกัน แล้วทำการสังเคราะห์โปรแกรมรวมพร้อมกัน

6. ปัญหาในการทดสอบบอร์ด FPGAs ต้องใช้แหล่งจ่ายไฟจำนวน 4 ตัว ทำให้เกิดความยุ่งยากในการต่อวงจรใช้งาน

แนวทางแก้ไข ใช้สวิทชิ่งเพาเวอร์ซัพพลาย จากเครื่องคอมพิวเตอร์ ซึ่งมีแรงดันครบตามที่ต้องการ

7. ปัญหาในการกำหนดขาอุปกรณ์ FPGAs ในการใช้งานซึ่งไม่สามารถกำหนดขาตามที่ได้ทำการออกแบบไว้

แนวทางแก้ไข การแก้ปัญหาในครั้งแรกคือให้ใส่ PAD อิน์พุทหรือ PAD เอาท์พุทเข้าไปในส่วนของอินพุทหรือเอาท์พุทของรูปวงจรที่ได้จากการสร้าง Schematic แล้วทำการกำหนดขาที่ต้องการลงบน PAD นั้น แต่ในโปรแกรม Foundation Serial ไม่สามารถนำวงจร

ดังกล่าวมาทำเป็นขั้นตอนของการ Implement ได้ จึงทำการแก้ปัญหาโดยการเขียนโปรแกรมเพื่อกำหนดขาโดยตั้งชื่อให้ตรงกับชื่อโปรเจ็กที่สร้างไว้ โดยให้มีนามสกุลเป็น *.cst

8. ปัญหาในการกำหนดขาด้วยการเขียนโปรแกรม *.cst คือการกำหนดขาที่เป็นอินพุทหรือเอาท์พุทที่เป็นลักษณะของเวกเตอร์ไม่สามารถที่จะกำหนดได้ เนื่องจากไม่ทราบรูปแบบการเขียนวิธีดังกล่าว

แนวทางแก้ไข ทำการเขียนโปรแกรมตัวอย่างที่ใช้อินพุทหรือเอาท์พุทที่มีลักษณะเป็นเวกเตอร์ แล้วทดลองใช้ Floor Planner ในการกำหนดขาเอง ซึ่งหลังจากที่ใช้โปรแกรม Floor Planner กำหนดขาแล้วโปรแกรมจะทำการสร้างไฟล์ที่มีนามสกุล *.cst ให้อัตโนมัติแล้วใช้โปรแกรมเวอร์ด เพื่อเปิดไฟล์ดังกล่าว ดูรูปแบบในการเขียนโปรแกรมกำหนดอินพุทหรือเอาท์พุทที่มีลักษณะเป็นเวคเตอร์

5.3 แนวทางการพัฒนา

1. ในโครงงานนี้ การเก็บค่าสัมประสิทธิ์ของตัวกรองสัญญาณเชิงเลข จะใช้หน่วยความจำภายนอก เพื่อเก็บค่าไว้ แต่เราสามารถที่จะพัฒนาใช้หน่วยความจำที่มีอยู่ภายในตัวอุปกรณ์ FPGAs ได้

2. สามารถลดจำนวนเกตในการสังเคราะห์โปรแกรมลงได้ โดยการเรียกใช้ฟังก์ชั่นการทำงานของภาษา VHDL

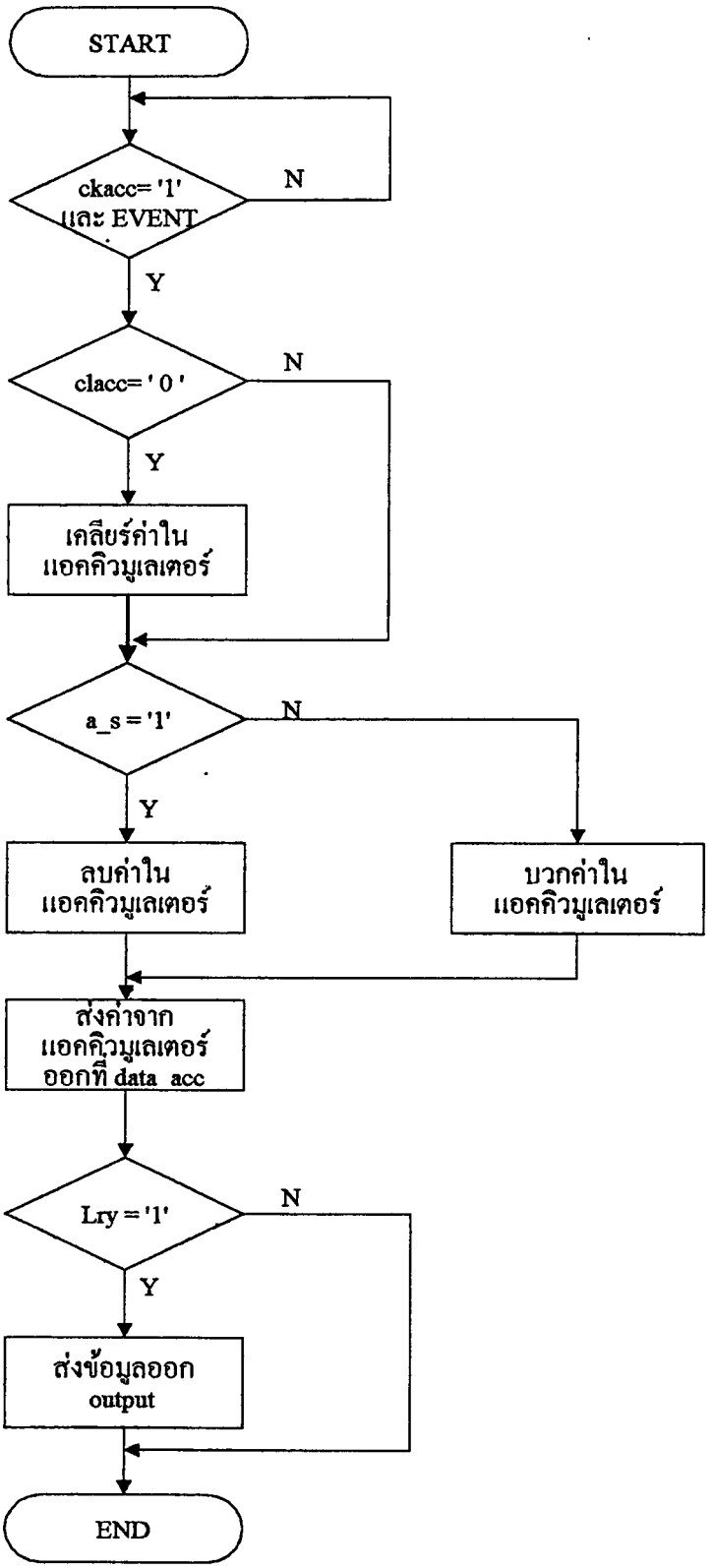
3. การเก็บค่าสัมประสิทธิ์ของตัวกรองสัญญาณเชิงเลข สามารถที่จะเก็บไว้ภายในหน่วยความจำภายนอกตัวเดียวกันได้แต่ต้องทำการออกแบบชุดถอดรหัสตำแหน่งหน่วยความจำใหม่

4. สามารถใช้ภาษา VHDL เขียนบรรยายการทำงานของวงจรเชิงเลขได้ทุกวงจร เช่น วงจรเลื่อนข้อมูล , วงจรเข้ารหัส , วงจรนับ เป็นต้น

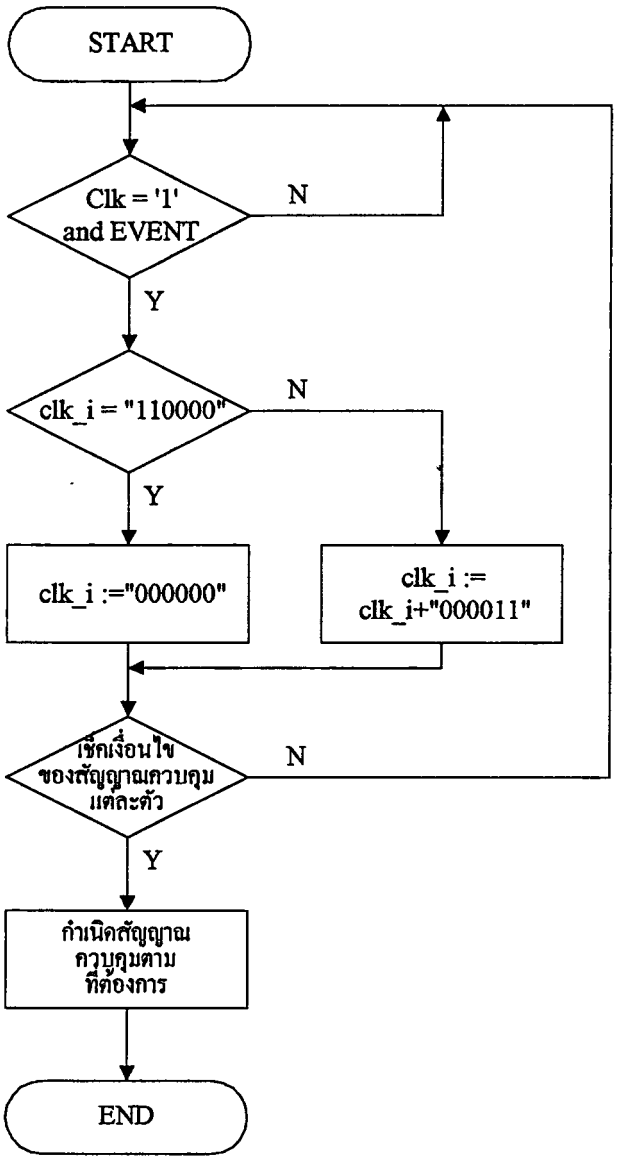
5) สามารถออกแบบวงจรเชิงเลขเฉพาะงานอื่น ๆ ตามที่ผู้ใช้ต้องการ โดยใช้ภาษา VHDL และ อุปกรณ์ FPGAs ได้

ภาคผนวก ก

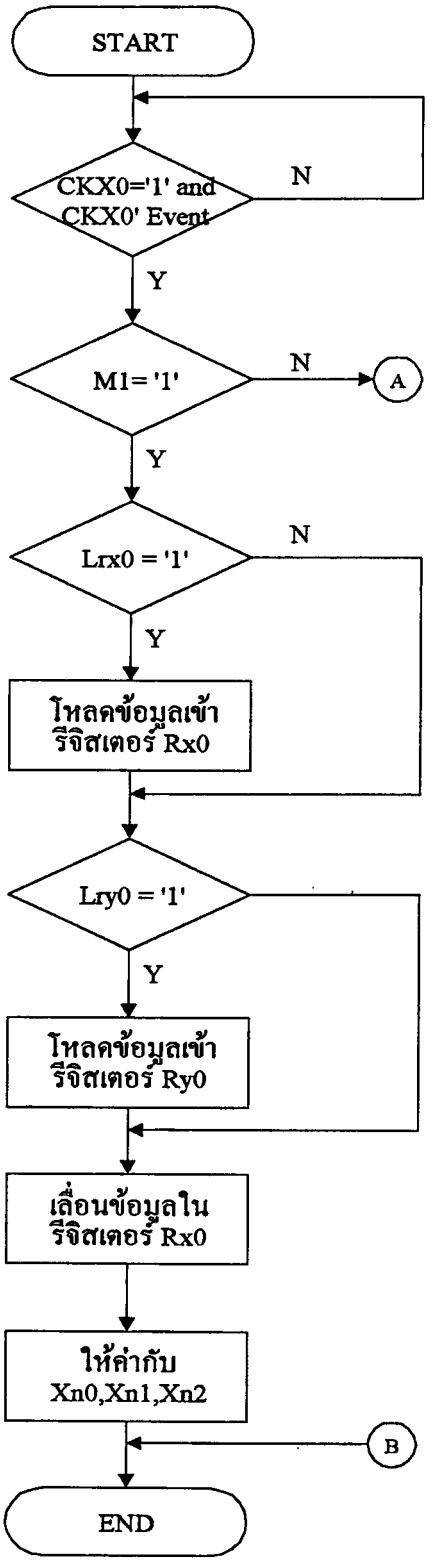
ผังงานและโปรแกรมภาษา VHDL ของวงจรกรองสัญญาณป้อนกลับความถี่ต่ำเชิงเลข แบบบัตเตอร์เวอร์ธอันดับที่ 6 โดยใช้ภาษา VHDL และอุปกรณ์ FPGAs



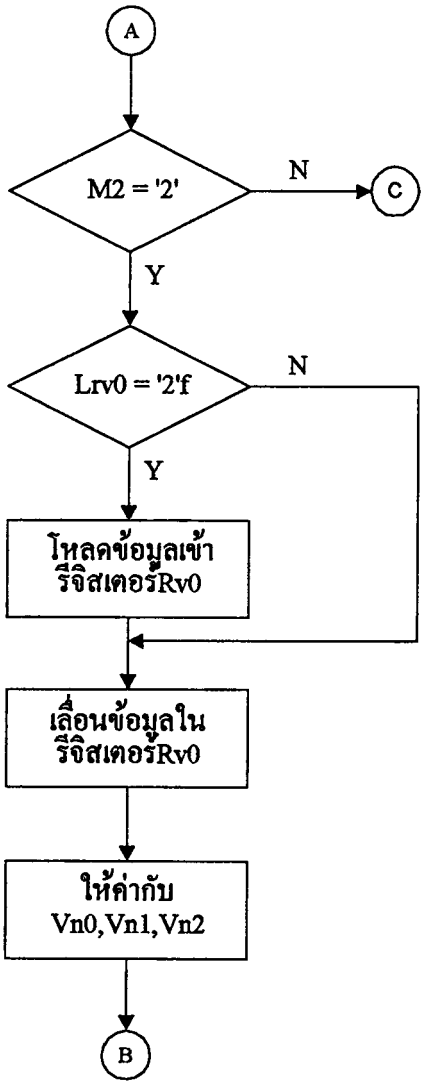
รูปที่ ก.1 ผังงานของโปรแกรมส่วนประมวลผล



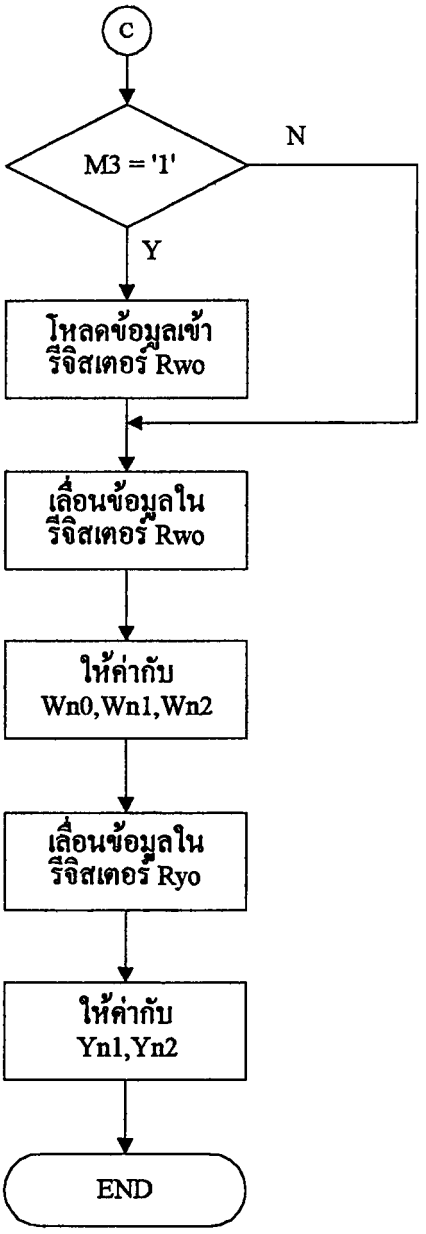
รูปที่ ก.2 ผังงานของโปรแกรมส่วนควบคุม



รูปที่ ก.3 ผังงานของโปรแกรมส่วนเลื่อนข้อมูล



รูปที่ ก.3 (ต่อ) ผังงานของโปรแกรมส่วนเลื่อนข้อมูล



รูปที่ ก.3 (ต่อ) ผังงานของโปรแกรมส่วนเลื่อนข้อมูล

โปรแกรมที่ ก.1 ส่วนประมวลผล

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY acc IS
  PORT (
        ckacc          :IN STD_LOGIC;
        clacc          :IN STD_LOGIC;
        rom_data       :IN STD_LOGIC_VECTOR (7  downto 0);
        a_s            :IN STD_LOGIC;
        cky            :IN STD_LOGIC;
        data_acc       :BUFFER STD_LOGIC_VECTOR (7 downto 0);
        data_out       :BUFFER STD_LOGIC_VECTOR (7 downto 0)
  );
END ACC;

ARCHITECTURE acc_arch of acc is
BEGIN
        PROCESS(ckacc,clacc,rom_data,a_s,cky)
        variable base,oper,result : std_logic_vector(7 downto 0):="00000000";
        variable carry : std_logic;
        BEGIN

                IF(ckacc='1' and ckacc'EVENT ) THEN
                        IF(clacc = '0' ) THEN
                                result:="00000000";
                                base:="00000000";
                        END IF;
                                base:=result;
                                oper:=rom_data;
                                IF (a_s='1') THEN
```

```
                    carry:='1';
                ELSE carry:='0';
                END IF;
                FOR lp in 0 to 7 LOOP
                    IF(a_s='1')THEN
                        oper(lp):=not oper(lp);
                    END IF;
            result(lp) := (base(lp) xor oper(lp)) xor carry;
            carry:=(base(lp) and oper(lp)) or (carry and (base(lp) or oper(lp)));
                END LOOP;
                data_acc<=result;
            IF(cky='1' ) THEN
                data_out<=result;
            END IF;
        END IF;
    END PROCESS;
END acc_arch;
```

โปรแกรมที่ ก.2 ส่วนควบคุม

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
LIBRARY SYNOPSYS;
USE SYSNOPSYS.STD_LOGIC_UNSIGNED.ALL;

ENTITY control IS
  PORT (
        clk: IN STD_LOGIC;
        clacc: OUT STD_LOGIC;
        a_s: OUT STD_LOGIC;
        ckx0: OUT STD_LOGIC;
        lrx0: OUT STD_LOGIC;
        lrv0: OUT STD_LOGIC;
        lrw0: OUT STD_LOGIC;
        lry0 : OUT STD_LOGIC;
        cky0: OUT STD_LOGIC;
        m1: OUT STD_LOGIC;
        m2: OUT STD_LOGIC;
        m3: OUT STD_LOGIC;
        sc: OUT STD_LOGIC;
        ckacc :OUT STD_LOGIC
  );
END control;

ARCHITECTURE control_arch OF control IS
  BEGIN
        PROCESS(clk)
        VARIABLE ckx : std_logic_vector(1 downto 0):="00";
        VARIABLE clk_i : std_logic_vector(5 downto 0):="000000";
        BEGIN
```

```
IF(clk ='1' and clk'EVENT ) THEN
----------clock x----------
        IF(clk_i="110000")THEN
                clk_i:="000000";
        END IF;
        clk_i:=clk_i+"000001";
        ckx:=ckx+"01";
        IF(ckx<="01") THEN
                ckx0<='0';
                ckacc<='0';
        ELSIF(ckx>"01")THEN
                ckx0<= '1';
                ckacc<='1' ;
                ckx:="00";
        END IF;
---------- clock lrx ----------
        IF(clk_i="000010")THEN
                lrx0<='1';
        ELSE
                lrx0<='0';
        END IF;
----------clock y --------
        IF(clk_i="100010" or clk_i= "100100" or clk_i="100110"
         or clk_i="101000" or clk_i="101010" or clk_i="101100"
         or clk_i="101110" or clk_i="110000" )THEN
                cky0<='1';
        ELSE    cky0<='0';
        END IF;

---------- clock lrv----------
        IF(clk_i="010010")THEN
                lrv0<='1';
```

```
        ELSE
                lrv0<='0';
        END IF;
---------- clock lrw ----------
        IF(clk_i="100010")THEN
                lrw0<='1';
                lry0<='1';
        ELSE
                lrw0<='0';
                lry0<='0';
        END IF;
----------clock add sub --------
        IF(clk_i="001111" or clk_i="010000" or clk_i="011111"
         or clk_i="100000" or clk_i="101111" or clk_i="110000")THEN
                a_s<='1';
        ELSE
                a_s<='0';
        END IF;

----------clock clear acc--------
        IF(clk_i = "010010" or clk_i="100010" or clk_i="000010")THEN
                clacc<='0';
        ELSE
                clacc<='1';
        END IF;

----------clock m1----------
        IF(clk_i>="000001" and clk_i<="010000")THEN
                m1<='1';
        ELSE
                m1<='0';
        END IF;
```

```
            ----------clock m2----------
                IF(clk_i>="010001" and clk_i<="100000")THEN
                    m2<='1';
                ELSE
                    m2<='0';
                END IF;
            ---------- clock m3 ----------
                IF(clk_i>="100001" and clk_i<="110000")THEN
                    m3<='1';
                ELSE    m3<='0';
                END IF;
            ----------clock sc----------
                IF(clk_i="000001")THEN
                    sc<='0';
                ELSE
                    sc<='1';
                END IF;
            END IF;
        END PROCESS;
END control_arch;
```

