ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ การบีบอัดสัญญาณเสียงโดยกระบวนการ DCT

DCT Audio Compression

ชื่อนักศึกษา 1. นายเทพรัตน์ ภิรมย์รื่น รหัสประจำตัว 42035369

2. นายสุทธิศักดิ์ สุขัมศรี รหัสประจำตัว 42035383

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** เทคโนโลยีการวัดคุมทางอุตสาหกรรม

อาจารย์ที่ปรึกษา อาจารย์ไพบูลย์ พวงวงศ์ตระกูล

อาจารย์ที่ปรึกษาร่วม อาจารย์กิติพงค์ มะโน

คณะกรรมการสอบปริญญานิพนธ์		ลายมือชื่อ
1. อาจารย์ไพบูลย์	พวงวงศ์ตระกูล	.....
2. อาจารย์กิติพงค์	มะโน	.....
3. อาจารย์โกศล	ตราชู	.....
4. อาจารย์ปิยะ	จิตธรรมมาภิรมย์	.....
5. อาจารย์อมรชัย	ชัยชนะ	.....

วัน/เดือน/ปีที่สอบ วันเสาร์ที่ 9 ธันวาคม พ.ศ. 2543 เวลา 13.00 น.

สถานที่สอบ ห้อง ค.301 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.วิสุทธิ์ อธิพรธรรม)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 4 เดือน พ.ค. พ.ศ. ....

ปริญญานิพนธ์

การบีบอัดสัญญาณเสียงโดยใช้กระบวนการ DCT

Discrete Cosine Transform (DCT) Audio Compression

นายสุทธิศักดิ์ สุขัมศรี
นายเทพรัตน์ ภิรมย์รื่น

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาเทคโนโลยีการวัดคุมทางอุตสาหกรรม
ภาควิชาครุศาสตร์อุตสาหกรรม คณะครุศาสตร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

ปริญญานิพนธ์

เรื่อง การบีบอัดสัญญาณเสียงโดยใช้กระบวนการ DCT

Discrete Cosine Transform (DCT) Audio Compression

วัตถุประสงค์

1. เพื่อศึกษาหลักการและกระบวนการบีบอัดข้อมูลแบบ DCT และแบบต่างๆ
2. เพื่อศึกษาและประยุกต์ใช้งานอุปกรณ์โปรแกรมได้ (Programmable Devices)
3. เพื่อศึกษาและเขียนชุดคำสั่งภาษา VHDL ควบคุมการทำงานอุปกรณ์โปรแกรมได้
4. เพื่อนนำความรู้ที่ได้ศึกษามาประยุกต์ใช้กับการทำงานจริง

ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถเข้าใจหลักการและกระบวนการบีบอัดข้อมูล
2. สามารถประยุกต์ใช้งานอุปกรณ์โปรแกรมได้ในกระบวนการบีบอัดข้อมูล
3. สามารถเขียนชุดคำสั่งภาษา VHDL ในกระบวนการบีบอัดข้อมูลได้
4. สามารถนำความรู้มาประยุกต์ใช้งานจริงได้

ชื่อหัวข้อ	การบีบอัดสัญญาณเสียงโดยใช้กระบวนการ DCT
นักศึกษา	นายเทพรัตน์ ภิรมย์รื่น
	นายสุทธิศักดิ์ สุขัมศรี
อาจารย์ที่ปรึกษา	อาจารย์กิติพงค์ มะโน
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชา	เทคโนโลยีการวัดคุมทางอุตสาหกรรม
ปีการศึกษา	2543

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอการนำทฤษฎีการแปลงค่าดีสครีตโคไซน์มาใช้ในการบีบอัดสัญญาณเสียง โดยใช้อุปกรณ์โปรแกรมได้ FPGAs และภาษาบรรยายการทำงานของอุปกรณ์ VHDL โดยสามารถที่จะนำไปประยุกต์ใช้ได้กับการบีบอัดสัญญาณข้อมูลเชิงเลข หรือการสื่อสัญญาณได้

Thesis Title	Discrete Cosine Transform (DCT) Audio Compression
Students	Mr. Theppharat Phiromruen
	Mr. Suttisak Sukhamsri
Advisor	Mr. Kitipong Mano
Education Level	Bachelor of Science in Industrial Education
Program in	Industrial Instrument Technology
Acadamic	2000

ABSTRACT

This thesis presents an implementation of Discrete Cosine Transform (DCT) theorem. In this implementation used Programmable Logic Device FPGAs and hardware description Language VHDL to implement in audio compression. This project can be applied to use in compressing a digital data or in communication.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ ประสบความสำเร็จและเป็นไปได้ด้วยดีนั้น ทางคณะผู้จัดทำขอขอบพระคุณท่านอาจารย์ที่ปรึกษา คือ อาจารย์กิติพงค์ มะโน และอาจารย์โกศล ตราชู ที่ได้ให้คำปรึกษาเกี่ยวกับเรื่องงาน ไม่ว่าจะเป็นทาง Hardware หรือ Software ก็ตาม ตลอดจนคณาจารย์ทุกท่านในภาควิชาครุศาสตร์วิศวกรรม ที่ได้ให้ความอำนวยความสะดวกไม่ว่าจะเป็นทางด้านข้อมูล และ อุปกรณ์ที่เป็นประโยชน์ต่อการจัดทำปริญญานิพนธ์ ด้วยความกรุณาอย่างยิ่ง

สุดท้ายนี้ขอกราบขอบพระคุณ คุณพ่อ คุณแม่ ซึ่งให้พวกเราได้รับการศึกษามาตั้งแต่อดีตจนถึงในขณะนี้ อนึ่งคุณประโยชน์ตลอดจนคุณความดีที่เกิดจากปริญญานิพนธ์ฉบับนี้ ขอมอบให้ แด่ คุณพ่อ คุณแม่ คุณครูบาอาจารย์ที่ได้สิทธิประสาทวิชาความรู้ทั้งหลาย มาครั้นแต่อดีตจนกระทั่งถึงปัจจุบัน ตลอดบันดาเพื่อนฝูงที่ร่วมทำงานด้วยกัน แม้ว่าพวกเราอาจมีปัญหาในการทำงานก็ตาม แต่ด้วยความช่วยเหลือของทุกท่านดังที่ได้กล่าวมาข้างต้นทำให้การจัดทำปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมา และความสำคัญของปริญญานิพนธ์

ในปัจจุบัน ความก้าวหน้าทางเทคโนโลยี โดยเฉพาะในวงการอิเล็กทรอนิกส์นับว่ามีการพัฒนาไปอย่างรวดเร็ว ทั้งนี้เนื่องจากการพัฒนาทางด้านกายภาพของอุปกรณ์สารกึ่งตัวนำให้มีคุณสมบัติสำคัญ ที่มีคุณลักษณะการทำงานได้ตามความต้องการ รวมทั้งยังผนวกการเขียนโปรแกรมเพื่อควบคุมการทำงานของอุปกร์เหล่านั้นให้มีความสามารถทำงานในการทำงานที่ซับซ้อนได้ และในการทำงายที่ซับซ้อนอย่างเช่น การประมวลผลสัญญาณ ซึ่งเหมาะเป็นอย่างมากที่จะใช้อุปกรณ์โปรแกรมได้จำพวก FPGAs นำมาเป็นส่วนประมวลผลสัญญาณนั้น ๆ โดยในโครงงานชิ้นนี้ได้นำกรณีการบีบอัดสัญญาณเสียงโดยกระบวนการ DCT ซึ่งในปัจจุบันถือได้ว่าเป็นการประมวลผลสัญญาณทางมัลติมีเดียที่กำลังได้รับความนิยม ทั้งการใช้งาน และการพัฒนาอย่างกว้างขวาง ซึ่งทางผู้จัดทำ ได้เล็งเห็นถึงความสำคัญในการเรียนรู้กระบวนการในจุดนี้ จึงได้นำเรื่องนี้ขึ้นมาศึกษา ดังในปริญญานิพนธ์ฉบับนี้

1.2 ขีดความสามรถของโครงงาน

โครงงานนี้มีขีดความสามารถดังนี้

1. ใช้อุปกรณ์โปรแกรมได้ FPGAs ทำหน้าที่ในส่วนของการประมวลผลสัญญาณ
2. ใช้กับสัญญาณเสียงขนาด 8 บิต
3. มีส่วนการการแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก ขนาด 8 บิต
4. ในส่วนการประมวลผลสัญญาณจะมีส่วน DCT และ IDCT

1.3 เนื้อหาโดยสังเขป

เนื้อหาภายในปริญญานิพนธ์ฉบับนี้แบ่งออกเป็นบทต่าง ๆ เพื่อสะดวกต่อการศึกษา และทำความเข้าใจ ในแต่ละบท จะประกอบด้วยเนื้อหาดังต่อไปนี้

บทที่ 2 ทฤษฎีและหลักการ ประกอบไปด้วยเนื้อหาดังนี้ คือ ทฤษฎีการแปลงสัญญาณ , การแปลงดิสครีตโคไซน์ , การแปลงกลับดิสครีตโคไซน์ , การแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก , การแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล , ทฤษฎีการสุ่มสัญญาณ และทฤษฎีและหลักการของ FPGAs

บทที่ 3 การออกแบบ การสร้าง และการทำงาน กล่าวถึงวิธีการสร้างในส่วนต่าง ๆ ของโครงงาน ได้แก่ การสร้างในส่วนวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล , วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก , วงจรกำเนิดสัญญาณนาฬิกา , การเชื่อมต่อวงจรการทำงานเข้าด้วยกัน , วงจรภาคจ่ายไฟ , การคำนวณหาค่าสัมประสิทธิ์ของการแปลง DCT , การคำนวณหาค่าสัมประสิทธิ์ของการแปลง IDCT , การปรับปรุงค่าสัมประสิทธิ์ของการแปลง DCT และ IDCT , การเขียนโปรแกรมส่วน Buffer , การเขียนโปรแกรมส่วน Parallel in Serial out , การเขียนโปรแกรมส่วน DCT , การเขียนโปรแกรมส่วน IDCT และการเขียนวงจร Schematic

บทที่ 4 การทดลอง และผลการทดลอง ประกอบไปด้วยการทดลองการทำงานของโครงงาน ดังนี้ การทดสอบการทำงานของวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และวงจรแปลงสัญญาณดิจิตอล เป็นสัญญาณแอนะลอก , การจำลองการทำงานส่วนโปรแกรม Buffer , การจำลองการทำงานส่วนโปรแกรม Parallel in Serial out , การจำลองการทำงานส่วนโปรแกรม DCT , การจำลองการทำงานส่วนโปรแกรม IDCT , การทดลองโดยใช้โปรแกรมสำเร็จรูป Wave Lab 2.0 ช่วยวิเคราะห์ และการทดลองจากการฟังเสียง

บทที่ 5 บทสรุป ปัญหา แนวทางแก้ไข และพัฒนา ได้รวมรวมปัญหาที่เกิดขึ้นในขั้นตอนต่าง ๆ ระหว่างการจัดทำโครงงาน และได้เสนอแนวทางการแก้ไขปัญหา และการพัฒนาเพื่อให้โครงงานมีประสิทธิภาพมากยิ่งขึ้น

ภาคผนวก ก เครื่องต้นแบบ

ภาคผนวก ข วงจรการทำงาน

ภาคผนวก ค โปรแกรมการทำงาน

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

การแปลงดีสครีตโคไซน์ (Discrete Cosine Transform : DCT) เป็นหนึ่งในกระบวนการการประมวลผลสัญญาณเชิงเลข (Digital Signal Processing : DSP) ซึ่งเป็นการแปลงสัญญาณเชิงเลขที่อยู่ในรูปโดเมนของเวลา ให้อยู่ในรูปโดเมนของโคไซน์ ซึ่งผลที่ได้จากการแปลงโดเมนจะทำให้ลดขนาดของข้อมูล ซึ่งเป็นกระบวนการในการบีบอัดสัญญาณ (Signal Compressing) ซึ่งสามารถแปลงกลับค่าโดเมนของโคไซน์ (Inverse Discrete Cosine Transform : IDCT) ทำให้ได้ข้อมูลกลับเช่นเดิม

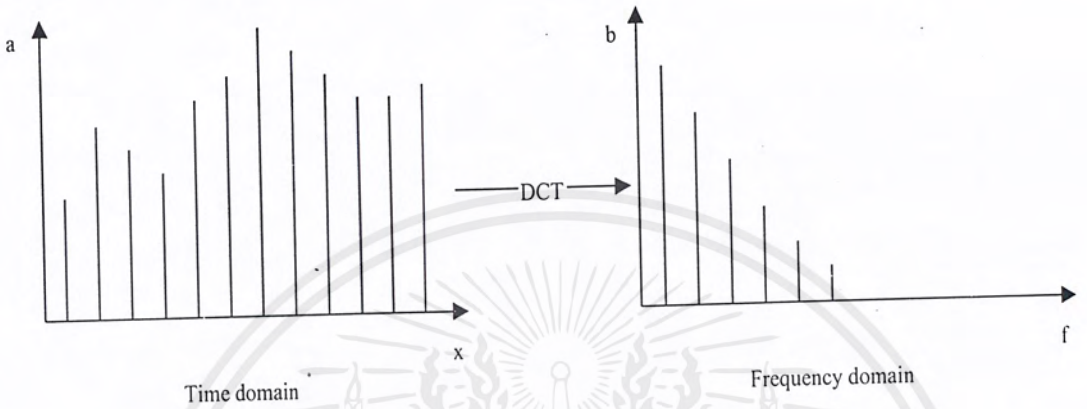
2.2 ทฤษฎีการแปลงสัญญาณ

รูปแบบของสัญญาณไฟฟ้าโดยมากมักอยู่ในรูปสัญญาณแอนะลอก การนำเอาสัญญาณไฟฟ้ามาประมวลผล เพื่อให้เกิดรูปแบบที่ต้องการนั้น ต้องใช้อุปกรณ์ทางแอนะลอกแต่ปัจจุบันนี้เทคโนโลยีทางด้านดิจิตอลก้าวหน้าไปมาก ทำให้การประมวลผลสัญญาณทางดิจิตอล สามารถทำได้อย่างรวดเร็ว และมีประสิทธิภาพ

ดังนั้นการแปลงรูปแบบสัญญาณ (Conversion) จึงมีความจำเป็นในการแปลงสัญญาณแอนะลอกที่มีอยู่แล้วให้เป็นสัญญาณดิจิตอล โดยอุปกรณ์การแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และจะถูกประมวลผลโดยตัวประมวลผลสัญญาณดิจิตอล เช่น คอมพิวเตอร์ เป็นต้น จากผลลัพธ์ที่ได้อาจถูกนำมาแสดงผลโดยตรงเลย หรืออาจถูกแปลงให้อยู่ในรูปของสัญญาณแอนะลอกที่ใช้งานได้ การที่จะแปลงสัญญาณแอนะลอกเป็นดิจิตอลได้นั้น สามารถทำได้โดยใช้อุปกรณ์แปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก สำหรับระบบที่มีการประมวลผลข้อมูลทางดิจิตอลแสดงดังรูปที่ 2.1

จากรูปที่ 2.1 การเปลี่ยนแปลงทางกายภาพในลักษณะใดๆ ก็ตาม เช่น อุณหภูมิ ความดัน ความเร็ว จะถูกเปลี่ยนให้มาเป็นสัญญาณไฟฟ้าแบบแอนะลอก โดยทรานสดิวเซอร์ (Transducer) ทฤษฎีการสุ่มที่มีรูปแบบที่เหมาะสมกับลักษณะทางกายภาพนั้นๆ จากสัญญาณทางไฟฟ้าก็จะถูกปรับให้อยู่ในรูปแบบ และขนาดที่เหมาะสมก่อน โดยวงจรต่างๆ เช่น วงจรขยาย หรือวงจรกรองสัญญาณเป็นต้น วงจรแซมปลิ้งแอนด์โฮลด์ (Sampling and Hold) จะสุ่มขนาดของสัญญาณแอนะ

ลอกมาแล้วจะทำการโฮลด์สัญญาณนั้นไว้ชั่วขณะโดยไม่จำเป็นต้องใช้ วงจร ADC แล้วข้อมูลทางดิจิตอลจะถูกส่งต่อไปยังบัส (Bus) ระบบ จากนั้นตัวโปรเซสเซอร์ จะทำการประมวลผลข้อมูลกลับมาเพื่อควบคุมกิจการทางกายภาพของระบบโดยผ่านตัวกระทำทางกล (Analog Actuator)



รูปที่ 2.1 ระบบที่มีการประมวลผลข้อมูลทางดิจิตอล

2.2.1 การแปลงดีสครีตโคไซน์ (Discrete Cosine Transform : DCT)

เป็นการลดขนาดข้อของมูล โดยการแปลงสัญญาณข้อมูลจากโดเมนของเวลาไปสู่โดเมนของโคไซน์ โดยการผลรวมของผลคูณของโคไซน์ ดังสมการ (2.1)

$$F(u)=\frac{1}{\sqrt{2N}}C(u)\sum_{i=0}^{N-1}f(i)\cos\left[\frac{(2i+1)\pi u}{2N}\right] \qquad (2.1)$$

โดย

$F(u)$ เป็นสัมประสิทธิ์ของการแปลง

$f(i)$ ข้อมูลเริ่มต้น

$$C(u)=\begin{cases}\frac{1}{\sqrt{2}};u=0\\1;n\neq 0\end{cases}$$

ข้อมูลที่ได้หลังจากการหาผลรวมของผลคูณของค่าตั้งต้นกับค่าสัมประสิทธิ์โคไซน์ จะมีขนาดของข้อมูลก่อนทำการแปลงค่าโดยที่การแปลงสัญญาณข้อมูลให้อยู่ในรูปของโดเมนโคไซน์จะมีลักษณะเด่นอยู่ 3 ประการ ดังนี้

1. ค่าความถี่ศูนย์จะเป็นค่าของความเข้มเฉลี่ยของข้อมูล
2. ค่าความถี่สูง เป็นค่าที่บอกถึงข้อมูลที่มีการเปลี่ยนแปลงสูง
3. ค่าวามถี่ต่ำ เป็นค่าที่บอกรานละเอียดโดยรวมของข้อมูล

2.2.2 การแปลงกลับดีสครีตโคไซน์ (Inverse Discrete Cosine Transform)

เป็นการแปลงสัญญาณที่อยู่ในรูปโดเมนของโคไซน์ให้กลับไปอยู่ในรูปโดเมนของเวลาเช่นเดิม โดยมีสมการการแปลงค่ากลับดังสมการ (2.2)

$$f(i)=\frac{1}{\sqrt{2N}}\sum_{u=0}^{N-1}C(u)F(u)\cos\left[\frac{(2i+1)\pi u}{2N}\right] \tag{2.2}$$

โดย

$F(u)$ เป็นสัมประสิทธิ์ของการแปลง

$f(i)$ ข้อมูลเริ่มต้น

$$C(u)=\begin{cases}\frac{1}{\sqrt{2}};u=0\\1;n\neq 0\end{cases}$$

ซึ่งทั้งในกระบวนการการแปลงค่าไปสู่โดเมนของโคไซน์ก็ดี หรือการแปลงค่ากลับไปก็ดี จะมีส่วนของการประมาณค่าของข้อมูลที่ได้ ซึ่งในส่วนนี้เองที่เป็นการสูญเสีย ซึ่งอัตราการสูญเสียจะขึ้นอยู่กับการกระจายของค่าที่นำเข้ามาแปลง โดยจะส่งผลทำให้ค่าหลังจากการแปลงกลับนั้นมีความเพี้ยนไปจากข้อมูลนำเข้าบ้าง แต่โดยในภาพรวมของกลุ่มข้อมูลยังถือว่า ข้อมูลยังในรูปลักษณะที่เกาะกลุ่มกัน ซึ่งค่าของอัตราการสูญเสียนี้ ก็จะขึ้นอยู่กับสาเหตุต่างๆ ดังนี้

1) การจัดและปรับแต่งค่าสัมประสิทธิ์ของโคไซน์

ในการนำค่าสัมประสิทธิ์ของโคไซน์ไปใช้งาน สามารถที่จะปรับแต่งค่าต่างๆในตารางสัมประสิทธิ์ให้เหมาะสมกับขนาดและกลุ่มของข้อมูล

2) การประมาณค่าของผลลัพธ์จากการแปลงค่า

ผลลัพธ์ที่ได้จารกการแปลงค่า ในบางครั้งจะอยู่ในรูปของทศนิยม โดยส่วนมากแล้วจะตัดค่าของตัวเลขนำตัวเลขในส่วนของจำนวนเต็มเท่านั้นที่มาทำการจัดเก็บ ซึ่งก็ทำให้ข้อมูลหลักจากทำการแปลงค่ากลับมีความเพี้ยนเกิดขึ้น

2.3 การแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก และการแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล

การติดต่อระหว่างมนุษย์ส่วนมากจะใช้สัญญาณต่อเนื่อง (Analog) เป็นสัญญาณติดต่อกัน แต่ว่าการทำงานของระบบคอมพิวเตอร์ จะใช้สัญญาณเป็นช่วงดิจิตอลเป็นสัญญาณในการทำงาน ดังนั้นถ้าเราต้องการให้คอมพิวเตอร์ช่วยเราทำงานแล้ว เราจะต้องเปลี่ยนสัญญาณแอนะลอกที่เราใช้อยู่ให้เป็นสัญญาณดิจิตอลเพื่อให้คอมพิวเตอร์หรือเครื่องประมวลสัญญาณดิจิตอลรับรู้เมื่อประมวลสัญญาณเสร็จก็จะส่งข้อมูลออกมาเป็นสัญญาณดิจิตอล ซึ่งเป็นเรื่องที่ยุ่งยากที่เราจะเข้าใจข้อมูลนั้น ดังนั้นพอสรุปได้ว่าการเปลี่ยนสัญญาณแอนะลอกเป็นสัญญาณดิจิตอลและการเปลี่ยนแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอกซึ่งเป็นการประสานโลกของคอมพิวเตอร์ให้เข้ากับโลกมนุษย์ได้ เพื่อให้มนุษย์ได้ใช้คอมพิวเตอร์ได้อย่างสมบูรณ์ยิ่งขึ้น การเปลี่ยนแปลงสัญญาณแอนะลอกไปเป็นดิจิตอลเรียกว่า Analog to Digital Converter (ADC) หรือเรียกย่อๆว่า A TO D หรือ A/D ในทำนองเดียวกันการเปลี่ยนแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอกเรียกว่า Digital to Analog Converter (DAC หรือ D/A)

2.3.1 การแปลงสัญญาณดิจิตอลและสัญญาณแอนะลอก

อุปกรณ์ที่ภายในใช้สัญญาณดิจิตอลทั้งหมด เช่น นาฬิกาดิจิตอล เครื่องคิดเลข และ คอมพิวเตอร์ มีน้อยมาก อุปกรณ์ส่วนใหญ่ยังต้องรับอินพุต และส่งเอาต์พุตในรูปของสัญญาณ แอนะลอกอยู่ ดังนั้นการแปลงสัญญาณแอนะลอกเป็นดิจิตอลซึ่งเรียกว่า A/D Converter และการแปลงสัญญาณดิจิตอลเป็นแอนะลอกหรือที่เรียกว่า D/A Converter จึงเป็นเรื่องสำคัญของวงจรดิจิตอล ถ้าเรามีทั้งวงจร A/D และ D/A ครบแล้ว การสร้างอุปกรณ์ทางอิเล็กทรอนิกส์ไม่ว่าจะซับซ้อนเท่าใดก็สามารถทำได้ง่ายขึ้น อุปกรณ์จะใช้วงจร A/D ในการแปลงสัญญาณอินพุตต่าง ๆ ที่เป็นสัญญาณแอนะลอกให้เป็นดิจิตอลป้อนสัญญาณดิจิตอลที่ได้เข้าในวงจรดิจิตอลเพื่อประมวลผลจากนั้นแสดงเป็นตัวเลขและใช้วงจร D/A แปลงสัญญาณแอนะลอกใหม่เพื่อเป็นเอาต์พุตนำออกไปใช้ในการควบคุมกระบวนการภายนอกได้

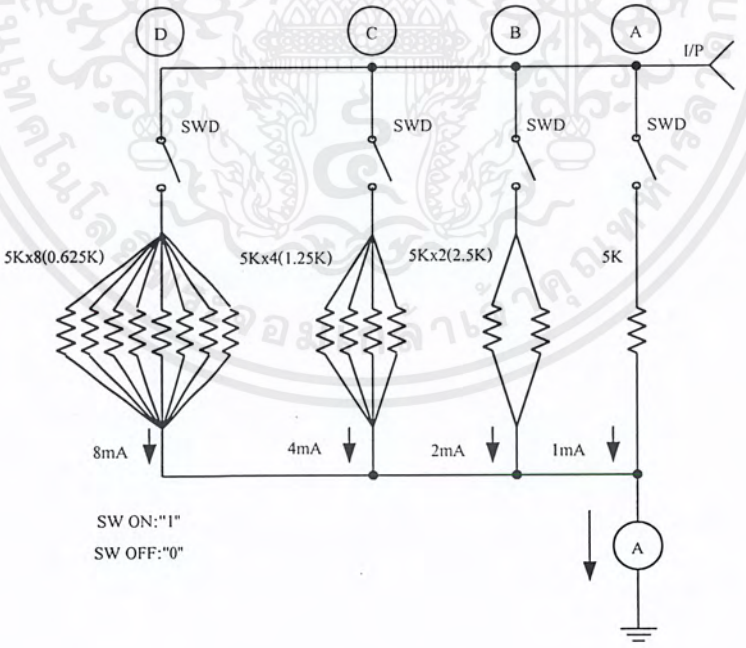
วงจร D/A มีด้วยกันหลายแบบด้วยกัน เราคงไม่สามารถพูดอธิบายได้ทุกแบบในที่นี้จะขอกล่าววงจร D/A แบบรวมกระแสกับแบบวงจรบันไดของตัวต้านทาน

1) วงจร D/A แบบรวมกระแส

ลองต่อวงจรในรูปที่ 2.2 เป็นหลักการของวงจร D/A แบบรวมกระแส (Current Summing Type) นี้เราจะใช้สวิตช์ 4 ตัว ใช้ตัวต้านทาน 5 กิโลโอห์ม จำนวน 15 ตัว และมิเตอร์วัดกระแสพิสัยไม่เกิน 20 มิลลิแอมป์ สวิตช์ทั้ง 4 ตัวนี้คือ SWA, SWB, SWC, และ SWD ซึ่งเปรียบเสมือนปริมาณดิจิตอลแต่ละบิต

เมื่อปิดสวิตช์หมายถึง "1" ถ้าเปิดสวิตช์หมายถึง "0" เมื่อสวิตช์ทั้งสี่แทนเลขฐานสอง SWA จะเป็นบิตนัยสำคัญต่ำสุด (บิตต่ำสุดทางขวามือ) และ SWD จะเป็นบิตสำคัญสูงสุด

เมื่อปิดสวิตช์แต่ละตัวจะมีกระแสขนาดต่างกันไหลมาที่มิเตอร์วัดกระแส เช่น ปิดสวิตช์ SWA จะมีกระแสไหล 1 มิลลิแอมป์ ปิดสวิตช์ SWB จะมีกระแสไหล 2 มิลลิแอมป์ เป็นต้น ถ้าปิดสวิตช์หลายสวิตช์พร้อมกันจะมีกระแสไหลจากแต่ละกิ่งมารวมกันแล้วไหลไปยังมิเตอร์และกระแสที่ไหลมาที่มิเตอร์นี้คือ ปริมาณแอนะลอกที่แปลงมาได้จากวงจรนี้นั่นเอง



