ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ ตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

Codebook Search Processor

ชื่อนักศึกษา

1. นางสาวปราณี	ทิพย์สุวรรณ์	รหัสประจำตัว	41031413
2. นางสาวศันสนีย์	กิมเกถนอม	รหัสประจำตัว	41031421
3. นายสุทธิพงศ์	ปราชญ์นคร	รหัสประจำตัว	41031426

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** อิเล็กทรอนิกส์และคอมพิวเตอร์

อาจารย์ที่ปรึกษา อาจารย์กิติพงค์ มะโน

อาจารย์ที่ปรึกษาร่วม อาจารย์วรวิทย์ สมหา

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

วัน/เดือน/ปีที่สอบ วันอังคารที่ 2 พฤษภาคม พ.ศ. 2543 เวลา 16.00 น.

สถานที่สอบ ห้อง ค.310 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.วิสุทธิ์ อธิพรธรรม)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 11 เดือน ก.ค. พ.ศ. 2543

ปริญญานิพนธ์

ตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

CODEBOOK SEARCH PROCESSOR

นางสาวปราณี ทิพย์สุวรรณ์

นางสาวศันสนีย์ กิมเกถนอม

นายสุทธิพงศ์ ปราชญ์นคร

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์

เรื่อง ตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

Codebook Search Processor

วัตถุประสงค์

1. เพื่อศึกษาหลักการของตัวประมวลผลสำหรับค้นหาข้อมูลใน Codebook, ภาษา VHDL และอุปกรณ์ FPGAs
2. เพื่อศึกษาวิธีการออกแบบสถาปัตยกรรมและสร้างฮาร์ดแวร์ของตัวประมวลผลสำหรับค้นหาข้อมูลใน Codebook
3. เพื่อออกแบบตัวประมวลผลสำหรับค้นหาข้อมูลใน Codebook
4. เพื่อสร้างตัวประมวลผลสำหรับค้นหาข้อมูลใน Codebook โดยใช้อุปกรณ์ FPGAs
5. เพื่อทดสอบผลการออกแบบและวิธีการนำไปใช้งาน

ประโยชน์ที่คาดว่าจะได้รับ

1. ได้เข้าใจถึงหลักการและสถาปัตยกรรมของตัวประมวลผลสำหรับค้นหาข้อมูลใน Codebook, ภาษา VHDL และอุปกรณ์ FPGAs
2. ได้เข้าใจถึงหลักการออกแบบสถาปัตยกรรมและสร้างฮาร์ดแวร์ของตัวประมวลผลสำหรับค้นหาข้อมูลใน Codebook
3. ได้ศึกษาและเข้าใจถึงการออกแบบตัวประมวลผลสำหรับค้นหาข้อมูลใน Codebook
4. ได้ตัวประมวลผลสำหรับค้นหาข้อมูลใน Codebook โดยใช้อุปกรณ์ FPGAs
5. ได้ผลการทดสอบจากการออกแบบวงจรและสามารถนำไปใช้งานได้

ชื่อหัวข้อ	ตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค	
ชื่อนักศึกษา	นางสาวปราณี	ทิพย์สุวรรณ์
	นางสาวศันสนีย์	กิมเกถนอม
	นายสุทธิพงศ์	ปราชญ์นคร
อาจารย์ที่ปรึกษา	อาจารย์กิติพงค์	มะโน
อาจารย์ที่ปรึกษาร่วม	อาจารย์วรวิทย์	สมหา
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต	
สาขาวิชา	อิเล็กทรอนิกส์และคอมพิวเตอร์	
ปีการศึกษา	2542	

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ออกแบบตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊คโดยใช้อุปกรณ์ FPGAs เบอร์ 4010E ตัวประมวลผลดังกล่าวออกแบบโดยใช้ภาษา VHDL และใช้วิธีการค้นหาข้อมูลแบบเต็ม (Full Search) ซึ่งโค้ดบุ๊คมีขนาด 64 ×5 จากผลการทดลองตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊คสามารถทำงานได้ที่เวลาจริง นอกจากนี้ยังสามารถทำงานได้ที่อัตราบิต 1.2 บิตต่ออีลีเมนต์

Thesis Title	Codebook Search Processor	
Students	Miss.Pranee	Thipsuwan
	Miss.Sansanee	Kimkathanom
	Mr.Soottipong	Prachnakorn
Advisor	Mr.Kitipong	Mano
Co - Advisor	Mr.Worawit	Somha
Education Level	Bachelor of Science Industrial Education	
Program in	Electronics and Computer	
Acadamic Year	1999	

ABSTRACT

This thesis presents the design of a codebook search processor using the FPGA # 4010E device. The codebook search processor was designed by implementing the VHDL language and using the standard method. The size of the codebook that used in this design was 64x5. As shown by the result, the codebook search processor works properly in real time. In addition, this system, which can work at the bit rate of 1.2 bit per element, can properly code and decode.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากความกรุณาของอาจารย์ที่ปรึกษา และอาจารย์ประจำภาควิชาครุศาสตร์วิศวกรรมทุกท่าน ในการให้คำปรึกษา แนะนำ และความช่วยเหลือต่างๆ ตลอดจนให้โอกาสในการทำโครงงานอย่างเต็มที่ ทั้งด้านเวลา สถานที่ เครื่องมืออุปกรณ์ต่างๆ และขอขอบพระคุณคุณพ่อ คุณแม่ ผู้ให้กำเนิดและให้โอกาสในการศึกษา ตลอดจนเพื่อนๆ และผู้ที่เกี่ยวข้องทุกท่านที่ให้คำแนะนำและคอยเป็นกำลังใจในการทำงาน จนทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญต่อ

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

ในปัจจุบัน ความเจริญก้าวหน้าทางด้านอิเล็กทรอนิกส์ได้มีการพัฒนาไปมาก ซึ่งต่างจากสมัยก่อนที่อุปกรณ์อิเล็กทรอนิกส์มีการพัฒนาช้ามากและมีโครงสร้างขนาดใหญ่ ในปัจจุบันได้มีการสร้างอุปกรณ์อิเล็กทรอนิกส์ที่มีขนาดเล็กๆ บนชิ้นสารกึ่งตัวนำ ซึ่งเป็นผลให้ประสิทธิภาพในการทำงานสูงขึ้น ตลอดจนความน่าเชื่อถือ และความคงทนต่อสภาพแวดล้อมด้วย ในขณะเดียวกันได้มีการออกแบบวงจรเพื่อสนับสนุนระบบการประมวลผลเชิงเลขซึ่งเป็นระบบดิจิตอล ซึ่งจะให้ประสิทธิภาพดีกว่าระบบแอนะลอก ทั้งขนาดของข้อมูล ความผิดเพี้ยนของสัญญาณ ความละเอียดในการส่ง โดยเฉพาะอย่างยิ่งในด้านการสื่อสารข้อมูล

ดังนั้นโครงงานนี้จึงได้ออกแบบตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค ซึ่งมีความสำคัญมากในด้านการใช้งานเกี่ยวกับการเข้ารหัสข้อมูล ซึ่งใช้หลักการควอนไตซ์เวกเตอร์แบบมาตรฐานช่วยในการสุ่มสัญญาณ ทำให้มีการผิดเพี้ยนของสัญญาณน้อยลง และสามารถทำงานได้ที่เวลาจริงมากขึ้น โครงงานชิ้นนี้สามารถใช้ได้กับข้อมูลดิจิตอลขนาด 8 บิต และสามารถค้นหาข้อมูลภายในโค้ดบุ๊คที่มีขนาด 64×5 การประมวลผลจะเป็นแบบไปป์ไลน์ ซึ่งจะช่วยให้การทำงานเร็วขึ้น ในการออกแบบได้ใช้ภาษา VHDL บรรยายการทำงานของฮาร์ดแวร์ที่แสดงถึงความสัมพันธ์ระหว่างสัญญาณอินพุตกับสัญญาณเอาต์พุต และมีการสร้างจริงด้วยอุปกรณ์ FPGAs

1.2 ขีดความสามารถของโครงงาน

1. สามารถใช้กับข้อมูลดิจิตอลขนาด 8 บิต
2. สามารถค้นหาข้อมูลใน Codebook ขนาด 64 ×5 ได้
3. สามารถเข้ารหัสและถอดรหัสสัญญาณเสียงได้
4. สามารถรับส่งข้อมูลในอัตรา 9,600 บิตต่อวินาที
5. ความถี่ที่ใช้ในการสุ่มสัญญาณ 8 กิโลเฮิร์ต

1.3 เนื้อหาโดยสังเขป

เนื้อหาของปริญญานิพนธ์ฉบับนี้ ได้กล่าวถึงการออกแบบตัวประมวลผลสำหรับค้นหาข้อมูลใน Codebook ซึ่งสามารถนำไปประยุกต์ใช้งานในด้านต่างๆ ได้มากมาย ซึ่งจะใช้อุปกรณ์ FPGAs ในการทำงานเพื่อสะดวกต่อการศึกษาและทำความเข้าใจจึงได้เสนอเนื้อหาที่สำคัญในแต่ละบทดังนี้

บทที่ 2 ทฤษฎีและหลักการ การควอนไตซ์เวกเตอร์แบบมาตรฐาน การออกแบบโค้ดบุ๊ค การออกแบบ Top – Down Design โครงสร้างของภาษา VHDL ลักษณะการบรรยายด้วยภาษา VHDL การบรรยายเชิงพฤติกรรม การบรรยายเชิงข้อมูล การบรรยายเชิงโครงสร้าง ทฤษฎีและหลักการของ FPGAs โครงสร้างภายในของอุปกรณ์ FPGAs รายละเอียดการใช้งานอุปกรณ์ FPGAs

บทที่ 3 การออกแบบการสร้างและการทำงาน ลักษณะการออกแบบ วิธีการออกแบบวงจร ขั้นตอนการออกแบบวงจร การออกแบบและการสร้างวงจรทดสอบ

บทที่ 4 การทดลองและผลการทดลอง นำโปรแกรมมา Simulate Block ส่วนย่อยแต่ละส่วน จากนั้นก็ทำการ Simulate Block รวมทั้งหมด แล้วก็นำไป Simulate กับตัวประมวลผลสำหรับค้นหา ข้อมูลในโค้ดบุ๊ค และก็นำไปทดลองกับวงจรเครื่องรับ

บทที่ 5 บทสรุป ปัญหา แนวทางแก้ไขและพัฒนา การสรุปผลและอภิปรายเกี่ยวกับความสามารถ ประสิทธิภาพของตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค ปัญหาที่เกิดขึ้นระหว่างการทำโครงงาน แนวทางการแก้ไขปัญหาที่เกิดขึ้น และข้อเสนอแนะ แนวทางในการพัฒนาปรับปรุงโครงงานให้มีประสิทธิภาพมากยิ่งขึ้น

ในภาคผนวกแสดงรายละเอียดของโปรแกรมและรายการอุปกรณ์ต่าง ๆ ที่ใช้ในการจัดทำโครงงานมีดังนี้

ภาคผนวก ก รายละเอียดของวงจรและผลการทดสอบวงจร

ภาคผนวก ข โปรแกรม VHDL

ภาคผนวก ค รายละเอียดของอุปกรณ์

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

สำหรับทฤษฎีที่ใช้ในการออกแบบโครงงานนี้ประกอบด้วย ทฤษฎีของเวกเตอร์ควอนไตซ์ การออกแบบโค้ดบุ๊ค การออกแบบ Top-Down Design ทฤษฎีและหลักการของภาษา VHDL ทฤษฎีและหลักการของ FPGAs ซึ่งจะนำเสนอเป็นส่วนๆ ดังต่อไปนี้

2.2 การควอนไตซ์เวกเตอร์แบบมาตรฐาน

2.2.1. หลักการควอนไตซ์เวกเตอร์แบบมาตรฐาน

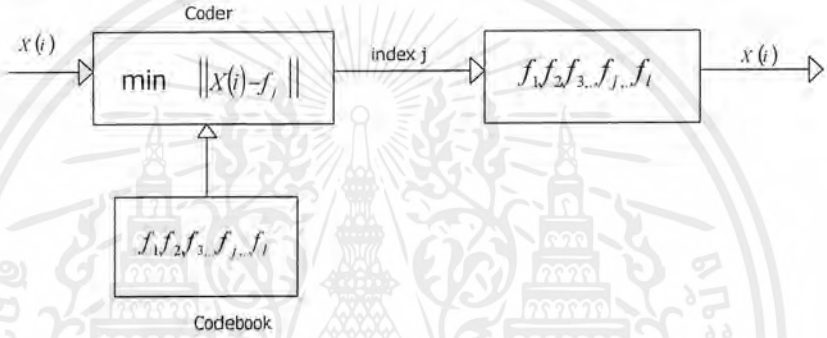
การควอนไซต์เวกเตอร์แบบมาตรฐาน เป็นการควอนไตซ์เวกเตอร์แบบแรกสุดที่ได้มีการนำไปประยุกต์ใช้งานในการเข้ารหัสสัญญาณเสียงและเป็นพื้นฐานในการพัฒนาการควอนไตซ์เวกเตอร์ในเวลาต่อมา ซึ่งวิธีการควอนไตซ์เวกเตอร์แบบมาตรฐานนี้ได้นำเสนอวิธีการออกแบบโค้ดบุ๊คซึ่งเป็นที่รู้จักกันดี คือวิธีการแบบ LBG (LBG Algorithms) หรือในบางครั้งเราเรียกว่าวิธีการแบบ Spliting Algorithms ซึ่งได้พัฒนามาจากวิธีการของ Lloyd (Lloyd Algorithms) หลักการควอนไตซ์เวกเตอร์แบบมาตรฐาน

รูปที่ 2.1 เป็นหลักการพื้นฐานของการควอนไตซ์เวกเตอร์แบบมาตรฐาน วงจรเครื่องส่ง (Transmitter) ประกอบไปด้วยส่วนที่เป็นโค้ดบุ๊คเวกเตอร์ที่มี N มิติ ซึ่ง N คือ จำนวนของตัวอย่าง (Samples) หรืออีลีเมนต์ (Element) ในหนึ่งเวกเตอร์และมีจำนวน L เวกเตอร์คือ $f_j \ \{ j = 1,2,3,\ldots,L\}$ โดยเป็นส่วนที่มีความสำคัญเนื่องจากโค้ดบุ๊คจะต้องเป็นที่เก็บคุณลักษณะของเสียงไว้ทั้งหมดและการจัดโค้ดบุ๊คเวกเตอร์ในหน่วยความจำได้จัดในลักษณะของอะเรย์ 2 มิติ ซึ่งแสดงได้ดังรูปที่ 2.2 และจากรูปที่ 2.1 ในขบวนการเริ่มต้นสัญญาณอินพุตเวกเตอร์ $X(i)$ จะถูกสุ่มเข้ามาจำนวน N แซมเปิ้ล (เท่ากับมิติของโค้ดบุ๊คหรือหนึ่งเวกเตอร์) โดยวงจรเข้ารหัสจะแทนสัญญาณเวกเตอร์ $X(i)$ ด้วยค่าเวกเตอร์ f_j จากโค้ดบุ๊ค ซึ่งจะมีเพียงหนึ่งเวกเตอร์ที่มีคุณลักษณะใกล้เคียงกับสัญญาณอินพุตเวกเตอร์ $X(i)$ มากที่สุด โดยการหาค่าต่ำสุด (Minimize the Euclidean Distance) ซึ่งมีเงื่อนไขดังนี้

$$\|X(i)-f_j\| < \|X(i)-f_k\| \qquad (2.1)$$

โดยที่ $k = 1,2,3,\ldots,L \qquad ;k \neq j$

และจะได้ค่าดัชนี (Index) ของเวกเตอร์จากโค้ดบุ๊คคือ j มาทำการถอดรหัสจากโค้ดบุ๊คที่เครื่องรับ โดยที่โค้ดบุ๊คที่เครื่องส่งและเครื่องรับจะต้องมีข้อมูลที่เหมือนกันหลังจากนั้นจะได้สัญญาณเอาต์พุตเวกเตอร์ที่ถูกสร้างขึ้นใหม่คือ เวกเตอร์ $X(i)$ ซึ่งมีค่าเท่ากับเวกเตอร์ f_j ที่เก็บอยู่ในโค้ดบุ๊คนั้นเอง โดยที่เวกเตอร์ f_j จะต้องมีค่าใกล้เคียงกับอินพุตเวกเตอร์ $X(i)$ มากที่สุด



รูปที่ 2.1 การควอนไตซ์เวกเตอร์แบบมาตรฐาน

ค่า Parameter ที่สำคัญอีกประการหนึ่งของควอนไตซ์เวกเตอร์ คือ ความละเอียด (Resolution) ของจำนวนบิตที่ใช้ในหนึ่งอีลีเมนต์ของเวกเตอร์สามารถคำนวณได้ดังนี้

$$R = \frac{Log_2 L}{N} \quad (bit/element) \qquad (2.2)$$

สมมติถ้ากำหนดให้โค้ดบุ๊คมีมิติ N = 5 และจำนวนเวกเตอร์ในโค้ดบุ๊ค L = 64 จะได้

$$R = \frac{Log_2 64}{5} = \frac{6}{5} = 1.2 \quad bit/element \qquad (2.3)$$

$f_{1(1)}$	$f_{1(2)}$	...	$f_{1(N)}$
$f_{2(1)}$	$f_{2(2)}$	...	$f_{2(N)}$
$f_{3(1)}$	$f_{3(2)}$	...	$f_{3(N)}$
.	.	.	.
.	.	.	.
.	.	.	.
$f_{L(1)}$	$f_{L(2)}$	...	$f_{L(N)}$

รูปที่ 2.2 การจัดเก็บโค้ดบุ๊คในหน่วยความจำ

นั่นหมายความว่า ในหนึ่งแซมเปิ้ลของการสุ่มสัญญาณที่ส่งไปยังเครื่องรับต้องการข้อมูลเพียง 1.2 บิต จะสังเกตเห็นว่า ความละเอียดของตัวแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอลจะไม่มีผลต่อบิตเรตในการส่งข้อมูล ถ้าหากเรานำหลักการควอนไตซ์เวกเตอร์ไปประยุกต์ใช้ในการเข้ารหัสสัญญาณเสียงจากสมการที่ 2.2 สามารถคำนวณหาค่าบิตเรตในการส่งข้อมูลได้ดังนี้

$$\text{Bitrate} = R \times Fs \quad \text{(bit/sec)} \tag{2.4}$$

$$Fs = \text{ความถี่ในการสุ่มสัญญาณ (Hz)}$$

ถ้าอัตราการสุ่มสัญญาณ Fs = 8 kHz มิติของโค้ดบุ๊ค N = 5 (หมายถึงการเข้ารหัสในแต่ละครั้งใช้สัญญาณจำนวน 5 แซมเปิ้ล) และจำนวนของเวกเตอร์ L = 64 นั่นหมายความว่าค่าดัชนีเวกเตอร์ j ที่ส่งไปยังเครื่องรับจำนวน 6 บิต สามารถสร้างสัญญาณเอาต์พุตเวกเตอร์ $X(i)$ ได้จำนวน 5 แซมเปิ้ล ดังนั้นจำนวนบิตเรตในการส่งข้อมูลจะได้เป็น

$$\text{Bitrate} = 1.2 \times 8000 = 9600 \text{ bps} \tag{2.5}$$

2.2.2 การคำนวณค่าอัตราส่วนสัญญาณต่อสัญญาณรบกวน

การวัดคุณภาพของเสียงที่ได้จากการควอนไตซ์เวกเตอร์กระทำได้โดยการทดสอบหาค่าอัตราส่วนสัญญาณต่อสัญญาณรบกวน (SNRseg หรือ Segment Signal to Noise Ratio) ระหว่างสัญญาณอินพุต $X(i)$ และสัญญาณเอาต์พุต $X(n)$ โดยทำการแบ่งสัญญาณ $X(n)$ เป็นเฟรม พร้อมทั้งทำการ

คำนวณครั้งละหนึ่งเฟรม ซึ่งเขียนแทนด้วย SNRm และในแต่ละเฟรมจะใช้จำนวนตัวอย่าง 128 ตัวอย่างดังนี้

$$SNRm = 10Log_{10} \frac{\sum_{n=1}^{128} \{X(n)\}^2}{\sum_{n=1}^{128} \{X(n) - X(n)\}^2} \quad (2.6)$$

$X(i)$ = สัญญาณอินพุต

$X(n)$ = สัญญาณที่ผ่านการควอนไตซ์เวกเตอร์

หลังจากนั้นจะนำเอาค่า SNRm ทั้งหมดที่ได้จากสมการที่ (2.11) มาทำการหาค่าเฉลี่ย ซึ่งเขียนแทนด้วย SNRseg ดังนี้

$$\text{SNRseg} = \frac{1}{S} \sum_{m=1}^{S} SNRm(dB) \quad (2.7)$$

S คือ จำนวนของเฟรมทั้งหมด

สำหรับการคำนวณในแต่ละเฟรม ถ้าหาก $\sum_{n=1} (X(i))^2 < 32 \times 10^4$ (หรือต่ำกว่า 43 dB ของสัญญาณ Sinusoidal ที่แปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอลแปลงได้เต็มสเกล) เราจะทำการตัดการคำนวณในเฟรมนี้ออกไป ทั้งนี้เนื่องจากสัญญาณที่เกิดขึ้นในระดับนี้ส่วนใหญ่จะเป็นสัญญาณรบกวน (Noise) ที่เกิดจากขบวนการควอนไตซ์ของตัวแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล เพราะสัญญาณเสียงที่พูดได้จะต้องมีพลังงาน (Energy) ที่มากกว่านี้

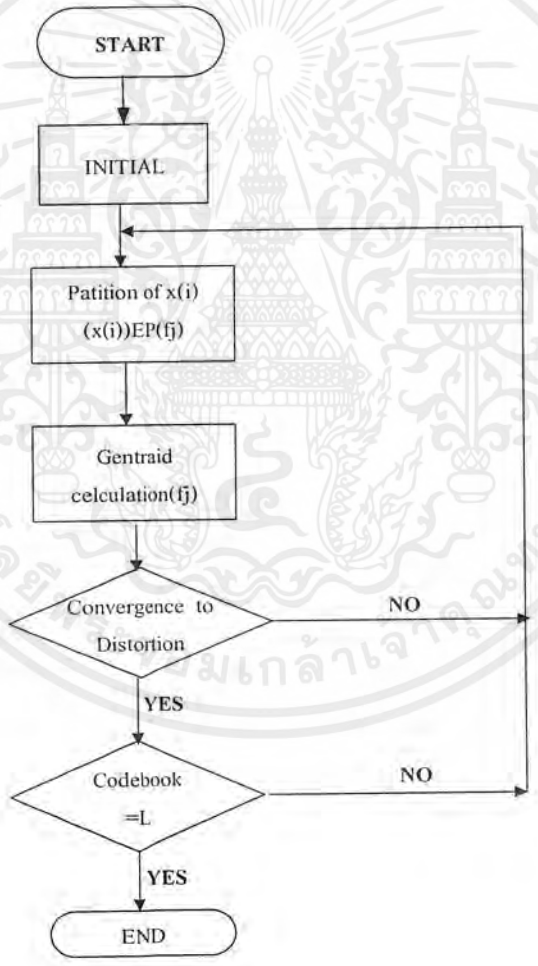
ในอดีตรูปแบบของสัญญาณไฟฟ้าโดยมากมักอยู่ในรูปสัญญาณแอนะลอก การนำเอาสัญญาณไฟฟ้ามาประมวลผลเพื่อให้เกิดรูปแบบที่ต้องการนั้น ต้องใช้อุปกรณ์ทางแอนะลอก แต่ปัจจุบันนี้เทคโนโลยีทางด้านดิจิตอลก้าวหน้าไปมาก ทำให้การประมวลผลสัญญาณทางดิจิตอลสามารถทำได้อย่างรวดเร็ว และมีประสิทธิภาพ

ดังนั้นการแปลงรูปแบบสัญญาณ (Conversion) จึงมีความจำเป็นในการแปลงสัญญาณ แอนะลอกที่มีอยู่แล้วให้เป็นสัญญาณดิจิตอล โดยอุปกรณ์การแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และจะถูกประมวลผลโดยตัวประมวลผลสัญญาณดิจิตอล เช่น คอมพิวเตอร์ เป็นต้น จากผลลัพธ์ที่ได้อาจถูกนำมาแสดงผลโดยตรงเลย หรืออาจถูกแปลงให้อยู่ในรูปของสัญญาณแอนะลอก

ที่ใช้งานได้ การที่จะแปลงสัญญาณแอนะลอกเป็นดิจิตอลได้นั้น สามารถทำได้โดยใช้อุปกรณ์แปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก สำหรับระบบที่มีการประมวลผลข้อมูลทางดิจิตอล

2.2.3 การออกแบบโค้ดบุ๊ค

ขั้นตอนการออกแบบจะนำเอาสัญญาณเสียงต้นฉบับที่ใช้ในการออกแบบโค้ดบุ๊ค $\{X(i)\}$ (Training Vector) มาทำการแยกกลุ่ม (Clustering) เพื่อหาคุณลักษณะของสัญญาณเสียงที่เหมาะสมที่สุดและให้มีความผิดพลาด (Distortion) น้อยที่สุด โดยกำหนดให้โค้ดบุ๊คเวกเตอร์ที่ต้องการออกแบบมีจำนวน L เวกเตอร์ ในแต่ละเวกเตอร์มีจำนวนอีลีเมนต์เท่ากับ N ซึ่งจากวิธีการของ LBG สามารถเขียนแผนผังงานการออกแบบโค้ดบุ๊คได้ดังรูปที่ 2.3



รูปที่ 2.3 วิธีการของ LBG โดยการประยุกต์จากวิธีการของ Lioyd

ซึ่งขั้นตอนการออกแบบโค้ดบุ๊คมีขั้นตอนการออกแบบดังนี้

กำหนดให้โค้ดบุ๊คเวกเตอร์ที่ต้องการมีจำนวน L เวกเตอร์ในแต่ละเวกเตอร์ มีจำนวนอีลีเมนต์เท่ากับ N

$P(f_j)$ = Voronoi Cell โดยมีโค้ดบุ๊คเวกเตอร์ f_j เป็นค่า Cenyriod ของ $P(f_j)$, $\{j = 1,2,3,\ldots L\}$

$\{X(i)\}$ = สัญญาณเสียงต้นฉบับสำหรับการออกแบบโค้ดบุ๊คโดย $\{i = 1,2,3,\ldots M\}$ และ M คือ จำนวนของเวกเตอร์ใน $\{X(i)\}$

α = เวกเตอร์สำหรับการ Spliting โดยจำนวนอีลีเมนต์เท่ากับ N ในที่นี้ใช้ค่า $\alpha = [0.01,0.01,\ldots 0.01]^t$

ε = Theshold เลือกค่า $\varepsilon = 0.005$

ขั้นตอนการออกแบบ

1. กำหนดให้ Iteration m = 0

2. ติดตั้งโค้ดบุ๊คเวกเตอร์ $f_{j(m)}$ ที่ $j = 1$ โดยหาค่า Centroid ของสัญญาณเสียงต้นฉบับ $\{X(i)\}$ ได้จาก

$$f_{j(m)} = \frac{1}{M}\sum_{i=1}^{M} x(i) \tag{2.8}$$

3. ทำการแยกโค้ดบุ๊คเวกเตอร์ $f_{j(m)}$ ออกเป็น 2 ส่วน โดยการบวกและลบด้วยเวกเตอร์ซึ่งมีจำนวนอีลีเมนต์เท่ากับโค้ดบุ๊คเวกเตอร์ และจะได้โค้ดบุ๊คเวกเตอร์ใหม่เป็น 2 เท่าของจำนวน เวกเตอร์เดิมดังนี้

$$\begin{aligned} f_{j(m)}^{new} &= f_{j_{(m)}}^{old} - \alpha \\ f_{j+1(m)}^{new} &= f_{j_{(m)}}^{old} - \alpha \end{aligned} \tag{2.9}$$

4. กำหนดให้ m = m+1

5. แยกสัญญาณเสียงต้นฉบับ $\{X(i)\}$ ให้เข้ากลุ่มของ $P(f_{j(m)})$ ดังนี้

$$x(i) \in p(f_{j(m)}) \; if \; X(i) = f_j$$

โดยที่ $\{i = 1,2,3,\ldots,M\}$

$\{j = 1,2,3,\ldots,L\}$ $;k \neq j$ (2.10)

6. คำนวณหาค่า Centroid ในแต่ละ $P(f_{j(m)})$ โดยการนำเอา *{X(i)}* ที่ถูกจัดกลุ่มในขั้นที่ 5 มาทำการคำนวณดังนี้

$$f_{j(m)} = \frac{1}{M_{jx(i)\in P(f_{j(m)})}}$$

$$f_{j(m)} = \frac{1}{M}\sum_{i} x(i) \qquad (2.11)$$

โดยที่ $\{j = 1,2,3,...L\}$

M_j คือ จำนวนของ training vector ในแต่ละ $P(f_{j(m)})$

7. คำนวณค่าความเพี้ยนเฉลี่ย (Average Distrotion) *D(m)* รวมทั้งหมดของโค้ดบุ๊คเวกเตอร์ $f_{j(m)}$

$$D(m) = \sum_{j=1}^{L} \frac{M_j}{M} D_{j(m)} \qquad (2.12)$$

โดยที่ $\{j = 1,2,3,...L\}$

โดยที่ $D_{j(m)}$ คือความเพี้ยนเฉลี่ยในแต่ละ $P(f_{j(m)})$ ซึ่งเกิดจากความเปรียบเทียบกันระหว่างสัญญาณเสียงต้นฉบับ *{X(i)}* และโค้ดบุ๊คเวกเตอร์ $f_{j(m)}$ ในแต่ละ $P(f_{j(m)})$

$$D_{j(m)} = \frac{1}{M_{jx(i)\in P(f_{j(m)})}} \left\| x(i) - f_{j(m)} \right\|^2 \qquad (2.13)$$

โดยที่ $\{j = 1,2,3,...L\}$

8. ตรวจสอบค่าความผิดเพี้ยนรวมทั้งหมดเทียบกับค่า Theshold

$$\frac{D(m-1) - D(m)}{D(m-1)} < \varepsilon \qquad (2.14)$$

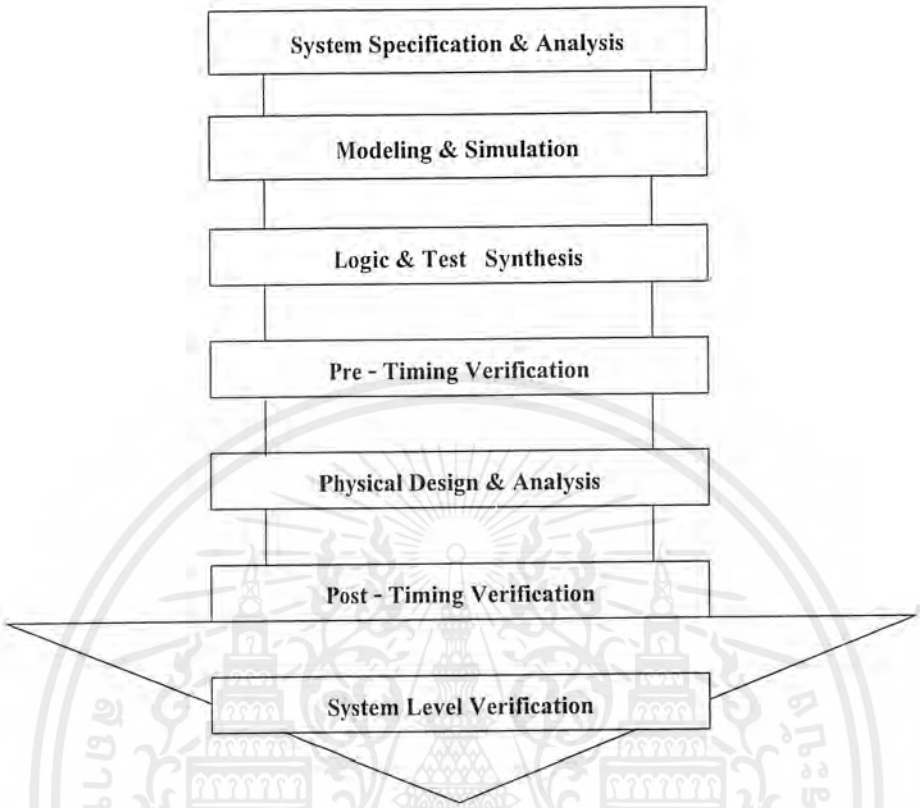
เมื่อ $D(m-1)$ คือค่าความผิดเพี้ยนที่เกิดขึ้นในครั้งก่อน ถ้าข้อกำหนดจากสมการในข้อที่ 8 ไม่เป็นจริง ให้กลับไปทำในขั้นตอนที่ 4 และข้อกำหนดจากสมการในข้อที่ 8 เป็นจริงจะกำหนดให้ $m = 0$ และกระจายโค้ดบุ๊คเวกเตอร์ f_i ในแต่ละ *P(f)* ออกเป็น 2 เวกเตอร์โดยการบวกและลบด้วยเวกเตอร์ เช่นเดียวกับขั้นตอนที่ 3 และให้กลับไปกระทำในขั้นตอนที่ 4 โดยการกระทำในแต่ละ

รอบจำนวนของโค้ดบุ๊คเวกเตอร์ f_j จะมีค่าเพิ่มขึ้นเป็น 2 เท่า ซึ่งจะได้โค้ดบุ๊คเวกเตอร์ f_j ในแต่ละ $P(f_j)$ ในแต่ละ $P(f_j)$ จำนวน L เวกเตอร์ อันเป็นการสิ้นสุดการออกแบบโค้ดบุ๊ค

2.3. Top – Down Design

การนำภาษา VHDL มาใช้กำหนดและบรรยายพฤติกรรมฟังก์ชั่นการทำงานของฮาร์ดแวร์ในระบบดิจิตอลนั้น คือความอ่อนตัวของภาษาที่สามารถจำลองการทำงานจากหลักการของรูปแบบ (Simulate Conceptual Designs) แต่ในขณะเดียวกันก็สามารถจำลองการทำงานของฮาร์ดแวร์ ที่ให้รายละเอียดเกี่ยวกับเวลาอย่างถูกต้อง (Timing Based Simulation) และจากโครงสร้างของภาษายังสามารถจำลองการทำงานในรูปของลำดับชั้น (Hierarchy of Simulation Levels) ความสามารถดังกล่าวนี้ จึงช่วยให้การออกแบบสามารถที่จะเขียนรูปแบบบรรยายจากระดับบนสุดของวงจรที่อยู่ในรูปสังเขป (High Level of Abstraction) ลงสู่รายละเอียดในระดับล่างของวงจรได้ เช่น Gate Level เป็นต้น ในเวลานั้นวงการอุตสาหกรรมไมโครอิเล็กทรอนิกส์ ตลอดจนสถาบันวิจัยและศึกษา กำลังพัฒนาภาษาที่ใช้สำหรับการสังเคราะห์วงจรแบบอัตโนมัติ เพื่อลดเวลาในการพัฒนาวงจรลง ภาษา VHDL จึงถูกนำเข้าพิจารณาในโครงการนี้ด้วย โดยเพิ่มขีดความสามารถของภาษาขึ้น นอกเหนือจากเป็นภาษาที่ใช้สำหรับสังเคราะห์วงจร (Synthesis Language)

การเริ่มต้นด้วยวิธีการเขียนรูปแบบจากแนวความคิดอย่างสังเขป พร้อมทั้งการจำลองการทำงานของรูปแบบที่เขียนขึ้นเพื่อตรวจสอบความถูกต้อง ประกอบกับการกลั่นกรองเพื่อเพิ่มเติมรายละเอียดลงสู่ระบบดิจิตอลที่สมบูรณ์ในรูปของวงจรไฟฟ้าที่แต่ละขั้นนั้นเป็นขบวนการของ Top-Down Design การที่เริ่มต้นด้วยการเขียนรูปแบบในระดับบน (Top-Level) ของแนวความคิดอย่างสังเขปนั้น วิศวกรออกแบบสามารถที่จะพัฒนาสภาพแวดล้อมต่าง ๆ เพื่อการตรวจสอบการทำงานของวงจร (Test Environment) ได้ตั้งแต่ในระยะแรก ๆ ของการออกแบบ เพื่อใช้สำหรับตรวจสอบความถูกต้องของแนวความคิดกับสิ่งที่ต้องการจริงหรือ Specification ของงาน ดังนั้นจึงเป็นไปได้ยากที่ในระดับล่างลงมาโดยที่ได้เพิ่มรายละเอียดของรูปแบบให้มากขึ้นตามลำดับจะเกิดข้อผิดพลาด หรือเบี่ยงเบนไปจากจุดประสงค์เดิม เพราะในแต่ละขั้นตอนย่อยจะมีการสร้างสภาพแวดล้อมเพื่อการตรวจสอบขึ้นใหม่ โดยอ้างอิงจากระดับที่อยู่สูงกว่าขึ้นไปเสมอ จากรูปที่ 2.4 แสดงให้เห็นขั้นตอนของการออกแบบในลักษณะของ Top-Down Design ทั้งนี้ในทางปฏิบัติอาจจะมีข้อแตกต่างไปจากนี้บ้างเล็กน้อย ก็เนื่องมาจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้ในหลาย ๆ เทคโนโลยี เช่น Programmable Logic Devices อันได้แก่ PLA, FPGAs หรือ CPLD เป็นต้น นอกจากนั้นยังมี Semi-Custom IC (Gate Array, Standard Cell) และ Full Custom IC



รูปที่ 2.4 ขั้นตอนของ Top-Down Design

2.3.1 ขบวนการออกแบบโดยใช้วิธี Top-Down Design

ขั้นตอนการออกแบบ Top-Down Design มีรายละเอียดดังนี้

1. System Specification and Analysis ขั้นตอนของการสร้างข้อกำหนดของความต้องการ (Specification) และวิเคราะห์ระบบ เพื่อหาแนวความคิดและหลักการ (Idea and Concept) ในการแก้ปัญหา

2. Modeling and Simulation การเขียนรูปแบบของระบบที่ต้องการจะออกแบบ โดยภาษา VHDL หรือ HDL อื่น ๆ จากแนวความคิดอย่างสังเขปที่ได้ สำหรับบรรยายพฤติกรรมการทำงาน พร้อมทั้งจำลองการทำงานเพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด

3. Logic and Test Synthesis หลังจากที่ได้หลักการขั้นต้นพร้อมกับแนวความคิดที่ผ่านการตรวจสอบแล้ว หลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมาเป็นลำดับขั้นตอนที่เหมือนกัน คือ Modeling and Simulation จนกระทั่งอยู่ในระดับที่จะนำไปผลิตวงจร หรือสังเคราะห์ (Synthesis) ในขั้นตอนนี้เอง เทคโนโลยีที่จะมารองรับวงจรออกแบบจะถูกกำหนดขึ้นและระบบช่วยการ

ออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้น ให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้น หรือไม่ก็อยู่ในรูปของ Netlist ที่สามารถนำไปผลิตลงบนอุปกรณ์อื่นได้ นอกจากนั้นการผลิตบางเทคโนโลยี อาทิเช่น Gate Array หรือ Standard Cell และ Full Custom IC อาจจะมีความจำเป็นที่ต้องสร้างโครงสร้างของวงจรใหม่หลังจากที่สังเคราะห์ครั้งแรกแล้ว เพื่อความสะดวกต่อการตรวจสอบการทำงานหลังจากที่ผลิตเป็นวงจรต้นแบบแล้ว หรือที่เรียกว่า "Design For Test" (DFT) พร้อมทั้งข้อมูลในการตรวจสอบจะถูกกำหนดในขั้นตอนนี้

4. Pre-Timing Verification หลังจากการสังเคราะห์วงจรให้อยู่ในรูป Gate-Level หรือ Netlist แล้วข้อมูลที่ได้จากผู้ผลิตอุปกรณ์วงจรนั้น นอกจากจะเป็นข้อมูลสำหรับจำลองการทำงานในเรื่องของความถูกต้องของฟังก์ชัน (Functional Simulation) แล้ว ยังมีข้อมูลที่เกี่ยวกับเวลาด้วย ซึ่งเป็นความจริงที่ว่า อุปกรณ์ทางอิเล็กทรอนิกส์ทุกชิ้นจะมี Propagation Delay เสมอ ถึงแม้ว่าจะเป็นเวลาที่น้อยมากในระดับ nanosecond (10^{-9} second) แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกตของฟังก์ชันต่าง ๆ จำนวน 10,000 เกทขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้น จนอาจจะทำให้การทำงานของวงจรรวมทั้งหมดผิดไป หรือไม่สามารถทำงานในย่านความถี่สัญญาณนาฬิกาที่สูง

5. Physical Design and Analysis คือขั้นตอนของการผลิตเป็นวงจรจริง (Technology and Device Mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจจะอยู่ในรูปของแผงวงจรไฟฟ้าที่ประกอบด้วยอุปกรณ์หลายๆ ชิ้น หรืออยู่ในรูปของวงจรรวมเฉพาะงาน (ASIC)

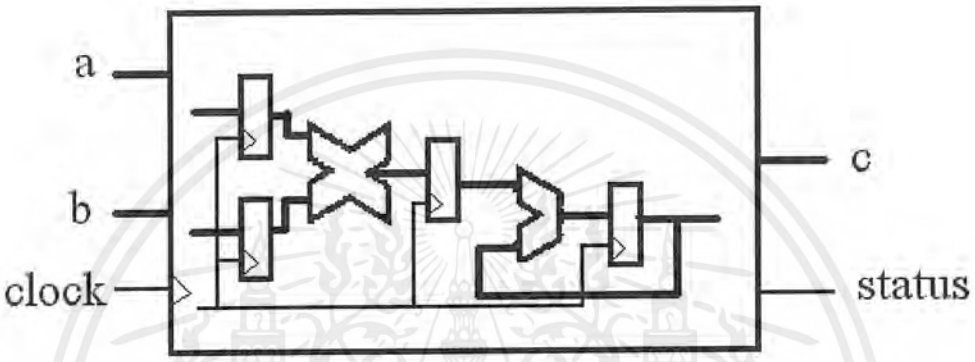
6. Post-Timing Verification หลังจากที่ได้วงจรจริงมาเรียบร้อยแล้ว ยังต้องมีความจำเป็นที่ต้องตรวจสอบการทำงานที่คำนึงถึงเวลาด้วย เพื่อความถูกต้องของวงจรครั้งสุดท้ายก่อนที่จะนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เพราะในขั้นตอนนี้วงจรที่ออกแบบ จะประกอบด้วย Input และ Output Pad ซึ่งเป็นจุดต่อสำหรับรับและส่งสัญญาณกับภายนอก

7. System Level Verification หลังจากที่นำวงจรที่ออกแบบรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอลแล้วนั้น จะต้องทดสอบการทำงานรวมทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้งเป็นการควบคุมคุณภาพของผลิตภัณฑ์

ในตัวอย่างของ multiply accumulate algorithm แสดงให้เห็นขบวนการออกแบบในลักษณะของ Top-Down Design ได้โดยการใช้ Multiply Accumulate Function จุดประสงค์แรกก็คือ การเขียนรูปแบบของฟังก์ชั่นในระดับบนสุดด้วยภาษา VHDL และจำลองการทำงานของรูปแบบที่ได้เพื่อตรวจสอบความถูกต้องซึ่งฟังก์ชันนี้อาจจะเป็น Discrete Fourier Transform (DFT) ก็ได้ หลังจากที่ได้ฟังก์ชั่นที่ต้องการแล้วและมีชื่อว่า "multiply _accum" สามารถนำไปใช้ได้ในรูปของ Function Call คือ

C <= multiply_accum(a,b); (2.15)

ถ้าฟังก์ชั่นเป็นที่พอใจแล้วในขั้นต่อไปจะเป็นการเริ่มของการแปลงแนวความคิดไปสู่การสร้างวงจรจริง ขบวนการดังกล่าวเกิดขึ้นโดยการเพิ่มรายละเอียดให้กับแนวความคิดเดิม รูปที่ 2.5 แสดงให้เห็นวิธีการบรรยายแบบ Dataflow (Dataflow Description) ของ Multiply Accumulate Function



รูปที่ 2.5 Dataflow Description ของ Mutiply-Accumulate Function

จาก Dataflow Format ที่ได้นี้ ขบวนการต่อไปคือการสร้างวงจรให้อยู่ในรูปของอุปกรณ์พื้นฐานหรือ Gate-Level Implementation ซึ่งวิธีนี้อาจใช้วิธีการสังเคราะห์อัตโนมัติ ผลลัพธ์ที่ได้จากการสังเคราะห์จะเป็นการแสดงภาพวงจรด้วยเกตของฟังก์ชันต่างๆ หรือในรูปของ Netlist และจะเป็นการกำหนดเทคโนโลยีจำเพาะที่จะนำไปผลิตในภายหลัง

คุณสมบัติหลักของภาษา VHDL คือสามารถที่จะบรรยายฮาร์ดแวร์ ได้ในทุกๆ ระดับของภาพรวมทั้งระบบ ฉะนั้น ผู้ออกแบบจึงสามารถใช้เครื่องมือ (ภาษา) เพียงอันเดียวในการบรรยายทั้งระบบ ซึ่งก็เช่นเดียวกันกับเครื่องมือจำลองการทำงาน

2.4 ภาษา VHDL

ภาษา VHDL เริ่มประมาณปี ค.ศ.1981 โดยที่กระทรวงกลาโหมสหรัฐอเมริกา หรือ Department Of Defense (DOD) เห็นว่าอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการทางทหาร เป็นอุปกรณ์ที่ได้รับการพัฒนามาเมื่อประมาณ 20 ปีก่อน ในขณะนั้นการพัฒนาอุปกรณ์อิเล็กทรอนิกส์เป็นไปอย่างล่าช้า เพราะเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์ ได้รับการพัฒนาไปอย่างรวดเร็ว ดังที่เห็นได้ว่ามีวงจรดิจิตอลอิเล็กทรอนิกส์หลายวงจร ที่แต่เดิมถูกสร้างขึ้นมาจาก

ชิ้นส่วนอุปกรณ์อิเล็กทรอนิกส์จำนวนหลายชิ้น ประกอบกันอยู่บนแผงวงจรไฟฟ้า ที่มีขนาดใหญ่ แต่ในปัจจุบัน สามารถที่จะใช้เทคโนโลยีการออกแบบและผลิตวงจรรวมขนาดใหญ่มาก รวมอุปกรณ์ต่าง ๆ เหล่านั้นให้อยู่บนชิ้นอุปกรณ์สารกึ่งตัวนำ ที่มีขนาดประมาณ 1-2 ตร.ซม. ได้ ซึ่งเป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้น ตลอดจนความน่าเชื่อถือและความคงทนต่อสภาพแวดล้อมสูง ขณะเดียวกันนั้นในวงการทหารได้มีการนำระบบคอมพิวเตอร์และอิเล็กทรอนิกส์มาใช้ในระบบอาวุธอย่างแพร่หลาย ดังนั้นอุปกรณ์ที่มีใช้อยู่จึงไม่เหมาะสมกับเทคโนโลยีด้านอาวุธของประเทศคู่แข่งขัน การที่จะเปลี่ยนอุปกรณ์ใหม่เป็นสิ่งที่ต้องใช้งบประมาณมาก และก็จะประสบกับปัญหาเช่นเดิมคือ อุปกรณ์ใหม่ได้รับการพัฒนามานานแล้วเช่นกัน เพราะในขณะนั้นขั้นตอนของการออกแบบ การผลิต และการตรวจสอบวงจรต้นแบบ เป็นขบวนการที่ต้องใช้วิศวกร และเวลาสำหรับดำเนินการมาก ฉะนั้นทาง DOD จึงตั้งโครงการขึ้นมาเพื่อศึกษา วิธีการที่จะช่วยพัฒนาวงจรอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่งวงจรระบบดิจิตอล ให้สามารถนำไปผลิตได้เร็วขึ้น และโครงการดังกล่าวมีชื่อว่า Very High Speed Integrated Circuits หรือ VHSIC ในระยะแรกนั้นโครงการเป็นความลับของประเทศและอยู่ในความดูแลควบคุมของ United States International Traffic and Arms Regulations (ITAR) ในปี ค.ศ. 1983 ตามคำแนะนำของคณะทำงาน (Woods Hole workshop) ทาง DODได้กำหนดความต้องการมาตรฐานของภาษาที่ใช้สำหรับบรรยายพฤติกรรมของ วงจรหรือฮาร์ดแวร์ของระบบสำหรับโครงการ VHSIC ซึ่งมีสาระสำคัญพอสรุปได้ดังนี้

1. ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถจะเข้าใจได้ทั้งคนและเครื่องโดยไม่ต้องมีการแปลหรือเปลี่ยนแปลงอีก

2. สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้ (Project Documentation)

3. ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร (Simulation Language)

ฉะนั้นภาษาดังกล่าวนี้จึงเป็นภาษาโปรแกรมระดับสูง (High Level Language) เช่นเดียวกับภาษา PASCAL, FORTRAN และ ADA ซึ่งในทางวิศวกรรมการออกแบบฮาร์ดแวร์ เรียกว่า Hardware Description Language หรือ HDL ดังนั้นภาษามาตรฐานนี้จึงมีชื่อว่า VHSIC-HDL หรือ VHDL นั้นเอง

โครงการ DOD ได้มอบหมายให้บริษัท IBM, Texas Instruments และ Intermetrics เป็นผู้ศึกษาและพัฒนา การดำเนินงานได้กระทำไปอย่างต่อเนื่อง และได้ผลเป็นที่น่าพอใจ จนกระทั่งปี ค.ศ. 1985 ทาง ITAR ได้ยกเลิกข้อจำกัดในการถ่ายทอดเทคโนโลยีทางการทหารออกจากโครงการนี้ ดังนั้น VHDL จึงเริ่มเป็นที่รู้จักกันโดยทั่วไป จนกระทั่งทาง IEEE จึงได้รับภาษานี้เข้ามาศึกษาประมาณปี ค.ศ. 1987 ได้ยอมรับกำหนดมาตรฐานของภาษา โดยให้ชื่อว่า IEEE 1076-1987 และมี