โปรแกรมที่ ก.3 ส่วนเลื่อนข้อมูล

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY shift_data IS
  PORT (
        data_in         :IN STD_LOGIC_VECTOR (7 downto 0);
        ckx0            :IN STD_LOGIC;
        lrx0            :IN STD_LOGIC;
        lrv0            :IN STD_LOGIC;
        m1              :IN STD_LOGIC;
        lrw0:            IN STD_LOGIC;
        cky0            :IN STD_LOGIC;
        m2              :IN STD_LOGIC;
        m3              :IN STD_LOGIC;
        lry0            :IN STD_LOGIC;
        data_Acc        :IN STD_LOGIC_VECTOR (7 downto 0);
        Xn0             : BUFFER STD_LOGIC;
        Xn1             : BUFFER STD_LOGIC;
        Xn2             : BUFFER STD_LOGIC;
        Vn0             : BUFFER STD_LOGIC;
        Vn1             : BUFFER STD_LOGIC;
        Vn2             : BUFFER STD_LOGIC;
        wn0             : BUFFER STD_LOGIC;
        wn1             : BUFFER STD_LOGIC;
        wn2              BUFFER_LOGIC;
        yn1             : BUFFER STD_LOGIC;
        Yn2             : BUFFER STD_LOGIC );
END shift_data;
```

```
ARCHITECTURE shift_data_arch OF shift_data IS
BEGIN
  PROCESS(m1,m2,m3,ckx0,cky0,lrx0,lrv0,lrw0,lry0)
  VARIABLE  rx0,rv0,rw0,ry0 : STD_LOGIC_VECTOR(10 downto 0):="00000000000";
       BEGIN
            IF(ckx0='1' and ckx0'EVENT) THEN
                 IF (m1='1') THEN
                      IF (lrx0='1') THEN
                           rx0(7 downto 0) := data_in;
                      END IF;
                 FOR lpx in 10 downto 0 LOOP
                      IF lpx=0 THEN
                           rx0(0):='0';
                      ELSE rx0(lpx):=rx0(lpx-1);
                      END IF;
                 END LOOP;
                 xn0<=rx0(8);
                 xn1<=rx0(9);
                 xn2<=rx0(10);
       ELSIF (m2='1' ) THEN
            IF (lrv0='1')THEN
                 rv0(7 downto 0):=data_acc;
            END IF;
                 FOR lpv in 10 downto 0 LOOP
                      IF lpv=0 THEN
                           rv0(0):='0';
                      ELSE rv0(lpv):=rv0(lpv-1);
                      END IF;
                 END LOOP;
                 vn0<=rv0(8);
                 vn1<=rv0(9);
                 vn2<=rv0(10);
```

```
        ELSIF m3='1' THEN
                IF (lrw0='1')THEN
                        rw0(7 downto 0):=data_acc;
                END IF;
                FOR lpw in 10 downto 0 LOOP
                        IF lpw=0 THEN
                                rw0(0):='0';
                        ELSE rw0(lpw) :=rw0(lpw-1);
                        END IF;
                END LOOP;
                        wn0<=rw0(8);
                        wn1<=rw0(9);
                        wn2<=rw0(10);
                IF(cky0 ='1')THEN
                        IF(lry0='1') THEN
                                ry0(7 downto 0):=data_acc;
                        END IF;
                        FOR lpy in 10 downto 0 LOOP
                                IF lpy=0 THEN
                                        ry0(0):='0';
                                ELSE ry0(lpy):=ry0(lpy-1);
                                END IF;
                        END LOOP;
                        yn2<=ry0(9);
                        yn1<=ry0(8);
                END IF;
        END IF;
END IF;
END PROCESS;
END shif_data_arch;
```

โปรแกรมที่ ก.4 วงจรกรอง (filter)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY filter IS
  PORT (
        SIGNAL clk          : IN STD_LOGIC;
        SIGNAL input        : IN STD_LOGIC_VECTOR (7 downto 0);
        SIGNAL data_rom     : IN STD_LOGIC_VECTOR (7 downto 0);
        SIGNAL data_out     : BUFFER STD_LOGIC_VECTOR (7 downto 0);
        SIGNAL xn           : OUT STD_LOGIC;
        SIGNAL xn1          : OUT STD_LOGIC;
        SIGNAL xn2          : OUT STD_LOGIC;
        SIGNAL vn           : OUT STD_LOGIC;
        SIGNAL vn1          : OUT STD_LOGIC;
        SIGNAL vn2          : OUT STD_LOGIC;
        SIGNAL wn           : OUT STD_LOGIC;
        SIGNAL wn1          : OUT STD_LOGIC;
        SIGNAL wn2          : OUT STD_LOGIC;
        SIGNAL yn1          : OUT STD_LOGIC;
        SIGNAL yn2          : OUT TD_LOGIC;
        SIGNAL m1           : BUFFER STD_LOGIC;
        SIGNAL m2           : BUFFER STD_LOGIC;
        SIGNAL m3           : BUFFER STD_LOGIC;
        SIGNAL s_h          : OUT STD_LOGIC;
  );
END filter;
```

```
ARCHITECTURE new_filter_arch OF filter IS

                --------- component acc ----------
COMPONENT acc
  PORT (
        SIGNAL ckacc          :IN STD_LOGIC;
        SIGNAL clacc          :IN STD_LOGIC;
        SIGNAL rom_data       :INT STD_LOGIC_VECTOR (7  downto 0);
        SIGNAL a_s            :IN STD_LOGIC;
        SIGNAL lry0           :IN STD_LOGIC;
        SIGNAL data_acc       :OUT STD_LOGIC_VECTOR (7 downto 0);
        SIGNAL data_out       :BUFFER STD_LOGIC_VECTOR (7 downto 0));
END COMPONENT;
                --------- component control ---------
COMPONENT control
  PORT (
        SIGNAL clk            :IN STD_LOGIC;
        SIGNAL clacc          :OUT STD_LOGIC;
        SIGNAL a_s            :OUT STD_LOGIC;
        SIGNAL ckx0           :OUT STD_LOGIC;
        SIGNAL lrx0           :OUT STD_LOGIC;
        SIGNAL  lrv0          :OUT STD_LOGIC;
        SIGNAL lrw0           :OUT STD_LOGIC;
        SIGNAL lry0           :OUT STD_LOGIC;
        SIGNAL cky0           :OUT STD_LOGIC;
        SIGNAL m1             :BUFFER STD_LOGIC;
        SIGNAL m2             :BUFFER STD_LOGIC;
        SIGNAL m3             :BUFFER STD_LOGIC;
        SIGNAL  sc            :OUT STD_LOGIC;
        SIGNAL ckacc          :OUT STD_LOGIC;
  );
END COMPONENT;
```

```
----------component shift_data--------
COMPONENT shift_data
  PORT (
        SIGNAL data_in              :IN STD_LOGIC_VECTOR (7 downto 0);
        SIGNAL ckx0          :IN STD_LOGIC;
        SIGNAL lrx0          :IN STD_LOGIC;
        SIGNAL lrv0          :IN STD_LOGIC;
        SIGNAL m1            :IN STD_LOGIC;
        SIGNAL lrw0          :IN STD_LOGIC;
        SIGNAL m2            :IN STD_LOGIC;
        SIGNAL m3            :IN STD_LOGIC;
        SIGNAL lry0          :IN STD_LOGIC;
        SIGNAL data_buff     :IN STD_LOGIC_VECTOR ( 7 downto 0 );
        SIGNAL data_Acc      :IN STD_LOGIC_VECTOR (7 downto 0);
        SIGNAL Xn0           :OUT STD_LOGIC;
        SIGNAL Xn1           :OUT STD_LOGIC ;
        SIGNAL Xn2           :OUT STD_LOGIC;
        SIGNAL Vn0           :OUT STD_LOGIC ;
        SIGNAL Vn1           :OUT STD_LOGIC;
        SIGNAL Vn2           :OUT STD_LOGIC ;
        SIGNAL wn0           :OUT STD_LOGIC ;
        SIGNAL wn1           :OUT STD_LOGIC;
        SIGNAL wn2           :OUT STD_LOGIC;
        SIGNAL Yn1           :OUT STD_LOGIC;
        SIGNAL Yn2           :OUT STD_LOGIC
);
END COMPONENT;
                ---------- end component---------
SIGNAL ckacc_s : STD_LOGIC
SIGNAL ckx0_s : STD_LOGIC;
SIGNAL clacc_s : STD_LOGIC;
SIGNAL as_s : STD_LOGIC;
```

```
SIGNAL cky0_s : STD_LOGIC;
SIGNAL lrx0_s : STD_LOGIC;
SIGNAL lrv0_s : STD_LOGIC;
SIGNAL lrw0_s : STD_LOGIC;
SIGNAL lry0_s : STD_LOGIC;
SIGNAL sc_s : STD_LOGIC;
SIGNAL data_acc_s : STD_LOGIC_VECTOR(7 downto 0);
BEGIN
        U1: acc3
        PORT MAP    (ckacc_s,clacc_s,data_rom,as_s,lry0_s,data_acc_s,data_out);
        U2: con3
        PORT map    (clk,clacc_s,as_s,ckx0_s,lrx0_s,lrv0_s,lrw0_s,lry0_s,cky0_s,m1,m2,
                    m3,s_h,ckacc_s);
        U3: sh3
        PORT MAP
                    (input,ckx0_s,lrx0_s,lrv0_s,m1,lrw0_s,cky0_s,m2,m3,lry_s,data_out,
                    data_acc_s,xn,xn1,xn2,vn,vn1,vn2,wn,wn1,wn2,yn1,yn2);
END new_filter_arch;
```

ภาคผนวก ข

รายละเอียดข้อมูลและคุณสมบัติของอุปกรณ์

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

DAC-08

HIGH SPEED 8-BIT MULTIPLYING D-TO-A CONVERTER

The DAC-08 series is a monolithic 8-bit high speed multiplying digital-to-analog converter, capable of settling to within 1/2 LSB (0.19%) in 85 ns. Monotonic multiplying performance is retained over a wide 40-to-1 reference current range. Full scale and reference currents are matched to within 1 LSB, therefore eliminating the need for full scale trim in most applications.

Dual complementary current outputs with high voltage compliance provide added versatility and allow differential mode of operation to effectively double the peak-to-peak output swing. In many applications, output current-to-voltage conversion can be accomplished without requiring an external op amp. Noise-immune inputs permit direct interface with TTL and DTL levels when the logic threshold control, V_{LC}, (Pin 1) is grounded. All other logic family thresholds are attainable by adjusting the voltage level of Pin 1. Performance characteristics are virtually unchanged over the entire ±4.5 V to ±18 V power supply range. Power consumption is typically 33 mW with ±5.0 V supplies.

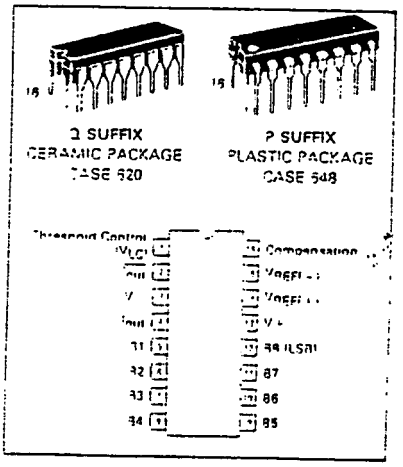
The DAC-08 is available in several versions, with nonlinearity as tight as ±0.1% (±1/4 LSB) over temperature. All versions are guaranteed monotonic over 8 bits. For an extra margin of performance, Motorola utilizes thin-film resistors permitting very accurate resistive values which are extremely stable over temperature.

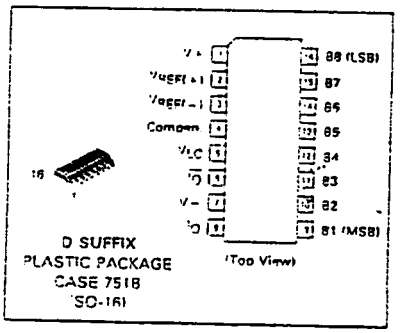
High performance characteristics along with low cost, make the DAC-08 an excellent selection for applications such as CRT displays, waveform generation, high speed modems, and high speed analog to digital converters.

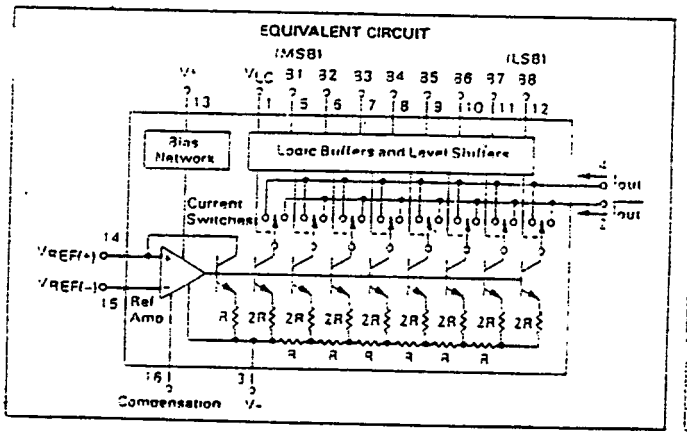
- Fast Settling Time — 85 ns
- Full Scale Current Prematched to ±1 LSB
- Nonlinearity Over Temperature to ±0.1% Max
- Differential Current Outputs
- High Voltage Compliance Outputs −10 V to +18 V
- Wide Range Multiplying Capability
- Inputs Comparable With TTL, DTL, CMOS, PMOS, ECL, HTL
- Low Full Scale Current Drift
- Wide Power Supply Range ±4.5 V to ±18 V
- Low Power Consumption
- Thin-Film Resistors
- Low Cost

HIGH SPEED 8-BIT MULTIPLYING D-TO-A CONVERTER

SILICON MONOLITHIC INTEGRATED CIRCUIT







ORDERING INFORMATION

Device	Nonlinearity	Temperature Range	Package
DAC-08HQ	±0.1%	0°C to +70°C	Ceramic
DAC-08EQ	±0.19%		Ceramic
DAC-08CD	±0.39%		SO-16
DAC-08ED	±0.19%		SO-16
DAC-08HP	±0.1%		Plastic
DAC-08EP	±0.19%		Plastic
DAC-08CP	±0.39%		Plastic

DAC-08

MAXIMUM RATINGS (T_A = 25°C unless otherwise noted)

Rating	Symbol	Value	Unit
V+ Supply to V- Supply	V_S	36	V
Logic Inputs	—	V- to V- Plus 36	V
Logic Threshold Control	V_{LC}	V- to V+	V
Analog Current Outputs	I_{out}	See Figure 7	mA
Reference Input (V14, V15)	V_{ref}	V- to V+	V
Reference Input Differential Voltage (V14 to V15)	$V_{ref(D)}$	±18	V
Reference Input Current (I14)	I_{ref}	5.0	mA
Operating Temperature Range	T_A	0 to +70	°C
Storage Temperature	T_{stg}	−65 to +150	°C
Power Dissipation Derate above 100°C	P_D $R_{\theta JA}$	500 10	mW mW/°C

ELECTRICAL CHARACTERISTICS (V_S = ±15 V, I_{ref} = 2.0 mA, T_A = 0°C to 70°C, unless otherwise noted)

Characteristic	Symbol	DAC-08H			DAC-08E			DAC-08C			Unit
		Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
Resolution	—	8	8	8	8	8	8	8	8	8	Bits
Monotonicity	—	8	8	8	8	8	8	8	8	8	Bits
Nonlinearity, T_A = 0°C to 70°C	NL	—	—	±0.1	—	—	±0.19	—	—	±0.39	%FS
Setting Time to ± 1/2 LSB (All Bits Switched On or Off T_A = 25°C) Figure 24 (Note 1)	t_s	—	85	—	—	85	—	—	85	—	ns
Propagation Delay, T_A = 25°C (Note 1) Each Bit All Bits Switched	t_{PLH} t_{PHL}	— —	35 35	— —	— —	35 35	— —	— —	35 35	— —	ns
Full Scale Tempco	TCIFS	—	±10	—	—	±10	—	—	±10	—	ppm/°C
Output Voltage Compliance Full Scale Current Change < 1/2 LSB, R_{out} > 20 MΩ Typ.	V_{OC}	−10	—	−18	−10	—	+18	−10	—	−18	V
Full Range Current (V_{ref} = 10.000 V; R14, R15 = 5.000 kΩ T_A = 25°C	I_{FR4}	1.984	1.992	2.000	1.94	1.99	2.04	1.94	1.99	2.04	mA
Full Range Symmetry (I_{FR4} − I_{FR2})	I_{FRS}	—	±0.5	±4.0	—	±1.0	±8.0	—	±2.0	±16.0	µA
Zero Scale Current	I_{ZS}	—	0.1	1.0	—	0.2	2.0	—	0.2	4.0	µA
Output Current Range V_- = −5.0 V V_- = −8.0 V to −18 V	I_{OR1} I_{OR2}	0 0	— —	2.1 4.2	0 0	— —	2.1 4.2	0 0	— —	2.1 4.2	mA
Logic Input Levels (V_{LC} = 0 V) Logic "0" Logic "1"	V_{IL} V_{IH}	— 2.0	— —	0.8 —	— 2.0	— —	0.8 —	— 2.0	— —	0.8 —	V
Logic Input Levels (V_{LC} = 0 V) Logic "0" (V_{in} = −10 V to +0.8 V) Logic "1" (V_{in} = +2.0 V to +18 V)	I_{IL} I_{IH}	— —	−2.0 0.002	−10 10	— —	−2.0 0.002	−10 10	— —	−2.0 0.002	−10 10	µA
Logic Input Swing, V_- = −15 V	V_{IS}	−10	—	−18	−10	—	−18	−10	—	−18	V
Logic Threshold Range, V_S = ±15 V	V_{THR}	−10	—	−13.5	−10	—	+13.5	−10	—	−13.5	V
Reference Bias Current	I_{15}	—	−1.0	−3.0	—	−1.0	−3.0	—	−1.0	−3.0	µA
Reference Input Slew Rate Figure 19 (Note 1)	dI/dt	—	8.0	—	—	8.0	—	—	8.0	—	mA/µs
Power Supply Sensitivity I_{ref} = 1.0 mA V_- = −4.5 V to −18 V V_- = −4.5 V to −18 V	$PSSIFS_+$ $PSSIFS_-$	— —	±0.0003 ±0.002	±0.01 ±0.01	— —	±0.0003 ±0.002	±0.01 ±0.01	— —	±0.0003 ±0.002	±0.01 ±0.01	%/%
Power Supply Current V_S = ±5.0 V, I_{ref} = 1.0 mA V_S = ±5.0 V, −15 V, I_{ref} = 2.0 mA V_S = ±15 V, I_{ref} = 2.0 mA	I_+ I_- I_+ I_- I_+ I_-	— — — — — —	2.3 −4.3 2.4 −6.4 2.5 −6.5	3.8 −5.8 3.8 −7.8 3.8 −7.8	— — — — — —	2.3 −4.3 2.4 −6.4 2.5 −6.5	3.8 −25.8 3.8 −7.8 3.8 −7.8	— — — — — —	2.3 −4.3 2.4 −6.4 2.5 −6.5	3.8 −5.8 3.8 −7.8 3.8 −7.8	mA
Power Dissipation V_S = ±5.0 V, I_{ref} = 1.0 mA V_S = ±5.0 V, −15 V, I_{ref} = 2.0 mA V_S = ±15 V, I_{ref} = 2.0 mA	P_D	— — —	33 108 135	48 136 174	— — —	33 108 135	48 136 174	— — —	33 108 135	48 136 174	mA

Note 1. Parameter is not 100% tested; guaranteed by design.

MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

MC1403,A
MC1503

LOW VOLTAGE REFERENCE

A precision band-gap voltage reference designed for critical instrumentation and D/A converter applications. This unit is designed to work with Motorola MC1508 and MC3510 D/A converters, and MC14433 A/D systems. Low temperature drift is a prime design consideration.

- Output Voltage: 2.5 V ±25 mV
- Input Voltage Range: 4.5 V to 40 V
- Quiescent Current: 1.2 mA Typ
- Output Current: 10 mA
- Temperature Coefficient: 10 ppm/°C Typ
- Guaranteed Temperature Drift Specification
- Equivalent to AD580
- Standard 8-Pin DIP, and 8-Pin SOIC Package

Typical Applications

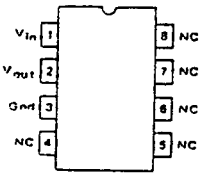
- Voltage Reference for 8–12 Bit D/A Converters
- Low T_C Zener Replacement
- High Stability Current Reference
- Voltmeter System Reference

PRECISION LOW VOLTAGE REFERENCE

LASER TRIMMED INTEGRATED CIRCUIT

U SUFFIX
CERAMIC PACKAGE
CASE 693

D SUFFIX
PLASTIC PACKAGE
CASE 751
(SO-8)



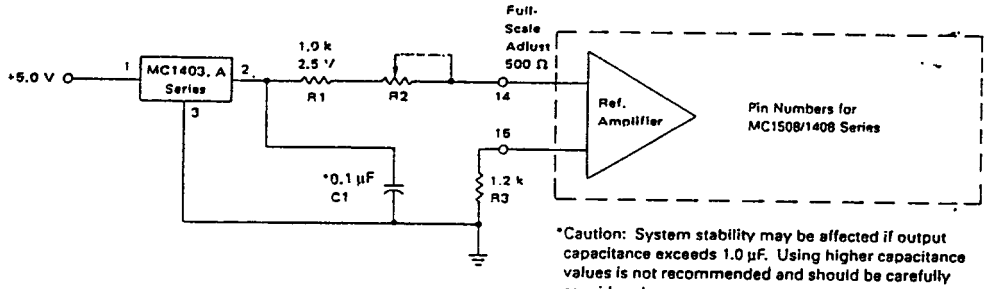
MAXIMUM RATINGS (T_A = 25°C unless otherwise noted.)