รูปที่ 2.2 การทดลองวงจร D/A แบบรวมกระแส

สวิตช์ทั้งสี่เรียงกันเป็นเลขฐานสอง 4 บิต ดังนั้นตัวเลขที่สามารถแสดงได้คือ 0-15 เช่นเดียวกัน กระแสที่เป็นปริมาณแอนาลอกก็สามารถมีค่าได้ระหว่าง 0-15 มิลลิแอมป์ เลขฐานสองเพิ่มทีละ 0001 ในขณะที่กระแสเพิ่มทีละ 1 มิลลิแอมป์ สวิตช์แต่ละตัวจะมี "น้ำหนัก" หรือขนาดกระแสที่บิตออกมาได้ต่างกันคือ SWA มีน้ำหนัก 1,SWB มีน้ำหนัก 2,SWC มีน้ำหนัก 4 และSWD 8 น้ำหนักที่เพิ่มทีละเท่าขึ้นมาเรื่อยๆ ตามหลักการของเลขฐานสองนั่นเอง

เราใช้ตัวต้านทานขนาด 5 กิโลโอห์มเพราะเราใช้แหล่งจ่ายไฟ 5 โวลต์และต้องการให้กระแสไหลผ่านตัวต้านท่านแต่ละตัวเพียง 1 มิลลิแอมป์เท่าๆ กัน เลือกใช้ตัวต้านทานเหมือนกัน 15 ตัว เพื่อให้ทดลองได้ง่ายและเข้าใจง่าย จะใช้ตัวต้านทาน 2.5 กิโลโอห์มเพียงตัวเดียวต่อที่ SWB หรือ 1.25 กิโลโอห์มต่อที่ SWC ก็ได้ แต่ค่าความต้านทานเหล่านี้หาได้ลำบากมาก

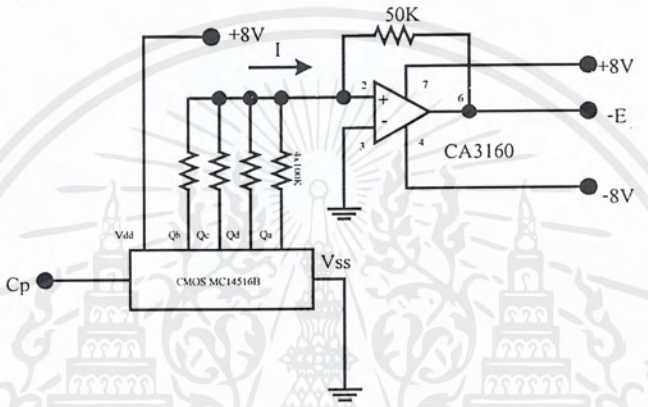
ลองเปิดสวิตช์ SWA จนถึง SWD ตามเลขฐานสองจะได้กระแสไหลผ่านมิเตอร์ตามที่แสดงในตารางในรูปที่ 1 เช่น ถ้าปิด SWA จะอ่านมิเตอร์ได้ 1 มิลลิแอมป์ ปิดสวิตช์ SWA, SWB จะได้ 3 มิลลิแอมป์ และถ้า ปิดสวิตช์ทุกตัวจะได้ 15 มิลลิแอมป์ เป็นต้น

การทดลองนี้ใช้สวิตช์เพียง 4 ตัวเท่านั้น ที่จริงเราสามารถเพิ่มสวิตช์ขึ้นอีกไปเรื่อยๆ ค่าความต้านทานของสวิตช์ถัดมาจะน้อยลงไปที่ละเท่า แต่ไม่ควรเพิ่มสวิตช์มากเพราะกระแสจะไหลมากเกินไปจนเกินพิสัยของมิเตอร์ ถ้าต้องการทดลองกับสวิตช์หลายๆ ตัวควรลดขั้น กระแสลงจาก 1 มิลลิแอมป์เป็น 0.1 หรือ 0.01 มิลลิแอมป์ก็ได้ โดยการเพิ่ม R จาก 5 กิโลโอห์มเป็น 50 หรือ 500 กิโลโอห์ม

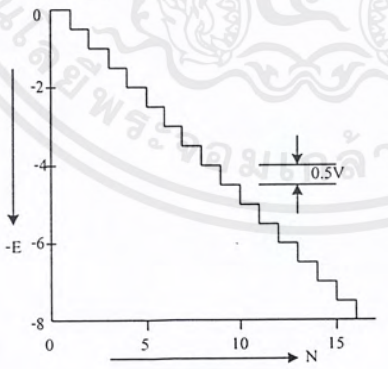
ใช้สวิตช์ดังรูปที่ 2.2 อาจจะยังมองไม่เห็นการใช้งานที่เป็นจริงลองดูวงจรในรูปที่ 2.3 วงจรนี้ใช้หลักการเดียวกันคือการรวมกระแส แต่ใช้เอาต์พุตของวงจรนับชนิด CMOS แทนสวิตช์ทั้งสี่ เอาต์พุต QA ถึง QD ของวงจรนับ เมื่อเป็น "1" จะจ่ายกระแสเข้าไปในวงจรเช่นเดียวกับการปิดสวิตช์ SWA ถึง SED ปกติ CMOS จ่ายกระแสได้ไม่มากนักเพราะไหลน้อย และแทนที่จะใช้มิเตอร์วัดกระแสต่อโดยตรง เราจะใช้ออปแอมป์ (OP-AMP) ต่อรับกระแสแทน และแปลงกระแสเป็นแรงดันออกทางเอาต์พุตของออปแอมป์ด้วย

ขั้วลบ (ขา 2) ของออปแอมป์จะมีศักย์ไฟฟ้า เท่ากับขาบวก (ขา 3) ตามหลักการทำงานของออปแอมป์ ขั้วลบจึงมีศักย์ไฟฟ้าเป็นศูนย์ จึงทำให้กระแส (I) ที่ไหลจากตัวต้านทานคำนวณให้ได้เหมือนรูปที่ 2.2 นอกจากนั้นกระแส (I) นี้จะไหลเข้าออปแอมป์เลย เพราะออปแอมป์มีความต้านทานสูงมาก กระแสทั้งหมดจะไหลผ่านตัวต้านทาน (R) ไปยังเอาต์พุต แรงดันที่ขาออกที่เอาต์พุตจึงเท่ากับแรงดันตกคร่อม (R) นั่นเอง แรงดัน (Eo) นี้จะเป็นค่าลบและจะมีขนาด 0.5 โวลต์ ต่อการเปลี่ยนแปลงสัญญาณดิจิตอล 1 ขั้นถ้าเปลี่ยน ค่าความต้านทาน (R) ให้เล็กหรือใหญ่ขึ้นก็สามารถเปลี่ยนขนาดของแรงดันขาออกนี้ได้

เมื่อเราป้อนพัลส์เข้าวงจรนับ เอาต์พุต QA จนถึง QD จะเริ่มเปลี่ยนแปลงและเพิ่มค่าตามหลักการของเลขฐานสอง เอาต์พุตใดเป็น “1” ก็จะจ่ายกระแสออกไป กระแสที่จ่ายออกไปนี้จะมีขนาดตาม “น้ำหนัก” ของบิตนั้น ถ้านำมัลติมิเตอร์มาวัดแรงดันขาออกของออปแอมป์ในขณะที่วงจรนับกำลังพัลส์จะเห็นเข็มมิเตอร์เพิ่มขึ้นเรื่อย ๆ จาก 0 โวลต์เป็น -0.5,-1,-1.5,-2.0 โวลต์โดยเพิ่มทีละ -0.5 โวลต์แบบนี้ไปเรื่อยๆ จนถึง -7.5 โวลต์ ในที่สุด จากนั้นจะตกกลับเป็น 0 โวลต์ใหม่แล้วจึงค่อยๆ เพิ่มแรงดันขึ้นไปอีก ถ้าใช้ออสซิลโลสโคป (Oscilloscope) จับรูปคลื่น Eo จะเห็นรูปคลื่นคล้ายขั้นบันไดตามรูปที่ 2.4



รูปที่ 2.3 ตัวอย่างวงจร D/A แบบรวมกระแส



รูปที่ 2.4 รูปคลื่นที่ได้จากวงจร D/A แบบรวมกระรวมกระแส

2) วงจร D/A แบบวงจรบันไดของตัวต้านทาน

วงจร D/A แบบแรกเป็นแบบรวมกระแส วงจร D/A ที่จะกล่าวต่อไปเป็นแบบรวมแรงดัน ใช้ตัวต้านทานเพียง 2 ค่าคือ R และ 2R (ตัวต้านทานที่มีค่า 2*R โอห์ม) ต่อสลับไปมาเหมือนขั้น

บันไดหรือเรียกว่า วงจร D/A แบบวงจรบันไดของตัวต้านทาน ตามที่แสดงในรูปที่ 2.5 ในรูปเป็นวงจร D/A ขนาด 8 บิต ใช้วงจรนับ 4 บิต 2 ตัวต่ออนุกรมกันและใช้เอาต์พุตของวงจรนับทำหน้าที่เป็นสวิตช์ป้อนเลขฐานสองเข้าวงจร D/A

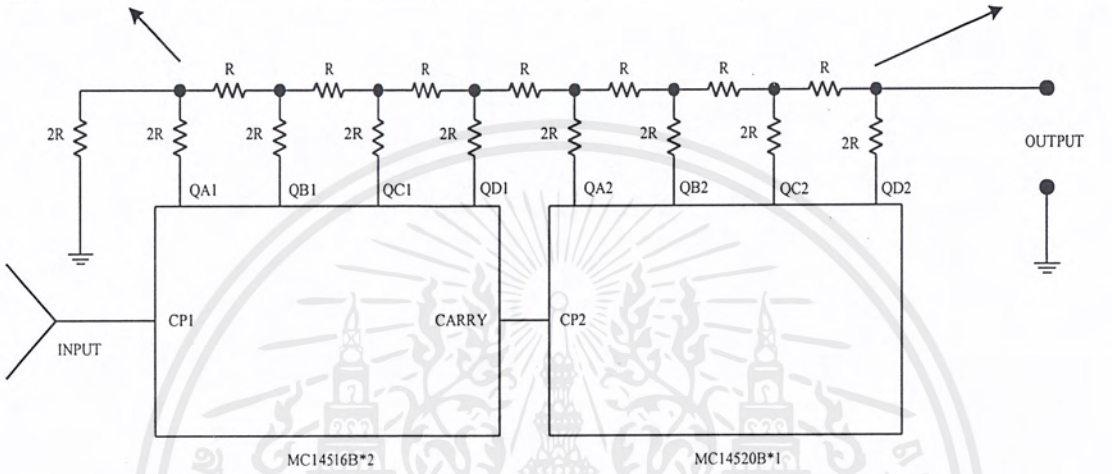
วงจรบันไดของตัวต้านทานนี้ผู้ออกแบบครั้งแรกนับว่ามีความสามารถมาก เพราะไม่ว่าวัดที่จุดไหนของแต่ละขั้นบันไดจะได้ความต้านทานรวมลงมาถึงปลายบันไดเท่ากับ R ดังนั้นถ้าบิตสูงสุดก็คือ QD2 เป็น "1" จะทำให้แรงดันขาออกเป็นครึ่งหนึ่งของแหล่งจ่ายไฟคือ VDD2 ถ้าบิตถัดไปคือ QC2 เป็น "1" ก็จะให้แรงดันขาออกลดลงไปครึ่งหนึ่ง VDD/4 บิตถัดๆ ไปก็จะทำให้แรงดันลดไปทีละครึ่งเรื่อยๆ จนถึงบิตสุดท้ายคือ QA1 จะให้แรงดันที่เอาต์พุตเป็น VDD/256 ถ้านับ แรงดัน VD/256 เป็นแรงดัน 1 หน่วย แต่ละบิตก็จะให้แรงดันคิดเป็นน้ำหนักเรียงจากบิตต่ำสุดไปสูงสุดคือ 1,2,3,…,128 เหมือนกับแบบรวมกระแสตามที่ได้อธิบายมาแล้ว

ถ้าป้อนพัลส์ให้วงจรนับและวัดแรงดันขาออกด้วยมัลติมิเตอร์ก็จะเห็นเข็มมิเตอร์ค่อยๆ กระดิกขึ้นหรือถ้าใช้ออสซิลโลสโคปจับดูรูปคลื่นก็จะเห็นเป็นรูปคลื่นฟันเลื่อยแบบขั้นบันไดเหมือนกัน ตัวต้านทาน R และ 2R ที่ใช้ในวงจรควรเป็นตัวต้านทานที่มีความแม่นยำสูงมาก เช่น ความผิดพลาดไม่เกิน 1 เปอร์เซ็นต์ ถ้าผิดพลาดมากจะทำให้สัญญาณแอนะลอกที่ได้ผิดพลาดได้

วงจรในรูปที่ 2.5 เป็นวงจร D/A ที่ใช้กับเลขฐานสอง ถ้าเป็นเลขฐานสิบรหัส BCD ซึ่งเรานิยมใช้กันมากนั้น ก็จำเป็นต้องดัดแปลงวงจรบันไดของตัวต้านทานใหม่เป็นอย่างในรูปที่ 2.6 วงจร D/A นี้มีวงจรบันไดอยู่ 2 วงจรเป็นของหลักหน่วยวงจรหนึ่งและเป็นของหลักสิบของอีกวงจรหนึ่ง แต่ละวงจรใช้ตัวต้านทาน R และ 2R เหมือนกัน สองวงจรนี้ต่อเชื่อมกันด้วยตัวต้านทานที่มีขนาด 9R เพื่อให้แรงดันขาออกของวงจรบันไดหลักหน่วยมีขนาดเล็กลง 10 เท่า เมื่อมารวมกับแรงดันขาออกของวงจรบันไดหลักสิบ ที่ใช้ค่า 9R เพราะความต้านทานขาออกของวงจรบันไดมีค่า R พอดีตามที่ได้อธิบายไปแล้ว

*แรงดันขาออกเมื่อบิตนั้นเป็น “1”

*	Vdd/256	Vdd/128	Vdd/64	Vdd/32		Vdd/16	Vdd/8	Vdd/4	Vdd/2
น้ำหนัก	1	2	4	8		16	32	64	128

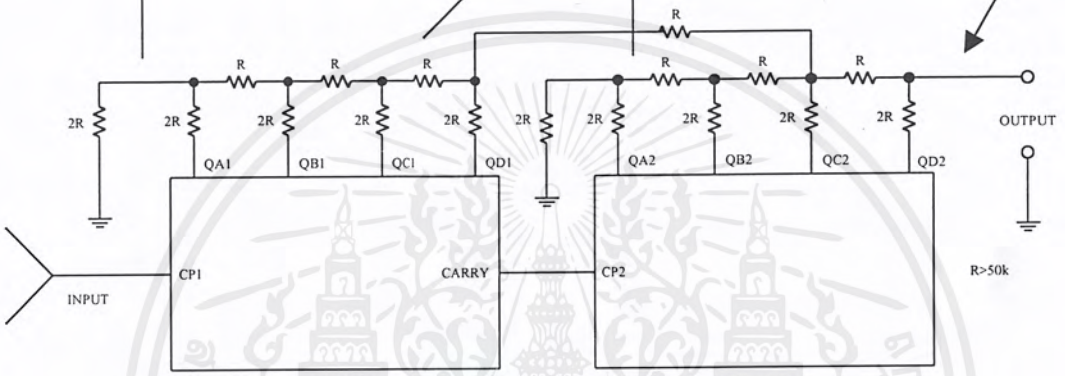


รูปที่ 2.5 ตัวอย่างวงจร D/A แบบวงจรบันไดของตัวต้านทาน

วงจรบันไดของตัวต้านทานนี้สามารถใช้งานในทางปฏิบัติได้ถึง 3 หลัก ของเลขฐานสิบ ถ้าเกินกว่า 3 หลักแล้ว จะต้องคัดเลือกค่าความต้านทานก็มีที่ทำเป็นไอซีสำเร็จรูปขาย ไอซีบางตัวที่เป็นวงจร D/A นั้นแท้จริงภายในจะมีวงจรบันไดของตัวต้านทานและแหล่งจ่ายกระแสที่ถูกควบคุมด้วยสัญญาณ “0” และ “1” ของอินพุตเท่านั้น ไอซี D/A ที่นิยมใช้กันมากได้แก่ เบอร์ MC 1408 ของบริษัทโมโตโรล่าเป็นวงจร D/A ขนาด 8 บิต

*แรงดันขาออกเมื่อบิตนั้นเป็น “1”

*	5Vdd / 8*110	5Vdd / 4*110	5Vdd / 2*110	5Vdd / 110		5Vdd / 8*11	5Vdd / 4*11	5Vdd / 2*11	5Vdd / 11
น้ำหนัก	1	2	4	8		10	20	40	80



เช่น MC14510B*2

MC145188*1

รูปที่ 2.6 วงจร D/A ที่ใช้กับเลขฐานสิบรหัส BCD 2 หลัก

2.3.2 การแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล

วงจร A/D (Analog to Digital Converter) จะทำหน้าที่แปลงแรงดันหรือกระแสที่เป็นสัญญาณแอนะลอกไปเป็นสัญญาณตัวเลขหรือสัญญาณดิจิตอลวงจร A/D มีด้วยกันหลายแบบแต่ที่นิยมใช้กันแพร่หลายมี 3 แบบคือ แบบสโลปคู่ (Dual Slope) แบบแปลงแรงดันเป็นความถี่ (V to F Converter) และแบบประมาณทีละบิต (Successive Approximation)

วงจร A/D แบบสโลปคู่เป็นแบบที่ง่ายที่สุด ไม่จำเป็นต้องใช้อุปกรณ์ที่มีคุณภาพดีมากนักก็สามารถแปลงสัญญาณได้อย่างเที่ยงตรง แต่มีข้อเสียตรงที่ใช้เวลาการแปลงสัญญาณนานมากไป จึงไม่เหมาะในการใช้วัดแรงดันในช่วงเวลาสั้นๆ เช่น การวัดรูปคลื่น ณ จุดเวลาใดเวลาหนึ่ง แบบสโลปคู่นี้เหมาะสมสำหรับใช้วัดค่าเฉลี่ยของแรงดันและกระแส จึงนิยมใช้กันมากในมัลติมิเตอร์แบบดิจิตอล และเครื่องวัดแสดงผลเป็นตัวเลขทั่วๆ ไป วงจร A/D แบบสโลปคู่ที่เป็นไอซีสำเร็จรูป

มีด้วยกันหลายเบอร์ ราคาไม่แพงมากนัก ส่วนมากจะให้ความเที่ยงตรงในการแปลงสัญญาณดีกว่า 0.1 เปอร์เซ็นต์ มีทั้งแบบแปลงเป็นตัวเลขขนาด 3 หลัก และ 4 หลัก หลักสุดท้ายจะแสดงค่าเป็น 0 หรือ 1 เท่านั้น จึงเรียกง่ายๆ ว่า หลัก

วงจร A/D แบบแปลงแรงดันเป็นความถี่และแบบประมาณทีละบิตนั้นมีข้อดีตรงที่สามารถแปลงสัญญาณได้รวดเร็ว มีความเที่ยงตรงดี เพียงแต่วงจรมีความซับซ้อนมากกว่าจึงมีราคาแพง

1) **วงจร A/D แบบสโลปคู่**

ให้ดูวงจรในรูปที่ 2.7 ซึ่งเป็นวงจรอินทิเกรเตอร์ (Integrator) แบบพื้นฐาน อุปกรณ์ที่มีสัญลักษณ์เป็นสามเหลี่ยมนั้นเป็นออปแอมป์ที่ทำหน้าที่เป็นวงจรขยายสัญญาณแตกต่าง R และ C ในวงจรเป็นอุปกรณ์สำคัญที่ทำให้เกิดการประจุเข้าไปใน C ทำให้ได้รูปคลื่นแรงดันขาออกซึ่งเท่ากับเป็นการอินทิเกรตสัญญาณอินพุต

สมมติว่ามีแรงดัน E_i ป้อนเข้าทางอินพุต จะมีการเกิดกระแส I ซึ่งมีค่า E_i/R ไหลผ่าน R ไหลเข้าไปในวงจร กระแสนี้จะไม่ไหลเข้าออปแอมป์ เพราะออปแอมป์มีความต้านทานขาเข้าสูง แต่จะไหลผ่านไปประจุที่ตัวเก็บประจุ C ทั้งหมดเป็นผลให้แรงดันคร่อม C สูงขึ้นไปเรื่อยๆ ขั้วลบของออปแอมป์จะมีศักย์ไฟฟ้าเป็น 0 โวลต์ เท่ากับขั้วบวกซึ่งต่อลงกราวนด์ (Ground) ดังนั้นแรงดันตกคร่อม C จึงเป็นแรงดันลบเมื่อเทียบกับกราวนด์ แรงดันขาออก E_oซึ่งเท่ากับแรงดันตกคร่อม C ซึ่งเป็นลบ แรงดันนี้ก็จะค่อยเพิ่มขึ้นเรื่อยๆ เป็นเส้นตรง แรงดันขาออกวงจรจึงเหมือนกับเป็นการอินทิเกรตแรงดันขาเข้า เพราะเมื่อเราอินทิเกรตค่าคงที่ได้เส้นตรงที่เปลี่ยนแปลงตามเวลา

รูปที่ 2.7 แสดงรูปคลื่นขาออกของวงจรอินทิเกรตเมื่อป้อนแรงดันอินพุตต่างๆ เช่น กรณีอินพุตเป็นบวก กรณีอินพุตเป็นลบ และกรณีอินพุตเป็นทั้งบวกและลบโดยเป็นบวกนาน t_1 วินาที และเป็นลบนาน t_2 วินาที จะเห็นว่าถ้าอินพุตเป็นบวกเอาต์พุตจะเป็นเส้นตรงที่มีสโลปเป็นลบ และถ้าอินพุตเป็นลบจะได้สโลปเป็นบวก การให้วงจรอินทิเกรตเตอร์ทำการอินทิเกรตขึ้นและลงแบบนี้จะได้เส้นตรงสโลปคู่

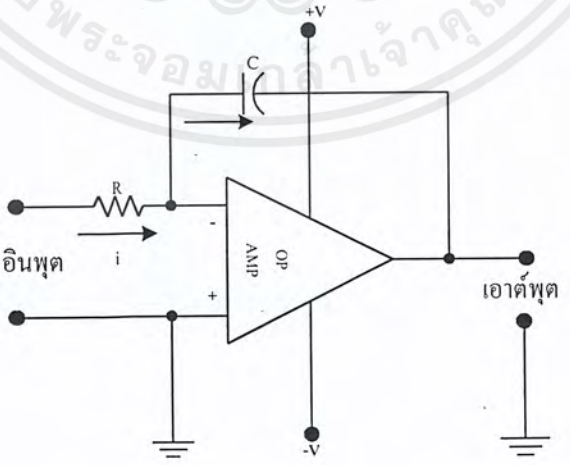
รูปที่ 2.7 เป็นรูปแสดงหลักการทำงานของวงจร A/D แบบสโลปคู่ อินพุตของ วงจรอินทิเกรตเป็นแรงดันไฟที่จะวัด E_i และมีอีกอินพุตหนึ่งเป็นแรงดันไฟมาตราฐาน E_{REF} อินพุตทั้งสองนี้มีขั้วสลับกันเสมอและจะผลัดกันต่อเข้ากับวงจรอินทิเกรเตอร์ ทำให้มีการอินทิเกรตขึ้นและลงเป็นจังหวะอยู่ตลอดเวลา เอาต์พุต E_0 ของวงจรอินทิเกรเตอร์จะต่อเข้ากับวงจรเปรียบเทียบ ซึ่งจะทำหน้าที่ตรวจจับว่ามีแรงดัน E_0 เมื่อใดเป็น 0 โวลต์

วงจรควบคุมในวงจร A/D นี้เป็นวงจรดิจิตอล ทำหน้าที่ ควบคุมการทำงานของวงจรทั้งหมดที่สำคัญ จะควบคุมสวิตช์ S_1 และ S_2 เพื่อต่ออินพุตให้กับวงจรอินทิเกรเตอร์

การทำงานของวงจรเริ่มจาก เมื่อแรงดัน E_O=0 โวลต์ วงจรควบคุมจะปิดสวิตช์เพื่อให้ แรงดันที่จะวัด E_i ต่อเข้ากับวงจรอินทิเกรเตอร์ วงจรควบคุม จะปล่อยให้วงจรอินทิเกรเตอร์ทำการอินทิเกรตสัญญาณ E_i เป็นระยะเวลา t_1 นี้ทำได้ไม่ยากนัก โดยทั่วไปมักจะใช้วงจรนับ นับพัลส์ที่มีความถี่คงที่จนได้ค่าที่กำหนดไว้

เมื่อครบเวลา t_1 วงจรควบคุมจะเปิดสวิตช์ S_1 ในขณะนั้นแรงดันเอาต์พุตของอินทิเกรเตอร์ E_O จะมีค่าแปรผันกับแรงดัน E_i จากนั้นวงจรควบคุมจะปิดสวิตช์ S_2 เพื่อต่อ แรงดันไฟมาตรฐาน E_{REF} เข้ากับวงจรอินทิเกรเตอร์และรีเซตวงจรนับ แรงดัน E_{REF} จะมีขั้วตรงข้ามกับ E_i สมมติให้ E_{REF} เป็นแรงดันลบ คราวนี้วงจรอินทิเกรเตอร์จะอินทิเกรตสัญญาณทำให้เกิดเส้นตรงที่มีสโลปตรงข้าม แรงดัน E_O จะเพิ่มขึ้นเรื่อยๆ จนในที่สุด E_O= 0 โวลต์ ในช่วงนี้สมมติว่าใช้เวลา t_2 ขณะเดียวกันวงจรนับจะนับพัลส์ไปเรื่อยๆ เมื่อ E_O = 0 โวลต์ วงจรเปรียบเทียบจะตรวจจับเหตุการณ์นี้แล้วส่งสัญญาณไปยังวงจรควบคุม วงจรควบคุมจะเปิดสวิตช์ S_2 และให้วงจรนับหยุดนับพัลส์ ค่าที่นับได้ในวงจรนับขณะนั้นจะเท่ากับค่าแรงดัน ที่ต้องการวัดพอดีที่เป็นเช่นนี้เพราะแรงดัน E_{REF} มีค่าคงที่ช่วงเวลา t_2 ในการอินทิเกรตสัญญาณจนแรงดันเป็นศูนย์ จะแปรผันกับแรงดันขาเข้า E_i ถ้า E_i มีค่ามาก ค่า E_O หลักจากการอินทิเกรตครั้งแรกจะมาก จึงใช้เวลา t_2 ในการอินทิเกรตครั้งที่ 2 มากตามไปด้วย สมมติให้ E_i=1 โวลต์ E_{REF}=-1 โวลต์ ได้ t_1= t_2 เท่ากับการนับพัลส์ 100 ลูก ถ้าให้ E_i=1.5 โวลต์ จะได้ t_2=1.5t_1 หรือเท่ากับการพัลส์ 1500 ลูก จะสังเกตเห็นว่าค่านับพัลส์นี้ตรงกับค่าแรงดัน E_i ซึ่งเป็นแรงดันที่ต้องการวัด