ชื่อเรียกว่า VHDL มาตรฐานนี้ได้รับการปรับปรุงจนปัจจุบัน ได้ชื่อว่า IEEE 1076-1993 หรือ VHDL 1993

Terminology and Conventions

การเขียนรูปแบบของระบบดิจิตอลด้วยภาษา VHDL นั้นจะมีศัพท์เทคนิคเฉพาะ ฉะนั้นจะอธิบายศัพท์บางคำที่จะต้องพบ

1. ลักษณะของรูปแบบ (model styles) : ลักษณะของการเขียนรูปแบบด้วยภาษา VHDL สามารถแบ่งได้เป็น

- Behavioral Model : หรือเรียกอีกอย่างได้ว่า Algorithmic description เป็นรูปแบบที่บรรยายพฤติกรรมของระบบดิจิตอล ในส่วนที่บรรยายมีโครงสร้างคล้ายกับภาษาชั้นสูง (High Level Language) ทั่วๆ ไป เช่น PASCAL หรือภาษา C เป็นต้น ในการจำลองการทำงานคำสั่งแต่ละคำสั่ง (Statement) จะถูกประเมินผลเป็นไปตามลำดับ (Sequential) จากบนลงล่าง ยกเว้นในกรณีของคำสั่ง LOOP หรือการเรียกใช้โปรแกรมย่อย รูปแบบนี้จะไม่ให้รายละเอียดที่เกี่ยวกับผลิตหรือโครงสร้างของฮาร์ดแวร์ แต่ในทางตรงข้ามจะให้รายละเอียดเกี่ยวกับความสัมพันธ์ระหว่างอินพุตกับเอาต์พุต
- Dtaflow Model : เรียกอีกอย่างหนึ่งได้ว่า "Register Transfer Level" (RTL) เป็นรูปแบบที่ถูกเขียนขึ้นเพื่อจุดประสงค์ที่จะใช้เครื่องมือสำหรับสังเคราะห์วงจรอัตโนมัติรูปแบบลักษณะนี้ส่วนใหญ่จะเป็น Procedural Constructs และ Functional Operators
- Structural Model : เป็นรูปแบบที่แสดงการเชื่อมต่อกันระหว่างอุปกรณ์ต่างๆ ที่ประกอบกันขึ้นเป็นวงจรหรือระบบดิจิตอลและสามารถเรียกอีกอย่างได้ว่า "Netlist Representation" เป็นการเขียนที่แสดงให้เห็นโครงสร้างของฮาร์ดแวร์
- Mixed-Level Model : จากความอ่อนตัวของภาษา VHDL จึงสามารถที่จะเขียนรูปแบบโดยใช้ลักษณะต่างๆ ที่กล่าวมาแล้วข้างต้น บรรยายวงจรหรือระบบดิจิตอลเดียวกันได้ฉะนั้นรูปแบบเช่นนี้จึงมีการเขียนแบบผสม

2. Concurrency : ในภาษา VHDL นั้นชุดคำสั่งแต่ละชุดจะทำงานในเวลาเดียวกันและอิสระต่อกันซึ่งเป็นคุณสมบัติที่เป็นความจริงทางฟิสิกส์ของวงจรอิเล็กทรอนิกส์ ชุดคำสั่งนี้เรียกว่า "Concurrent Statement" และจะทำงานก็ต่อเมื่อมีการเปลี่ยนแปลงค่าของสัญญาณ

3. Sequential : นอกจากความสามารถที่จะทำงานแบบ Concurrent แล้ว บางครั้งการเขียนรูปแบบในลักษณะที่บรรยายพฤติกรรมของวงจรมีความจำเป็นที่จะต้องให้ชุดคำสั่งทำงานเป็นลำดับขั้นเรียงกันจากบนลงล่าง อย่างเช่นการเขียนแบบ Behavioral Model เป็นต้น ชุดคำสั่งที่เป็น Sequential นี้จะใช้ในโปรแกรมย่อยและ Process Statement

4. Driver : สัญญาณต่างๆ ใน VHDL นั้นจะถูกควบคุมด้วยตัวขับหรือ "Driver" สัญญาณเหล่านี้จะรับค่าใหม่ (ระดับของสัญญาณ) ได้ด้วยตัวขับนี้เอง

5. Transaction : การเกิด Transaction กับ Signal นั้นเกิดขึ้นเมื่อมีการกำหนดค่าๆ หนึ่งให้กับสัญญาณนั้น ค่าใหม่ที่สัญญาณได้รับอาจจะมีหรือไม่มีผลทำให้เกิดการเปลี่ยนแปลงของระดับสัญญาณ (event) เช่นการเปลี่ยนจากค่า Logic '0' เป็นค่า Logic '1' เป็นต้น

6. Event : คือการเปลี่ยนระดับค่าของ SIGNAL จากระดับหนึ่งไปสู่ระดับอื่น อย่างเช่นในระบบดิจิตอลการเปลี่ยนจาก Logic '0' เป็น Logic '1' หรือในทางตรงกันข้ามถือว่า SIGNAL นั้นเกิด "Event" ฉะนั้นจะเห็นได้ว่า การที่จะเกิด Event ได้นั้นจะต้องเกิด Transaction ด้วย แต่ในทางตรงข้ามการเกิด Transaction ไม่จำเป็นต้องเกิด Event ทุกครั้ง

7. Sensitivity List : คือรายชื่อของ SIGNAL ต่างๆ ที่มีผลให้เกิดการทำงานของ Concurrent Statement เมื่อเกิด Event ขึ้นกับ SIGNAL ตัวใดตัวหนึ่งหรือหลายตัวพร้อมกันในรายชื่อนั้น

8. Objects : ในภาษา VHDL นั้นคำว่า Object ใช้เขียนเพื่อบ่งบอกถึงองค์ประกอบส่วนหนึ่งของรูปแบบ ซึ่งเปรียบได้เหมือนกับภาชนะที่มีไว้สำหรับบรรจุค่าต่างๆ สามารถแบ่งออกได้เป็นสามชั้นด้วยกันคือ

- CONSTANT : ได้แก่ Object ประเภทหนึ่งที่เมื่อกำหนดค่าเริ่มต้นให้แล้วจะคงค่านั้นไว้ตลอด ไม่สามารถดัดแปลงหรือแก้ไขได้ สามารถประกาศใช้ได้ในส่วนที่เป็นส่วนประกาศของรูปแบบ
- SIGNAL : หมายถึง Object ประเภทหนึ่งที่สามารถกำหนดค่าที่สัมพันธ์กับเวลาให้ได้นั้นหมายความว่า SIGNAL สามารถรับค่าได้เพียงค่าเดียวเท่านั้นในขณะเวลาหนึ่ง SIGNAL จะรับค่าๆ หนึ่งได้จากตัวขับสัญญาณหรือ Driver ซึ่งตัวขับนี้อาจจะเก็บค่าในอนาคตสำหรับ SIGNAL ไว้ด้วย SIGNAL สามารถประกาศใช้ได้ในส่วนที่เป็นเนื้อที่ของ Concurrent Body เท่านั้น ดังนั้น SIGNAL จึงสามารถถูกนำไปใช้ได้ตลอดโครงสร้างของรูปแบบหรือที่เรียกว่า Gobal Object

- VARIABLE : หรือตัวแปรได้แก่ Object ที่สามารถกำหนดค่าใดๆ ได้และสามารถที่จะเปลี่ยนแปลงค่าได้ตลอดการจำลองการทำงาน แต่จะเก็บค่าเพียงค่าเดียวเท่านั้น เนื่องจาก VARIABLE สามารถประกาศใช้ได้ในส่วนที่เป็นส่วนประกาศของ PROCESS, FUNCTION หรือ PROCEDURE ดังนั้น VARIABLE จึงสามารถนำไปใช้ได้เฉพาะในขอบเขตที่ถูกประกาศใช้

2.4.1โครงสร้างของภาษา VHDL

VHDL นั้นประกอบด้วยส่วนต่างๆ ที่สำคัญและเป็นพื้นฐานของการเขียนรูปแบบระบบดิจิตอลที่สำคัญ 4 หน่วยคือ

1. Entity Design Unit
2. Architecture Design Unit
3. Package Design Unit
4. Configuration Design Unit

2.4.2 ลักษณะของการเขียนบรรยายด้วยภาษา VHDL

การบรรยายเชิงพฤติกรรม

เป็นการบรรยายลักษณะการเปลี่ยนแปลงของข้อมูลในรูปแบบของอัลกอริธึม สำหรับการคำนวณผลลัพธ์ที่เกิดขึ้นสืบเนื่องมาจากการเปลี่ยนแปลงสภาวะของข้อมูลที่เข้ามาโดยไม่คำนึงถึงว่าลักษณะโครงสร้างหรือความสัมพันธ์ของอุปกรณ์ที่อยู่ภายใน

การบรรยายเชิงข้อมูล

เป็นการบรรยายถึงการเลื่อนไหลของข้อมูลผ่านรีจิสเตอร์และบัสของระบบ เป็นระดับขั้นการบรรยายที่อยู่ตรงกลางระหว่างการบรรยายเชิงพฤติกรรมและการบรรยายเชิงโครงสร้าง เครื่องมือที่ใช้ในการควบคุมการเลื่อนไหลของข้อมูลได้แก่ Conditional Selected และ Guarded

การบรรยายเชิงโครงสร้าง

การบรรยายการทำงานของระบบในเชิงโครงสร้างจะต้องแสดงรายการของอุปกรณ์ทั้งหมดที่ใช้ในระบบและต้องกำหนดการเชื่อมต่อระหว่างอุปกรณ์ต่าง ๆ ด้วย เพราะว่าการบรรยายในระดับนี้เป็นการบรรยายที่ใกล้เคียงลักษณะของฮาร์ดแวร์จริงที่สุด

2.5 ทฤษฎีและหลักการของ FPGAs

2.5.1 โครงสร้างภายในของอุปกรณ์ FPGAs (เบอร์ XC4010E)

FPGAs จัดเป็นวงจรรวมเฉพาะกิจชนิดหนึ่งที่สามารถโปรแกรมเป็นวงจรเชิงเลขใดๆ ก็ได้ เช่นเดียวกับ EPLD ต่างกันที่ EPLD โปรแกรมลงภายในอีพรอม และสามารถโปรแกรมใหม่ได้หลังจากนำไปลบด้วยแสง UV แต่ FPGAs โปรแกรมลงบนสแตติกแรมภายในด้วยข้อมูลที่อยู่ภายนอก และสามารถโปรแกรมใหม่ได้โดยการรีเซตด้วยสัญญาณไฟฟ้า นอกจากนั้น FPGAs ยังประหยัดไฟและมีความจุวงจรสูง (จำนวนเกตมาก) ได้อีกด้วย

วงจรรวมชนิดที่ใช้ในโครงงานนี้ผลิตโดยบริษัทไซลิงค์ (Xilinx) ซึ่งเป็นบริษัทที่ทำการค้นคว้าร่วมกับบริษัทเอ็มเอ็มไอ (MMI) สร้างเป็นกลุ่มของเกตจำนวน 600-25,000 เกท ดังแสดงในตารางที่ 2.1 การที่ต้องบอกขนาดของวงจรรวมเป็นจำนวนเกตเพราะจะได้รู้ว่าขนาดของวงจรที่ได้ออกแบบไว้สามารถโปรแกรมลงบนวงจรรวม FPGAs ได้หรือไม่

ตารางที่ 2.1 คุณสมบัติของ FPGAs ตระกูลต่าง ๆ

FPGAs	Appr.Gate Count	Max I/OS	Flip-Flops	RAM bits	Available CLBs
XC4002A	2000	64		2048	64
XC4003/4003A	3000	80		3200	100
XC4003H	3000	160		3200	100
XC4004A	4000	960		4608	144
XC4005/4005A	5000	122		6072	196
XC4005H	5000	192		6272	196
XC4006	6000	128		8192	256
XC4008	8000	144		10368	324
XC4010E*	**10000**	**160**		**12800**	**400**
XC4013	13000	192		18432	576
XC4025	25000	256		32768	1024

หมายเหตุ * หมายถึงเบอร์ที่ใช้ในโครงงานชิ้นนี้

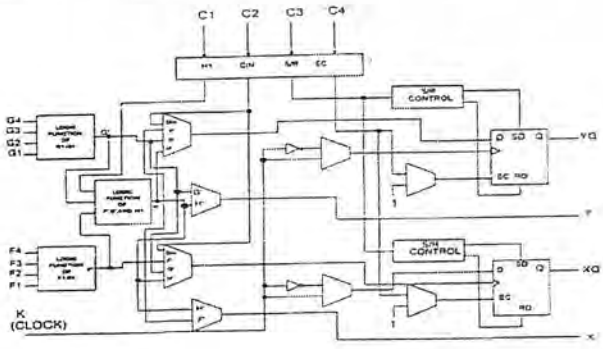
FPGAs มีโครงสร้างภายในใกล้เคียงกับสถาปัตยกรรมของเกตอะเรย์ (GAL, Gate Array Logic) มาก สามารถโปรแกรมและลบคอนฟิกกูเรชั่น (Configuration) ภายในสแตติกแรม (Static RAM) ได้โดยใช้กระแสไฟฟ้า ซึ่งทำการโปรแกรมได้โดยดึงข้อมูลฐานสิบหกมาจากภายนอก เช่น Parallel EPROM หรือ Serial PROM ต่างกับ EPLD, PAL ที่มี EPROM อยู่ในตัวภายใน FPGAs จะจัดเรียงเป็นลอจิกเซลล้อมรอบภายนอกด้วยอินพุตเอาต์พุตเซล FPGAs ตัวแรกที่ผลิตโดยบริษัทไซลิงค์คือเบอร์ XC2064 (2000 Family) ประกอบด้วยเซลเรียงกันเป็นเมตริกซ์ (Matrix) เป็นจำนวน 64 เซล หลังจากนั้นผลิต FPGAs ตระกูล 3000 และ 4000 มีโครงสร้างซับซ้อนขึ้นสามารถเพิ่มจำนวนเกตได้มากขึ้นและดีขึ้น แต่ละเซลเรียกว่า CLB (Configurable Logic Block)

ส่วนที่เป็นองค์ประกอบของลอจิก (Configurable Logic Block)

CLB จะจัดเรียงกันเป็นแบบเมตริกซ์แบบอะเรย์ขนาด M x N การออกแบบนั้นสามารถทำได้โดยการจัดการวาง CLB และต่อเชื่อมขาของ CLB ให้ต่อถึงกัน เราสามารถจัด CLB ให้เชื่อมต่อถึงกันได้โดยการทำด้วยมือหรือให้โปรแกรมที่สนับสนุน FPGAs ทำให้โดยอัตโนมัติ โดยวิธีของมันเองสำหรับไฟล์ที่ได้จากโปรแกรมเหล่านี้ เราเรียกว่าไฟล์ที่กำหนดการวางอุปกรณ์ (Configuration File) ซึ่งจะบรรจุโครงร่างภายในของ CLB ตามความเหมาะสม อีกด้านหนึ่งไฟล์กำหนดการวางอุปกรณ์นั้นจะเป็นไฟล์กระแสข้อมูล (Bit Stream) ซึ่งสามารถใช้โปรแกรมหน่วยความจำภายในของ FPGAs ได้ สำหรับรูปที่ 2.6 แสดง CLB ของ FPGAs ตระกูล 4000

ส่วนอินพุตและเอาต์พุตของอุปกรณ์ FPGAs

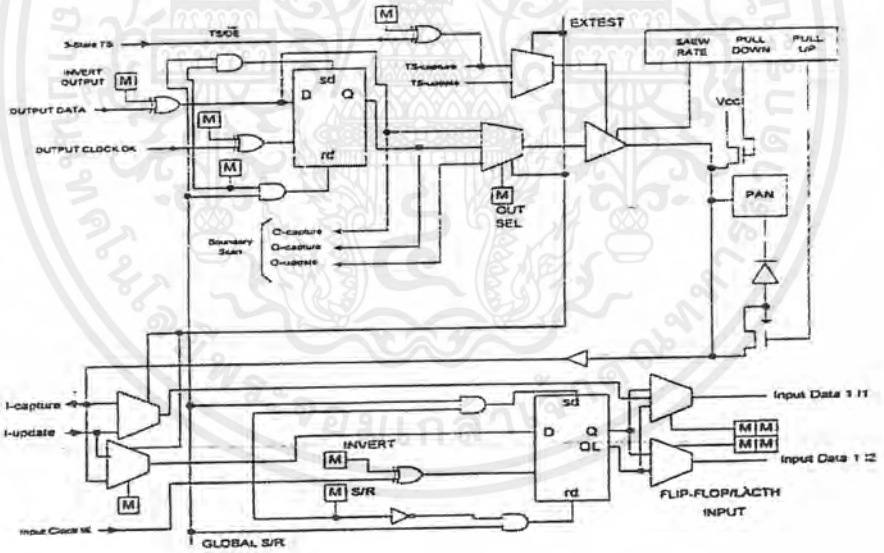
รอบนอกของ FPGAs จะประกอบด้วย IOBs ประมาณ 64 ถึง 144 ตัว ซึ่งขึ้นอยู่กับตระกูลของ FPGAs จะเป็นตัวเชื่อมต่อระหว่างภายในกับภายนอกของวงจรลอจิกของ FPGAs ลักษณะของ IOBs จะมีลักษณะ 2 ทิศทาง สามารถโปรแกรมให้เป็นอินพุตหรือเอาต์พุตก็ได้ สำหรับรูปที่ 2.7 แสดง IOBs ของ FPGAs ตระกูล 4000



รูปที่ 2.6 แผนผัง CLB ของตระกูล 4000

2.5.2 รายละเอียดการใช้งานของอุปกรณ์ FPGAs

FPGAs สามารถทำงานได้หลายลักษณะ โดยกำหนดได้ที่ขาสัญญาณ M0 M1 M2 ดังแสดงในตารางที่ 2.2 ในลักษณะมาสเตอร์พาราแรล (Master Parallel Mode) รับโปรแกรมคอนฟิกทีละ 1 ไบท์ (Byte) จากหน่วยความจำภายนอกที่เป็นแบบขนาน โดยสามารถรับโปรแกรมคอนฟิก (Cofig) จากแอดเดรส (Address) ต่ำหรือสูงก่อนก็ได้ การต่อลักษณะเพอริเฟอรัล (Peripheral) จะรับโปรแกรมคอนฟิกทีละ 1 ไบท์ จากไมโครโปรเซสเซอร์ โดยสามารถโต้ตอบกันได้ว่าพร้อมหรือไม่ที่จะรับข้อมูลต่อไป การต่อลักษณะสเลฟซีเรียล (Slave Serial) จะรับโปรแกรมคอนฟิกทีละ 1 บิต จากไมโครโปรเซสเซอร์ตามสัญญาณอินพุต CCLK ส่วนการต่อลักษณะมาสเตอร์ซีเรียล (Master Serial) จะรับโปรแกรมคอนฟิกทีละ 1 บิต จากหน่วยความจำภายนอกที่เป็นแบบอนุกรม



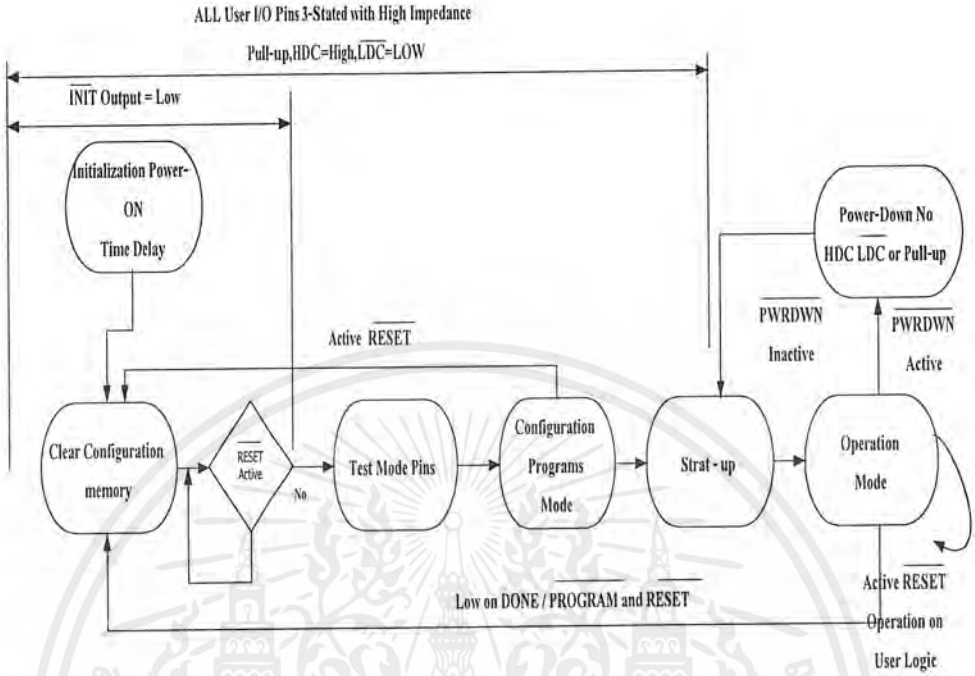
รูปที่ 2.7 แผนผัง IOBs ของตระกูล 4000

ตารางที่ 2.2 โหมดต่าง ๆ ของการคอนฟิกกูเรชัน

Mode	M2	M1	M0	CCLK	Date
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	1	1	0	Output	Byte-Wide,3FFFF
down	0	1	1	Input	down
Peripheral Synchr.	1	0	1	Output	Byte-Wide
Peripheral Asynchr.	0	1	0	---	Byte-Wide
Reserved	0	0	1	---	
Reserved					

จากความต้องการสร้างให้ใช้กระแสไฟฟ้าต่ำ จากลักษณะการต่อใช้งานทั้ง 5 แบบ จึงมีเพียง 2 แบบเท่านั้นที่เหมาะสม คือ แบบมาสเตอร์ซีเรียลและแบบสเลฟซีเรียล ส่วนแบบมาสเตอร์พาราแรล ต้องใช้ EPROM เบอร์ 27CXXX ซึ่งกินกระแสมากกว่า PROM เบอร์ XC17XXX เหมาะในการทดสอบต้นแบบก่อน เมื่อวงจรต้นแบบทำงานได้ถูกต้องแล้วจึงทำการอัดโปรแกรมลง PROM อีกทีหนึ่งเพราะว่าในแบบพาราแรลนั้น EPROM สามารถโปรแกรมได้ใหม่ต่างกับ PROM ที่โปรแกรมเพียงได้ครั้งเดียว

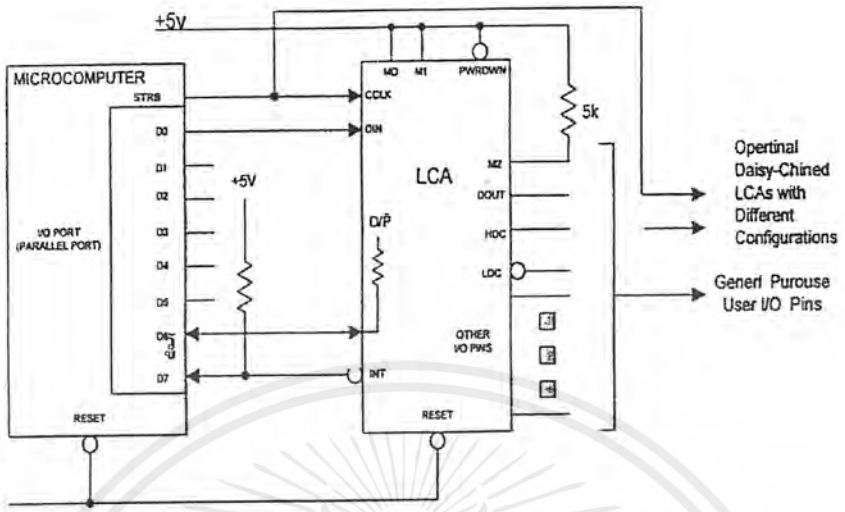
การใช้งาน FPGAs ในการต่อลักษณะสเลฟซีเรียลและมาสเตอร์ซีเรียล เมื่อเริ่มจ่ายไฟเข้าตัว FPGAs จะทำการลบข้อมูลหน่วยความจำที่ใช้ในคอนฟิก (Configuration Memory) ตรวจสอบลักษณะการคอนฟิกว่าเป็นลักษณะใดในตารางที่ 2.2 ว่าเป็นแบบอนุกรมหรือขนาน หลังจากนั้นจะเริ่มทำการโปรแกรมคอนฟิกสัญญาณ DONE/PROGRAM เป็น “0” ซึ่งอยู่ในระหว่างโปรแกรมและเมื่อข้อมูลในคอนฟิกที่รับมาจากภายนอกเต็มหน่วยความจำที่ใช้ในการคอนฟิก และความยาวของข้อมูลตรงกับที่ส่วนหัวของข้อมูลคอนฟิก สัญญาณ DONE/PROGRAM จะเป็น “1” ซึ่งหมายถึงโปรแกรมทำการคอนฟิกเสร็จสิ้น ดูรูปที่ 2.8 ประกอบ



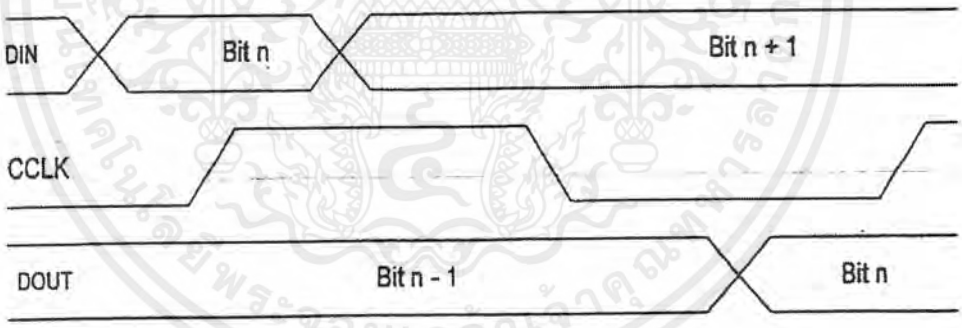
รูปที่ 2.8 ลำดับไดอะแกรมในการคอนฟิก

การใช้งานในลักษณะสเลฟซีเรียล

การต่อใช้งานในลักษณะนี้ เหมาะสมกับวงจรที่ออกแบบมาเพื่อทำงานร่วมกับไมโครคอมพิวเตอร์อยู่แล้ว ทั้งนี้เพราะ FPGAs ได้ใช้ความสามารถของไมโครคอมพิวเตอร์ในการเก็บและส่งข้อมูลคอนฟิกให้ เพียงแต่ต้องเขียนโปรแกรมเพื่อส่งโปรแกรมคอนฟิกให้เพิ่มลักษณะการต่อในลักษณะนี้เป็นดังรูปที่ 2.9 ซึ่งไมโครคอมพิวเตอร์จะสร้างสัญญาณเพื่อทำการคอนฟิกให้กับอุปกรณ์ FPGAs การป้อนโปรแกรมคอนฟิกให้ FPGAs ทำได้โดยต่อสัญญาณ Strobe เข้ากับขา CCLK และพอร์ต DO เข้ากับขา DIN สร้างสัญญาณคล็อกป้อนที่ขา CCLK และป้อนโปรแกรมคอนฟิกแบบอนุกรมเข้าที่ขา DIN ดังแผนภูมิในรูปที่ 2.10



รูปที่ 2.9 การต่อใช้งานในลักษณะสเลฟซีเรียล

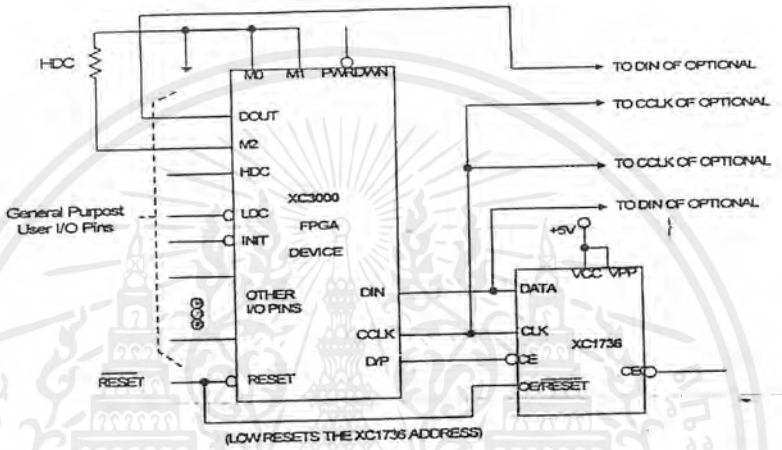


รูปที่ 2.10 แผนภูมิเวลาการป้อนข้อมูลโปรแกรมคอนฟิกในลักษณะสเลฟซีเรียล

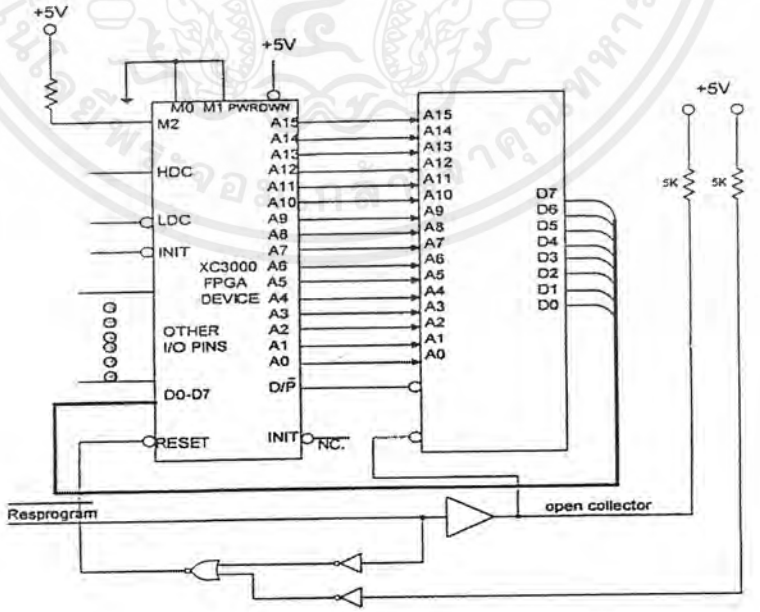
การใช้งานในลักษณะมาสเตอร์ซีเรียล

การต่อใช้งานในลักษณะนี้ ส่วนที่เก็บโปรแกรมคอนฟิกจะต่างจากการต่อลักษณะแรก คือ ใช้ PROM เบอร์ XC17XXX เป็นตัวเก็บโปรแกรม ทำให้ไม่ต้องเสียเวลาเขียนโปรแกรมเพื่อทำการคอนฟิก ซึ่งวิธีการอัดโปรแกรมคอนฟิกลง PROM ทำตามขั้นตอนดังนี้คือ เมคบิต (MakeBits) สร้างไฟล์ .BIT จากวงจรที่ออกแบบ และใช้โปรแกรม MakePROM สร้าง Hex ไฟล์แล้วทำการอัดโปรแกรมลง PROM ด้วยอุปกรณ์อัด PROM ที่มาพร้อมกับตัวโปรแกรมของ

ไซลิงค์ PROM XC17XXX จะส่งสัญญาณเพื่อทำการคอนฟิกให้กับอุปกรณ์ FPGAs ดังแสดงในรูปที่ 2.11 ขา D0-D7 เป็นขารับข้อมูลที่ใช้ในการคอนฟิกแบบขนาน A0-A15 เป็นแอดเดรสที่ FPGAs สร้างให้กับ EPROM เพื่ออ่านข้อมูลจากหน่วยความจำมาเก็บไว้ในสแตติกแรม (StaticRAM) แอดเดรสทั้ง 16 เส้นไม่จำเป็นต้องต่อให้ครบก็ได้ ขึ้นอยู่กับขนาดหน่วยความจำ EPROM ที่ใช้ และสามารถกำหนดให้นับขึ้นหรือลงได้



รูปที่ 2.11 การต่อใช้งานในลักษณะมาสเตอร์ซีเรียล



รูปที่ 2.12 การต่อใช้งานในลักษณะมาสเตอร์พาราแรล

2.5.3 ข้อควรระวังในการใช้อุปกรณ์ FPGAs

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

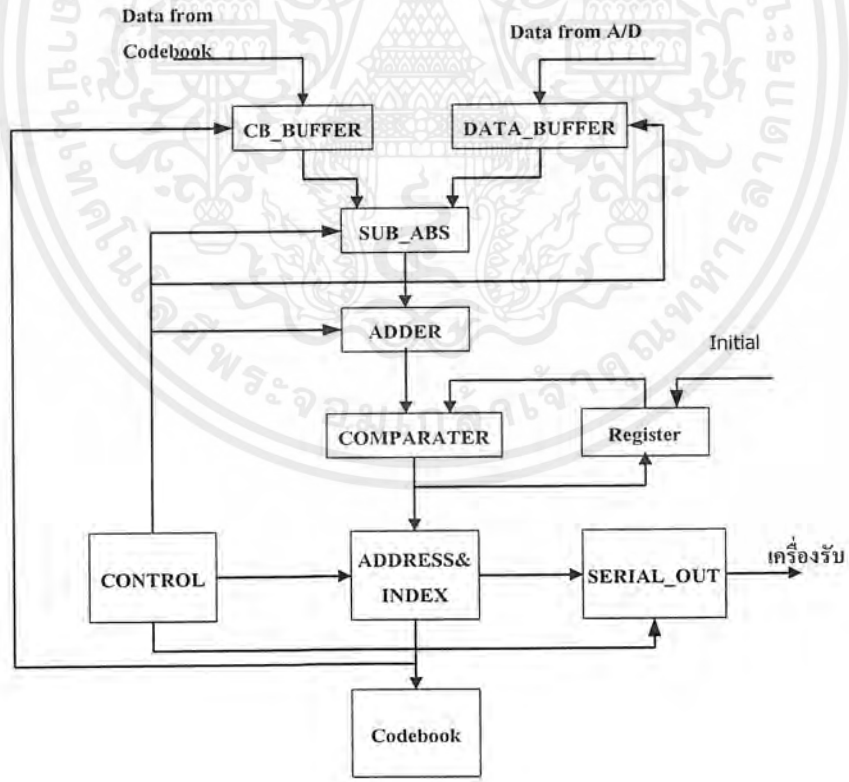
การป้องกันไอซีจากแรงดันไฟฟ้า ไม่ควรต่อสลับขั้วบวกกับขั้วลบจะทำให้ไอซีเสียได้ นอกจากนั้น แรงดันของแหล่งจ่ายไฟต้องอยู่ในช่วงที่โรงงานกำหนดมา สำหรับ FPGAs ค่าแรงดันที่ใช้งานอยู่ในช่วง Vcc = 4.75-5.25 V และแรงดันที่ทนได้อยู่ในช่วง –0.5 - 7 V ดังนั้นก่อนป้อนแรงดันควรตรวจเช็คให้แน่ใจก่อน

บทที่ 3
การออกแบบและการสร้าง

ในการออกแบบและการสร้างตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊คจะมีลำดับขั้นตอนการออกแบบโปรแกรมการใช้งาน ซึ่งจะใช้โปรแกรม Schemetic ของบริษัท Xilinx และในส่วนของชิ้นงานได้มีการปรับปรุงเพิ่มเติมต่อจากของรุ่นพี่ที่ไม่ได้ใช้งานแล้ว ซึ่งจะมีลำดับขั้นตอนการออกแบบและการสร้างดังต่อไปนี้

3.1 หลักการออกแบบ

ในการออกแบบจะมีขั้นตอนการออกแบบตามรูปที่ 3.1 ซึ่งเป็นขั้นตอนทั้งหมดของตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

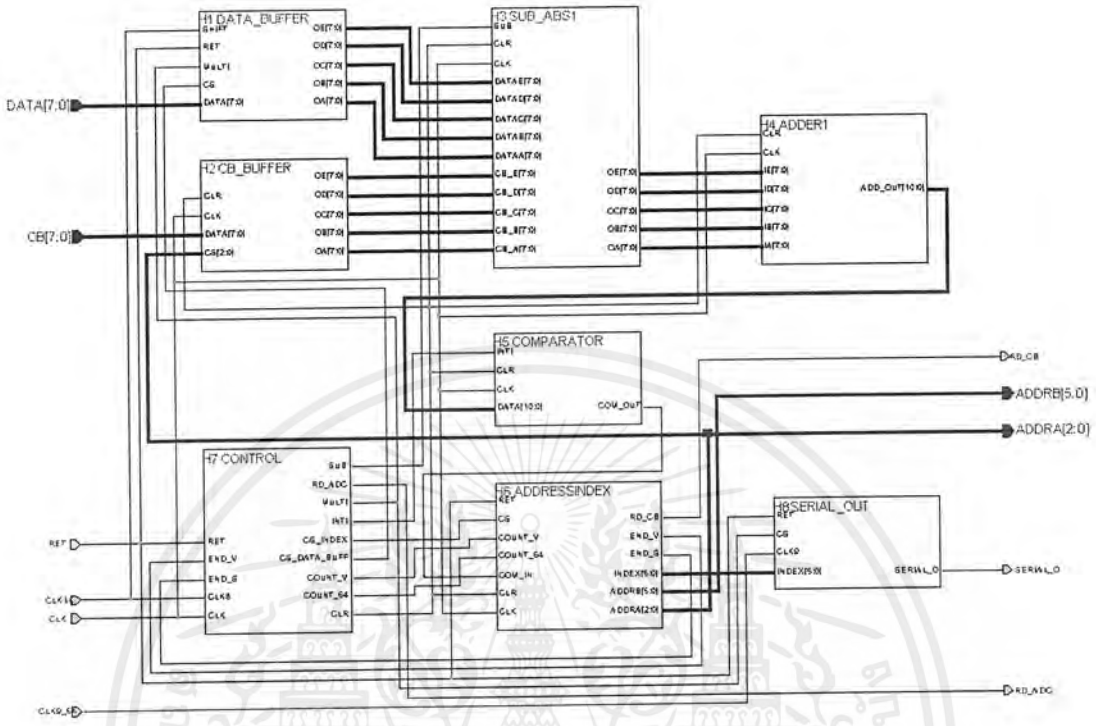


รูปที่ 3.1 แผนผังการทำงานของตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

ตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค เป็นการนำสัญญาณจากตัวแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอลมาเปรียบเทียบกับสัญญาณจากโค้ดบุ๊ค ว่ามีค่าใกล้เคียงกับสัญญาณในโค้ดบุ๊คตำแหน่งใดมากที่สุด แล้วทำการส่งตำแหน่งของสัญญาณชุดนั้นออกไปทางด้านเครื่องรับ ซึ่งมีหลักการทำงาน โดยภาค DATA_BUFFER จะรับสัญญาณมาจากตัวแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอลมาเก็บไว้และนำสัญญาณจาก DATA_BUFFER จำนวน 5 อีลีเมนต์มาลบกับสัญญาณจากโค้ดบุ๊คจำนวน 5 อีลีเมนต์ ซึ่งเก็บอยู่ใน CB_BUFFER เป็นการลบแบบไม่คิดเครื่องหมาย จากนั้นนำผลลัพธ์ที่ได้จากการลบจำนวน 5 อีลีเมนต์มาบวกกันโดยภาค ADDER ผลลัพธ์ที่ได้จากการบวกนำไปเปรียบเทียบกับค่าที่เก็บไว้ในรีจิสเตอร์ ซึ่งในครั้งแรกค่าในรีจิสเตอร์จะถูกกำหนดเป็นค่าสูงสุด หากผลลัพธ์ที่ได้จากการบวกมีค่าน้อยกว่าค่าที่เก็บอยู่ในรีจิสเตอร์ ภาค COMPARATOR จะส่งค่านี้ไปเก็บไว้แทนค่าเดิม แต่ถ้าค่าที่ได้จาก ภาค ADDER มีค่ามากกว่าค่าในรีจิสเตอร์ รีจิสเตอร์ก็จะเก็บค่าเดิมไว้ และทุกครั้งที่ผลลัพธ์จากการเปรียบเทียบมีค่าน้อยกว่าค่าเดิมในรีจิสเตอร์ ตัวเปรียบเทียบก็จะส่งสัญญาณให้ภาค ADDRESS & INDEX จำตำแหน่งของค่านั้นไว้

ตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค มีจำนวนรอบในการทำงานดังกล่าวข้างต้นจำนวน 64 ครั้งต่อสัญญาณอินพุตหนึ่งชุด ซึ่งมีรอบการทำงานเท่ากับจำนวนเวกเตอร์ของตัวโค้ดบุ๊ค และเมื่อทำงานครบ 64 ครั้ง ก็จะส่งตำแหน่งของข้อมูลที่มีค่าใกล้เคียงกับสัญญาณอินพุตมากที่สุดไปยังภาค SERIAL_OUT และภาค SERIAL_OUT จะส่งสัญญาณดังกล่าวแบบอนุกรมไปยังเครื่องรับต่อไป ซึ่งการทำงานทั้งหมดจะถูกควบคุมโดยภาค CONTROL

รูปที่ 3.2 เป็นบล็อกไดอะแกรมการทำงานของตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค มีหลักการทำงานเหมือนกับรูปที่ 3.1 ซึ่งจะแสดงถึงการออกแบบตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊คทั้งหมด



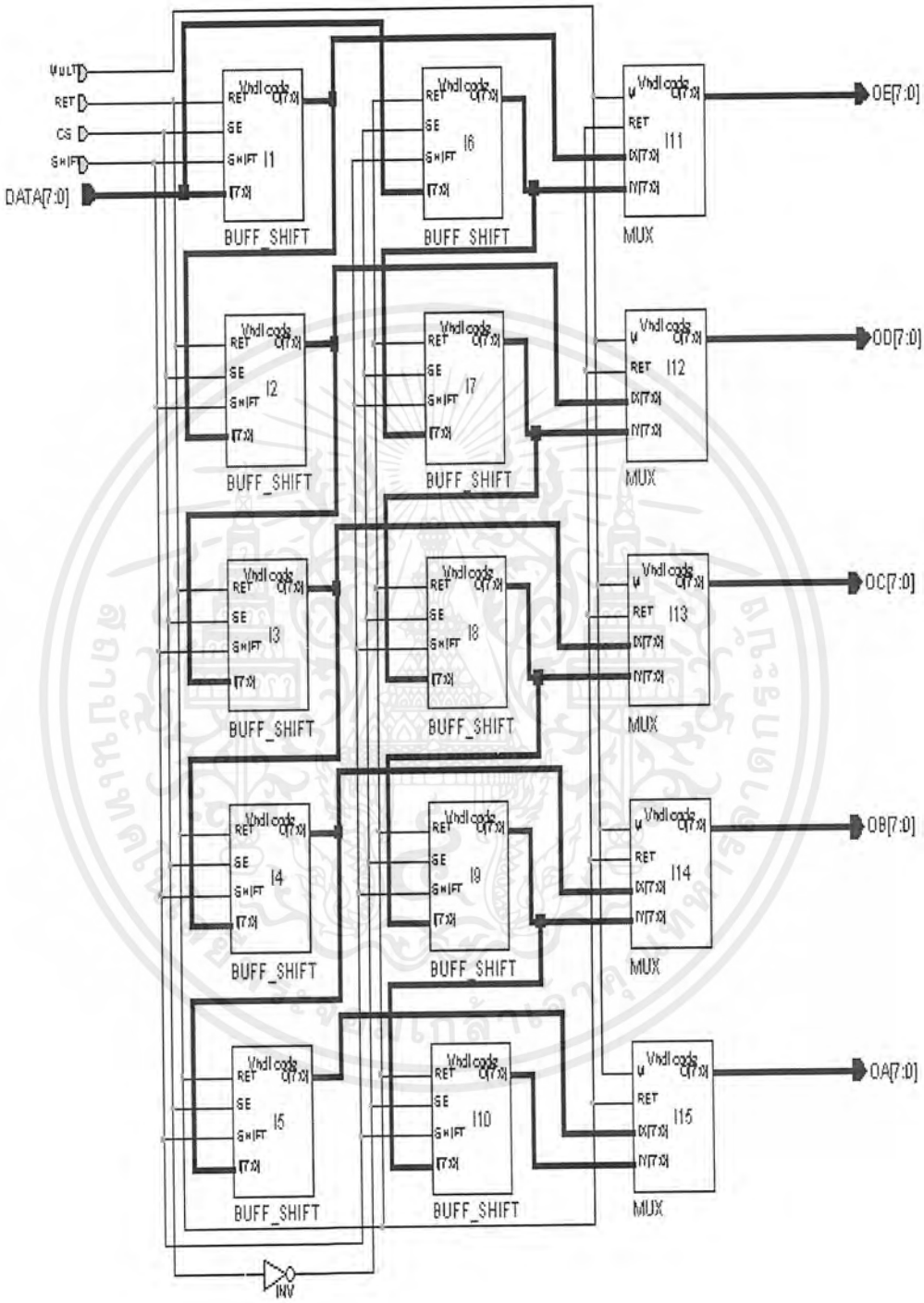
รูปที่ 3.2 บล็อกไดอะแกรมตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

3.2 ขั้นตอนการออกแบบตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

ขั้นตอนการออกแบบตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค แบ่งออกเป็นส่วนต่าง ๆ ได้ดังนี้

3.2.1 ภาค DATA_BUFFER

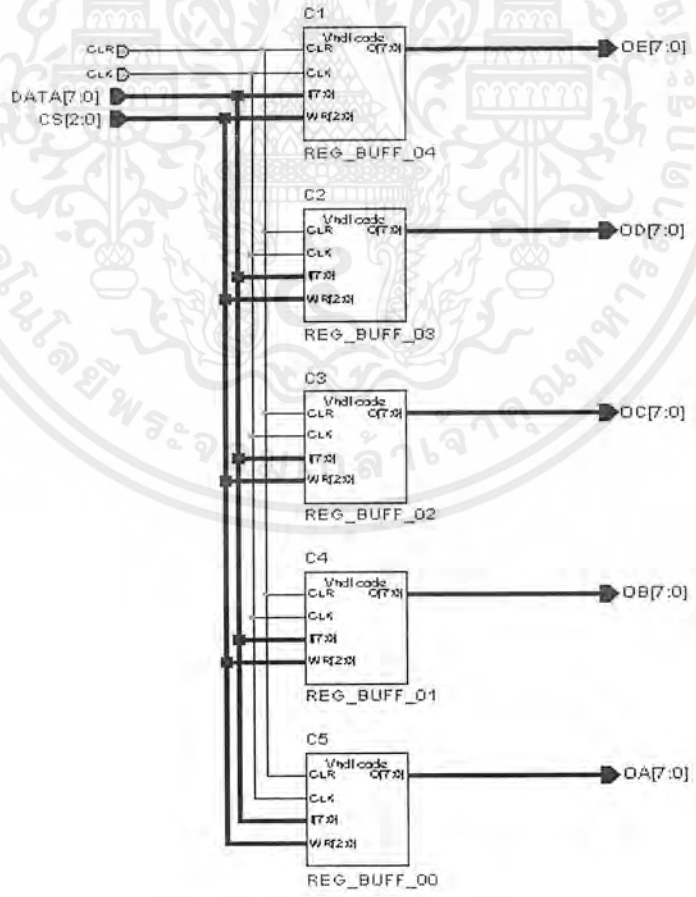
เป็นตัวเก็บสัญญาณอินพุตขนาด 5 ไบต์จำนวน 2 ชุด มีการเลื่อนข้อมูลเข้าครั้งละ 1 ไบต์โดยขา SHIFT ความถี่ 8 กิโลเฮิร์ต ในหนึ่งชุดจะทำการเลื่อนข้อมูลเข้าจำนวน 5 ไบต์เมื่อข้อมูลชุดแรกเข้ามาครบแล้ว สัญญาณจากขา CS จะควบคุมให้ตัวเก็บข้อมูลอีกชุดหนึ่งเลื่อนข้อมูลเข้ามาเก็บจนครบ 5 ไบต์เช่นกันและขาสัญญาณ MULTI จะเลือกข้อมูลชุดแรกออกไปทำการลบที่ภาค SUB_ABS ซึ่งจะสลับกันทำงานเช่นนี้เรื่อยไปดังรูปที่ 3.3