Rating	Symbol	Value	Unit
Input Voltage	V_I	40	V
Storage Temperature	T_{stg}	−65 to 150	°C
Junction Temperature	T_J	+175	°C
Operating Ambient Temeprature Range MC1503 MC1403,A	T_A	 −55 to +125 0 to +70	 °C °C

ORDERING INFORMATION

Device	Temperature Range	Package
MC1503U	−55 to +125°C	Ceramic DIP
MC1403D	0 to +70°C	SO-8
MC1403U		Ceramic DIP
MC1403AU		Ceramic DIP
MC1403BD		SO-8

FIGURE 1 – A REFERENCE FOR MOTOROLA MONOLITHIC D/A CONVERTERS



*Caution: System stability may be affected if output capacitance exceeds 1.0 µF. Using higher capacitance values is not recommended and should be carefully considered.

PROVIDING THE REFERENCE CURRENT FOR MOTOROLA MONOLITHIC D/A CONVERTERS

The MC1403/1503 makes an ideal reference for the Motorola monolithic D/A converters. The MC1408/1508 converter requires a stable current reference of nominally 2.0 mA. This can be easily obtained from the MC1403/1503 with the addition of a series resistor, R1. A variable resistor, R2, is recommended to provide means for full-scale adjust on the D/A converter.

The resistor R3 improves temperature performance by matching the impedance on both inputs of the D/A reference amplifier. The capacitor decouples any noise present on the reference line. It is essential if the D/A converter is located any appreciable distance from the reference.

A single MC1403/1503 reference can provide the required current input for up to five of the monolithic D/A converters.

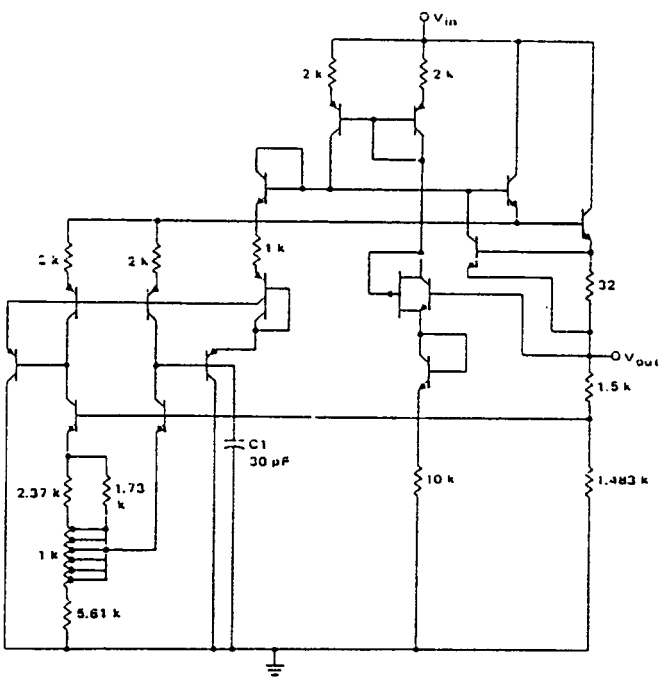
MC1403,A, MC1503

ELECTRICAL CHARACTERISTICS (V_{in} = 15 V, T_A = 25°C unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
Output Voltage (I_O = 0 mA)	V_{OUT}	2.475	2.5	2.525	V
Temperature Coefficient of Output Voltage	$\Delta V_O/\Delta T$				ppm/°C
MC1503		—	—	55	
MC1403*		—	10	40	
MC1403A		—	10	25	
Output Voltage Change (over specified temperature range)	ΔV_O				mV
MC1503 −55°C to +125°C		—	—	25	
MC1403* 0°C to +70°C		—	—	7.0	
MC1403A		—	—	4.4	
Line Regulation (I_O = 0 mA)	Regline				mV
(15 V ≤ V_I ≤ 40 V)		—	1.2	4.5	
(4.5 V ≤ V_I ≤ 15 V)		—	0.6	3.0	
Load Regulation (0 mA ≤ I_O ≤ 10 mA)	Regload	—	—	10	mV
Quiescent Current (I_O = 0 mA)	I_Q	—	1.2	1.5	mA

*This test is not applicable to the MC1403D surface mount device.

FIGURE 2 – MC1403/1503 SCHEMATIC



MOTOROLA
SEMICONDUCTOR
TECHNICAL DATA

MC1404
MC1404A
MC1504

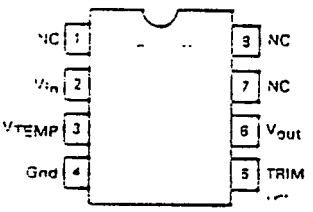
PRECISION LOW DRIFT
VOLTAGE REFERENCES

5.0, 6.25, and 10-VOLT OUTPUT VOLTAGES

LASER TRIMMED SILICON
MONOLITHIC INTEGRATED CIRCUIT

U SUFFIX
CERAMIC PACKAGE
CASE 693

PIN ASSIGNMENTS



VOLTAGE REFERENCE FAMILY

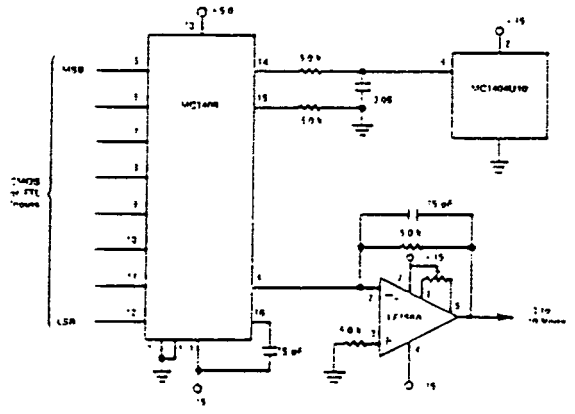
The MC1404 series of ICs is a family of temperature-compensated voltage references for precision data conversion applications, such as A/D, D/A, V/F, and F/V. Advances in laser-trimming and ion-implanted devices, as well as monolithic fabrication techniques, make these devices stable and accurate to 12 bits over both military and commercial temperature ranges. In addition to excellent temperature stability, these parts offer excellent long-term stability and low noise.

- Output Voltages: Standard, 5.0 V, 6.25 V, 10 V
- Trimmable Output: > ±6%
- Wide Input Voltage Range: V_{ref} = 2.5 V to 40 V
- Low Quiescent Current: 1.25 mA Typical
- Temperature Coefficient: 10 ppm/°C Typical
- Low Output Noise: 12 μV p-p Typical
- Excellent Ripple Rejection: > 60 dB Typical

TYPICAL APPLICATIONS

- Voltage Reference for 8 – 12 Bit D/A Converters
- Low T_C Zener Replacement
- High Stability Current Reference
- MPU D/A and A/D Applications

FIGURE 1 — VOLTAGE OUTPUT 8-BIT DAC USING MC1404U10



ORDERING INFORMATION

PACKAGE Ceramic DIP	
Device	**Temperature Range**
5.0 Volts	
MC1504U5	−55°C to +125°C
MC1404U5	0°C to +70°C
MC1404AU5	0°C to +70°C
6.25 Volts	
MC1504U6	−55°C to +125°C
MC1404U6	0°C to +70°C
MC1404AU6	0°C to +70°C
10 Volts	
MC1504U10	−55°C to +125°C
MC1404U10	0°C to +70°C
MC1404AU10	0°C to +70°C

MC1404,A, MC1504

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Input Voltage	V_{in}	40	V
Storage Temperature	T_{stg}	−65 to +150	°C
Junction Temperature	T_J	+175	°C
Operating Ambient Temperature Range MC1504 MC1404,A	T_A	 −55 to +125 0 to +70	°C

ELECTRICAL CHARACTERISTICS (V_{in} = 15 Volts, T_A = 25°C and Trim Terminal not connected unless otherwise noted)

Characteristic		Symbol	MC1404,A Min	MC1404,A Typ	MC1404,A Max	MC1504 Min	MC1504 Typ	MC1504 Max	Unit
Output Voltage I_O = 0 mA	MC1404U5, AU5,MC1504U5	V_O	4.95	5.0	5.05	4.95	5.0	5.05	Volt
	MC1404U6, AU6,MC1504U6		6.19	6.25	6.31	6.19	6.25	6.31	
	MC1404U10, AU10,MC1504U10		9.9	10	10.1	9.9	10	10.1	
Output Voltage Tolerance		—	—	±0.1	±1.0	—	±0.1	±1.0	%
Output Trim Range (Figure 10) R_P = 100 kΩ		ΔV_{TRIM}	±5.0	—	—	±5.0	—	—	%
Output Voltage Temperature Coefficient, Over Full Temperature Range		$\Delta V_O/\Delta T$							ppm/°C
	MC1404, MC1504		—	70	40	—	—	55	
	MC1404A		—	30	35	—	—	—	
Maximum Output Voltage Change Over Temperature Range		ΔV_O							mV
	MC1404U5, MC1504U5		—	—	75	—	—	50	
	MC1404AU5		—	—	2.0	—	—	—	
	MC1404U6, MC1504U6		—	—	7.5	—	—	52	
	MC1404AU6		—	—	11	—	—	—	
	MC1404U10, MC1504U10		—	—	15	—	—	99	
	MC1404AU10		—	—	19	—	—	—	
Line Regulation (1) (V_{in} = V_{out} + 2.5 V to 40 V, I_{out} = 0 mA)		Reg_{line}	—	2.0	5.0	—	2.0	5.0	mV
Load Regulation (1) (0 ≤ I_O ≤ 10 mA)		Reg_{load}	—	—	10	—	—	10	mV
Quiescent Current (I_O = 0 mA)		I_Q	—	1.2	1.5	—	1.2	1.5	mA
Short Circuit Current		I_{sc}	—	20	45	—	—	45	mA
Long Term Stability		—	—	35	—	—	35	—	ppm/1000 hrs

Note 1: Includes thermal effects.

DYNAMIC CHARACTERISTICS (V_{in} = 15 V, T_A = 25°C all voltage ranges unless otherwise noted)

Characteristic	Symbol	MC1404,A Min	MC1404,A Typ	MC1404,A Max	MC1504 Min	MC1504 Typ	MC1504 Max	Unit
Turn-On-Settling Time (to ±0.01%)	t_S	—	50	—	—	50	—	μs
Output Noise Voltage — P to P (Bandwidth 0.1 to 10 Hz)	V_n	—	12	—	—	12	—	μV
Small-Signal Output Impedance 120 Hz 500 Hz	Z_o	 — —	 0.15 0.2	 — —	 — —	 0.15 0.2	 — —	Ω
Power Supply Rejection Ratio	PSRR	70	80	—	70	80	—	dB

National Semiconductor Corporation

ADC0801, ADC0802, ADC0803, ADC0804, ADC0805 8-Bit μP Compatible A/D Converters

General Description

The ADC0801, ADC0802, ADC0803, ADC0804 and ADC0805 are CMOS 8-bit successive approximation A/D converters that use a differential potentiometric ladder—similar to the 256R products. These converters are designed to allow operation with the NSC800 and INS8080A derivative control bus with TRI-STATE® output latches directly driving the data bus. These A/Ds appear like memory locations or I/O ports to the microprocessor and no interfacing logic is needed.

Differential analog voltage inputs allow increasing the common-mode rejection and offsetting the analog zero input voltage value. In addition, the voltage reference input can be adjusted to allow encoding any smaller analog voltage span to the full 8 bits of resolution.

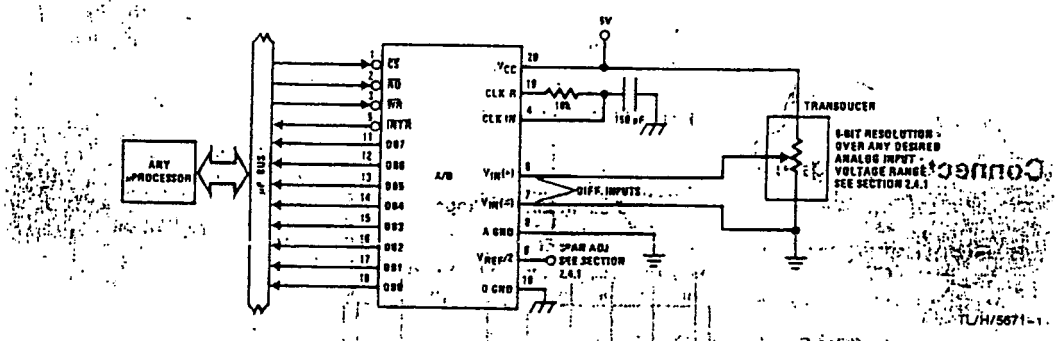
Features

- Compatible with 8080 μP derivatives—no interfacing logic needed - access time - 135 ns
- Easy interface to all microprocessors, or operates "stand alone"
- Differential analog voltage inputs
- Logic inputs and outputs meet both MOS and TTL voltage level specifications
- Works with 2.5V (LM336) voltage reference
- On-chip clock generator
- 0V to 5V analog input voltage range with single 5V supply
- No zero adjust required
- 0.3" standard width 20-pin DIP package
- 20-pin molded chip carrier or small outline package
- Operates ratiometrically or with 5 V_{DC}, 2.5 V_{DC}, or analog span adjusted voltage reference

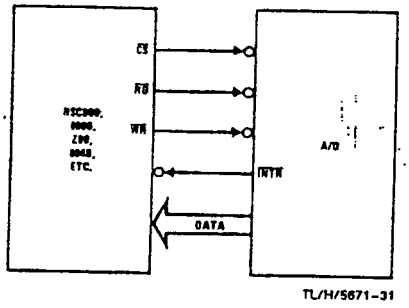
Key Specifications

- Resolution: 8 bits
- Total error: ±¼ LSB, ±½ LSB and ±1 LSB
- Conversion time: 100 μs

Typical Applications



8080 Interface



Part Number	Error Specification (Includes Full-Scale, Zero Error, and Non-Linearity) Full-Scale Adjusted	$V_{REF}/2 = 2.500\ V_{DC}$ (No Adjustments)	$V_{REF}/2$ = No Connection (No Adjustments)
ADC0801	±¼ LSB		
ADC0802		±½ LSB	
ADC0803	±½ LSB		
ADC0804		±1 LSB	
ADC0805			±1 LSB

Absolute Maximum Ratings (Notes 1 & 2)

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC}) (Note 3)	6.5V
Voltage	
Logic Control Inputs	−0.3V to +18V
At Other Input and Outputs	−0.3V to (V_{CC} + 0.3V)
Lead Temp. (Soldering, 10 seconds)	
Dual-In-Line Package (plastic)	260°C
Dual-In-Line Package (ceramic)	300°C
Surface Mount Package	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
Storage Temperature Range	−65°C to +150°C
Package Dissipation at T_A = 25°C	875 mW
ESD Susceptibility (Note 10)	800V

Operating Ratings (Notes 1 & 2)

Temperature Range	$T_{MIN} \le T_A \le T_{MAX}$
ADC0801/02LJ	$-55°C \le T_A \le +125°C$
ADC0801/02/03/04LCJ	$-40°C \le T_A \le +85°C$
ADC0801/02/03/05LCN	$-40°C \le T_A \le +85°C$
ADC0804LCN	$0°C \le T_A \le +70°C$
ADC0802/03/04LCV	$0°C \le T_A \le +70°C$
ADC0802/03/04LCWM	$0°C \le T_A \le +70°C$
Range of V_{CC}	4.5 V_{DC} to 6.3 V_{DC}

Electrical Characteristics

The following specifications apply for V_{CC} = 5 V_{DC}, $T_{MIN} \le T_A \le T_{MAX}$ and f_{CLK} = 640 kHz unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
ADC0801: Total Adjusted Error (Note 8)	With Full-Scale Adj. (See Section 2.5.2)			±¼	LSB
ADC0802: Total Unadjusted Error (Note 8)	$V_{REF}/2$ = 2.500 V_{DC}			±½	LSB
ADC0803: Total Adjusted Error (Note 8)	With Full-Scale Adj. (See Section 2.5.2)			±½	LSB
ADC0804: Total Unadjusted Error (Note 8)	$V_{REF}/2$ = 2.500 V_{DC}			±1	LSB
ADC0805: Total Unadjusted Error (Note 8)	$V_{REF}/2$-No Connection			±1	LSB
$V_{REF}/2$ Input Resistance (Pin 9)	ADC0801/02/03/05 ADC0804 (Note 9)	2.5 0.75	8.0 1.1		kΩ kΩ
Analog Input Voltage Range	(Note 4) V(+) or V(−)	Gnd−0.05		V_{CC}+0.05	V_{DC}
DC Common-Mode Error	Over Analog Input Voltage Range		±1/16	±⅛	LSB
Power Supply Sensitivity	V_{CC} = 5 V_{DC} ±10% Over Allowed V_{IN}(+) and V_{IN}(−) Voltage Range (Note 4)		±1/16	±⅛	LSB

AC Electrical Characteristics

The following specifications apply for V_{CC} = 5 V_{DC} and T_A = 25°C unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
T_C	Conversion Time	f_{CLK} = 640 kHz (Note 6)	103		114	μs
T_C	Conversion Time	(Note 5, 6)	66		73	$1/f_{CLK}$
f_{CLK}	Clock Frequency Clock Duty Cycle	V_{CC} = 5V, (Note 5) (Note 5)	100 40	640	1460 60	kHz %
CR	Conversion Rate in Free-Running Mode	$\overline{INTR}$ tied to $\overline{WR}$ with $\overline{CS}$ = 0 V_{DC}, f_{CLK} = 640 kHz	8770		9708	conv/s
$t_{W(WR)L}$	Width of $\overline{WR}$ Input (Start Pulse Width)	$\overline{CS}$ = 0 V_{DC} (Note 7)	100			ns
t_{ACC}	Access Time (Delay from Falling Edge of $\overline{RD}$ to Output Data Valid)	C_L = 100 pF		135	200	ns
t_{1H}, t_{0H}	TRI-STATE Control (Delay from Rising Edge of $\overline{RD}$ to Hi-Z State)	C_L = 10 pF, R_L = 10k (See TRI-STATE Test Circuits)		125	200	ns
t_{WI}, t_{RI}	Delay from Falling Edge of $\overline{WR}$ or $\overline{RD}$ to Reset of $\overline{INTR}$			300	450	ns
C_{IN}	Input Capacitance of Logic Control Inputs			5	7.5	pF
C_{OUT}	TRI-STATE Output Capacitance (Data Buffers)			5	7.5	pF
CONTROL INPUTS [Note: CLK IN (Pin 4) is the input of a Schmitt trigger circuit and is therefore specified separately]						
V_{IN}(1)	Logical "1" Input Voltage (Except Pin 4 CLK IN)	V_{CC} = 5.25 V_{DC}	2.0		15	V_{DC}

AC Electrical Characteristics (Continued)

The following specifications apply for $V_{CC} = 5V_{DC}$ and $T_{MIN} \leq T_A \leq T_{MAX}$, unless otherwise specified.

Symbol	Parameter	Conditions	Min.	Typ	Max	Units
CONTROL INPUTS [Note: CLK IN (Pin 4) is the input of a Schmitt trigger circuit and is therefore specified separately]						
V_{IN} (0)	Logical "0" Input Voltage (Except Pin 4 CLK IN)	$V_{CC} = 4.75\ V_{DC}$			0.8	V_{DC}
I_{IN} (1)	Logical "1" Input Current (All Inputs)	$V_{IN} = 5\ V_{DC}$		0.005	1	μA_{DC}
I_{IN} (0)	Logical "0" Input Current (All Inputs)	$V_{IN} = 0\ V_{DC}$	−1	−0.005		μA_{DC}
CLOCK IN AND CLOCK R						
V_T+	CLK IN (Pin 4) Positive Going Threshold Voltage		2.7	3.1	3.5	V_{DC}
V_T-	CLK IN (Pin 4) Negative Going Threshold Voltage		1.5	1.8	2.1	V_{DC}
V_H	CLK IN (Pin 4) Hysteresis $(V_T+)-(V_T-)$		0.6	1.3	2.0	V_{DC}
V_{OUT} (0)	Logical "0" CLK R Output Voltage	$I_O = 360\ \mu A$ $V_{CC} = 4.75\ V_{DC}$			0.4	V_{DC}
V_{OUT} (1)	Logical "1" CLK R Output Voltage	$I_O = -360\ \mu A$ $V_{CC} = 4.75\ V_{DC}$	2.4			V_{DC}
DATA OUTPUTS AND $\overline{INTR}$						
V_{OUT} (0)	Logical "0" Output Voltage Data Outputs $\overline{INTR}$ Output	 $I_{OUT} = 1.6\ mA$, $V_{CC} = 4.75\ V_{DC}$ $I_{OUT} = 1.0\ mA$, $V_{CC} = 4.75\ V_{DC}$			 0.4 0.4	 V_{DC} V_{DC}
V_{OUT} (1)	Logical "1" Output Voltage	$I_O = -360\ \mu A$, $V_{CC} = 4.75\ V_{DC}$	2.4			V_{DC}
V_{OUT} (1)	Logical "1" Output Voltage	$I_O = -10\ \mu A$, $V_{CC} = 4.75\ V_{DC}$	4.5			V_{DC}
I_{OUT}	TRI-STATE Disabled Output Leakage (All Data Buffers)	$V_{OUT} = 0\ V_{DC}$ $V_{OUT} = 5\ V_{DC}$	−3		 3	μA_{DC} μA_{DC}
I_{SOURCE}		V_{OUT} Short to Gnd, $T_A = 25°C$	4.5	6		mA_{DC}
I_{SINK}		V_{OUT} Short to V_{CC}, $T_A = 25°C$	9.0	16		mA_{DC}
POWER SUPPLY						
I_{CC}	Supply Current (Includes Ladder Current) ADC0801/02/03/04LCJ/05 ADC0804LCN/LCV/LCWM	$f_{CLK} = 640\ kHz$, $V_{REF}/2 = NC$, $T_A = 25°C$ and $\overline{CS} = 5V$		 1.1 1.9	 1.8 2.5	 mA mA

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its specified operating conditions.