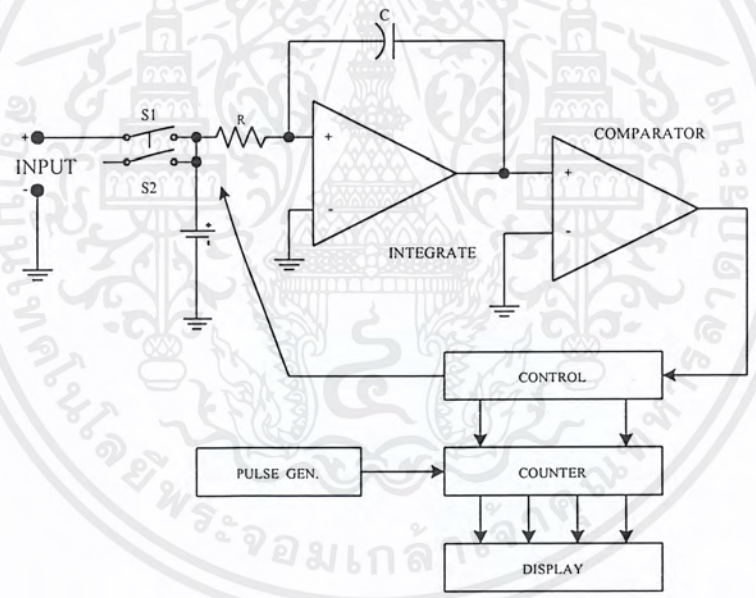
ค่านับของวงจรนับนี้สามารถส่งออกไปที่แลตช์ (Latch) และถอดรหัสเพื่อแสดงผลด้วย LED 7 Segment อีกทีหนึ่ง ทำให้เราสามารถอ่านค่าแรงดันเป็นตัวเลขได้โดยตรง



รูปที่ 2.7 การทำงานของวงจรอินทิเกรเตอร์

ดิจิตอลมัลติมิเตอร์ที่มีขายตามท้องตลาดทั่วไป มักแสดงผลสูงสุดเป็น 1999 แสดงว่า t_2 จะเท่ากับการนับพัลส์ไม่เกิน 2000 ลูก ถ้าในวงจรเราใช้พัลส์ความถี่ 10 กิโลเฮิรตซ์ ในการวัดหนึ่งครั้งจะต้องมีการอินทิเกรต 2 ครั้ง กินเวลา $t_1 + t_2$ หรือเท่ากับการนับพัลส์จำนวนสูงสุดไม่เกิน 1000+2000 ลูก ซึ่งจะกินเวลา 0.3 วินาที นับว่าเป็นเวลาที่ยาวนานพอสมควร ในหนึ่งวินาทีจะวัดได้ไม่เกิน 3 ครั้ง เท่านั้น ถ้าเป็นดิจิตอลมัลติมิเตอร์ที่แสดงผลสูงสุดเป็น 19999 ก็ยิ่งมีการนับพัลส์เป็นจำนวนมากขึ้นถึง 10 เท่า ถ้าใช้พัลส์ความถี่ 10 กิโลเฮิรตซ์เท่าเดิม จะต้องใช้เวลาถึง 3 วินาที ในการวัดหนึ่งครั้งซึ่งยาวนานเกินไป ดังนั้นคงต้องเพิ่มความถี่ของพัลส์ให้สูงขึ้น

ไอซีที่เป็นวงจร A/D แบบสโลปคู่นี้มีด้วยกันหลายเบอร์ และผลิตกันหลายบริษัท บางเบอร์ก็ใช้เพียงไอซีตัวเดียว บางเบอร์ก็ใช้ไอซี 2-3 ตัวต่อเป็นชุด ปกติมักใช้อุปกรณ์ภายนอก เช่น RC ต่อเพิ่มอีกเพียงเล็กน้อยก็สามารถใช้งานได้ดี

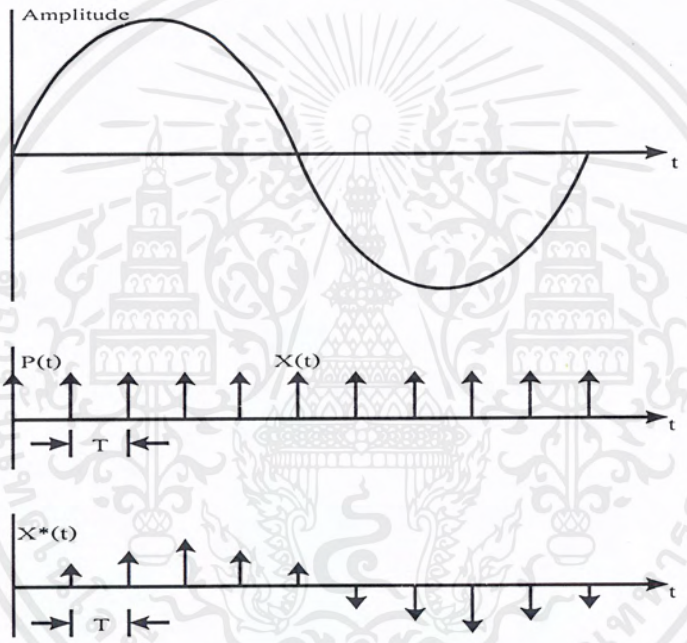


รูปที่ 2.8 หลักการทำงานของวงจร A/D แบบสโลปคู่

2.3.3 ทฤษฎีการสุ่มสัญญาณ (Sampling)

การสุ่มสัญญาณเป็นขั้นตอนแรกในการแปลงสัญญาณแอนะลอกให้เป็นสัญญาณดิจิตอล โดยวงจรสุ่มสัญญาณจะตรวจจับขนาดของสัญญาณแอนาลอกที่ถูกส่งตามช่วงเวลาที่ถูกกำหนดโดย ทั่วไปมักจะวัดขนาดในรูปของแรงดันไฟฟ้า ซึ่งแท้จริงแล้วกระบวนการสุ่มสัญญาณ เป็นกระบวนการตรวจวัดค่าแรงดันของสัญญาณ ที่ช่วงเวลาต่างๆ ซึ่งมีคาบการตรวจจับคงที่ อัตราหรือความถี่

ของการสุ่มสัญญาณเป็นคาบนี้จะกำหนดให้อยู่ในหน่วยของจำนวนจุดสุ่มต่อหนึ่งหน่วยเวลา ยกตัวอย่าง เช่น ในระบบโทรศัพท์อัตราการสุ่มดังกล่าวจะมีค่าเป็น 8000 ครั้งต่อวินาที หรือ 8 KHz หรืออาจกล่าวได้ว่าคาบของการสุ่ม (ช่วงเวลาระหว่างการสุ่มแต่ละครั้ง) มีค่าเป็น 1/8000 หรือ 125 ไมโครวินาที สำหรับวิธีในการคำนวณหาค่าอัตราการสุ่มของระบบโทรศัพท์โดยเป็นไปตามข้อกำหนดในทฤษฎีของการสุ่มสัญญาณ ซึ่งถูกกำหนดขึ้นโดยแชนนอน นักคณิตศาสตร์ชาวสหรัฐอเมริกา กล่าวว่าอัตราการสุ่มจะต้องมีความถี่ไม่น้อยไปกว่า 2 เท่า ของความถี่สูงสุดของสัญญาณแอนาลอกที่จะทำการส่งนั้นจึงจะสามารถสร้างสัญญาณต้นฉบับกลับคืนจากสัญญาณสุ่มได้



รูปที่ 2.9 สัญญาณที่ผ่านการสุ่มเทียบกับสัญญาณต้นฉบับ

2.4 ทฤษฎีและหลักการของ FPGAs

FPGAs จัดเป็นวงจรรวมเฉพาะกิจชนิดหนึ่งที่สามารถโปรแกรมเป็นวงจรดิจิตอลใดๆ ก็ได้ เช่นเดียวกับ EPLD ต่างกันที่ EPLD โปรแกรมลงบนอีพรอม (EPROM) ภายใน และสามารถโปรแกรมใหม่ได้หลังจากนำไปลบด้วยแสงอุลตราไวโอเลต แต่สำหรับ FPGAs จะโปรแกรมลงบนสแตติกแรม (Static RAM) ภายในด้วยข้อมูลที่อยู่ภายนอก และสามารถโปรแกรมใหม่ได้โดยการรีเซตด้วยสัญญาณไฟฟ้านอกจากนั้น FPGAs ยังประหยัดพลังงานและมีความจุวงจรที่มีขนาดใหญ่ (จำนวนเกตมากๆ) ได้อีกด้วย

2.4.1 โครงสร้างภายใน

วงจรรวมชนิดนี้ผลิตโดยบริษัทไซลิงค์ซึ่งเป็นบริษัทที่ร่วมกันทำการค้นคว้ากับ บริษัท เอ็มเอ็มไอ (MMI) สร้างเป็นอะเรย์ที่ประกอบด้วยเกตจำนวน 600-25000 เกต ดังแสดงในตารางที่ 2.2 การที่ต้องบอกขนาดของวงจรรวมเป็นจำนวนเกตเพราะจะได้รู้ว่าขนาดของ วงจรที่ได้ออกแบบไว้สามารถโปรแกรมลงบนวงจรรวม FPGAs ได้หรือไม่

FPGAs มีโครงสร้างภายในใกล้เคียงกับสถาปัตยกรรมของเกตอะเรย์มาก สามารโปรแกรมและลบองค์ประกอบ สแตติกแรม ภายในได้โดยใช้กระแส ไฟฟ้า ซึ่งทำการโปรแกรมได้โดยดึงข้อมูลฐานสิบหกมาจากภายนอก เช่น Parallel EPROM หรือ Serial PROM ต่างกับ EPLD, PAL ที่มีอีพรอมอยู่ในตัว ภายใน FPGAs จะจัดเรียงเป็นลอจิกเซลล้อมรอบภายนอกด้วยอินพุตเอาต์พุตเซล FPGAs ตัวแรกที่ผลิตโดยไซลิงค์คือ XC2064 (2000 Family) ประกอบด้วยเซลเรียงกันเป็นเมตริกซ์ (Matrix) เป็นจำนวน 64 เซล หลังจากนั้นผลิตตระกูล 3000 และ 4000 มีโครงสร้างซับซ้อนขึ้น สามารถบรรจุจำนวนเกตได้สูงและดีขึ้นแต่ละเซลเรียกว่าCLB (Configuration Logic Block)

ตารางที่ 2.1 การนับเกตของเกตพื้นฐาน

Gate	Equivalent gate count	Gate	Equivalent gate count
INV	1	RS Latch	3
NAIINR2	1	D Latch	4
NAIINR3	2	D Latch with CLR	5
NAIINR4	2	D Lath with PRE	5
NAIINR6	5	D Lath with PRE/CLR	6
NAIINR8	6	DEF	6
NAIINR9	7	DEF with CLR	7
NAIINR12	8	DEF with PRE	7
NAIINR16	11	DEF with PRE/CLR	8
BUFF	2	JKFF	9
ANIIOR2	2	JKFF with CLR	12
ANIIOR2	2	JKFF with CLR/PRE	13
ANIIOR2	3	TFF with CLR	8
XOR2	3	TEF with PRE	8
XNOR2	3	TEF with PRE/CLR	9

ตารางที่ 2.2 คุณสมบัติของ FPGAs ตระกูลต่างๆ

FPGAs	Appr. Gate count	Max I/O	Flip-Flop	Ram bit	Available CLBs
XC2064	1,000	58	122	0	64
XC2018	1,500	74	174	0	100
XC3020/3120	1,800	64	256	0	64
XC3030/3130	2,700	80	360	0	100
XC3042/3142	3,700	96	480	0	144
XC3064/3164	5,500	120	688	0	224
XC3090/3190	7,500	144	928	0	320
XC3195	9,000	176	1,320	0	484
XC4002A	2,000	64	256	2,048	64
XC4003/4003A	3,000	80	360	3,200	100
XC4003H	3,000	160	200	3,200	100
XC4004A	4,000	960	480	4,608	144
XC4005/4005A	5,000	122	616	6,272	196
XC4005H	5,000	192	392	6,272	196
XC4006	6,000	128	768	8,192	256
XC4008	8,000	144	936	103,687	324
XC4010	10,000	160	1,120	12,800	400
XC4010D	10,000	160	1,120	0	400
XC4013	13,000	192	1,536	18,432	576
XC4025	25,000	256	2,560	32,768	1,024

2.4.2 ส่วนที่เป็นองค์ประกอบของลอจิก (Configurable Logic Block)

CLB จะจัดเรียงกันเป็นแบบเมตริกซ์แบบอะเรย์ขนาด M*N การออกแบบนั้นสามารถทำได้โดยการจัดวาง CLB ให้ต่อถึงกัน เราสามารถจัด CLB ให้เชื่อมต่อถึงกันได้โดยการทำมือหรือโปรแกรมที่สนับสนุน FPGAs ทำให้โดยอัตโนมัติโดยวิธีของมันเอง สำหรับแฟ้มที่ได้จากโปรแกรมเหล่านี้เราเรียกว่า แฟ้มโครงร่าง (Configuration file) ซึ่งจะบรรจุโครงร่างภายในของ CLB ตามความเหมาะสม ในอีกด้านหนึ่งแฟ้มโครงร่างนั้นจะเป็นแฟ้มกระแสข้อมูล (Bitstrem) ซึ่งสามารถใช้โปรแกรมหน่วยความจำภายในของ FPGAs (Intermal FPGAs configuration memory) ได้ สำหรับรูปแสดง CLB ของ FPGAs ตระกูล 4000 แสดงดังรูปที่ 2.10

2.4.3 ส่วนอินพุตและเอาต์พุต (I/O Block)

รอบๆ นอกของ FPGAs จะประกอบด้วย IOBs ประมาณ 64 ถึง 144 ตัวซึ่งขึ้นอยู่กับตระกูลของ FPGAs ซึ่ง IOBs จะเป็นตัวเชื่อมต่อระหว่างภายในกับภายนอกของวงจรลอจิกของ FPGAs ลักษณะของ IOBs จะมีลักษณะ 2 ทิศทาง สามารถโปรแกรมให้เป็นอินพุตหรือเอาต์พุตก็ได้ สำหรับตารางที่ 2.2 แสดง IOBs ของ FPGAs ตระกูล 4000

2.4.4 รายละเอียดการใช้งาน

FPGAs สามารถทำงานได้หลายรูปแบบ (Mode) โดยกำหนดได้ที่ขาสัญญาณ M0 M1 และ M2 ดังแสดงในตารางที่ 2.3 ในแบบมาสเตอร์พาราแรล (Master Parallel) รับโปรแกรมคอนฟิก (Config) ทีละ 1 ไบต์ (Byte) จากหน่วยความจำภายนอกที่เป็นแบบขนานโดยสามารถรับโปรแกรมคอนฟิก จากตำแหน่ง (Address) ต่ำหรือสูงก่อนก็ได้ แบบเพอริเฟอรัล (Peripheral) รับโปรแกรมคอนฟิกทีละ 1 ไบต์จากไมโครโปรเซสเซอร์ โดยสามารถโต้ตอบกันได้ว่าพร้อมหรือไม่พร้อมที่จะรับข้อมูลต่อไป แบบสเลฟ (Slave) รับโปรแกรมคอนฟิก ทีละ1 บิตจากไมโครโปรเซสเซอร์ตามสัญญาณอินพุต CCLK ส่วนในแบบมาสเตอร์ซีเรียล (Master Serial) รับโปรแกรมคอนฟิกทีละ 1 บิต จากหน่วยความจำจากภายนอกที่เป็นแบบอนุกรม

จากความต้องการสร้างให้ใช้กระแสไฟฟ้าต่ำ ในแบบการต่อใช้งานทั้ง 5 แบบจึงมีเพียง 2 แบบเท่านั้นที่เหมาะสม คือ แบบมาสเตอร์ซีเรียล (Master Serial Mode) และแบบสเลฟซีเรียล (Slave Serial Mode) ส่วนในแบบมาสเตอร์พาราแรลต้องใช้อีพรอม 27 Cxxx ซึ่งกินกระแสมากกว่าพรอม XC17xxx เหมาะในการทดสอบต้นแบบก่อนเมื่อวงจรต้นแบบทำงานได้ถูกต้องแล้วจึง

ทำการอัดโปรแกรมลงพรอมอีกทีหนึ่งเพราะว่าในแบบพาราแรลอีพรอมสามารถโปรแกรมได้ใหม่ต่างกับอีพรอมที่โปรแกรมได้ครั้งเดียว

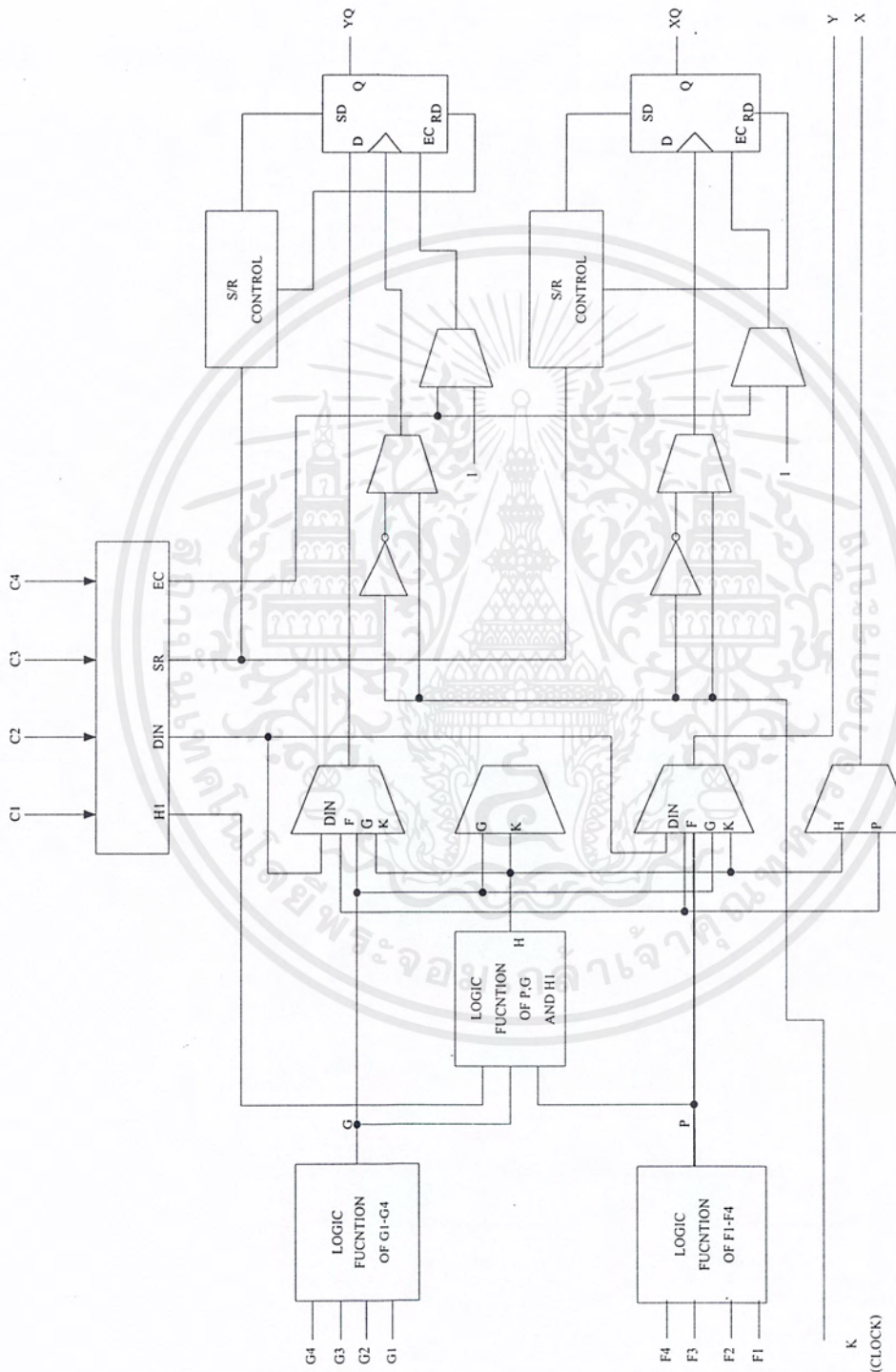
การใช้งาน FPGAs ในแบบสเลฟซีเรียล เมื่อเริ่มจ่ายไฟเข้าตัว FPGAs จะทำการเคลียร์ (Clear) หน่วยความจำที่ใช้ในคอนฟิก (Configuration Memory) ตรวจสอบแบบกระทำการคอนฟิกว่าแบบใดในตารางที่ 2.3 (เป็นแบบอนุกรมหรือขนาน) หลังจากนั้นจะเริ่มทำการโปรแกรมคอนฟิก สัญญาณ Done/Program เป็น "0" (อยู่ในระหว่างโปรแกรม) และเมื่อข้อมูลในการคอนฟิกที่รับมาจากภายนอกเต็มหน่วยความจำที่ใช้ในการคอนฟิกและความยาวของข้อมูลตรงกับที่ส่วนหัวคอนฟิกสัญญาณ Done/Program เป็น "1" (โปรแกรมคอนฟิกเสร็จสิ้น) ดูรูป 2.12 ประกอบ

ตารางที่ 2.3 รูปแบบต่างๆ ของการกำหนดองค์ประกอบในการทำงาน

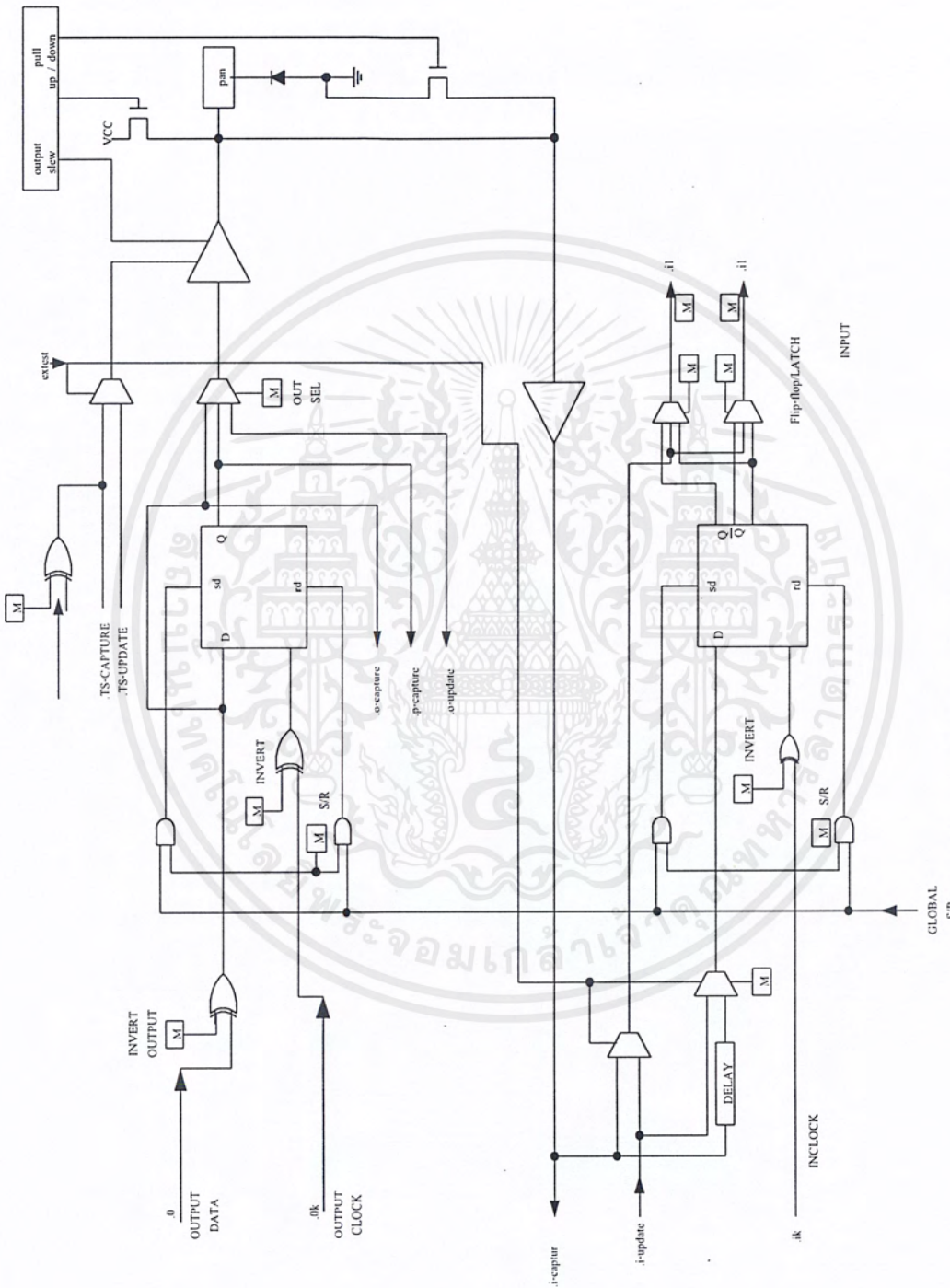
M0	M1	M2	CLK	MODE	DATA
0	0	0	Output	Master Serail	Bit Serial
0	0	1	Output	Master Parallel	Bit Wide Add.=0000 up
0	1	0	--------	Reseved	--------
0	1	1	Output	Master Parallel	Bit Wide Add.=FFFF down
1	0	0	--------	Reseved	--------
1	0	1	Output	Peripheral	Byte Wide
1	1	0	--------	Reseved	--------
1	1	1	Input	Slave Serail	Bit Serial

1) การใช้งานแบบสเลฟซีเรียล (Slave Serial)

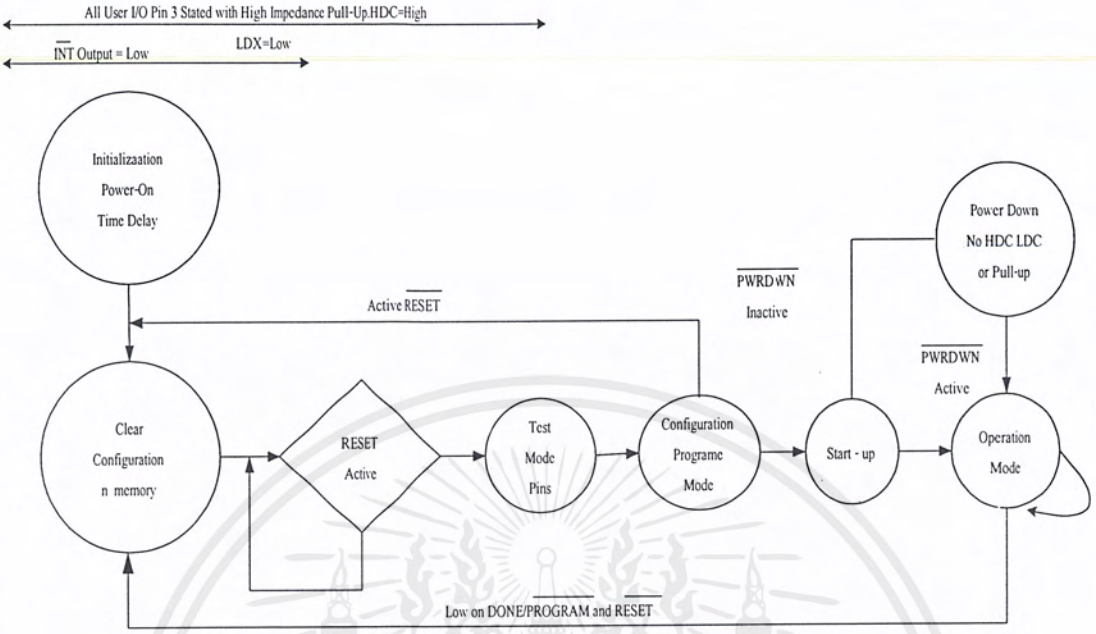
การต่อใช้งานแบบนี้เหมาะสมกับวงจรที่ออกแบบมาเพื่อทำงานร่วมกับไมโครคอมพิวเตอร์อยู่แล้วทั้งนี้เพราะ FPGAs ได้ใช้ความสามารถของไมโครคอมพิวเตอร์ในการเก็บและส่งข้อมูลคอนฟิกให้ เพียงแต่ต้องเขียนโปรแกรมเพื่อส่งโปรแกรมคอนฟิกให้เพิ่ม ลักษณะการต่อในแบบนี้เป็นดังรูปที่ 2.13 ซึ่งในแบบนี้ไมโครคอมพิวเตอร์จะสร้างสัญญาณเพื่อการทำคอนฟิกให้กับอุปกรณ์ FPGAs ทำได้โดยต่อสัญญาณ Strobe เข้ากับขา CCLK และพอร์ต DO เข้ากับขา DIN สร้างสัญญาณนาฬิกาป้อนเข้าที่ขา CCLK และป้อนโปรแกรมคอนฟิกแบบอนุกรมเข้าที่ขา DIN ดังแผนภูมิใน รูปที่ 2.14