รูปที่ 3.3 วงจรภาค DATA_BUFFER

3.2.2 ภาค CB_BUFFER

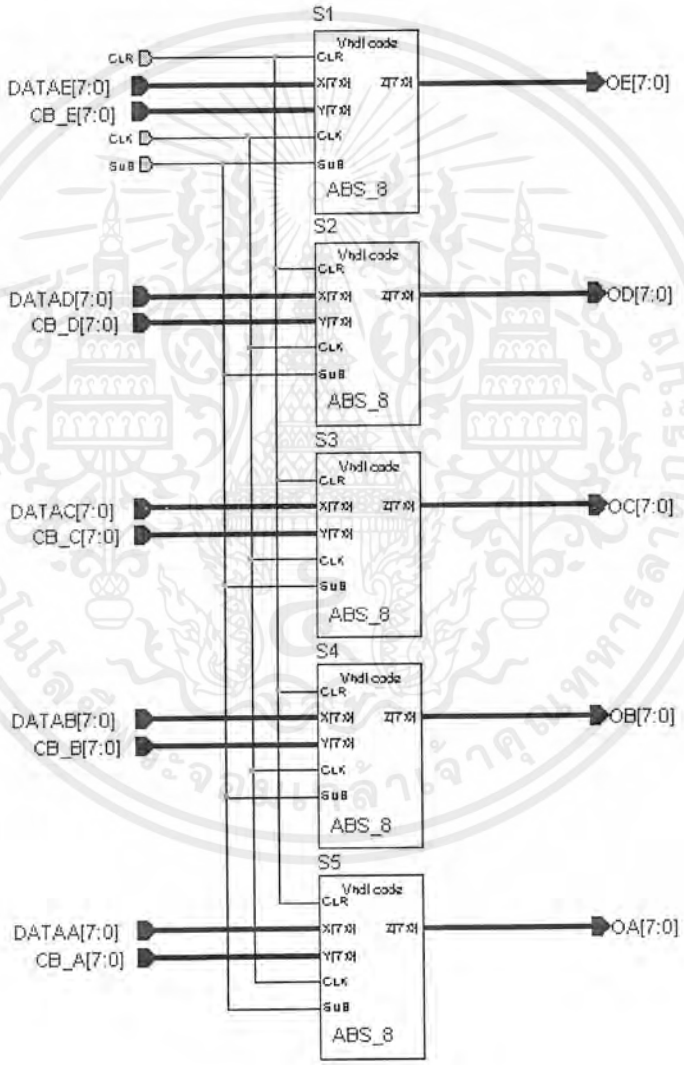
เป็นตัวรับสัญญาณจากโค้ดบุ๊ค มีขา DATA และขา Chip Select (CS) เป็นตัวกำหนดการทำงาน โดยที่ข้อมูลจะส่งมาที่ BUFFER ทุกตัวพร้อมกัน มี BUFFER อยู่ทั้งหมด 5 ตัว ซึ่งที่ตัว BUFFER จะกำหนดค่าไว้ที่ขา WR มีค่า 00, 01, 02, 03 และ 04 ตามลำดับ การทำงานจะขึ้นอยู่กับค่าของขา CS ถ้าขา CS มีค่า 02 ก็แสดงว่า BUFFER ตัวที่สามที่ถูกกำหนดค่าไว้ที่ 02 จะเป็นตัวเก็บข้อมูลไว้ ซึ่งที่ขา CS จะมีค่า 00, 01, 02, 03 และ 04 เช่นกัน ซึ่งค่าของ CS จะถูกกำหนดโดย CONTROL อีกทีหนึ่ง ฉะนั้น BUFFER จะรับข้อมูลจาก DATA ได้ก็ต่อเมื่อ ค่า WR ที่กำหนดไว้มีค่าเท่ากับค่าของ CS และ BUFFER จะสลับกันทำงานก็ต่อเมื่อ ค่าที่ขา CS เปลี่ยนไปซึ่งจะทำงานพร้อมกับการดึงข้อมูลจาก BUFFER ไปใช้งานด้วย ซึ่งจะมีขา CLK ต่อให้กับ BUFFER ทุกตัว ส่วนขา CLR จะทำงาน ก็ต่อเมื่อมีการส่งข้อมูลครบ 64 ครั้ง ก็จะเคลียร์ BUFFER ทุกตัวให้ว่างเพื่อเริ่มต้นรับค่าใหม่ดังรูปที่ 3.4



รูปที่ 3.4 วงจรภาค CB_BUFFER

3.2.3 ภาค SUB_ABS

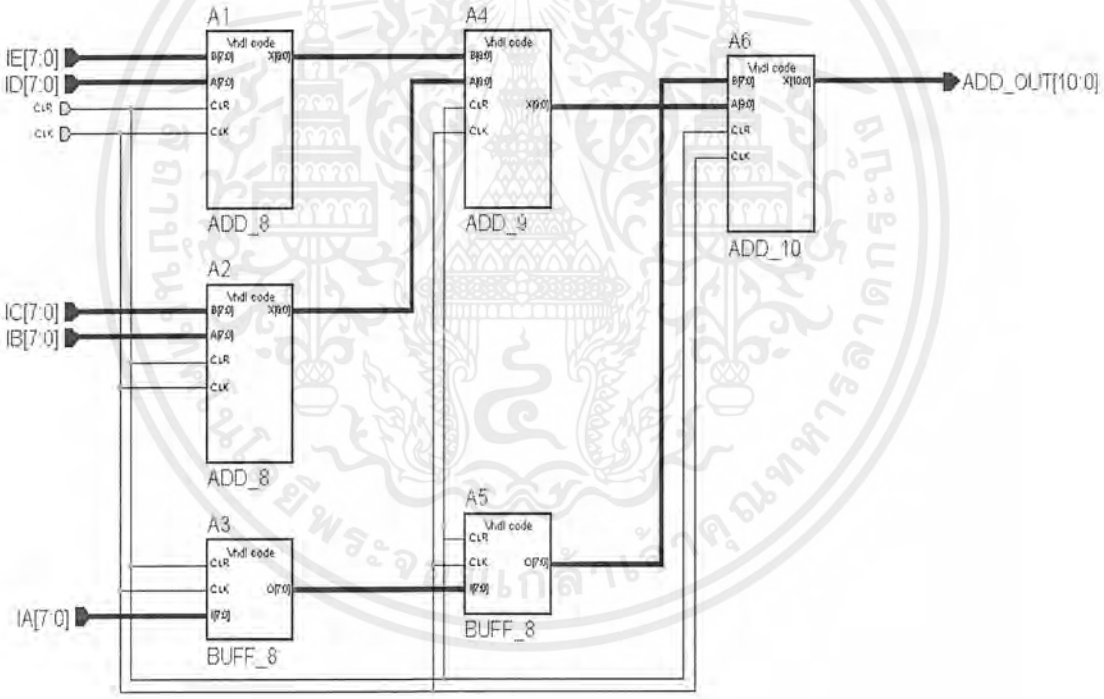
ภาค SUB_ABS ทำหน้าที่ในการลบโดยทำการลบสัญญาณที่มาจากอินพุตกับสัญญาณจากโค้ดบุ๊คโดยไม่มีการคิดเครื่องหมาย ซึ่งจะมีสัญญาณอยู่ 5 อีลีเมนต์โดยวงจร SUB_ABS จะทำงานเมื่อขา SUB มีสถานะเป็น High ดังรูปที่ 3.5



รูปที่ 3.5 วงจรภาค SUB_ABS

3.2.4 ภาค ADDER

ภาค ADDER เป็นภาคที่ทำหน้าที่ในการบวกข้อมูลที่ได้จากภาค SUB_ABS โดยมีข้อมูล 5 อีลีเมนต์มาป้อนเข้าวงจรบวก โดยแต่ละอีลีเมนต์ให้ชื่อว่า IA, IB, IC, ID และ IE ซึ่งเป็นข้อมูลขนาด 8 บิต การทำงานจะให้ข้อมูลบวกกัน 2 ชุดคือ IE บวกกับข้อมูลจาก ID และ IC บวกกับ IB ซึ่งจะได้ผลลัพธ์ที่มีขนาด 9 บิต ซึ่งเป็นการเผื่อบิตตัวทดไว้ จากนั้นนำผลลัพธ์ที่ได้ทั้งสองค่ามาบวกกันต่อไป ซึ่งจะได้ผลลัพธ์อีกค่าหนึ่งซึ่งมีขนาด 10 บิตเป็นการเผื่อบิตตัวทดไว้เช่นกัน จากนั้นผลลัพธ์ที่ได้นำไปบวกกับข้อมูลจาก IA ซึ่งจากรูปที่ 3.6 จะเห็นว่า IA จะต่อผ่าน BUFFER 2 ตัว ด้วยกันผลลัพธ์ที่ได้จากการบวกจะกำหนดให้มีขนาด 11 บิตซึ่งเป็นการเผื่อบิตตัวทดไว้ ผลลัพธ์ที่ได้จากภาค ADDER จะส่งต่อไปยังภาค COMPARATOR ต่อไป

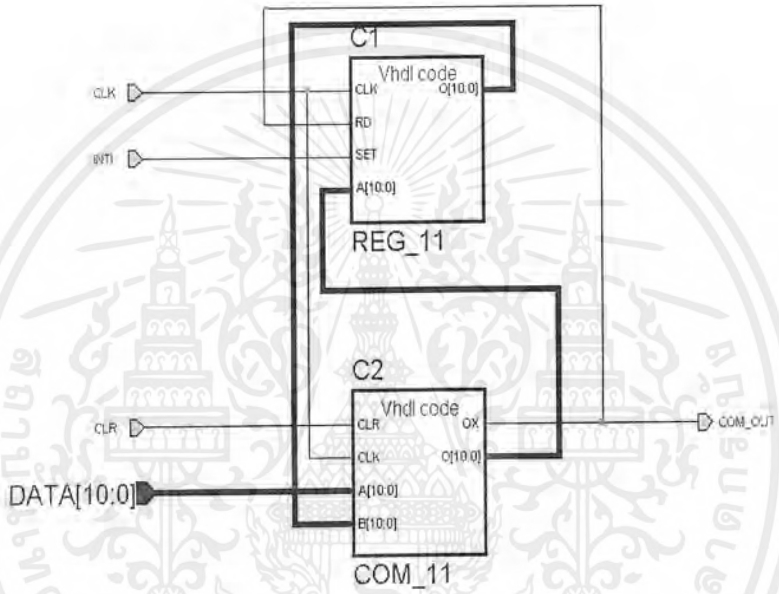


รูปที่ 3.6 วงจรภาค ADDER

3.2.5 ภาค COMPARATOR

เป็นภาคที่ทำหน้าที่เปรียบเทียบค่าที่ได้จากภาค ADDER แต่ละเวกเตอร์กับค่าที่เก็บไว้ในรีจิสเตอร์ การทำงานจะควบคุมด้วยภาค CONTROL ซึ่งจะกำหนดค่าแรก ให้กับรีจิสเตอร์ในภาค COMPARATOR ให้เป็นค่าสูงสุด เมื่อภาคเปรียบเทียบได้รับค่ามาจากวงจรบวกก็นำค่าที่ได้นี้ไป

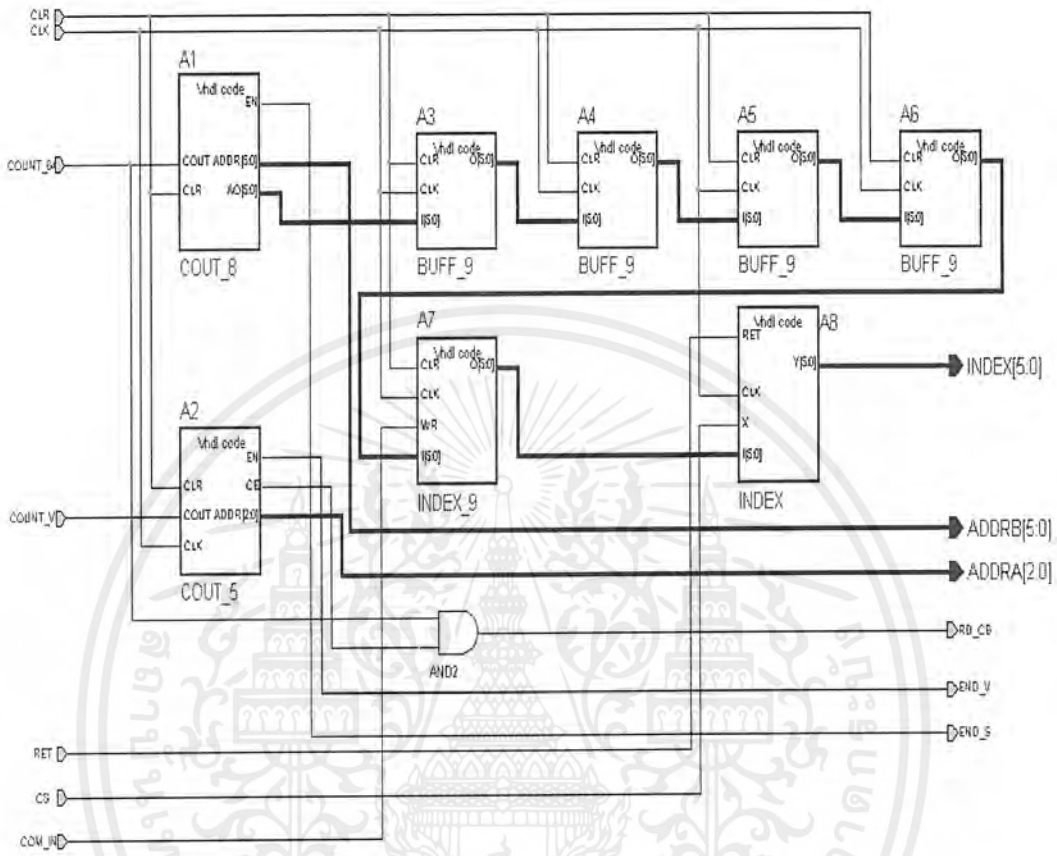
เปรียบเทียบกับค่าสูงสุดที่ถูกกำหนดไว้คือ '7FF' ถ้าค่าที่ได้จากวงจรบวกมีค่าน้อยกว่าค่าในวงจรเปรียบเทียบ วงจรเปรียบเทียบก็จะนำค่าจากวงจรบวกไปเก็บไว้แทนที่ค่าเดิม แต่ถ้าค่าจากวงจรบวกมีค่ามากกว่า วงจรเปรียบเทียบก็จะเก็บค่าเดิมไว้ ซึ่งในการทำงานทั้งหมดจะเปรียบเทียบกัน 64 ครั้งต่อสัญญาณอินพุตหนึ่งสัญญาณ ค่าที่ได้ก็จะมีค่าใกล้เคียงกับสัญญาณจากอินพุตมากที่สุด การทำงานจะเห็นได้จากรูปที่ 3.7



รูปที่ 3.7 วงจรภาค COMPARATOR

3.2.6 ภาค ADDRESS & INDEX

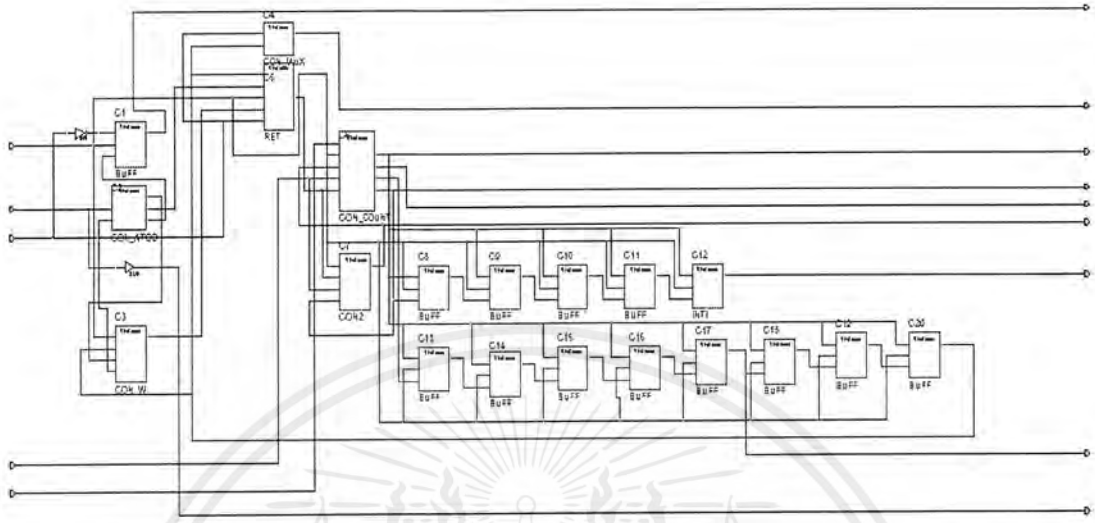
เป็นภาคที่ทำหน้าที่ส่งตำแหน่งของข้อมูลไปเปิดโค้ดบุ๊ค และส่งสัญญาณค่าดัชนีไปยังภาคต่อไป ซึ่งจะมีวงจรนับอยู่ 2 ชุด คือวงจรนับ 64 กับวงจรนับ 5 ซึ่งวงจรนับ 64 จะนับจำนวนเวกเตอร์ของข้อมูลในโค้ดบุ๊คโดยควบคุมจากภาค CONTROL เมื่อมีสัญญาณพัลส์เข้ามาลูกแรกวงจรนับก็จะเพิ่มค่าครั้งละ 1 และมี BUFFER 4 ตัวทำหน้าที่หน่วงเวลาเพื่อให้เวลาในการทำงานของวงจรอื่นๆ ทำงานกันได้ทันเวลาพอดี และมีสัญญาณมาจากภาค COMPARATOR ควบคุมการจดจำค่าดัชนี และจะส่งค่าดัชนีออกไปเมื่อขา CS อยู่ในสถานะเป็น High แสดงวงจรดังรูปที่ 3.8



รูปที่ 3.8 วงจรภาค ADDRESS & INDEX

3.2.7 ภาค CONTROL

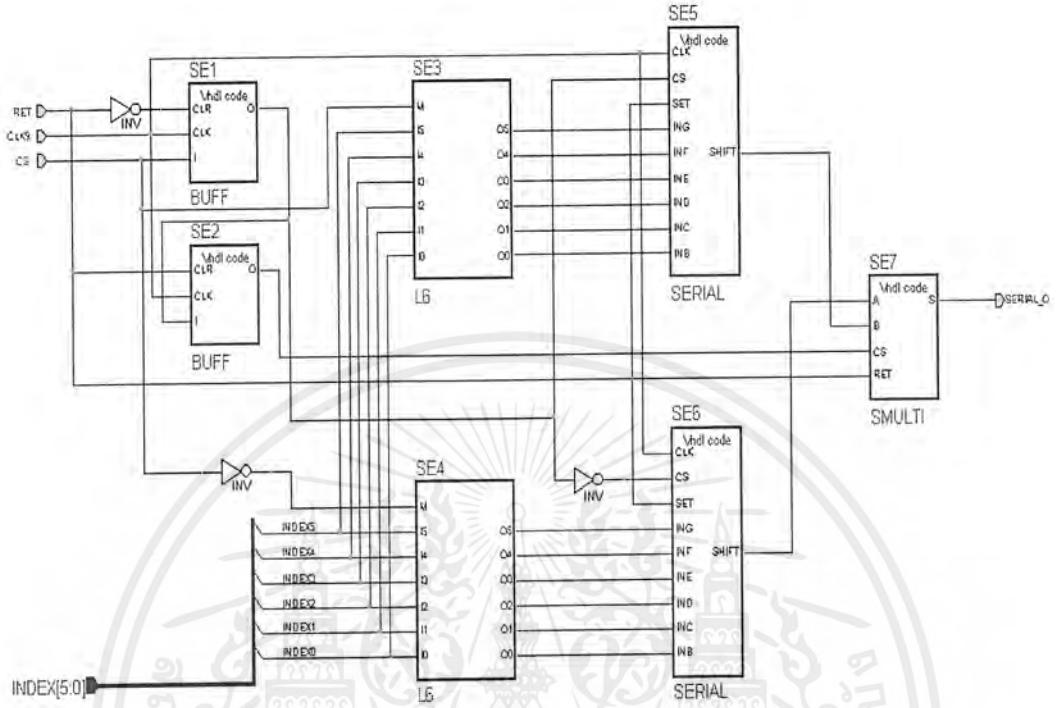
เป็นภาคที่ทำหน้าที่ควบคุมเวลาการทำงานของภาคอื่นๆ โดยสัญญาณที่ป้อนให้ภาค CONTROL จะมีสัญญาณนาฬิกา 2 สัญญาณคือ ความถี่ 5.5 เมกะเฮิร์ต และความถี่ 8 กิโลเฮิร์ต และรับสัญญาณจากภาค ADDRESS&INDEX ซึ่งเป็นสัญญาณสิ้นสุดการทำงานแต่ละเวกเตอร์ของสัญญาณอินพุต เมื่อสัญญาณ RET เป็น '1' ภาค CONTROL จะส่งสัญญาณไปควบคุมการทำงานของภาคต่างๆ และเมื่อภาค CONTROL ได้รับสัญญาณสิ้นสุดการทำงานจากขา END_S ภาค CONTROL จะส่งสัญญาณไปให้ภาค SUB_ABS และภาค ADDER หยุดการทำงาน และนับเวลาจากเริ่มการทำงานจนครบ 625 ไมโครวินาที ภาค CONTROL จะส่งสัญญาณให้ภาค SUB_ABS และภาค ADDER เริ่มการทำงานอีกครั้งหนึ่ง ซึ่งแสดงการทำงานดังรูปที่ 3.9



รูปที่ 3.9 วงจรภาค CONTROL

3.2.8 ภาค SERIAL_OUT

เป็นภาคที่ส่งตำแหน่งของข้อมูลไปยังเครื่องรับ มีขาที่ใช้งานคือ ขา CLK 9 กิโลเฮิร์ต ซึ่งจะจ่ายให้กับอุปกรณ์ในภาคนี้ ส่วนขา CS จะใช้ควบคุมการทำงานของแต่ละบล็อกโดยส่งมาจากภาค CONTROL ส่วน INDEX จะถูกส่งมารออยู่ที่บล็อก SE3 และ SE4 ถ้าขา CS มีค่าเป็น '1' INDEX จะถูกนำเข้าไปเก็บใน SE3 และเมื่อ CLK เข้ามาอีกลูกหนึ่ง SERIAL SE5 จะนำข้อมูลเข้าไปเพื่อส่งไปยังเครื่องรับต่อไป แต่ถ้า CS มีค่าเป็น '0' ก็จะสลับการทำงานเป็น SE4 รับข้อมูลเข้าไป และเมื่อสัญญาณ CLK เข้ามาอีกลูกหนึ่ง SERIAL SE6 ก็จะรับข้อมูลเพื่อส่งไปยังเครื่องรับต่อไป การทำงานจะสลับกันไปตามสัญญาณจาก CLK ที่เข้ามา ซึ่งแสดงได้ดังรูปที่ 3.10

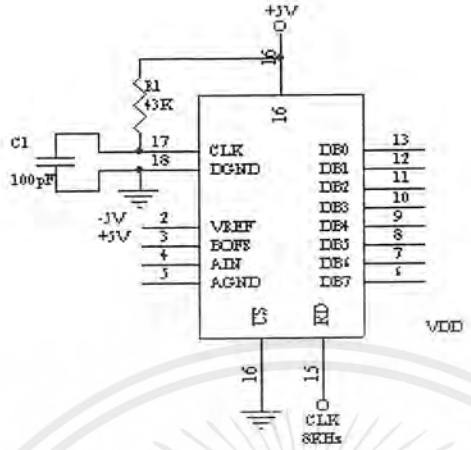


รูปที่ 3.10 วงจรภาค SERIAL_OUT

3.3 การออกแบบและการสร้างวงจรทดลอง

3.3.1 วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล

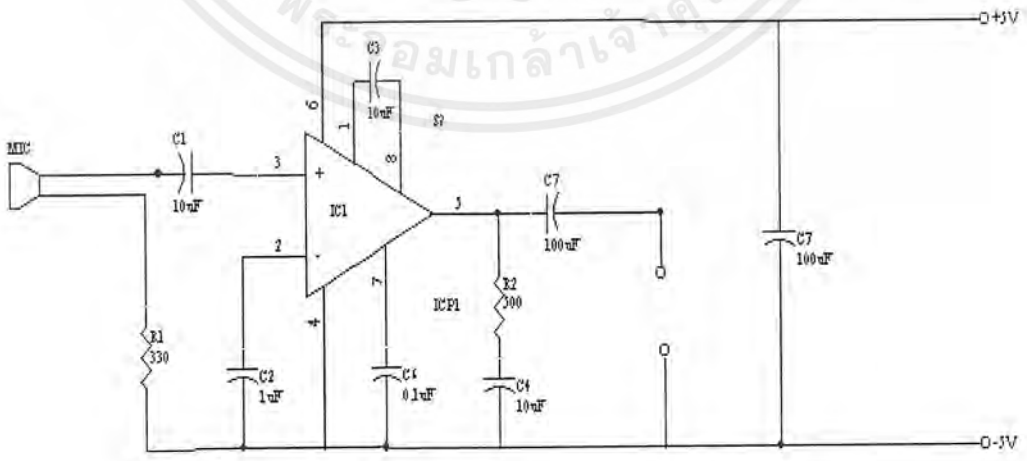
วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล ในโครงงานนี้จะใช้ไอซีเบอร์ ADC908 ซึ่งเป็นไอซีสำหรับแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอลแบบ Successive Approximation และให้ผลการแปลงเป็นสัญญาณดิจิตอล 8 บิต โดยการต่อวงจรใช้งานเป็นดังรูปที่ 3.3 จะเห็นได้ว่าวงจรนั้นต่อ R1 และ C1 ไว้สำหรับสร้างสัญญาณนาฬิกาภายในตัวไอซี ซึ่งทำให้มีช่วงเวลาในการแปลงสัญญาณ (Conversion time) ได้ต่ำสุด 6 ไมโครวินาที ส่วนขา RD ใช้สำหรับการต่อสัญญาณนาฬิกาจากภายนอก ซึ่งสามารถใช้ความถี่ได้สูงถึง 1.35 MHz



รูปที่ 3.11 วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล

3.3.2 วงจรปรีไมโครโฟน

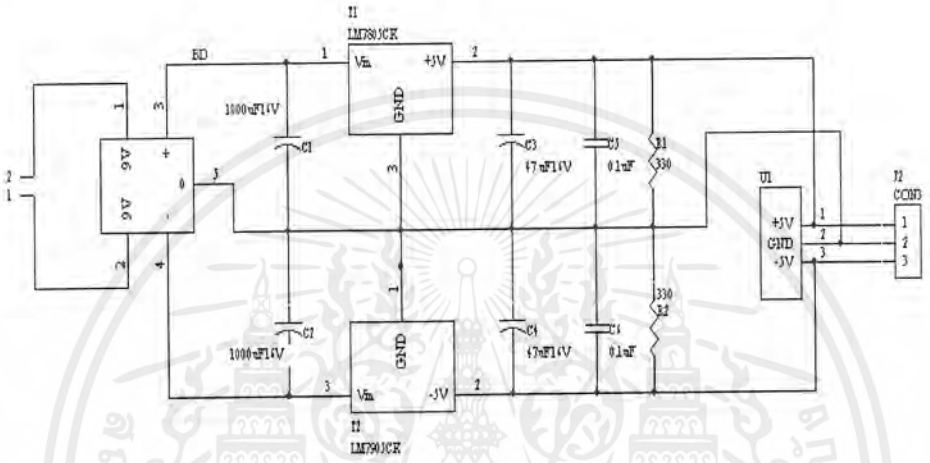
วงจรปรีไมโครโฟน ทำหน้าที่ขยายสัญญาณเสียงที่ได้จากไมโครโฟนชนิดคอนเดนเซอร์ให้มีความแรงของสัญญาณสูงขึ้น เพื่อส่งสัญญาณเสียงนั้นไปยังวงจรประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค ดังแสดงในรูปที่ 3.12 โดยใช้ไอซีเบอร์ LM386 ซึ่งเป็นไอซีขยายสัญญาณเสียง วงจรมีอินพุตอิมพีแดนซ์เท่ากับ 3.587 กิโลโอห์ม เอาต์พุตอิมพีแดนซ์เท่ากับ 2 กิโลโอห์ม อัตราขยายแรงดันประมาณ 66 เท่า และอัตราขยายกระแสประมาณ 127 เท่า



รูปที่ 3.12 วงจรปรีไมโครโฟน

3.3.3 วงจรแหล่งจ่ายไฟ

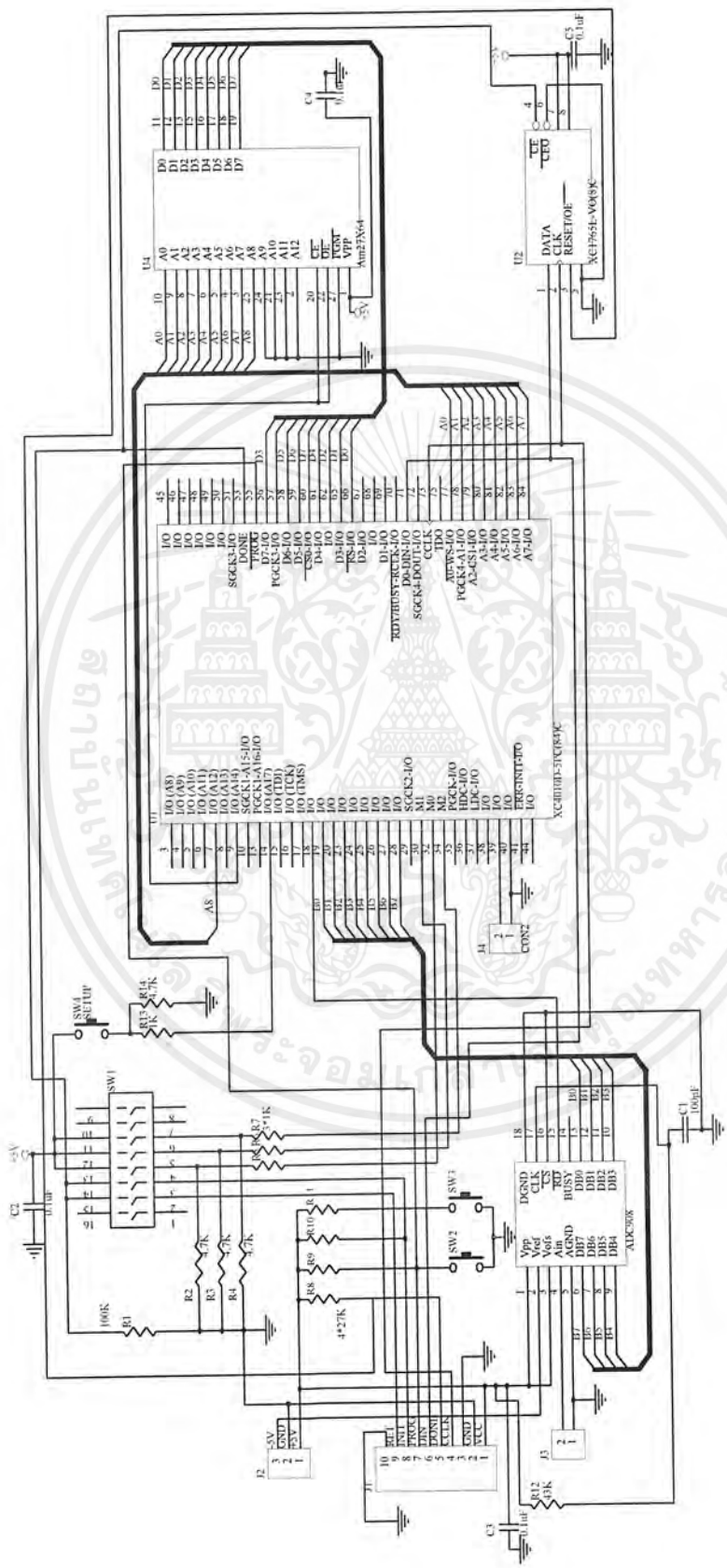
เนื่องจากวงจรทั้งหมดที่ออกแบบมา ต้องการใช้แรงดันเลี้ยงวงจร 5 โวลต์ ดังนั้น เราจึงต้องใช้วงจรแหล่งจ่ายไฟที่ให้แรงดันเป็น 5 โวลต์ ซึ่งวงจรแหล่งจ่ายไฟแสดงดังรูปที่ 3.13 ทำงานโดยใช้วงจรไอซีเรกกูเลเตอร์ เบอร์ 7805 และ 7905 เป็นตัวให้แรงดันที่เอาต์พุตของวงจร



รูปที่ 3.13 วงจรแหล่งจ่ายไฟ

3.3.4 วงจรประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

การออกแบบวงจรประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊คใช้อุปกรณ์ FPGAs เบอร์ XC 4010E ซึ่งเป็นส่วนสำคัญของวงจรทดสอบนี้ วงจรประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊คจะประกอบไปด้วย วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอลซึ่งใช้ไอซีเบอร์ ADC908 ข้อมูลบรรจุอยู่ในโค้ดบุ๊ค (Codebook) เป็นอีพรอม (EPROM) เบอร์ 27C256 โปรแกรมที่ทำการกำหนดโครงสร้างให้กับอุปกรณ์ FPGAs จะถูกโปรแกรมลงในพรอม (PROM) เบอร์ 17C256DPC ใช้ตัวกำเนิดความถี่ในการกำหนดเวลาทำงานให้กับวงจรอยู่ภายใน FPGAs และสวิตช์สำหรับเลือกโหมดในการทำงานของอุปกรณ์ FPGAs โดยที่การเชื่อมต่อของวงจรต่างๆ กับอุปกรณ์ FPGAs นั้นจะทำการต่อร่วมโดยคำนึงถึงวงจรภายใน FPGAs ที่ได้ออกแบบไว้หลังจากที่ได้ออกแบบวงจรภายในแล้วส่วนของอินพุต และส่วนของเอาต์พุตจะต้องตรงกับวงจรที่ต่อร่วมด้วย ซึ่งแสดงวงจรดังรูปที่ 3.14



รูปที่ 3.14 วงจรประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

บทที่ 4
การทดลองและผลการทดลอง

4.1 กล่าวนำ

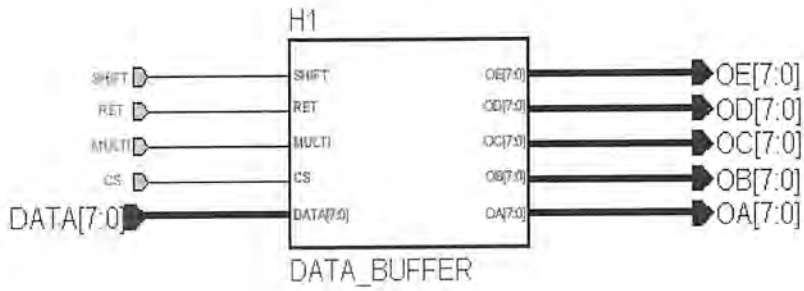
ตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค โดยใช้อุปกรณ์ FPGAs ได้สร้างขึ้นโดยอาศัยหลักการควอนไตซ์เวกเตอร์ เพื่อใช้ในการเข้ารหัสสัญญาณเสียง จากบทที่ 1-3 ได้กล่าวถึงแนวความคิด ทฤษฎี การออกแบบและการสร้างตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค โดยใช้ภาษา VHDL และโปรแกรม Schematic ดังรูปวงจรในบทที่ 3 นั้น ส่วนในบทนี้จะกล่าวถึงการทดสอบการทำงานและผลการทดลองของตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค โดยได้แบ่งการลักษณะผลการทดลองออกเป็น 2 ส่วนคือ ในส่วนของการเลียนแบบการทำงาน (Simulate) สัญญาณต่างๆ โดยใช้โปรแกรม Logic Simulater ของ Xilinx Foundation F1.5 และส่วนของการนำตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊คมาประยุกต์ใช้เป็นตัวเข้ารหัสสัญญาณเสียง

4.2 การ Simulate ตัวประมวลผลค้นหาข้อมูลในโค้ดบุ๊ค

จากการออกแบบตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค ได้แบ่งการทำงานออกเป็นบล็อกต่างๆ ดังนี้ คือ DATA_BUFFER, CB_BUFFER, SUB_ABS, ADDER, COMPARATOR, CONTROL, ADDRESS&INDEX และ SERIAL_OUT ซึ่งสัญญาณแต่ละตัวก็จะมีหน้าที่การทำงานและผลการทำงานที่แตกต่างกัน ในหัวข้อนี้ จะยกตัวอย่างสัญญาณผลการเลียนแบบการทำงานของอุปกรณ์ที่กล่าวมาแล้วข้างต้นเพียงบางตัวเท่านั้น ส่วนผลการทดลองที่ไม่ได้อธิบายไว้ในหัวข้อนี้ ก็สามารถดูผลการ Simulate ได้จากภาคผนวก ก

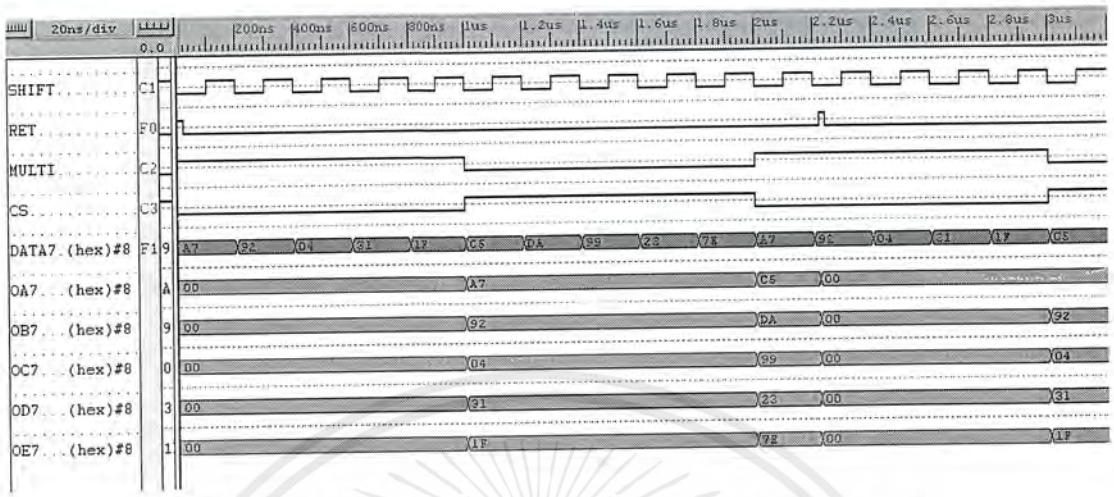
4.2.1 ผลการ Simulate สัญญาณของภาค DATA_BUFFER

จากการทดลองเลียนแบบการทำงานของบล็อก DATA_BUFFER ในรูปที่ 4.1 โดยใช้โปรแกรม Logic Simulater ของ Xilinx Foundation F1.5 ได้ผลการทดลองดังรูปที่ 4.2



รูปที่ 4.1 บล็อกไดอะแกรมของ DATA_BUFFER

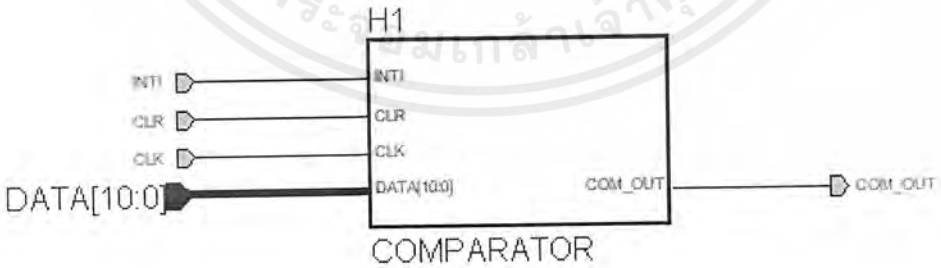
จากผลการทดลองดังรูปที่ 4.2 จะเห็นว่าเมื่อมีสัญญาณข้อมูลขนาด 8 บิต (เท่ากับ 1 อีลีเมนต์) เข้ามาที่ขา DATA[7:0] ข้อมูลก็จะถูกเลื่อน (shift) ไปเก็บไว้ยังตำแหน่งต่างๆ ของ buffer ชุดแรก โดยข้อมูลที่เข้ามาเป็นอีลีเมนต์แรก จะถูกนำไปเก็บไว้ที่รีจิสเตอร์ตำแหน่งล่างสุด คือ OA7 ส่วนข้อมูลในอีลีเมนต์ถัดไปก็จะถูกนำไปเก็บที่ตำแหน่ง OB7, OC7, OD7 และ OE7 ซึ่งเป็นรีจิสเตอร์ตำแหน่งสูงสุดตามลำดับ จนเมื่อมีข้อมูลเก็บใน buffer ชุดแรกจนครบ 5 อีลีเมนต์แล้ว สัญญาณ MULTI ก็จะเปลี่ยนสถานะจาก high เป็น low เพื่อทำการเลือกให้ข้อมูลที่อยู่ใน buffer ชุดแรกไปแสดงผลยังเอาต์พุต ในขณะเดียวกัน สัญญาณ CS ก็จะเปลี่ยนสถานะจาก low เป็น high เพื่อเลือกให้ buffer อีกชุดหนึ่งรับข้อมูลเวกเตอร์ถัดไปเข้ามาเก็บใน buffer เช่นเดียวกับ buffer ชุดแรก จนครบ 5 เวกเตอร์อีกครั้งหนึ่งแล้วจึงนำข้อมูลใน buffer นั้น ส่งผ่านไปยังเอาต์พุตพร้อมๆ กัน ซึ่งจะทำงานในลักษณะเช่นนี้เรื่อยๆ ไป แต่ถ้าสัญญาณ RET เป็น high เมื่อใด ก็จะทำการเคลียร์ค่าข้อมูลที่มีอยู่ใน buffer นั้น ให้เป็นศูนย์ จนกระทั่งสัญญาณ RET เปลี่ยนสถานะเป็น low ก็จะเริ่มทำการรับข้อมูลเข้ามาเก็บใน buffer อีกครั้งในคล็อกการทำงานถัดไป



รูปที่ 4.2 ผลการเลียนแบบการทำงานของภาค DATA_BUFFER

4.2.2 ผลการ Simulate สัญญาณของภาค COMPARATOR

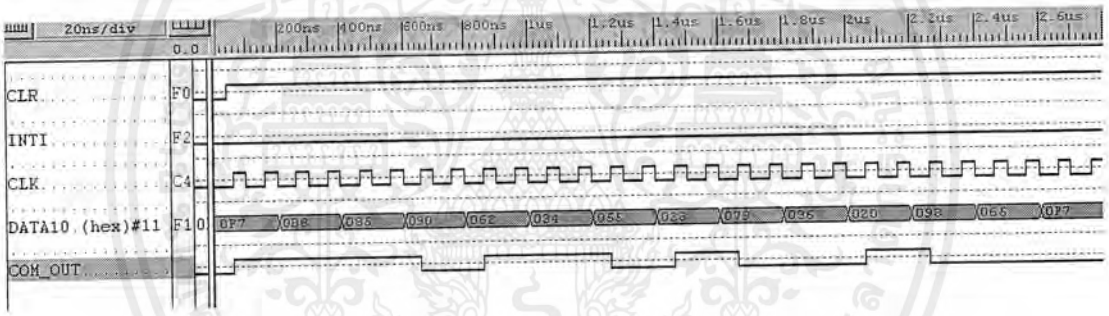
ในรูปที่ 4.3 เป็นบล็อกไดอะแกรมที่ใช้ในการทดลองเลียนแบบการทำงานของสัญญาณภาค COMPARATOR โดยมีขา INTI, CLR, CLK และ DATA[10:0] เป็นสัญญาณอินพุต และมีขา COM_OUT เป็นสัญญาณเอาต์พุต ก่อนการทดลองเลียนแบบการทำงาน จะต้องต่อขาสัญญาณภายนอกให้กับภาค COMPARATOR แล้วจึงเลียนแบบการทำงานด้วยโปรแกรม Logic Simulater ซึ่งแสดงผลการทดลองเลียนแบบสัญญาณการทำงานได้ดังรูปที่ 4.4



รูปที่ 4.3 บล็อกไดอะแกรมของ COMPARATOR

เมื่อเริ่มต้นการทำงาน จะกำหนดให้สัญญาณ INTI เป็น high เพื่อกำหนดค่าเริ่มต้นภายในรีจิสเตอร์ให้เป็นค่าสูงสุด ซึ่งมีค่าเท่ากับ '7FF' เมื่อมีสัญญาณข้อมูลขนาด 10 บิต เข้ามาที่ขา

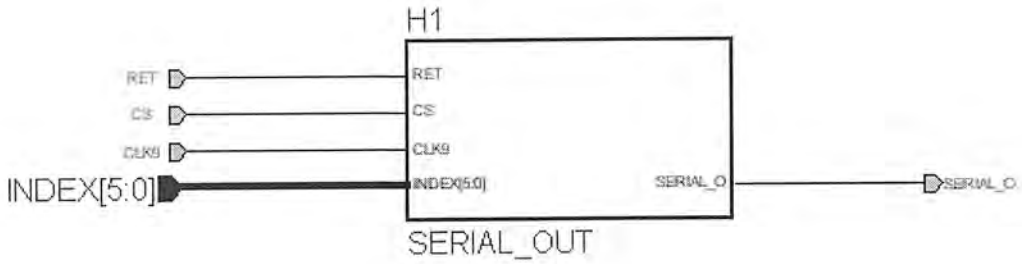
DATA[10:0] ตัว COMPARATOR ก็จะนำสัญญาณข้อมูลที่เข้ามาไปเปรียบเทียบกับสัญญาณข้อมูลที่เก็บอยู่ภายในรีจิสเตอร์ โดยถ้าค่าข้อมูลที่รับเข้ามานั้น มีค่าน้อยกว่าหรือเท่ากับค่าที่เก็บอยู่เดิมในรีจิสเตอร์ตัว COMPARATOR ก็จะทำการเก็บค่าข้อมูลตัวนั้นไว้ในรีจิสเตอร์แทนค่าเดิมที่มีอยู่ เพื่อใช้สำหรับเปรียบเทียบกับสัญญาณข้อมูลที่จะเข้ามาเป็นอันดับต่อไป พร้อมกันนี้ ก็จะส่งสัญญาณ high ออกทางขาเอาต์พุต COM_OUT เพื่อบอกกับภาค ADDRESS & INDEX ว่าข้อมูลตำแหน่งนั้นมีค่าใกล้เคียงกับสัญญาณข้อมูลอินพุตที่รับเข้ามาจากอุปกรณ์แปลงสัญญาณแอนะลอกเป็นดิจิตอลพร้อมทั้งให้เก็บตำแหน่งนั้นไว้สำหรับการอ่านค่าดัชนีในโค้ดบุ๊คต่อไป แต่ถ้าข้อมูลที่เข้ามานั้นมีค่ามากกว่า ค่าในรีจิสเตอร์ก็จะไม่มีการเปลี่ยนแปลงจะเก็บค่าตำแหน่งเดิมไว้ แล้วส่งสัญญาณ low ออกทางขาเอาต์พุต COM_OUT บอกกับภาค ADDRESS & INDEX ว่าค่าตำแหน่งที่รับเข้ามานั้นไม่ได้ใกล้เคียงกับสัญญาณข้อมูลอินพุตที่รับมาจากอุปกรณ์แปลงสัญญาณแอนะลอกเป็นดิจิตอล



รูปที่ 4.4 ผลการเลียนแบบการทำงานของภาค COMPARATOR

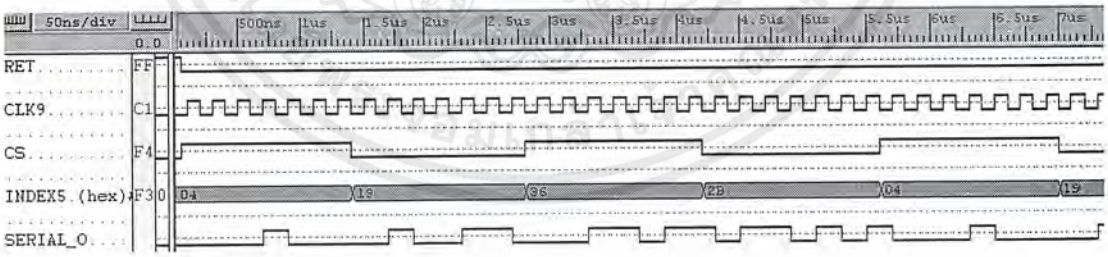
4.2.3 ผลการ Simulate สัญญาณของภาค SERIAL_OUT

ภาค SERIAL_OUT เป็นภาคที่ทำหน้าที่ส่งข้อมูลที่เป็น INDEX ออกไปยังภาครับในรูปแบบของสัญญาณแอนะลอก โดยส่งเรียงกันไปทีละบิต ลักษณะการต่อสัญญาณเพื่อทำการทดลองเลียนแบบการทำงานก็มีลักษณะเช่นเดียวกับ DATA_BUFFER และ COMPARATOR โดยการต่อสายสัญญาณภายนอกให้กับภาค SERIAL_OUT มีลักษณะดังรูปที่ 4.5



รูปที่ 4.5 บล็อกไดอะแกรมของ SERIAL_OUT

จากรูปที่ 4.6 เมื่อเริ่มต้นการทดลอง จะต้องให้สัญญาณ RET เป็น high เพื่อทำการเคลียร์ข้อมูลที่อยู่ใน buffer ของ SERIAL_OUT ก่อนการทำงานของภาค SERIAL_OUT คือจะต้องส่งข้อมูลขนาด 6 บิต ในลักษณะอนุกรมออกทางด้านเอาต์พุตโดยมีบิตเรตในการส่งข้อมูล 9,600 บิตต่อวินาที เมื่อมีสัญญาณอินพุตเข้ามาที่ขา INDEX[5:0] ชิฟรีจิสเตอร์ภายในของภาค SERIAL_OUT ซึ่งมี 2 ตัว จะสลับกันส่งสัญญาณออกมายังเอาต์พุต ในขณะที่สัญญาณ CS มีการเปลี่ยนแปลงสภาวะจาก high เป็น low หรือเปลี่ยนจาก low เป็น high สังเกตได้จากรูปที่ 4.6 โดยที่ภาค SERIAL_OUT จะส่งข้อมูลของบิตนัยสำคัญต่ำสุดออกไปก่อน แล้วจึงตามด้วยบิตถัดไปจนครบทั้ง 6 บิต



รูปที่ 4. 6 ผลการเลียนแบบการทำงานของอุปกรณ์ SERIAL_OUT

4.3 การทดลองการใช้ตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

4.3.1 ขั้นตอนการทดสอบตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

จากที่ได้ออกแบบวงจรประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค ซึ่งประกอบด้วยวงจรต่างๆ ตามแผนผังการทำงานแล้วนั้น ขั้นตอนต่อไปเป็นการทดสอบการทำงานของวงจรที่ได้ออกแบบไว้ ซึ่งมีขั้นตอนการทดสอบการทำงานดังนี้