Note 2: All voltages are measured with respect to Gnd, unless otherwise specified. The separate A Gnd point should always be wired to the D Gnd.

Note 3: A zener diode exists, internally, from V_{CC} to Gnd and has a typical breakdown voltage of 7 V_{DC}.

Note 4: For $V_{IN}(-) \geq V_{IN}(+)$ the digital output code will be 0000 0000. Two on-chip diodes are tied to each analog input (see block diagram) which will forward conduct for analog input voltages one diode drop below ground or one diode drop greater than the V_{CC} supply. Be careful, during testing at low V_{CC} levels (4.5V), as high level analog inputs (5V) can cause this input diode to conduct—especially at elevated temperatures, and cause errors for analog inputs near full-scale. The spec allows 50 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 50 mV, the output code will be correct. To achieve an absolute 0 V_{DC} to 5 V_{DC} input voltage range will therefore require a minimum supply voltage of 4.950 V_{DC} over temperature variations, initial tolerance and loading.

Note 5: Accuracy is guaranteed at $f_{CLK} = 640$ kHz. At higher clock frequencies accuracy can degrade. For lower clock frequencies, the duty cycle limits can be extended so long as the minimum clock high time interval or minimum clock low time interval is no less than 275 ns.

Note 6: With an asynchronous start pulse, up to 8 clock periods may be required before the internal clock phases are proper to start the conversion process. The start request is internally latched, see *Figure 2* and section 2.0.

Note 7: The $\overline{CS}$ input is assumed to bracket the $\overline{WR}$ strobe input and therefore timing is dependent on the $\overline{WR}$ pulse width. An arbitrarily wide pulse width will hold the converter in a reset mode and the start of conversion is initiated by the low to high transition of the $\overline{WR}$ pulse (see timing diagrams).

Note 8: None of these A/Ds requires a zero adjust (see section 2.5.1). To obtain zero code at other analog input voltages see section 2.5 and *Figure 5*.

Note 9: The $V_{REF}/2$ pin is the center point of a two resistor divider connected from V_{CC} to ground. Each resistor is 2.2k, except for the ADC0804LCJ where each resistor is 16k. Total ladder input resistance is the sum of the two equal resistors.

Note 10: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Design Migration from XC4000 to XC4000E

XAPP 062 October 15,1996 (Version1.0)

Application Note by Lois Cartier and Marc Baker

Summary

The XC4000E is an enhanced architecture based on the XC4000 family, but offers many new features, particularly SelectRAM™ memory. When converting XC4000, XC4000A, XC4000D, and XC4000H designs, the XC4000E is an excellent choice. The conversion process may be as simple as downloading the same bitstream into the XC4000E device (XC4000 and XC4000D bitstreams only), or it may involve changes to the schematic or HDL code. This Application Note describes techniques that should be employed to convert from any of the XC4000, XC4000A, XC4000D, or XC4000H families to the XC4000E family.

Xilinx Family

XC4000E

Demonstrates

Techniques for migrating XC4000 designs to the XC4000E architecture

Introduction

Some variations of the XC4000 family are no longer recommended for new designs. These include the XC4000 standard family, XC4000A, XC4000D, and XC4000H (referred to in this application note as the older XC4000 families). These have been superseded by the newer XC4000 Series families, which include the XC4000E, XC4000L, XC4000EX, and XC4000XL families. Designs in the older versions of the XC4000 can be converted to the new XC4000 Series. In some cases, particularly for XC4000D designs or other designs not including RAM, the XC5200 family should be considered for its lower cost. For a discussion of converting XC4000 family designs to the XC5200 family, see the Xilinx application note XAPP060, "Design Migration from XC4000 to XC5200."

Converting an older XC4000 design to an XC4000E design may or may not require schematic changes. Except for certain I/O capabilities, the XC4000E architecture offers a superset of the architectural features in the older XC4000 families, in addition to many new enhancements. (See Table 1.) Therefore, if none of the special I/O features in the XC4000A or XC4000H are used, the conversion becomes merely a matter of selecting the XC4000E speed grade that will give comparable performance, and recompiling the design targeting the selected device. If retargeting an XC4000 or XC4000D design, in fact, the XC4000E can actually accept the original bitstream. XC4000A and XC4000H designs must be recompiled.

Table 1: Comparison of Features in XC4000 Families

	XC4000	XC4000A	XC4000D	XC4000H	XC4000E
Output Drive/Pin (mA)	12	24	12	4/24	12
Output Slew Rate Options	Fast, Slow	Fast, Slow, Medium Fast, Medium Slow	Fast, Slow	SoftEdge/Resistive load	Fast, Slow
RAM	Yes	Yes	No	Yes	Yes
I/O Registers	Yes	Yes	Yes	No	Yes
Input Thresholds	TTL only	TTL only	TTL only	TTL or CMOS, per I/O	TTL or CMOS, Global
Output High Levels	TTL only	TTL only	TTL only	TTL or CMOS, per I/O	TTL or CMOS, Global
Wide Edge Decoders (per edge)	4	2	4	4	4
Speed Grades	-6, -5, -4	-6, -5, -4	-6, -5, -4	-5, -5	-4, -3, -2

Enhancements

The XC4000E offers a number of architectural improvements to the XC4000. Although the simplest conversion path uses only the resources provided in the originally targeted older XC4000 device, a design can be altered to take advantage of these features if desired.

- **PCI Compliance:** Fully compliant for -3 speed grades and faster.
- **Select-RAM Memory:** In addition to the XC4000 modes, RAM can now be configured for synchronous, edge-triggered Write operation or for dual-port RAM with simultaneous Read/Write. Also, XC4000E RAM can now be configured with user-defined initial values at power-up.
- **Increased System Speed:** Improvements in both device processing and system architecture result in path performance benefits that may have significant impact on the critical paths in a design.
- **More Flexible H Function Generator:** Additional inputs gives this resource the capability of being either partially or fully independent of the other two function generators, increasing the logic capacity and routability of the device.
- **Input/Output Block (IOB) Clock Enable:** I/O flip-flops have clock enables, making them more useful in many applications.
- **Increased Global Access to F and G Function Generators:** More function generator inputs are available to global routing resources, increasing flexibility and performance.
- **CMOS Threshold Compatibility:** Both inputs and/or outputs can be globally programmed for CMOS threshold compatibility.
- **Increased Carry Logic Speed:** The speed of the carry logic chain has increased dramatically, doubled in some cases.
- **Soft Startup:** When the configuration process is complete, outputs programmed for fast slew-rate operation are initially kept slow, avoiding ground-bounce problems when all the outputs are switched on.
- **Configuration Pin Pull-up Resistors:** During configuration, the three mode pins, M0, M1, and M2, have weak pull-up resistors. The effect on converted designs is discussed in "Configuration" on page 2.

Design Guidelines & Considerations

Because the XC4000E is a superset of the older XC4000 architecture, migrating a design from one of the older families to the XC4000E is relatively straightforward. The following issues may need to be addressed.

Configuration

The XC4000E can be configured using the same bitstream as an XC4000 or XC4000D device, or the design can be altered and recompiled to take advantage of some of the new features available in the XC4000E. XC4000A and XC4000H designs must be recompiled, targeting the new device.

One of the new features in the XC4000E is the addition of weak (50 kΩ) pull-up resistors to the configuration mode pins. These internal resistors make it possible to configure in Serial Slave mode (the most common configuration mode) without the addition of any external resistors. Older XC4000 families did not include these pull-up resistors, therefore a high-value pull-down resistor was acceptable to establish a Low value on any of these inputs. The XC4000E requires a resistor value of less than 7 kΩ to reliably generate a logic Low on the input.

When converting from an older XC4000 device to an XC4000E, check the design for pull-down resistor values and change them, if necessary, to 4.7 kΩ resistors. Resistor values above the recommended level may prevent the XC4000E from properly configuring.

Footprint

Older XC4000 and XC4000E devices are footprint compatible for every common package. All control pins, configuration pins, and power pins are in the same locations. In general, no board re-layout is necessary when replacing an XC4000 device with an XC4000E.

An exception may occur when converting from an XC4000A or XC4000H to an XC4000E. In these transitions, the design is moved from a device with a maximum 24 mA sink current to one with a maximum 12 mA sink current, and it may be necessary to split the load across two outputs.

Design Performance

The XC4000E is available in -4, -3 and -2 speed grades. Consequently, the fastest XC4000E is significantly faster than any of the older XC4000 families. However, when converting an existing design, it is recommended that an XC4000E speed grade two grades faster than the original design be used for the initial compilation run. The resulting design should then be tested using XDelay, the Timing Analyzer, or timing simulation, to verify functionality at the required speed.

The reason for the recommended speed grade change is that the XC4000E is optimized for a three layer metal process and a higher overall system level performance, not for each individual specification. Although the majority of specifications are faster in the XC4000E than in any of the older XC4000 families, a few parameters are slower. The recommended approach should result in a successful conversion for the vast majority of designs.

After this first pass, it may be possible to use a slower speed grade. The best way to ensure that design performance will be met in the XC4000E is to use XACT-Performance™ to define the timing requirements of the design.

To place this suggestion in perspective, 80% of designs show equal or better performance in an XC4000E with the same speed grade as the original design. This conservative approach is recommended to maximize the chance that a given design will meet the required timing. See "Reporting Performance at Various Speed Grades" on page 5 for a quick method of evaluating the performance of a single routed design in several different speed grades.

Development System

The XC4000E is supported by the same XACT*step*™ development system and uses the basic software tools that support the older XC4000 families. A new XC4000E library has been added to the set of Unified Libraries to support the architectural features of the XC4000E.

A software version that supports the XC4000E must be used. Users who have upgraded to at least the 6.0.1 (Windows) or 5.2.1 (DOS and UNIX) versions of XACT*step* have the XC4000E library features and the necessary place and route tools. Otherwise, contact the local Xilinx sales representative for the availability of the most recent upgrade.

Input/Output Pads

For XC4000A and XC4000H designs, the difference in I/O capabilities may affect the conversion to an XC4000E.

XC4000A to XC4000E

XC4000A I/O have a 24 mA output drive, while the XC4000E has only 12 mA. Where a 24 mA drive is required, use two pins tied together to double the drive. Add the FAST parameter where speed is important.

XC4000A I/O allow two additional slew rate options, Medium Fast (MEDFAST) and Medium Slow (MEDSLOW). These parameters, if used in an XC4000A design, will need to be changed to FAST in the XC4000E version to achieve higher-than-default speed.

XC4000H to XC4000E

XC4000H devices have a higher I/O count than the corresponding XC4000E devices. If a high I/O count is required, a larger XC4000E device must be used to provide the same number of pins, as shown in Table 2.

Table 2: XC4000H Device Replacement Guide

XC4000H	Max. I/O	XC4000E	Max. I/O
XC4003H	160	XC4010E	160
XC4005H	192	XC4013E	192

XC4000H I/O have a default SoftEdge slew-rate control that limits drive to 4 mA. This is intended for capacitively loaded outputs, and can be selected with the CAP parameter. The alternative is the resistive (RES) mode, which increases drive to 24 mA. XC4000H designs will need the CAP or RES parameters removed when converted to the XC4000E. Add the FAST parameter where speed is important, and use two pins tied together to double the drive to 24 mA if necessary.

XC4000H I/O allow per-pin designation of CMOS or TTL thresholds and output levels, selected for each pin using a CMOS or TTL parameter (TTL is default). In the XC4000E, all input thresholds have the same value, and all output levels are the same; although the two are selected independently. If the XC4000H design has mixed I/O, the recommended approach is to designate the XC4000E inputs as TTL, since an input set for TTL can resolve both TTL and CMOS levels. The output level must depend on the destination devices. XC4000H designs will need the CMOS or TTL parameters removed when converted to the XC4000E.

Migration Methodology

If none of the new features in the XC4000E are used, an existing design can be compiled to the XC4000E using the XC4000 library, which supports all of the XC4000, XC4000A, XC4000D, and XC4000H families. If certain of the new features are to be used, new library components will be required. In this case, it is necessary to change the design to use the new XC4000E libraries. The features requiring the XC4000E library are:

- Synchronous (edge-triggered) or dual-port memory
- IOB clock enables

The library elements supporting these features are described in the *Libraries Supplement Guide*.

Because of the Xilinx Unified Library approach, a design can be migrated between families with a minimum of effort. The migration methodology itself is simple, and following the guidelines and considerations prescribed in this document will greatly improve the success of the migration.

This section describes how to perform the actual migration of an XC4000 design into the XC4000E library for three third-party CAE interfaces.

VIEWlogic

To migrate an XC4000 VIEWlogic schematic to the XC4000E library, perform the following steps:

1. Add the XC4000E library to the VIEWlogic library search path. Edit the `viewdraw.ini` file in the project directory and add the XC4000E library path so that it appears in `viewdraw.ini` before the path to the XC4000 library.
2. To convert the XC4000 alias to XC4000E, run altran, the VIEWlogic library alias maintenance program:

   `altran -l primary xc4000=xc4000e`

   where xc4000 is the alias assigned to the XC4000 library, and xc4000e is the alias assigned to the XC4000E library.
3. Reprocess the design by running XMake or the Flow Engine.

Mentor

To migrate an XC4000 Mentor schematic to the XC4000E library, perform the following steps:

1. Invoke `PLD_DA` (it is not necessary to open the schematic).
2. On DA's desktop background (that is, outside of any schematic or symbol windows), call up the session pop-up menu with the mouse button on the right and select `Convert Design`.

   Of the fields in the resulting dialog box these are the most relevant:
3. `Select a group of designs from a list file?` Whether you answer "`yes`" or "`no`" to this question affects the following field.
4. `Enter Design name (List file = no)`. The name of the design to retarget. `Convert Design` does *not* traverse the hierarchy of a schematic.
5. `Enter list file name (List file = yes)`. A file which lists designs, one per line, to retarget. This is useful if your design has many lower-level schematics.

TIP: A list file can easily be created by typing:

```
ls *.mgc_component.attr | sed
 s/.mgc_component.attr//g > listfile
```

The `ls` command lists all MGC components within a single directory. The `sed` command strips away the `.mgc_component.attr` trailer. The result is redirected to `listfile`.

6. `Schematic name`. The name of the schematic model (the default is "`schematic`").
7. `Check & Save switch`. Because all schematic sheets in `Convert Design` are literally re-drawn in Design Architect, you must apply `Check & Save` to each sheet. This switch controls whether to do this automatically. By default, this switch is set for manual checking because it allows you to spot Xilinx components that did not convert properly. Use the manual setting until you are comfortable with how `Convert Design` works and you are certain that all Xilinx components will convert properly.
3. `From Technology`. The device family from which you are converting (e.g., `XC4000`). This and the next field are case insensitive.
9. `To Technology`. The device family to which you are converting.
10. After filling out the fields in the dialog box and selecting "`OK`," you will see Convert Design doing its job directly in Design Architect.
11. Reprocess the design by running XMake or the Flow Engine.

Foundation

To migrate an XC4000 Foundation schematic to the XC4000E library, perform the following steps:

1. Select the `Project Type` option from the `Menu` file. Change `Family`, `Part`, and `Speed` settings, as desired, and click the `Change` button.
2. Open and save each schematic sheet macro. To do so, run the Schematic Editor and select the `Open` option from the `File` menu. The Open Sheet window allows you to quickly open all project top-level sheets and project schematic macros. Inspect the Project Manager messages for any warnings and errors.
3. Re-synthesize and update all FSM and HDL macros. Use the hierarchy browser in the Project Manager to search the project for the macros.
4. Note: if your project contains components that are not available in the new system library, you have to modify the project so as to preserve its functionality. In the case of a top level HDL project, you will need to re-synthesize the entire project.
5. Reprocess the design by running XMake or the Flow Engine.

Additional Software Tips

Following are some additional tips that may help in converting a design.

Using the Old Design as a Guide

An XC4000 or XC4000D design can be used as a guide for an XC4000E design. Rename the old design, for example to "guide.lca", and add the following parameter when running PPR:

guide=guide.lca

Alternatively, in the Windows environment simply choose a previous XC4000 revision in the Guide File pull-down of the implementation window.

Converting a Routed LCA File

The XDelay "convert" option can be used to translate a routed XC4000 or XC4000D LCA file to an XC4000E. The syntax is as follows:

xdelay -convert <design name>.lca <part & package> <new name>.lca

For example, to convert an XC4003 to an XC4003E, type:

xdelay -convert old.lca 4003EPC84 new.lca

The conversion utility will not allow you to choose a speed grade or write delay information into the new LCA file, so to perform these steps, run XDelay again with the following syntax:

xdelay -w -u -<new speed> <design name>

For example:

xdelay -w -u -3 new.lca

This command writes the *new* speed grade and delay information into the file without otherwise changing the design.

Reporting Performance at Various Speed Grades

XDelay can be used to report performance at various speed grades without changing the LCA file. To show the delays of the most critical paths, create a short XDelay report using the following command:

xdelay -u -<speed> -o critical.rpt <design name>

For example:

xdelay -u -2 -o critical.rpt new.lca

This command produces a text file called critical.rpt that contains the minimum worst-case pad-to-setup, clock-to-setup, and clock-to-pad values allowable for each clock in the design. Effectively, this report provides all the information necessary to evaluate the performance of the new speed grade.

Alternatively, in the Windows environment use the Performance Summary in the Timing Analyzer. Both XDelay and the Timing Analyzer can also be used to examine specific path delays in more detail if desired.

Additional Information

If there are problems with the conversion process, please contact the Xilinx Technical Support hotline for assistance.

Email: hotline@xilinx.com (24 hours)

Voice: 1-800-255-7778 (6:30AM - 5PM PST)

FAX: 1-408-879-4442 (24 hours)