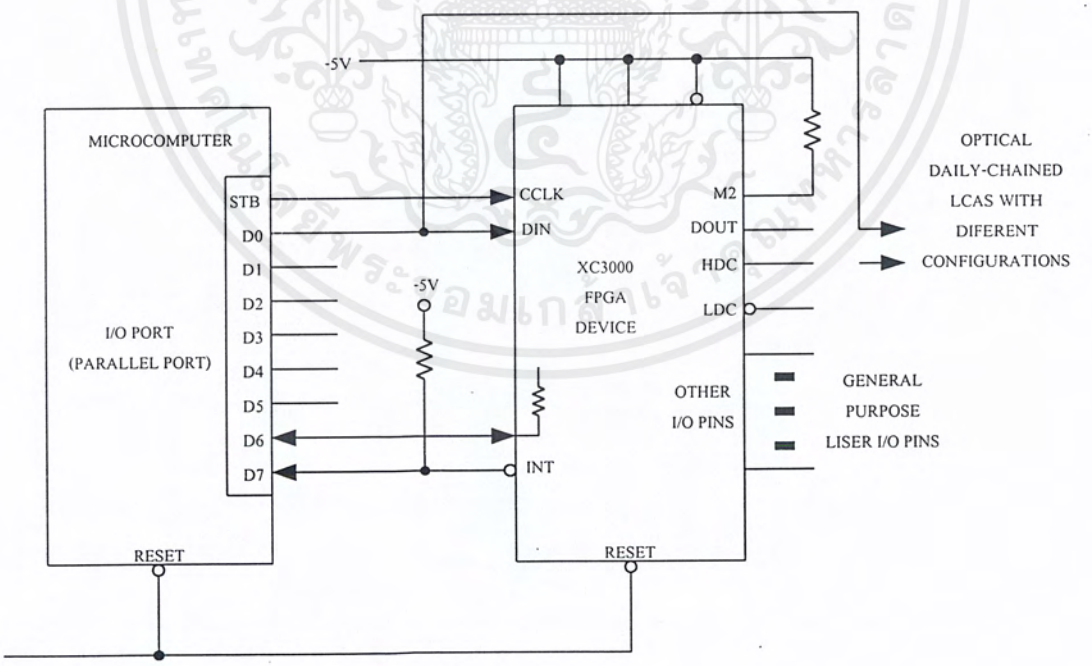
รูปที่ 2.10 แผนผัง CLB ของตระกูล 4000



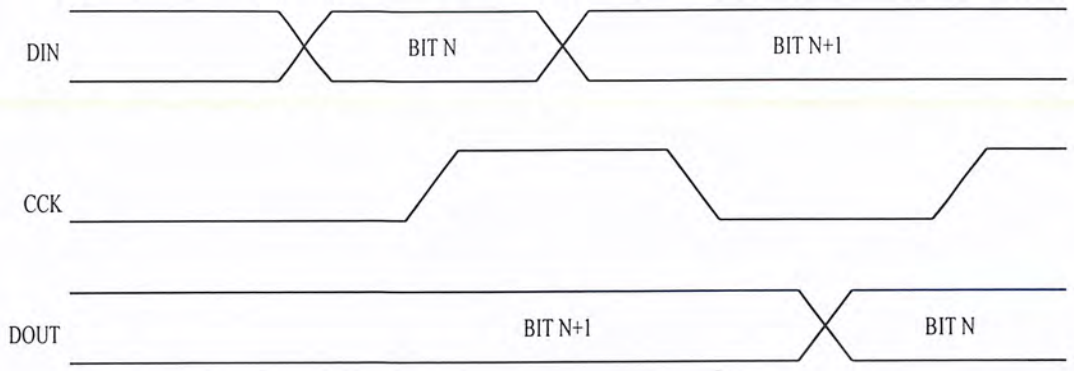
รูปที่ 2.11 แผนผัง IOB ของตระกูล 4000



รูปที่ 2.12 แผนผังลำดับในการคอนฟิกเมื่อป้อนแหล่งจ่ายไฟเข้าในไอซีและการโปรแกรมใหม่



รูปที่ 2.13 การต่อใช้งานในแบบสเลฟซีเรียล



รูปที่ 2.14 แผนภูมิเวลาการป้อนข้อมูลโปรแกรมคอนฟิกในแบบสเลฟซีเรียล

2) การต่อใช้งานแบบมาสเตอร์ซีเรียล (Master Serial)

การต่อใช้งานในแบบนี้ ส่วนที่เก็บโปรแกรมคอนฟิกจะต่างจากแบบแรกคือใช้พรอม XC17xxx เป็นตัวเก็บโปรแกรม ทำให้ไม่ต้องเสียเวลาเขียนโปรแกรมเพื่อทำการคอนฟิก ซึ่งวิธีการอัดโปรแกรมคอนฟิกลงพรอมทำตามขั้นตอนดังนี้คือ เมกบิต (Make Bits) สร้างแฟ้ม BIT จากวงจรที่ออกแบบและใช้โปรแกรม Make PROM สร้างแฟ้ม Hex แล้วทำการอัดโปรแกรมลงพรอม ด้วยอุปกรณ์อัดพรอมที่มาพร้อมกับตัวโปรแกรมของไซลิงค์ (Xilinx)

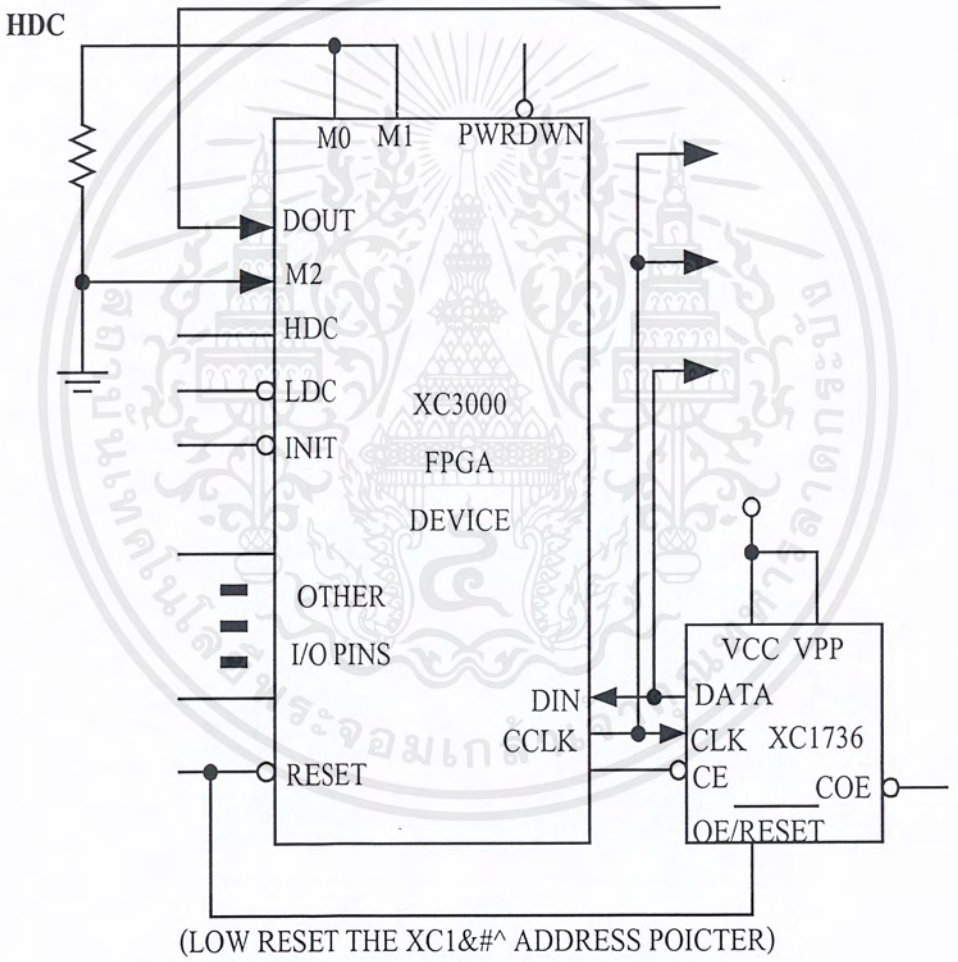
แบบนี้พรอม XC17xxx จะส่งสัญญาณเพื่อทำการคอนฟิกให้กับอุปกรณ์ FPGAs ดังแสดงในรูปที่ 2.15 DIN เป็นขารับข้อมูลที่ใช้ในการคอนฟิกแบบอนุกรม ส่วนอุปกรณ์ FPGAs จะกำเนิดสัญญาณ CCLK ให้กับพรอมเพื่อเป็นจังหวะในการอ่านข้อมูลโปรแกรมคอนฟิกมาไว้ในสแตติกแรม

3) การใช้งานในแบบมาสเตอร์พาราแรล (Master Parallel)

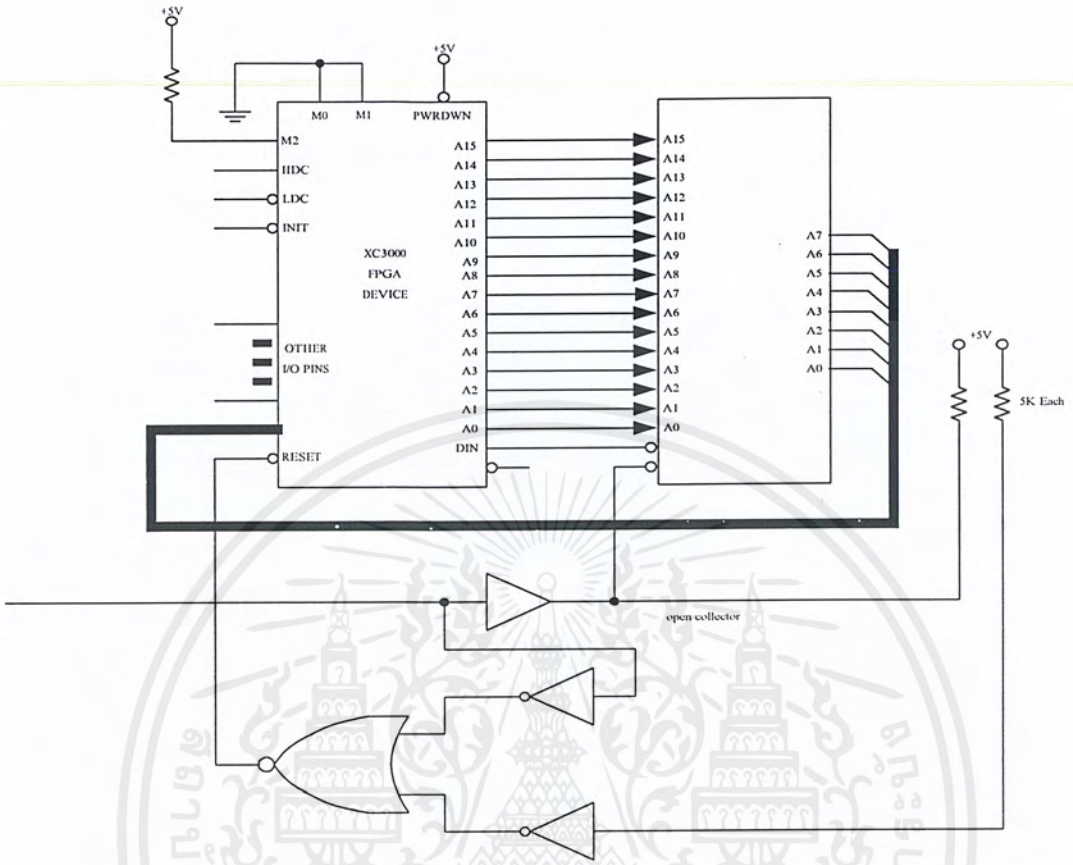
การต่อใช้งานในแบบนี้ ส่วนที่เก็บโปรแกรมคอนฟิกจะต่างจากแบบมาสเตอร์ซีเรียลคือใช้พรอม 27 xzx เป็นตัวโปรแกรม ทำให้ไม่ต้องเสียเวลาเขียนโปรแกรมเพื่อทำการคอนฟิก ซึ่งวิธีการอัดโปรแกรมคอนฟิกลงพรอม ทำได้โดยใช้โปรแกรม MakePROM ใน XACT สร้างแฟ้ม Hex MCS แล้วทำการอัดโปรแกรมลงอีพรอมด้วยอุปกรณ์อัดอีพรอมที่มีใช้ทั่วไป

แบบนี้อีพรอม 27 xxx จะส่งสัญญาณทีละไบต์เพื่อทำการคอนฟิกให้กับอุปกรณ์ FPGAs ดังแสดงในรูปที่ 2.16 D0-D7 เป็นขารับข้อมูลที่ใช้ในการคอนฟิกแบบขนานนี้ A0-A15 เป็นตำแหน่งที่ FPGAs สร้างให้กับอีพรอมเพื่ออ่านข้อมูลจากหน่วยความจำ (โปรแกรมคอนฟิก) มาเก็บไว้ในสแตติกแรมตำแหน่งทั้ง16เส้นไม่จำเป็นต้องต่อครบก็ได้ขึ้นอยู่กับขนาดหน่วยความจำอีพรอมที่ใช้ และสามารถกำหนดให้นับขึ้นหรือลงก็ได้

แบบนี้อีพรอม 27 xxx จะส่งสัญญาณทีละไบต์เพื่อทำการคอนฟิกให้กับอุปกรณ์ FPGAs ดังแสดงในรูปที่ 2.16 D0-D7 เป็นขารับข้อมูลที่ใช้ในการคอนฟิกแบบขนานนี้ A0-A15 เป็นตำแหน่งที่ FPGAs สร้างให้กับอีพรอมเพื่ออ่านข้อมูลจากหน่วยความจำ (โปรแกรมคอนฟิก) มาเก็บไว้ในสเตติกแรม ตำแหน่งทั้ง 16 เส้นไม่จำเป็นต้องต่อครบก็ได้ขึ้นอยู่กับขนาดหน่วยความจำอีพรอมที่ใช้ และสามารถกำหนดให้นับขึ้นหรือลงก็ได้



รูปที่ 2.15 การต่อใช้งานในแบบมาสเตอร์ซีเรียล



รูปที่ 2.16 การต่อใช้งานในแบบมาสเตอร์พาราแรล

4) **ข้อควรระวังในการใช้ FPGAs**

สิ่งแรกที่สำคัญคือ ไอซีกลัวความร้อนเป็นที่สุด การบัดกรีโดยหัวแร้งกำลังสูง หรือ บัดกรีโดยจี้หัวแร้งที่ขาไอซีนานๆ จะทำให้ไอซีเสียหายได้ง่าย ระยะเวลาในการบัดกรีหนึ่งจุดไม่ควรเกิน 5-10 วินาที ควรใช้ซ็อกเกต (Socket) ไอซีในการประกอบวงจรลงแผ่นลายวงจรพิมพ์

การป้องกันแหล่งจ่ายไฟให้ไอซีไม่ควรผิดขั้ว ถ้าสลับขั้วบวกหรือลบกันจะทำให้ไอซีเสียได้ นอกจากนั้นแรงดันของแหล่งจ่ายไฟต้องอยู่ในช่วงที่โรงงานกำหนดมา สำหรับ FPGAs แรงดันที่ไอซีทำงานอยู่ในช่วงการทำงานได้ Vcc = 4.75-5.25 โวลต์ และแรงดันที่ทนได้อยู่ในช่วง -0.5-7 โวลต์ ดังนั้นก่อนป้อนแรงดันควรตรวจสอบให้แน่ใจก่อน

การต่อวงจรผิดที่มีผลให้ไอซีเสียหายทันทีได้แก่ การต่อเอาต์พุตของไอซีไปเข้า ขั้วบวกหรือลบของแหล่งจ่ายไฟโดยตรง ไอซีทีทีแอล (TTL) ในขณะที่ CMOS ยังพอทนได้บ้างเพราะมีการจำกัดกระแสขาออกอยู่ภายในตัว ดังนั้นก่อนป้อนแรงดันเข้าไอซีควรตรวจ ดูบริเวณเอาต์พุตของไอซีด้วย

อีกประการหนึ่งที่ไอซีประเภท CMOS กลัวมากที่สุด คือ ไฟฟ้าสถิตย์ ขาที่เป็นอัตราได้ง่ายที่สุดคือขาอินพุต ความต้านทานขาเข้าของขาอินพุตของไอซีประเภทนี้มักจะสูงมากจนมีคุณสมบัติเหมือนกันกับเป็นฉนวนที่ขาอินพุตนี้อาจรับไฟฟ้าสถิตย์จากที่ใดมาเก็บไว้ เมื่อสะสมไว้มากๆ ไม่มีทางระบายออกก็จะทำให้วงจรบริเวณอินพุตเสียหายได้ ในปัจจุบันเทคโนโลยีการผลิตไอซีเจริญรุดหน้าไปมาก ทางด้านอินพุตของไอซี CMOS จะมีวงจรป้องกันไว้ภายในอย่างไรก็ตามเพื่อป้องกันเหตุการณ์ดังกล่าวควรเก็บรักษาไอซีที่ถูกต้องและควรเก็บไว้ในที่ปลอดภัยจากไฟฟ้าสถิตย์โดยเก็บไว้บนแผ่นเยื่อสังเคราะห์แบบนำกระแสได้หรือห่อหุ้มด้วยกระดาษอะลูมิเนียม

ไอซีประเภท CMOS มีความต้านทานขาเข้าสูงมาก กระแสจึงไม่ไหลจากแหล่งจ่ายไฟเข้าไปที่ขาอินพุตของ CMOS นั้นถึงแม้จะต่อความต้านทานดั่นกลางอยู่ แรงดัน 5 โวลต์จากแหล่งจ่ายไฟก็ยังป้อนเข้าขาอินพุต 5 โวลต์ โดยที่ไม่มีแรงดันตกคร่อมบนตัวต้านทานเลย ถ้ามีกระแสไหลแสดงว่าไอซีตัวนั้นเสีย เอาต์พุตของไอซี CMOS อยู่ในสภาวะ 1 จะสามารถขับโหลดได้หลายมิลลิแอมป์ และถ้าเอาต์พุตอยู่ในสภาวะ 0 ก็สามารถจะรับกระแสไหลเข้าได้หลายมิลลิแอมป์

บทที่ 3
การออกแบบ การสร้าง และการทำงาน

3.1 กล่าวนำ

ในส่วนของการออกแบบ และการสร้างนั้นได้แบ่งออกเป็นส่วนการทำงานทั้งหมด 4 ส่วนใหญ่ ๆ ด้วยกันคือ วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก , การคำนวณหาค่าสัมประสิทธิ์ของการแปลง DCT และ IDCT , การเขียนชุดคำสั่งในส่วนประมวลผล , การโปรแกรมบนอุปกรณ์ FPGA

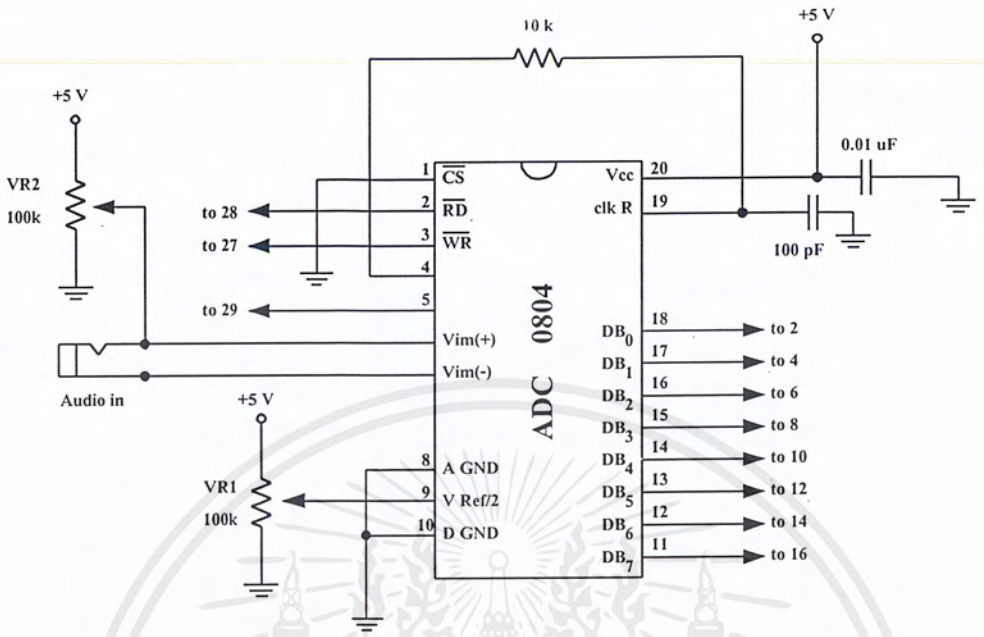
3.2 วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก

ในส่วนของวงจรการทำงานในส่วนนี้จะเป็นส่วนในการแปลงสัญญาณเสียงเพื่อให้ออยู่ในรูปของสัญญาณดิจิตอล และแปลงสัญญาณเสียงซึ่งได้ผ่านกระบวนการการบีบอัดแล้วให้กลับคืนอยู่ในรูปของสัญญาณเสียงเช่นเดิม เพื่อรับฟังเสียงเป็นผลการทดลอง

3.2.1 วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล

ในการออกแบบวงจรส่วนนี้ได้เลือกใช้ไอซี แปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล เบอร์ ADC 0804 ซึ่งเป็นไอซีแปลงสัญญาณ ขนาด 8 บิต ซึ่งมีคุณสมบัติพิเศษ คือ สามารถสร้างสัญญาณออสซิเลเตอร์ ได้โดยการต่อค่า ความต้านทาน และตัวเก็บประจุ เข้าไปได้ และยังสามารถกำหนดค่ากึ่งกลางของสัญญาณอ้างอิงได้ในส่วนของ Vref/2 ได้

การออกแบบวงจรได้กำหนดในส่วนรับสัญญาณอินพุตให้รับสัญญาณอินพุทได้อยู่ในช่วงขนาดแรงดัน -2.5 V ถึง 2.5 V โดยวงจรการทำงานแสดงดังรูปที่ 3.1



รูปที่ 3.1 วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล

1) **การปรับแต่งวงจร**

1.1) ทำการจ่ายไฟให้กับวงจร

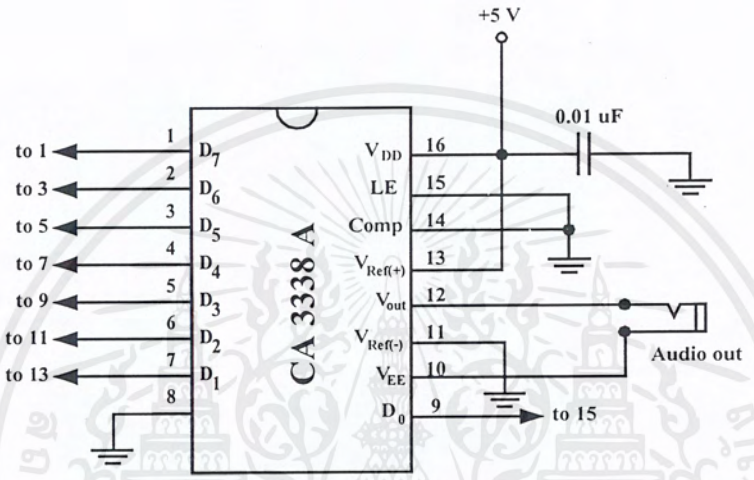
1.2) ทำการปรับค่าแรงดันอ้างอิงอินพุต โดยการวัดแรงดันตกคร่อม VR1 และทำการปรับ VR1 ให้มีค่าแรงดันตกคร่อม 2.5 V

1.3) ทำการปรับค่าแรงดันยกระดับสัญญาณอินพุต โดยการวัดแรงดันตกคร่อม VR2 และทำการปรับ VR2 ให้มีค่าแรงดันตกคร่อม 2.5 V

1.4) ทำการปรับระดับความแรงของสัญญาณอินพุต โดยใช้ออสซิลโลสโคปวัดรูปคลื่นสัญญาณอินพุต จากนั้นป้อนสัญญาณเสียงที่เป็นลักษณะเสียงที่ต่อเนื่อง และมีช่วงระดับแรงดันของสัญญาณที่ค่อนข้างคงที่ จากนั้นทำการปรับสัญญาณให้อยู่ในช่วงระดับแรงดัน –2.5 V ถึง 2.5 V

3.2.2 วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก

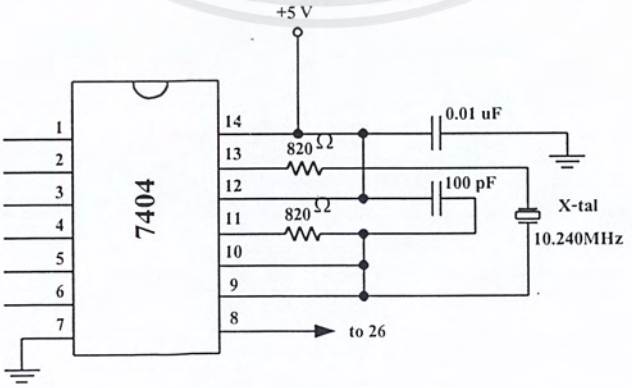
ในการออกแบบวงจรส่วนนี้ได้เลือกใช้ไอซี แปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก เบอร์ CA3338A ซึ่งเป็นไอซีแปลงสัญญาณขนาด 8 บิต และให้ค่าสัญญาณเอาท์พุตอยู่ในช่วง 0 V ถึง 5 V โดยการออกแบบวงจรได้ใช้การต่อวงจรมาตรฐานดังแสดงในรูปที่ 3.2



รูปที่ 3.2 วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก

3.2.3 วงจรกำเนิดสัญญาณนาฬิกา

ในการออกแบบวงจรส่วนนี้ได้ใช้ไอซีลอจิกเกตเบอร์ 7404 ในการสร้างสัญญาณพัลส์ โดยการออสซิเลเตอร์ของ คริสตอล กับวงจร R – C ดังแสดงในรูปที่ 3.3