4.3.2 ขั้นตอนการสร้างไฟล์องค์ประกอบ

หลังจากทำขั้นตอนการทดสอบ โดยโปรแกรมเสร็จเรียบร้อยแล้ว ต่อไปเป็นการตรวจสอบวงจรโดยการดาวน์โหลดข้อมูลลงบนอุปกรณ์ FPGAs และขั้นตอนการทดสอบนี้จะต้องใช้ดาวน์โหลดเคเบิ้ล (Download Cable) เป็นตัวส่งองค์ประกอบสำหรับควบคุมให้ FPGAs เบอร์ XC4010E ทำงานโดยองค์ประกอบที่ใช้ควบคุม FPGAs จะถูกเก็บเป็นแฟ้ม .BIT การกำหนดองค์ประกอบให้กับ XDM และ XMAKE ใน XDM (XACT Design Manager) นั้น XMAKE จะทำการแปลงโปรแกรมส่วนต่างๆ โดยอัตโนมัติโดยใช้กระบวนการของ Xilinx (Xilinx design flow) ในการทำกระบวนการออกแบบที่ซับซ้อนเช่นในการทำโปรแกรมหนึ่งต้องป้อนชื่อในชั้นสูงสุดของ Schemetics ที่สร้างไว้แล้ว XMAKE จะทำการค้นและกระทำทุกๆ ชั้นที่ต่ำลงมาของการออกแบบซึ่งจะได้แฟ้ม LCA ที่ทำการจัดวางและหาเส้นทางแล้วให้แฟ้ม .BIT ออกมาซึ่งพร้อมที่จะดาวน์โหลดให้ FPGAs ทำงานได้

ตารางที่ 4.1 การตั้งสวิตช์ต่าง ๆ ในแผงวงจรของ FPGAs

Switch(DIP.SW)	Lable	Setting
1	PWR	X
2	MPE	OFF
3	SPE	OFF
4	M0	ON
5	M1	ON
6	M2	ON
7	RST	X
8	INIT	OFF

4.4 การโปรแกรม Serial PROM

การเขียนโปรแกรม Serial PROM จะเป็นการกำหนดโครงสร้างภายในให้กับอุปกรณ์ FPGAs โดยไม่ต้องใช้วิธีการดาวน์โหลดจากโปรแกรมทุกครั้งที่ต้องการใช้งาน ซึ่งการดาวน์โหลดจากโปรแกรมนั้น จะมีข้อเสียคือ หลังจากที่ทำการดาวน์โหลดแล้ววงจรไม่สามารถที่จะขาดไฟเลี้ยงวงจรได้เพราะโครงสร้างภายในของ FPGAs นั้นจะหายไป ซึ่งจำเป็นที่จะต้องทำการดาวน์โหลดใหม่ ดังนั้น การใช้อุปกรณ์ Serial PROM มาเก็บโครงสร้างให้กับอุปกรณ์ FPGAs นั้นเป็นวิธีการที่ดีที่สุด ซึ่งในการทดสอบได้ใช้ Serial PROM เบอร์ 17256

4.5 ขั้นตอนการทดลอง

เมื่อได้กำหนดโครงสร้างของวงจรประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊คลงใน Serial PROM เรียบร้อยแล้ว จากนั้นก็ทำการทดลองตามขั้นตอนต่อไปนี้

1) นำ Serial PROM ที่กำหนดโครงสร้างเรียบร้อยแล้ว ประกอบเข้าไปในวงจรประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค
2) ตั้งสวิตช์ที่แผงวงจรประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค ตามตารางที่ 4.2 และแผงวงจรถอดรหัสตามตารางที่ 4.3 ซึ่งแผงวงจรถอดรหัสนำมาใช้ประกอบการทดลอง โดยนำของเดิมที่มีอยู่แล้วมาปรับปรุงให้สามารถใช้งานได้

ตารางที่ 4.2 การตั้งสวิตช์ต่าง ๆ ในวงจรประมวลผลการค้นหาข้อมูล

Switch (DIP.SW)	Lable	Setting
1	PWR	X
2	MPE	ON
3	SPE	ON
4	M0	OFF
5	M1	OFF
6	M2	OFF
7	RST	X
8	INIT	OFF

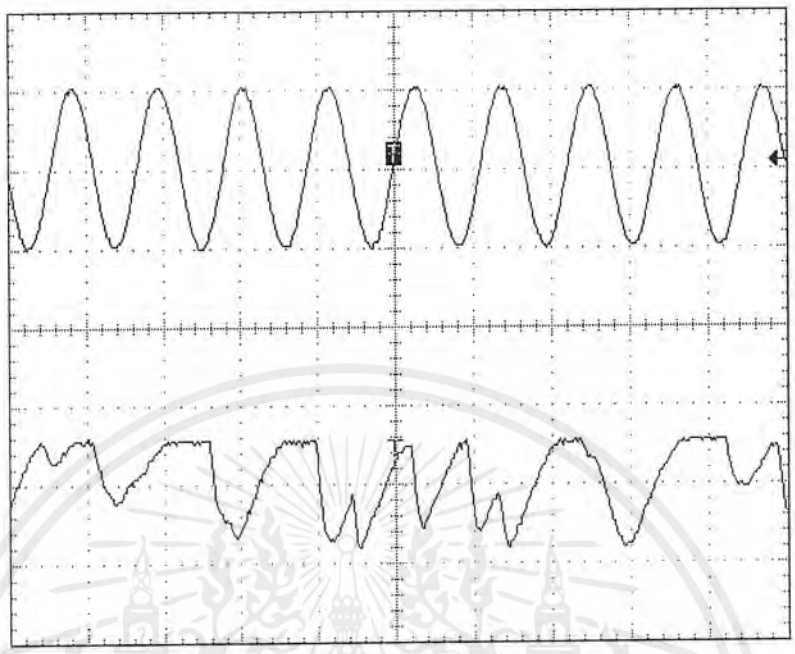
3) จ่ายไฟเลี้ยงให้กับวงจรประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค วงจรถอดรหัสเสียง วงจรปรีไมโครโฟน
4) นำออสซิลโลสโคป (Oscilloscope) วัดสัญญาณที่ต้องการทดสอบ โดยทำการวัดที่ส่วนอินพุตของวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล ทดสอบสัญญาณโดยทำการส่งสัญญาณผ่านทางอินพุต แล้วทำการวัดรูปสัญญาณ
5) ทำการทดลองขั้นตอนที่ 4 ซ้ำอีกครั้งแต่ทำการเปลี่ยนสัญญาณทดสอบให้แตกต่างกันในการทดลองแต่ละครั้ง

ตารางที่ 4.3 การตั้งสวิตช์ต่าง ๆ ในแผงวงจรถอดรหัสเสียง

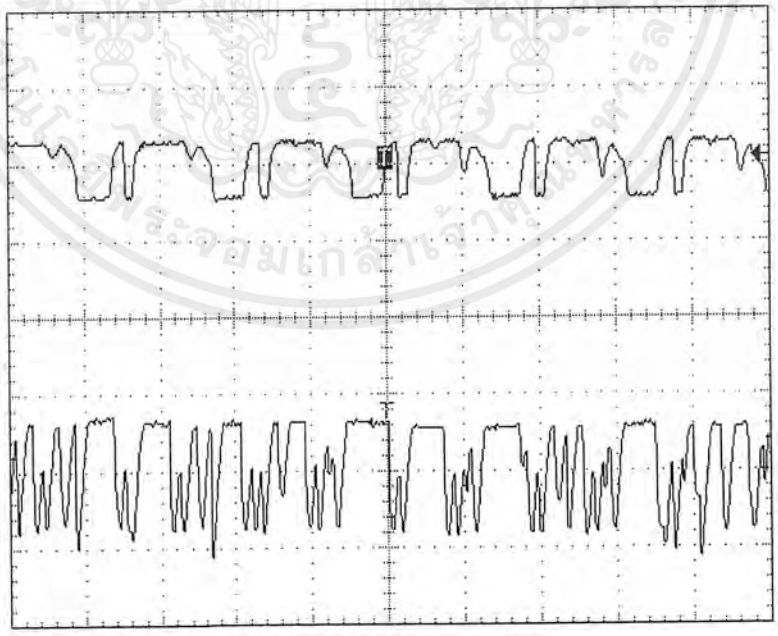
Switch (DIP.SW)	Lable	Setting
1	INP	X
2	MPE	ON
3	SPE	ON
4	M0	OFF
5	M1	OFF
6	M2	OFF
7	MCLK	X
8	DOUT	OFF

4.6 ผลการทดลอง

จากการทดลองวัดสัญญาณเอาต์พุตจะมีสัญญาณรบกวนค่อนข้างมาก แต่เมื่อส่งสัญญาณเสียงผ่านไมโครโฟน และทดสอบฟังสัญญาณเสียงทางเอาต์พุตของภาครับ ปรากฏว่าสามารถรับฟังสัญญาณเสียงได้ชัดเจนในระดับหนึ่ง แต่ก็เกิดสัญญาณรบกวนขึ้นขณะทดสอบสัญญาณเสียง ซึ่งผลการทดสอบสัญญาณจะดูได้ดังรูปที่ 4.7 แสดงสัญญาณเอาต์พุตขณะป้อนสัญญาณไซน์โดยสัญญาณเอาต์พุตที่ได้มีความผิดเพี้ยนจากสัญญาณอินพุตไปบ้าง ส่วนรูปที่ 4.8 แสดงสัญญาณเอาต์พุตโดยป้อนสัญญาณเสียงพูดคำว่า “ฮัลโหล...โหล...โหล...” เข้าไปทดสอบ ซึ่งสัญญาณเอาต์พุตที่ได้ก็จะมีสัญญาณออสซิลเลตรบกวนด้วย



รูปที่ 4.7 สัญญาณอินพุตและสัญญาณเอาต์พุต



รูปที่ 4.8 สัญญาณเอาต์พุตขณะป้อนสัญญาณเสียง

บทที่ 5
บทสรุป ปัญหา แนวทางการแก้ไขและพัฒนา

5.1 บทสรุป

จากที่ได้ทำการสร้างตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊คขึ้นมานั้น ผลที่ได้เป็นไปตามวัตถุประสงค์ที่วางไว้คือ เข้าใจถึงหลักการทำงานของตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค, การใช้ภาษา VHDL และอุปกรณ์ FPGAs รวมถึงการออกแบบตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค ซึ่งผลที่ได้สามารถส่งข้อมูลในอัตราบิตเรต 9,600 บิตต่อวินาที สามารถเข้ารหัสและถอดรหัสสัญญาณเสียงได้ และสามารถค้นหาข้อมูลในโค้ดบุ๊คได้ซึ่งเป็นไปตามทฤษฎีและวัตถุประสงค์ที่วางไว้

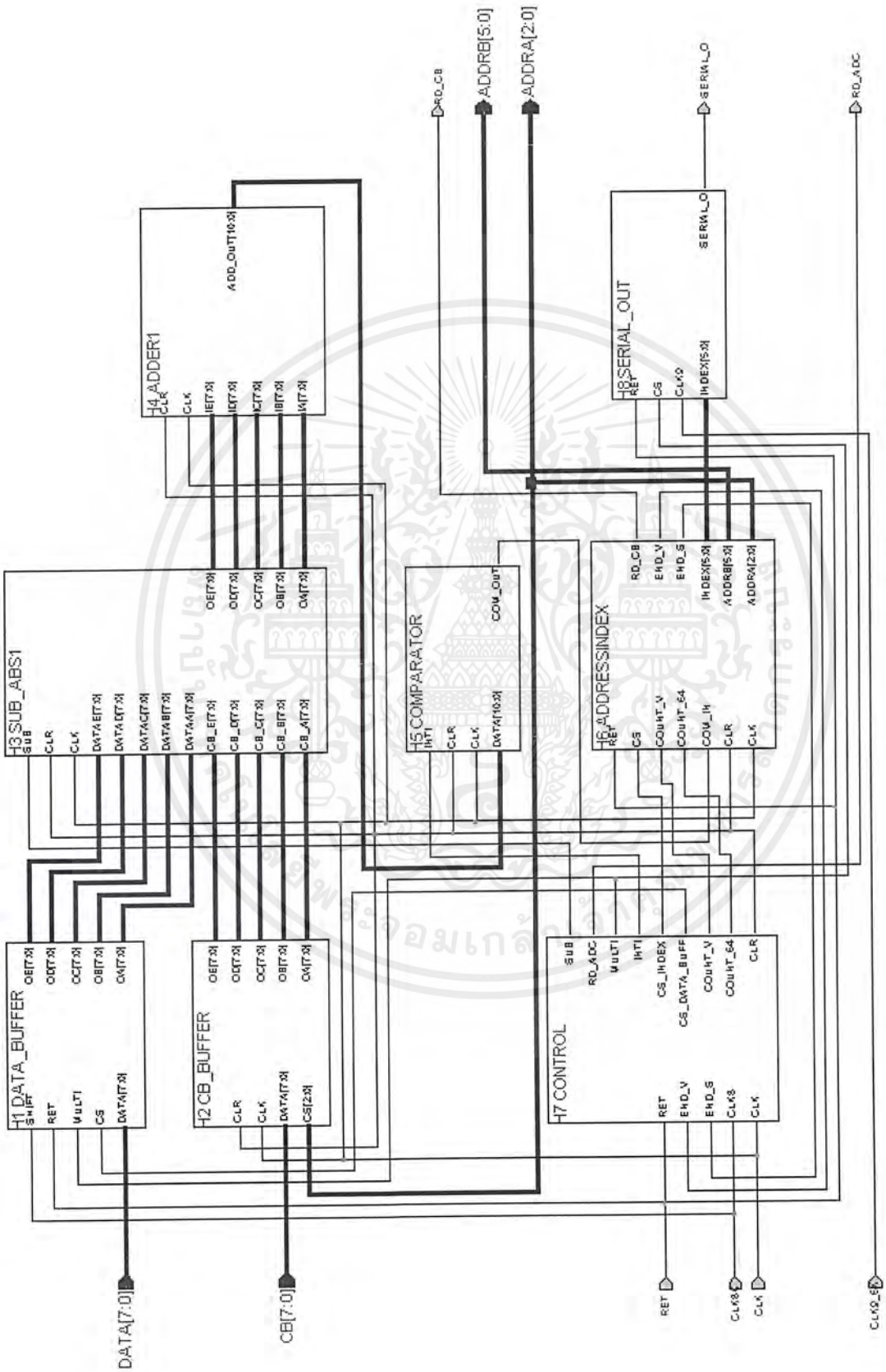
5.2 ปัญหาและแนวทางการแก้ไข

จากที่ได้ทำโครงงานชิ้นนี้ก็ได้เกิดปัญหาในขั้นต้นคือ การศึกษาการใช้งานอุปกรณ์ FPGAs และการใช้ภาษา VHDL ซึ่งใช้เวลาในการศึกษานานพอสมควรเนื่องจากไม่มีความรู้พื้นฐานมาก่อน และโครงงานชิ้นนี้ก็ได้นำเอาชิ้นงานของรุ่นพี่ที่ไม่ใช้แล้วมาประยุกต์ใช้งาน ซึ่งการออกแบบวงจรไม่เป็นไปตามหลักวิชาการ ทำให้ผลการทดลองที่ได้ผิดพลาด ซึ่งต้องทำการแก้ไขและปรับปรุงใหม่บางส่วน จากการทดลองพบว่าเกิดสัญญาณออสซิลเลตขึ้นเมื่อทำการเข้ารหัสสัญญาณเสียงได้ทำการแก้ไขโดยใช้สายสัญญาณที่มีการชีลด์ แต่สัญญาณออสซิลเลตก็ยังมีอยู่จึงตรวจสอบวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล ซึ่งจะเกิดปัญหาก็คือแรงดันออฟเซตของวงจรไม่เสถียรภาพ จากนั้นก็ทำการแก้ไขจุดต่อสัญญาณกราวด์ให้ใหญ่ขึ้น กำหนดตัวสร้างสัญญาณความถี่ขึ้นภายในอุปกรณ์ FPGAs เพื่อป้องกันการเกิดสัญญาณรบกวน และทำการต่อตัวเก็บประจุที่ขาแหล่งจ่ายไฟของอุปกรณ์ FPGAs ทุกขาเพื่อป้องกันการเกิดกระแสกระชาก ซึ่งอาจทำให้วงจรเสียหายได้ เมื่อทำการทดสอบปรากฏว่าสัญญาณออสซิลเลตก็ยังมีอยู่ ก็ได้ทำการแก้ไขวงจรปรีไมโครโฟนเพิ่ม โดยทำการเปลี่ยนทรานซิสเตอร์เป็นไอซีเบอร์ LM386 ซึ่งทำให้สัญญาณออสซิลเลตลดลงบ้าง พอฟังสัญญาณเสียงได้ในระดับหนึ่ง

5.3 แนวทางในการพัฒนา

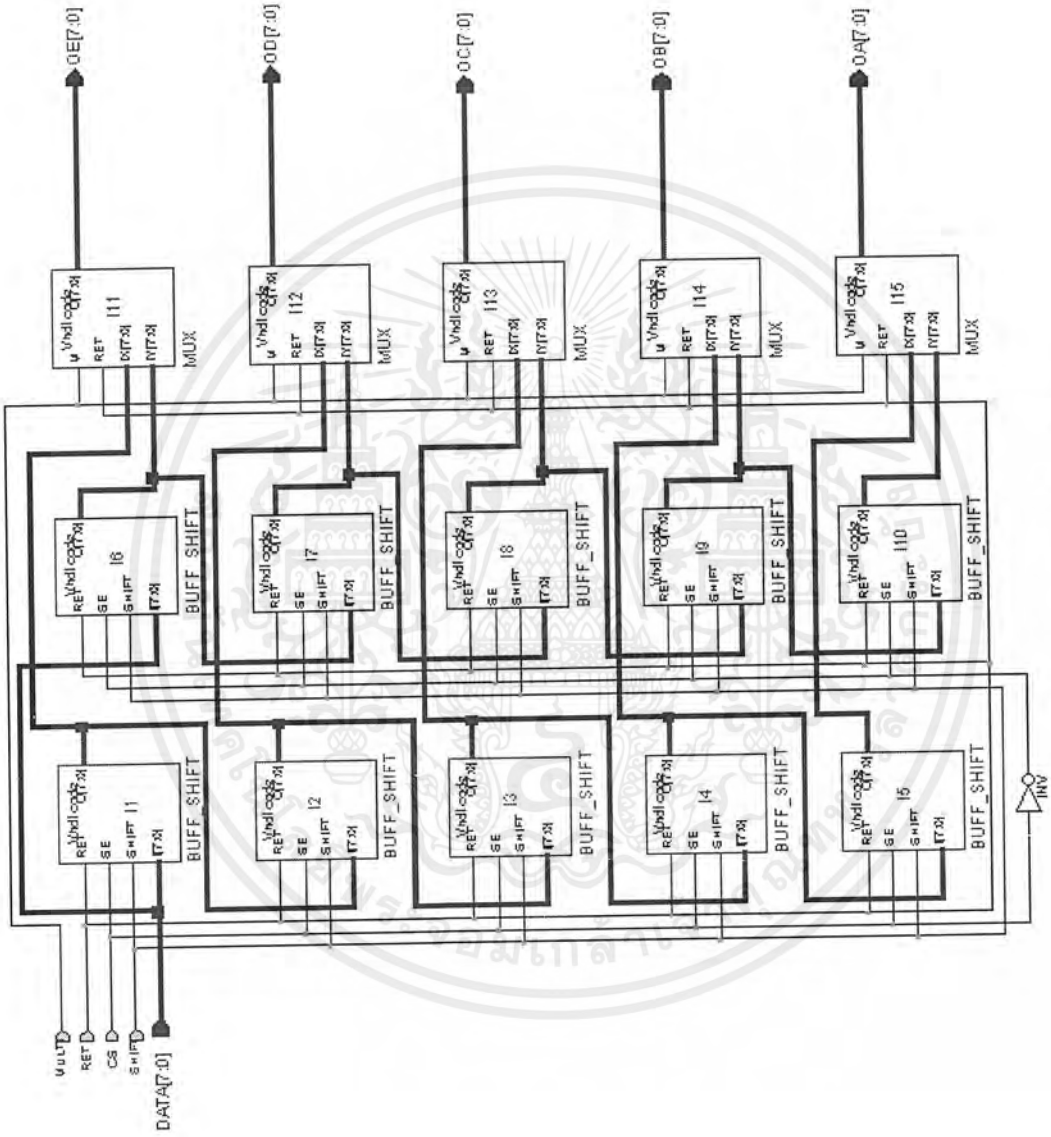
1) ใช้หลักการค้นหาข้อมูลในโค้ดบุ๊คแบบ Half Search เพื่อลดเวลาในการทำงานให้รวดเร็วขึ้น
2) ในการออกแบบตัวโค้ดบุ๊คควรใช้ตัวแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอลตัวเดียวกับที่ใช้งานในวงจร
3) ลดขนาดของฮาร์ดแวร์ให้เล็กลงเพื่อสามารถนำไปใช้งานเป็นอุปกรณ์เข้ารหัสถอดรหัสสัญญาณเสียงได้
4) ออกแบบวงจรฮาร์ดแวร์ให้ถูกต้องตามหลักวิชาการ
5) เพิ่มวงจรปรับแรงดันออฟเซตให้มีเสถียรภาพมากขึ้น

ภาคผนวก ก
รายละเอียดวงจรและผลการทดสอบวงจร

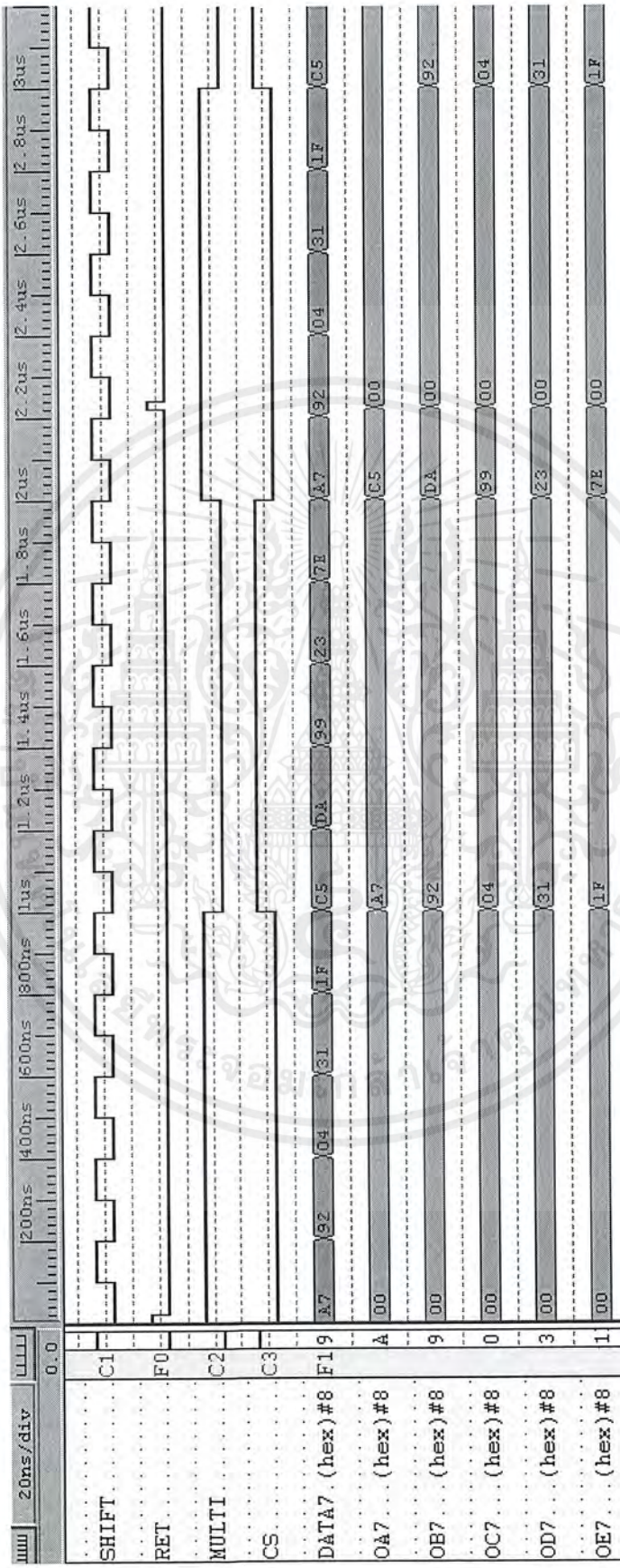


รูปที่ ก.1 บล็อกการทำงานตัวประมวลผลสำหรับค้นหาข้อมูลในโค้ดบุ๊ค

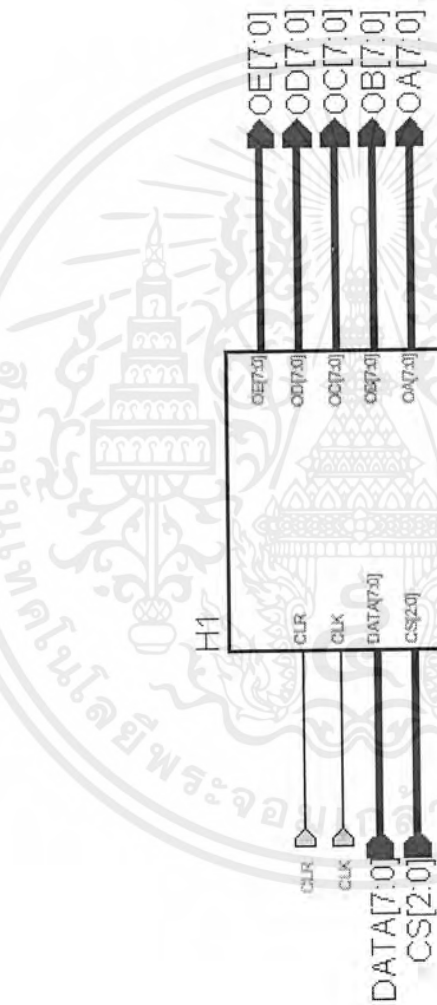
รูปที่ ก.2 บล็อกการทำงานของ DATA_BUFFER



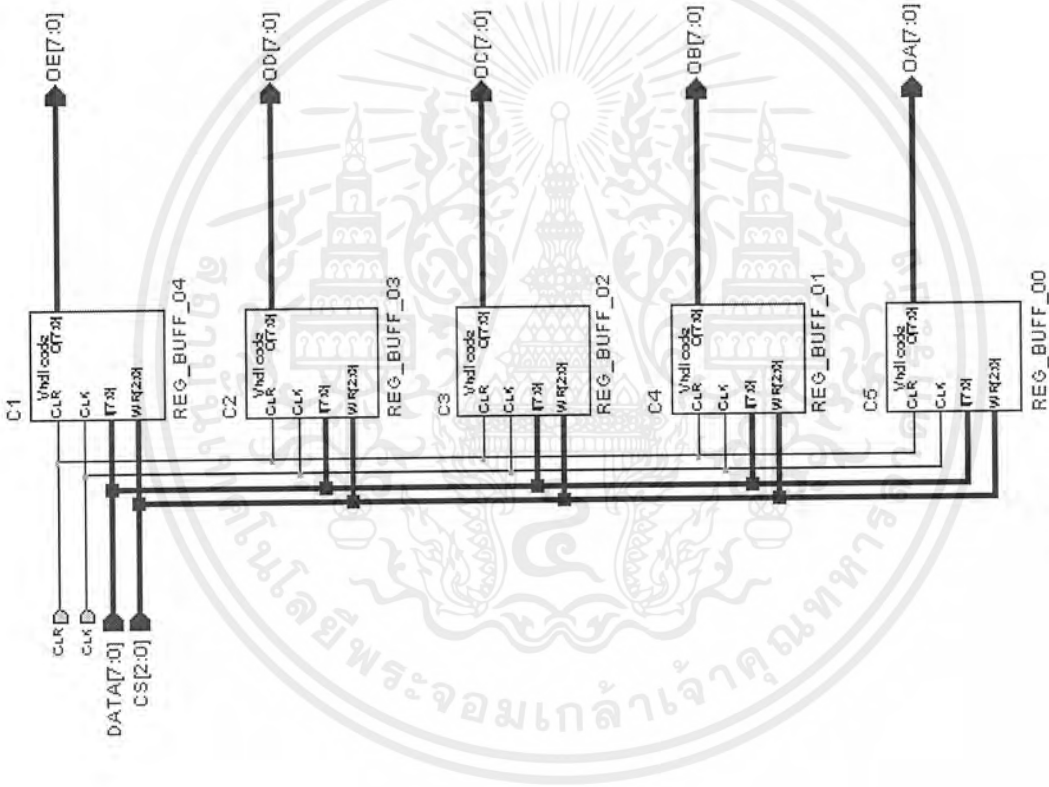
รูปที่ ก.3 วงจรDATA_BUFFER



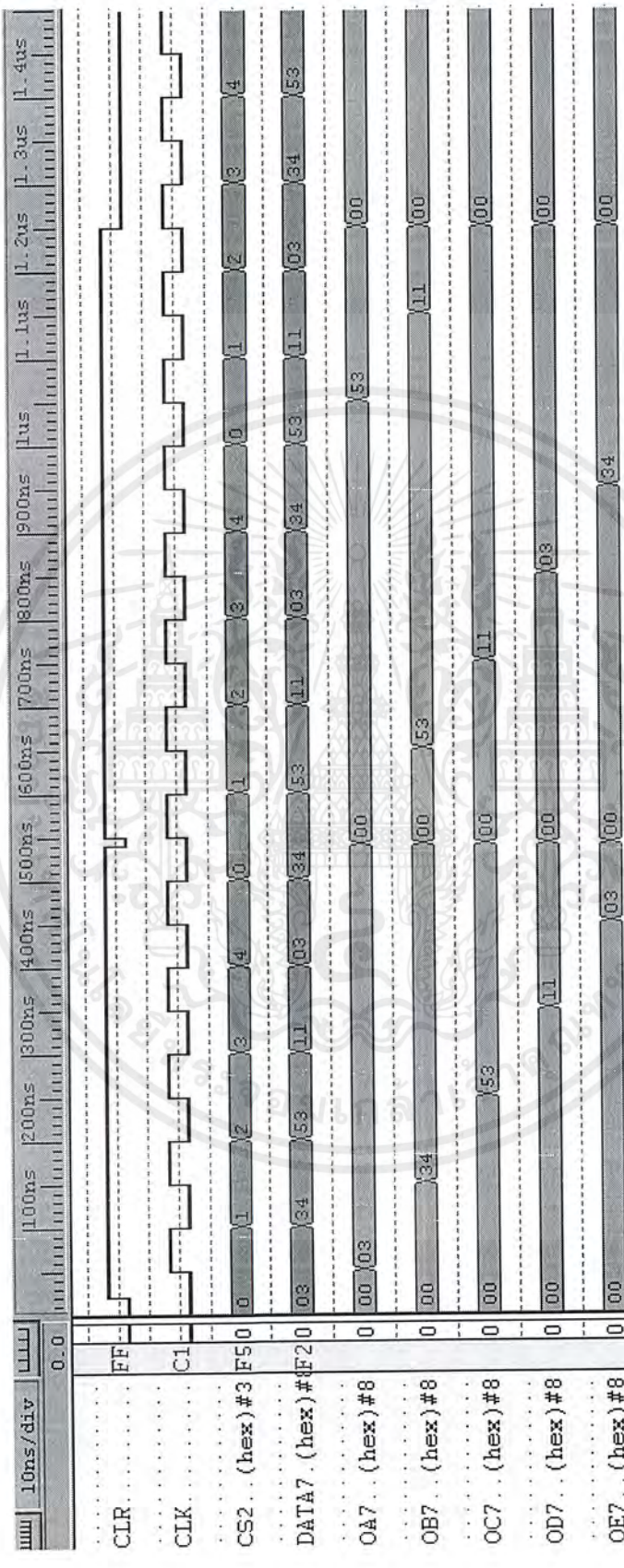
รูปที่ ก.4 ผลการเลียนแบบการทำงานวงจร DATA_BUFFER



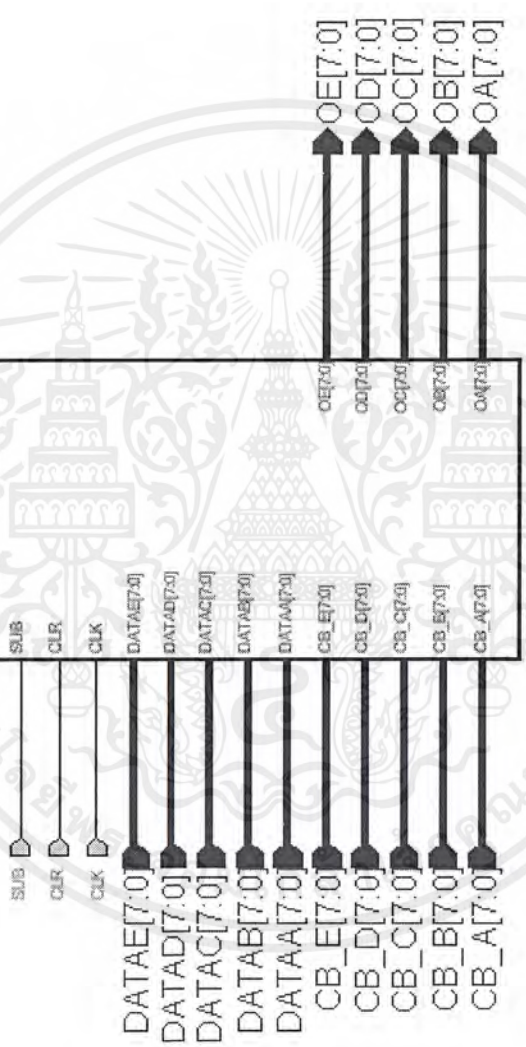
รูปที่ ก.5 บล็อกการทำงานของวงจร CB_BUFFER



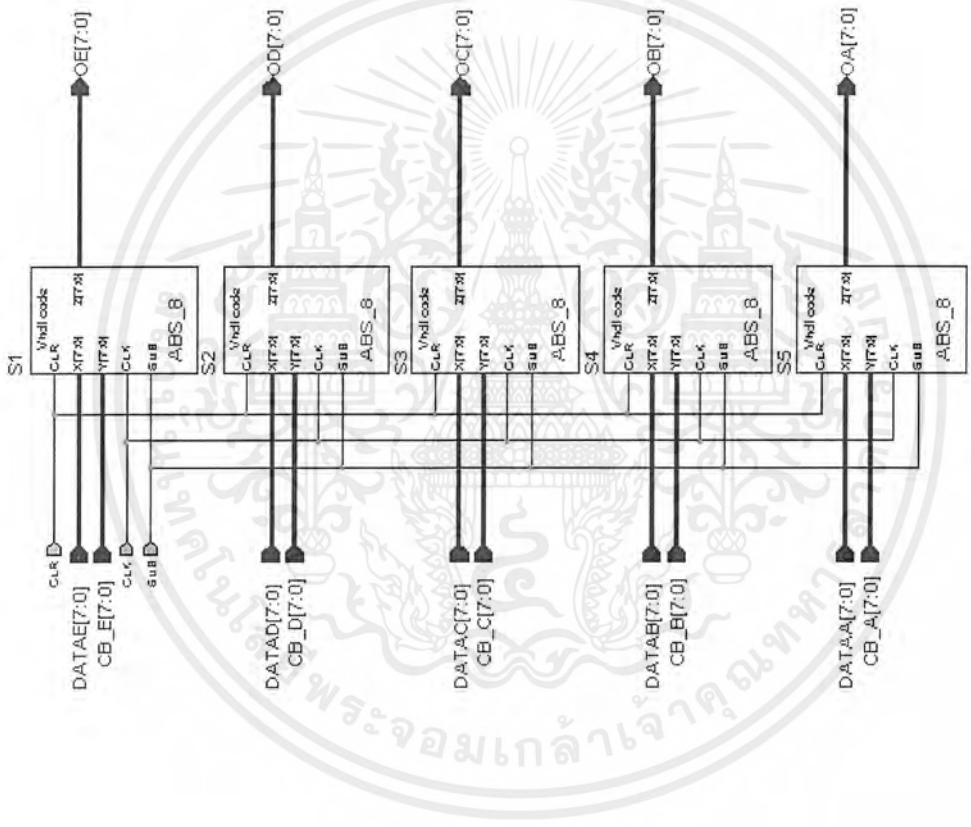
รูปที่ ก.6 วงจร CB_BUFFER



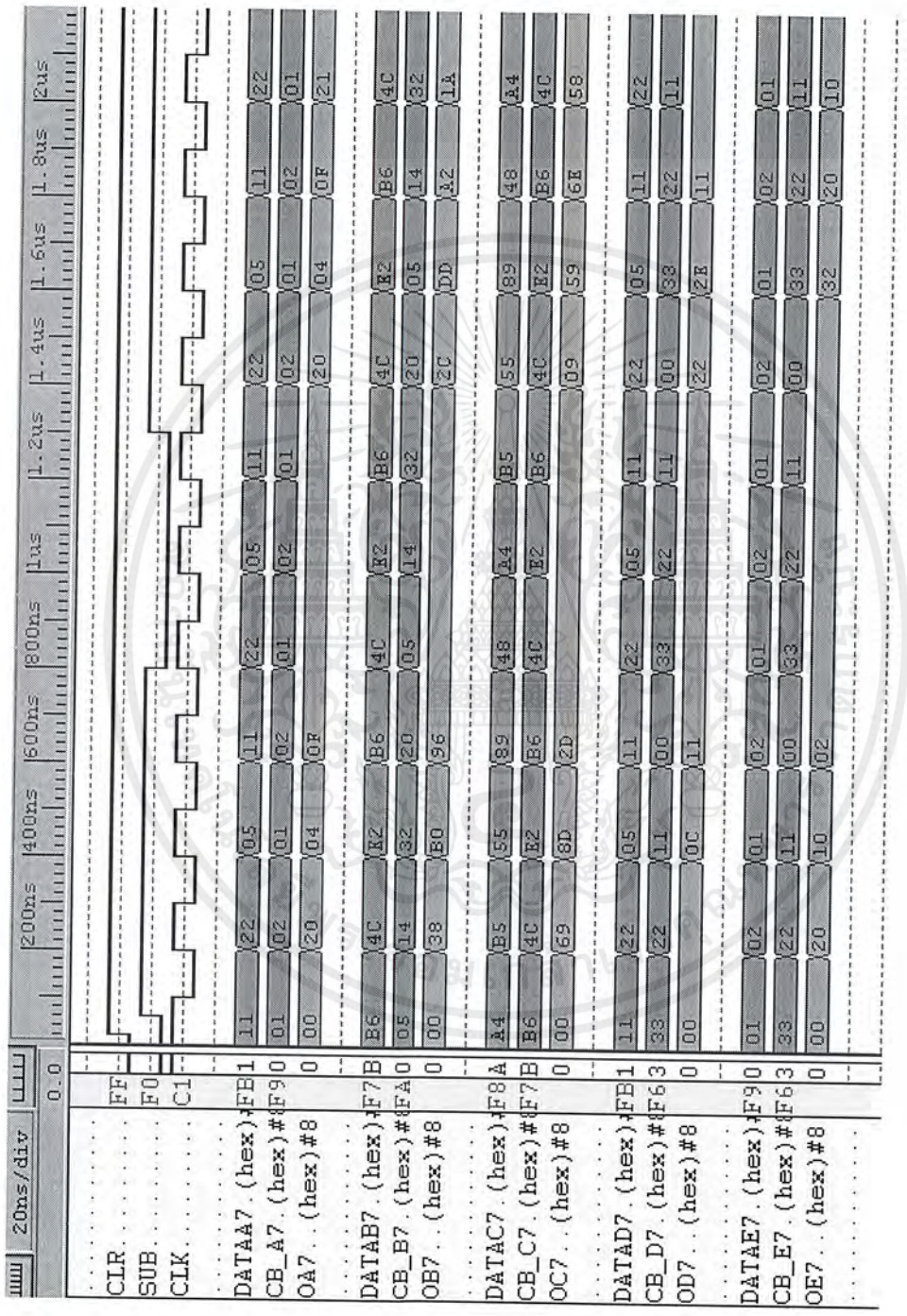
รูปที่ ก.7 ผลการเลียนแบบการทำงานของวงจร CB_BUFFER



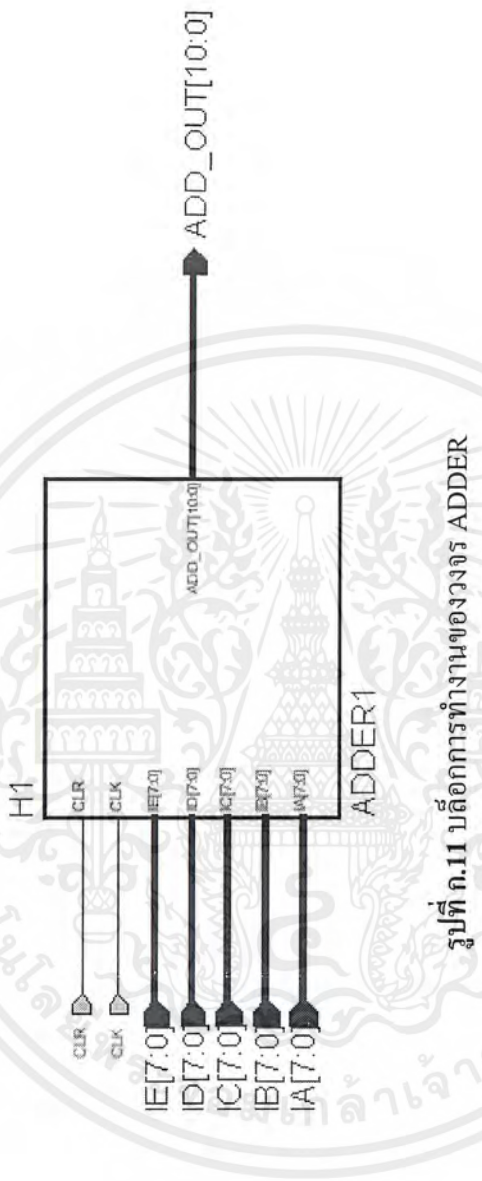
รูปที่ ก.8 บล็อกการทำงานของวงจร SUB_ABS



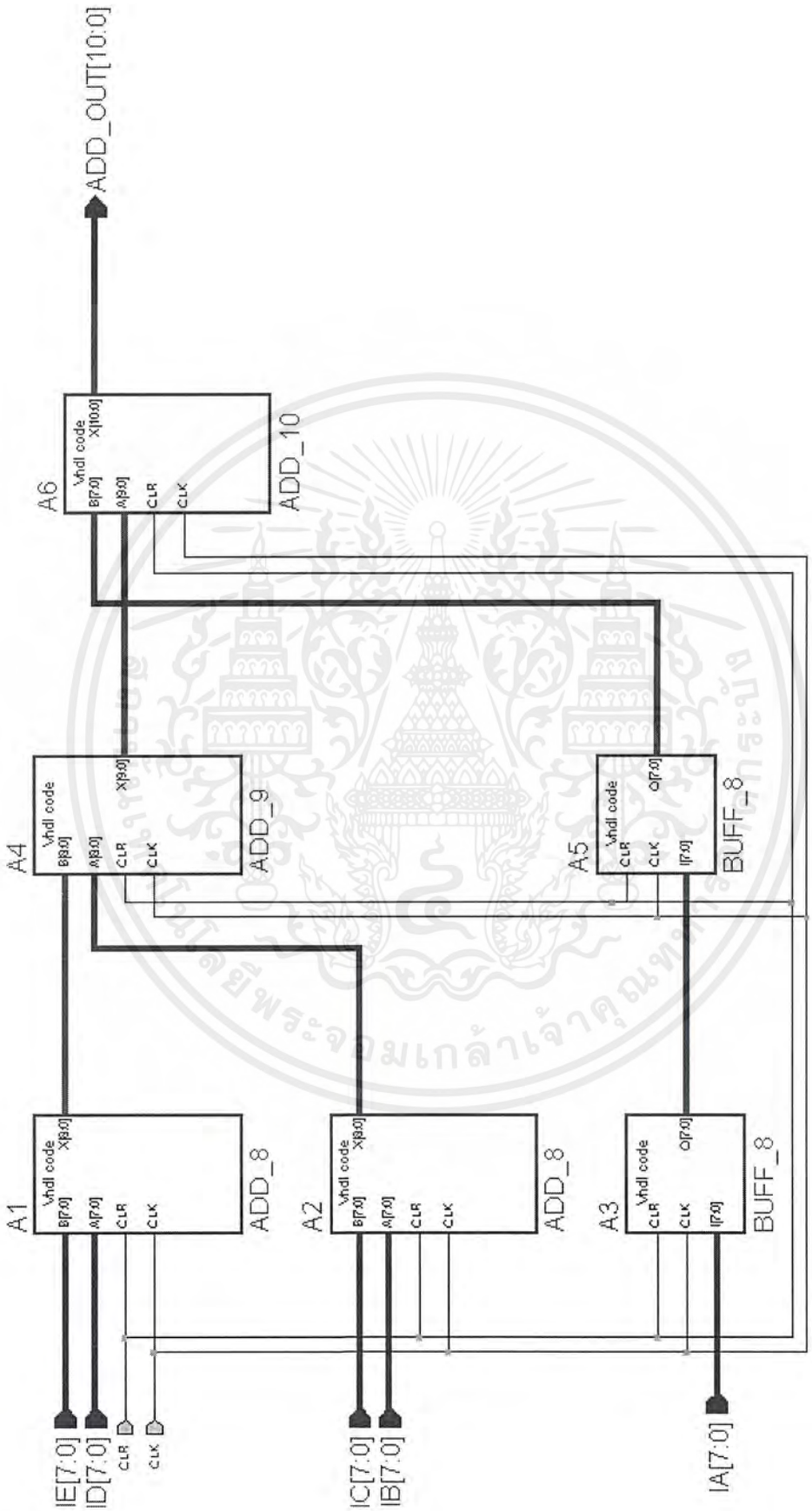
รูปที่ ก.9 วงจร SUB_ABS



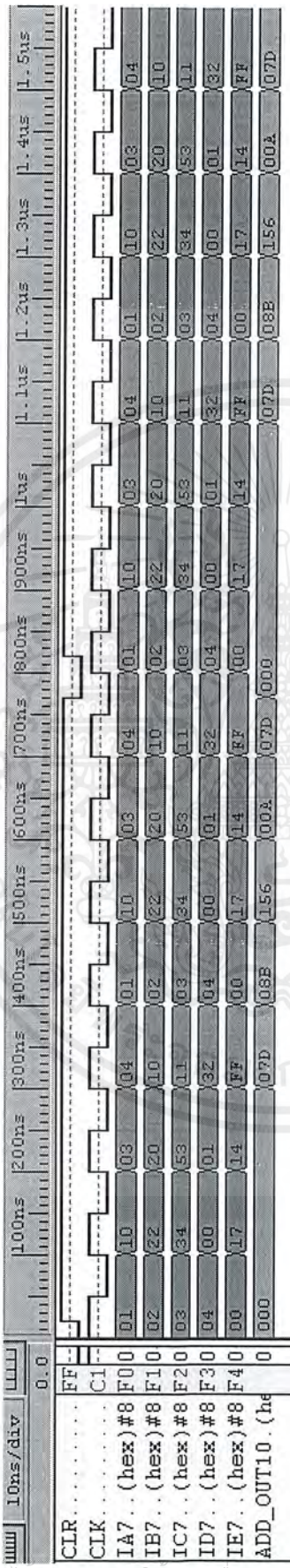
รูปที่ ก.10 ผลการเลียนแบบการทำงานของวงจร SUB_ABS