Package-Specific Pinout Tables

PC84 Package Pinouts

Pin	XC4003E	XC4005E XC4005L	XC4006E	XC4008E	XC4010E XC4010L
P1	GND	GND	GND	GND	GND
P2	VCC	VCC	VCC	VCC	VCC
P3	I/O (A8)	I/O (A8)	I/O (A8)	I/O (A8)	I/O (A8)
P4	I/O (A9)	I/O (A9)	I/O (A9)	I/O (A9)	I/O (A9)
P5	I/O (A10)	I/O (A10)	I/O (A10)	I/O (A10)	I/O (A10)
P6	I/O (A11)	I/O (A11)	I/O (A11)	I/O (A11)	I/O (A11)
P7	I/O (A12)	I/O (A12)	I/O (A12)	I/O (A12)	I/O (A12)
P8	I/O (A13)	I/O (A13)	I/O (A13)	I/O (A13)	I/O (A13)
P9	I/O (A14)	I/O (A14)	I/O (A14)	I/O (A14)	I/O (A14)
P10	I/O, SGCK1 (A15)	I/O, SGCK1 (A15)	I/O, SGCK1 (A15)	I/O, SGCK1 (A15)	I/O, SGCK1 (A15)
P11	VCC	VCC	VCC	VCC	VCC
P12	GND	GND	GND	GND	GND
P13	I/O, PGCK1 (A16)	I/O, PGCK1 (A16)	I/O, PGCK1 (A16)	I/O, PGCK1 (A16)	I/O, PGCK1 (A16)
P14	I/O (A17)	I/O (A17)	I/O (A17)	I/O (A17)	I/O (A17)
P15	I/O, TDI	I/O, TDI	I/O, TDI	I/O, TDI	I/O, TDI
P16	I/O, TCK	I/O, TCK	I/O, TCK	I/O, TCK	I/O, TCK
P17	I/O, TMS	I/O, TMS	I/O, TMS	I/O, TMS	I/O, TMS
P18	I/O	I/O	I/O	I/O	I/O
P19	I/O	I/O	I/O	I/O	I/O
P20	I/O	I/O	I/O	I/O	I/O
P21	GND	GND	GND	GND	GND
P22	VCC	VCC	VCC	VCC	VCC
P23	I/O	I/O	I/O	I/O	I/O
P24	I/O	I/O	I/O	I/O	I/O
P25	I/O	I/O	I/O	I/O	I/O
P26	I/O	I/O	I/O	I/O	I/O
P27	I/O	I/O	I/O	I/O	I/O
P28	I/O	I/O	I/O	I/O	I/O
P29	I/O, SCGK2	I/O, SCGK2	I/O, SCGK2	I/O, SCGK2	I/O, SCGK2
P30	O (M1)	O (M1)	O (M1)	O (M1)	O (M1)
P31	GND	GND	GND	GND	GND
P32	I (M0)	I (M0)	I (M0)	I (M0)	I (M0)
P33	VCC	VCC	VCC	VCC	VCC
P34	I (M2)	I (M2)	I (M2)	I (M2)	I (M2)
P35	I/O, PGCK2	I/O, PGCK2	I/O, PGCK2	I/O, PGCK2	I/O, PGCK2
P36	I/O (HDC)	I/O (HDC)	I/O (HDC)	I/O (HDC)	I/O (HDC)
P37	I/O ($\overline{LDC}$)	I/O ($\overline{LDC}$)	I/O ($\overline{LDC}$)	I/O ($\overline{LDC}$)	I/O ($\overline{LDC}$)
P38	I/O	I/O	I/O	I/O	I/O
P39	I/O	I/O	I/O	I/O	I/O
P40	I/O	I/O	I/O	I/O	I/O
P41	I/O ($\overline{INIT}$)	I/O ($\overline{INIT}$)	I/O ($\overline{INIT}$)	I/O ($\overline{INIT}$)	I/O ($\overline{INIT}$)
P42	VCC	VCC	VCC	VCC	VCC
P43	GND	GND	GND	GND	GND
P44	I/O	I/O	I/O	I/O	I/O
P45	I/O	I/O	I/O	I/O	I/O
P46	I/O	I/O	I/O	I/O	I/O
P47	I/O	I/O	I/O	I/O	I/O
P48	I/O	I/O	I/O	I/O	I/O
P49	I/O	I/O	I/O	I/O	I/O
P50	I/O	I/O	I/O	I/O	I/O
P51	I/O, SGCK3	I/O, SGCK3	I/O, SGCK3	I/O, SGCK3	I/O, SGCK3
P52	GND	GND	GND	GND	GND
P53	DONE	DONE	DONE	DONE	DONE
P54	VCC	VCC	VCC	VCC	VCC
P55	$\overline{PRO-GRAM}$	$\overline{PRO-GRAM}$	$\overline{PRO-GRAM}$	$\overline{PRO-GRAM}$	$\overline{PRO-GRAM}$
P56	I/O (D7)	I/O (D7)	I/O (D7)	I/O (D7)	I/O (D7)
P57	I/O, PGCK3	I/O, PGCK3	I/O, PGCK3	I/O, PGCK3	I/O, PGCK3
P58	I/O (D6)	I/O (D6)	I/O (D6)	I/O (D6)	I/O (D6)
P59	I/O (D5)	I/O (D5)	I/O (D5)	I/O (D5)	I/O (D5)
P60	I/O ($\overline{CS0}$)	I/O ($\overline{CS0}$)	I/O ($\overline{CS0}$)	I/O ($\overline{CS0}$)	I/O ($\overline{CS0}$)
P61	I/O (D4)	I/O (D4)	I/O (D4)	I/O (D4)	I/O (D4)
P62	I/O	I/O	I/O	I/O	I/O
P63	VCC	VCC	VCC	VCC	VCC
P64	GND	GND	GND	GND	GND
P65	I/O (D3)	I/O (D3)	I/O (D3)	I/O (D3)	I/O (D3)
P66	I/O ($\overline{RS}$)	I/O ($\overline{RS}$)	I/O ($\overline{RS}$)	I/O ($\overline{RS}$)	I/O ($\overline{RS}$)
P67	I/O (D2)	I/O (D2)	I/O (D2)	I/O (D2)	I/O (D2)
P68	I/O	I/O	I/O	I/O	I/O
P69	I/O (D1)	I/O (D1)	I/O (D1)	I/O (D1)	I/O (D1)
P70	I/O ($\overline{RCLK}$, RDY/ $\overline{BUSY}$)	I/O ($\overline{RCLK}$, RDY/ $\overline{BUSY}$)	I/O ($\overline{RCLK}$, RDY/ $\overline{BUSY}$)	I/O ($\overline{RCLK}$, RDY/ $\overline{BUSY}$)	I/O ($\overline{RCLK}$, RDY/ $\overline{BUSY}$)
P71	I/O (D0, DIN)	I/O (D0, DIN)	I/O (D0, DIN)	I/O (D0, DIN)	I/O (D0, DIN)
P72	I/O, SGCK4 (DOUT)	I/O, SGCK4 (DOUT)	I/O, SGCK4 (DOUT)	I/O, SGCK4 (DOUT)	I/O, SGCK4 (DOUT)
P73	CCLK	CCLK	CCLK	CCLK	CCLK
P74	VCC	VCC	VCC	VCC	VCC
P75	O, TDO	O, TDO	O, TDO	O, TDO	O, TDO
P76	GND	GND	GND	GND	GND
P77	I/O (A0, $\overline{WS}$)	I/O (A0, $\overline{WS}$)	I/O (A0, $\overline{WS}$)	I/O (A0, $\overline{WS}$)	I/O (A0, $\overline{WS}$)
P78	I/O, PGCK4 (A1)	I/O, PGCK4 (A1)	I/O, PGCK4 (A1)	I/O, PGCK4 (A1)	I/O, PGCK4 (A1)
P79	I/O (CS1, A2)	I/O (CS1, A2)	I/O (CS1, A2)	I/O (CS1, A2)	I/O (CS1, A2)
P80	I/O (A3)	I/O (A3)	I/O (A3)	I/O (A3)	I/O (A3)
P81	I/O (A4)	I/O (A4)	I/O (A4)	I/O (A4)	I/O (A4)
P82	I/O (A5)	I/O (A5)	I/O (A5)	I/O (A5)	I/O (A5)
P83	I/O (A6)	I/O (A6)	I/O (A6)	I/O (A6)	I/O (A6)
P84	I/O (A7)	I/O (A7)	I/O (A7)	I/O (A7)	I/O (A7)

AD7821

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 WR of RD. The provision for 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.

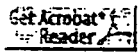
The AD7821 has a built-in track-and-hold function capable of digitizing full-scale signals up to 100 kHz max. It also uses a half-flash conversion technique that eliminates the need to generate a CLK signal for the ADC.

The AD7821 is designed with standard microprocessor control signals ³CS, RD, WR, RDY, INT) and latched, three-state data outputs capable of interfacing to high-speed data buses. An overflow output (OFL) is also provided for cascading devices to achieve higher resolution.

The AD7821 is fabricated in Linear-Compatible CMOS (LC^2MOS), an advanced, mixed technology process combining precision bipolar circuits with low-power CMOS logic. The part features a low power dissipation of 50 mW.

Datasheets

Listed below is the datasheet for this product. This datasheet is available in Adobe Acrobat PDF format. To view Acrobat files you must install Adobe Acrobat Reader: Acrobat Reader is available free of charge. To download Adobe Acrobat visit their website.



- LC^2MOS High Speed, µP-Compatible 8-Bit ADC with Track/Hold Function *(255631 bytes)*

Military Products

Listed below is the military datasheet for this product. This datasheet is available in Adobe Acrobat PDF format. To view Acrobat files you must install Adobe Acrobat Reader. Acrobat Reader is available free of charge. To download Adobe Acrobat visit their website.

- 8-Bit, 600nS, Sampling"
  (Contact your local Analog Devices sales office or distributor for this datasheet)

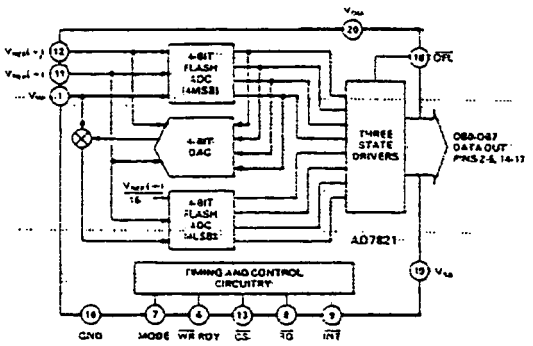
ANALOG DEVICES

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.

REV. A

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 617/329-4700 Fax: 617/326-8703

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)[4]	45	45	dB min	V_{IN} = 99.85 kHz Full-Scale Sine Wave with $f_{SAMPLING}$ = 500 kHz
Total Harmonic Distortion (THD)[4]	-60	-60	dB max	V_{IN} = 99.85 kHz Full-Scale Sine Wave with $f_{SAMPLING}$ = 500 kHz
Peak Harmonic or Spurious Noise[4]	-60	-60	dB max	V_{IN} = 99.85 kHz Full-Scale Sine Wave with $f_{SAMPLING}$ = 500 kHz
Intermodulation Distortion (IMD)[4]				fa (84.72 kHz) and fb (94.97 kHz) Full-Scale Sine Waves with $f_{SAMPLING}$ = 500 kHz
	-60	-60	dB max	Second Order Terms
	-60	-60	dB max	Third Order Terms
Slew Rate, Tracking[5]	1.8	1.8	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[6]	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[6]	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[6] (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[6]	8	8	pF max	Typically 5 pF
POWER SUPPLY				
I_{DD}[7]	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.

[6]Specifications subject to change without notice.

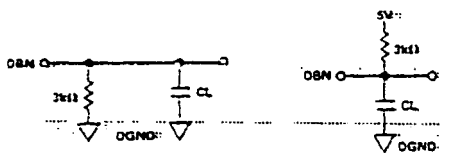
TIMING CHARACTERISTICS[1] (V_{DD} = +5 V ± 5%, V_{SS} = 0 V or −5 V ± 5%; Unipolar or Bipolar Input Range)

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	35	100	ns max	$\overline{CS}$ to RDY Delay. Pull-Up Resistor 5 kΩ.
t_{CRD} / t_{ACC2}[3]	700	375	975	ns max	Conversion Time (RD Mode) 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	ns 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]	280	–	–	ns typ	$\overline{WR}$ to $\overline{INT}$ Delay
	500	610	700	ns max	
t_{READ2}	65	75	35	ns min	$\overline{RD}$ Pulse Width (WR-RD Mode, see Figure 12a) Determined by t_{ACC2}
t_{ACC2}[3]	65	75	85	ns max	Data Access Time (WR-RD Mode, see Figure 12a) C_L = 20 pF
	90	110	130	ns max	C_L = 100 pF
t_{HWR}[2] / t_{ID}[3]	50	100	120	ns max	$\overline{WR}$ to $\overline{INT}$ Delay (Stand-Alone Operation) Data Access Time after $\overline{INT}$ (Stand-Alone Operation)
	30	35	40	ns max	C_L = 20 pF
	45	50	70	ns max	C_L = 100 pF

NOTES
[1]Sample tested at +25°C to ensure compliance. All input control signals are specified with $t_r = t_f$ = 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.

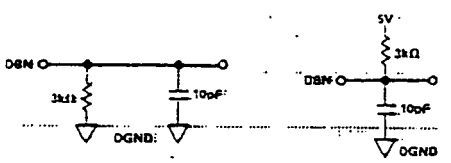
Specifications subject to change without notice.

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) ..... +0°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

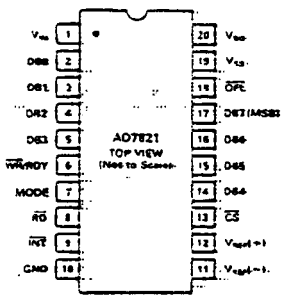
*Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure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

CAUTION

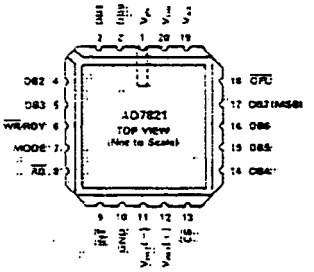
ESD (electrostatic discharge) sensitive device. Electrostatic charges as high as 4000 V readily accumulate on the human body and test equipment and can discharge without detection. Although the AD7821 features proprietary ESD protection circuitry, permanent damage may occur on devices subjected to high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

PIN CONFIGURATIONS

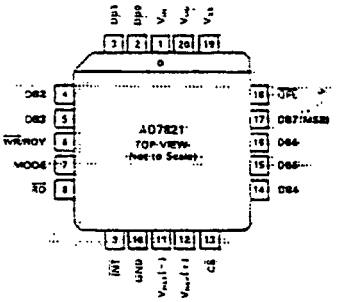
DIP AND SOIC



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:

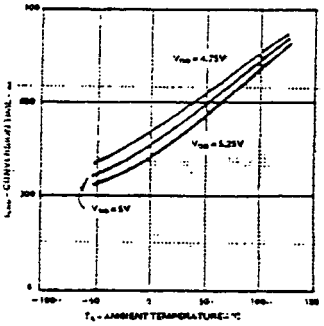
$$SNR = (6.02\ N + 1.76)\ dB$$

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

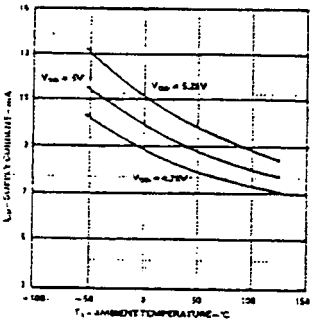
PEAK HARMONIC OR SPURIOUS NOISE

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.

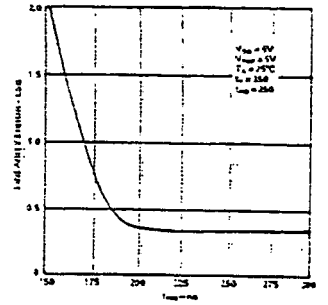
Typical Performance Curves–AD7821



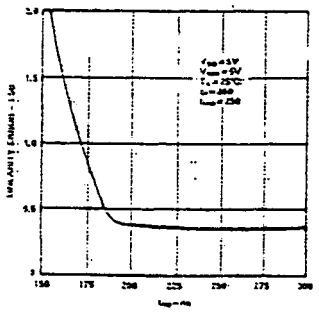
Conversion Time (RD Model vs. Temperature



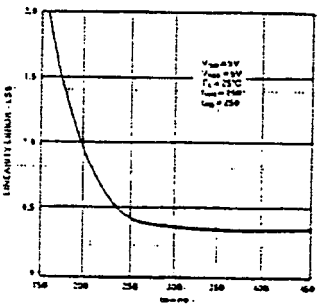
Power Supply Current vs. Temperature (Not Including Reference Ladder)



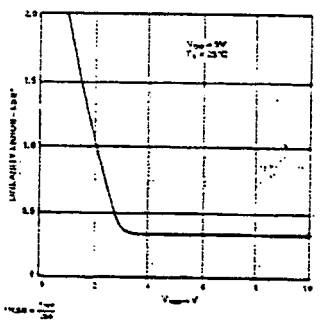
Accuracy vs. t_{WR}



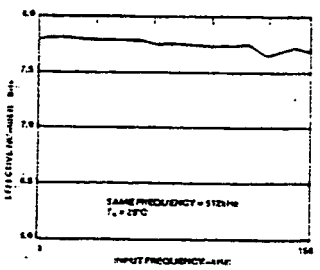
Accuracy vs. t_{AO}



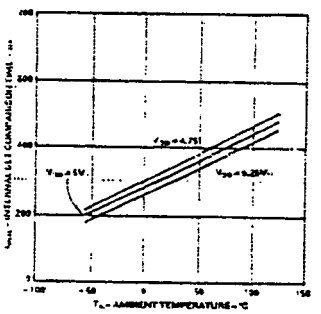
Accuracy vs. t_S



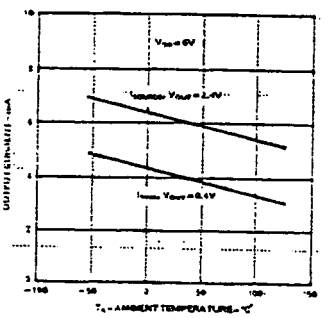
Accuracy vs. V_{REF} ($V_{REF} = V_{REF}(+) - V_{REF}(-)$)



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



t_{INTL}, Internal Time Delay vs. Temperature



Output Current vs. Temperature

AD7821

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.

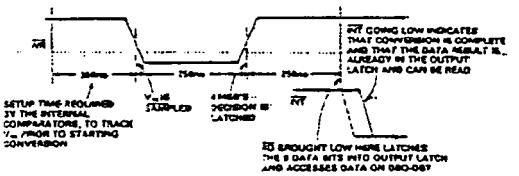


Figure 3. Operating Sequence (WR-RD Mode)

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.

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:

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

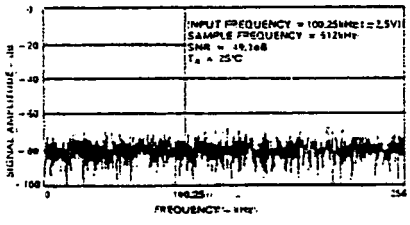


Figure 8. AD7821 FFT Plot

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.

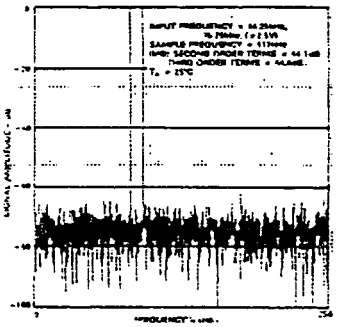


Figure 9. FFT Plot for IMD

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.

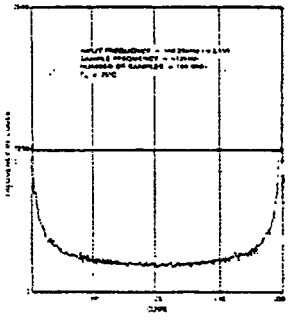


Figure 10. AD7821 Histogram Plot

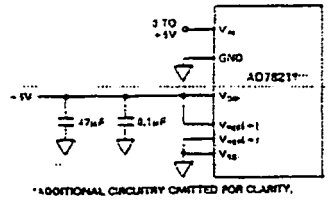


Figure 4. Power Supply as Reference. Unipolar Operation (0 to +5 V)

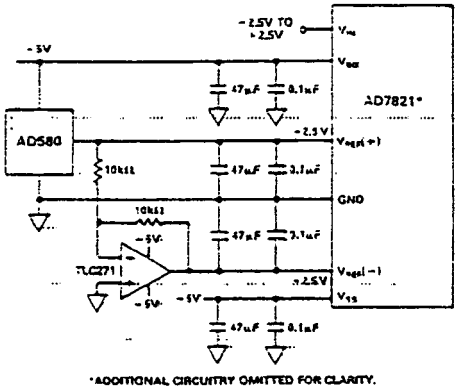


Figure 5. External Reference. Bipolar Operation (–2.5 V to +2.5 V)

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.

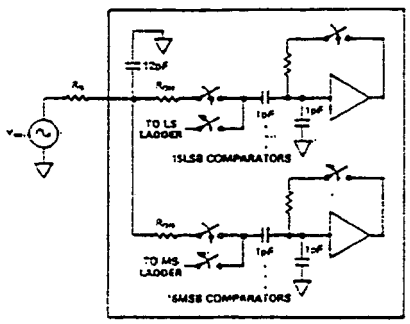


Figure 6. AD7821 Equivalent Input Circuit

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.

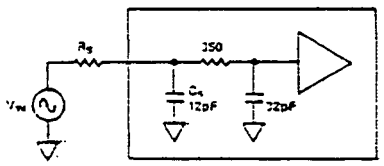


Figure 7. RC Network Model

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-

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.

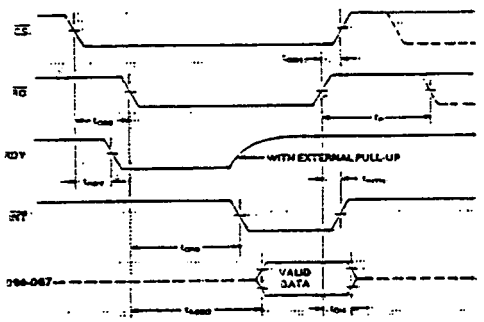


Figure 11. RD Mode

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 $\overline{CS}$.

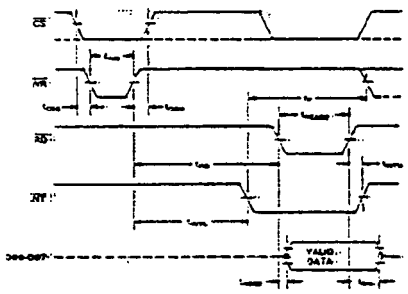


Figure 12a. WR-RD Mode ($t_{aD} > t_{INTL}$)

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.

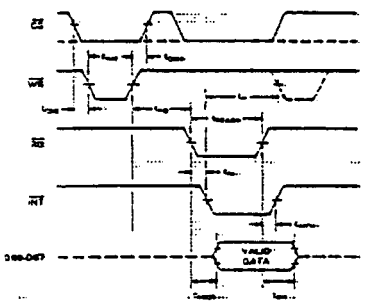


Figure 12b. WR-RD Mode ($t_{aD} < t_{INTL}$)

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 ($t_{INTL} + t_{1D}$) after the rising edge of $\overline{WR}$. The timing diagram for this mode is shown in Figure 13.

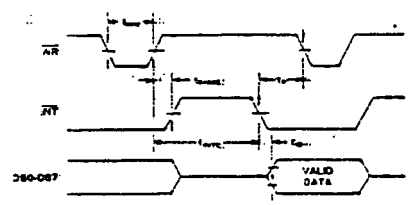


Figure 13. WR-RD Mode Stand-Alone Operation, $\overline{CS}$ = $\overline{RD}$ = 0

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.

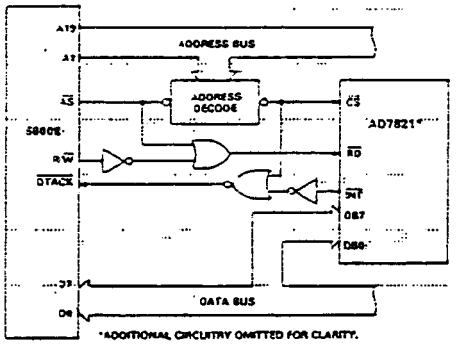


Figure 14. AD7821 to 68008 Interface

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.

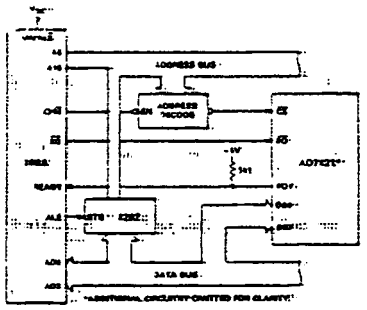


Figure 15. AD7821 to 8088 Interface

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).

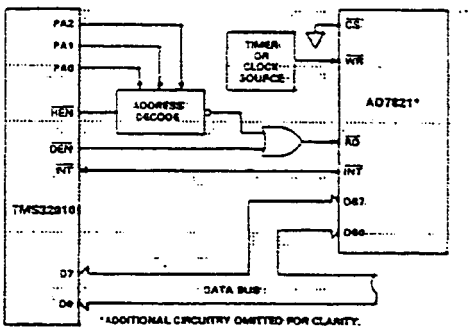


Figure 16. AD7821 to TMS32010 Interface

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).