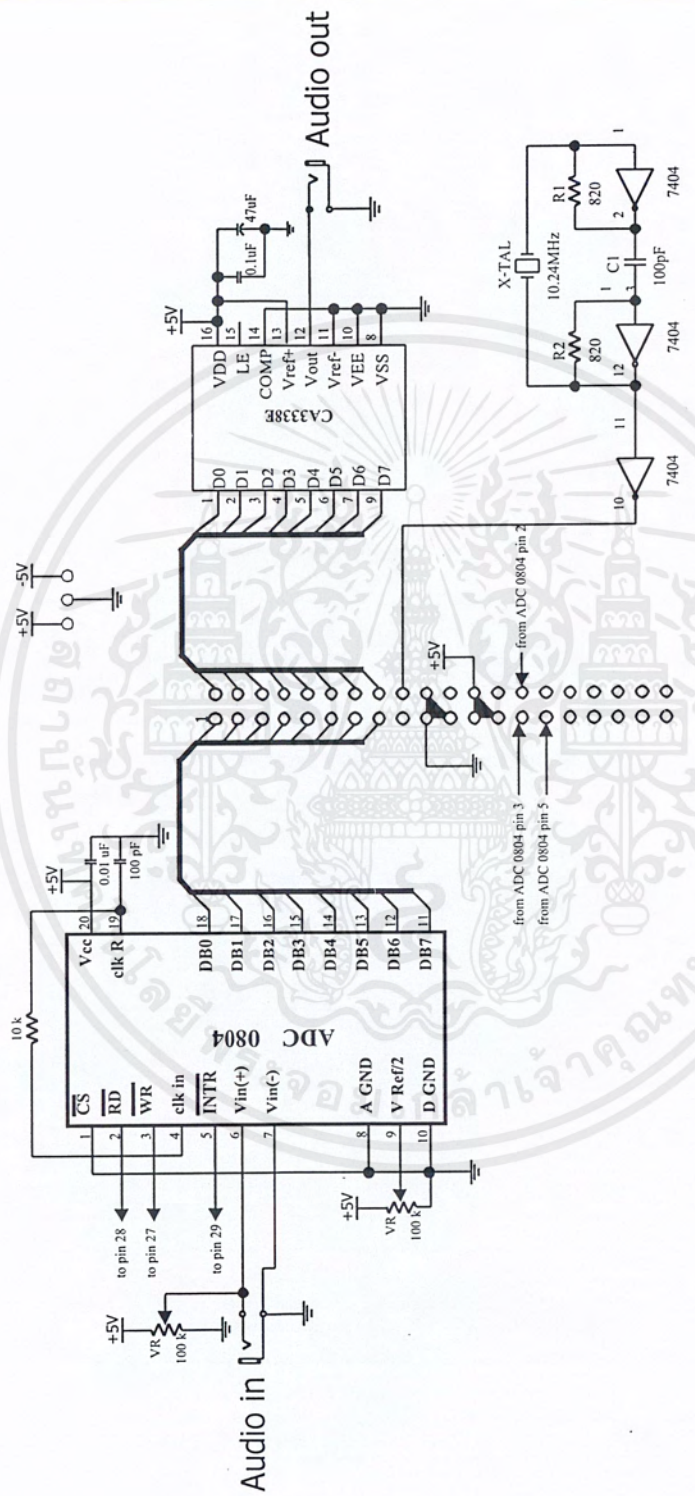


รูปที่ 3.3 วงจรกำเนิดสัญญาณนาฬิกา

3.2.4 การเชื่อมต่อวงจรการทำงานเข้าด้วยกัน

จากวงจรการทำงานทั้งหมดที่กล่าวมาข้างต้น จะได้ทำการรวมวงจรทั้งหมดไว้ในแผ่นวงจรเดียวกัน โดยที่จะประกอบไปด้วย วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล , วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก และวงจรกำเนิดสัญญาณนาฬิกา ซึ่งในส่วนของสัญญาณดิจิตอลที่เป็น อินพุต และเอาท์พุต จะทำการเชื่อมต่อกับ Connector ขนาด 40 pins เพื่อสามารถที่จะเชื่อมต่อกับส่วนประมวลผล ดังแสดงในรูปที่ 3.4

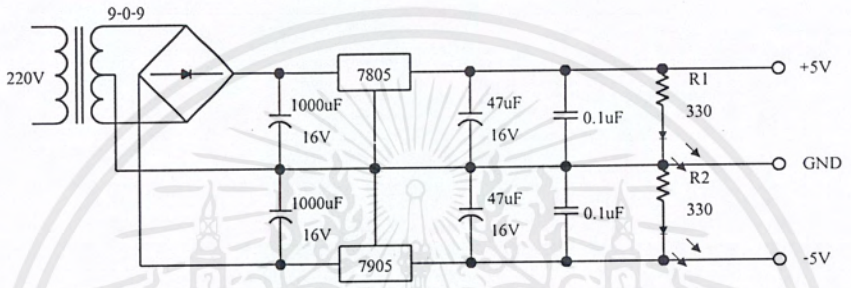
ส่วนการทดสอบการทำงานของวงจร นั้นจะกล่าวไว้ในบทที่ 4 หัวข้อที่ 4.1 การทดสอบการทำงานวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และ วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก



รูปที่ ข.6 การเชื่อมต่อวงจรการทำงานเข้าด้วยกัน

3.3 วงจรภาคจ่ายไฟ

เนื่องจากวงจรทั้งหมดที่ออกแบบมาต้องการใช้แรงดันเลี้ยงวงจร ±5 โวลต์ ดังนั้นเราจึงต้องใช้วงจรแหล่งจ่ายไฟที่ให้แรงดันเป็น ±5 โวลต์ ซึ่งวงจรแหล่งจ่ายไฟแสดงดังรูปที่ 3.5 ทำงานโดยวงจรใช้ไอซีเรกกูเลเตอร์ เบอร์ 7805 และ 7905 เป็นตัวให้แรงดันที่เอาต์พุตของวงจรเป็น ±5 โวลต์



รูปที่ 3.5 วงจรภาคจ่ายไฟ

3.4 การคำนวณหาค่าสัมประสิทธิ์ของการแปลง DCT และ IDCT

ในการหาค่าสัมประสิทธิ์ของการแปลงสัญญาณนั้น จะทำการคำนวณค่าจากสมการทางคณิตศาสตร์ของการแปลงสัญญาณนั้น ๆ โดยในที่นี้ได้ใช้โปรแกรมสำเร็จรูป MATLAB ช่วยในการคำนวณในส่วนของเมตริกซ์

3.4.1 การคำนวณหาค่าสัมประสิทธิ์ของการแปลง DCT

จากสมการที่ (3.1) คือสมการของการแปลงสัญญาณ DCT ซึ่งกำหนดให้ความยาวของสัญญาณมีขนาด 8 บิต ดังนั้น N = 8 ซึ่งผลของการคำนวณจะได้ค่าสัมประสิทธิ์ดังตารางที่ 3.1

$$F(u)=\frac{1}{\sqrt{2N}}C(u)\sum_{i=0}^{N-1}f(i)\cos\left[\frac{(2i+1)\pi u}{2N}\right] \tag{3.1}$$

โดย

$F(u)$ เป็นสัมประสิทธิ์ของการแปลง

$f(i)$ ข้อมูลเริ่มต้น

$$C(u)=\begin{cases}\frac{1}{\sqrt{2}};u=0\\1;n\neq 0\end{cases}$$

3.4.2 การคำนวณหาค่าสัมประสิทธิ์ของการแปลง IDCT

จากสมการที่ (3.2) คือสมการของการแปลงสัญญาณ IDCT ซึ่งกำหนดให้ความยาวของสัญญาณมีขนาด 8 บิต ดังนั้น N = 8 ซึ่งผลของการคำนวณจะได้ค่าสัมประสิทธิ์ดังตารางที่ 3.2

$$f(i)=\frac{1}{\sqrt{2N}}\sum_{u=0}^{N-1}C(u)F(u)\cos\left[\frac{(2i+1)\pi u}{2N}\right] \tag{3.2}$$

โดย

$F(u)$ เป็นสัมประสิทธิ์ของการแปลง

$f(i)$ ข้อมูลเริ่มต้น

$$C(u)=\begin{cases}\frac{1}{\sqrt{2}};u=0\\1;n\neq 0\end{cases}$$

3.4.3 การปรับปรุงค่าสัมประสิทธิ์ของการแปลง DCT และ IDCT

จากผลการคำนวณค่าสัมประสิทธิ์ของการแปลง DCT และ IDCT ดังตารางที่ 3.1 และ 3.2 ตามลำดับจะเห็นได้ว่า ผลลัพธ์ที่ได้นั้นจะอยู่ในรูปของเมตริกซ์ ขนาด 8 x 8 และคาทีได้ทั้งหมดนั้นจะเป็นเลขทศนิยมที่มีค่าไม่เกิน 1 ซึ่งจะอยู่ในรูปของจำนวนที่เรียกว่า Floating Point ซึ่งในการเขียนชุดคำสั่งในลำดับต่อไปนั้นจำเป็นต้องใช้การคำนวณค่าทั้งหมดให้อยู่ในรูปของเลขจำนวนเต็ม Fixed Point ดังนั้น จึงจำเป็นที่จะต้องทำการปรับปรุงค่าสัมประสิทธิ์ที่ได้ให้อยู่ในรูปของเลขจำนวณเต็ม โดยการคูณค่าทั้งหมดด้วย 10 แล้วทำการปัดจำนวนทศนิยมให้เป็นจำนวนเต็ม ซึ่งการปรับปรุงค่าสัมประสิทธิ์นั้นจะยังคงความเป็นเอกลักษณ์ที่ค่าแต่ละจำนวนในเมตริกซ์นั้นยังคงมีความสัมพันธ์กันอยู่

ค่าสัมประสิทธิ์ DCT และ IDCT ที่ได้ทำการปรับปรุงแล้วแสดงอยู่ในตารางที่ 3.3 และ 3.4 ตามลำดับ

ตารางที่ **3.1** ค่าสัมประสิทธิ์ของการแปลง DCT

	I = 0	I = 1	I = 2	I = 3	I = 4	I = 5	I = 6	I = 7
U = 0	0.70710	0.98078	0.92387	0.83146	0.70710	0.55557	0.38268	0.19509
U = 1	0.70710	0.83146	0.38268	- 0.19509	- 0.70710	- 0.98078	- 0.92387	- 0.55557
U = 2	0.70710	0.55557	- 0.38268	- 0.98078	- 0.70710	0.19509	0.92387	0.83146
U = 3	0.70710	0.19509	- 0.92387	- 0.55557	0.70710	0.83146	- 0.38268	- 0.98078
U = 4	0.70710	- 0.19509	- 0.92387	0.55557	0.70710	- 0.83146	- 0.38268	0.98078
U = 5	0.70710	- 0.55557	- 0.38268	0.98078	- 0.70710	- 0.19509	0.92387	- 0.83146
U = 6	0.70710	- 0.83146	0.38268	0.19509	- 0.70710	0.98078	- 0.92387	0.55557
U = 7	0.70710	- 0.98078	0.92387	- 0.83146	0.70710	- 0.55557	0.38268	- 0.19509

ตารางที่ 3.2 ค่าสัมประสิทธิ์ของการแปลง IDCT

	I = 0	I = 1	I = 2	I = 3	I = 4	I = 5	I = 6	I = 7
U = 0	0.70710	0.70710	0.70710	0.70710	0.70710	0.70710	0.70710	0.70710
U = 1	0.98078	0.83146	0.55557	0.19509	- 0.19509	- 0.55557	- 0.83146	- 0.98078
U = 2	0.92387	0.38268	- 0.38268	- 0.92387	- 0.92387	- 0.38268	0.38268	0.92387
U = 3	0.83146	- 0.19509	- 0.98078	- 0.55557	0.55557	0.98078	0.19509	- 0.83146
U = 4	0.70710	- 0.70710	- 0.70710	0.70710	0.70710	- 0.70710	- 0.70710	0.70710
U = 5	0.55557	- 0.98078	0.19509	0.83146	- 0.83146	- 0.19509	0.98078	- 0.55557
U = 6	0.38268	- 0.92387	0.92387	- 0.38268	- 0.38268	0.92387	- 0.92387	0.38268
U = 7	0.19509	- 0.55557	0.83146	- 0.98078	0.98078	- 0.83146	0.55557	- 0.19509

ตารางที่ 3.3 ค่าสัมประสิทธิ์ของการแปลง DCT หลังจากการปรับปรุงค่า

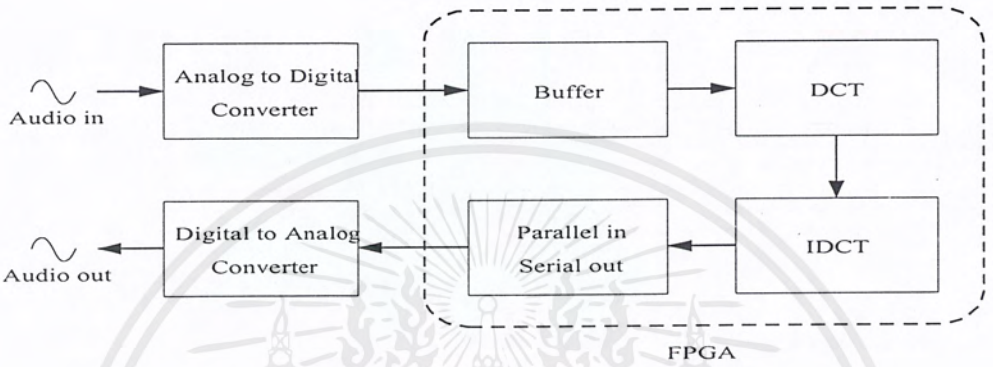
	I = 0	I = 1	I = 2	I = 3	I = 4	I = 5	I = 6	I = 7
U = 0	7	10	9	8	7	5	3	1
U = 1	7	8	3	-1	-7	-10	-9	-5
U = 2	7	5	-3	-10	-7	1	9	8
U = 3	7	1	-9	-5	7	-8	-3	10
U = 4	7	-1	-9	5	5	-8	-3	10
U = 5	7	-5	-3	10	-7	-1	9	-8
U = 6	7	-8	3	1	-7	10	-9	5
U = 7	7	-10	9	-8	7	-5	3	-1

ตารางที่ 3.4 ค่าสัมประสิทธิ์ของการแปลง IDCT หลังจากการปรับปรุงค่า

	I = 0	I = 1	I = 2	I = 3	I = 4	I = 5	I = 6	I = 7
U = 0	7	7	7	7	7	7	7	7
U = 1	10	8	5	1	-1	-5	-8	-10
U = 2	9	3	-3	-9	-9	-3	3	9
U = 3	8	-1	-10	-5	5	10	1	-8
U = 4	7	-7	-7	7	7	-7	-7	7
U = 5	5	-10	1	8	-8	-1	10	-5
U = 6	3	-9	9	-3	-3	9	-9	3
U = 7	1	-5	8	-10	10	-8	5	-1

3.5 การเขียนชุดคำสั่งในการประมวลผล

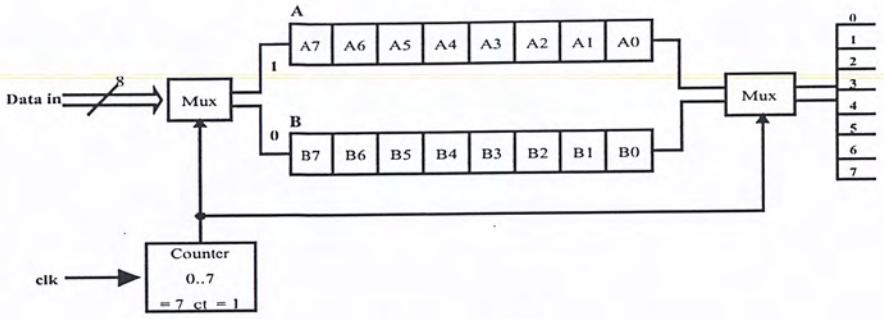
ในการเขียนโปรแกรมในส่วนของการทำงาน เพื่อนำไปโปรแกรมลงในส่วนของอุปกรณ์ FPGAs นั้นได้ใช้ภาษา VHDL ในการเขียนโปรแกรม โดยที่จะแบ่งส่วนการเขียนโปรแกรมออกเป็นส่วน ๆ ตามการทำงานคร่าว ๆ ดังในรูปที่ 3.6



รูปที่ 3.6 ไดอะแกรมการทำงานของโปรแกรม

3.5.1 การเขียนโปรแกรมส่วน Buffer

ในส่วนของ Buffer นั้นจะเป็นส่วนที่ทำการเตรียมข้อมูลนำเข้า ซึ่งมีลักษณะสัญญาณดิจิตอล ขนาดความยาวข้อมูล 8 บิต มีลักษณะการส่งสัญญาณมาเป็นลักษณะสัญญาณอนุกรม ซึ่งโปรแกรม Buffer นี้จะเป็นส่วนที่จะจัดข้อมูล ให้มีลักษณะเป็นเมตริกซ์ ขนาด 8 x 1 โดยจะมีการทำงานคือทำการนับข้อมูลที่เข้ามาทางด้านอินพุตให้ครบจำนวน 8 (0 – 7) แล้วก็ทำการนำข้อมูลออกไปในลักษณะของข้อมูล ขนานขนาด 8 ช่องสัญญาณ ดังรูปที่ 3.7



รูปที่ 3.7 ไดอะแกรมการทำงานโปรแกรมส่วน Buffer

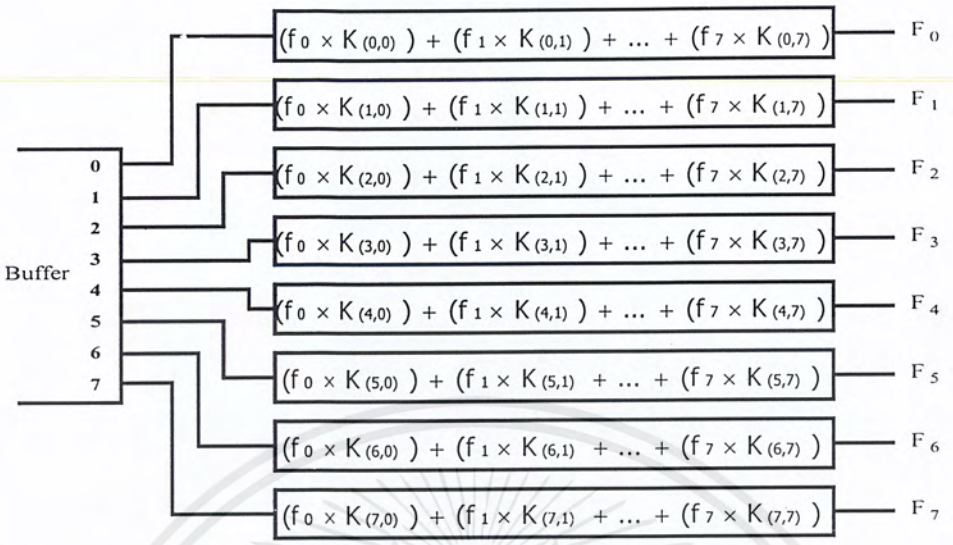
3.5.2 การเขียนโปรแกรมส่วน Parallel in Serial out

การทำงานในส่วนนี้จะเป็นส่วนการทำงานหลังจากส่วน IDCT ซึ่งจะเป็นส่วนการทำงานที่จะจัดข้อมูลขนานที่อยู่ในลักษณะเมตริกซ์ ขนาด 1 x 8 ให้เป็นข้อมูลอนุกรมเพื่อต่อเข้าไปยังส่วนวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอกต่อไป

หลักการเขียนโปรแกรมในส่วนนี้ จะใช้หลักการที่นำข้อมูลขนานซึ่งไม่มีสัญญาณนาฬิกาเข้าจังหวะสัญญาณนำมา ซิงโครไนซ์ รวมกับสัญญาณนาฬิกาที่กำหนดขึ้นจากในส่วนของ FPGAs เองและแยกข้อมูลออกเป็นห้วง ๆ จึงได้ข้อมูลกลับเป็นลักษณะของข้อมูลอนุกรมนั่นเอง

3.5.3 การเขียนโปรแกรมส่วน DCT

โปรแกรมส่วนนี้จะเป็นส่วนประมวลผลซึ่งจำเป็นต้องทำความเข้าใจ และทราบถึงขั้นตอนของการคำนวณจากสมการ DCT อย่างละเอียด เพื่อสามารถที่จะนำมาเขียนเป็นโปรแกรมการทำงานซึ่งจะต้องแยกส่วนของการคำนวนเป็นส่วน ๆ ดังแสดงไว้ในรูปที่ 3.8 ซึ่งเป็นไดอะแกรมการทำงาน โดยจะมีส่วนของการคำนวณทางคณิตศาสตร์หลัก ๆ ด้วยกัน 2 ส่วนคือ ส่วนของการคูณค่าข้อมูลเข้า (ตัวแปร f) กับค่าสัมประสิทธิ์การแปลง DCT (ตัวแปร K) ดังแสดงในตารางที่ 3.3 และส่วนของการบวก ซึ่งคือการบวกแบบผลรวม ค่าที่ได้จากการคูณค่าสัมประสิทธิ์แต่และค่า ได้เป็นผลลัพธ์ หรือข้อมูลที่ผ่านการแปลง DCT แล้ว



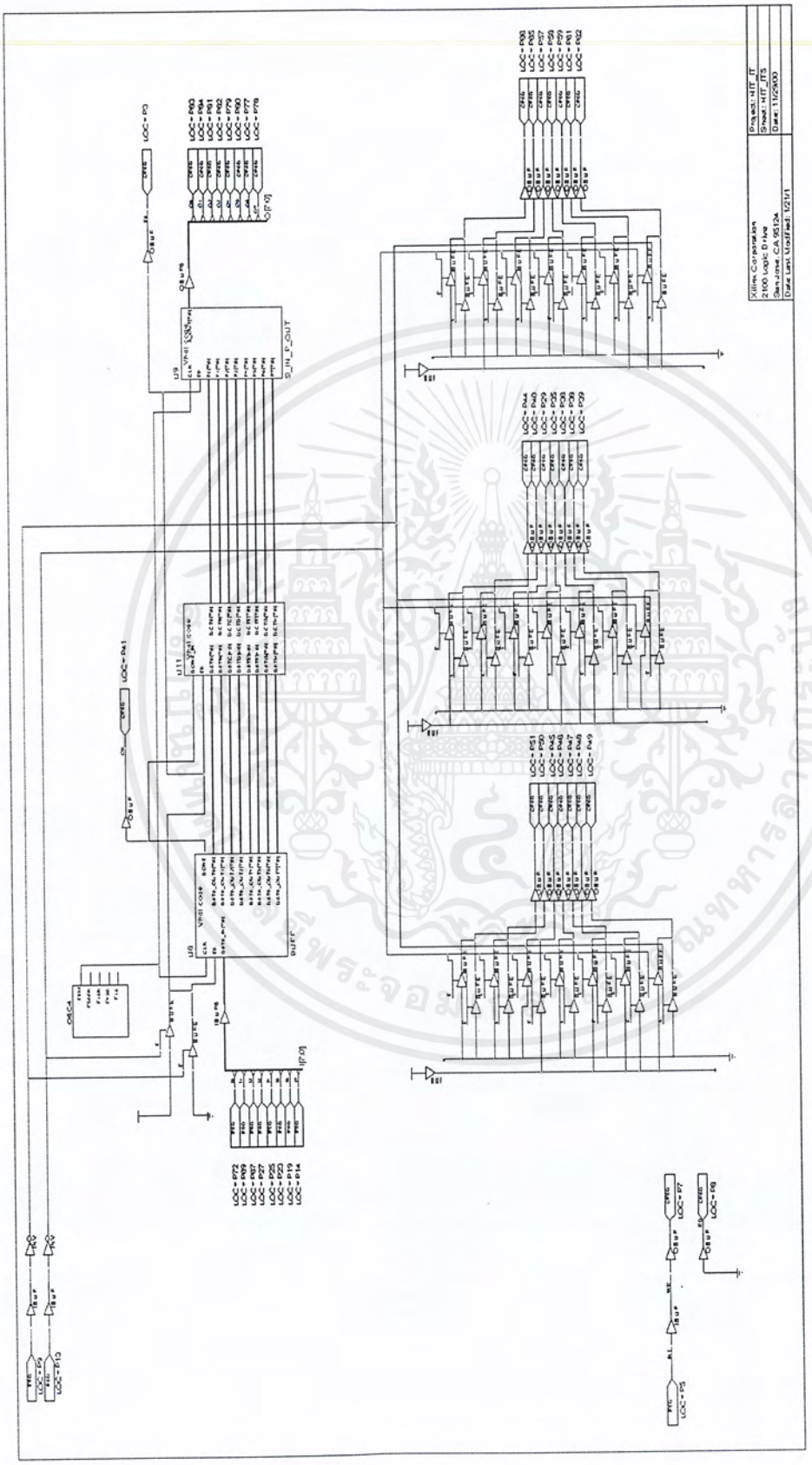
รูปที่ 3.8 ไดอะแกรมการทำงานโปรแกรมส่วน DCT

3.5.4 การเขียนโปรแกรมส่วน IDCT

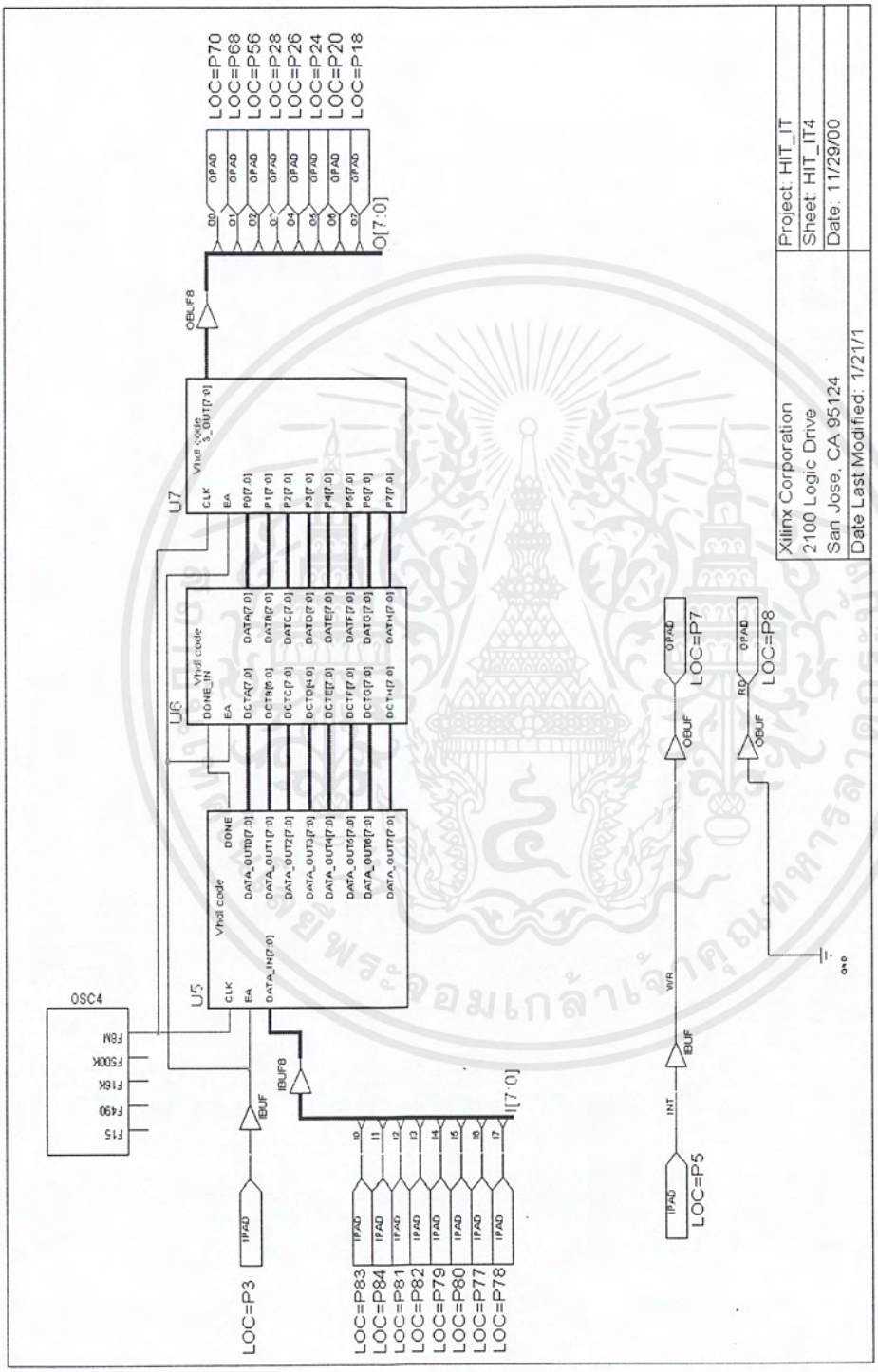
ในส่วนของการเขียนโปรแกรมในส่วนของ IDCT นั้น โครงสร้างหลักของโปรแกรมจะเหมือนกับโปรแกรม DCT เพียงแต่ ค่าสัมประสิทธิ์ที่นำมาคูณกับข้อมูลนั้น จะเป็นค่าสัมประสิทธิ์ที่ได้จากการคำนวณสมการ IDCT ในตารางที่ 3.4