รูปที่ ก.11 บล็อกการทำงานของวงจร ADDER

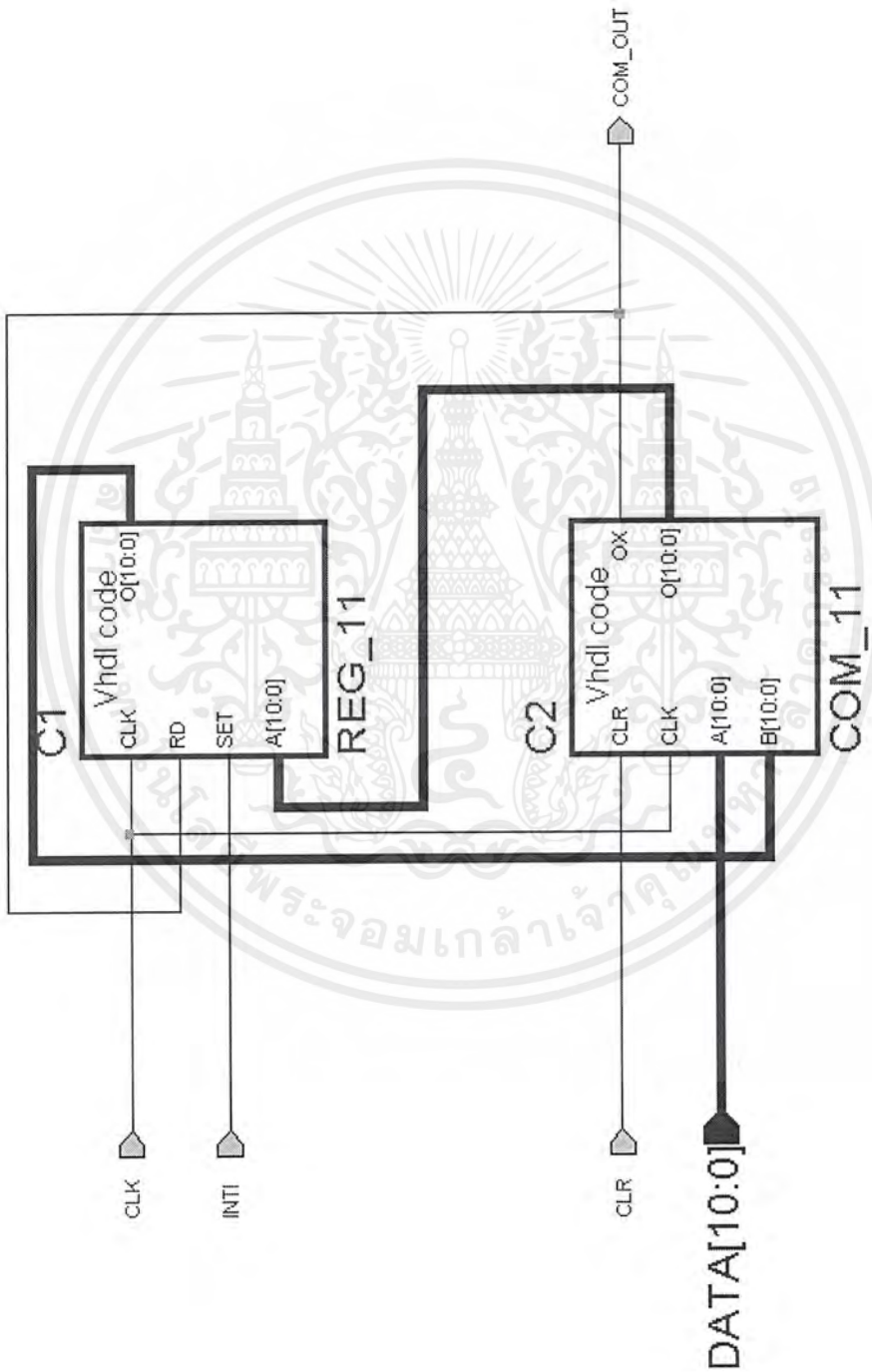


รูปที่ ก.12 วงจร ADDER

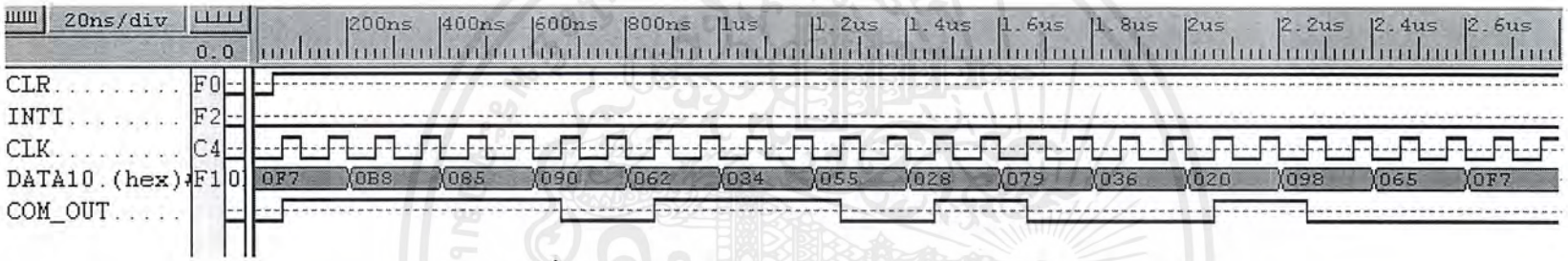


รูปที่ ก.13 ผลการเสียนแบบการทำงานของวงจร ADDER

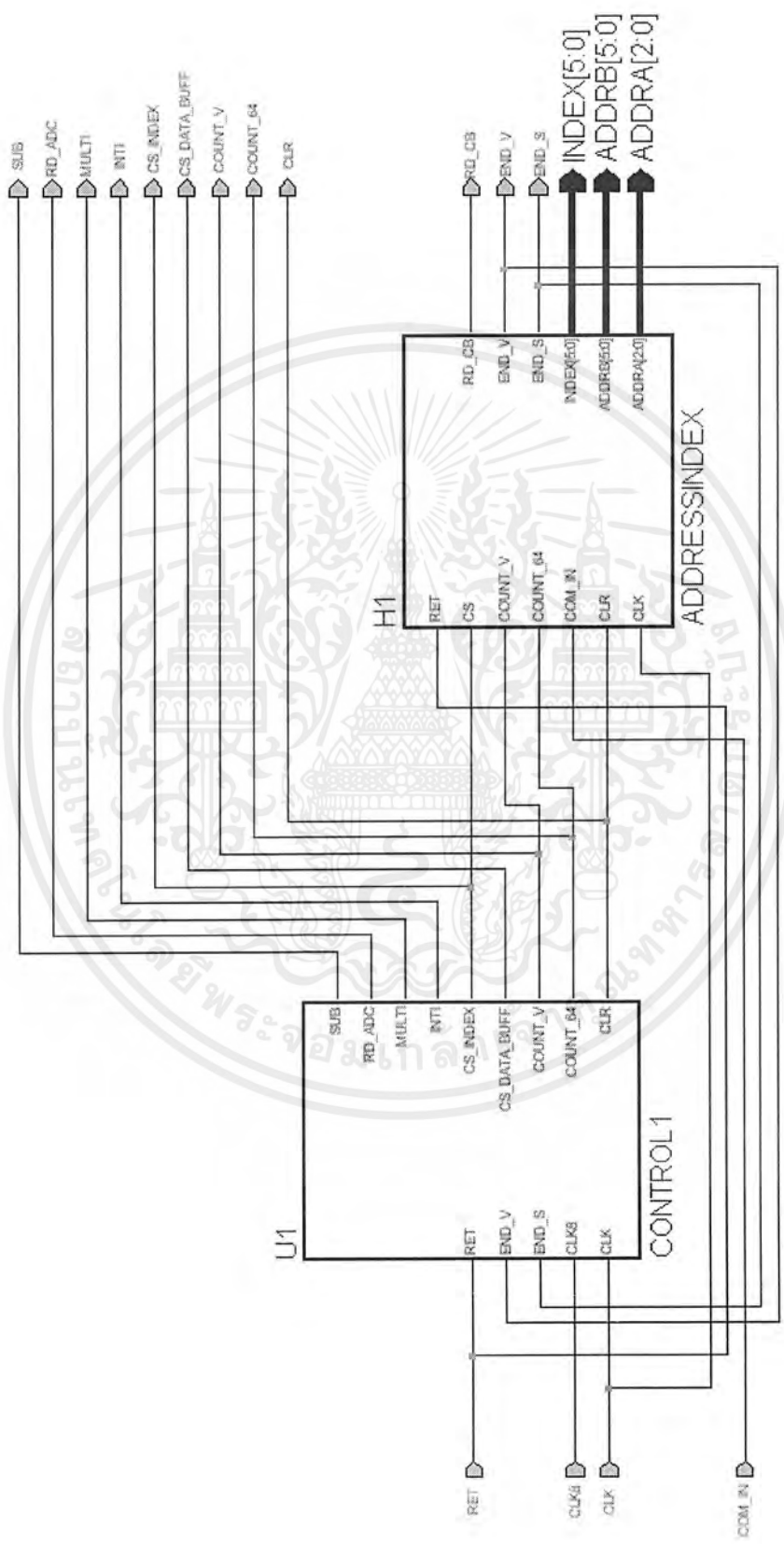
รูปที่ ก.14 บล็อกการทำงานของวงจร COMPARATOR



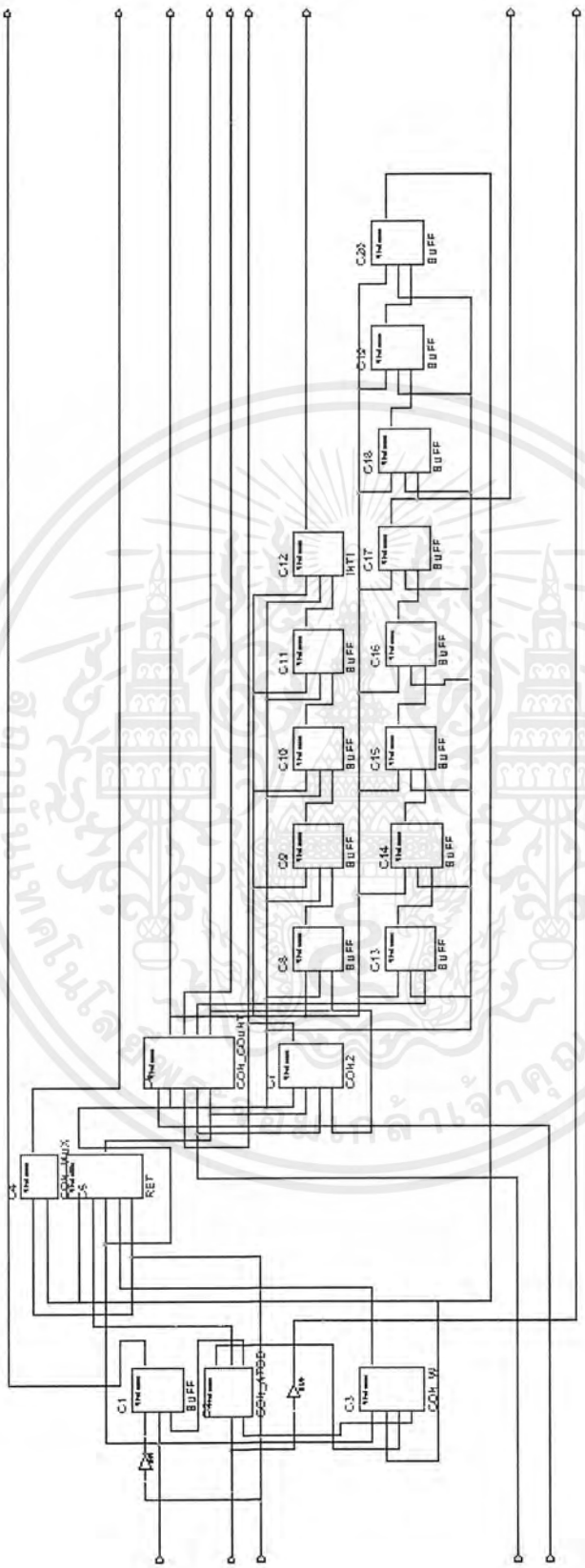
รูปที่ ก.15 วงจร COMPARATOR



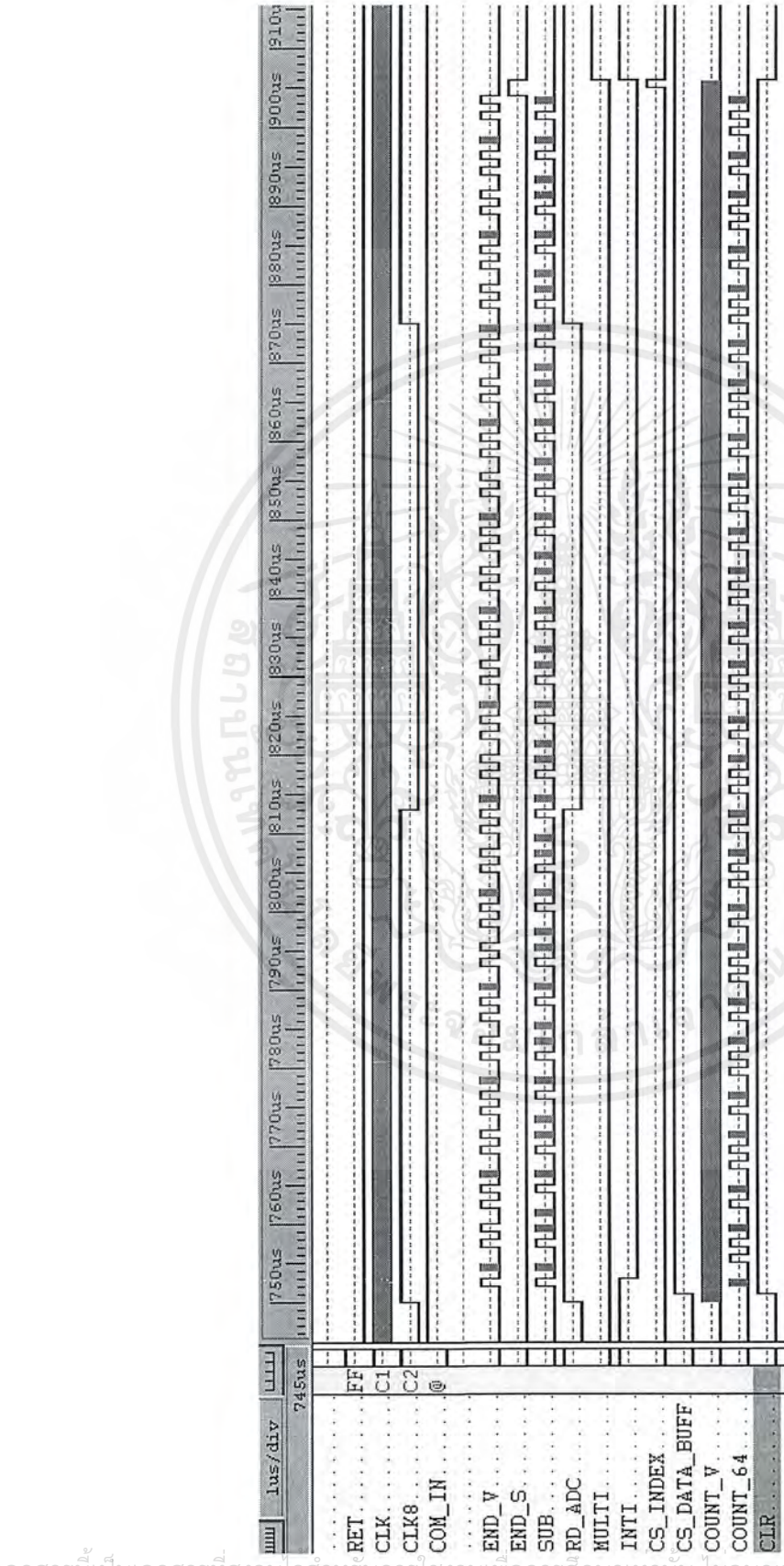
รูปที่ ก.16 ผลการเลียนแบบการทำงานของวงจร COMPARATOR



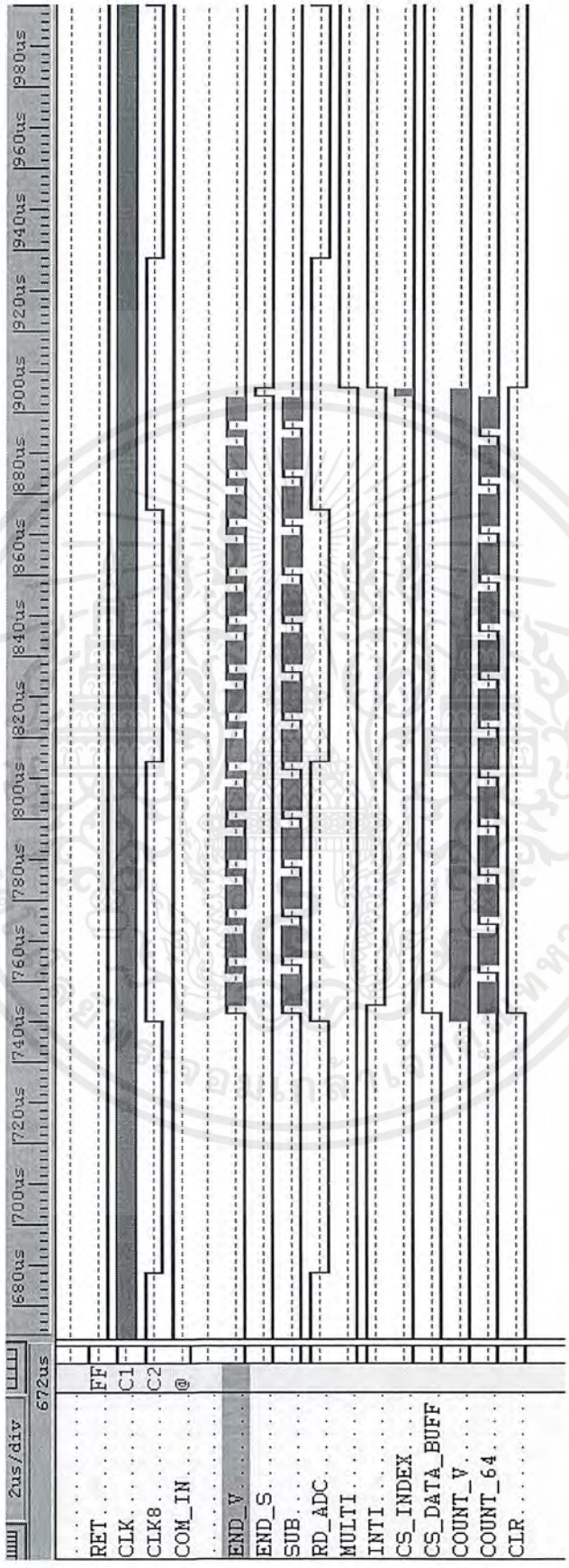
รูปที่ ก.17 บล็อกการทำงานของวงจร CONTROL และวงจร ADDRESS & INDEX



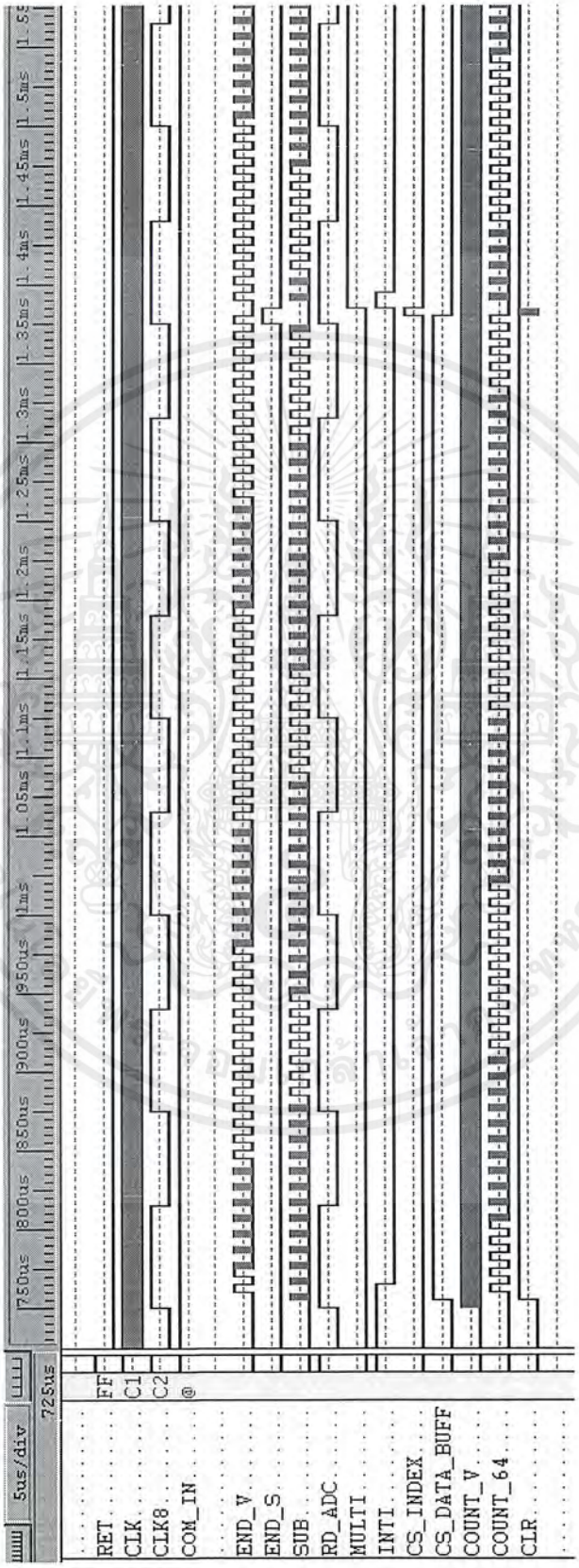
รูปที่ ก.18 วงจร CONTROL



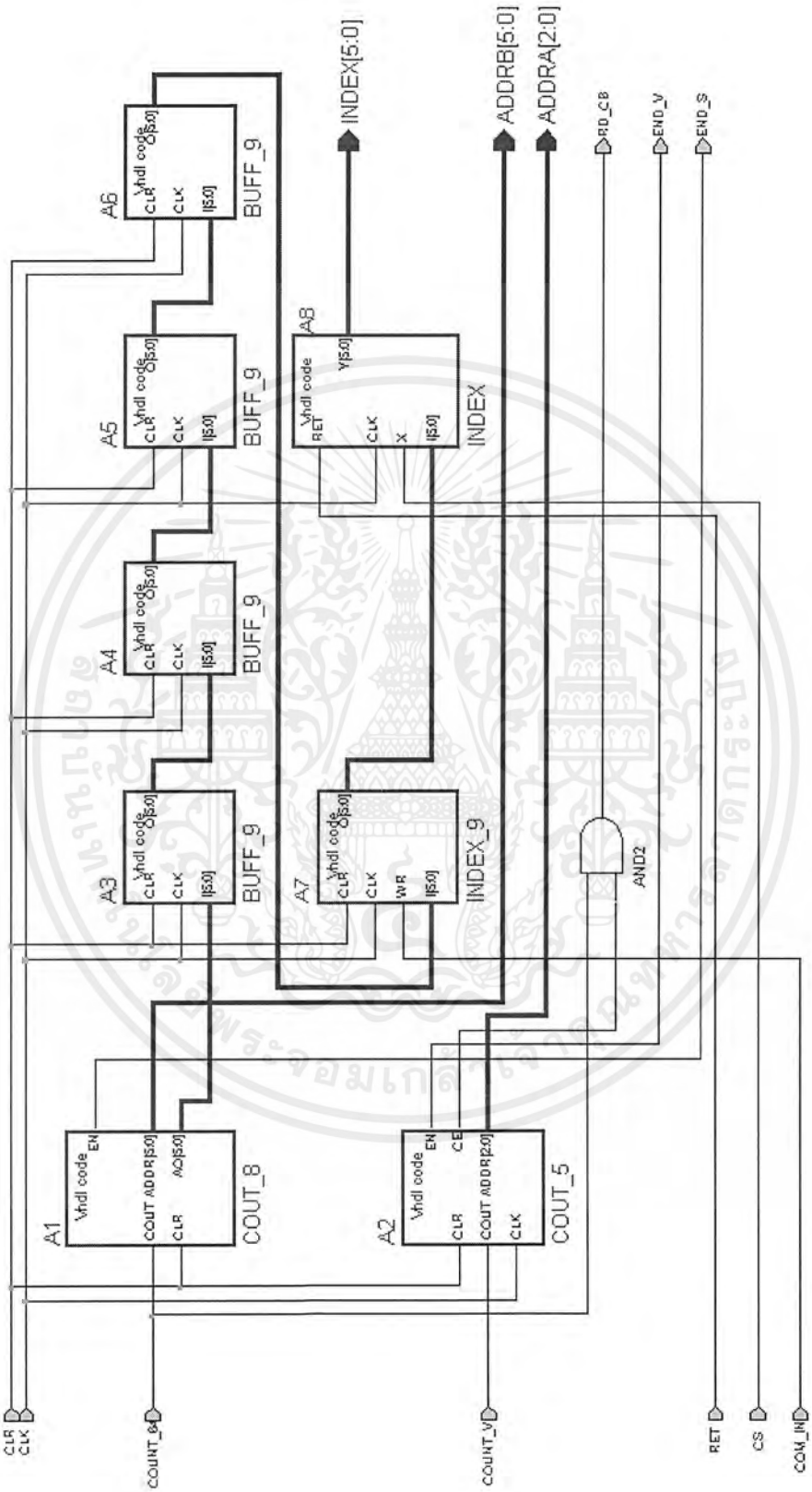
รูปที่ ก.19 ผลการเขียนแบบการทำงานของวงจร CONTROL



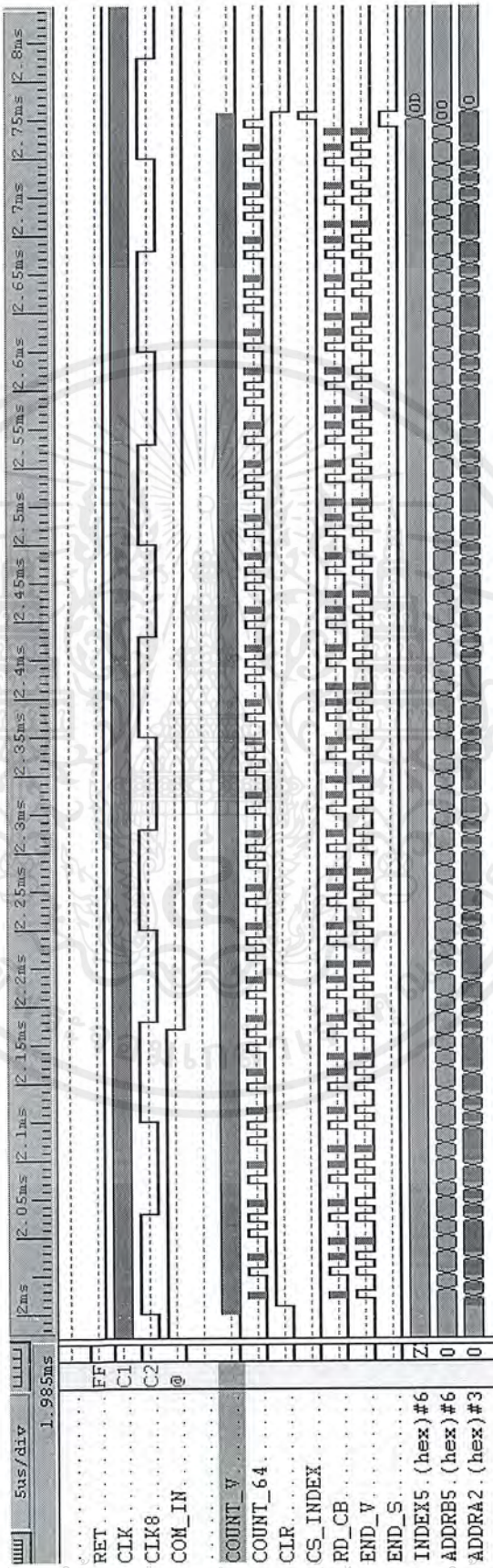
รูปที่ ก.20 ผลการเลียนแบบการทำงานส่วนย่อยของวงจร CONTROL



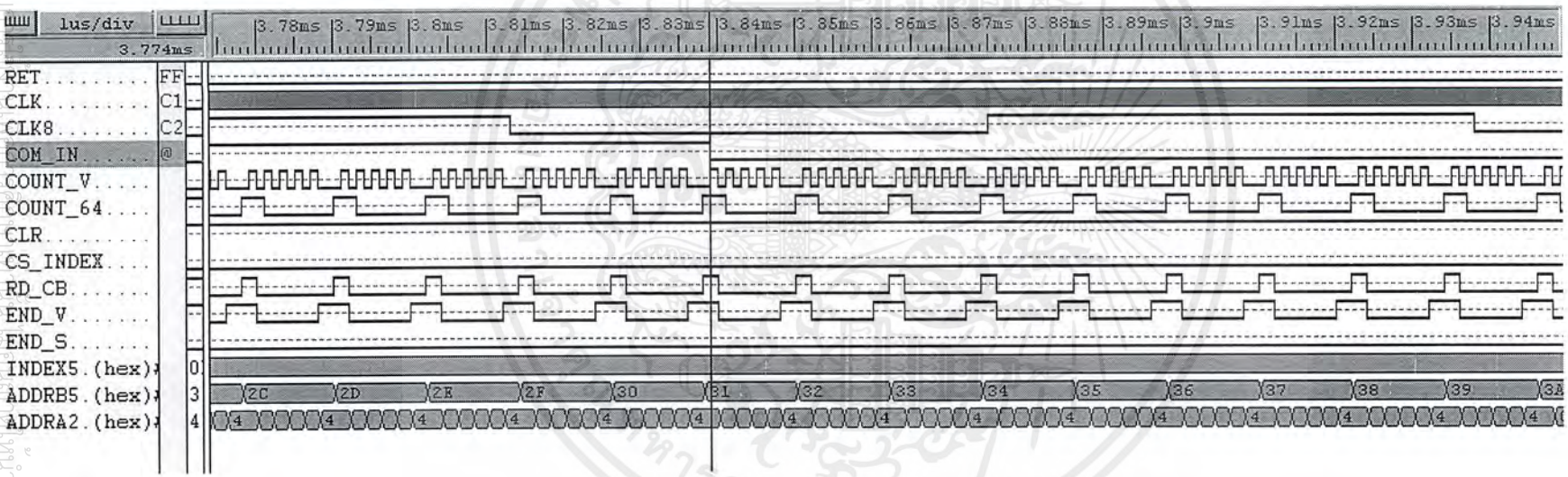
รูปที่ ก.21 ผลการเลียนแบบการทำงานของวงจร CONTROL ที่ความถี่ต่ำสุด



รูปที่ ก.22 วงจร ADDRESS & INDEX

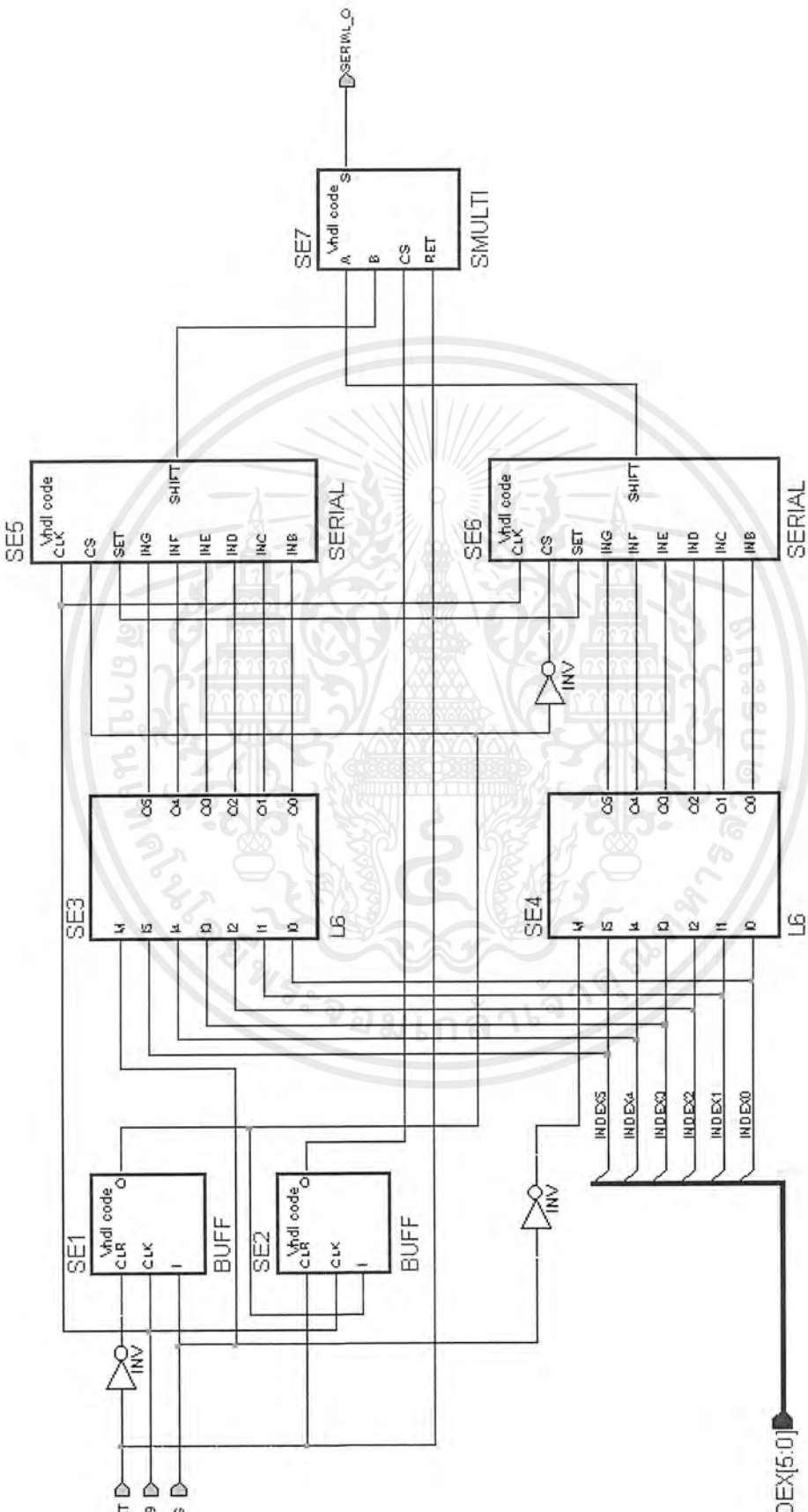


รูปที่ ก.23 ผลการเลียนแบบการทำงานของวงจร ADDRESS & INDEX

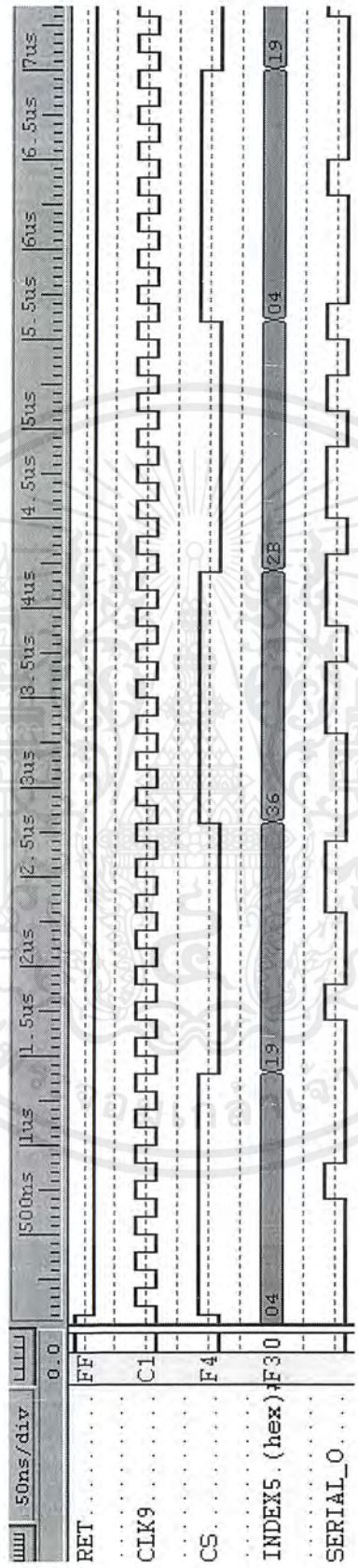


รูปที่ ก.24 ผลการเลียนแบบการทำงานของวงจร ADDRESS & INDEX

รูปที่ ก.25 บล็อกการทำงานของวงจร SERIAL_OUT

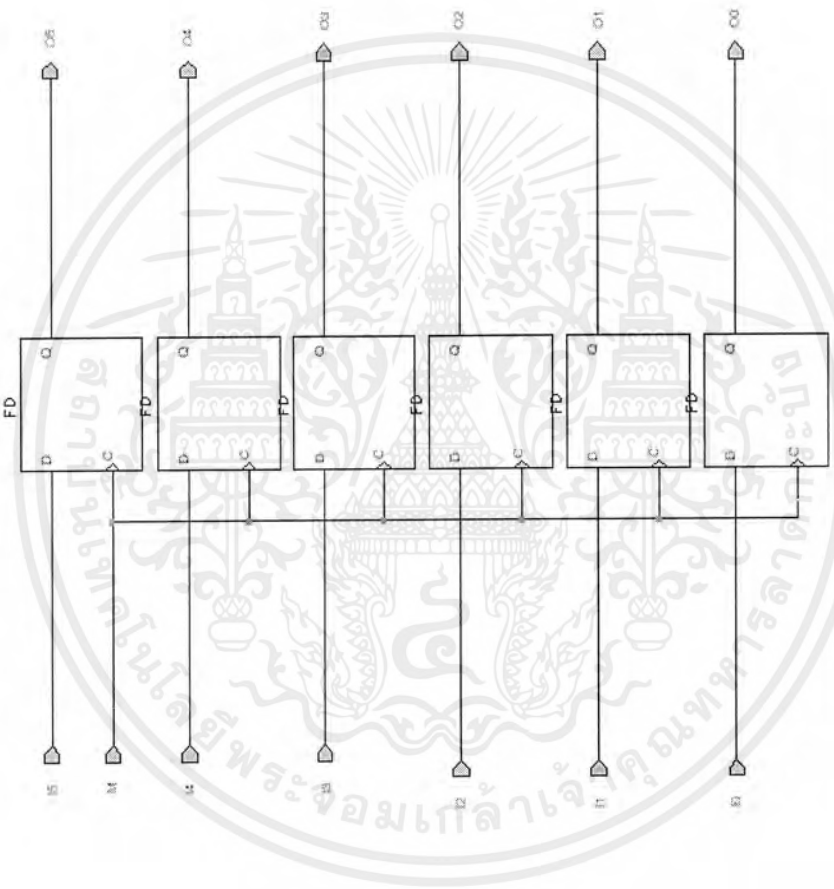


รูปที่ ก.26 วงจร SERIAL_OUT



รูปที่ ก.27 ผลการเลียนแบบการทำงานของวงจร SERIAL_OUT

รูปที่ ก.28 บล็อกการทำงานของวงจร REGISTER



รูปที่ ก.29 วงจร REGISTER

ภาคผนวก ข

โปรแกรม VHDL

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

entity abs_8 is
    port (
        x   :   in INTEGER range 0 to 255;
        y   :   in INTEGER range 0 to 255;
        clk :   in STD_LOGIC;
        clr :   in STD_LOGIC;
        sub :   in STD_LOGIC;
        z   :   out INTEGER range 0 to 255
        );
end abs_8;

architecture abs_8_arch of abs_8 is
begin
    process (clk, clr, x, y, sub)
    begin
        if clr = '0' then
            z <= 0;
        elsif clk'event and clk = '1' then
            if sub='1' then
                if x < y then z<= (y-x);
                    else z <= (x-y);
                end if;
            end if;
```

```
        end  if;
    end  process;
end  abs_8_arch;
```

รูปที่ ข.1 โปรแกรมวงจร abs_8

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

entity add_10 is
    port (
        a   :  in  INTEGER range 0 to 1023;
        b   :  in  INTEGER range 0 to 255;
        clk :  in  STD_LOGIC;
        clr :  in  STD_LOGIC;
        x   :  out INTEGER range 0 to 2047
        );
end add_10;

architecture add_10_arch of add_10 is
begin
    process (clr, clk)
    begin
        if (clr) = '0' then
            x <= 0;
        elsif CLK'event and CLK = '1' then
            x <= (a+b);
        end if;
    end process;
end add_10_arch;
```

รูปที่ ข.2 โปรแกรมวงจร add_10

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

entity add_9 is
port (
    a   :   in  INTEGER range 0 to 511;
    b   :   in  INTEGER range 0 to 511;
    clk :   in  STD_LOGIC;
    clr :   in  STD_LOGIC;
    x   :   out INTEGER range 0 to 1023
    );
end add_9;

architecture add_9_arch of add_9 is
begin
    process (clr, clk)
    begin
        if (clr) = '0' then
            x <= 0;
        elsif CLK'event and CLK = '1' then
            x <= (a+b);
        end if;
    end process;
end add_9_arch;
```

รูปที่ ข.3 โปรแกรมวงจร add_9

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity add_8 is
port (
        a   :  in  INTEGER  range 0 to 255;
        b   :  in   INTEGER  range 0 to 255;
        clk :  in   STD_LOGIC;
        clr :  in   STD_LOGIC;
        x   :  out  INTEGER  range 0 to 511
        );
end add_8;

architecture add_8_arch of add_8 is
begin
        process (clr, clk)
        begin
                if (clr) = '0' then
                        x <= 0;
                elsif CLK'event and CLK = '1' then
                        x <= (a+b);
                end if;
        end process;
end add_8_arch;
```

รูปที่ ข.4 โปรแกรมวงจร add_8

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity buff_8 is
        port (
                I   : in INTEGER range 0 to 255;
                clk : in STD_LOGIC;
                clr : in STD_LOGIC;
                o   : out INTEGER range 0 to 255
                );
end buff_8;

architecture buff_8_arch of buff_8 is
begin
        process (clk, clr)
        begin
                if (clr) = '0' then
                        o <= 0;
                elsif clk'event and clk = '1' then
                        o <= i;
                end if;
        end process;
end buff_8_arch;
```

รูปที่ ข.5 โปรแกรมวงจร buff_8

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity buff_9 is
        port (
                clk : in STD_LOGIC;
                clr : in STD_LOGIC;
                I   : in INTEGER range 0 to 63;
                o   : out INTEGER range 0 to 63
                );
end buff_9;

architecture buff_9_arch of buff_9 is
begin
        process (clk, clr)
        begin
                if clr = '0' then
                        o <= 0;
                elsif clk'event and clk = '1' then
                        o <= i;
                end if;
        end process;
end buff_9_arch;
```

รูปที่ ข.6 โปรแกรมวงจร buff_9

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

entity buff_shift is
    port (
        shift : in STD_LOGIC;
        ret   : in STD_LOGIC;
        se    : in STD_LOGIC;
        i     : in INTEGER range 0 to 255;
        o     : out INTEGER range 0 to 255
        );
end buff_shift;

architecture buff_shift_arch of buff_shift is
begin
    process (shift, ret, se)
    begin
        if ret = '1' then
            o <= 0;
        elsif shift'event and shift = '1' then
            if se = '1' then
                o <= i;
            end if;
        end if;
    end process;
end buff_shift_arch;
```

รูปที่ ข.7 โปรแกรมวงจร buff_shift

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity com_11 is
    port (
        clk : in STD_LOGIC;
        clr : in STD_LOGIC;
        ox : out STD_LOGIC;
        o : out INTEGER range 0 to 2047;
        a : in INTEGER range 0 to 2047;
        b : in INTEGER range 0 to 2047
        );
end com_11;

architecture com_11_arch of com_11 is
begin
    process (clr, clk)
    begin
        if clr = '0' then
            o <= 0;
        elsif clk'event and clk = '1' then
            if (a < b) or (a = b) then
                ox <= '1';
                o <= a;
            else ox <= '0';
                o <= b;
            end if;
```

```
            end  if;
        end  process;
    end  com_11_arch;
```

รูปที่ ข.8 โปรแกรมวงจร com_11

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

entity con_AtoD is
    port (
        ret     :  in    STD_LOGIC;
        clk8k   :  in    STD_LOGIC;
        con_ret :  out   STD_LOGIC;
        c_wait  :  out   STD_LOGIC;
        sel     :  inout STD_LOGIC
        );
end con_AtoD;

architecture con_AtoD_arch of con_AtoD is
    signal x : INTEGER range 0 to 7;
begin
    process (ret, clk8k)
    begin
        if ret = '1' then
            con_ret <= '1';
            sel <= '0';
            x <= 0;
            c_wait <= '1';
        elsif clk8k'event and clk8k = '1' then
            x <= x+1;
```

```
            if  x = 5  then
                x <= 1;
                sel <= not sel;
                con_ret <= '0';
                c_wait <= '0';
            else  c_wait <= '1';
            end  if;
        end  if;
    end  process;
end  con_AtoD_arch;
```

รูปที่ ข.9 โปรแกรมวงจร con_AtoD

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity con_count is
        port (
            clk    : in     STD_LOGIC;
            clr    : out    STD_LOGIC;
            reset  : in     STD_LOGIC;
            rd     : in     STD_LOGIC;
            goto   : in     STD_LOGIC;
            set    : in     STD_LOGIC;
            run    : inout  STD_LOGIC;
            sub    : out    STD_LOGIC;
            s_Index : out STD_LOGIC
            );
end con_count;

architecture con_count_arch of con_count is
begin
        process (clk, reset, rd, goto, set)
        begin
            if reset = '1' then
                clr <= '0';
                run <= '0';
                s_index <= '0';
            elsif clk'event and clk = '1' then
                clr <= '1';
                run <= not run;
```

```
            if  rd = '1'  then
                run <= '0';
                sub <= '1';
            end  if;
            if  goto = '1'  then
                sub <= '0';
                run <= not run;
            end  if;
            if  set = '1'  then
                s_index <= '1';
            else  s_index <= '0';
            end  if;
        end  if;
    end  process;
end  con_count_arch;
```

รูปที่ ข.10 โปรแกรมวงจร con_count

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

entity con_mux is
        port (
            ret  :  in     STD_LOGIC;
            set  :  in     STD_LOGIC;
            om   :  inout  STD_LOGIC
            );
end con_mux;

architecture con_mux_arch of con_mux is
begin
        process (ret, set)
        begin
            if ret = '1' then
                om <= '0';
            elsif set'event and set = '1' then
                om <= not om;
            end if;
        end process;
end con_mux_arch;
```

รูปที่ ข.11 โปรแกรมวงจร con_mux

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity con_wait is
    port (
        clk : in STD_LOGIC;
        ina : in STD_LOGIC;
        inb : in STD_LOGIC;
        ret : in STD_LOGIC;
        o : out STD_LOGIC
        );
end con_wait;

architecture con_wait_arch of con_wait is
begin
    process (clk, ret, ina, inb)
    begin
        if ret = '1' then
            o <= '0';
        elsif clk'event and clk = '1' then
            if ina = '1' and inb = '1' then
                o <= '1';
            end if;
            if ina = '0' then
                o <= '0';
            end if;
```

```
            end  if;
        end  process;
end  con_wait_arch;
            end  if;
        end  process;
end  con_wait_arch;
```

รูปที่ ข.12 โปรแกรมวงจร con_wait

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

entity con2 is
        port (
            clk  :  in   STD_LOGIC;
            ret  :  in   STD_LOGIC;
            rd   :  in   STD_LOGIC;
            sub  :  in   STD_LOGIC;
            go   :  out  STD_LOGIC
            );
end con2;

architecture con2_arch of con2 is
begin
        process (clk, ret, rd, sub)
        begin
            if ret = '1' then
                go <= '0';
            elsif clk'event and clk = '1' then
                if sub = '1' then
                    go <= '1';
                end if;
                if rd = '0' then
                    go <= '0';
```

```
            end  if;
        end  if;
    end  process;
end  con2_arch;
```

รูปที่ ข.13 โปรแกรมวงจร con2

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity cout_5 is
        port (
            clk   : in    STD_LOGIC;
            clr   : in    STD_LOGIC;
            ce    : out   STD_LOGIC;
            en    : out   STD_LOGIC;
            addr  : inout INTEGER range 0 to 7;
            cout  : in    STD_LOGIC
            );
end cout_5;

architecture cout_5_arch of cout_5 is
begin
        process (clk, clr, cout)
        begin
            if clr = '0' then
                addr <= 0;
                ce <= '0';
                en <= '0';
            elsif clk = '1' and clk'event then
                if cout = '1' then
```

```
            if  addr = 3  then
                ce <= '1';
            else  ce <= '0';
            end  if;
            if  addr = 4  then
                en <= '0';
            end  if;
            addr <= (addr + 1);
            if  addr = 3  then
                en <= '1';
            end  if;
            if  addr > 3  then
                addr <= addr - 4;
            end  if;
        end  if;
    end  if;
end  process;
end  cout_5_arch;
```

รูปที่ ข.14 โปรแกรมวงจร cout_5

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity cout_8 is
    port (
        clr  : in    STD_LOGIC;
        en   : out   STD_LOGIC;
        addr : inout INTEGER range 0 to 63;
        ao   : out   INTEGER range 0 to 63;
        cout : in    STD_LOGIC
    );
end cout_8;

architecture cout_8_arch of cout_8 is
begin
    process (clr, cout)
    begin
        if clr = '0' then
            addr <= 0;
            ao <= 0;
            en <= '0';
        elsif cout = '1' and cout'event then
            addr <= (addr + 1);
            ao <= addr;
            if addr = 63 then
```

```
                    en <= '1';
                else  en <= '0';
                end  if;
            end  if;
        end  process;
end  cout_8_arch;
```

รูปที่ ข.15 โปรแกรมวงจร cout_8

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity index is
        port (
            clk : in    STD_LOGIC;
            ret : in    STD_LOGIC;
            x   : in    STD_LOGIC;
            I   : in    STD_LOGIC_VECTOR(5 downto 0);
            y   : out   STD_LOGIC_VECTOR(5 downto 0)
            );
end index;

architecture index_arch of index is
begin
        process (clk, ret, x)
        begin
            if ret = '1' then
                y <= "ZZZZZZ";
            elsif clk'event and clk = '1' then
                if x = '1' then
                    y <= I;
                end if;
            end if;
        end process;
end index_arch;
```

รูปที่ ข.16 โปรแกรมวงจร index

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity index_9 is
        port (
            clk : in STD_LOGIC;
            clr : in STD_LOGIC;
            wr : in STD_LOGIC;
            I : in INTEGER range 0 to 63;
            o : out INTEGER range 0 to 63
            );
end index_9;

architecture index_9_arch of index_9 is
begin
        process (clk, clr, wr)
        begin
            if clr = '0' then
                o <= 63;
            elsif clk'event and clk = '1' then
                if wr = '1' then
                    o <= i;
                end if;
            end if;
        end process;
end index_9_arch;
```

รูปที่ ข.17 โปรแกรมวงจร index_9

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity inti is
    port (
        clk  : in  STD_LOGIC;
        clr  : in  STD_LOGIC;
        down : in  STD_LOGIC;
        inti : out STD_LOGIC
        );
end inti;

architecture inti_arch of inti is
begin
    process (clk, clr, down)
    begin
        if clr = '0' then
            inti <= '1';
        elsif clk'event and clk = '1' then
            if down = '1' then
                inti <= '0';
            end if;
        end if;
    end process;
end inti_arch;
```