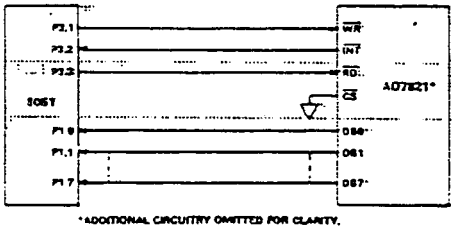


Figure 17. AD7821 to 8051 Interface

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.

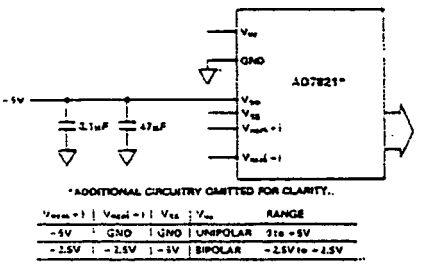


Figure 18. AD7821 Unipolar/Bipolar Operation

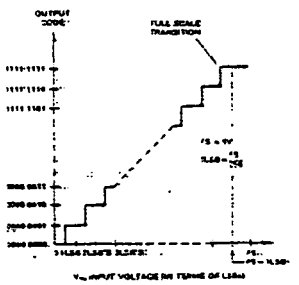


Figure 19. Nominal Transfer Characteristic for Unipolar (0 V to −5 V) Operation

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.

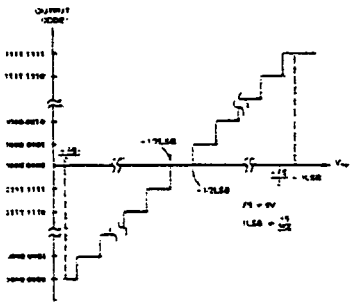


Figure 20. Nominal Transfer Characteristic for Bipolar (−2.5 V to +2.5 V) Operation

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.

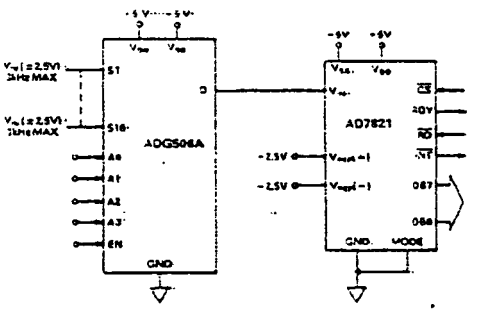


Figure 21. 16-Channel Telecom A/D Converter System

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.

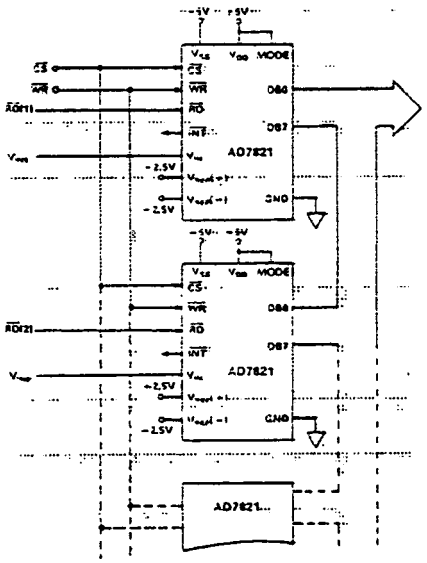


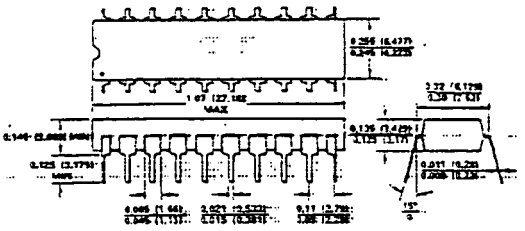
Figure 22. Simultaneous Sampling A/D Converters

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.

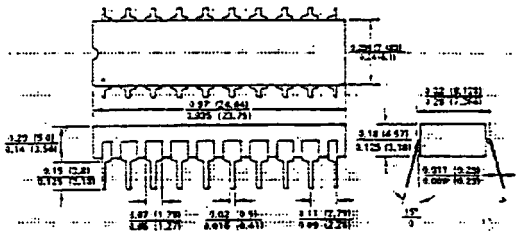
OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).

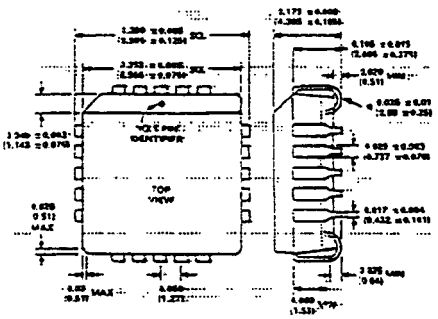
20-Pin Plastic DIP (N-20)



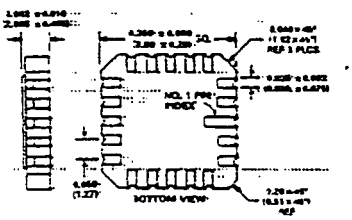
20-Pin Cerdip (Q-20)



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



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



PMI

ADC-908

CMOS MICROPROCESSOR-COMPATIBLE FAST 8-BIT A/D CONVERTER

Precision Monolithics Inc.

FEATURES

- 8-Bit Resolution and Accuracy
- No Missing Codes over Full Temperature Range
- 6μs Conversion Time
- Flexible μP Interface
- 2.5mA Maximum Standby Current
- Replaces AD7574 with Improved Speed
- Available in Die Form

ORDERING INFORMATION †

		PACKAGE: 18-PIN DIP AND SO		
INL (LSB)	DNL (LSB)	MILITARY* TEMPERATURE −55°C TO +125°C	EXTENDED INDUSTRIAL TEMPERATURE −40°C TO +85°C	COMMERCIAL TEMPERATURE 0°C TO +70°C
±1/2	±3/4	ADC908AX	ADC908EX	ADC908GP
±3/4	±7/8	ADC908BX	ADC908FX	—
±3/4	±7/8	—	ADC908FP	—
±3/4	±7/8	—	ADC908FS	—

* For devices processed in total compliance to MIL-STD-883, add /883 after part number. Consult factory for 883 data sheet.

† Burn-in is available on commercial and industrial temperature range parts in CerDIP, plastic DIP, and TO-can packages. For ordering information, see PMI's Data Book, Section 2.

GENERAL DESCRIPTION

The ADC-908 is a monolithic CMOS successive-approximation analog-to-digital converter. When used with a 1.35MHz clock, a conversion time of 6μs is achieved, with full accuracy over the operating temperature range.

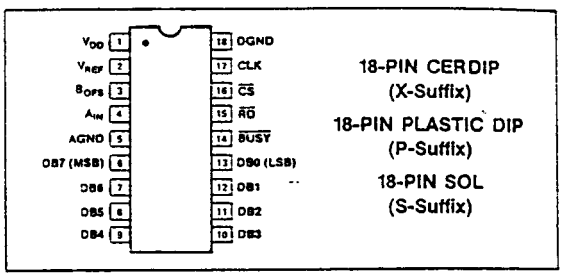
The ADC-908 outputs use 3-state logic, allowing direct connection to the data bus or system input port. Active-LOW chip select ($\overline{CS}$) and read/write ($\overline{RD}$) inputs are used to control all operations. This input structure permits the ADC-908 to be used as a memory-mapped input device. Depending on the control timing waveforms, the ADC-908 is interfaced like static RAM, ROM, or slow memory.

The low power consumption of the ADC-908 is derived from a single +5V supply. A negative reference voltage must also be supplied. Optimum accuracy is achieved when the reference is at −10.00V with a low output resistance. For a low-cost precision −10V/−10.24V reference, ask your PMI sales representative about the REF-08.

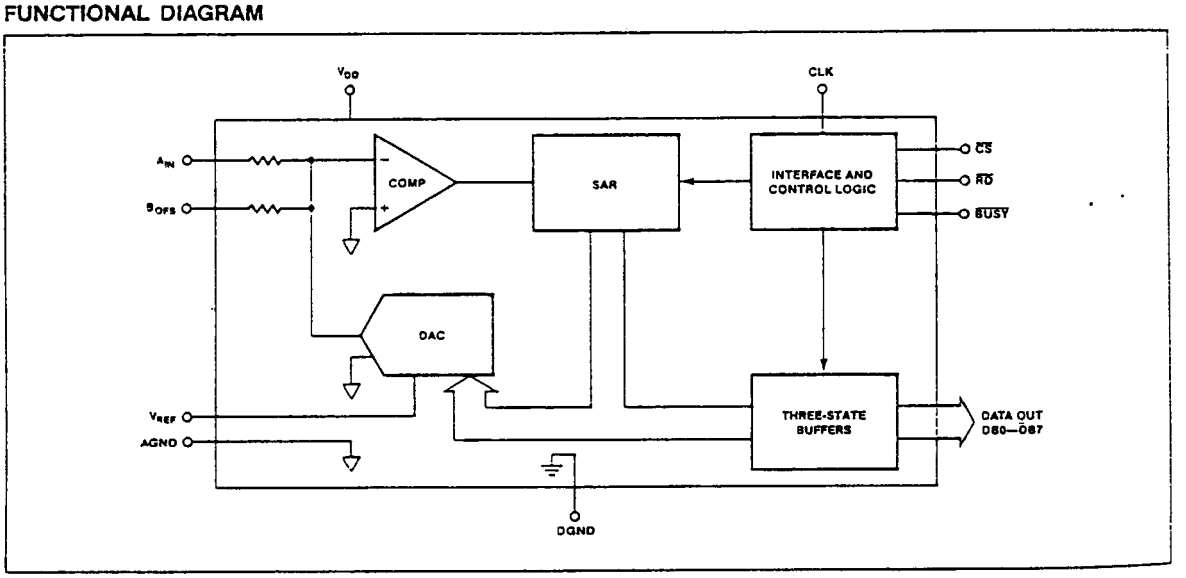
With its on-board comparator, interface logic, optional internal clock, and +5V operation, the ADC-908 is the ideal low-cost solution for microprocessor-based 8-bit A/D systems.

PMI's ADC-908 is pin-and-function compatible with the PM-7574, but offers faster conversion time and faster microprocessor bus interface timing. Conversion time has been reduced by 60% and most key timing specifications, including data access time, START command propagation delay (t_{WBPD}), and reset time, have been improved.

PIN CONNECTIONS



FUNCTIONAL DIAGRAM



ABSOLUTE MAXIMUM RATINGS (T_A = +25°C, unless otherwise noted)

V_{DD} to AGND ..... 0V, +7.0V
V_{DD} to DGND ..... 0V, +7.0V
AGND to DGND ..... –0.3V, V_{DD}
$\overline{CS}$, $\overline{RD}$ to DGND ..... –0.3V, V_{DD} + –.3V
DB_0-DB_7 to DGND ..... –0.3V, V_{DD}
CLK, $\overline{BUSY}$ to DGND ..... –0.3, V_{DD}
B_{OFS}, A_{IN} ..... ±20V
V_{REF} ..... 0V, –20V
Operating Temperature Range
ADC-908AX, BX ..... –55°C to +125°C
ADC-908EX, FX, FP, FS ..... –40°C to +85°C
ADC-908GP ..... 0°C to +70°C
Storage Temperature ..... –65°C to +150°C
Lead Temperature (Soldering, 10 sec) ..... +300°C

PACKAGE TYPE	θ_{JA} (Note 2)	θ_{JC}	UNITS
18-Pin Hermetic DIP (X)	79	11	°C/W
18-Pin Plastic DIP (P)	70	30	°C/W
18-Pin SOL (S)	88	25	°C/W

NOTES:
1. Digital pins are zener protected. However, proper ESD handling precautions are recommended.
2. θ_{JA} is specified for worst case mounting conditions, i.e., θ_{JA} is specified for device in socket for TO, CerDIP, P-DIP, and LCC packages; θ_{JA} is specified for device soldered to printed circuit board for SO and PLCC packages.

ELECTRICAL CHARACTERISTICS at V_{DD} = +5V, V_{REF} = –10V, Unipolar Configuration, R_{CLK} = 43kΩ, C_{CLK} = 100pF; –40°C ≤ T_A ≤ +85°C for ADC-908E/F, 0°C ≤ T_A ≤ +70°C for ADC-908G, –55°C ≤ T_A ≤ +125°C for ADC-908A/B, unless otherwise noted.

PARAMETER	SYMBOL	CONDITIONS	ADC-908 MIN	TYP	MAX	UNITS
ACCURACY						
Resolution	N		8	–	–	Bits
Integral Nonlinearity	INL	A/E/G Grades	–1/2	–	+1/2	LSB
		B/F Grades	–3/4	–	+3/4	
Differential Nonlinearity	DNL	A/E/G Grades	–3/4	–	+3/4	LSB
		B/F Grades	–7/8	–	+7/8	
Gain Error	G_{FSE}	A/E/G Grades T_A = +25°C	–3	–	+3	LSB
		T_A = Full Temp Range	–4.5	–	+4.5	
		B/F Grades T_A = +25°C	–5	–	+5	
		T_A = Full Temp Range	–6.5	–	+6.5	
Offset Error	V_{ZSE}	A/E/G Grades T_A = +25°C	–30	–	+30	mV
		T_A = Full Temp Range	–50	–	+50	
		B/F Grades T_A = +25°C	–60	–	+60	
		T_A = Full Temp Range	–80	–	+80	
ANALOG INPUTS						
Resistance Mismatch B_{OFS} to A_{IN}	ΔR_{AB}		–1	–	+1	%
Input Resistance at V_{REF} (Note 1)	R_{REF}		5	–	15	kΩ
Input Resistance at B_{OFS}, A_{IN}	R_{BOFS} R_{AIN}		10	–	30	kΩ
Reference Voltage Range	V_{REF}	Specified Conversion Accuracy	–	–10	–	V
Reference Voltage Range	V_{REF}	Degraded Conversion Accuracy	–5	–	–15	V
Reference Current (Note 6)	I_{REF}	Conversion Complete Prior to Reset	–	–	2.4	mA
Nominal Analog Input Range						
Unipolar Mode	V_{INU}		–	0 to +$\lvert V_{REF}\rvert$	–	V
Bipolar Mode	V_{INB}		–	–$\lvert V_{REF}\rvert$ to +$\lvert V_{REF}\rvert$	–	

ELECTRICAL CHARACTERISTICS at V_{DD} = +5V, V_{REF} = −10V, Unipolar Configuration, R_{CLK} = 43kΩ, C_{CLK} = 100pF; −40°C ≤ T_A ≤ +85°C for ADC-908E/F, 0°C ≤ T_A ≤ +70°C for ADC-908G, −55°C ≤ T_A ≤ +125°C for ADC-908A/B, unless otherwise noted. *Continued*

PARAMETER	SYMBOL	CONDITIONS	ADC-908 MIN	TYP	MAX	UNITS
LOGIC INPUTS						
Input HIGH Voltage $\overline{RD}$, $\overline{CS}$ Inputs	V_{IH}		2.4	—	—	V
Input LOW Voltage $\overline{RD}$, $\overline{CS}$ Inputs	V_{IL}		—	—	0.8	V
Input Current $\overline{RD}$, $\overline{CS}$ Inputs	I_{IN}	T_A = +25°C	—	—	1	µA
		T_A = Full Temp Range	—	—	10	
Input Capacitance $\overline{RD}$, $\overline{CS}$ Inputs (Note 6)	C_{IN}		—	—	5	pF
Input HIGH Voltage, Clock Input	V_{IH}		2.4	—	—	V
Input LOW Voltage, Clock Input	V_{IL}		—	—	0.8	V
Input HIGH Current, Clock Input	I_{IH}		—	—	2	mA
Input LOW Current, Clock Input	I_{IL}	T_A = +25°C	—	—	1	µA
		T_A = Full Temp Range	—	—	10	
LOGIC OUTPUTS						
Output HIGH Voltage $\overline{BUSY}$, DB0-7	V_{OH}	I_{SOURCE} = 40µA	4.0	—	—	V
Output LOW Voltage $\overline{BUSY}$, DB0-7	V_{OL}	I_{SINK} = 1.6mA	—	—	0.4	V
Floating Leakage Current, DB0-7	I_{LKG}	T_A = +25°C	—	—	1	µA
		T_A = Full Temp Range	—	—	10	
Floating State Output Capacitance	C_{OZ}	(Note 6)	—	—	7	pF
POWER REQUIREMENTS						
Standby Current	I_{DD}	V_{DD} = +4.75V to +5.25V	—	—	2.5	mA
DIGITAL INTERFACE TIMING						
$\overline{CS}$ Minimum Pulse Width (Note 6)	t_{CS}	T_A = +25°C	60	—	—	ns
		T_A = T_{MIN}	50	—	—	
		T_A = T_{MAX}	90	—	—	
$\overline{RD}$ to $\overline{CS}$ Setup Time (Note 6)	t_{WSCS}		0	—	—	ns
$\overline{CS}$ to $\overline{BUSY}$ Propagation Delay (Note 6)	t_{CBPD}	$\overline{BUSY}$ Load = 20pF				ns
		T_A = +25°C	—	—	120	
		T_A = T_{MIN}	—	—	100	
		T_A = T_{MAX}	—	—	150	
		$\overline{BUSY}$ Load = 100pF				
		T_A = +25°C	—	—	150	
		T_A = T_{MIN}	—	—	120	
		T_A = T_{MAX}	—	—	200	
$\overline{BUSY}$ to $\overline{RD}$ Setup Time (Notes 2, 6)	t_{SSR}		0	—	—	ns
$\overline{BUSY}$ to $\overline{CS}$ Setup Time (Note 6)	t_{SSCS}		0	—	—	ns

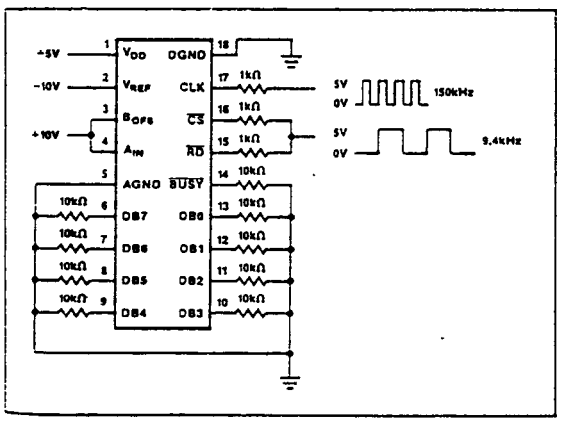
ELECTRICAL CHARACTERISTICS at V_{DD} = +5V, V_{REF} = −10V, Unipolar Configuration, R_{CLK} = 43kΩ, C_{CLK} = 100pF; −40°C ≤ T_A ≤ +85°C for ADC-908E/F, 0°C ≤ T_A ≤ +70°C for ADC-908G, −55°C ≤ T_A ≤ +125°C for ADC-908A/B, unless otherwise noted. *Continued*

PARAMETER	SYMBOL	CONDITIONS	ADC-908 MIN	TYP	MAX	UNITS
Data Access Time (Note 6)	t_{RAD}	C_L = 20pF				ns
		T_A = +25°C	—	—	140	
		T_A = T_{MIN}	—	—	100	
		T_A = T_{MAX}	—	—	200	
		C_L = 100pF				
		T_A = +25°C	—	—	170	
		T_A = T_{MIN}	—	—	150	
		T_A = T_{MAX}	—	—	230	
Data Hold Time (Notes 3, 6)	t_{RHD}	T_A = +25°C (Note 3)	30	—	100	ns
		T_A = T_{MIN}	20	—	70	
		T_A = T_{MAX}	40	—	140	
$\overline{CS}$ to $\overline{RD}$ Hold Time (Note 6)	t_{RHCS}	T_A = +25°C	—	—	200	ns
		T_A = T_{MIN}	—	—	120	
		T_A = T_{MAX}	—	—	250	
Reset Time Requirement (Note 6)	t_{RESET}	T_A = +25°C	450	—	—	ns
		T_A = Full Temp. Range	500	—	—	
Conversion Time (Note 4) (Notes 4, 5, 6)	$t_{CONVERT}$	Static RAM Mode External Clock f = 1.35MHz	—	—	6	µs
		ROM Mode Internal Clock	—	—	7	
$\overline{RD}$ HIGH to $\overline{BUSY}$ Propagation Delay, ROM Mode (Notes 4, 5, 6)	t_{WBPD}	C_L = 20pF				ns
		T_A = +25°C	—	—	600	
		T_A = T_{MIN}	—	—	400	
		T_A = T_{MAX}	—	—	800	

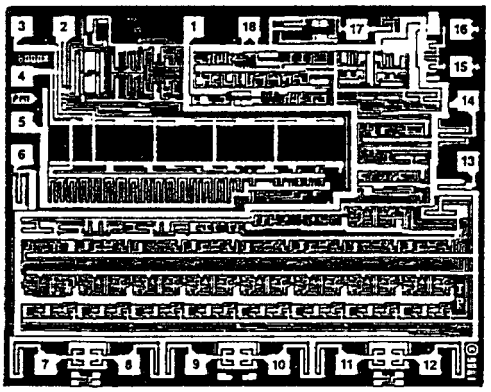
NOTES:

1. For optimum gain accuracy over the full temperature range, the source resistance at pin 2 should be kept low.
2. In ROM mode, $\overline{RD}$ can go LOW prior to $\overline{BUSY}$ = HIGH, but must not return HIGH until $\overline{BUSY}$ = HIGH.
3. Output loading 10pF. A 3kΩ pullup resistor to +5V is used for V_{OL} to High-Z, for V_{OH} to High-Z, a 3kΩ pulldown to GND is used. Measured to 0.5V output change.
4. When using the ADC-908 internal oscillator, actual conversion time depends on clock resistor and capacitor as well as temperature.
5. ROM interface mode conversion times are typically 1µs longer than conversion times for other modes, but the ROM interface mode includes an automatic reset in the conversion time.
6. Guaranteed but not tested.