3.5.5 การเขียนวงจร Schematic

หลังจากที่ได้ทำการเขียนโปรแกรมในทุกส่วนเสร็จสิ้นแล้ว ให้ทำการสร้างมาโครของโปรแกรมแต่ละส่วน และนำมาโครที่ได้นำมาต่อวงจรในส่วน Schematic Editor ของโปรแกรม XILINX Foundation 2.1 ซึ่งจะเป็นการเชื่อต่อในส่วนของ อินพุต เอาท์พุต และการกำหนดขาข้อมูลของตัวชิพด้วย ซึ่งการเชื่อมต่อส่วนของมาโครจะแบ่งออกเป็นสองส่วน ส่วนที่หนึ่งจะประกอบไปด้วยส่วน Buffer , DCT และส่วนการแสดงผล ดังแสดงในรูปที่ 3.9 และส่วนที่สองจะประกอบไปด้วยส่วน IDCT และParallel in Serial out ดังแสดงในรูปที่ 3.10



รูปที่ 3.9 วงจร Schematic ส่วน Buffer , DCT และส่วนแสดงผล



รูปที่ 3.10 วงจร Schematic ส่วนส่วน IDCT และParallel in Serial out

บทที่ 4

การทดลอง และผลการทดลอง

การทดลองได้แบ่งออกเป็น 3 ส่วนใหญ่ ๆ ด้วยกันคือ ส่วนที่หนึ่ง การทดสอบการทำงานวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และ วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก ส่วนที่สอง การจำลองการทำงานของโปรแกรม และส่วนที่สามการทดลองการบีบอัดสัญญาณเสียง

4.1 การทดสอบการทำงานวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และ วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก

ในการทดสอบการทำงานในส่วนนี้ ใช้วิธีการทดสอบโดยการบายพาสสัญญาณเสียงจากส่วนของวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล ไปยังวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก โดยการนำสัญญาณดิจิตอลขนาด 8 บิตจากวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอลนำไปต่อโดยตรงที่อินพุตของวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก

ลำดับขั้นการทดลอง

1. ใช้ jumper เชื่อมต่อขา 1 กับขา 2 , ขา 3 กับขา 4 , ขา 5 กับขา 6 , ขา 7 กับขา 8 , ขา 9 กับขา 10 , ขา 11 กับขา 12 , ขา 13 กับขา 14 และขา 15 กับขา 16 ที่ connectorของวงจร
2. ป้อนสัญญาณลอจิก "0" ที่ขาสัญญาณ $\overline{R}$, $\overline{RD}$ และขา $\overline{INTR}$ ที่ connector ขา 27 , 28 และ 29 ตามลำดับ
3. ป้อนสัญญาณเสียงเข้าที่ Audio in ของวงจร
4. ต่อชุดขยายเสียงที่ Audio out ของวงจร
5. ทำการจ่ายไฟให้แก่วงจร
6. สังเกตผลการทดลองจากการฟังเสียง

ผลการทดลอง

จากการทดลอง โดยการป้อนสัญญาณเสียงเข้าที่วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และฟังเสียงที่ได้จากวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก ซึ่งได้ต่อไว้กับเครื่องขยายเสียง ผลปรากฏว่า เสียงที่ได้ค่อนข้างมีความแตกต่างจากเสียงต้นฉบับพอสมควร แต่โดยรวมแล้วสามารถจะจับใจความ และตีความหมายของเสียงที่ได้ โดยเฉพาะเสียงพูดได้ดี

4.2 การจำลองทำงานของโปรแกรม

ในส่วนของการจำลองการทำงานของโปรแกรมนั้นได้ใช้โปรแกรมจำลองการทำงาน logic simulator จากโปรแกรม Xilinx Foundation ในการแสดงผลการจำลองการทำงาน ซึ่งจะเป็นการทดสอบโปรแกรมด้วยกันสี่ส่วนด้วยกัน คือ ส่วนโปรแกรม Buffer , ส่วนโปรแกรม Parallel in serial out , ส่วนโปรแกรม DCT , ส่วนโปรแกรม IDCT ซึ่งเป็นการตรวจสอบความถูกต้องในการทำงานของโปรแกรม

4.2.1 การจำลองการทำงานส่วนโปรแกรม Buffer

ในส่วนของการจำลองการทำงานในส่วนนี้จะทำการป้อนสัญญาณลอจิกขนาด 8 บิต เป็นการนับขึ้น และผลที่ได้จากการทำงานของโปรแกรม คือเมื่อส่วนของโปรแกรม Buffer ทำการเรียงข้อมูลอินพุตได้ครบ 8 ชุด จะทำการส่องข้อมูลทั้งแปดออกเป็นลักษณะข้อมูลขนาน และจะส่งสัญญาณลอจิก “1” ที่ขา DONE

ลำดับขั้นการทดลอง

1. เรียกโปรแกรมจำลองการทำงาน logic simulator จากโปรแกรม Xilinx Foundation
2. เลือกมาโคร Buff ในส่วนของ chip selection
3. ทำการเลือกสัญญาณอินพุต และเอาต์พุต และป้อนสัญญาณตาม ตารางที่ 4.1

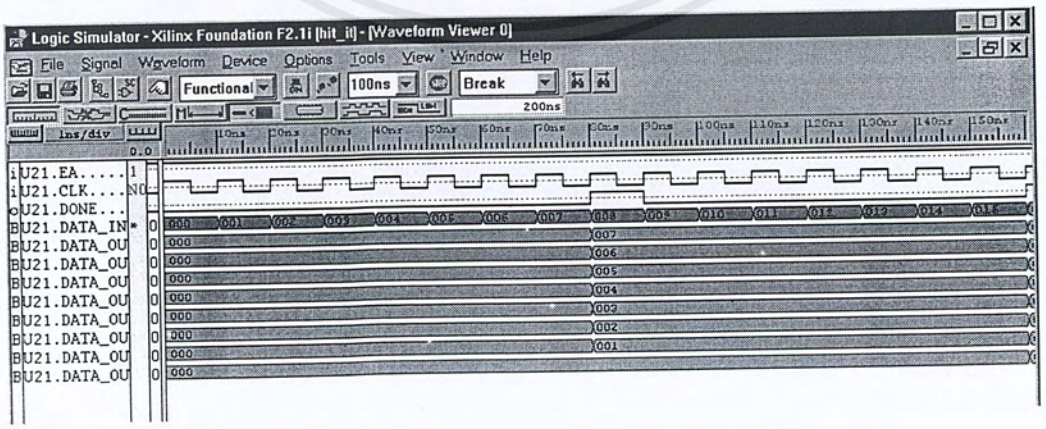
ตารางที่ 4.1 การป้อนสัญญาณจำลองการทำงานโปรแกรม Buff

สัญญาณอินพุต	
ชื่อขาสัญญาณ	สัญญาณที่ป้อน
CLK	Nbe 0
EA	1
DATA_IN7	Be 1
DATA_IN6	Be 2
DATA_IN5	Be 3
DATA_IN4	Be 4
DATA_IN3	Be 5
DATA_IN2	Be 6
DATA_IN1	Be 7
DATA_IN0	Be 8

4. ทำการรันการจำลองการทำงานโดยเลือกที่ simulation step
5. สังเกตและบันทึกผลการทดลอง

ผลการทดลอง

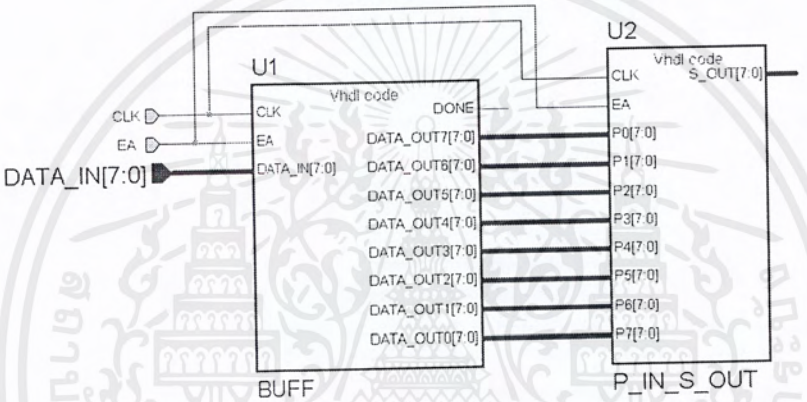
จากผลการจำลองการทำงานที่ได้ โปรแกรมสามารถทำงานได้ถูกต้องตามลำดับการทำงานที่ได้เขียนไว้ในโปรแกรม โดยผลการจำลองการทำงานแสดงไว้ดังในรูปที่ 4.1



รูปที่ 4.1 ผลการจำลองการทำงานโปรแกรม Buffer

4.2.2 การจำลองการทำงานส่วนโปรแกรม Parallel in Serial out

ในการจำลองการทำงานในส่วนนี้เป็นการทดสอบการทำงานของโปรแกรม Parallel in Serial out ซึ่งจะทำการเรียงข้อมูลที่เป็นลักษณะข้อมูลขนาน ให้เป็นข้อมูลลักษณะอนุกรม โดยการทดลองในส่วนนี้จะใช้โปรแกรมส่วน Buffer มาต่อร่วมด้วยเพื่อการป้อนสัญญาณที่ง่ายขึ้น และง่ายต่อการอ่านผลการทดลอง โดยต้องทำการต่อวงจรดังรูปที่ 4.2 ในส่วนของ Schematic Editor



รูปที่ 4.2 แสดงการเชื่อมต่อส่วนโปรแกรม Buff และ Parallel in Serial out

ลำดับขั้นการทดลอง

1. ทำการต่อวงจรดังรูปที่ 4.2 โดยใช้ส่วนโปรแกรม Schematic Editor จากโปรแกรม Xilinx Foundation
2. เรียกโปรแกรมจำลองการทำงาน logic simulator จากโปรแกรม Xilinx Foundation
3. ทำการเลือกสัญญาณอินพุต และเอาต์พุต และป้อนสัญญาณตาม ตารางที่ 4.2

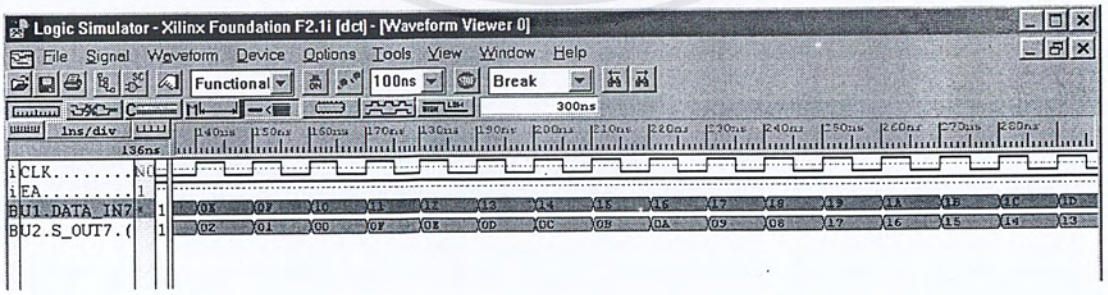
ตารางที่ 4.2 การป้อนสัญญาณจำลองการทำงานโปรแกรม Parallel in Serial out

สัญญาณอินพุต	
ชื่อขาสัญญาณ	สัญญาณที่ป้อน
CLK	Nbe 0
EA	1
DATA_IN7	Be 1
DATA_IN6	Be 2
DATA_IN5	Be 3
DATA_IN4	Be 4
DATA_IN3	Be 5
DATA_IN2	Be 6
DATA_IN1	Be 7
DATA_IN0	Be 8

4. ทำการรันการจำลองการทำงานโดยเลือกที่ simulation step
5. สังเกตและบันทึกผลการทดลอง

ผลการทดลอง

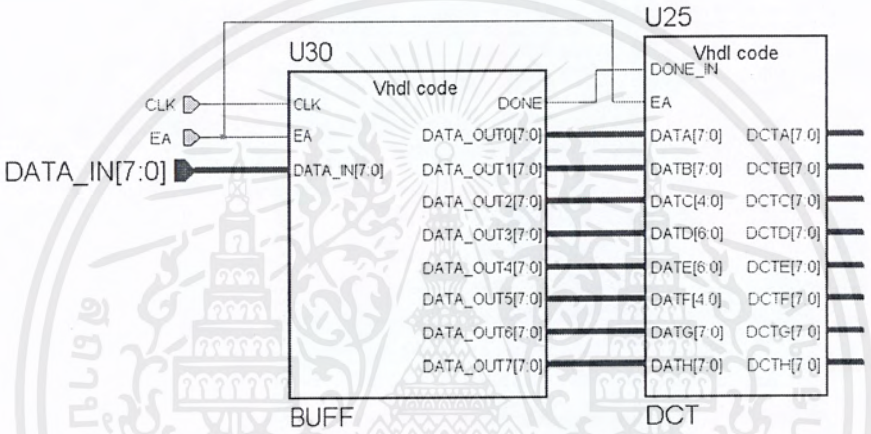
จากผลการจำลองการทำงานที่ได้ โปรแกรมสามารถทำงานได้ถูกต้องตามลำดับการทำงานที่ได้เขียนไว้ในโปรแกรม โดยผลการจำลองการทำงานแสดงไว้ดังในรูปที่ 4.3



รูปที่ 4.3 ผลการจำลองการทำงานโปรแกรม Parallel in Serial out

4.2.3 การจำลองการทำงานส่วนโปรแกรม DCT

ในการจำลองการทำงานในส่วนนี้เป็นการทดสอบการทำงานของโปรแกรม DCT ซึ่งเป็นส่วนที่ทำการประมวลข้อมูลจากส่วนโปรแกรม Buffer โดยข้อมูลที่ได้ในส่วนเอาต์พุตจะเป็นลักษณะข้อมูลขนาน โดยการทดลองในส่วนนี้จะใช้โปรแกรมส่วน Buffer มาต่อร่วมด้วย โดยต้องทำการต่อวงจรดังรูปที่ 4.5 ในส่วนของ Schematic Editor



รูปที่ 4.4 แสดงการเชื่อมต่อส่วนโปรแกรม Buff และโปรแกรม DCT

ลำดับขั้นการทดลอง

1. ทำการต่อวงจรดังรูปที่ 4.4 โดยใช้ส่วนโปรแกรม Schematic Editor จากโปรแกรม Xilinx Foundation
2. เรียกโปรแกรมจำลองการทำงาน logic simulator จากโปรแกรม Xilinx Foundation
3. ทำการเลือกสัญญาณอินพุต และเอาต์พุต และป้อนสัญญาณตาม ตารางที่ 4.3

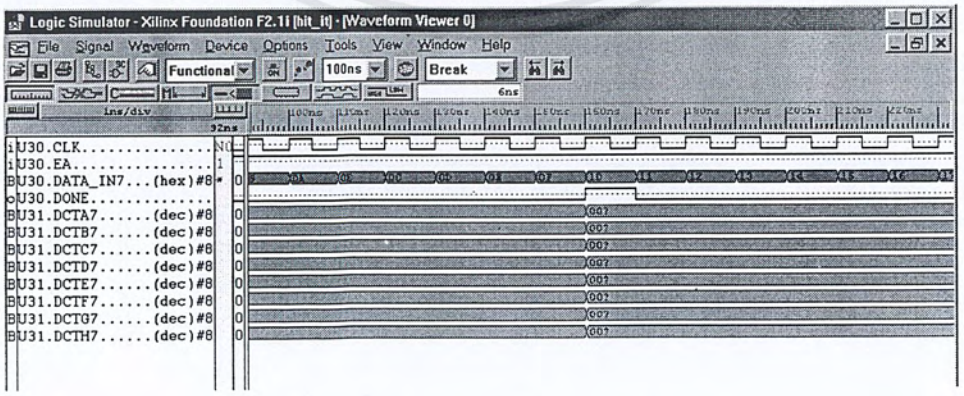
ตารางที่ 4.3 การป้อนสัญญาณจำลองการทำงานโปรแกรม DCT

สัญญาณอินพุต	
ชื่อขาสัญญาณ	สัญญาณที่ป้อน
CLK	Nbe 0
EA	1
DATA_IN7	Be 1
DATA_IN6	Be 2
DATA_IN5	Be 3
DATA_IN4	Be 4
DATA_IN3	Be 5
DATA_IN2	Be 6
DATA_IN1	Be 7
DATA_IN0	Be 8

4. ทำการรันการจำลองการทำงานโดยเลือกที่ simulation step
5. สังเกตและบันทึกผลการทดลอง

ผลการทดลอง

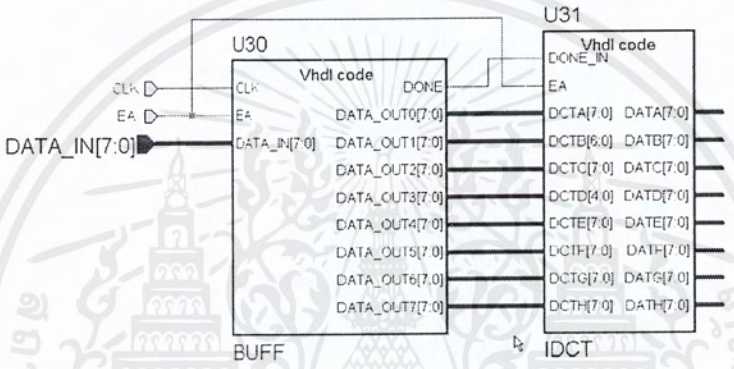
จากผลการจำลองการทำงานที่ได้ โปรแกรมสามารถทำงานได้ถูกต้องตามลำดับการทำงานที่ได้เขียนไว้ในโปรแกรม โดยผลการจำลองการทำงานแสดงไว้ดังในรูปที่ 4.5



รูปที่ 4.5 ผลการจำลองการทำงานโปรแกรม DCT

4.2.4 การจำลองการทำงานส่วนโปรแกรม IDCT

ในการจำลองการทำงานในส่วนนี้เป็นการทดสอบการทำงานของโปรแกรม DCT ซึ่งเป็นส่วนที่ทำการประมวลข้อมูลกลับจากส่วนโปรแกรม DCT โดยข้อมูลที่ได้ในส่วนเอาต์พุตจะเป็นลักษณะข้อมูลขนาน โดยการทดลองในส่วนนี้จะใช้โปรแกรมส่วน Buffer มาต่อร่วมด้วย โดยต้องทำการต่อวงจรดังรูปที่ 4.6 ในส่วนของ Schematic Editor



รูปที่ 4.6 แสดงการเชื่อมต่อส่วนโปรแกรม Buff และโปรแกรม IDCT

ลำดับขั้นการทดลอง

1. ทำการต่อวงจรดังรูปที่ 4.6 โดยใช้ส่วนโปรแกรม Schematic Editor จากโปรแกรม Xilinx Foundation
2. เรียกโปรแกรมจำลองการทำงาน logic simulator จากโปรแกรม Xilinx Foundation
3. ทำการเลือกสัญญาณอินพุต และเอาต์พุต และป้อนสัญญาณตาม ตารางที่ 4.4

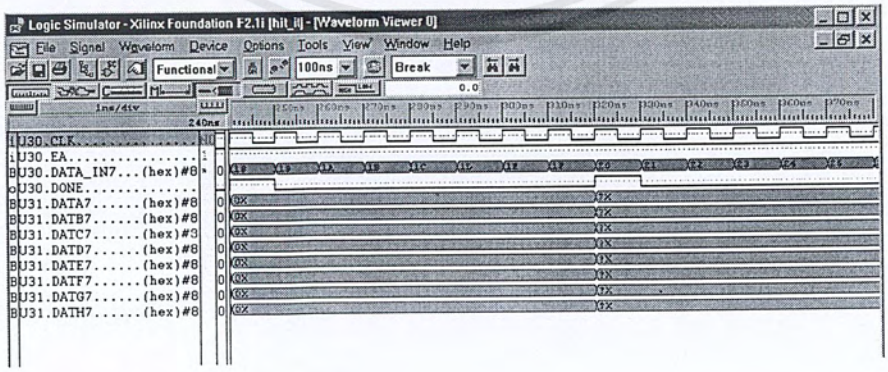
ตารางที่ 4.4 การป้อนสัญญาณจำลองการทำงานโปรแกรม DCT

สัญญาณอินพุต	
ชื่อขาสัญญาณ	สัญญาณที่ป้อน
CLK	Nbe 0
EA	1
DATA_IN7	Be 1
DATA_IN6	Be 2
DATA_IN5	Be 3
DATA_IN4	Be 4
DATA_IN3	Be 5
DATA_IN2	Be 6
DATA_IN1	Be 7
DATA_IN0	Be 8

4. ทำการรันการจำลองการทำงานโดยเลือกที่ simulation step
5. สังเกตและบันทึกผลการทดลอง

ผลการทดลอง

จากผลการจำลองการทำงานที่ได้ โปรแกรมสามารถทำงานได้ถูกต้องตามลำดับการทำงานที่ได้เขียนไว้ในโปรแกรม โดยผลการจำลองการทำงานแสดงไว้ดังในรูปที่ 4.7

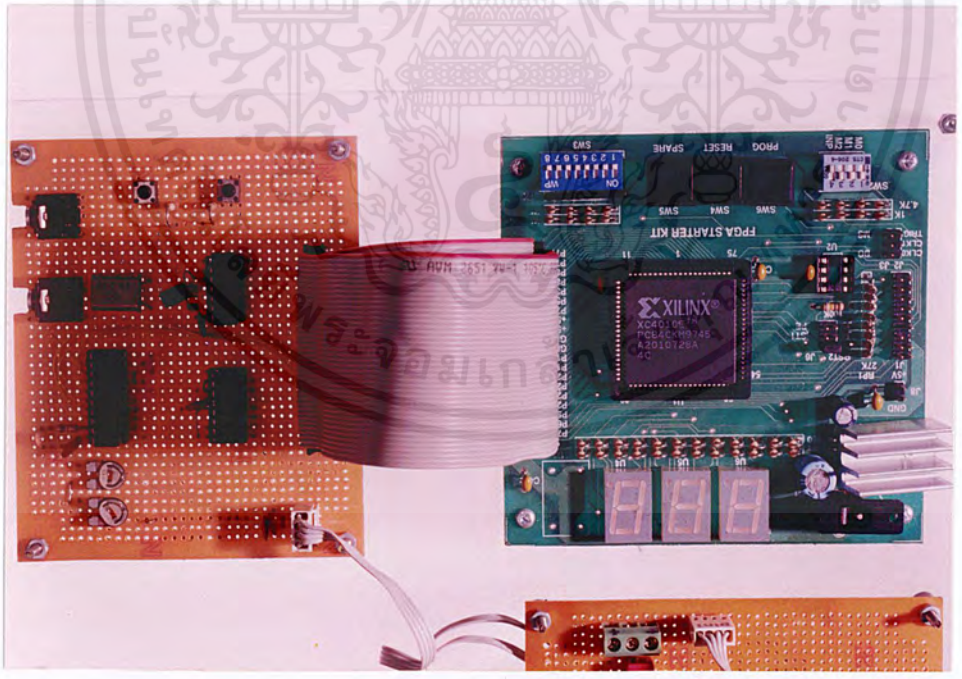


รูปที่ 4.7 ผลการจำลองการทำงานโปรแกรม IDCT

4.3 การทดลองการบีบอัดสัญญาณเสียง

ในการทดลองนี้ถือได้ว่าเป็นการทดสอบการทำงานของทั้งส่วนของวงจร และส่วนของโปรแกรมเนื่องจากการทดลองนี้จะทำการเชื่อต่อระหว่างวงจรวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และ วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก กับส่วนของ FPGA ซึ่งได้ทำการ Down load โปรแกรมการทำงานในส่วนต่าง ๆ ไว้เรียบร้อยแล้ว โดยที่การเชื่อมต่อวงจรทั้งสองได้แสดงไว้ในรูปที่ 4.8 ดังนั้นการทดลองในหัวข้อนี้จึงเป็นการทดสอบคุณภาพของสัญญาณเสียงที่ได้หลังจากการบีบอัด โดยใช้สัญญาณเสียงลักษณะต่าง ๆ ซึ่งได้แก่ สัญญาณเสียงรูปคลื่น sine wave , สัญญาณเสียงรูปคลื่น saw tooth , สัญญาณเสียงรูปคลื่น square wave , เสียงผู้หญิง และเสียงผู้ชาย

โดยการทดลองป้อนสัญญาณเสียงแต่ละประเภทเข้าไป สามารถอัดเสียงที่ได้หลังจากการบีบอัด แล้วนำมาดูลักษณะของสัญญาณเสียงที่ได้ เปรียบเทียบกับสัญญาณอินพุต ว่ามีความแตกต่างหรือมีความเพี้ยนเป็นอย่างไร ทั้งยังสามารถวิเคราะห์สัญญาณเสียงในลักษณะ สามมิติได้ ทั้งนี้ในการทดลองส่วนนี้ได้ใช้โปรแกรมสำเร็จรูป Wave Lab 2.0 ของบริษัท Steinberg ในการวิเคราะห์



รูปที่ 4.8 แสดงการเชื่อมต่อระหว่างส่วนวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก กับส่วนของ FPGA

4.3.1 การทดลองโดยใช้โปรแกรมสำเร็จรูป Wave Lab 2.0 ช่วยวิเคราะห์

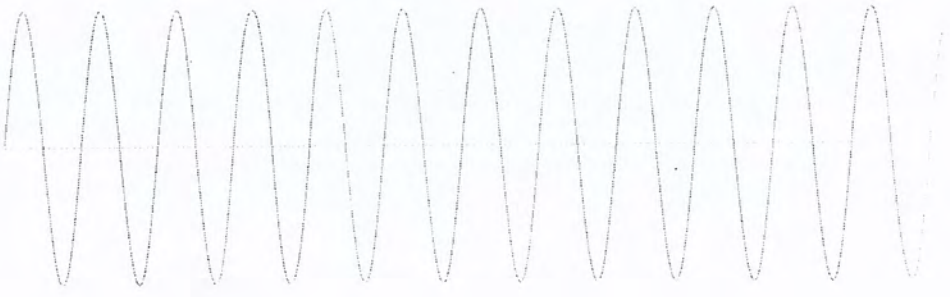
การทดลองในหัวข้อนี้ จะทำการบันทึกเสียงที่ได้จากการบีบอัดสัญญาณเสียงแล้ว เก็บไว้ยังเครื่องไมโครคอมพิวเตอร์ เพื่อใช้โปรแกรมสำเร็จรูป Wave Lab 2.0 วิเคราะห์ลักษณะสัญญาณที่ได้เปรียบเทียบกับสัญญาณเสียงต้นฉบับ