รูปที่ ข.18 โปรแกรมวงจร inti

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

entity mux is
        port    (
                ix  : in  INTEGER range 0 to 255;
                iy  : in  INTEGER range 0 to 255;
                o   : out INTEGER range 0 to 255;
                ret : in  STD_LOGIC;
                m   : in  STD_LOGIC
                );
end mux;

architecture mux_arch of mux is
begin
        process (ret, m, ix, iy)
        begin
                if ret = '1' then
                        o <= 0;
                elsif m = '1' then
                        o <= ix;
                 else o <= iy;
                end if;
        end process;
end mux_arch;
```

รูปที่ ข.19 โปรแกรมวงจร mux

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

entity reg_buff_00 is
        port    (
                clk     : in    STD_LOGIC;
                clr     : in    STD_LOGIC;
                i       : in    INTEGER range 0 to 255;
                o       : out   INTEGER range 0 to 255;
                wr      : in    INTEGER range 0 to 7
                );
        end reg_buff_00;

architecture reg_buff_00_arch of reg_buff_00 is
begin
        process (clk, clr, wr)
         begin
                if clr = '0' then
                        o <= 0;
                elsif clk'event and clk = '1' then
                        if wr = 0 then
                                o <= i;
                        end if;
                end if;
        end process;
end reg_buff_00_arch;
```

รูปที่ ข.20 โปรแกรมวงจร reg_buff_00

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

entity reg_buff_01 is
        port    (
                clk     : in    STD_LOGIC;
                clr     : in    STD_LOGIC;
                i       : in    INTEGER range 0 to 255;
                o       : out   INTEGER range 0 to 255;
                wr      : in    INTEGER range 0 to 7
                );
        end reg_buff_01;

architecture reg_buff_01_arch of reg_buff_01 is
begin
        process (clk, clr, wr)
         begin
                if clr = '0' then
                        o <= 0;
                elsif clk'event and clk = '1' then
                        if wr = 1 then
                                o <= i;
                        end if;
                end if;
        end process;
end reg_buff_01_arch;
```

รูปที่ ข.21 โปรแกรมวงจร reg_buff_01

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

entity reg_buff_02 is
        port    (
                clk     : in    STD_LOGIC;
                clr     : in    STD_LOGIC;
                i       : in    INTEGER range 0 to 255;
                o       : out   INTEGER range 0 to 255;
                wr      : in    INTEGER range 0 to 7
                );
        end reg_buff_02;

architecture reg_buff_02_arch of reg_buff_02 is
begin
        process (clk, clr, wr)
        begin
                if clr = '0' then
                        o <= 0;
                elsif clk'event and clk = '1' then
                        if wr = 2 then
                                o <= i;
                        end if;
                end if;
        end process;
end reg_buff_02_arch;
```

รูปที่ ข.22 โปรแกรมวงจร reg_buff_02

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

entity reg_buff_03 is
        port    (
                clk     : in    STD_LOGIC;
                clr     : in    STD_LOGIC;
                i       : in    INTEGER range 0 to 255;
                o       : out   INTEGER range 0 to 255;
                wr      : in    INTEGER range 0 to 7
                );
        end reg_buff_03;

architecture reg_buff_03_arch of reg_buff_03 is
begin
        process (clk, clr, wr)
         begin
                if clr = '0' then
                        o <= 0;
                elsif clk'event and clk = '1' then
                        if wr = 3 then
                                o <= i;
                        end if;
                end if;
        end process;
end reg_buff_03_arch;
```

รูปที่ ข.23 โปรแกรมวงจร reg_buff_03

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

entity reg_buff_04 is
        port    (
                clk     : in    STD_LOGIC;
                clr     : in    STD_LOGIC;
                i       : in    INTEGER range 0 to 255;
                o       : out   INTEGER range 0 to 255;
                wr      : in    INTEGER range 0 to 7
                );
        end reg_buff_04;

architecture reg_buff_04_arch of reg_buff_04 is
begin
        process (clk, clr, wr)
         begin
                if clr = '0' then
                        o <= 0;
                elsif clk'event and clk = '1' then
                        if wr = 4 then
                                o <= i;
                        end if;
                end if;
         end process;
end reg_buff_04_arch;
```

รูปที่ ข.24 โปรแกรมวงจร reg_buff_04

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity regis_11 is
        port (
            clk : in STD_LOGIC;
            set : in STD_LOGIC;
            rd : in STD_LOGIC;
            a : in INTEGER range 0 to 2047;
            o : out INTEGER range 0 to 2047
            );
end regis_11;

architecture regis_11_arch of regis_11 is
begin
        process (clk, set, rd)
        begin
            if set = '1' then
                o <= 2047;
            elsif clk'event and clk = '1' then
                if rd = '1' then
                    o <= a;
                end if;
            end if;
        end process;
end regis_11_arch;
```

รูปที่ ข.25 โปรแกรมวงจร regis_11

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity ret is
    port (
        clk  : in  STD_LOGIC;
        ret  : in  STD_LOGIC;
        cona : in  STD_LOGIC;
        conb : in  STD_LOGIC;
        conc : in  STD_LOGIC;
        o    : out STD_LOGIC
        );
end ret;

architecture ret_arch of ret is
begin
    process (clk, ret, cona, conb, conc)
    begin
        if ret = '1' then
            o <= '1';
        elsif clk'event and clk = '1' then
            if ( cona or conb or conc ) = '1' then
                o <= '1';
            else o <= '0';
            end if;
```

```
        end  if;
    end  process;
end  ret_arch;
```

รูปที่ ข.26 โปรแกรมวงจร ret

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity serial is
        port (
            clk   :  in   STD_LOGIC;
            set   :  in   STD_LOGIC;
            cs    :  in   STD_LOGIC;
            inb   :  in   STD_LOGIC;
            inc   :  in   STD_LOGIC;
            ind   :  in   STD_LOGIC;
            ine   :  in   STD_LOGIC;
            inf   :  in   STD_LOGIC;
            ing   :  in   STD_LOGIC;
            shift :  out  STD_LOGIC
            );
end serial;

architecture serial_arch of serial is
        signal   b, c, d, e, f, g  :  STD_LOGIC;
begin
        process (clk, set, cs, inb, inc, ind, ine, inf, ing)
        begin
            if set = '1' then
                shift <= '0';
            elsif clk'event and clk = '1' then
```

```
            if  cs  =  '0'  then
                b  <=  inb;
                c  <=  inc;
                d  <=  ind;
                e  <=  ine;
                f  <=  inf;
                g  <=  ing;
            end  if;
            if  cs  =  '1'  then
                shift  <=  b;
                b  <=  c;
                c  <=  d;
                d  <=  e;
                e  <=  f;
                f  <=  g;
                g  <=  '0';
            end  if;
        end  if;
    end  process;
end  serial_arch;
```

รูปที่ ข.27 โปรแกรมวงจร serial

```
Library  IEEE;
use  IEEE.std_logic_1164.all;

entity  smulti  is
        port (
            ret  :  in  STD_LOGIC;
            cs   :  in  STD_LOGIC;
            a    :  in  STD_LOGIC;
            b    :  in  STD_LOGIC;
            s    :  out  STD_LOGIC
            );
end  smulti;

architecture  smulti_arch  of  smulti  is
begin
        process (ret, cs, a, b)
        begin
            if  ret = '1'  then
                s <= '0';
            elsif  cs = '0'  then
                s <= a;
            else  s <= b;
            end  if;
        end  process;
end  smulti_arch;
```

รูปที่ ข.28 โปรแกรมวงจร smulti

```
Library IEEE;
use IEEE.std_logic_1164.all;

entity buff is
        port (
            clk :   in  STD_LOGIC;
            clr :   in  STD_LOGIC;
            I   :   in  STD_LOGIC;
            o   :   out STD_LOGIC
            );
end buff;