BURN-IN CIRCUIT



DICE CHARACTERISTICS



DIE SIZE 0.129 X 0.103 inch, 13,287 sq. mils
(3.28 X 2.62 mm, 8.58 sq. mm)

1. V_{DD}
2. V_{REF}
3. B_{OFS}
4. A_{IN}
5. AGND
6. DB7(MSB)
7. DB6
8. DB5
9. DB4
10. DB3
11. DB2
12. DB1
13. DB0(LSB)
14. $\overline{BUSY}$
15. $\overline{RD}$
16. $\overline{CS}$
17. CLK
18. DGND

For additional DICE ordering infromation, refer to PMI's Data Book, Secion 2.

WAFER TEST LIMITS at V_{DD} = +5V, V_{REF} = −10.000V, AGND = DGND = 0V, T_A = +25°C, unless otherwise noted.

PARAMETER	SYMBOL	CONDITIONS	ADC-908 LIMIT	UNITS
STATIC ACCURACY				
Resolution	N		8	Bits MIN
Integral Nonlinearity	INL		±3/4	LSB MAX
Differential Nonlinearity	DNL		±7/8	LSB MAX
Gain Error	G_{FSE}		±5	LSB MAX
Offset Error	V_{ZSE}		±60	mV MAX
ANALOG INPUTS				
Resistance Mismatch B_{OFS} to A_{IN}	ΔR_{AB}		±1	% MAX
Input Resistance at V_{REF}	R_{REF}		5/15	kΩ MIN/MAX
Input Resistance at B_{OFS}, A_{IN}	R_{BOFS}, R_{IN}		10/30	kΩ MIN/MAX
DIGITAL INPUTS				
Input HIGH Voltage at $\overline{RD}$, $\overline{CS}$ Inputs	V_{IH}		2.4	V MIN
Input LOW Voltage at $\overline{RD}$, $\overline{CS}$ Inputs	V_{IL}		0.8	V MAX
Input Current $\overline{RD}$, $\overline{CS}$ Inputs	I_{IN}		±1	µA MAX
Input HIGH Voltage Clock Input	V_{IH}		2.4	V MIN
Input LOW Voltage Clock Input	V_{IL}		0.8	V MAX
Input HIGH Current Clock Input	I_{IH}		2	mA MAX
Input LOW Current Clock Input	I_{IL}		1	µA MAX

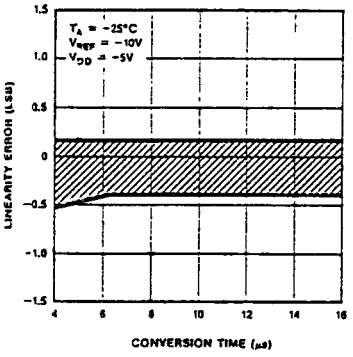
WAFER TEST LIMITS at V_{DD} = +5V, V_{REF} = −10.000V, AGND = DGND = 0V, T_A = +25°C, unless otherwise noted. (Continued)

PARAMETER	SYMBOL	CONDITIONS	ADC-908 LIMIT	UNITS
DIGITAL OUTPUTS				
Output HIGH Voltage $\overline{BUSY}$, DB0-7	V_{OH}	I_{SOURCE} = 40μA	4	V MIN
Output LOW Voltage $\overline{BUSY}$, DB0-7	V_{OL}	I_{SINK} = 1.6mA	0.4	V MAX
Floating Leakage Current	I_{LKG}		1	μA
POWER REQUIREMENTS				
Standby Current	I_{DD}	V_{DD} = +4.75V to 5.25V	2.5	mA MAX
TIMING				
Conversion Time	$t_{CONVERT}$	Static RAM Mode, External Clock, f = 1.35MHz	6	μs MAX

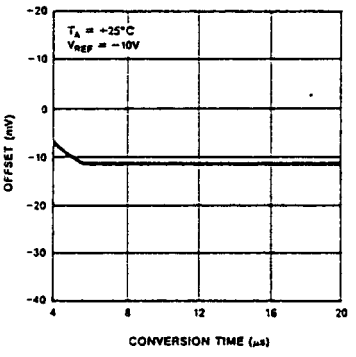
NOTE:
Electrical tests are performed at wafer probe to the limits shown. Due to variations in assembly methods and normal yield loss, yield after packaging is not guaranteed for standard product dice. Consult factory to negotiate specifications based on dice lot qualification through sample lot assembly and testing.

TYPICAL PERFORMANCE CHARACTERISTICS

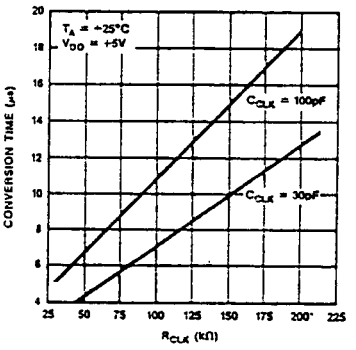
LINEARITY ERROR vs CONVERSION TIME



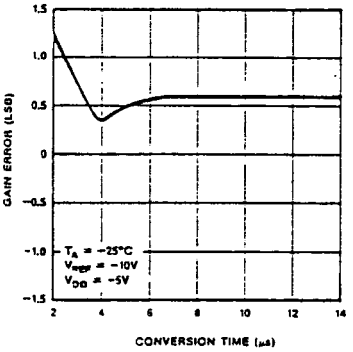
OFFSET ERROR vs CONVERSION TIME RAM MODE



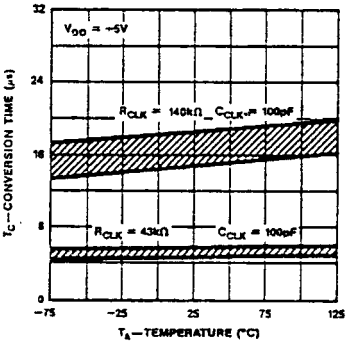
CONVERSION TIME vs R_{CLK} AND C_{CLK}



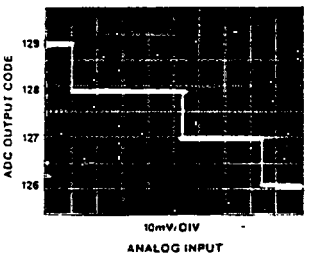
GAIN ERROR vs CONVERSION TIME



CONVERSION TIME vs TEMPERATURE WITH INTERNAL CLOCK



CONVERSION UNCERTAINTY AROUND MAJOR CARRY



GENERAL CIRCUIT INFORMATION

The ADC-908 is an 8-bit analog-to-digital converter which uses a successive approximation technique to convert an unknown analog input into a digital code output. The control logic inputs allow easy interface to most microprocessors while three-state outputs allow direct connection to the data bus. Most applications require only passive RC clock components, a −10V reference, and a +5V power supply. The RC-timed internal clock may be used, or an external clock may be applied to the ADC to maximize performance.

When a Start Conversion command is applied to the $\overline{CS}$ or $\overline{RD}$ inputs (see Operating Descriptions for details), $\overline{BUSY}$ goes LOW indicating a conversion in progress. $\overline{BUSY}$ may be used as an interrupt to halt the controlling microprocessor during conversion or may be polled to prevent premature data reads.

Starting with the most significant bit (MSB), each successive bit in the DAC is turned on (see Figure 1). The comparator then decides if the DAC output is less than or greater than the signal being converted, and that bit is latched on or off, respectively, before proceeding to the next lower bit and repeating the cycle. When all eight bits have been tested, $\overline{BUSY}$ goes HIGH, signaling a completed conversion.

Under control of the $\overline{RD}$ input, the three-state data outputs (D0-D7) change from high-impedance to presenting the new conversion results to the data bus. Following the data read, $\overline{RD}$ returns HIGH resetting the SAR to 1000 0000 and preparing the ADC for its next conversion.

PIN FUNCTIONS

NOTE: For greater detail on digital input functions, consult Truth Tables and Timing Diagrams.

Pin 1. V_{DD} Power Supply input, +5V.

Pin 2. V_{REF} Voltage Reference input, nominal −10V.

Pin 3. B_{OFS} Bipolar Offset input. +10V input for bipolar mode operation, tie to V_{IN} for unipolar mode operation.

Pin 4. A_{IN} Analog Input. 0V to +10V in unipolar mode, −10V to +10V in bipolar mode.

Pin 14. $\overline{BUSY}$ Conversion status output. BUSY indicates conversion in progress by going LOW at start of conversion and returning HIGH at end of conversion. May be used to interrupt controlling microprocessor or to gate control inputs.

Pin 15. $\overline{RD}$ READ input. Used to read data (on falling edge) and to reset converter (on rising edge).

Pin 16. $\overline{CS}$ Chip Select input. Asserted to allow ADC operation. Starts conversion when converter is in reset condition. Note: Holding $\overline{CS}$ HIGH will not prevent a rising edge on $\overline{RD}$ from resetting the converter.

Pin 17. CLK External clock input/internal clock RC timing input.

APPLICATIONS INFORMATION

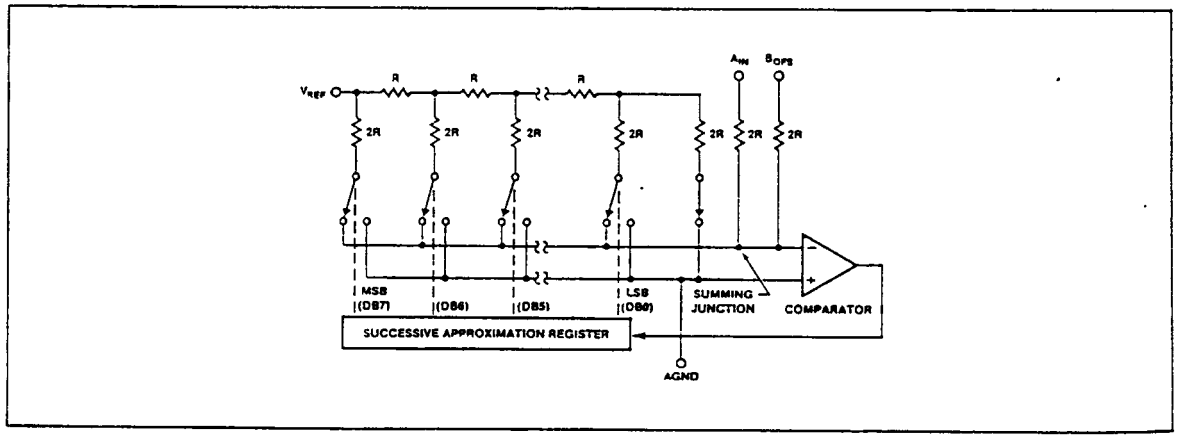
The ADC-908 may be interfaced as if it were a static RAM, a ROM, or a slow-memory device. Each of these interface modes has its own timing and software requirements as described below. These requirements must be rigidly met, as improper timing may cause the ADC-908 to change modes.

HOW TO CHOOSE AN OPERATING MODE

The static-RAM interface mode offers advantages in a tightly controlled hardware and software environment, where the relationship between WRITE and READ instruction pairs is certain. As long as minimum timing is satisfied, converted data may be read at any convenient time after conversion. The use of separate commands to start a conversion, and then read the results, is conceptually easy. However, if the software is subject to uncontrolled modifications, then the paired relationship between WRITE and READ instructions may be lost. Resulting software bugs may result in converted data of unknown age, or altogether invalid data being read.

By contrast, the ROM mode may be more resistant to software bugs. As long as minimum timing is satisfied, each READ instruction obtains new, valid data. However, since the data

FIGURE 1: D/A Converter Used in ADC-908



output at any previous READ instruction is obtained from a conversion performed just after the previous READ instruction, data may be out-of-date. To be sure of obtaining up-to-date data, READ instructions may be coded in pairs (with some NOPs between them); use only the data from the second READ in each pair. The first READ starts the conversion, acting as a substitute for the static-RAM mode WRITE command; the second READ gets the results. The advantage of the ROM mode is the use of a single command, rather than the alternating READ-WRITE required by static-RAM mode.

The slow-memory mode is the simplest mode of all. It is the method of choice where compact coding is essential, or where software bugs are a hazard. In this mode, a single READ instruction will initiate a data conversion, interrupt the microprocessor until completion (WAIT states are introduced), then read the results. If the system throughput tolerates WAIT states, and the hardware is correct, then the slow-memory mode is virtually immune to subsequent software modifications.

OPERATING DESCRIPTION: STATIC-RAM MODE

In this mode, input $\overline{CS}$ is derived from the ADC-908 address decoder, and input $\overline{RD}$ is derived from an active-LOW memory READ signal. (See Figure 2.)

To start a conversion, execute a memory WRITE to the ADC-908. The completed conversion data is obtained by executing a memory READ to the ADC-908. During conversion, output $\overline{BUSY}$ will be LOW. Do not attempt to read data until $\overline{BUSY}$ returns HIGH. The required minimum time between WRITE and READ is usually obtained by including one or more NOP or other program instructions. The use of branch or conditional commands between the WRITE and READ instructions is not recommended due to the possibility of software bugs.

It is important that the WRITE and READ commands be alternately executed. A WRITE instruction has no effect unless the results of the previous WRITE have already been read. Once data has been read, the ADC-908 is internally reset. In other words, two or more READ operations cannot be used in succession, since only the first READ will produce valid data. A new conversion must be started using WRITE, and the conversion must be completed, before a new READ will produce valid data.

TABLE 1: Truth Table, Static RAM Mode

INPUTS $\overline{CS}$	INPUTS $\overline{RD}$	OUTPUTS $\overline{BUSY}$	OUTPUTS DB7-DB0	ADC-908 OPERATION
L	H	H	HIGH-Z	Start Convert (Write Cycle)
L	↓	H	HIGH-Z to DATA	Read Data (Read Cycle)
L	↑	H	DATA to HIGH-Z	Reset Converter
H	X (Note 1)	X	HIGH-Z	No Effect (Not Selected)
L	H	L	HIGH-Z	No Effect (Converter Busy)
L	↓	L	HIGH-Z	No Effect (Converter Busy)
L	↑ (Note 1)	L	HIGH-Z	Conversion Error Not Allowed

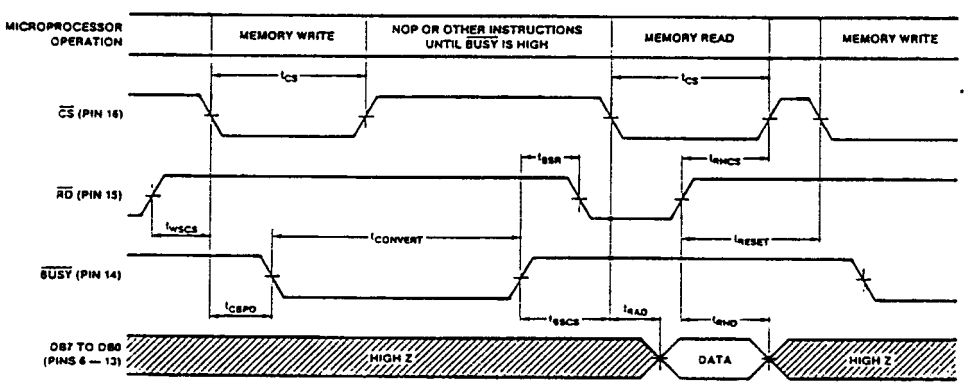
NOTE 1: If $\overline{RD}$ goes LOW to HIGH, the ADC is internally reset, regardless of the states of $\overline{CS}$ or $\overline{BUSY}$.

OPERATING DESCRIPTION: ROM MODE

In ROM mode, input $\overline{CS}$ is tied LOW, and input $\overline{RD}$ is derived from the ADC-908 address decoder. To satisfy timing, it is recommended that the decoder be enabled by a system MEMRD (8080), VMA (6800), or similar strobe. (See Figure 3.)

In ROM mode, data is read by executing a READ instruction to the ADC-908 address. At the conclusion of the READ instruction, the ADC-908 automatically resets itself and then proceeds to perform a new data conversion. Output $\overline{BUSY}$ is LOW during conversion. A new READ instruction to the ADC-908 must not be executed until $\overline{BUSY}$ returns HIGH.

FIGURE 2: Static RAM Mode Timing Diagram



This requirement may be met by inserting NOP or other program instructions between consecutive READ operations. Conditional or branch instructions may be used, but keep in mind that data may become out-of-date if excessive time elapses between consecutive READ instructions.

TABLE 2: Truth Table, ROM Mode

INPUTS $\overline{CS}$	INPUTS $\overline{RD}$	OUTPUTS $\overline{BUSY}$	OUTPUTS DB7-DB0	ADC-908 OPERATION
L	↓	H	HIGH-Z to DATA	Read Data
L	↑	↓	DATA to HIGH-Z	Reset and Start New Conversion
L	↓	L	HIGH-Z	No Effect (Converter Busy)
L	↑ (Note 1)	L	HIGH-Z	Conversion Error Not Allowed

NOTE 1: If $\overline{RD}$ goes LOW to HIGH, the ADC is internally reset, regardless of the states of $\overline{CS}$ or $\overline{BUSY}$.

OPERATING DESCRIPTION: SLOW-MEMORY MODE

The slow-memory mode is intended for systems in which the ADC-908 $\overline{BUSY}$ output is used as an interrupt to force the microprocessor into WAIT states during data conversion.

In slow-memory mode, inputs $\overline{CS}$ and $\overline{RD}$ are tied together. The common $\overline{RD}$ and $\overline{CS}$ signal is derived from the ADC-908 address decoder. To satisfy the timing requirements, it is advisable to latch the address using ALE (8085) or SYNC (8080). For 8080 or 8085-based systems, connect the microprocessor READY input to the ADC-908 $\overline{BUSY}$ output. (See Figure 4.)

TABLE 3: Truth Table, Slow-Memory Mode

INPUTS $\overline{CS}$ & $\overline{RD}$	OUTPUTS $\overline{BUSY}$	OUTPUTS DB7-DB0	ADC-908 OPERATION
H	H	HIGH-Z	No Effect (Not Selected)
↓	↓	HIGH-Z	Start Conversion
L	L	HIGH-Z	Conversion in Progress. μP in WAIT State
L	↑	HIGH-Z to DATA	Conversion Complete. Read Data
↑	H	DATA to HIGH-Z	Reset and Deselect Converter

FIGURE 3: ROM Mode Timing Diagram ($\overline{CS}$ Held LOW)

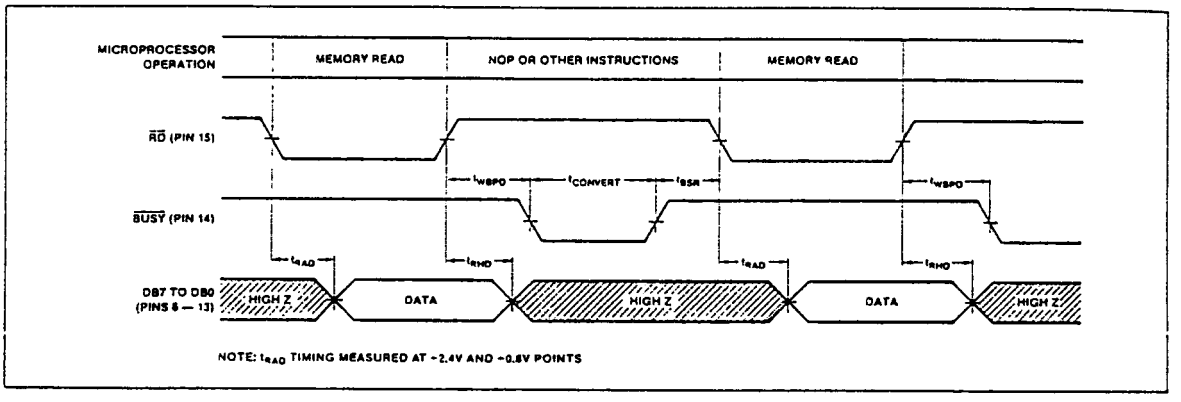
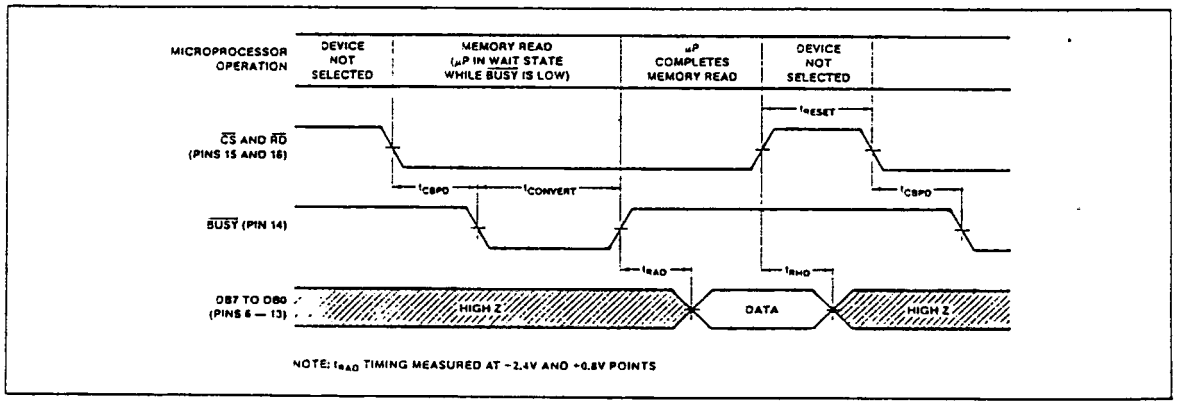


FIGURE 4: Slow-Memory Mode Timing Diagram ($\overline{CS}$ and $\overline{RD}$ Tied Together)



Do not execute a WRITE instruction at the ADC-908 address when in slow-memory mode, since bus conflicts will arise. In some architectures, an accidental WRITE instruction may be locked out in hardware, by proper strobing of the ADC-908 address decoder.