ลำดับขั้นการทดลอง

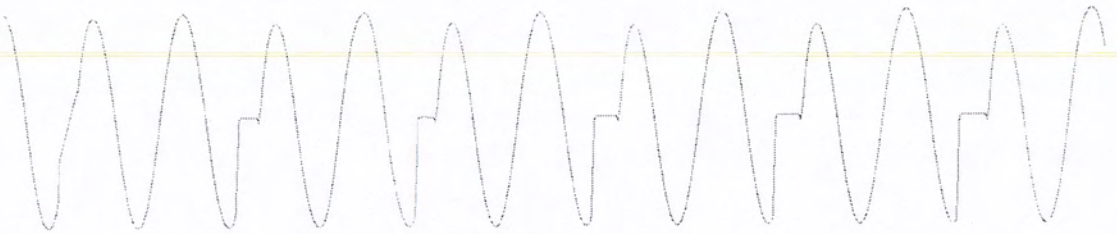
1. เชื่อมต่อวงจรทั้งหมดเข้าด้วยกัน
2. เชื่อมต่อ Audio in กับ Phone out จากซาว์ดการ์ดของเครื่องไมโครคอมพิวเตอร์ และเชื่อมต่อ Audio out กับ Aux in จากซาว์ดการ์ดของเครื่องไมโครคอมพิวเตอร์เช่นกัน
3. เรียกโปรแกรมสำเร็จรูป Wave Lab 2.0 เปิดแฟ้มเลือกสัญญาณเสียง Sine.wav
4. เลือก Icon Record แล้วทำการบันทึกเสียง โดยเลือกที่ Recoard
5. ทำการ Play สัญญาณเสียงที่ได้เลือกไว้
6. เมื่อบันทึกเสียงตัวอย่างจนจบ ทำการ Save ไฟล์ที่ได้จากการบันทึกเสียง
7. นำสัญญาณเสียงที่ได้จากการบันทึก เปรียบเทียบกับสัญญาณเสียงต้นฉบับ และบันทึกผลการสังเกต
8. เลือกสัญญาณเสียงต้นฉบับ แล้วทำการเลือกที่ เมนู 3D Frequency Analysis
9. เลือกสัญญาณเสียงที่ได้จากการบันทึกแล้วทำการเลือกที่ เมนู 3D Frequency Analysis
10. เปรียบเทียบ และวิเคราะห์จากกราฟ 3 มิติที่ได้ พร้อมทั้งบันทึกผลการทดลอง

ผลการทดลอง

จากเสียงต้นฉบับดังแสดงในรูปที่ 4.9 เปรียบเทียบกับเสียงที่ได้จากการบันทึกหลังจากการบีบอัดในรูปที่ 4.10 แสดงให้เห็นว่า ลักษณะของรูปคลื่น Sine มีลักษณะที่เปลี่ยนไป หรือเพี้ยนไปจากต้นฉบับ

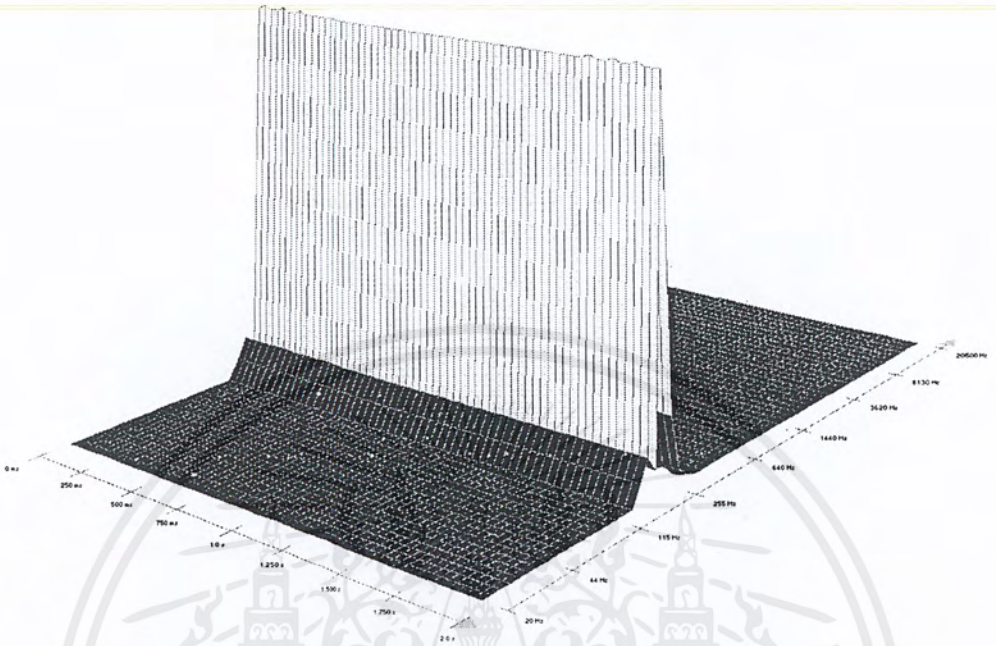


รูปที่ 4.9 แสดงลักษณะสัญญาณเสียง Sine Wave

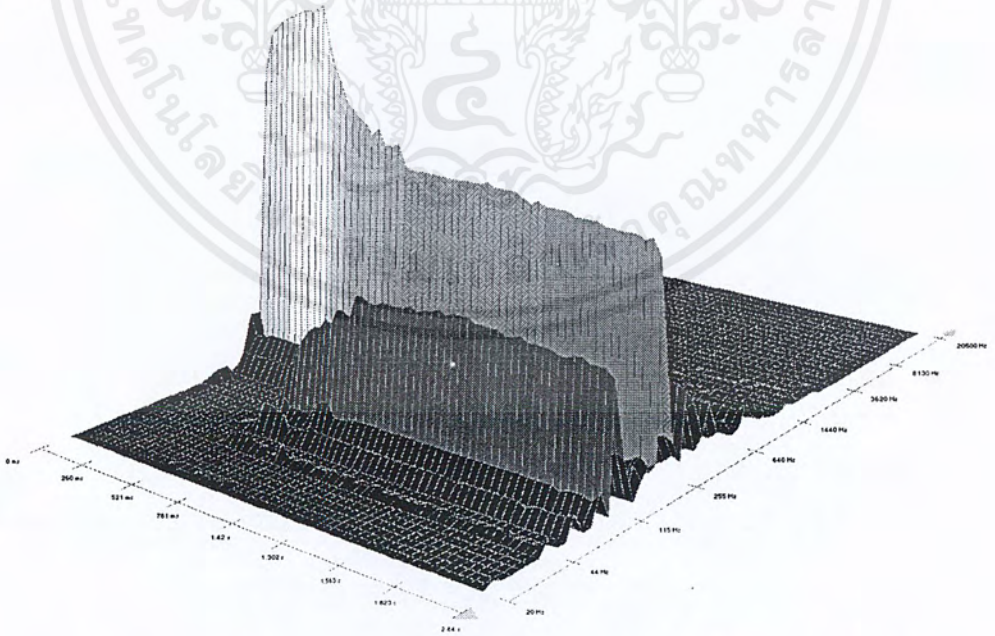


รูปที่ 4.10 แสดงลักษณะสัญญาณเสียงที่ได้หลังจากการบีบอัด

จากการใช้การวิเคราะห์สามมิติของเสียงต้นฉบับในรูปที่ 4.11 กับเสียงที่ได้จากการบันทึกหลังจากการบีบอัดในรูปที่ 4.12 แสดงให้เห็นว่า ในช่วงแรกเสียงที่ได้ค่อนข้างมีลักษณะเหมือนกับสัญญาณเสียงต้นฉบับ แต่ในช่วงเวลาต่อมาได้เกิดคลื่นแทรกขึ้นมาที่นอกเหนือจากย่านความถี่เดิมของเสียงต้นฉบับ ซึ่งก็คือเกิดความเพี้ยนเกิดขึ้น ทำให้แอมปลิจูดของความถี่หลักลดลงและไปเพิ่มในส่วนของสัญญาณรบกวนนอกเหนือจากความถี่หลัก จากการสังเกตความถี่เพี้ยนที่เกิดขึ้นส่วนใหญ่จะเกิดขึ้นในความถี่ที่ต่ำกว่าความถี่หลัก แต่จากภาพสัญญาณเสียงหลักโดยรวมแล้วยังถือว่าเสียงที่ได้หลังจากการบีบอัด ยังคงรักษาความถี่หลักไว้ได้



รูปที่ 4.11 แสดงการวิเคราะห์สามมิติ ของสัญญาณเสียง Sine Wave



รูปที่ 4.12 แสดงการวิเคราะห์สามมิติ ของสัญญาณเสียงที่ได้หลังจากการบีบอัด

4.3.2 การทดลองจากการฟังเสียง

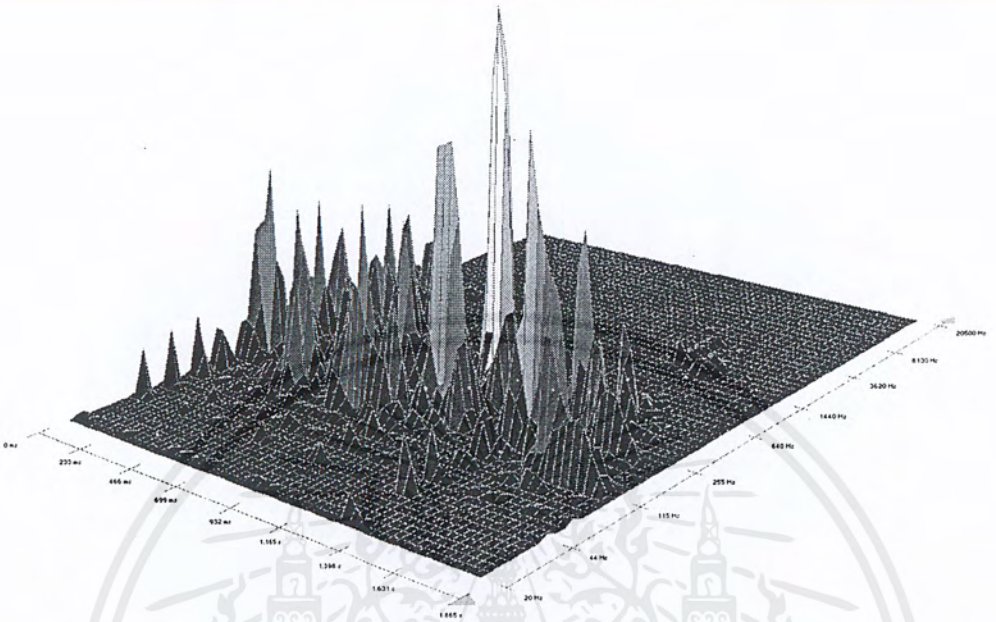
การทดลองในหัวข้อนี้จะเป็นการทดสอบเสียงโดยการฟังจับใจความจากเสียงพูด โดยจะใช้เสียงพูดอยู่สองลักษณะคือ เสียงพูดของผู้ชาย และเสียงพูดของผู้หญิง

ลำดับขั้นการทดลอง

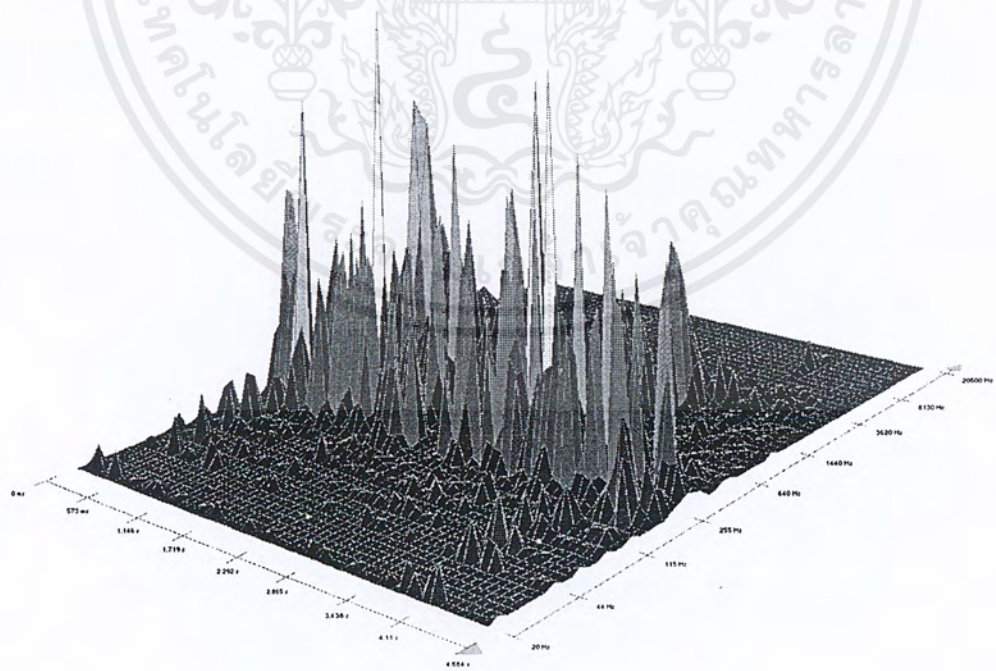
1. เชื่อมต่อวงจรทั้งหมดเข้าด้วยกัน
2. เชื่อมต่อ Audio in กับ Phone out จากซาว์ดการ์ดของเครื่องไมโครคอมพิวเตอร์ และเชื่อมต่อ Audio out กับ อุปกรณ์ขยายเสียง
3. เรียกโปรแกรมสำเร็จรูป Wave Lab 2.0 เปิดแฟ้มเลือกสัญญาณเสียง Man.wav
4. ทำการ Play สัญญาณเสียงที่เลือกไว้
5. ฟังเสียงที่ได้จากการบีบอัดเปรียบเทียบกับเสียงต้นฉบับ แล้วบันทึกผลการทดลอง
6. เปิดแฟ้มเลือกสัญญาณเสียง Woman.wav
7. ทำการ Play สัญญาณเสียงที่เลือกไว้
8. ฟังเสียงที่ได้จากการบีบอัดเปรียบเทียบกับเสียงต้นฉบับ แล้วบันทึกผลการทดลอง

ผลการทดลอง

จากการผลการรับฟังเพื่อจับใจความ เสียงของผู้ชายไม่สามารถจับใจความได้ชัดเจน แต่ในกรณีของเสียงผู้หญิงสามารถจับใจความได้ชัดเจนกว่าเสียงผู้ชาย ซึ่งเมื่อเปรียบเทียบกับเสียงต้นฉบับแล้วเสียงที่ได้หลังจากการบีบอัดในกรณีของเสียงพูดนั้น ค่อนข้างจะมีความเพี้ยนอยู่สูง และในช่วงเสียงของเสียงผู้ชาย ซึ่งมีกลุ่มของความถี่เสียงอยู่ในความถี่ต่ำดังรูปที่ 4.13 จะเกิดสัญญาณเพี้ยนมากกว่าเสียงผู้หญิงที่อยู่กลุ่มความถี่เสียงที่สูงกว่า ดังรูปที่ 4.14



รูปที่ 4.13 แสดงการวิเคราะห์สามมิติ ของสัญญาณเสียงพูดผู้ชาย



รูปที่ 4.14 แสดงการวิเคราะห์สามมิติ ของสัญญาณเสียงพูดผู้หญิง

บทที่ 5
บทสรุป แนวทางแก้ไข และพัฒนา

5.1 บทสรุป

ในการจัดทำโครงงานครั้งนี้ ถือได้ว่าเป็นการนำความรู้ที่มีในบทเรียน และนอกเหนือจากบทเรียน ที่ได้เรียนมา จัดเป็นผลทำให้เกิดการเรียนรู้จากการค้นคว้า ซึ่งเป็นส่วนที่สำคัญยิ่งกว่าความสำเร็จของโครงงานเองเนื่องจาก การค้นคว้านั้นสามารถที่จะให้ความรู้ได้มาก และกว้างกว่าการเรียนในบทเรียน

ประโยชน์ของการทำโครงงานก็คงจะเป็นความรู้ และทักษะในการทำงานจริง ทั้งยัง รู้จักการทำงานเป็นทีม เป็นสิ่งที่ให้มากกว่าชิ้นงานที่สมบูรณ์

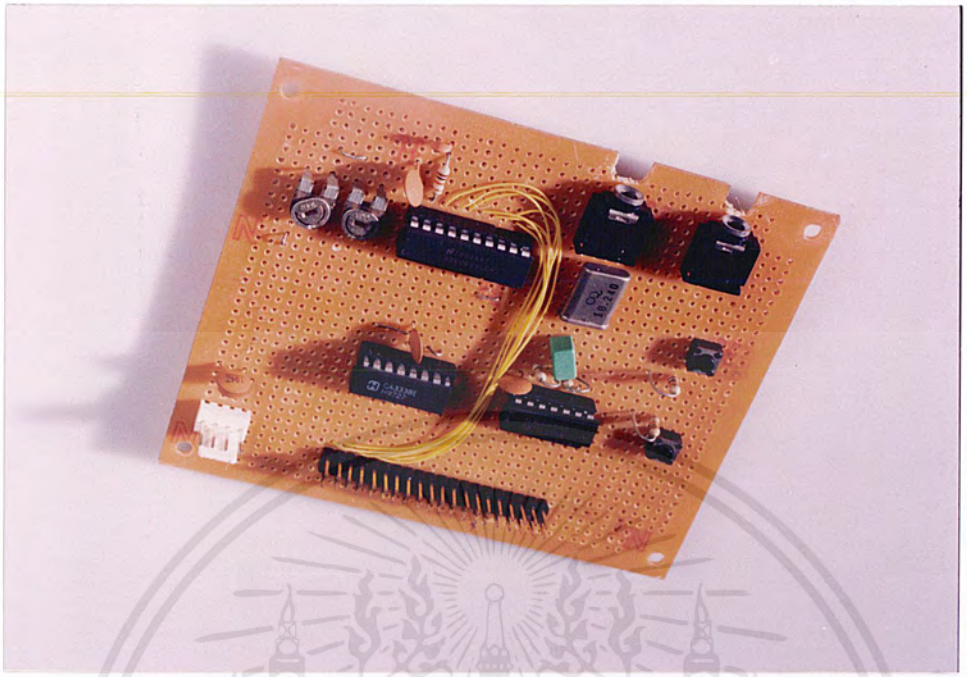
5.2 ปัญหาที่เกิดขึ้นในการจัดทำโครงงาน

1. ทางผู้จัดทำมีความรู้และความเข้าใจในเรื่อง Digital signal processing ไม่มากเท่าใดทำให้ต้องใช้เวลาในการหาข้อมูล และทำความเข้าใจมากพอสมควรซึ่งในส่วนนี้ก็ได้รับความช่วยเหลือจากอาจารย์ที่ปรึกษา คือ อาจารย์กิติพงค์ มะโน เป็นอย่างมาก ทำให้ทางผู้จัดทำมีความเข้าใจส่วนของทฤษฎีการจัดทำโครงงานมากขึ้น
2. ข้อจำจัดของภาษา VHDL ที่ไม่สามารถที่จะรองรับการทำงานได้ทุกจุดโดยเฉพาะอย่างยิ่งในส่วนของการคำนวณที่ซับซ้อน ซึ่งในส่วนนี้ทำให้ชุดคำสั่งมีขนาดที่ใหญ่และจำนวณของแต่ละภาคมีจำนวนมาก ทำให้ใช้จำนวนเกตในตัว FPGAs มาก

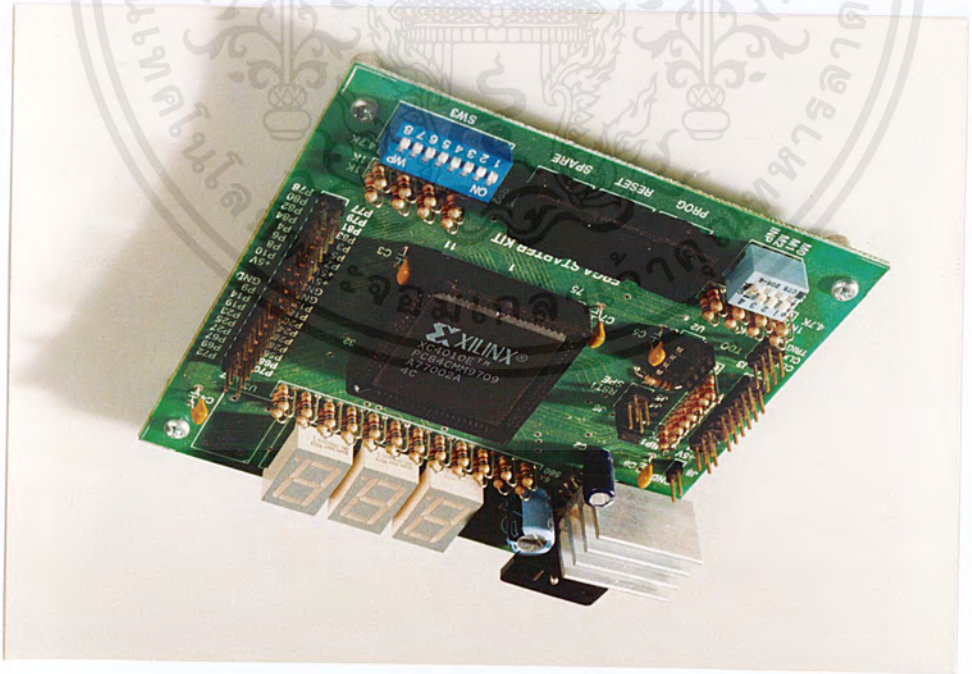
5.3 แนวทางการแก้ไขปัญหา และพัฒนา

1. ในส่วนของภาคประมวลผลสัญญาณ DCT และ IDTC นั้น ทางผู้จัดทำได้ใช้ระบบการคำนวณแบบจำนวนเต็ม (Fixed – Point) ซึ่งจะให้ค่าที่ให้ความคลาดเคลื่อนสูงกว่าแบบคิดจุดทศนิยม (Floating – Point) เนื่องจากต้องประมาณค่า แต่เนื่องจากการคำนวณแบบจุดทศนิยมนั้นยากและมีกระบวนการที่ซับซ้อน แต่ให้ผลลัพธ์ที่มีความถูกต้องสูง
2. ในส่วนของการออกแบบชุดคำสั่ง หากต้องการที่จะให้ชุดคำสั่งมีขนาดที่เล็กและใช้จำนวนเกตภายใน FPGAs น้อย ควรมีการออกแบบชุดคำสั่งในลักษณะของการทำงานแบบ RTL

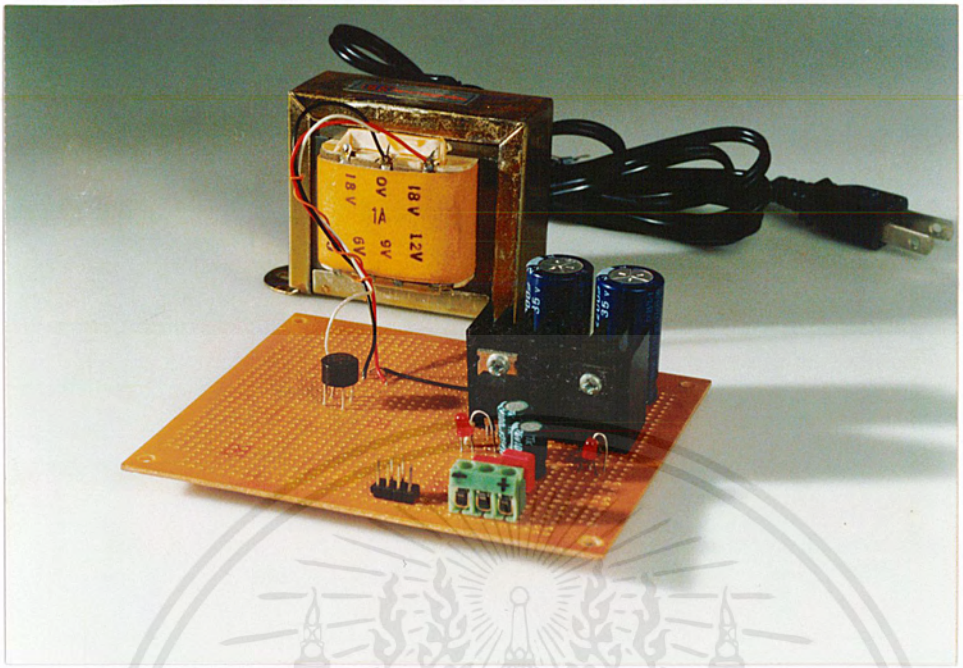
ภาคผนวก ก
เครื่องต้นแบบ



รูปที่ ก.1 เครื่องต้นแบบส่วนวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และวงจรแปลงสัญญาณดิจิตอล เป็นสัญญาณแอนะลอก