architecture buff_arch of buff is
begin
        process (clk, clr)
        begin
            if clr = '0' then
                o <= '0';
            elsif clk'event and clk = '1' then
                o <= i;
            end if;
        end process;
end buff_arch;
```

รูปที่ ข.29 โปรแกรมวงจร buff

ภาคผนวก ค
รายละเอียดของอุปกรณ์

XILINX®

XC4000 Series
Field Programmable Gate Arrays

July 30, 1996 (Version 1.03)

Product Specification

XC4000-Series Features

Note: XC4000-Series devices described in this data sheet include the XC4000E, XC4000EX, XC4000L, and XC4000XL. This information does not apply to the older Xilinx families: XC4000, XC4000A, XC4000D or XC4000H. For information on these devices, see the Xilinx WEBLINX at http://www.xilinx.com.

- Third Generation Field-Programmable Gate Arrays
  - Select-RAM™ memory: on-chip ultra-fast RAM with
    - synchronous write option
    - dual-port RAM option
  - Fully PCI compliant (speed grades -3 and faster)
  - Abundant flip-flops
  - Flexible function generators
  - Dedicated high-speed carry logic
  - Wide edge decoders on each edge
  - Hierarchy of interconnect lines
  - Internal 3-state bus capability
  - 8 global low-skew clock or signal distribution networks
- System Performance to 66 MHz
- Flexible Array Architecture
- Systems-Oriented Features
  - IEEE 1149.1-compatible boundary scan logic support
  - Individually programmable output slew rate
  - Programmable input pull-up or pull-down resistors
  - 12-mA sink current per XC4000E output (4 mA per XC4000L output)
- Configured by Loading Binary File
  - Unlimited reprogrammability
- Readback Capability
- Backward Compatible with XC4000 Devices
- XACT*step* Development System runs on '386/'486/ Pentium-type PC, Sun-4, and Hewlett-Packard 700 series
  - Interfaces to popular design environments
  - Fully automatic mapping, placement and routing
  - Interactive design editor for design optimization
  - RAM/ROM compiler

Low-Voltage Versions Available

- Low-Voltage Devices Function at 3.0 - 3.6 Volts
- XC4000L: Low-Voltage Versions of XC4000E devices
- XC4000XL: Low-Voltage Versions of XC4000EX devices

Additional XC4000EX/XL Features

- Highest Capacity — Over 130,000 Usable Gates
- Additional Routing Over XC4000E
  - almost twice the routing capacity for high-density designs
- Buffered Interconnect for Maximum Speed
- New Latch Capability in Configurable Logic Blocks
- Improved VersaRing™ I/O Interconnect for Better Fixed Pinout Flexibility
- Flexible New High-Speed Clock Network
  - 8 additional Early Buffers for shorter clock delays
  - 4 additional FastCLK™ buffers for fastest clock input
  - Virtually unlimited number of clock signals
- Optional Multiplexer or 2-input Function Generator on Device Outputs
- High-Speed Parallel Express™ Configuration Mode
- Improved I/O Setup and Clock-to-Output with FastCLK and Global Early Buffers
- 4 Additional Address Bits in Master Parallel Configuration Mode

Introduction

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

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

The XC4000 Series currently has 19 members, as shown in Table 1.

Table 1: XC4000-Series Field Programmable Gate Arrays

Device	Max Logic Gates (No RAM)	Max. RAM Bits (No Logic)	Typical Gate Range (Logic and RAM)*	CLB Matrix	Total Logic Blocks	Number of Flip-Flops	Max. Decode Inputs per side	Max. User I/O
XC4003E	3,000	3,200	2,000 - 5,000	10 x 10	100	360	30	80
XC4005E/L	5,000	6,272	3,000 - 9,000	14 x 14	196	616	42	112
XC4006E	6,000	8,192	4,000 - 12,000	16 x 16	256	768	48	128
XC4008E	8,000	10,368	6,000 - 15,000	18 x 18	324	936	54	144
XC4010E/L	10,000	12,800	7,000 - 20,000	20 x 20	400	1,120	60	160
XC4013E/L	13,000	18,432	10,000 - 30,000	24 x 24	576	1,536	72	192
XC4020E	20,000	25,088	13,000 - 40,000	28 x 28	784	2,016	84	224
XC4025E	25,000	32,768	15,000 - 45,000	32 x 32	1,024	2,560	96	256
XC4028EX/XL	28,000	32,768	18,000 - 50,000	32 x 32	1,024	2,560	96	256
XC4036EX/XL	36,000	41,472	22,000 - 65,000	36 x 36	1,296	3,168	108	288
XC4044EX/XL	44,000	51,200	27,000 - 80,000	40 x 40	1,600	3,840	120	320
XC4052XL	52,000	61,952	33,000 - 100,000	44 x 44	1,936	4,576	132	352
XC4062XL	62,000	73,728	40,000 - 130,000	48 x 48	2,304	5,376	144	384
	Larger Devices Available in the First Half of 1997							

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

Note: *Throughout the functional descriptions in this document, references to the XC4000E device family include the XC4000L, and references to the XC4000EX device family include the XC4000XL, unless explicitly stated otherwise. References to the XC4000 Series include the XC4000E, XC4000EX, XC4000L, and XC4000XL families. All functionality in low-voltage families is the same as in the corresponding 5-Volt family, except where numerical references are made to timing, power, or current-sinking capability.*

Description

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

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

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

Because Xilinx FPGAs can be reprogrammed an unlimited number of times, they can be used in innovative designs where hardware is changed dynamically, or where hardware must be adapted to different user applications. FPGAs are ideal for shortening design and development cycles, and also offer a cost-effective solution for production rates well beyond 5,000 systems per month. For lowest high-volume unit cost, a design can first be implemented in the XC4000E or XC4000EX, then migrated to one of Xilinx' compatible HardWire mask-programmed devices.

Table 2 shows density and performance for a few common circuit functions that can be implemented in XC4000-Series devices.

Table 2: Density and Performance for Several Common Circuit Functions in XC4000E[1]

Design Class	Function	CLBs Used	XC4000E-3	XC4000E-2	Units
Memory	256 x 8 Single Port (read/modify/write)	72	63	80	MHz
	32 x 16 bit FIFO				
	simultaneous read/write	48	63	80	MHz
	MUXed read/write	32	63	80	MHz
Logic	9 bit Shift Register (with enable)	5	170	200	MHz
	16 bit Pre-Scaled Counter	8	142	170	MHz
	16 bit Loadable Counter	8	65	76	MHz
	16 bit Accumulator	9	65	76	MHz
	8 bit, 16 tap FIR Filter sample rate				
	parallel	400	55	65	MHz
	serial	68	8.1	10	MHz
	8 x 8 Parallel Multiplier				
	single stage, register to register	73	37	30	ns
	16 bit Address Decoder (internal decode)	3	4.7	3.9	ns
	9 bit Parity Checker	1	4.3	2.7	ns

Note: 1. Most functions are faster in XC4000EX due to faster carry logic, direct connects, and other additional interconnect.

Taking Advantage of Reconfiguration

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

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

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

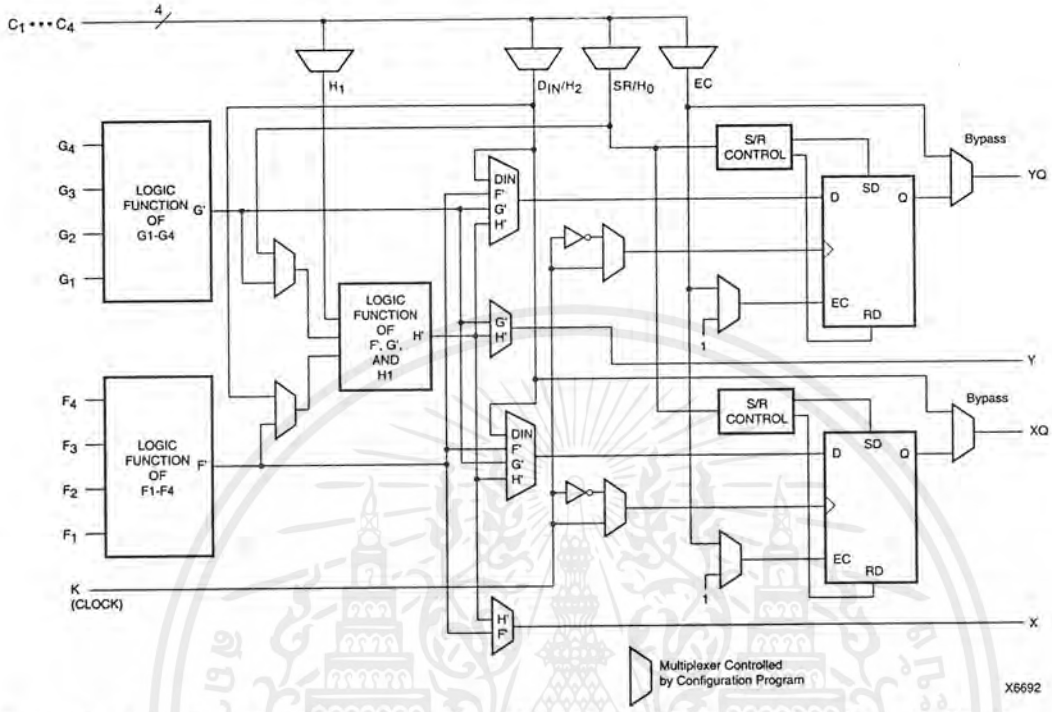


Figure 1: Simplified Block Diagram of XC4000-Series CLB (RAM and Carry Logic functions not shown)

Implementing wide functions in a single block reduces both the number of blocks required and the delay in the signal path, achieving both increased capacity and speed.

The versatility of the CLB function generators significantly improves system speed. In addition, the design-software tools can deal with each function generator independently. This flexibility improves cell usage.

Flip-Flops

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

The two edge-triggered D-type flip-flops have common clock (K) and clock enable (EC) inputs. Either or both clock inputs can also be permanently enabled. Storage element functionality is described in Table 4.

Latches (XC4000EX only)

The CLB storage elements can also be configured as latches. The two latches have common clock (K) and clock enable (EC) inputs. Storage element functionality is described in Table 4.

Table 4: CLB Storage Element Functionality (active rising edge is shown)

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

Legend:

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

Clock Input

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

Table 18: Pin Descriptions

Pin Name	I/O During Config.	I/O After Config.	Pin Description
Permanently Dedicated Pins			
VCC	I	I	Eight or more (depending on package) connections to the nominal +5 V supply voltage (+3.3 V for low-voltage devices). All must be connected, and each must be decoupled with a 0.01 - 0.1 µF capacitor to Ground.
GND	I	I	Eight or more (depending on package type) connections to Ground. All must be connected.
CCLK	I or O	I	During configuration, Configuration Clock (CCLK) is an output in Master modes or Asynchronous Peripheral mode, but is an input in Slave mode, Synchronous Peripheral mode, and Express mode. After configuration, CCLK has a weak pull-up resistor and can be selected as the Readback Clock. There is no CCLK High time restriction on XC4000-Series devices, except during Readback. See "Violating the Maximum High and Low Time Specification for the Readback Clock" on page 65 for an explanation of this exception.
DONE	I/O	O	DONE is a bidirectional signal with an optional internal pull-up resistor. As an output, it indicates the completion of the configuration process. As an input, a Low level on DONE can be configured to delay the global logic initialization and the enabling of outputs. The optional pull-up resistor is selected as an option in MakeBits, the XACT*step* program that creates the configuration bitstream. The resistor is included by default.
$\overline{\text{PROGRAM}}$	I	I	$\overline{\text{PROGRAM}}$ is an active Low input that forces the FPGA to clear its configuration memory. It is used to initiate a configuration cycle. When $\overline{\text{PROGRAM}}$ goes High, the FPGA finishes the current clear cycle and executes another complete clear cycle, before it goes into a WAIT state and releases $\overline{\text{INIT}}$. The $\overline{\text{PROGRAM}}$ pin has a permanent weak pull-up, so it need not be externally pulled up to Vcc.
User I/O Pins That Can Have Special Functions			
RDY/$\overline{\text{BUSY}}$	O	I/O	During Peripheral mode configuration, this pin indicates when it is appropriate to write another byte of data into the FPGA. The same status is also available on D7 in Asynchronous Peripheral mode, if a read operation is performed when the device is selected. After configuration, RDY/$\overline{\text{BUSY}}$ is a user-programmable I/O pin. RDY/$\overline{\text{BUSY}}$ is pulled High with a high-impedance pull-up prior to $\overline{\text{INIT}}$ going High.
$\overline{\text{RCLK}}$	O	I/O	During Master Parallel configuration, each change on the A0-A17 outputs (A0 - A21 for XC4000EX) is preceded by a rising edge on $\overline{\text{RCLK}}$, a redundant output signal. $\overline{\text{RCLK}}$ is useful for clocked PROMs. It is rarely used during configuration. After configuration, $\overline{\text{RCLK}}$ is a user-programmable I/O pin.
M0, M1, M2	I	I (M0), O (M1), I (M2)	As Mode inputs, these pins are sampled after $\overline{\text{INIT}}$ goes High to determine the configuration mode to be used. After configuration, M0 and M2 can be used as inputs, and M1 can be used as a 3-state output. These three pins have no associated input or output registers. During configuration, these pins have weak pull-up resistors. For the most popular configuration mode, Slave Serial, the mode pins can thus be left unconnected. The three mode inputs can be individually configured with or without weak pull-up or pull-down resistors. A pull-down resistor value of 4.7 kΩ is recommended. These pins can only be used as inputs or outputs when called out by special schematic definitions. To use these pins, place the library components MD0, MD1, and MD2 instead of the usual pad symbols. Input or output buffers must still be used.

Table 18: Pin Descriptions (Continued)

Pin Name	I/O During Config.	I/O After Config.	Pin Description
TDO	O	O	If boundary scan is used, this pin is the Test Data Output. If boundary scan is not used, this pin is a 3-state output without a register, after configuration is completed. This pin can be user output only when called out by special schematic definitions. To use this pin, place the library component TDO instead of the usual pad symbol. An output buffer must still be used.
TDI, TCK, TMS	I	I/O or I (JTAG)	If boundary scan is used, these pins are Test Data In, Test Clock, and Test Mode Select inputs respectively. They come directly from the pads, bypassing the IOBs. These pins can also be used as inputs to the CLB logic after configuration is completed. If the BSCAN symbol is not placed in the design, all boundary scan functions are inhibited once configuration is completed, and these pins become user-programmable I/O. In this case, they must be called out by special schematic definitions. To use these pins, place the library components TDI, TCK, and TMS instead of the usual pad symbols. Input or output buffers must still be used.
HDC	O	I/O	High During Configuration (HDC) is driven High until the I/O go active. It is available as a control output indicating that configuration is not yet completed. After configuration, HDC is a user-programmable I/O pin.
$\overline{\text{LDC}}$	O	I/O	Low During Configuration ($\overline{\text{LDC}}$) is driven Low until the I/O go active. It is available as a control output indicating that configuration is not yet completed. After configuration, $\overline{\text{LDC}}$ is a user-programmable I/O pin.
$\overline{\text{INIT}}$	I/O	I/O	Before and during configuration, $\overline{\text{INIT}}$ is a bidirectional signal. A 1 kΩ - 10 kΩ external pull-up resistor is recommended. As an active-Low open-drain output, $\overline{\text{INIT}}$ is held Low during the power stabilization and internal clearing of the configuration memory. As an active-Low input, it can be used to hold the FPGA in the internal WAIT state before the start of configuration. Master mode devices stay in a WAIT state an additional 30 to 300 µs after $\overline{\text{INIT}}$ has gone High. During configuration, a Low on this output indicates that a configuration data error has occurred. After the I/O go active, $\overline{\text{INIT}}$ is a user-programmable I/O pin.
PGCK1 - PGCK4 (XC4000E only)	Weak Pull-up	I or I/O	Four Primary Global inputs each drive a dedicated internal global net with short delay and minimal skew. If not used to drive a global buffer, any of these pins is a user-programmable I/O. The PGCK1-PGCK4 pins drive the four Primary Global Buffers. Any input pad symbol connected directly to the input of a BUFGP symbol is automatically placed on one of these pins.
SGCK1 - SGCK4 (XC4000E only)	Weak Pull-up	I or I/O	Four Secondary Global inputs each drive a dedicated internal global net with short delay and minimal skew. These internal global nets can also be driven from internal logic. If not used to drive a global net, any of these pins is a user-programmable I/O pin. The SGCK1-SGCK4 pins provide the shortest path to the four Secondary Global Buffers. Any input pad symbol connected directly to the input of a BUFGS symbol is automatically placed on one of these pins.
GCK1 - GCK8 (XC4000EX only)	Weak Pull-up	I or I/O	Eight inputs can each drive a Global Low-Skew buffer. In addition, each can drive a Global Early buffer. Each pair of global buffers can also be driven from internal logic, but must share an input signal. If not used to drive a global buffer, any of these pins is a user-programmable I/O. Any input pad symbol connected directly to the input of a BUFGLS or BUFGE symbol is automatically placed on one of these pins.
FCLK1 - FCLK4 (XC4000EX only)	Weak Pull-up	I or I/O	Four FCLK inputs can each drive a FastCLK buffer. The FastCLK buffers cannot be driven from internal logic. If not used to drive a global buffer, any of these pins is a user-programmable I/O. Any input pad symbol connected directly to the input of a BUFFCLK symbol is automatically placed on one of these pins.

Table 18: Pin Descriptions (Continued)

Pin Name	I/O During Config.	I/O After Config.	Pin Description
$\overline{CS0}$, CS1, $\overline{WS}$, $\overline{RS}$	I	I/O	These four inputs are used in Asynchronous Peripheral mode. The chip is selected when $\overline{CS0}$ is Low and CS1 is High. While the chip is selected, a Low on Write Strobe ($\overline{WS}$) loads the data present on the D0 - D7 inputs into the internal data buffer. A Low on Read Strobe ($\overline{RS}$) changes D7 into a status output — High if Ready, Low if Busy — and drives D0 - D6 High. In Express mode, CS1 is used as a serial-enable signal for daisy-chaining. $\overline{WS}$ and $\overline{RS}$ should be mutually exclusive, but if both are Low simultaneously, the Write Strobe overrides. After configuration, these are user-programmable I/O pins.
A0 - A17	O	I/O	During Master Parallel configuration, these 18 output pins address the configuration EPROM. After configuration, they are user-programmable I/O pins.
A18 - A21 (XC4000EX only)	O	I/O	During Master Parallel configuration with an XC4000EX master, these 4 output pins add 4 more bits to address the configuration EPROM. After configuration, they are user-programmable I/O pins.
D0 - D7	I	I/O	During Master Parallel and Peripheral configuration, these eight input pins receive configuration data. After configuration, they are user-programmable I/O pins.
DIN	I	I/O	During Slave Serial or Master Serial configuration, DIN is the serial configuration data input receiving data on the rising edge of CCLK. During Parallel configuration, DIN is the D0 input. After configuration, DIN is a user-programmable I/O pin.
DOUT	O	I/O	During configuration in any mode but Express mode, DOUT is the serial configuration data output that can drive the DIN of daisy-chained slave FPGAs. DOUT data changes on the falling edge of CCLK, one-and-a-half CCLK periods after it was received at the DIN input. In Express mode, DOUT is the status output that can drive the CS1 of daisy-chained FPGAs, to enable and disable downstream devices. After configuration, DOUT is a user-programmable I/O pin.
Unrestricted User-Programmable I/O Pins			
I/O	Weak Pull-up	I/O	These pins can be configured to be input and/or output after configuration is completed. Before configuration is completed, these pins have an internal high-value pull-up resistor (50 kΩ - 100 kΩ) that defines the logic level as High.

Table 22: XC4000E Program Data

Device	XC4003E	XC4005E/L	XC4006E	XC4008E	XC4010E/L	XC4013E/L	XC4020E	XC4025E
Max Logic Gates	3,000	5,000	6,000	8,000	10,000	13,000	20,000	25,000
CLBs (Row x Col.)	100 (10 x 10)	196 (14 x 14)	256 (16 x 16)	324 (18 x 18)	400 (20 x 20)	576 (24 x 24)	784 (28 x 28)	1,024 (32 x 32)
IOBs	80	112	128	144	160	192	224	256
Flip-Flops	360	616	768	936	1,120	1,536	2,016	2,560
Horizontal Longlines	20	28	32	36	40	48	56	64
TBUFs per Longline	12	16	18	20	22	26	30	34
Bits per Frame	126	166	186	206	226	266	306	346
Frames	428	572	644	716	788	932	1,076	1,220
Program Data	53,936	94,960	119,792	147,504	178,096	247,920	329,264	422,128
PROM Size (bits)	53,984	95,008	119,840	147,552	178,144	247,968	329,312	422,176

Notes: 1. Bits per Frame = (10 x number of rows) + 7 for the top + 13 for the bottom + 1 + 1 start bit + 4 error check bits
Number of Frames = (36 x number of columns) + 26 for the left edge + 41 for the right edge + 1
Program Data = (Bits per Frame x Number of Frames) + 8 postamble bits
PROM Size = Program Data + 40

2. The user can add more "one" bits as leading dummy bits in the header, or, if CRC = off, as trailing dummy bits at the end of any frame, following the four error check bits. However, the Length Count value **must** be adjusted for all such extra "one" bits, even for extra leading ones at the beginning of the header.

The MakeBits software creates the configuration bitstream. In Express mode, only non-CRC error checking is supported. In all other modes, MakeBits allows a selection of CRC or non-CRC error checking. The non-CRC error checking tests for a designated end-of-frame field for each frame. For CRC error checking, MakeBits calculates a running CRC and inserts a unique four-bit partial check at the end of each frame. The 11-bit CRC check of the last frame of an FPGA includes the last seven data bits.

Detection of an error results in the suspension of data loading and the pulling down of the $\overline{\text{INIT}}$ pin. In Master modes, CCLK and address signals continue to operate externally. The user must detect $\overline{\text{INIT}}$ and initialize a new configuration by pulsing the $\overline{\text{PROGRAM}}$ pin Low or cycling Vcc.

Cyclic Redundancy Check (CRC) for Configuration and Readback

The Cyclic Redundancy Check is a method of error detection in data transmission applications. Generally, the transmitting system performs a calculation on the serial bitstream. The result of this calculation is tagged onto the data stream as additional check bits. The receiving system performs an identical calculation on the bitstream and compares the result with the received checksum.

Each data frame of the configuration bitstream has four error bits at the end, as shown in Table 21. If a frame data error is detected during the loading of the FPGA, the configuration process with a potentially corrupted bitstream is terminated. The FPGA pulls the $\overline{\text{INIT}}$ pin Low and goes into a Wait state.

During Readback, 11 bits of the 16-bit checksum are added to the end of the Readback data stream. The checksum is computed using the CRC-16 CCITT polynomial, as shown in Figure 47. The checksum consists of the 11 most significant bits of the 16-bit code. A change in the checksum indicates a change in the Readback bitstream. A comparison to a previous checksum is meaningful only if the readback data is independent of the current device state. CLB outputs should not be included (Read Capture MakeBits option not used), and if RAM is present, the RAM content must be unchanged.

Statistically, one error out of 2048 might go undetected.

Table 23: XC4000EX Program Data

Device	XC4028EX/XL	XC4036EX/XL	XC4044EX/XL	XC4052XL	XC4062XL
Max Logic Gates	28,000	36,000	44,000	52,000	62,000
CLBs (Row x Col.)	1,024 (32 x 32)	1,296 (36 x 36)	1,600 (40 x 40)	1,936 (44 x 44)	2,304 (48 x 48)
IOBs	256	288	320	352	384
Flip-Flops	2,560	3,168	3,840	4,576	5,376
Horizontal Longlines	192	216	240	264	288
TBUFs per Longline	34	38	42	46	50
Bits per Frame	421	469	517	565	613
Frames	1587	1775	1963	2151	2,339
Program Data	668,127	832,483	1,014,879	1,215,323	1,433,807
PROM Size (bits)	668,167	832,523	1,014,919	1,215,363	1,433,847

Notes: 1. Bits per Frame = (12 x number of rows) + 8 for the top + 16 for the bottom + 8 + 1 start bit + 4 error check bits
Number of Frames = (47 x number of columns) + 27 for the left edge + 52 for the right edge + 4
Program Data = (Bits per Frame x Number of Frames) + 8 postamble bits
PROM Size = Program Data + 40

2. The user can add more "one" bits as leading dummy bits in the header, or, if CRC = off, as trailing dummy bits at the end of any frame, following the four error check bits. However, the Length Count value **must** be adjusted for all such extra "one" bits, even for extra leading ones at the beginning of the header.
3. Express mode bitfiles are slightly larger (see Table 21).

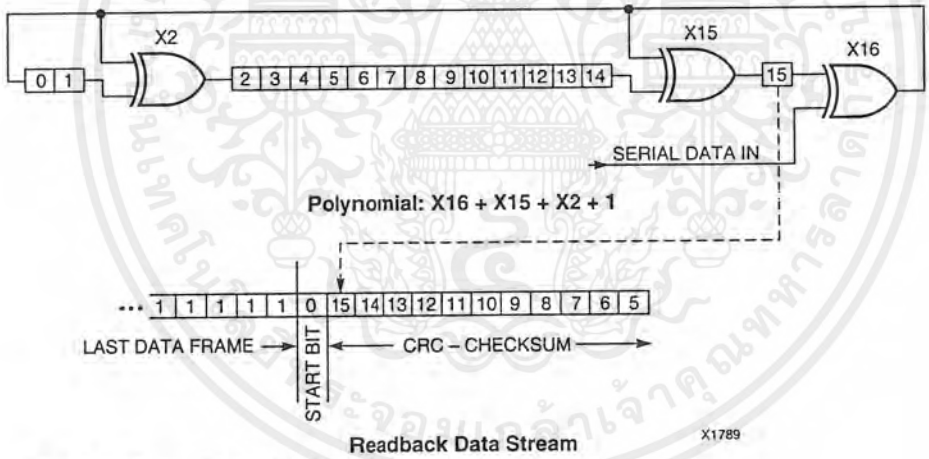


Figure 47: Circuit for Generating CRC-16

Synchronous Peripheral Mode

Synchronous Peripheral mode can also be considered Slave Parallel mode. An external signal drives the CCLK input(s) of the FPGA(s). The first byte of parallel configuration data must be available at the Data inputs of the lead FPGA a short setup time before the rising CCLK edge. Subsequent data bytes are clocked in on every eighth consecutive rising CCLK edge.

The same CCLK edge that accepts data, also causes the RDY/$\overline{\text{BUSY}}$ output to go High for one CCLK period. The pin name is a misnomer. In Synchronous Peripheral mode it is really an ACKNOWLEDGE signal. Synchronous operation does not require this response, but it is a meaningful signal for test purposes. Note that RDY/$\overline{\text{BUSY}}$ is pulled High with a high-impedance pullup prior to $\overline{\text{INIT}}$ going High.

The lead FPGA serializes the data and presents the preamble data (and all data that overflows the lead device) on its DOUT pin. There is an internal delay of 1.5 CCLK periods, which means that DOUT changes on the falling CCLK edge, and the next FPGA in the daisy chain accepts data on the subsequent rising CCLK edge.

In order to complete the serial shift operation, 10 additional CCLK rising edges are required after the last data byte has been loaded, plus one more CCLK cycle for each daisy-chained device.

Synchronous Peripheral mode is selected by a <011> on the mode pins (M2, M1, M0).

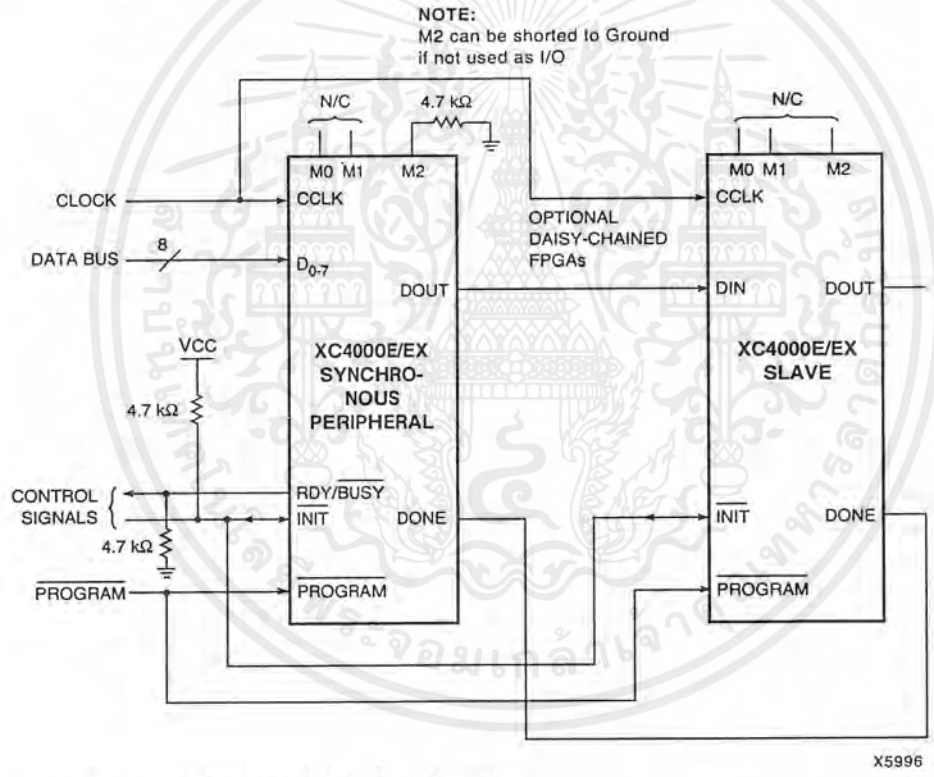


Figure 59: Synchronous Peripheral Mode Circuit Diagram

Device-Specific Pinout Tables

Pin Locations for XC4003E Devices

XC4003E Pad Name	PC 84	PQ 100	VQ 100	PG 120	Bndry Scan
VCC	P2	P92	P89	G3	-
I/O (A8)	P3	P93	P90	G1	32
I/O (A9)	P4	P94	P91	F1	35
I/O	-	P95	P92	E1	38
I/O	-	P96	P93	F2	41
I/O (A10)	P5	P97	P94	F3	44
I/O (A11)	P6	P98	P95	D1	47
I/O (A12)	P7	P99	P96	C1	50
I/O (A13)	P8	P100	P97	D2	53
I/O (A14)	P9	P1	P98	C2	56
I/O, SGCK1 (A15)	P10	P2	P99	D3	59
VCC	P11	P3	P100	C3	-
GND	P12	P4	P1	C4	-
I/O, PGCK1 (A16)	P13	P5	P2	B2	62
I/O (A17)	P14	P6	P3	B3	65
I/O, TDI	P15	P7	P4	C5	68
I/O, TCK	P16	P8	P5	B4	71
I/O, TMS	P17	P9	P6	B5	74
I/O	P18	P10	P7	A4	77
I/O	-	-	-	C6	80
I/O	-	P11	P8	A5	83
I/O	P19	P12	P9	B6	86
I/O	P20	P13	P10	A6	89
GND	P21	P14	P11	B7	-
VCC	P22	P15	P12	C7	-
I/O	P23	P16	P13	A7	92
I/O	P24	P17	P14	A8	95
I/O	-	P18	P15	A9	98
I/O	-	-	-	B8	101
I/O	P25	P19	P16	C8	104
I/O	P26	P20	P17	A10	107
I/O	P27	P21	P18	B9	110
I/O	-	P22	P19	A11	113
I/O	P28	P23	P20	C9	116
I/O, SCGK2	P29	P24	P21	A12	119
O (M1)	P30	P25	P22	B11	122
GND	P31	P26	P23	C10	-
I (M0)	P32	P27	P24	C11	125
VCC	P33	P28	P25	D11	-
I (M2)	P34	P29	P26	B12	126
I/O, PGCK2	P35	P30	P27	C12	127
I/O (HDC)	P36	P31	P28	A13	130
I/O	-	P32	P29	D12	133
I/O (LDC)	P37	P33	P30	C13	136
I/O	P38	P34	P31	E12	139
I/O	P39	P35	P32	D13	142
I/O	-	P36	P33	F11	145
I/O	-	P37	P34	E13	148
I/O	P40	P38	P35	F12	151
I/O (INIT)	P41	P39	P36	F13	154
VCC	P42	P40	P37	G12	-
GND	P43	P41	P38	G11	-
I/O	P44	P42	P39	G13	157
I/O	P45	P43	P40	H13	160
I/O	-	P44	P41	J13	163
I/O	-	P45	P42	H12	166
I/O	P46	P46	P43	H11	169
I/O	P47	P47	P44	K13	172
I/O	P48	P48	P45	J12	175
I/O	P49	P49	P46	L13	178
I/O	P50	P50	P47	M13	181
I/O, SGCK3	P51	P51	P48	L12	184
GND	P52	P52	P49	K11	-
DONE	P53	P53	P50	L11	-
VCC	P54	P54	P51	L10	-
PROGRAM	P55	P55	P52	M12	-
I/O (D7)	P56	P56	P53	M11	187
I/O, PGCK3	P57	P57	P54	N13	190
I/O (D6)	P58	P58	P55	M10	193
I/O	-	P59	P56	N11	196
I/O (D5)	P59	P60	P57	M9	199
I/O (CS0)	P60	P61	P58	N10	202
I/O	-	P62	P59	L8	205
I/O	-	P63	P60	N9	208
I/O (D4)	P61	P64	P61	M8	211
I/O	P62	P65	P62	N8	214
VCC	P63	P66	P63	M7	-
GND	P64	P67	P64	L7	-
I/O (D3)	P65	P68	P65	N7	217
I/O (RS)	P66	P69	P66	N6	220
I/O	-	P70	P67	N5	223
I/O	-	-	-	M6	226
I/O (D2)	P67	P71	P68	L6	229
I/O	P68	P72	P69	N4	232
I/O (D1)	P69	P73	P70	M5	235
I/O (RCLK, RDY/BUSY)	P70	P74	P71	N3	238
I/O (D0, DIN)	P71	P75	P72	N2	241
I/O, SGCK4 (DOUT)	P72	P76	P73	M3	244
CCLK	P73	P77	P74	L4	-
VCC	P74	P78	P75	L3	-
O, TDO	P75	P79	P76	M2	0
GND	P76	P80	P77	K3	-
I/O (A0, WS)	P77	P81	P78	L2	2
I/O, PGCK4 (A1)	P78	P82	P79	N1	5
I/O (CS1, A2)	P79	P83	P80	K2	8
I/O (A3)	P80	P84	P81	L1	11
I/O (A4)	P81	P85	P82	J2	14
I/O (A5)	P82	P86	P83	K1	17
I/O	-	P87	P84	H3	20
I/O	-	P88	P85	J1	23
I/O (A6)	P83	P89	P86	H2	26
I/O (A7)	P84	P90	P87	H1	29
GND	P1	P91	P88	G2	-

4/2/96

Additional No Connect (N.C.) Connections on PG120 Package

PG120	PG120	PG120	PG120
A1	B13	J11	M4
A2	E2	K12	N12
A3	E3	L5	
B1	E11	L9	
B10	J3	M1	

2/28/96

Pin Locations for XC4005E/L Devices

XC4005 E/L Pad Name	PC 84	PQ 100	TQ 144	PG 156	PQ 160	PQ 208	Bndry Scan
VCC	P2	P92	P128	H3	P142	P183	-
I/O (A8)	P3	P93	P129	H1	P143	P184	44
I/O (A9)	P4	P94	P130	G1	P144	P185	47
I/O	-	P95	P131	G2	P145	P186	50
I/O	-	P96	P132	G3	P146	P187	53
I/O (A10)	P5	P97	P133	F1	P147	P190	56
I/O (A11)	P6	P98	P134	F2	P148	P191	59
I/O	-	-	P135	E1	P149	P192	62
I/O	-	-	P136	E2	P150	P193	65
GND	-	-	P137	F3	P151	P194	-
I/O (A12)	P7	P99	P138	E3	P154	P199	68
I/O (A13)	P8	P100	P139	C1	P155	P200	71
I/O	-	-	P140	C2	P156	P201	74
I/O	-	-	P141	D3	P157	P202	77
I/O (A14)	P9	P1	P142	B1	P158	P203	80
I/O, SGCK1 (A15)	P10	P2	P143	B2	P159	P204	83
VCC	P11	P3	P144	C3	P160	P205	-
GND	P12	P4	P1	C4	P1	P2	-
I/O, PGCK1 (A16)	P13	P5	P2	B3	P2	P4	86
I/O (A17)	P14	P6	P3	A1	P3	P5	89
I/O	-	-	P4	A2	P4	P6	92
I/O	-	-	P5	C5	P5	P7	95
I/O, TDI	P15	P7	P6	B4	P6	P8	98
I/O, TCK	P16	P8	P7	A3	P7	P9	101
GND	-	-	P8	C6	P10	P14	-
I/O	-	-	P9	B5	P11	P15	104
I/O	-	-	P10	B6	P12	P16	107
I/O, TMS	P17	P9	P11	A5	P13	P17	110
I/O	P18	P10	P12	C7	P14	P18	113
I/O	-	-	P13	B7	P15	P21	116
I/O	-	P11	P14	A6	P16	P22	119
I/O	P19	P12	P15	A7	P17	P23	122
I/O	P20	P13	P16	A8	P18	P24	125
GND	P21	P14	P17	C8	P19	P25	-
VCC	P22	P15	P18	B8	P20	P26	-
I/O	P23	P16	P19	C9	P21	P27	128
I/O	P24	P17	P20	B9	P22	P28	131
I/O	-	P18	P21	A9	P23	P29	134
I/O	-	-	P22	B10	P24	P30	137
I/O	P25	P19	P23	C10	P25	P33	140
I/O	P26	P20	P24	A10	P26	P34	143
I/O	-	-	P25	A11	P27	P35	146
I/O	-	-	P26	B11	P28	P36	149

XC4005 E/L Pad Name	PC 84	PQ 100	TQ 144	PG 156	PQ 160	PQ 208	Bndry Scan
GND	-	-	P27	C11	P29	P37	-
I/O	P27	P21	P28	B12	P32	P42	152
I/O	-	P22	P29	A13	P33	P43	155
I/O	-	-	P30	A14	P34	P44	158
I/O	-	-	P31	C12	P35	P45	161
I/O	P28	P23	P32	B13	P36	P46	164
I/O, SCGK2	P29	P24	P33	B14	P37	P47	167
O (M1)	P30	P25	P34	A15	P38	P48	170
GND	P31	P26	P35	C13	P39	P49	-
I (M0)	P32	P27	P36	A16	P40	P50	173
VCC	P33	P28	P37	C14	P41	P55	-
I (M2)	P34	P29	P38	B15	P42	P56	174
I/O, PGCK2	P35	P30	P39	B16	P43	P57	175
I/O (HDC)	P36	P31	P40	D14	P44	P58	178
I/O	-	-	P41	C15	P45	P59	181
I/O	-	-	P42	D15	P46	P60	184
I/O	-	P32	P43	E14	P47	P61	187
I/O (LDC)	P37	P33	P44	C16	P48	P62	190
GND	-	-	P45	F14	P51	P67	-
I/O	-	-	P46	F15	P52	P68	193
I/O	-	-	P47	E16	P53	P69	196
I/O	P38	P34	P48	F16	P54	P70	199
I/O	P39	P35	P49	G14	P55	P71	202
I/O	-	P36	P50	G15	P56	P74	205
I/O	-	P37	P51	G16	P57	P75	208
I/O	P40	P38	P52	H16	P58	P76	211
I/O (INIT)	P41	P39	P53	H15	P59	P77	214
VCC	P42	P40	P54	H14	P60	P78	-
GND	P43	P41	P55	J14	P61	P79	-
I/O	P44	P42	P56	J15	P62	P80	217
I/O	P45	P43	P57	J16	P63	P81	220
I/O	-	P44	P58	K16	P64	P82	223
I/O	-	P45	P59	K15	P65	P83	226
I/O	P46	P46	P60	K14	P66	P86	229
I/O	P47	P47	P61	L16	P67	P87	232
I/O	-	-	P62	M16	P68	P88	235
I/O	-	-	P63	L15	P69	P89	238
GND	-	-	P64	L14	P70	P90	-
I/O	P48	P48	P65	P16	P73	P95	241
I/O	P49	P49	P66	M14	P74	P96	244
I/O	-	-	P67	N15	P75	P97	247
I/O	-	-	P68	P15	P76	P98	250
I/O	P50	P50	P69	N14	P77	P99	253
I/O, SGCK3	P51	P51	P70	R16	P78	P100	256
GND	P52	P52	P71	P14	P79	P101	-

XC4005 E/L Pad Name	PC 84	PQ 100	TQ 144	PG 156	PQ 160	PQ 208	Bndry Scan
DONE	P53	P53	P72	R15	P80	P103	-
VCC	P54	P54	P73	P13	P81	P106	-
$\overline{\text{PRO-GRAM}}$	P55	P55	P74	R14	P82	P108	-
I/O (D7)	P56	P56	P75	T16	P83	P109	259
I/O, PGCK3	P57	P57	P76	T15	P84	P110	262
I/O	-	-	P77	R13	P85	P111	265
I/O	-	-	P78	P12	P86	P112	268
I/O (D6)	P58	P58	P79	T14	P87	P113	271
I/O	-	P59	P80	T13	P88	P114	274
GND	-	-	P81	P11	P91	P119	-
I/O	-	-	P82	R11	P92	P120	277
I/O	-	-	P83	T11	P93	P121	280
I/O (D5)	P59	P60	P84	T10	P94	P122	283
I/O ($\overline{\text{CS0}}$)	P60	P61	P85	P10	P95	P123	286
I/O	-	P62	P86	R10	P96	P126	289
I/O	-	P63	P87	T9	P97	P127	292
I/O (D4)	P61	P64	P88	R9	P98	P128	295
I/O	P62	P65	P89	P9	P99	P129	298
VCC	P63	P66	P90	R8	P100	P130	-
GND	P64	P67	P91	P8	P101	P131	-
I/O (D3)	P65	P68	P92	T8	P102	P132	301
I/O ($\overline{\text{RS}}$)	P66	P69	P93	T7	P103	P133	304
I/O	-	P70	P94	T6	P104	P134	307
I/O	-	-	P95	R7	P105	P135	310
I/O (D2)	P67	P71	P96	P7	P106	P138	313
I/O	P68	P72	P97	T5	P107	P139	316
I/O	-	-	P98	R6	P108	P140	319
I/O	-	-	P99	T4	P109	P141	322
GND	-	-	P100	P6	P110	P142	-
I/O (D1)	P69	P73	P101	T3	P113	P147	325
I/O ($\overline{\text{RCLK}}$, RDY/ $\overline{\text{BUSY}}$)	P70	P74	P102	P5	P114	P148	328
I/O	-	-	P103	R4	P115	P149	331
I/O	-	-	P104	R3	P116	P150	334
I/O (D0, DIN)	P71	P75	P105	P4	P117	P151	337
I/O, SGCK4 (DOUT)	P72	P76	P106	T2	P118	P152	340
CCLK	P73	P77	P107	R2	P119	P153	-
VCC	P74	P78	P108	P3	P120	P154	-
O, TDO	P75	P79	P109	T1	P121	P159	0
GND	P76	P80	P110	N3	P122	P160	-
I/O (A0, $\overline{\text{WS}}$)	P77	P81	P111	R1	P123	P161	2
I/O, PGCK4 (A1)	P78	P82	P112	P2	P124	P162	5
I/O	-	-	P113	N2	P125	P163	8
I/O	-	-	P114	M3	P126	P164	11
I/O (CS1, A2)	P79	P83	P115	P1	P127	P165	14
I/O (A3)	P80	P84	P116	N1	P128	P166	17
GND	-	-	P118	L3	P131	P171	-
I/O	-	-	P119	L2	P132	P172	20
I/O	-	-	P120	L1	P133	P173	23
I/O (A4)	P81	P85	P121	K3	P134	P174	26
I/O (A5)	P82	P86	P122	K2	P135	P175	29
I/O	-	P87	P123	K1	P137	P178	32
I/O	-	P88	P124	J1	P138	P179	35
I/O (A6)	P83	P89	P125	J2	P139	P180	38
I/O (A7)	P84	P90	P126	J3	P140	P181	41
GND	P1	P91	P127	H2	P141	P182	-

4/2/96

Additional No Connect (N.C.) Connections on TQ144, PG156, PQ160 & PQ208 Packages

TQ144	PG156	PQ160	PQ208
P117	A4	P8	P1
	A12	P9	P3
	D1	P30	P10-P13
	D2	P31	P19-P20
	D16	P49	P31-P32
	E15	P50	P38-P41
	M1	P71	P51-P54
	M2	P72	P63-P66
	M15	P89	P72-P73
	N16	P90	P84-P85
	R5	P111	P91-P94
	R12	P112	P102
	T12	P129	P104-P105
		P130	P107
		P136	P115-P118
		P152	P124-P125
		P153	P136-P137
			P143-P146
			P155-P158
			P167-P170
			P176-P177
			P188-P189
			P195-P198
			P206-P208

3/12/96

Pin Locations for XC4006E Devices

XC4006E Pad Name	PC 84	TQ 144	PG 156	PQ 160	PQ 208	Bndry Scan
VCC	P2	P128	H3	P142	P183	-
I/O (A8)	P3	P129	H1	P143	P184	50
I/O (A9)	P4	P130	G1	P144	P185	53
I/O	-	P131	G2	P145	P186	56
I/O	-	P132	G3	P146	P187	59
I/O (A10)	P5	P133	F1	P147	P190	62
I/O (A11)	P6	P134	F2	P148	P191	65
I/O	-	P135	E1	P149	P192	68
I/O	-	P136	E2	P150	P193	71
GND	-	P137	F3	P151	P194	-
I/O	-	-	D1	P152	P197	74
I/O	-	-	D2	P153	P198	77
I/O (A12)	P7	P138	E3	P154	P199	80
I/O (A13)	P8	P139	C1	P155	P200	83
I/O	-	P140	C2	P156	P201	86
I/O	-	P141	D3	P157	P202	89
I/O (A14)	P9	P142	B1	P158	P203	92
I/O, SGCK1 (A15)	P10	P143	B2	P159	P204	95
VCC	P11	P144	C3	P160	P205	-
GND	P12	P1	C4	P1	P2	-
I/O, PGCK1 (A16)	P13	P2	B3	P2	P4	98
I/O (A17)	P14	P3	A1	P3	P5	101
I/O	-	P4	A2	P4	P6	104
I/O	-	P5	C5	P5	P7	107
I/O, TDI	P15	P6	B4	P6	P8	110
I/O, TCK	P16	P7	A3	P7	P9	113
I/O	-	-	A4	P8	P10	116
I/O	-	-	-	P9	P11	119
GND	-	P8	C6	P10	P14	-
I/O	-	P9	B5	P11	P15	122
I/O	-	P10	B6	P12	P16	125
I/O, TMS	P17	P11	A5	P13	P17	128
I/O	P18	P12	C7	P14	P18	131
I/O	-	P13	B7	P15	P21	134
I/O	-	P14	A6	P16	P22	137
I/O	P19	P15	A7	P17	P23	140
I/O	P20	P16	A8	P18	P24	143
GND	P21	P17	C8	P19	P25	-
VCC	P22	P18	B8	P20	P26	-
I/O	P23	P19	C9	P21	P27	146
I/O	P24	P20	B9	P22	P28	149
I/O	-	P21	A9	P23	P29	152
I/O	-	P22	B10	P24	P30	155
I/O	P25	P23	C10	P25	P33	158
I/O	P26	P24	A10	P26	P34	161
I/O	-	P25	A11	P27	P35	164
I/O	-	P26	B11	P28	P36	167
GND	-	P27	C11	P29	P37	-
I/O	-	-	A12	P30	P40	170
I/O	-	-	-	P31	P41	173
I/O	P27	P28	B12	P32	P42	176
I/O	-	P29	A13	P33	P43	179
I/O	-	P30	A14	P34	P44	182
I/O	-	P31	C12	P35	P45	185
I/O	P28	P32	B13	P36	P46	188
I/O, SCGK2	P29	P33	B14	P37	P47	191
O (M1)	P30	P34	A15	P38	P48	194
GND	P31	P35	C13	P39	P49	-
I (M0)	P32	P36	A16	P40	P50	197
VCC	P33	P37	C14	P41	P55	-
I (M2)	P34	P38	B15	P42	P56	198
I/O, PGCK2	P35	P39	B16	P43	P57	199
I/O (HDC)	P36	P40	D14	P44	P58	202
I/O	-	P41	C15	P45	P59	205
I/O	-	P42	D15	P46	P60	208
I/O	-	P43	E14	P47	P61	211
I/O ($\overline{\text{LDC}}$)	P37	P44	C16	P48	P62	214
I/O	-	-	E15	P49	P63	217
I/O	-	-	D16	P50	P64	220
GND	-	P45	F14	P51	P67	-
I/O	-	P46	F15	P52	P68	223
I/O	-	P47	E16	P53	P69	226
I/O	P38	P48	F16	P54	P70	229
I/O	P39	P49	G14	P55	P71	232
I/O	-	P50	G15	P56	P74	235
I/O	-	P51	G16	P57	P75	238
I/O	P40	P52	H16	P58	P76	241
I/O ($\overline{\text{INIT}}$)	P41	P53	H15	P59	P77	244
VCC	P42	P54	H14	P60	P78	-
GND	P43	P55	J14	P61	P79	-
I/O	P44	P56	J15	P62	P80	247
I/O	P45	P57	J16	P63	P81	250
I/O	-	P58	K16	P64	P82	253
I/O	-	P59	K15	P65	P83	256
I/O	P46	P60	K14	P66	P86	259
I/O	P47	P61	L16	P67	P87	262
I/O	-	P62	M16	P68	P88	265
I/O	-	P63	L15	P69	P89	268
GND	-	P64	L14	P70	P90	-
I/O	-	-	N16	P71	P93	271
I/O	-	-	M15	P72	P94	274
I/O	P48	P65	P16	P73	P95	277
I/O	P49	P66	M14	P74	P96	280
I/O	-	P67	N15	P75	P97	283
I/O	-	P68	P15	P76	P98	286
I/O	P50	P69	N14	P77	P99	289
I/O, SGCK3	P51	P70	R16	P78	P100	292
GND	P52	P71	P14	P79	P101	-
DONE	P53	P72	R15	P80	P103	-
VCC	P54	P73	P13	P81	P106	-
$\overline{\text{PROGRAM}}$	P55	P74	R14	P82	P108	-

XC4006E Pad Name	PC 84	TQ 144	PG 156	PQ 160	PQ 208	Bndry Scan
I/O (D7)	P56	P75	T16	P83	P109	295
I/O, PGCK3	P57	P76	T15	P84	P110	298
I/O	-	P77	R13	P85	P111	301
I/O	-	P78	P12	P86	P112	304
I/O (D6)	P58	P79	T14	P87	P113	307
I/O	-	P80	T13	P88	P114	310
I/O	-	-	R12	P89	P115	313
I/O	-	-	T12	P90	P116	316
GND	-	P81	P11	P91	P119	-
I/O	-	P82	R11	P92	P120	319
I/O	-	P83	T11	P93	P121	322
I/O (D5)	P59	P84	T10	P94	P122	325
I/O ($\overline{CS0}$)	P60	P85	P10	P95	P123	328
I/O	-	P86	R10	P96	P126	331
I/O	-	P87	T9	P97	P127	334
I/O (D4)	P61	P88	R9	P98	P128	337
I/O	P62	P89	P9	P99	P129	340
VCC	P63	P90	R8	P100	P130	-
GND	P64	P91	P8	P101	P131	-
I/O (D3)	P65	P92	T8	P102	P132	343
I/O ($\overline{RS}$)	P66	P93	T7	P103	P133	346
I/O	-	P94	T6	P104	P134	349
I/O	-	P95	R7	P105	P135	352
I/O (D2)	P67	P96	P7	P106	P138	355
I/O	P68	P97	T5	P107	P139	358
I/O	-	P98	R6	P108	P140	361
I/O	-	P99	T4	P109	P141	364
GND	-	P100	P6	P110	P142	-
I/O	-	-	R5	P111	P145	367
I/O	-	-	-	P112	P146	370
I/O (D1)	P69	P101	T3	P113	P147	373
I/O ($\overline{RCLK}$, RDY/$\overline{BUSY}$)	P70	P102	P5	P114	P148	376
I/O	-	P103	R4	P115	P149	379
I/O	-	P104	R3	P116	P150	382
I/O (D0, DIN)	P71	P105	P4	P117	P151	385
I/O, SGCK4 (DOUT)	P72	P106	T2	P118	P152	388
CCLK	P73	P107	R2	P119	P153	-
VCC	P74	P108	P3	P120	P154	-
O, TDO	P75	P109	T1	P121	P159	0
GND	P76	P110	N3	P122	P160	-
I/O (A0, $\overline{WS}$)	P77	P111	R1	P123	P161	2
I/O, PGCK4 (A1)	P78	P112	P2	P124	P162	5
I/O	-	P113	N2	P125	P163	8
I/O	-	P114	M3	P126	P164	11
I/O (CS1, A2)	P79	P115	P1	P127	P165	14
I/O (A3)	P80	P116	N1	P128	P166	17
I/O	-	P117	M2	P129	P167	20
I/O	-	-	M1	P130	P168	23
GND	-	P118	L3	P131	P171	-
I/O	-	P119	L2	P132	P172	26
I/O	-	P120	L1	P133	P173	29
I/O (A4)	P81	P121	K3	P134	P174	32
I/O (A5)	P82	P122	K2	P135	P175	35
I/O	-	P123	K1	P137	P178	38
I/O	-	P124	J1	P138	P179	41
I/O (A6)	P83	P125	J2	P139	P180	44
I/O (A7)	P84	P126	J3	P140	P181	47
GND	P1	P127	H2	P141	P182	-

4/2/96

Additional No Connect (N.C.) Connections on PQ160 & PQ208 Packages

PQ160	PQ208
P136	P1
	P3
	P12-P13
	P19-20
	P31-P32
	P38-P39
	P51-P54
	P65-P66
	P72-P73
	P84-P85
	P91-P92
	P102
	P104-P105
	P107
	P117-P118
	P124-P125
	P136-P137
	P143-P144
	P155-P158
	P169-P170
	P176-P177
	P188-P189
	P195-P196
	P206-P208

2/28/96

Pin Locations for XC4008E Devices

XC4008E Pad Name	PC 84	PQ 160	PG 191	PQ 208	Bndry Scan
VCC	P2	P142	J4	P183	-
I/O (A8)	P3	P143	J3	P184	56
I/O (A9)	P4	P144	J2	P185	59
I/O	-	P145	J1	P186	62
I/O	-	P146	H1	P187	65
I/O	-	-	H2	P188	68
I/O	-	-	H3	P189	71
I/O (A10)	P5	P147	G1	P190	74
I/O (A11)	P6	P148	G2	P191	77
I/O	-	P149	F1	P192	80
I/O	-	P150	E1	P193	83
GND	-	P151	G3	P194	-
I/O	-	P152	C1	P197	86
I/O	-	P153	E2	P198	89
I/O (A12)	P7	P154	F3	P199	92
I/O (A13)	P8	P155	D2	P200	95
I/O	-	P156	B1	P201	98
I/O	-	P157	E3	P202	101
I/O (A14)	P9	P158	C2	P203	104
I/O, SGCK1 (A15)	P10	P159	B2	P204	107
VCC	P11	P160	D3	P205	-
GND	P12	P1	D4	P2	-
I/O, PGCK1 (A16)	P13	P2	C3	P4	110
I/O (A17)	P14	P3	C4	P5	113
I/O	-	P4	B3	P6	116
I/O	-	P5	C5	P7	119
I/O, TDI	P15	P6	A2	P8	122
I/O, TCK	P16	P7	B4	P9	125
I/O	-	P8	C6	P10	128
I/O	-	P9	A3	P11	131
GND	-	P10	C7	P14	-
I/O	-	P11	A4	P15	134
I/O	-	P12	A5	P16	137
I/O, TMS	P17	P13	B7	P17	140
I/O	P18	P14	A6	P18	143
I/O	-	-	C8	P19	146
I/O	-	-	A7	P20	149