INITIALIZATION

In all operating modes, the ADC-908 is initialized by executing a READ instruction to the ADC-908 address. The data obtained should be ignored.

CLOCK OSCILLATOR

The ADC-908 may be used with its internal asynchronous clock oscillator. An external resistor and capacitor are required. Typical values are R = 43kΩ and C = 100pF, for conversion times in the 6μs range. For applications in which the fastest conversion times are required, an external clock is recommended. The external clock must be gated by the use of a 74125-type three-state buffer, with an output pullup resistor. Optimum conversion accuracy is obtained when $\overline{CS}$ goes LOW on a positive clock edge. The maximum external clock frequency is 1.35MHz (See Figure 5 and 6.)

REFERENCE VOLTAGE

A negative reference voltage must be applied to the ADC-908 V_{REF} input. Optimum full-scale accuracy is obtained using −10.00V, although V_{REF} may be −5.00V, −10.24V, or other voltages within its specified range.

Over the full temperature range, optimum gain accuracy is obtained when the input to the V_{REF} pin is from a low-impedance source. A resistor or trimmer may be used in series with the V_{REF} pin, but this trim technique is not as accurate as a low-impedance source. (See Figure 7.)

For a cost-effective −10.00V or −10.24V reference with excellent accuracy and low temperature coefficient, ask for PMI's REF-08. Consult your sales representative for availability.

FIGURE 5: Using the Internal Clock Oscillator

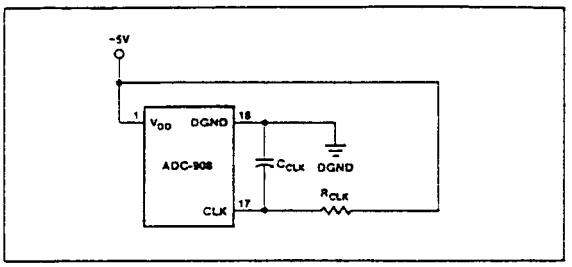


FIGURE 6: Using an External Clock

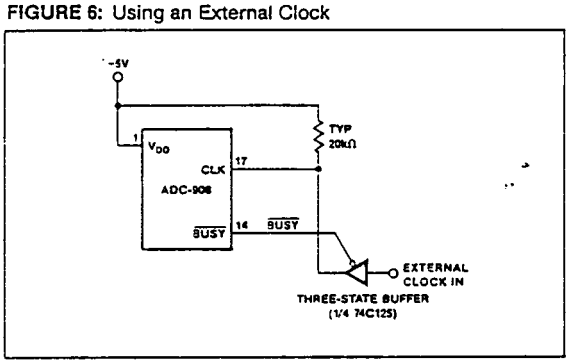
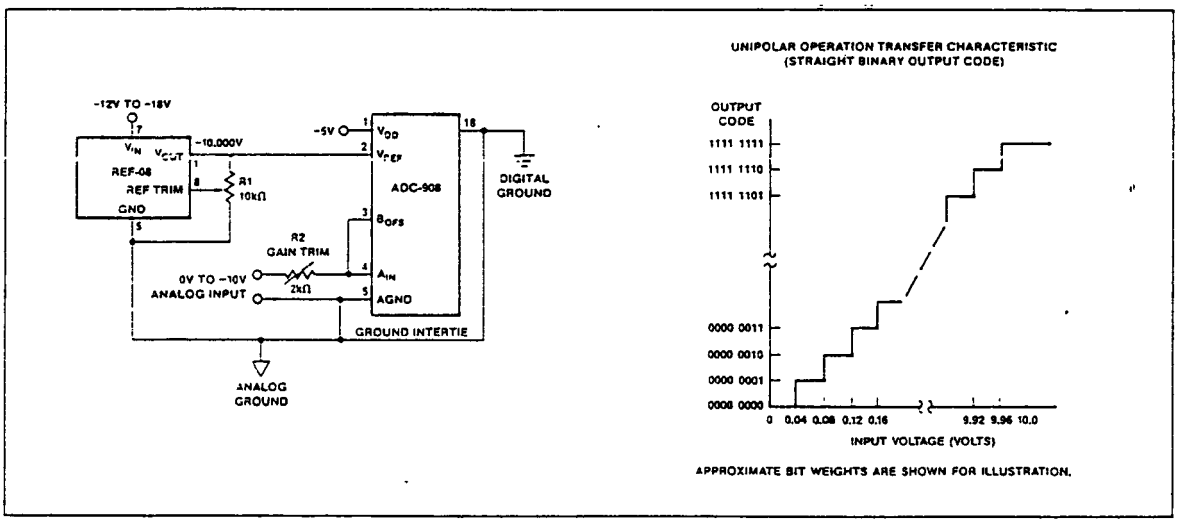


FIGURE 7: Unipolar Operation



ANALOG INPUT VOLTAGE

The ADC-908 unipolar operation is obtained when the analog input voltage is between 0V and $|V_{REF}|$. With the A_{IN} and B_{OFS} pins tied together, input 0V will correspond to code 0000 0000, and input full-scale will correspond to code 1111 1111.

Bipolar operation is obtained by using the B_{OFS} input to offset the A_{IN} input voltage. For example, with $V_{REF} = -10V$, an offset voltage of +10V may be applied to B_{OFS}. The analog signal range will then be −10V to +10V at A_{IN}. Code 0000 0000 will correspond to −10V, and positive full scale will be code 1111 1111. Calibration may be performed using trimmers in series with A_{IN} and B_{OFS}. (See Figure 8).

Another method of obtaining bipolar operation is to use an op-amp with gain = −1/2, to sum the analog signal with the reference voltage. With a −10V reference and −10V to +10V analog signal, the op amp output will then be 0V to +10V. This signal is then treated as an ordinary unipolar input to the ADC-908. With this arrangement, input +10V corresponds to code 0000 0000, and negative full-scale corresponds to code 1111 1111.

UNIPOLAR BINARY OPERATION

Figure 7 shows the analog circuit connections for unipolar operation. The REF-08 supplies the necessary −10V reference input.

Calibration for offset should be made before gain calibration is attempted.

Offset calibration must be performed in the signal conditioning circuitry which drives the A_{IN} input.

To adjust offset:

1) Apply −39.1mV (1 LSB) to the input of the buffer amplifier driving A_{IN}.
2) While performing continuous conversions, adjust the buffer amplifier's offset adjustment potentiometer until DB7 to DB1 are LOW and DB0 (LSB) flickers.

Following offset calibration, full scale gain can be calibrated:

1) Apply −9.961V to the input of the buffer amplifier.
2) While performing continuous conversions, adjust the reference trim pot until DB7 to DB1 are HIGH, and DB0 (LSB) flickers.

BIPOLAR OPERATION

Offset Binary—Figure 8 shows a circuit for offset binary bipolar operation. Offset correction should be made at the buffer amplifier driving A_{IN}. Gain error correction should be accomplished by adjusting V_{REF}.

To calibrate this circuit:

1) Adjust R1 until $V_{REF} = -10.00V$.
2) Adjust R2 and R3 to their mid-points.
3) Apply +10.000V to the input buffer amplifier.
4) While performing continuous conversions, adjust R2 until DB7 to DB1 are LOW and DB0 (LSB) flickers.
5) Ground the input of the input buffer circuit.
6) While performing continuous conversions, adjust R3 until the ADC's output code flickers between 0111 1111 and 1000 0000.
7) Apply −10.000V to the signal input.
8) While performing continuous conversions, adjust R1 until DB7 to DB1 are LOW and the DB0 (LSB) flickers.
9) Apply +9.922V to the signal input.
10) If the ADC output code is not 1111 1110 ±1 bit, repeat the calibration procedure, omitting step 1.

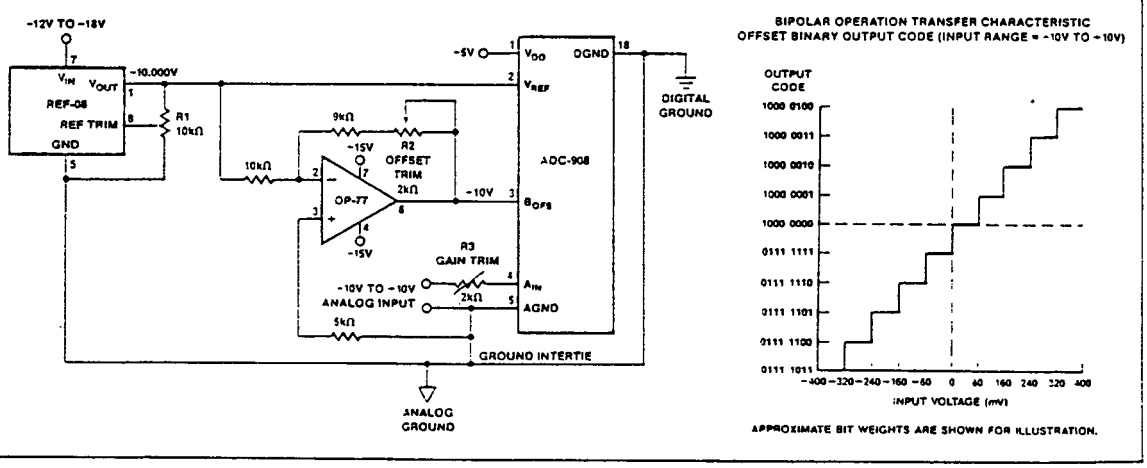
Complementary Offset Binary—Figure 9 shows a complementary offset binary circuit. In this bipolar mode, the +10V to −10V analog input is conditioned to a 0 to +10V signal range for normal unipolar conversion.

In calibrating this circuit, adjust offset before gain.

Offset Adjustment:

1) Adjust R1 until $V_{REF} = -10.000V$.
2) Adjust R3 to its mid-point.
3) Adjust R2 until its tap is at 0V.
4) Ground the analog input.
5) While performing continuous conversions, adjust R2 until the ADC output flickers between 0111 1111 and 1000 0000.

FIGURE 8: Offset Binary Operation



Gain Adjustment:

1) Apply +9.922V across the analog input.
2) While performing continuous conversions, adjust R3 until DB7 to DB1 are HIGH and DB0 (LSB) flickers.

DIGITAL CONSIDERATIONS

Control Timing—Fresh data from a recent conversion must be read before beginning a new conversion. Following the data READ, as $\overline{RD}$ goes HIGH, it resets the SAR and clears the data from the previous conversion.

The timing restrictions detailed in the interface timing diagrams must be observed to prevent the ADC-908 from changing interface modes. For example, if $\overline{CS}$ is held LOW too long while in RAM mode, the converter will change to ROM mode and initiate a new conversion.

Logic Deglitching—Unrelated activity on the address bus may cause unexpected glitch inputs to the ADC. The glitches may cause unwanted READs, resets, or conversions. In ROM or RAM modes, these may be avoided by gating the address decode logic with $\overline{RD}$ or $\overline{WR}$ (8080) or VMA (6800). In slow-memory mode, ALE (8085) or SYNC (8080) may be used to latch the address.

Initialization—Following power-up, the SAR is in an unknown state. Executing a memory READ (disregard the data) will reset the ADC.

ANALOG CONSIDERATIONS

Analog Input Impedances—Low impedance sources must be used to drive the V_{REF}, A_{IN}, and B_{OFS} inputs. Excessive source impedances may cause errors due to the loading effects of the inputs' finite impedances.

Ground Management—AGND and DGND pins should be connected at or near the ADC to minimize noise effects. If the two grounds cannot be connected near the ADC, the grounds should be clamped with back-to-back Schottky diodes between the AGND and DGND pins.

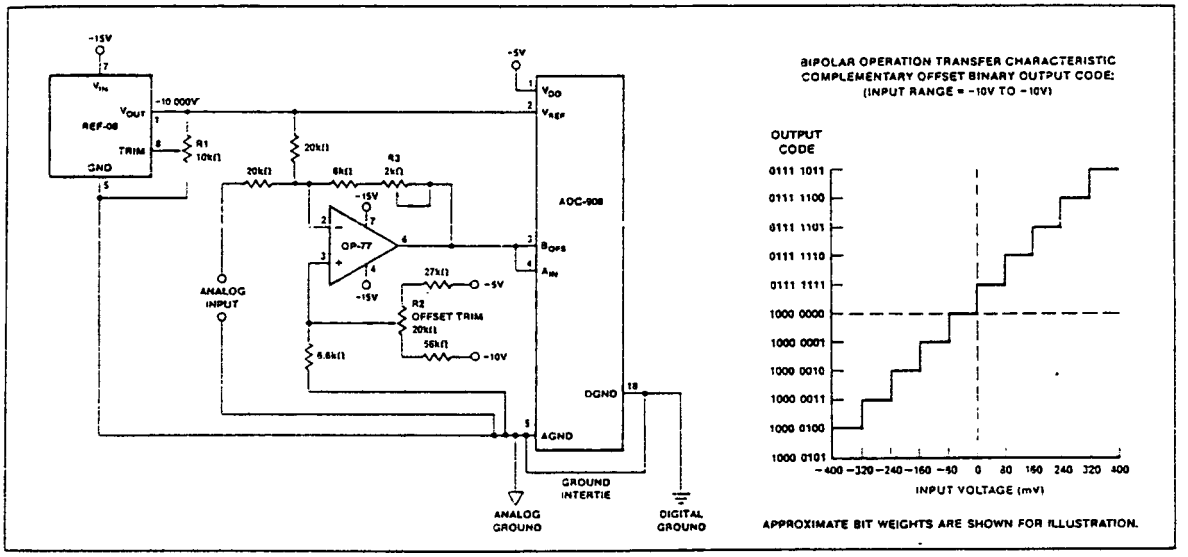
Offset Correction—Conversion offset errors may be corrected by counter-offsetting the buffer amplifier driving A_{IN}. This offset correction may be accomplished by applying a correction current to the buffer's summing junction or by tapping a voltage divider sitting between V_{DD} and V_{REF}, and applying this tap voltage to the noninverting input of the buffer.

Ratiometric Operation—The R-2R type DAC in the ADC-908 permits ratiometric operation of the ADC. Performance degradation may, however, occur as V_{REF} varies from −10.000V. This decrease in performance is due to comparator limitations including offset-voltage, gain, and input noise.

The ADC-908 uses the reference as a power supply for the comparator to increase speed and accuracy. Reference voltages of a magnitude less than −9V must be avoided for accurate comparator operation. For best accuracy, the use of a 0.1μF bypass capacitor from V_{REF}(Pin 2 to AGND) is recommended.

Power Supply Bypassing—For best accuracy, V_{DD} (Pin 1) should be bypassed to AGND with a 0.1μF capacitor.

FIGURE 9: Complementary Offset Bipolar Operation



บรรณานุกรม

[1] บรรจง ปียธำรง และสุรเชษฐ์ ศรีพลกรัง, การออกแบบระบบดิจิตอลโดยใช้ VHDL : การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 8

[2] วัลลภ สุระกำพลธร, การประมวลผลสัญญาณเชิงเลข , สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2533

[3] วินัย ทองตัน และคณะ, การออกแบบและสร้างวงจรกรองสัญญาณเชิงเลขแบบบัตเตอร์เวอร์ธอันดับที่ 6, วิศวสารลาดกระบัง, ปีที่ 13 ฉบับที่ 1, กรกฎาคา 2539

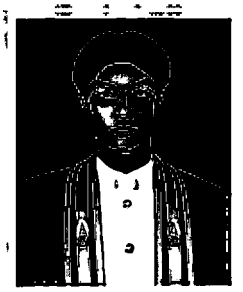
[4] Douglas L. Perry. VHDL : McGraw-Hill,Inc.1993

[5] Jayaram Bhasker, A VHDL Primer : Pracitce-Hall Inc 1993

[6] Jean-Michael Berge and Rolan Airiau, Circuit Synthesis with VHDL : Kluwer Acadamic Publishers. 1994

[7] Xiling Inc, Schematic-Base Programmable Logic Training Course,1996

ประวัติผู้แต่ง

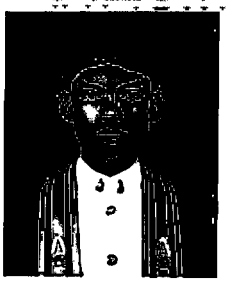


ชื่อผู้ทำปริญญานิพนธ์	นายทองปาน ปริวัตร
วันเดือนปีเกิด	7 ตุลาคม 2518
สถานที่เกิด	จังหวัดหนองคาย
ภูมิลำเนาเดิม	575 ถ.ประจักษ์ อ.เมือง จ.หนองคาย 43000
ที่อยู่ปัจจุบัน	หอพักชายยุคลรัตน์ 333 ม.1 ถ.ลาดกระบัง เขตลาดกระบัง กรุงเทพมหานคร 10520
โทรศัพท์	(02) 7390255-8 ห้อง 200

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนเทศบาล 1 สว่างวิทยา
มัธยมศึกษาตอนต้น	โรงเรียนปทุมเทพวิทยาคาร
ประกาศนียบัตรวิชาชีพ(ปวช.)	วิทยาลัยเทคนิคหนองคาย
ประกาศนียบัตรวิชาชีพชั้นสูง(ปวส.)	สถาบันเทคโนโลยีราชมงคล วิทยาเขต ภาคตะวันออกเฉียงเหนือ นครราชสีมา
ปริญญาตรี	สาขาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	-
คติพจน์	ฝันให้ไกลแล้วไปให้ถึง

ประวัติผู้แต่ง



ชื่อผู้ทำปริญญานิพนธ์	นายธงชัย ขำเทศเจริญ
วันเดือนปีเกิด	17 ธันวาคม 2519
สถานที่เกิด	จังหวัดนครราชสีมา
ภูมิลำเนาเดิม	67 ม.6 ต.บ้านโพธิ์ อ.เมือง จ.นครราชสีมา 30310
ที่อยู่ปัจจุบัน	หอพักชายยุคลรัตน์ 333 ม.1 ถ.ลาดกระบัง
	เขตลาดกระบัง กรุงเทพมหานคร 10520
โทรศัพท์	(02) 7390255-8 ห้อง 2102
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนตำบลบ้านโพธิ์
มัธยมศึกษาตอนต้น	โรงเรียนมหิศราธิบดี
มัธยมศึกษาตอนปลาย (ม.4)	โรงเรียนมหิศราธิบดี
ประกาศนียบัตรวิชาชีพชั้นสูง(ปวส.4 ปี)	สถาบันเทคโนโลยีราชมงคล วิทยาเขต
	ภาคตะวันออกเฉียงเหนือ นครราชสีมา
ปริญญาตรี	สาขาอิเล็กทรอนิกส์และคอมพิวเตอร์
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	ทุนยกเว้นหน่วยกิต
คติพจน์	ตั้งใจ ขยัน อดทน

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายสุระชัย พิมพ์สาลี
วันเดือนปีเกิด	20 ธันวาคม 2518
สถานที่เกิด	จังหวัดอุดรธานี
ภูมิลำเนาเดิม	462 หมู่ 1 ต.เชียงพิณ อ.เมือง จ.อุดรธานี 41000
ที่อยู่ปัจจุบัน	หอพักชายยุคลรัตน์ 333 ม.1 ถ.ลาดกระบัง เขตลาดกระบัง กรุงเทพมหานคร 10520
โทรศัพท์	(02) 7390255-8 ห้อง 1116
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนบ้านเชียงพิณ
มัธยมศึกษาตอนต้น	โรงเรียนอุดรพิทยานุกูล
ประกาศนียบัตรวิชาชีพ(ปวช.)	วิทยาลัยเทคนิคอุดรธานี
ประกาศนียบัตรวิชาชีพชั้นสูง(ปวส.)	วิทยาลัยเทคนิคอุดรธานี
ปริญญาตรี	สาขาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	ทุนยกเว้นหน่วยกิต
คติพจน์	ทำวันนี้ให้ดีที่สุด

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นางสาวเสาวลักษณ์ แสงแก
วันเดือนปีเกิด	22 มีนาคม 2518
สถานที่เกิด	จังหวัดอุบลราชธานี
ภูมิลำเนาเดิม	382 ม.19 ต.เมืองเดช อ.เดชอุดม จ.อุบลราชธานี 34160
ที่อยู่ปัจจุบัน	81 ถ.เจริญกรุง85 แขวงวัดพระยาไกร
	เขตบางคอแหลม กรุงเทพมหานคร 10120
โทรศัพท์	(02) 2916090

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนนาคสมุทรสงเคราะห์
มัธยมศึกษาตอนต้น	โรงเรียนเดชอุดม
ประกาศนียบัตรวิชาชีพ(ปวช.)	วิทยาลัยเทคนิคอุบลราชธานี
ประกาศนียบัตรวิชาชีพชั้นสูง(ปวส.)	สถาบันเทคโนโลยีราชมงคล วิทยาเขตเทคนิคกรุงเทพ
ปริญญาตรี	สาขาอิเล็กทรอนิกส์และคอมพิวเตอร์
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	ทุนผูกพันธ์สถาบันเทคโนโลยีราชมงคล
คติพจน์	การมีชีวิตอยู่เพื่อหน้าที่