I/O	-	P15	B8	P21	152
I/O	-	P16	A8	P22	155
I/O	P19	P17	B9	P23	158
I/O	P20	P18	C9	P24	161
GND	P21	P19	D9	P25	-
VCC	P22	P20	D10	P26	-
I/O	P23	P21	C10	P27	164
I/O	P24	P22	B10	P28	167
I/O	-	P23	A9	P29	170
I/O	-	P24	A10	P30	173
I/O	-	-	A11	P31	176
I/O	-	-	C11	P32	179
I/O	P25	P25	B11	P33	182

XC4008E Pad Name	PC 84	PQ 160	PG 191	PQ 208	Bndry Scan
I/O	P26	P26	A12	P34	185
I/O	-	P27	B12	P35	188
I/O	-	P28	A13	P36	191
GND	-	P29	C12	P37	-
I/O	-	P30	A15	P40	194
I/O	-	P31	C13	P41	197
I/O	P27	P32	B14	P42	200
I/O	-	P33	A16	P43	203
I/O	-	P34	B15	P44	206
I/O	-	P35	C14	P45	209
I/O	P28	P36	A17	P46	212
I/O, SCGK2	P29	P37	B16	P47	215
O (M1)	P30	P38	C15	P48	218
GND	P31	P39	D15	P49	-
I (M0)	P32	P40	A18	P50	221
VCC	P33	P41	D16	P55	-
I (M2)	P34	P42	C16	P56	222
I/O, PGCK2	P35	P43	B17	P57	223
I/O (HDC)	P36	P44	E16	P58	226
I/O	-	P45	C17	P59	229
I/O	-	P46	D17	P60	232
I/O	-	P47	B18	P61	235
I/O ($\overline{\text{LDC}}$)	P37	P48	E17	P62	238
I/O	-	P49	F16	P63	241
I/O	-	P50	C18	P64	244
GND	-	P51	G16	P67	-
I/O	-	P52	E18	P68	247
I/O	-	P53	F18	P69	250
I/O	P38	P54	G17	P70	253
I/O	P39	P55	G18	P71	256
I/O	-	-	H16	P72	259
I/O	-	-	H17	P73	262
I/O	-	P56	H18	P74	265
I/O	-	P57	J18	P75	268
I/O	P40	P58	J17	P76	271
I/O ($\overline{\text{INIT}}$)	P41	P59	J16	P77	274
VCC	P42	P60	J15	P78	-
GND	P43	P61	K15	P79	-
I/O	P44	P62	K16	P80	277
I/O	P45	P63	K17	P81	280
I/O	-	P64	K18	P82	283
I/O	-	P65	L18	P83	286
I/O	-	-	L17	P84	289
I/O	-	-	L16	P85	292
I/O	P46	P66	M18	P86	295
I/O	P47	P67	M17	P87	298
I/O	-	P68	N18	P88	301
I/O	-	P69	P18	P89	304
GND	-	P70	M16	P90	-
I/O	-	P71	T18	P93	307
I/O	-	P72	P17	P94	310
I/O	P48	P73	N16	P95	313

XC4008E Pad Name	PC 84	PQ 160	PG 191	PQ 208	Bndry Scan
I/O	P49	P74	T17	P96	316
I/O	-	P75	R17	P97	319
I/O	-	P76	P16	P98	322
I/O	P50	P77	U18	P99	325
I/O, SGCK3	P51	P78	T16	P100	328
GND	P52	P79	R16	P101	-
DONE	P53	P80	U17	P103	-
VCC	P54	P81	R15	P106	-
$\overline{\text{PROGRAM}}$	P55	P82	V18	P108	-
I/O (D7)	P56	P83	T15	P109	331
I/O, PGCK3	P57	P84	U16	P110	334
I/O	-	P85	T14	P111	337
I/O	-	P86	U15	P112	340
I/O (D6)	P58	P87	V17	P113	343
I/O	-	P88	V16	P114	346
I/O	-	P89	T13	P115	349
I/O	-	P90	U14	P116	352
GND	-	P91	T12	P119	-
I/O	-	P92	U13	P120	355
I/O	-	P93	V13	P121	358
I/O (D5)	P59	P94	U12	P122	361
I/O ($\overline{\text{CS0}}$)	P60	P95	V12	P123	364
I/O	-	-	T11	P124	367
I/O	-	-	U11	P125	370
I/O	-	P96	V11	P126	373
I/O	-	P97	V10	P127	376
I/O (D4)	P61	P98	U10	P128	379
I/O	P62	P99	T10	P129	382
VCC	P63	P100	R10	P130	-
GND	P64	P101	R9	P131	-
I/O (D3)	P65	P102	T9	P132	385
I/O ($\overline{\text{RS}}$)	P66	P103	U9	P133	388
I/O	-	P104	V9	P134	391
I/O	-	P105	V8	P135	394
I/O	-	-	U8	P136	397
I/O	-	-	T8	P137	400
I/O (D2)	P67	P106	V7	P138	403
I/O	P68	P107	U7	P139	406
I/O	-	P108	V6	P140	409
I/O	-	P109	U6	P141	412
GND	-	P110	T7	P142	-
I/O	-	P111	U5	P145	415
I/O	-	P112	T6	P146	418
I/O (D1)	P69	P113	V3	P147	421
I/O ($\overline{\text{RCLK}}$, RDY/$\overline{\text{BUSY}}$)	P70	P114	V2	P148	424
I/O	-	P115	U4	P149	427
I/O	-	P116	T5	P150	430
I/O (D0, DIN)	P71	P117	U3	P151	433
I/O, SGCK4 (DOUT)	P72	P118	T4	P152	436
CCLK	P73	P119	V1	P153	-
VCC	P74	P120	R4	P154	-

XC4008E Pad Name	PC 84	PQ 160	PG 191	PQ 208	Bndry Scan
O, TDO	P75	P121	U2	P159	0
GND	P76	P122	R3	P160	-
I/O (A0, $\overline{\text{WS}}$)	P77	P123	T3	P161	2
I/O, PGCK4 (A1)	P78	P124	U1	P162	5
I/O	-	P125	P3	P163	8
I/O	-	P126	R2	P164	11
I/O (CS1, A2)	P79	P127	T2	P165	14
I/O (A3)	P80	P128	N3	P166	17
I/O	-	P129	P2	P167	20
I/O	-	P130	T1	P168	23
GND	-	P131	M3	P171	-
I/O	-	P132	P1	P172	26
I/O	-	P133	N1	P173	29
I/O (A4)	P81	P134	M2	P174	32
I/O (A5)	P82	P135	M1	P175	35
I/O	-	-	L3	P176	38
I/O	-	P136	L2	P177	41
I/O	-	P137	L1	P178	44
I/O	-	P138	K1	P179	47
I/O (A6)	P83	P139	K2	P180	50
I/O (A7)	P84	P140	K3	P181	53
GND	P1	P141	K4	P182	-

4/2/96

Additional No Connect (N.C.) Connections on PG191 & PQ208 Packages

PG191	PQ208	PQ208
A14	P1	P107
B5	P3	P117
B6	P12	P118
B13	P13	P143
D1	P38	P144
D18	P39	P155
F2	P51	P156
F17	P52	P157
N2	P53	P158
N17	P54	P169
R1	P65	P170
R18	P66	P195
V4	P91	P196
V5	P92	P206
V14	P102	P207
V15	P104	P208
	P105	

2/28/96

Pin Locations for XC4010E/L Devices

XC4010E/L Pad Name	PC 84	PQ 160	TQ 176	PG 191	PQ/ HQ 208	BG 225	Bndry Scan
VCC	P2	P142	P155	J4	P183	D8	-
I/O (A8)	P3	P143	P156	J3	P184	E8	62
I/O (A9)	P4	P144	P157	J2	P185	B7	65
I/O	-	P145	P158	J1	P186	A7	68
I/O	-	P146	P159	H1	P187	C7	71
I/O	-	-	P160	H2	P188	D7	74
I/O	-	-	P161	H3	P189	E7	77
I/O (A10)	P5	P147	P162	G1	P190	A6	80
I/O (A11)	P6	P148	P163	G2	P191	B6	83
I/O	-	P149	P164	F1	P192	A5	86
I/O	-	P150	P165	E1	P193	B5	89
GND	-	P151	P166	G3	P194	GND*	-
I/O	-	-	-	F2	P195	D6	92
I/O	-	-	P167	D1	P196	C5	95
I/O	-	P152	P168	C1	P197	A4	98
I/O	-	P153	P169	E2	P198	E6	101
I/O (A12)	P7	P154	P170	F3	P199	B4	104
I/O (A13)	P8	P155	P171	D2	P200	D5	107
I/O	-	P156	P172	B1	P201	B3	110
I/O	-	P157	P173	E3	P202	F6	113
I/O (A14)	P9	P158	P174	C2	P203	A2	116
I/O, SGCK1 (A15)	P10	P159	P175	B2	P204	C3	119
VCC	P11	P160	P176	D3	P205	B2	-
GND	P12	P1	P1	D4	P2	A1	-
I/O, PGCK1 (A16)	P13	P2	P2	C3	P4	D4	122
I/O (A17)	P14	P3	P3	C4	P5	B1	125
I/O	-	P4	P4	B3	P6	C2	128
I/O	-	P5	P5	C5	P7	E5	131
I/O, TDI	P15	P6	P6	A2	P8	D3	134
I/O, TCK	P16	P7	P7	B4	P9	C1	137
I/O	-	P8	P8	C6	P10	D2	140
I/O	-	P9	P9	A3	P11	G6	143
I/O	-	-	-	B5	P12	E4	146
I/O	-	-	-	B6	P13	D1	149
GND	-	P10	P10	C7	P14	GND*	-
I/O	-	P11	P11	A4	P15	F5	152
I/O	-	P12	P12	A5	P16	E1	155
I/O, TMS	P17	P13	P13	B7	P17	F4	158
I/O	P18	P14	P14	A6	P18	F3	161
I/O	-	-	P15	C8	P19	G4	164
I/O	-	-	P16	A7	P20	G3	167
I/O	-	P15	P17	B8	P21	G2	170
I/O	-	P16	P18	A8	P22	G1	173
I/O	P19	P17	P19	B9	P23	G5	176
I/O	P20	P18	P20	C9	P24	H3	179
GND	P21	P19	P21	D9	P25	H2	-
VCC	P22	P20	P22	D10	P26	H1	-
I/O	P23	P21	P23	C10	P27	H4	182
I/O	P24	P22	P24	B10	P28	H5	185
I/O	-	P23	P25	A9	P29	J2	188
I/O	-	P24	P26	A10	P30	J1	191
I/O	-	-	P27	A11	P31	J3	194
I/O	-	-	P28	C11	P32	J4	197
I/O	P25	P25	P29	B11	P33	K2	200
I/O	P26	P26	P30	A12	P34	K3	203
I/O	-	P27	P31	B12	P35	J6	206
I/O	-	P28	P32	A13	P36	L1	209
GND	-	P29	P33	C12	P37	GND*	-
I/O	-	-	-	B13	P38	L3	212
I/O	-	-	-	A14	P39	M1	215
I/O	-	P30	P34	A15	P40	K5	218
I/O	-	P31	P35	C13	P41	M2	221
I/O	P27	P32	P36	B14	P42	L4	224
I/O	-	P33	P37	A16	P43	N1	227
I/O	-	P34	P38	B15	P44	M3	230
I/O	-	P35	P39	C14	P45	N2	233
I/O	P28	P36	P40	A17	P46	K6	236
I/O, SCGK2	P29	P37	P41	B16	P47	P1	239
O (M1)	P30	P38	P42	C15	P48	N3	242
GND	P31	P39	P43	D15	P49	GND*	-
I (M0)	P32	P40	P44	A18	P50	P2	245
VCC	P33	P41	P45	D16	P55	R1	-
I (M2)	P34	P42	P46	C16	P56	M4	246
I/O, PGCK2	P35	P43	P47	B17	P57	R2	247
I/O (HDC)	P36	P44	P48	E16	P58	P3	250
I/O	-	P45	P49	C17	P59	L5	253
I/O	-	P46	P50	D17	P60	N4	256
I/O	-	P47	P51	B18	P61	R3	259
I/O ($\overline{LDC}$)	P37	P48	P52	E17	P62	P4	262
I/O	-	P49	P53	F16	P63	K7	265
I/O	-	P50	P54	C18	P64	M5	268
I/O	-	-	-	D18	P65	R4	271
I/O	-	-	-	F17	P66	N5	274
GND	-	P51	P55	G16	P67	GND*	-
I/O	-	P52	P56	E18	P68	R5	277
I/O	-	P53	P57	F18	P69	M6	280
I/O	P38	P54	P58	G17	P70	N6	283
I/O	P39	P55	P59	G18	P71	P6	286
I/O	-	-	P60	H16	P72	R6	289

XC4010E/L Pad Name	PC 84	PQ 160	TQ 176	PG 191	PQ/ HQ 208	BG 225	Bndry Scan
I/O	-	-	P61	H17	P73	M7	292
I/O	-	P56	P62	H18	P74	R7	295
I/O	-	P57	P63	J18	P75	L7	298
I/O	P40	P58	P64	J17	P76	N8	301
I/O ($\overline{INIT}$)	P41	P59	P65	J16	P77	P8	304
VCC	P42	P60	P66	J15	P78	R8	-
GND	P43	P61	P67	K15	P79	M8	-
I/O	P44	P62	P68	K16	P80	L8	307
I/O	P45	P63	P69	K17	P81	P9	310
I/O	-	P64	P70	K18	P82	R9	313
I/O	-	P65	P71	L18	P83	N9	316
I/O	-	-	P72	L17	P84	M9	319
I/O	-	-	P73	L16	P85	L9	322
I/O	P46	P66	P74	M18	P86	N10	325
I/O	P47	P67	P75	M17	P87	K9	328
I/O	-	P68	P76	N18	P88	R11	331
I/O	-	P69	P77	P18	P89	P11	334
GND	-	P70	P78	M16	P90	GND*	-
I/O	-	-	-	N17	P91	R12	337
I/O	-	-	-	R18	P92	L10	340
I/O	-	P71	P79	T18	P93	P12	343
I/O	-	P72	P80	P17	P94	M11	346
I/O	P48	P73	P81	N16	P95	R13	349
I/O	P49	P74	P82	T17	P96	N12	352
I/O	-	P75	P83	R17	P97	P13	355
I/O	-	P76	P84	P16	P98	K10	358
I/O	P50	P77	P85	U18	P99	R14	361
I/O, SGCK3	P51	P78	P86	T16	P100	N13	364
GND	P52	P79	P87	R16	P101	GND*	-
DONE	P53	P80	P88	U17	P103	P14	-
VCC	P54	P81	P89	R15	P106	R15	-
$\overline{PROGRAM}$	P55	P82	P90	V18	P108	M12	-
I/O (D7)	P56	P83	P91	T15	P109	P15	367
I/O, PGCK3	P57	P84	P92	U16	P110	N14	370
I/O	-	P85	P93	T14	P111	L11	373
I/O	-	P86	P94	U15	P112	M13	376
I/O (D6)	P58	P87	P95	V17	P113	J10	379
I/O	-	P88	P96	V16	P114	L12	382
I/O	-	P89	P97	T13	P115	M15	385
I/O	-	P90	P98	U14	P116	L13	388
I/O	-	-	-	V15	P117	L14	391
I/O	-	-	-	V14	P118	K11	394
GND	-	P91	P99	T12	P119	GND*	-
I/O	-	P92	P100	U13	P120	K13	397
I/O	-	P93	P101	V13	P121	K14	400
I/O (D5)	P59	P94	P102	U12	P122	K15	403

XC4010E/L Pad Name	PC 84	PQ 160	TQ 176	PG 191	PQ/ HQ 208	BG 225	Bndry Scan
I/O ($\overline{CS0}$)	P60	P95	P103	V12	P123	J12	406
I/O	-	-	P104	T11	P124	J13	409
I/O	-	-	P105	U11	P125	J14	412
I/O	-	P96	P106	V11	P126	J15	415
I/O	-	P97	P107	V10	P127	J11	418
I/O (D4)	P61	P98	P108	U10	P128	H13	421
I/O	P62	P99	P109	T10	P129	H14	424
VCC	P63	P100	P110	R10	P130	H15	-
GND	P64	P101	P111	R9	P131	GND*	-
I/O (D3)	P65	P102	P112	T9	P132	H12	427
I/O ($\overline{RS}$)	P66	P103	P113	U9	P133	H11	430
I/O	-	P104	P114	V9	P134	G14	433
I/O	-	P105	P115	V8	P135	G15	436
I/O	-	-	P116	U8	P136	G13	439
I/O	-	-	P117	T8	P137	G12	442
I/O (D2)	P67	P106	P118	V7	P138	G11	445
I/O	P68	P107	P119	U7	P139	F15	448
I/O	-	P108	P120	V6	P140	F14	451
I/O	-	P109	P121	U6	P141	F13	454
GND	-	P110	P122	T7	P142	GND*	-
I/O	-	-	-	V5	P143	E13	457
I/O	-	-	-	V4	P144	D15	460
I/O	-	P111	P123	U5	P145	F11	463
I/O	-	P112	P124	T6	P146	D14	466
I/O (D1)	P69	P113	P125	V3	P147	E12	469
I/O ($\overline{RCLK}$, $\overline{RDY}$/ $\overline{BUSY}$)	P70	P114	P126	V2	P148	C15	472
I/O	-	P115	P127	U4	P149	D13	475
I/O	-	P116	P128	T5	P150	C14	478
I/O (D0, DIN)	P71	P117	P129	U3	P151	F10	481
I/O, SGCK4 (DOUT)	P72	P118	P130	T4	P152	B15	484
CCLK	P73	P119	P131	V1	P153	C13	-
VCC	P74	P120	P132	R4	P154	B14	-
O, TDO	P75	P121	P133	U2	P159	A15	0
GND	P76	P122	P134	R3	P160	D12	-
I/O (A0, $\overline{WS}$)	P77	P123	P135	T3	P161	A14	2
I/O, PGCK4 (A1)	P78	P124	P136	U1	P162	B13	5
I/O	-	P125	P137	P3	P163	E11	8
I/O	-	P126	P138	R2	P164	C12	11
I/O (CS1, A2)	P79	P127	P139	T2	P165	A13	14
I/O (A3)	P80	P128	P140	N3	P166	B12	17

XC4010E/L Pad Name	PC 84	PQ 160	TQ 176	PG 191	PQ/ HQ 208	BG 225	Bndry Scan
I/O	-	P129	P141	P2	P167	A12	20
I/O	-	P130	P142	T1	P168	C11	23
I/O	-	-	-	R1	P169	B11	26
I/O	-	-	-	N2	P170	E10	29
GND	-	P131	P143	M3	P171	GND*	-
I/O	-	P132	P144	P1	P172	A11	32
I/O	-	P133	P145	N1	P173	D10	35
I/O (A4)	P81	P134	P146	M2	P174	A10	38
I/O (A5)	P82	P135	P147	M1	P175	D9	41
I/O	-	-	P148	L3	P176	C9	44
I/O	-	P136	P149	L2	P177	B9	47
I/O	-	P137	P150	L1	P178	A9	50
I/O	-	P138	P151	K1	P179	E9	53
I/O (A6)	P83	P139	P152	K2	P180	C8	56
I/O (A7)	P84	P140	P153	K3	P181	B8	59
GND	P1	P141	P154	K4	P182	A8	-

4/2/96

* Pads labelled GND* are internally bonded to a Ground plane within the BG225 package. They have no direct connection to any package pin.

Additional Ground (GND) Connections on BG225 Package

GND
F8
G7
G8
G9
H6
H7
H8
H9
H10
J7
J8
J9
K8

2/28/96

Note: The package pins in this table are bonded to an internal Ground plane within the BG225 package. They should all be externally connected to Ground.

Additional No Connect (N.C.) Connections on PQ/ HQ208 & BG225 Packages

PQ/HQ208	BG225
P1	A3
P3	B10
P51	C4
P52	C6
P53	C10
P54	D11
P102	E2
P104	E3
P105	E14
P107	E15
P155	F1
P156	F2
P157	F7
P158	F9
P206	F12
P207	G10
P208	J5
	K1
	K4
	K12
	L2
	L6
	L15
	M10
	M14
	N7
	N11
	N15
	P5
	P7
	P10
	R10

3/12/96

Pin Locations for XC4013E/L Devices

XC4013E/L Pad Name	PQ 160	PQ/ HQ 208	PG 223	BG 225	PQ/ HQ 240	Bndry Scan
VCC	P142	P183	J4	D8	P212	-
I/O (A8)	P143	P184	J3	E8	P213	74
I/O (A9)	P144	P185	J2	B7	P214	77
I/O	P145	P186	J1	A7	P215	80
I/O	P146	P187	H1	C7	P216	83
I/O	-	P188	H2	D7	P217	86
I/O	-	P189	H3	E7	P218	89
I/O (A10)	P147	P190	G1	A6	P220	92
I/O (A11)	P148	P191	G2	B6	P221	95
VCC	-	-	-	VCC*	P222	-
I/O	-	-	H4	C6	P223	98
I/O	-	-	G4	F7	P224	101
I/O	P149	P192	F1	A5	P225	104
I/O	P150	P193	E1	B5	P226	107
GND	P151	P194	G3	GND*	P227	-
I/O	-	P195	F2	D6	P228	110
I/O	-	P196	D1	C5	P229	113
I/O	P152	P197	C1	A4	P230	116
I/O	P153	P198	E2	E6	P231	119
I/O (A12)	P154	P199	F3	B4	P232	122
I/O (A13)	P155	P200	D2	D5	P233	125
I/O	-	-	F4	A3	P234	128
I/O	-	-	E4	C4	P235	131
I/O	P156	P201	B1	B3	P236	134
I/O	P157	P202	E3	F6	P237	137
I/O (A14)	P158	P203	C2	A2	P238	140
I/O, SGCK1 (A15)	P159	P204	B2	C3	P239	143
VCC	P160	P205	D3	B2	P240	-
GND	P1	P2	D4	A1	P1	-
I/O, PGCK1 (A16)	P2	P4	C3	D4	P2	146
I/O (A17)	P3	P5	C4	B1	P3	149
I/O	P4	P6	B3	C2	P4	152
I/O	P5	P7	C5	E5	P5	155
I/O, TDI	P6	P8	A2	D3	P6	158
I/O, TCK	P7	P9	B4	C1	P7	161
I/O	P8	P10	C6	D2	P8	164
I/O	P9	P11	A3	G6	P9	167
I/O	-	P12	B5	E4	P10	170
I/O	-	P13	B6	D1	P11	173
I/O	-	-	D5	E3	P12	176
I/O	-	-	D6	E2	P13	179
GND	P10	P14	C7	GND*	P14	-
I/O	P11	P15	A4	F5	P15	182
I/O	P12	P16	A5	E1	P16	185
I/O, TMS	P13	P17	B7	F4	P17	188
I/O	P14	P18	A6	F3	P18	191
VCC	-	-	-	VCC*	P19	-

XC4013E/L Pad Name	PQ 160	PQ/ HQ 208	PG 223	BG 225	PQ/ HQ 240	Bndry Scan
I/O	-	-	D7	F2	P20	194
I/O	-	-	D8	F1	P21	197
I/O	-	P19	C8	G4	P23	200
I/O	-	P20	A7	G3	P24	203
I/O	P15	P21	B8	G2	P25	206
I/O	P16	P22	A8	G1	P26	209
I/O	P17	P23	B9	G5	P27	212
I/O	P18	P24	C9	H3	P28	215
GND	P19	P25	D9	H2	P29	-
VCC	P20	P26	D10	H1	P30	-
I/O	P21	P27	C10	H4	P31	218
I/O	P22	P28	B10	H5	P32	221
I/O	P23	P29	A9	J2	P33	224
I/O	P24	P30	A10	J1	P34	227
I/O	-	P31	A11	J3	P35	230
I/O	-	P32	C11	J4	P36	233
I/O	-	-	D11	J5	P38	236
I/O	-	-	D12	K1	P39	239
VCC	-	-	-	VCC*	P40	-
I/O	P25	P33	B11	K2	P41	242
I/O	P26	P34	A12	K3	P42	245
I/O	P27	P35	B12	J6	P43	248
I/O	P28	P36	A13	L1	P44	251
GND	P29	P37	C12	GND*	P45	-
I/O	-	-	D13	L2	P46	254
I/O	-	-	D14	K4	P47	257
I/O	-	P38	B13	L3	P48	260
I/O	-	P39	A14	M1	P49	263
I/O	P30	P40	A15	K5	P50	266
I/O	P31	P41	C13	M2	P51	269
I/O	P32	P42	B14	L4	P52	272
I/O	P33	P43	A16	N1	P53	275
I/O	P34	P44	B15	M3	P54	278
I/O	P35	P45	C14	N2	P55	281
I/O	P36	P46	A17	K6	P56	284
I/O, SCGK2	P37	P47	B16	P1	P57	287
O (M1)	P38	P48	C15	N3	P58	290
GND	P39	P49	D15	GND*	P59	-
I (M0)	P40	P50	A18	P2	P60	293
VCC	P41	P55	D16	R1	P61	-
I (M2)	P42	P56	C16	M4	P62	294
I/O, PGCK2	P43	P57	B17	R2	P63	295
I/O (HDC)	P44	P58	E16	P3	P64	298
I/O	P45	P59	C17	L5	P65	301
I/O	P46	P60	D17	N4	P66	304
I/O	P47	P61	B18	R3	P67	307
I/O ($\overline{LDC}$)	P48	P62	E17	P4	P68	310
I/O	P49	P63	F16	K7	P69	313
I/O	P50	P64	C18	M5	P70	316
I/O	-	P65	D18	R4	P71	319

XC4013E/L Pad Name	PQ 160	PQ/ HQ 208	PG 223	BG 225	PQ/ HQ 240	Bndry Scan
I/O	-	P66	F17	N5	P72	322
I/O	-	-	E15	P5	P73	325
I/O	-	-	F15	L6	P74	328
GND	P51	P67	G16	GND*	P75	-
I/O	P52	P68	E18	R5	P76	331
I/O	P53	P69	F18	M6	P77	334
I/O	P54	P70	G17	N6	P78	337
I/O	P55	P71	G18	P6	P79	340
VCC	-	-	-	VCC*	P80	-
I/O	-	P72	H16	R6	P81	343
I/O	-	P73	H17	M7	P82	346
I/O	-	-	G15	N7	P84	349
I/O	-	-	H15	P7	P85	352
I/O	P56	P74	H18	R7	P86	355
I/O	P57	P75	J18	L7	P87	358
I/O	P58	P76	J17	N8	P88	361
I/O ($\overline{INIT}$)	P59	P77	J16	P8	P89	364
VCC	P60	P78	J15	R8	P90	-
GND	P61	P79	K15	M8	P91	-
I/O	P62	P80	K16	L8	P92	367
I/O	P63	P81	K17	P9	P93	370
I/O	P64	P82	K18	R9	P94	373
I/O	P65	P83	L18	N9	P95	376
I/O	-	P84	L17	M9	P96	379
I/O	-	P85	L16	L9	P97	382
I/O	-	-	L15	R10	P99	385
I/O	-	-	M15	P10	P100	388
VCC	-	-	-	VCC*	P101	-
I/O	P66	P86	M18	N10	P102	391
I/O	P67	P87	M17	K9	P103	394
I/O	P68	P88	N18	R11	P104	397
I/O	P69	P89	P18	P11	P105	400
GND	P70	P90	M16	GND*	P106	-
I/O	-	-	N15	M10	P107	403
I/O	-	-	P15	N11	P108	406
I/O	-	P91	N17	R12	P109	409
I/O	-	P92	R18	L10	P110	412
I/O	P71	P93	T18	P12	P111	415
I/O	P72	P94	P17	M11	P112	418
I/O	P73	P95	N16	R13	P113	421
I/O	P74	P96	T17	N12	P114	424
I/O	P75	P97	R17	P13	P115	427
I/O	P76	P98	P16	K10	P116	430
I/O	P77	P99	U18	R14	P117	433
I/O, SGCK3	P78	P100	T16	N13	P118	436
GND	P79	P101	R16	GND*	P119	-
DONE	P80	P103	U17	P14	P120	-
VCC	P81	P106	R15	R15	P121	-
$\overline{PROGRAM}$	P82	P108	V18	M12	P122	-
I/O (D7)	P83	P109	T15	P15	P123	439

XC4013E/L Pad Name	PQ 160	PQ/ HQ 208	PG 223	BG 225	PQ/ HQ 240	Bndry Scan
I/O, PGCK3	P84	P110	U16	N14	P124	442
I/O	P85	P111	T14	L11	P125	445
I/O	P86	P112	U15	M13	P126	448
I/O	-	-	R14	N15	P127	451
I/O	-	-	R13	M14	P128	454
I/O (D6)	P87	P113	V17	J10	P129	457
I/O	P88	P114	V16	L12	P130	460
I/O	P89	P115	T13	M15	P131	463
I/O	P90	P116	U14	L13	P132	466
I/O	-	P117	V15	L14	P133	469
I/O	-	P118	V14	K11	P134	472
GND	P91	P119	T12	GND*	P135	-
I/O	-	-	R12	L15	P136	475
I/O	-	-	R11	K12	P137	478
I/O	P92	P120	U13	K13	P138	481
I/O	P93	P121	V13	K14	P139	484
VCC	-	-	-	VCC*	P140	-
I/O (D5)	P94	P122	U12	K15	P141	487
I/O ($\overline{CS0}$)	P95	P123	V12	J12	P142	490
I/O	-	P124	T11	J13	P144	493
I/O	-	P125	U11	J14	P145	496
I/O	P96	P126	V11	J15	P146	499
I/O	P97	P127	V10	J11	P147	502
I/O (D4)	P98	P128	U10	H13	P148	505
I/O	P99	P129	T10	H14	P149	508
VCC	P100	P130	R10	H15	P150	-
GND	P101	P131	R9	GND*	P151	-
I/O (D3)	P102	P132	T9	H12	P152	511
I/O ($\overline{RS}$)	P103	P133	U9	H11	P153	514
I/O	P104	P134	V9	G14	P154	517
I/O	P105	P135	V8	G15	P155	520
I/O	-	P136	U8	G13	P156	523
I/O	-	P137	T8	G12	P157	526
I/O (D2)	P106	P138	V7	G11	P159	529
I/O	P107	P139	U7	F15	P160	532
VCC	-	-	-	VCC*	P161	-
I/O	P108	P140	V6	F14	P162	535
I/O	P109	P141	U6	F13	P163	538
I/O	-	-	R8	G10	P164	541
I/O	-	-	R7	E15	P165	544
GND	P110	P142	T7	GND*	P166	-
I/O	-	-	R6	E14	P167	547
I/O	-	-	R5	F12	P168	550
I/O	-	P143	V5	E13	P169	553
I/O	-	P144	V4	D15	P170	556
I/O	P111	P145	U5	F11	P171	559
I/O	P112	P146	T6	D14	P172	562
I/O (D1)	P113	P147	V3	E12	P173	565
I/O ($\overline{RCLK}$, RDY/$\overline{BUSY}$)	P114	P148	V2	C15	P174	568
I/O	P115	P149	U4	D13	P175	571

XC4013E/L Pad Name	PQ 160	PQ/ HQ 208	PG 223	BG 225	PQ/ HQ 240	Bndry Scan
I/O	P116	P150	T5	C14	P176	574
I/O (D0, DIN)	P117	P151	U3	F10	P177	577
I/O, SGCK4 (DOUT)	P118	P152	T4	B15	P178	580
CCLK	P119	P153	V1	C13	P179	-
VCC	P120	P154	R4	B14	P180	-
O, TDO	P121	P159	U2	A15	P181	0
GND	P122	P160	R3	D12	P182	-
I/O (A0, $\overline{WS}$)	P123	P161	T3	A14	P183	2
I/O, PGCK4 (A1)	P124	P162	U1	B13	P184	5
I/O	P125	P163	P3	E11	P185	8
I/O	P126	P164	R2	C12	P186	11
I/O (CS1, A2)	P127	P165	T2	A13	P187	14
I/O (A3)	P128	P166	N3	B12	P188	17
I/O	-	-	P4	F9	P189	20
I/O	-	-	N4	D11	P190	23
I/O	P129	P167	P2	A12	P191	26
I/O	P130	P168	T1	C11	P192	29
I/O	-	P169	R1	B11	P193	32
I/O	-	P170	N2	E10	P194	35
GND	P131	P171	M3	GND*	P196	-
I/O	P132	P172	P1	A11	P197	38
I/O	P133	P173	N1	D10	P198	41
I/O	-	-	M4	C10	P199	44
I/O	-	-	L4	B10	P200	47
VCC	-	-	-	VCC*	P201	-
I/O (A4)	P134	P174	M2	A10	P202	50
I/O (A5)	P135	P175	M1	D9	P203	53
I/O	-	P176	L3	C9	P205	56
I/O	P136	P177	L2	B9	P206	59
I/O	P137	P178	L1	A9	P207	62
I/O	P138	P179	K1	E9	P208	65
I/O (A6)	P139	P180	K2	C8	P209	68
I/O (A7)	P140	P181	K3	B8	P210	71
GND	P141	P182	K4	A8	P211	-

4/2/96

Pads labelled GND* are internally bonded to a Ground plane within the BG225 package. They have no direct connection to any package pin.

Pads labelled VCC* are internally bonded to a Vcc plane within the BG225 package. They have no direct connection to any package pin.

Additional Ground (GND) Connections on BG225 Packages

BG225	BG225
K8	H9
J7	H10
J8	G7
J9	G8
H6	G9
H7	F8
H8	

3/11/96

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

Additional No Connect (N.C.) Connections on PQ/ HQ208 & PQ/HQ240 Packages

PQ/HQ208	PQ/HQ240
P1	P22 ‡
P3	P37 ‡
P51	P83 ‡
P52	P98 ‡
P53	P143 ‡
P54	P158 ‡
P102	P195
P104	P204 ‡
P105	P219 ‡
P107	
P155	
P156	
P157	
P158	
P206	
P207	
P208	

3/20/96

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

Pin Locations for XC4020E Devices

XC4020E Pad Name	HQ 208	PG 223	HQ 240	Bndry Scan
VCC	P183	J4	P212	-
I/O (A8)	P184	J3	P213	86
I/O (A9)	P185	J2	P214	89
I/O	P186	J1	P215	92
I/O	P187	H1	P216	95
I/O	P188	H2	P217	98
I/O	P189	H3	P218	101
I/O (A10)	P190	G1	P220	104
I/O (A11)	P191	G2	P221	107
I/O	-	-	-	110
I/O	-	-	-	113
VCC	-	-	P222	-
I/O	-	H4	P223	116
I/O	-	G4	P224	119
I/O	P192	F1	P225	122
I/O	P193	E1	P226	125
GND	P194	G3	P227	-
I/O	P195	F2	P228	128
I/O	P196	D1	P229	131
I/O	P197	C1	P230	134
I/O	P198	E2	P231	137
I/O (A12)	P199	F3	P232	140
I/O (A13)	P200	D2	P233	143
I/O	-	-	-	146
I/O	-	-	-	149
I/O	-	F4	P234	152
I/O	-	E4	P235	155
I/O	P201	B1	P236	158
I/O	P202	E3	P237	161
I/O (A14)	P203	C2	P238	164
I/O, SGCK1 (A15)	P204	B2	P239	167
VCC	P205	D3	P240	-
GND	P2	D4	P1	-
I/O, PGCK1 (A16)	P4	C3	P2	170
I/O (A17)	P5	C4	P3	173
I/O	P6	B3	P4	176
I/O	P7	C5	P5	179
I/O, TDI	P8	A2	P6	182
I/O, TCK	P9	B4	P7	185
I/O	-	-	-	188
I/O	-	-	-	191
I/O	P10	C6	P8	194
I/O	P11	A3	P9	197
I/O	P12	B5	P10	200
I/O	P13	B6	P11	203
I/O	-	D5	P12	206
I/O	-	D6	P13	209
GND	P14	C7	P14	-
I/O	P15	A4	P15	212
I/O	P16	A5	P16	215

XC4020E Pad Name	HQ 208	PG 223	HQ 240	Bndry Scan
I/O, TMS	P17	B7	P17	218
I/O	P18	A6	P18	221
VCC	-	-	P19	-
I/O	-	D7	P20	224
I/O	-	D8	P21	227
I/O	-	-	-	230
I/O	-	-	-	233
I/O	P19	C8	P23	236
I/O	P20	A7	P24	239
I/O	P21	B8	P25	242
I/O	P22	A8	P26	245
I/O	P23	B9	P27	248
I/O	P24	C9	P28	251
GND	P25	D9	P29	-
VCC	P26	D10	P30	-
I/O	P27	C10	P31	254
I/O	P28	B10	P32	257
I/O	P29	A9	P33	260
I/O	P30	A10	P34	263
I/O	P31	A11	P35	266
I/O	P32	C11	P36	269
I/O	-	-	-	272
I/O	-	-	-	275
I/O	-	D11	P38	278
I/O	-	D12	P39	281
VCC	-	-	P40	-
I/O	P33	B11	P41	284
I/O	P34	A12	P42	287
I/O	P35	B12	P43	290
I/O	P36	A13	P44	293
GND	P37	C12	P45	-
I/O	-	D13	P46	296
I/O	-	D14	P47	299
I/O	P38	B13	P48	302
I/O	P39	A14	P49	305
I/O	P40	A15	P50	308
I/O	P41	C13	P51	311
I/O	-	-	-	314
I/O	-	-	-	317
I/O	P42	B14	P52	320
I/O	P43	A16	P53	323
I/O	P44	B15	P54	326
I/O	P45	C14	P55	329
I/O	P46	A17	P56	332
I/O, SCGK2	P47	B16	P57	335
O (M1)	P48	C15	P58	338
GND	P49	D15	P59	-
I (M0)	P50	A18	P60	341
VCC	P55	D16	P61	-
I (M2)	P56	C16	P62	342
I/O, PGCK2	P57	B17	P63	343
I/O (HDC)	P58	E16	P64	346

XC4020E Pad Name	HQ 208	PG 223	HQ 240	Bndry Scan
I/O	P59	C17	P65	349
I/O	P60	D17	P66	352
I/O	P61	B18	P67	355
I/O ($\overline{LDC}$)	P62	E17	P68	358
I/O	-	-	-	361
I/O	-	-	-	364
I/O	P63	F16	P69	367
I/O	P64	C18	P70	370
I/O	P65	D18	P71	373
I/O	P66	F17	P72	376
I/O	-	E15	P73	379
I/O	-	F15	P74	382
GND	P67	G16	P75	-
I/O	P68	E18	P76	385
I/O	P69	F18	P77	388
I/O	P70	G17	P78	391
I/O	P71	G18	P79	394
VCC	-	-	P80	-
I/O	P72	H16	P81	397
I/O	P73	H17	P82	400
I/O	-	-	-	403
I/O	-	-	-	406
I/O	-	G15	P84	409
I/O	-	H15	P85	412
I/O	P74	H18	P86	415
I/O	P75	J18	P87	418
I/O	P76	J17	P88	421
I/O ($\overline{INIT}$)	P77	J16	P89	424
VCC	P78	J15	P90	-
GND	P79	K15	P91	-
I/O	P80	K16	P92	427
I/O	P81	K17	P93	430
I/O	P82	K18	P94	433
I/O	P83	L18	P95	436
I/O	P84	L17	P96	439
I/O	P85	L16	P97	442
I/O	-	-	-	445
I/O	-	-	-	448
I/O	-	L15	P99	451
I/O	-	M15	P100	454
VCC	-	-	P101	-
I/O	P86	M18	P102	457
I/O	P87	M17	P103	460
I/O	P88	N18	P104	463
I/O	P89	P18	P105	466
GND	P90	M16	P106	-
I/O	-	N15	P107	469
I/O	-	P15	P108	472
I/O	P91	N17	P109	475
I/O	P92	R18	P110	478
I/O	P93	T18	P111	481
I/O	P94	P17	P112	484

XC4020E Pad Name	HQ 208	PG 223	HQ 240	Bndry Scan
I/O	-	-	-	487
I/O	-	-	-	490
I/O	P95	N16	P113	493
I/O	P96	T17	P114	496
I/O	P97	R17	P115	499
I/O	P98	P16	P116	502
I/O	P99	U18	P117	505
I/O, SGCK3	P100	T16	P118	508
GND	P101	R16	P119	-
DONE	P103	U17	P120	-
VCC	P106	R15	P121	-
$\overline{PROGRAM}$	P108	V18	P122	-
I/O (D7)	P109	T15	P123	511
I/O, PGCK3	P110	U16	P124	514
I/O	P111	T14	P125	517
I/O	P112	U15	P126	520
I/O	-	R14	P127	523
I/O	-	R13	P128	526
I/O	-	-	-	529
I/O	-	-	-	532
I/O (D6)	P113	V17	P129	535
I/O	P114	V16	P130	538
I/O	P115	T13	P131	541
I/O	P116	U14	P132	544
I/O	P117	V15	P133	547
I/O	P118	V14	P134	550
GND	P119	T12	P135	-
I/O	-	R12	P136	553
I/O	-	R11	P137	556
I/O	P120	U13	P138	559
I/O	P121	V13	P139	562
VCC	-	-	P140	-
I/O (D5)	P122	U12	P141	565
I/O ($\overline{CS0}$)	P123	V12	P142	568
I/O	-	-	-	571
I/O	-	-	-	574
I/O	P124	T11	P144	577
I/O	P125	U11	P145	580
I/O	P126	V11	P146	583
I/O	P127	V10	P147	586
I/O (D4)	P128	U10	P148	589
I/O	P129	T10	P149	592
VCC	P130	R10	P150	-
GND	P131	R9	P151	-
I/O (D3)	P132	T9	P152	595
I/O ($\overline{RS}$)	P133	U9	P153	598
I/O	P134	V9	P154	601
I/O	P135	V8	P155	604
I/O	P136	U8	P156	607
I/O	P137	T8	P157	610
I/O	-	-	-	613
I/O	-	-	-	616

XC4020E Pad Name	HQ 208	PG 223	HQ 240	Bndry Scan
I/O (D2)	P138	V7	P159	619
I/O	P139	U7	P160	622
VCC	-	-	P161	-
I/O	P140	V6	P162	625
I/O	P141	U6	P163	628
I/O	-	R8	P164	631
I/O	-	R7	P165	634
GND	P142	T7	P166	-
I/O	-	R6	P167	637
I/O	-	R5	P168	640
I/O	P143	V5	P169	643
I/O	P144	V4	P170	646
I/O	P145	U5	P171	649
I/O	P146	T6	P172	652
I/O (D1)	P147	V3	P173	655
I/O ($\overline{RCLK}$, RDY/$\overline{BUSY}$)	P148	V2	P174	658
I/O	-	-	-	661
I/O	-	-	-	664
I/O	P149	U4	P175	667
I/O	P150	T5	P176	670
I/O (D0, DIN)	P151	U3	P177	673
I/O, SGCK4 (DOUT)	P152	T4	P178	676
CCLK	P153	V1	P179	-
VCC	P154	R4	P180	-
O, TDO	P159	U2	P181	0
GND	P160	R3	P182	-
I/O (A0, $\overline{WS}$)	P161	T3	P183	2
I/O, PGCK4 (A1)	P162	U1	P184	5
I/O	P163	P3	P185	8
I/O	P164	R2	P186	11
I/O (CS1, A2)	P165	T2	P187	14
I/O (A3)	P166	N3	P188	17
I/O	-	-	-	20
I/O	-	-	-	23
I/O	-	P4	P189	26
I/O	-	N4	P190	29
I/O	P167	P2	P191	32
I/O	P168	T1	P192	35
I/O	P169	R1	P193	38
I/O	P170	N2	P194	41
GND	P171	M3	P196	-
I/O	P172	P1	P197	44
I/O	P173	N1	P198	47
I/O	-	M4	P199	50
I/O	-	L4	P200	53
VCC	-	-	P201	-
I/O	-	-	-	56
I/O	-	-	-	59
I/O (A4)	P174	M2	P202	62
I/O (A5)	P175	M1	P203	65
I/O	P176	L3	P205	68
I/O	P177	L2	P206	71

XC4020E Pad Name	HQ 208	PG 223	HQ 240	Bndry Scan
I/O	P178	L1	P207	74
I/O	P179	K1	P208	77
I/O (A6)	P180	K2	P209	80
I/O (A7)	P181	K3	P210	83
GND	P182	K4	P211	-

4/2/96

Additional No Connect (N.C.) Connections on HQ208 & HQ240 Packages

HQ208	HQ240
P1	P22 ‡
P3	P37 ‡
P51	P83 ‡
P52	P98 ‡
P53	P143 ‡
P54	P158 ‡
P102	P195
P104	P204 ‡
P105	P219 ‡
P107	
P155	
P156	
P157	
P158	
P206	
P207	
P208	

3/20/96

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

Pin Locations for XC4025E, XC4028EX, & XC4028XL Devices

XC4025E, /28EX/XL Pad Name	HQ 208	PG 223	HQ 240	PG 299	HQ 304	BG 352	Bndry Scan
VCC	P183	J4	P212	K1	P38	VCC*	-
I/O (A8)	P184	J3	P213	K2	P37	D14	98
I/O (A9)	P185	J2	P214	K3	P36	C14	101
I/O (A19)	P186	J1	P215	K5	P35	A15	104
I/O (A18)	P187	H1	P216	K4	P34	B15	107
I/O	P188	H2	P217	J1	P33	C15	110
I/O	P189	H3	P218	J2	P32	D15	113
I/O (A10)	P190	G1	P220	H1	P31	A16	116
I/O (A11)	P191	G2	P221	J3	P30	B16	119
GND	-	-	-	-	-	GND*	-
I/O	-	-	-	J4	P29	C16	122
I/O	-	-	-	J5	P28	B17	125
I/O	-	-	-	H2	P27	C17	128
I/O	-	-	-	G1	P26	B18	131
VCC	-	-	P222	E1	P25	VCC*	-
I/O	-	H4	P223	H3	P23	C18	134
I/O	-	G4	P224	G2	P22	D17	137
I/O	P192	F1	P225	H4	P21	A20	140
I/O	P193	E1	P226	F2	P20	B19	143
GND	P194	G3	P227	F1	P19	GND*	-
I/O	-	-	-	H5	P18	C19	146
I/O	-	-	-	G3	P17	D18	149
I/O	P195	F2	P228	D1	P16	A21	152
I/O	P196	D1	P229	G4	P15	B20	155
I/O	P197	C1	P230	E2	P14	C20	158
I/O	P198	E2	P231	F3	P13	B21	161
I/O (A12)	P199	F3	P232	G5	P12	B22	164
I/O (A13)	P200	D2	P233	C1	P10	C21	167
GND	-	-	-	-	-	GND*	-
VCC	-	-	-	-	-	VCC*	-
I/O	-	-	-	F4	P9	D20	170
I/O	-	-	-	E3	P8	A23	173
I/O	-	F4	P234	D2	P7	D21	176
I/O	-	E4	P235	C2	P6	C22	179
I/O	P201	B1	P236	F5	P5	B24	182
I/O	P202	E3	P237	E4	P4	C23	185
I/O (A14)	P203	C2	P238	D3	P3	D22	188
I/O, SGCK1, GCK8 (A15)	P204	B2	P239	C3	P2	C24	191
VCC	P205	D3	P240	A2	P1	VCC*	-
GND	P2	D4	P1	B1	P304	GND*	-
I/O, PGCK1, GCK1 (A16)	P4	C3	P2	D4	P303	D23	194
I/O (A17)	P5	C4	P3	B2	P302	C25	197
I/O	P6	B3	P4	B3	P301	D24	200
I/O	P7	C5	P5	E6	P300	E23	203
I/O, TDI	P8	A2	P6	D5	P299	C26	206
I/O, TCK	P9	B4	P7	C4	P298	E24	209
I/O	-	-	-	A3	P297	F24	212

XC4025E, /28EX/XL Pad Name	HQ 208	PG 223	HQ 240	PG 299	HQ 304	BG 352	Bndry Scan
I/O	-	-	-	D6	P296	E25	215
VCC	-	-	-	-	-	VCC*	-
GND	-	-	-	-	-	GND*	-
I/O	P10	C6	P8	E7	P295	D26	218
I/O	P11	A3	P9	B4	P294	G24	221
I/O	P12	B5	P10	C5	P293	F25	224
I/O	P13	B6	P11	A4	P292	F26	227
I/O	-	D5	P12	D7	P291	H23	230
I/O	-	D6	P13	C6	P290	H24	233
I/O	-	-	-	E8	P289	G25	236
I/O	-	-	-	B5	P288	G26	239
GND	P14	C7	P14	A5	P287	GND*	-
I/O, FCLK1	P15	A4	P15	B6	P286	J23	242
I/O	P16	A5	P16	D8	P285	J24	245
I/O, TMS	P17	B7	P17	C7	P284	H25	248
I/O	P18	A6	P18	B7	P283	K23	251
VCC	-	-	P19	A6	P282	VCC*	-
I/O	-	D7	P20	C8	P280	K24	254
I/O	-	D8	P21	E9	P279	J25	257
I/O	-	-	-	A7	P278	L24	260
I/O	-	-	-	D9	P277	K25	263
GND‡	-	-	P22	-	-	GND*	-
I/O	-	-	-	B8	P276	L25	266
I/O	-	-	-	A8	P275	L26	269
I/O	P19	C8	P23	C9	P274	M23	272
I/O	P20	A7	P24	B9	P273	M24	275
I/O	P21	B8	P25	E10	P272	M25	278
I/O	P22	A8	P26	A9	P271	M26	281
I/O	P23	B9	P27	D10	P270	N24	284
I/O	P24	C9	P28	C10	P269	N25	287
GND	P25	D9	P29	A10	P268	GND*	-
VCC	P26	D10	P30	A11	P267	VCC*	-
I/O	P27	C10	P31	B10	P266	N26	290
I/O	P28	B10	P32	B11	P265	P25	293
I/O	P29	A9	P33	C11	P264	P23	296
I/O	P30	A10	P34	E11	P263	P24	299
I/O	P31	A11	P35	D11	P262	R26	302
I/O	P32	C11	P36	A12	P261	R25	305
I/O	-	-	-	B12	P260	R24	308
I/O	-	-	-	A13	P259	R23	311
GND‡	-	-	P37	-	-	GND*	-
I/O	-	-	-	C12	P258	T26	314
I/O	-	-	-	D12	P257	T25	317
I/O	-	D11	P38	E12	P256	T23	320
I/O	-	D12	P39	B13	P255	V26	323
VCC	-	-	P40	A16	P253	VCC*	-
I/O	P33	B11	P41	A14	P252	U24	326
I/O	P34	A12	P42	C13	P251	V25	329
I/O	P35	B12	P43	B14	P250	V24	332
I/O, FCLK2	P36	A13	P44	D13	P249	U23	335
GND	P37	C12	P45	A15	P248	GND*	-

XC4025E, /28EX/XL Pad Name	HQ 208	PG 223	HQ 240	PG 299	HQ 304	BG 352	Bndry Scan
I/O	-	-	-	B15	P247	Y26	338
I/O	-	-	-	E13	P246	W25	341
I/O	-	D13	P46	C14	P245	W24	344
I/O	-	D14	P47	A17	P244	V23	347
I/O	P38	B13	P48	D14	P243	AA26	350
I/O	P39	A14	P49	B16	P242	Y25	353
I/O	P40	A15	P50	C15	P241	Y24	356
I/O	P41	C13	P51	E14	P240	AA25	359
GND	-	-	-	-	-	GND*	-
VCC	-	-	-	-	-	VCC*	-
I/O	-	-	-	A18	P239	AB25	362
I/O	-	-	-	D15	P238	AA24	365
I/O	P42	B14	P52	C16	P237	Y23	368
I/O	P43	A16	P53	B17	P236	AC26	371
I/O	P44	B15	P54	B18	P235	AA23	374
I/O	P45	C14	P55	E15	P234	AB24	377
I/O	P46	A17	P56	D16	P233	AD25	380
I/O, SGCK2, GCK2	P47	B16	P57	C17	P232	AC24	383
O (M1)	P48	C15	P58	A20	P231	AB23	386
GND	P49	D15	P59	A19	P230	GND*	-
I (M0)	P50	A18	P60	C18	P229	AD24	389
VCC	P55	D16	P61	B20	P228	VCC*	-
I (M2)	P56	C16	P62	D17	P227	AC23	390
I/O, PGCK2, GCK3	P57	B17	P63	B19	P226	AE24	391
I/O (HDC)	P58	E16	P64	C19	P225	AD23	394
I/O	P59	C17	P65	F16	P224	AC22	397
I/O	P60	D17	P66	E17	P223	AF24	400
I/O	P61	B18	P67	D18	P222	AD22	403
I/O ($\overline{\text{LDC}}$)	P62	E17	P68	C20	P221	AE23	406
I/O	-	-	-	F17	P220	AE22	409
I/O	-	-	-	G16	P219	AF23	412
VCC	-	-	-	-	-	VCC*	-
GND	-	-	-	-	-	GND*	-
I/O	P63	F16	P69	D19	P218	AD20	415
I/O	P64	C18	P70	E18	P217	AE21	418
I/O	P65	D18	P71	D20	P216	AF21	421
I/O	P66	F17	P72	G17	P215	AC19	424
I/O	-	E15	P73	F18	P214	AD19	427
I/O	-	F15	P74	H16	P213	AE20	430
I/O	-	-	-	E19	P212	AF20	433
I/O	-	-	-	F19	P211	AC18	436
GND	P67	G16	P75	E20	P210	GND*	-
I/O	P68	E18	P76	H17	P209	AD18	439
I/O	P69	F18	P77	G18	P208	AE19	442
I/O	P70	G17	P78	G19	P207	AC17	445
I/O	P71	G18	P79	H18	P206	AD17	448
VCC	-	-	P80	F20	P204	VCC*	-
I/O	P72	H16	P81	J16	P203	AE18	451
I/O	P73	H17	P82	G20	P202	AF18	454

XC4025E, /28EX/XL Pad Name	HQ 208	PG 223	HQ 240	PG 299	HQ 304	BG 352	Bndry Scan
I/O	-	-	-	J17	P201	AE17	457
I/O	-	-	-	H19	P200	AE16	460
GND‡	-	-	P83	-	-	GND*	-
I/O	-	-	-	H20	P199	AF16	463
I/O	-	-	-	J18	P198	AC15	466
I/O	-	G15	P84	J19	P197	AD15	469
I/O	-	H15	P85	K16	P196	AE15	472
I/O	P74	H18	P86	J20	P195	AF15	475
I/O	P75	J18	P87	K17	P194	AD14	478
I/O	P76	J17	P88	K18	P193	AE14	481
I/O ($\overline{\text{INIT}}$)	P77	J16	P89	K19	P192	AF14	484
VCC	P78	J15	P90	L20	P191	VCC*	-
GND	P79	K15	P91	K20	P190	GND*	-
I/O	P80	K16	P92	L19	P189	AE13	487
I/O	P81	K17	P93	L18	P188	AC13	490
I/O	P82	K18	P94	L16	P187	AD13	493
I/O	P83	L18	P95	L17	P186	AF12	496
I/O	P84	L17	P96	M20	P185	AE12	499
I/O	P85	L16	P97	M19	P184	AD12	502
I/O	-	-	-	N20	P183	AC12	505
I/O	-	-	-	M18	P182	AF11	508
GND‡	-	-	P98	-	-	GND*	-
I/O	-	-	-	M17	P181	AE11	511
I/O	-	-	-	M16	P180	AD11	514
I/O	-	L15	P99	N19	P179	AF9	517
I/O	-	M15	P100	P20	P178	AD10	520
VCC	-	-	P101	T20	P177	VCC*	-
I/O	P86	M18	P102	N18	P175	AE9	523
I/O	P87	M17	P103	P19	P174	AD9	526
I/O	P88	N18	P104	N17	P173	AC10	529
I/O	P89	P18	P105	R19	P172	AF7	532
GND	P90	M16	P106	R20	P171	GND*	-
I/O	-	-	-	N16	P170	AE8	535
I/O	-	-	-	P18	P169	AD8	538
I/O	-	N15	P107	U20	P168	AC9	541
I/O	-	P15	P108	P17	P167	AF6	544
I/O	P91	N17	P109	T19	P166	AE7	547
I/O	P92	R18	P110	R18	P165	AD7	550
I/O	P93	T18	P111	P16	P164	AE6	553
I/O	P94	P17	P112	V20	P163	AE5	556
GND	-	-	-	-	-	GND*	-
VCC	-	-	-	-	-	VCC*	-
I/O	-	-	-	R17	P162	AD6	559
I/O	-	-	-	T18	P161	AC7	562
I/O	P95	N16	P113	U19	P160	AF4	565
I/O	P96	T17	P114	V19	P159	AF3	568
I/O	P97	R17	P115	R16	P158	AD5	571
I/O	P98	P16	P116	T17	P157	AE3	574
I/O	P99	U18	P117	U18	P156	AD4	577
I/O, SGCK3, GCK4	P100	T16	P118	X20	P155	AC5	580

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 max	—	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.

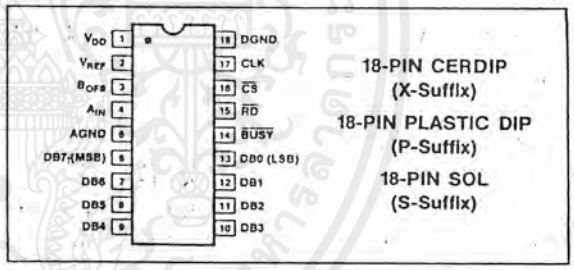
The ADC-908 outputs use 3-state logic, allowing direct connection to the data bus or system input port. Active-LOW chip select ($\overline{CS}$) and read/write ($\overline{RD}$) inputs are used to control all operations. This input structure permits the ADC-908 to be used as a memory-mapped input device.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

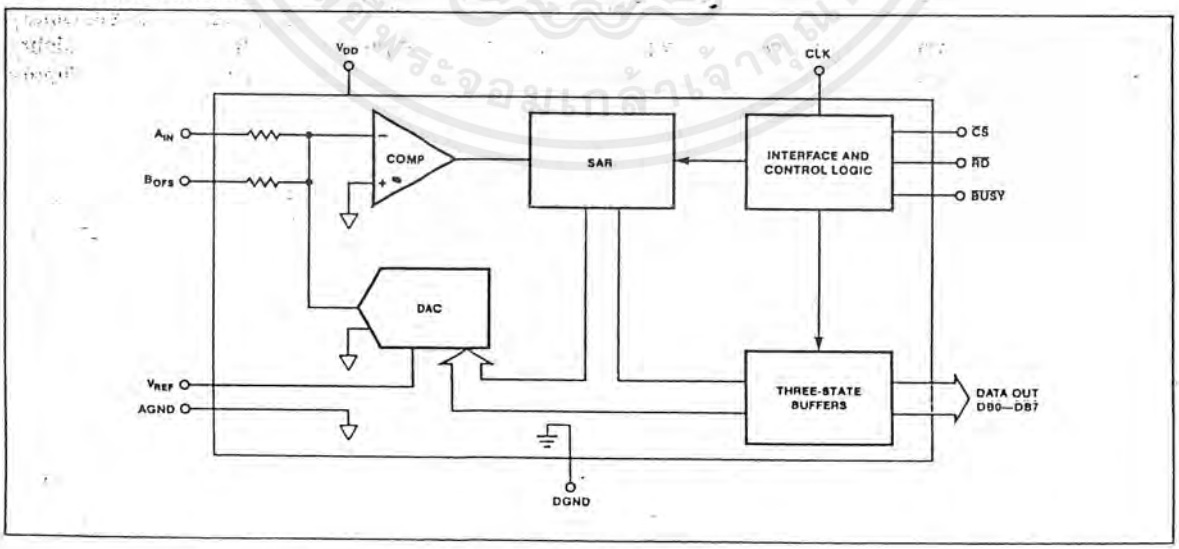


18-PIN CERDIP (X-Suffix)

18-PIN PLASTIC DIP (P-Suffix)

18-PIN SOL (S-Suffix)

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 T_A = Full Temp Range	— —	— —	1 10	µA
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 T_A = Full Temp Range	— —	— —	1 10	µA
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 T_A = Full Temp Range	— —	— —	1 10	µA
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 T_A = T_{MIN} T_A = T_{MAX}	60 50 90	— — —	— — —	ns
$\overline{RD}$ 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 T_A = +25°C T_A = T_{MIN} T_A = T_{MAX} $\overline{BUSY}$ Load = 100pF T_A = +25°C T_A = T_{MIN} T_A = T_{MAX}	 — — — — — —	 — — — — — —	 120 100 150 150 120 200	ns
$\overline{BUSY}$ to $\overline{RD}$ Setup Time (Notes 2, 6)	t_{BSR}		0	—	—	ns
$\overline{BUSY}$ to $\overline{CS}$ Setup Time (Note 6)	t_{BSCS}		0	—	—	ns

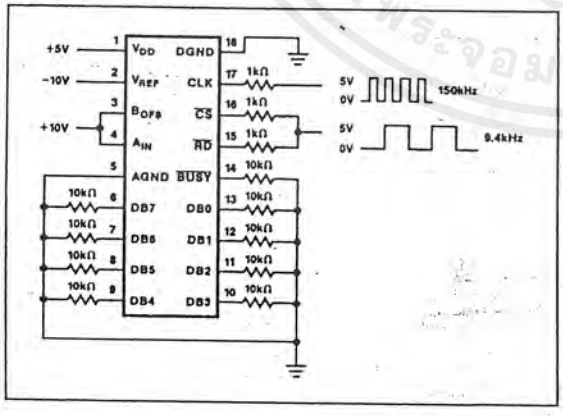
ELECTRICAL CHARACTERISTICS at V_{DD} = +5V, V_{REF} = −10V, Unipolar Configuration, R_{CLK} = 43kΩ, C_{CLK} = 100pF; −40°C ≤ T_A ≤ +85°C for ADC-908E/F, 0°C ≤ T_A ≤ +70°C for ADC-908G, −55°C ≤ T_A ≤ +125°C for ADC-908A/B, unless otherwise noted. *Continued*

PARAMETER	SYMBOL	CONDITIONS	ADC-908 MIN	TYP	MAX	UNITS
Data Access Time (Note 6)	t_{RAD}	C_L = 20pF				
		T_A = +25°C	—	—	140	ns
		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				
		T_A = +25°C	—	—	600	ns
		T_A = T_{MIN}	—	—	400	
		T_A = T_{MAX}	—	—	800	

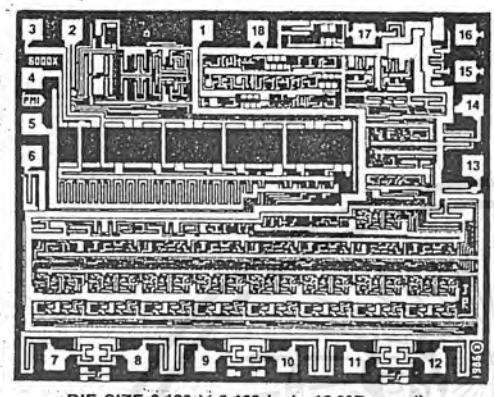
NOTES:

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 × 0.103 inch, 13,287 sq. mils
(3.28 × 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, Section 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

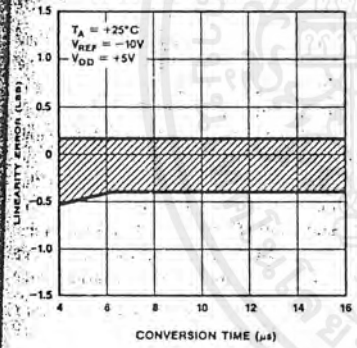
WAFER TEST LIMITS at V_{DD} = +5V, V_{REF} = −10.000V, AGND = DGND = 0V, T_A = +25°C, unless otherwise noted. (Continued)

PARAMETER	SYMBOL	CONDITIONS	ADC-908 LIMIT	UNITS
DIGITAL OUTPUTS				
Output HIGH Voltage BUSY, DB0-7	V_{OH}	I_{SOURCE} = 40μA	4	V MIN
Output LOW Voltage 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

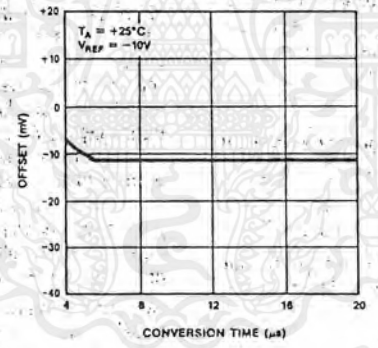
NOTE:
Electrical tests are performed at wafer probe to the limits shown. Due to variations in assembly methods and normal yield loss, yield after packaging is not guaranteed for standard product dice. Consult factory to negotiate specifications based on dice lot qualification through sample lot assembly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

TYPICAL PERFORMANCE CHARACTERISTICS

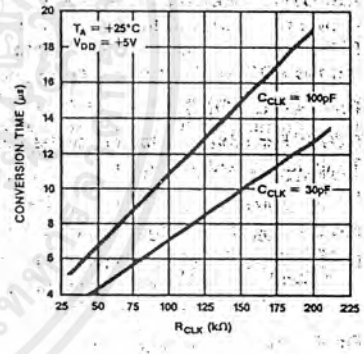
LINEARITY ERROR vs CONVERSION TIME



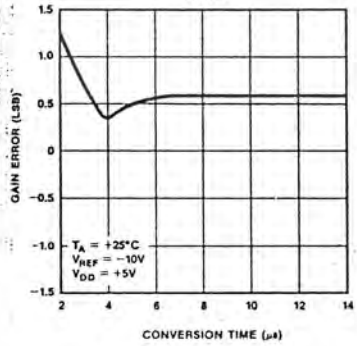
OFFSET ERROR vs CONVERSION TIME RAM MODE



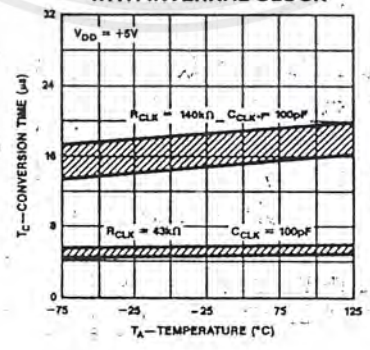
CONVERSION TIME vs R_{CLK} AND C_{CLK}



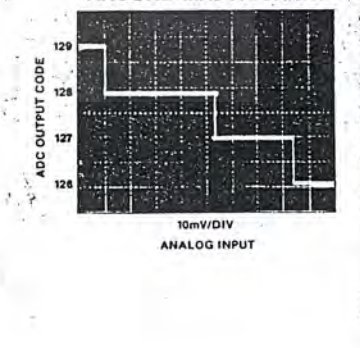
GAIN ERROR vs CONVERSION TIME



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

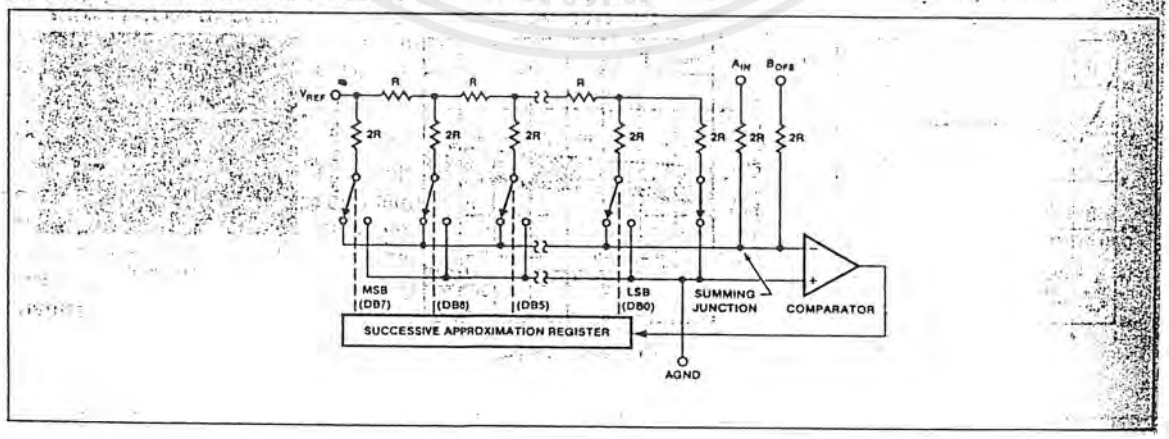
The ADC-908 may be interfaced as if it were a static RAM, a ROM, or a slow-memory device. Each of these interface modes has its own timing and software requirements as described below. These requirements must be rigidly met, as improper timing may cause the ADC-908 to change modes.

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

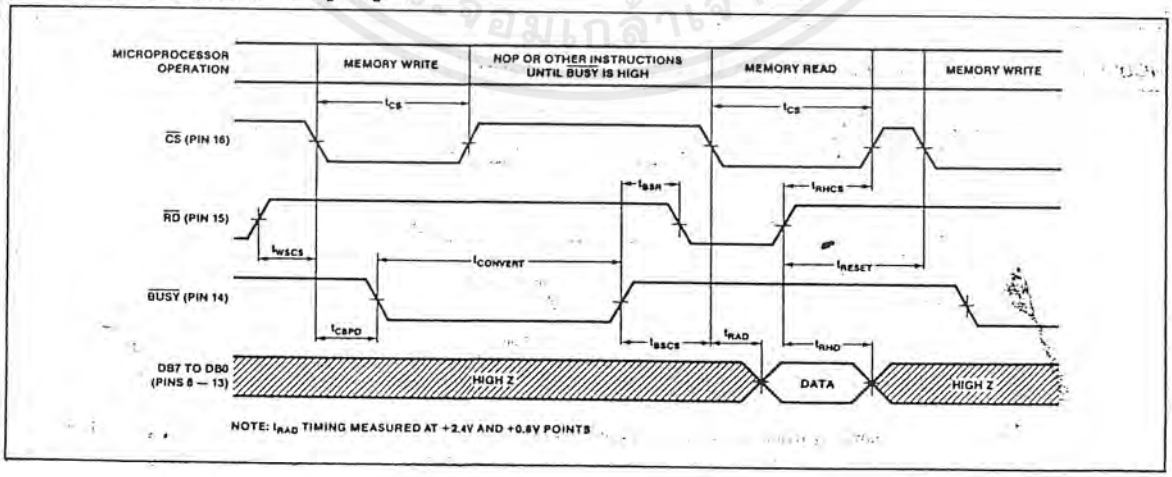
NOTE 1: If $\overline{RD}$ goes LOW to HIGH, the ADC is internally reset, regardless of the states of $\overline{CS}$ or $\overline{BUSY}$.

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)

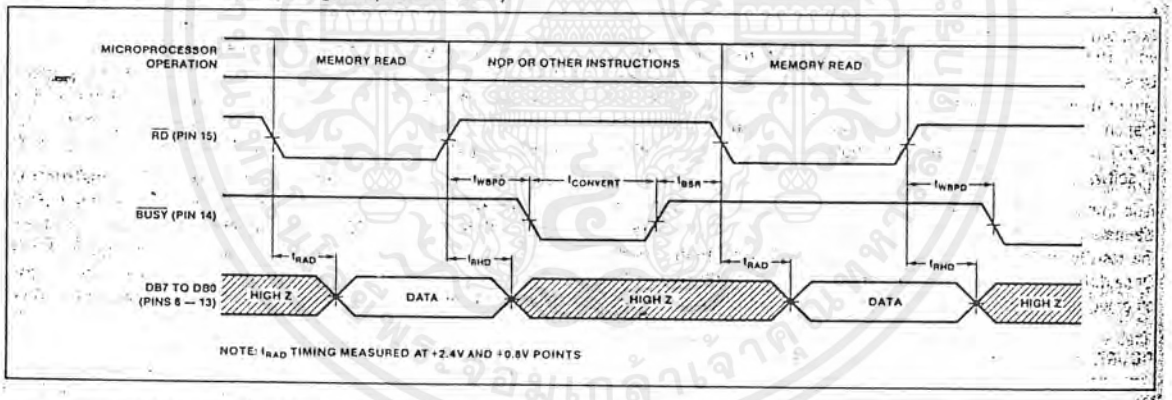
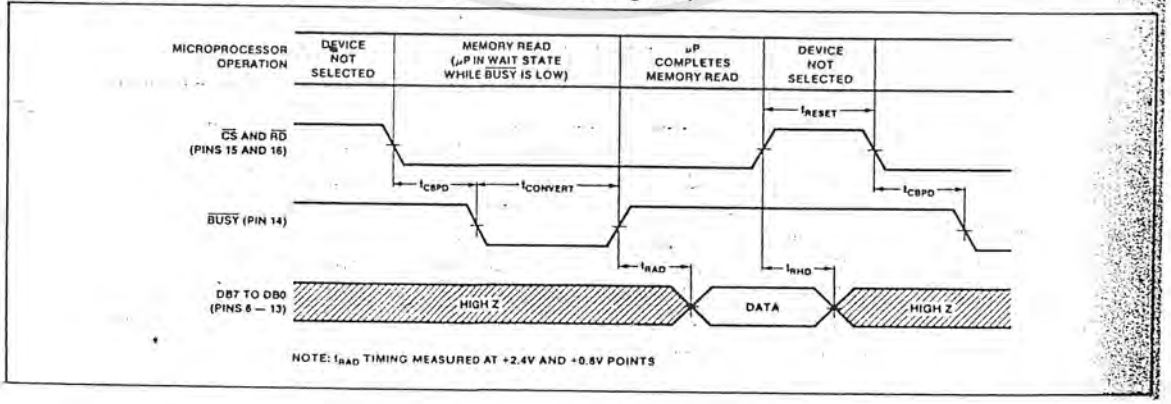


FIGURE 4: Slow-Memory Mode Timing Diagram ($\overline{CS}$ and $\overline{RD}$ Tied Together)



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

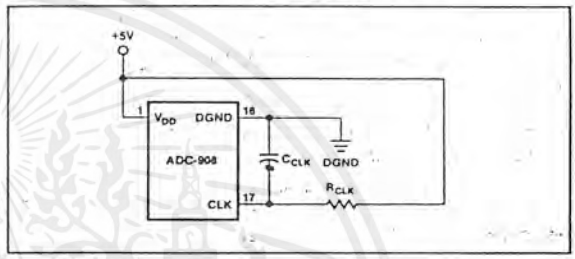


FIGURE 6: Using an External Clock

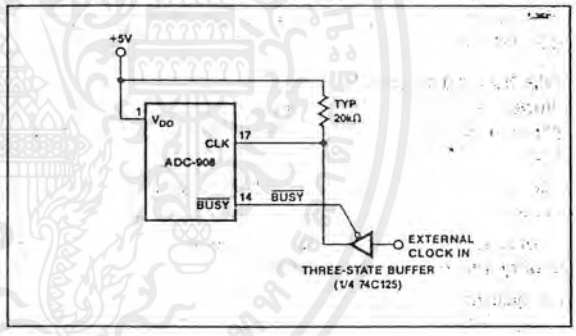
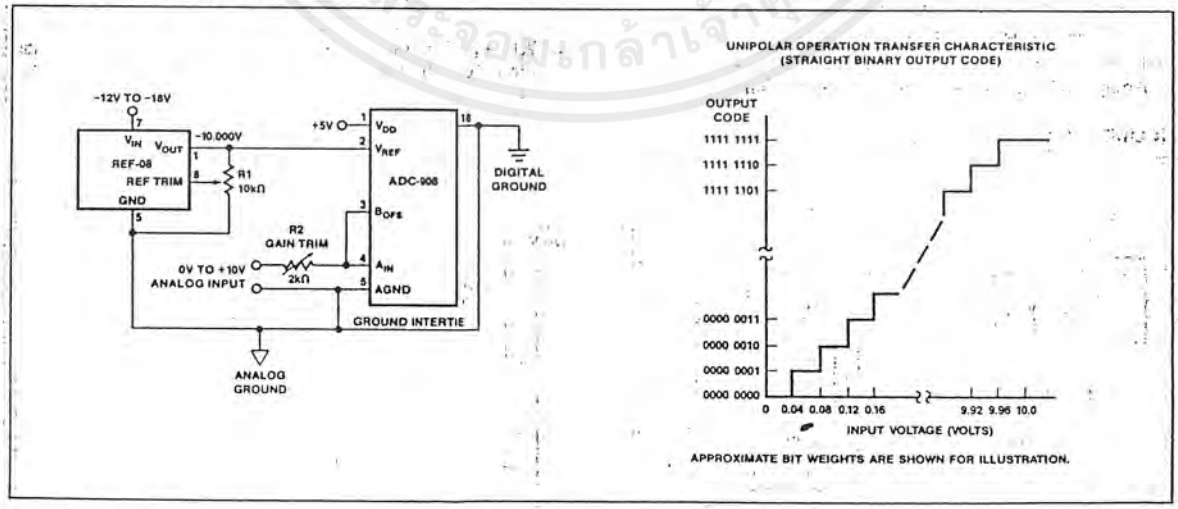


FIGURE 7: Unipolar Operation



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.

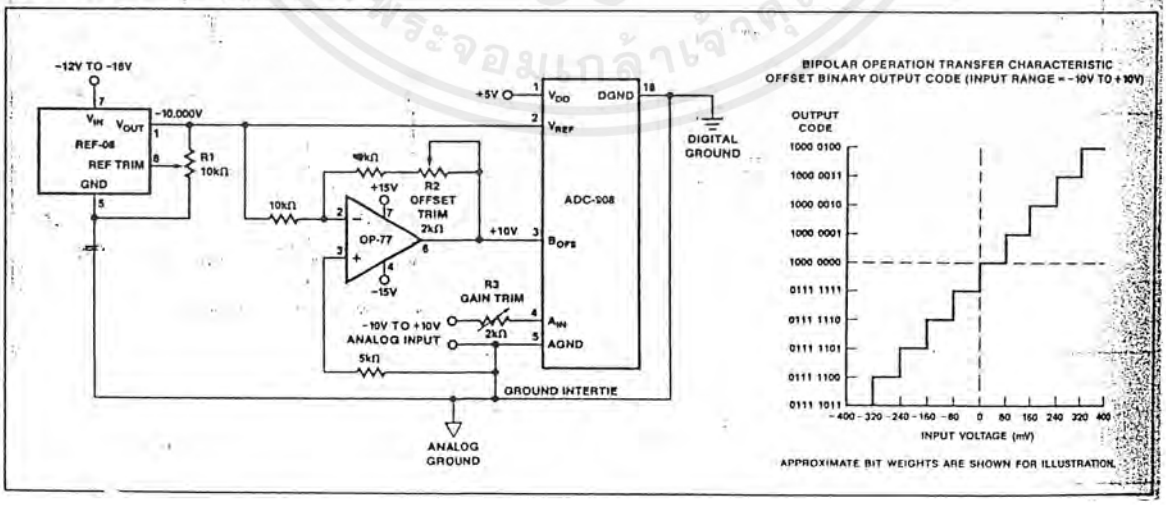
Complementary Offset Binary—Figure 9 shows a complementary offset binary circuit. In this bipolar mode, the +10V to −10V analog input is conditioned to a 0 to +10V signal range for normal unipolar conversion.

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.

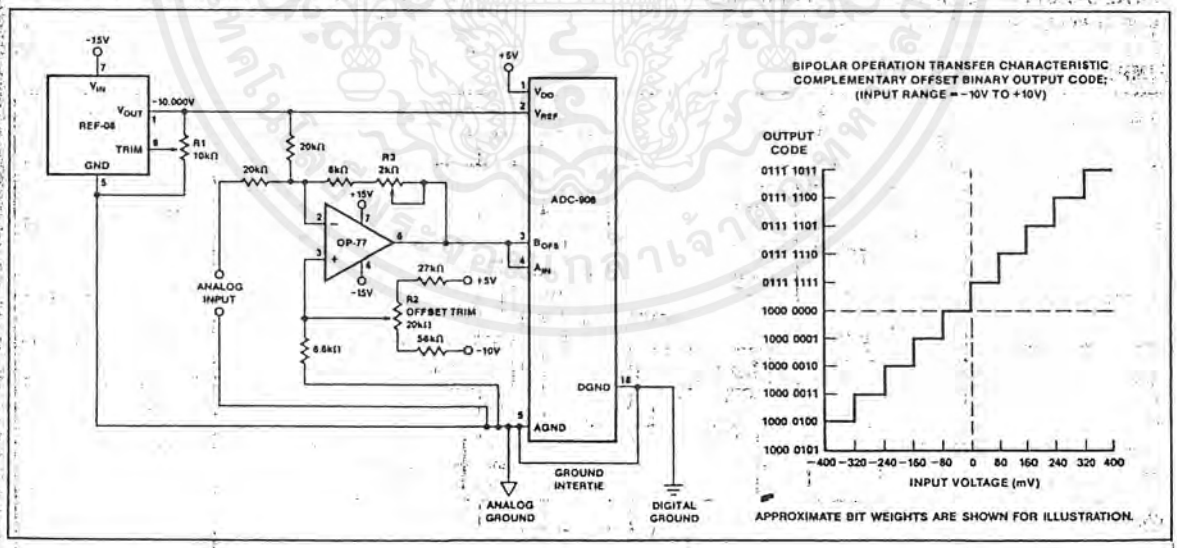
Offset Correction—Conversion offset errors may be corrected by counter-offsetting the buffer amplifier driving A_{IN}. This offset correction may be accomplished by applying a correction current to the buffer's summing junction or by tapping a voltage divider sitting between V_{DD} and V_{REF}, and applying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

ดาบชัย พัวพานิชและคณะ. "การพัฒนาตัวเข้ารหัสเสียงโดยวิธีการควอนไตซ์เวกเตอร์แบบมาตรฐานและอุปกรณ์ FPGAs." ปริญญานิพนธ์ครุศาสตร์อุตสาหกรรม,สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2540

บุญยอด แสงคุณธรรมและคณะ. "การเข้ารหัสเสียงโดยวิธีการควอนไตซ์เวกเตอร์ด้วยอุปกรณ์ FPGAs."ปริญญานิพนธ์ครุศาสตร์อุตสาหกรรม,สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2540

Xilink. The Programmable Logic data Book :3 rd ed. Xilink Inc 1994

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นางสาวปราณี ทิพย์สุวรรณ์
วันเดือนปีเกิด	10 เมษายน 2519
สถานที่เกิด	จังหวัดพัทลุง
ภูมิลำเนาเดิม	105 หมู่ 5 ต. คลองเฉลิม อ. กงหรา จ. พัทลุง 93180
ที่อยู่ปัจจุบัน	397/1 หมู่ 1 ซ. จินดานิเวศน์ 10 เขตลาดกระบัง
	กรุงเทพ ฯ 10520
โทรศัพท์	(02) 3268456 ต่อห้อง 1 ใหม่

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนบ้านพูด กรป. กลาง
มัธยมศึกษาตอนต้น	โรงเรียนกงหราพิชากร
ประกาศนียบัตรวิชาชีพ	วิทยาลัยเทคนิคพัทลุง
ประกาศนียบัตรวิชาชีพชั้นสูง	วิทยาลัยเทคนิคปัตตานี
ปริญญาตรี	สาขาอิเล็กทรอนิกส์และคอมพิวเตอร์
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	พ.ศ. 2534 - 2538 ทุนมูลนิธิร่วมจิตน้อมเกล้าเพื่อเยาวชน
	พ.ศ. 2539 - 2540 ทุนมูลนิธิทิสโก้
คติพจน์	การเริ่มต้นที่ดี ไม่มีคำว่าสาย

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นางสาวศันสนีย์ กิมเกถนอม
วันเดือนปีเกิด	15 พฤศจิกายน 2521
สถานที่เกิด	สุราษฎร์ธานี
ภูมิลำเนาเดิม	38/82 หมู่บ้านแสงชัย ถ.ปทุม – กรุงเทพฯ
	ต. บางปรอก อ. เมือง จ .ปทุมธานี 12000
ที่อยู่ปัจจุบัน	397/1 หมู่ 1 ซ. จินดานิเวศน์ 10 เขตลาดกระบัง
	กรุงเทพ ฯ 10520
โทรศัพท์	(02) 3268456 ต่อ ห้อง 11 เก่า
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนพุทธยาศรม
มัธยมศึกษาตอนต้น	โรงเรียนคณะราษฎร์บำรุงปทุมธานี
มัธยมศึกษาตอนปลาย(ม.4)	โรงเรียนปทุมวิไล
ประกาศนียบัตรวิชาชีพชั้นสูง	สถาบันเทคโนโลยีราชมงคลวิทยาเขตเทคนิคนนทบุรี
ปริญญาตรี	สาขาอิเล็กทรอนิกส์และคอมพิวเตอร์
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	-
คติพจน์	ฝันให้ไกล แล้วไปให้ถึง

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายสุทธิพงศ์ ปราชญ์นคร
วันเดือนปีเกิด	17 มกราคม 2520
สถานที่เกิด	จังหวัดชุมพร
ภูมิลำเนาเดิม	19 หมู่ 11 ต. บางลึก อ. เมือง จ. ชุมพร 86000
ที่อยู่ปัจจุบัน	394/9 หมู่ 1 ซ. จินดานิเวศน์ 12 เขตลาดกระบัง
	กรุงเทพฯ 10520
โทรศัพท์	(02) 3268444 ต่อห้อง 5
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนบ้านหนองเนียน
มัธยมศึกษาตอนต้น	โรงเรียนศรียาภัย
ประกาศนียบัตรวิชาชีพ	วิทยาลัยเทคนิคชุมพร
ประกาศนียบัตรวิชาชีพชั้นสูง	สถาบันเทคโนโลยีราชมงคล วิทยาเขตเทคนิคกรุงเทพ ฯ
ปริญญาตรี	สาขาอิเล็กทรอนิกส์และคอมพิวเตอร์
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม
ผลงานที่ได้รับ	-
ทุนการศึกษา	-
คติพจน์	เก่งอย่างคนฉลาด คือทำเรื่องยากให้เป็นเรื่องง่าย