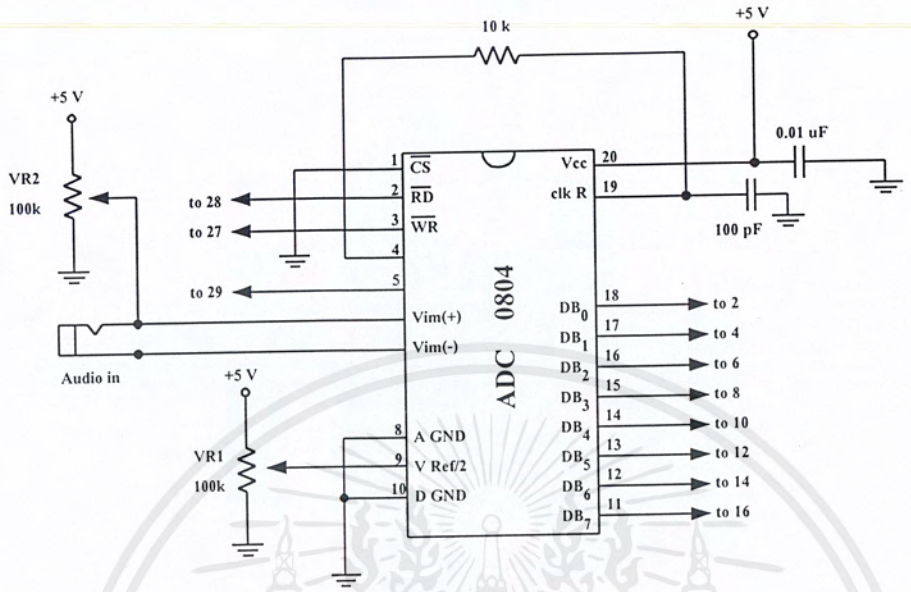
รูปที่ ก.2 เครื่องต้นแบบส่วน FPGA



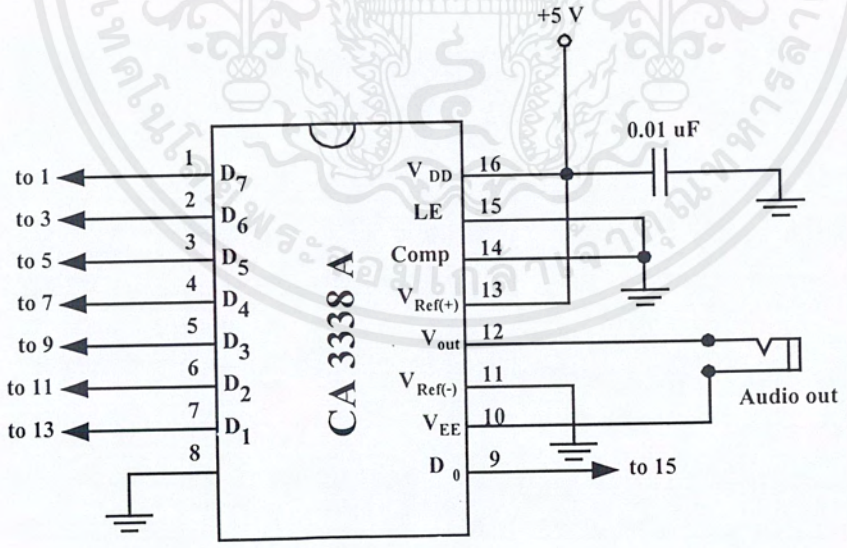
รูปที่ ก.3 เครื่องต้นแบบส่วนภาคจ่ายไฟ

ภาคผนวก ข

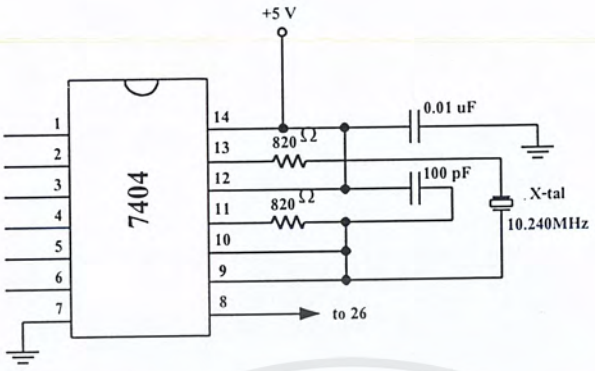
วงจรการทำงาน



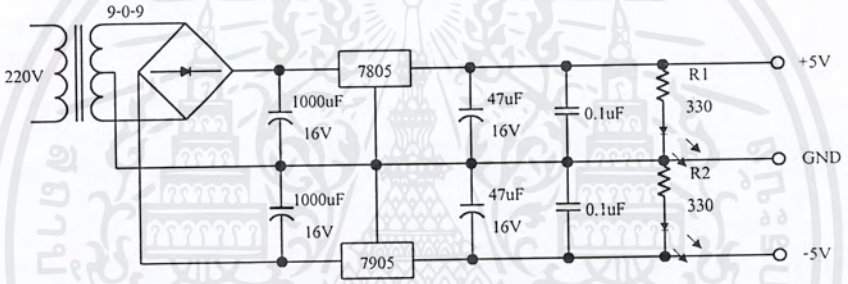
รูปที่ ข.1 วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล



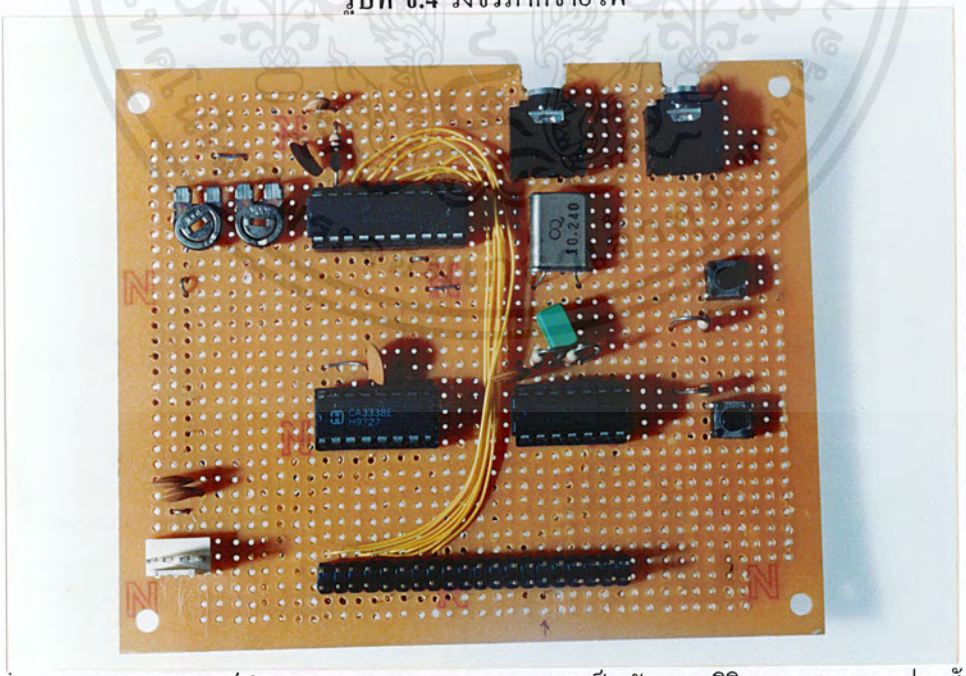
รูปที่ ข.2 วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก



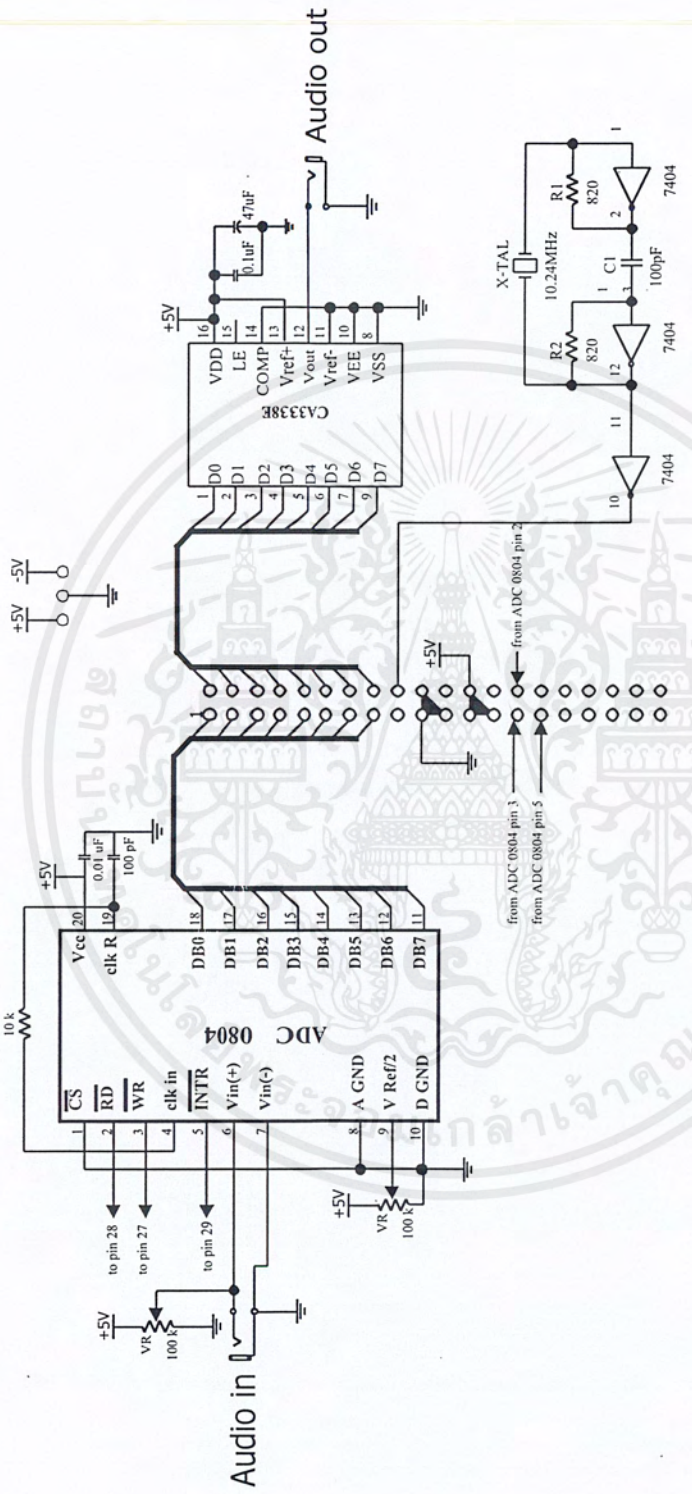
รูปที่ ข.3 วงจรกำเนิดสัญญาณนาฬิกา



รูปที่ ข.4 วงจรภาคจ่ายไฟ



รูปที่ ข.5 แผ่นวงจรพิมพ์ส่วนวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล และวงจรแปลงสัญญาณดิจิตอล เป็นสัญญาณแอนะลอก



รูปที่ ข.6 การเชื่อมต่อวงจรการทำงานเข้าด้วยกัน

ภาคผนวก ค
โปรแกรมการทำงาน

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

entity project_H is
   port (
         data_in: in INTEGER range 0 to 255;
         clk: in STD_LOGIC;
         EA: in STD_LOGIC;
         done: out STD_LOGIC;
         data_out0: out INTEGER range 0 to 255;
         data_out1: out INTEGER range 0 to 255;
         data_out2: out INTEGER range 0 to 255;
         data_out3: out INTEGER range 0 to 255;
         data_out4: out INTEGER range 0 to 255;
         data_out5: out INTEGER range 0 to 255;
         data_out6: out INTEGER range 0 to 255;
         data_out7: out INTEGER range 0 to 255
         );
end project_H;

architecture project_H_arch of project_H is


         signal   a0,a1,a2,a3,a4,a5,a6,a7,
                  b0,b1,b2,b3,b4,b5,b6,b7  : INTEGER range 0 to 255;
         signal ai,bi : integer range 0 to 7;
```

```
begin
process (clk,ea,ai)
begin

        if clk'event and clk ='1' then
        if ea = '1' then

        if ai = 7 then
        done <= '1';

        data_out0 <= a7;
        data_out1 <= a0;
        data_out2 <= a1;
        data_out3 <= a2;
        data_out4 <= a3;
        data_out5 <= a4;
        data_out6 <= a5;
        data_out7 <= a6;
        else done <= '0';
        end if;

        if      (ai=1) then a1 <= data_in;
        elsif   (ai=2) then a2 <= data_in;
        elsif   (ai=3) then a3 <= data_in;
        elsif   (ai=4) then a4 <= data_in;
        elsif   (ai=5) then a5 <= data_in;
        elsif   (ai=6) then a6 <= data_in;
        elsif   (ai=7) then a7 <= data_in;
        else a0 <= data_in;
        end if;
```

```
                ai <= (ai+1);


end if;
end if;
end process;
end project_H_arch;
```

รูปที่ ค.1 โปรแกรมส่วน Buffer

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


entity s_in_p_out is
   port (
      p0: in INTEGER range 0 to 255;
      p1: in INTEGER range 0 to 255;
      p2: in INTEGER range 0 to 255;
      p3: in INTEGER range 0 to 255;
      p4: in INTEGER range 0 to 255;
      p5: in INTEGER range 0 to 255;
      p6: in INTEGER range 0 to 255;
      p7: in INTEGER range 0 to 255;
      clk: in STD_LOGIC;
      ea: in STD_LOGIC;
      s_out: out INTEGER range 0 to 255
   );
end s_in_p_out;
```

```
architecture s_in_p_out_arch of s_in_p_out is

        signal ci : integer range 0 to 7;
begin
process (ea,clk,ci)

begin
--      ci <= 0;

        if clk'event and clk = '1'then
        if ea = '1' then

        if (ci=1) then s_out <= p1;
        elsif (ci=2) then s_out <= p2;
        elsif (ci=3) then s_out <= p3;
        elsif (ci=4) then s_out <= p4;
        elsif (ci=5) then s_out <= p5;
        elsif (ci=6) then s_out <= p6;
        elsif (ci=7) then s_out <= p7;
        else s_out <= p0;
        end if;

ci <= (ci+1);
end if;
end if;
end process;
end s_in_p_out_arch;
```

รูปที่ ค.2 โปรแกรมส่วน Parallel in Serial out

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

entity DCT is
   port (
         ea: in STD_LOGIC;
         done_in: in STD_LOGIC;
         datA: in INTEGER range 0 to 255;
         datb: in INTEGER range 0 to 255;
         datc: in INTEGER range 0 to 255;
         datd: in INTEGER range 0 to 255;
         date: in INTEGER range 0 to 255;
         datf: in INTEGER range 0 to 255;
         datg: in INTEGER range 0 to 255;
         dath: in INTEGER range 0 to 255;

         dctA: out INTEGER range -128 to 127;
         dctb: out INTEGER range -128 to 127;
         dctc: out INTEGER range -128 to 127;
         dctd: out INTEGER range -128 to 127;
         dcte: out INTEGER range -128 to 127;
         dctf: out INTEGER range -128 to 127;
         dctg: out INTEGER range -128 to 127;
         dcth: out INTEGER range -128 to 127
         );
end DCT;

architecture DCT_arch of DCT is
```

```
        signal s0,s1,s2,s3,s4,s5,s6,s7 : integer range -128 to 127;

begin
process (ea,done_in)

        type co is array (0 to 7) of integer;

        constant coeff0 : co := (7,10,9,8,7,5,3,1);
        constant coeff1 : co := (7,8,3,-1,-7,-10,-9,-5);
        constant coeff2 : co := (7,5,-3,-10,-7,1,9,8);
        constant coeff3 : co := (7,1,-9,-5,7,8,-3,-10);
        constant coeff4 : co := (7,-1,-9,5,7,-8,-3,10);
        constant coeff5 : co := (7,-5,-3,10,-7,-1,9,-8);
        constant coeff6 : co := (7,-8,3,1,-7,10,-9,5);
        constant coeff7 : co := (7,-10,9,-8,7,-5,3,-1);

begin

        if ea = '1' then
        if done_in'event and done_in = '1' then

        for i in 0 to 7 loop
        s0 <= data*coeff0(i);
        s1 <= datb*coeff1(i);
        s2 <= datc*coeff2(i);
        s3 <= datd*coeff3(i);
        s4 <= date*coeff4(i);
        s5 <= datf*coeff5(i);
        s6 <= datg*coeff6(i);
```

```
        s7 <= dath*coeff7(i);

        if i = 0 then
        dcta <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 1 then
        dctb <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 2 then
        dctc <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 3 then
        dctd <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 4 then
        dcte <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 5 then
        dctf <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 6 then
        dctg <= (s0+s1+s2+s3+s4+s5+s6+s7);
        else
        dcth <= (s0+s1+s2+s3+s4+s5+s6+s7);
        end if;

end loop;

end if;
end if;
end process;
end DCT_arch;
```

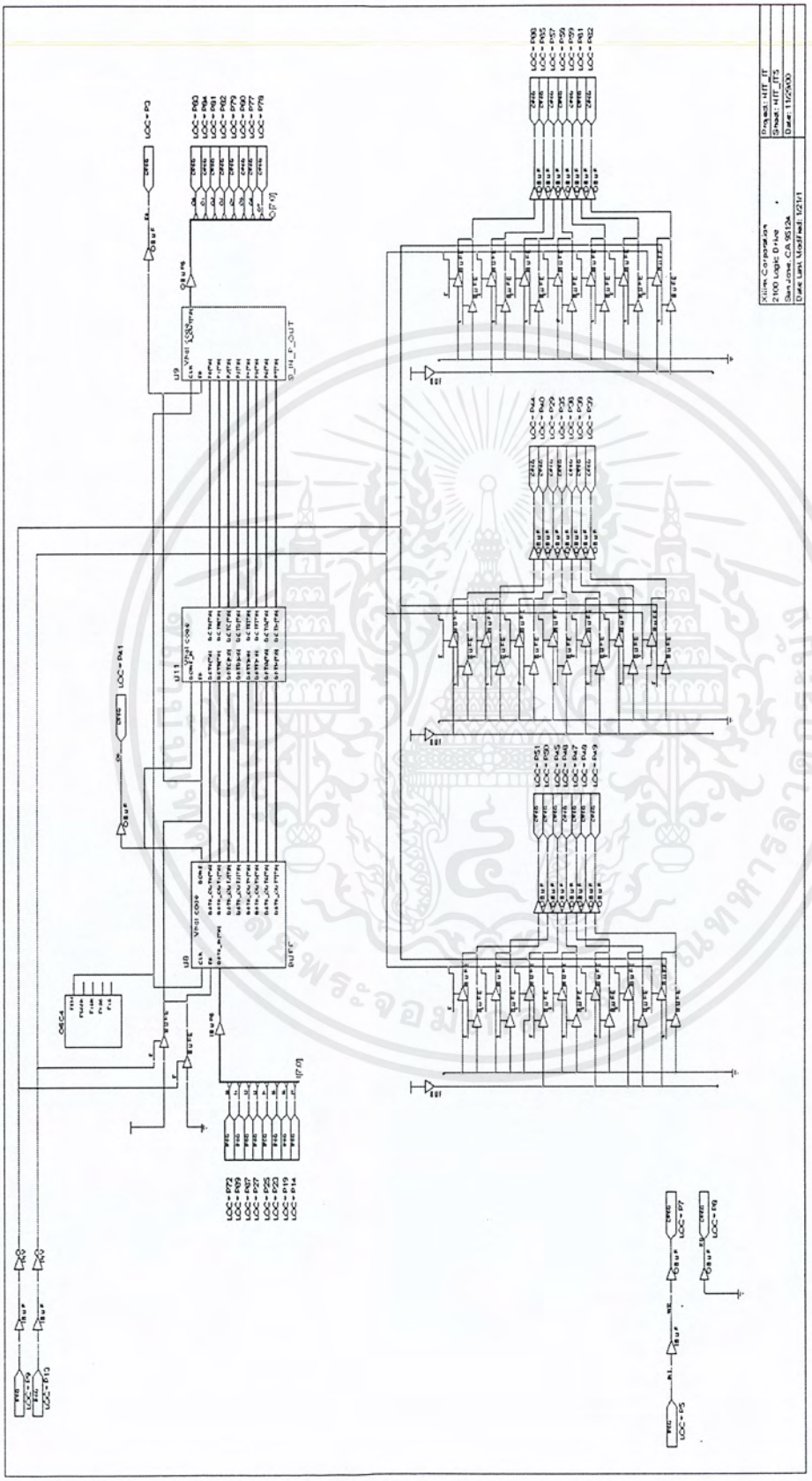
รูปที่ ค.3 โปรแกรมส่วน DCT

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

entity IDCT is
   port (
         ea: in STD_LOGIC;
         done_in: in STD_LOGIC;
         dctA: in INTEGER range -128 to 127;
         dctb: in INTEGER range -128 to 127;
         dctc: in INTEGER range -128 to 127;
         dctd: in INTEGER range -128 to 127;
         dcte: in INTEGER range -128 to 127;
         dctf: in INTEGER range -128 to 127;
         dctg: in INTEGER range -128 to 127;
         dcth: in INTEGER range -128 to 127;

         datA: out INTEGER range 0 to 255;
         datb: out INTEGER range 0 to 255;
         datc: out INTEGER range 0 to 255;
         datd: out INTEGER range 0 to 255;
         date: out INTEGER range 0 to 255;
         datf: out INTEGER range 0 to 255;
         datg: out INTEGER range 0 to 255;
         dath: out INTEGER range 0 to 255
         );
end IDCT;

architecture IDCT_arch of IDCT is
```

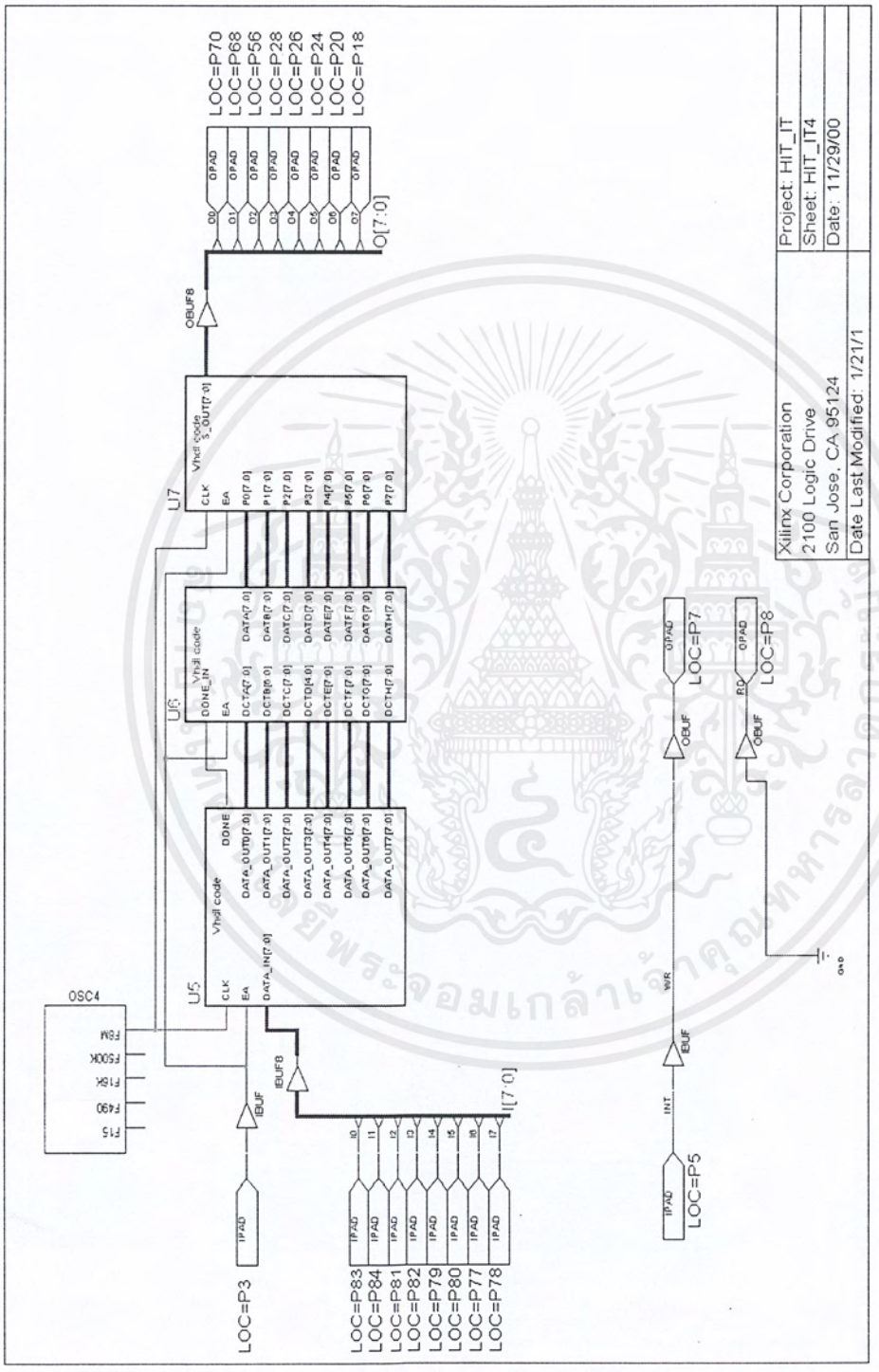
```
        signal s0,s1,s2,s3,s4,s5,s6,s7 : integer range -128 to 127;

begin
process (ea,done_in)

        type co is array (0 to 7) of integer;

        constant coeff0 : co := (7,7,7,7,7,7,7,7);
        constant coeff1 : co := (10,8,5,1,-1,-5,-8,-10);
        constant coeff2 : co := (9,3,-3,-9,-9,-3,3,9);
        constant coeff3 : co := (8,-1,-10,-5,5,10,1,-8);
        constant coeff4 : co := (7,-7,-7,7,7,-7,-7,7);
        constant coeff5 : co := (5,-10,1,8,-8,-1,10,-5);
        constant coeff6 : co := (3,-9,9,-3,-3,9,-9,3);
        constant coeff7 : co := (1,-5,8,-10,10,-8,5,-1);

begin

        if ea = '1' then
        if done_in'event and done_in = '1' then

        for i in 0 to 7 loop
        s0 <= dcta*coeff0(i);
        s1 <= dctb*coeff1(i);
        s2 <= dctc*coeff2(i);
        s3 <= dctd*coeff3(i);
        s4 <= dcte*coeff4(i);
        s5 <= dctf*coeff5(i);
        s6 <= dctg*coeff6(i);
```

```
        s7 <= dcth*coeff7(i);


        if i = 0 then
        data <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 1 then
        datb <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 2 then
        datc <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 3 then
        datd <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 4 then
        date <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 5 then
        datf <= (s0+s1+s2+s3+s4+s5+s6+s7);
        elsif i = 6 then
        datg <= (s0+s1+s2+s3+s4+s5+s6+s7);
        else
        dath <= (s0+s1+s2+s3+s4+s5+s6+s7);
        end if;

end loop;
end if;
end if;
end process;
end IDCT_arch;
```

รูปที่ ค.4 โปรแกรมส่วน IDCT



รูปที่ ค.5 วงจร Schematic ส่วน Buffer , DCT และส่วนแสดงผล



รูปที่ ค.6 วงจร Schematic ส่วนส่วน IDCT และParallel in Serial out

บรรณานุกรม

Omondi, A.R. **Computer Arithmetic System : Algorithms , Architecture and Implementation.** Cambridge : Prentice Hall International (UK) Limited. 1994

Sjoholm, S. and Lindh, L. **VHDL for Designers.** Hertfordshire : Prentice Hall Europe. 1997

Veldhuis, R. and Breeuwer, M. **An Introduction to Source Coding.** New York : Prentice – Hall, Inc. 1994

Xilinx, Inc. **Data Book : The Programmable Logic Data Book 1998.** California : Xilinx, Inc. 1997

Yalamanchili, S. **VHDL Starter's Guide.** New Jersey : Prentice – Hall , Inc. 1998

มนัส สังวงศิลป์ และวรรัตน์ ภัทรอมรกุล. **คู่มือโปรแกรม MATLAB ฉบับสมบูรณ์.** กรุงเทพฯ : อินโฟเพรส. 2543

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายสุทธิศักดิ์ สุขัมศรี
วัน เดือน ปี เกิด	วันที่ 15 ธันวาคม 2521
สถานที่เกิด	โรงพยาบาลมหาราชนครเชียงใหม่
ภูมิลำเนาเดิม	เชียงใหม่
ที่อยู่ปัจจุบัน	16 ถนนป่าตัน – บ้านท่อ ซอย 4 ตำบลป่าตัน อำเภอเมืองเชียงใหม่ จังหวัดเชียงใหม่ 50300
โทรศัพท์	(053) 232-963
ประวัติการศึกษา	
- ประถมศึกษา	โรงเรียนมงฟอร์ตวิทยาลัย (แผนกประถม)
- มัธยมศึกษาตอนต้น	โรงเรียนมงฟอร์ตวิทยาลัย (แผนกมัธยม)
- มัธยมศึกษาตอนปลาย	โรงเรียนยุพราชวิทยาลัย
- ประกาศนียบัตรวิชาชีพขั้นสูง (ปวส.)	สถาบันเทคโนโลยีราชมงคลวิทยาเขตภาคพายัพ สาขาวิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม
- ปริญญาตรี	ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
คติพจน์	“Everything must gone But famous is not”

ประวัติผู้แต่ง (ต่อ)

ชื่อผู้ทำปริญญานิพนธ์	นายเทพรัตน์ ภิรมย์รื่น
วัน เดือน ปี เกิด	วันที่ 26 พฤศจิกายน 2520
สถานที่เกิด	โรงพยาบาลนครนายก
ภูมิลำเนาเดิม	นครนายก
ที่อยู่ปัจจุบัน	71 หมู่ 4 ตำบล ท่าเรือ อำเภอ ปากพลี จังหวัด นครนายก 26130
โทรศัพท์	-
ประวิติการศึกษา	
- ประถมศึกษา	โรงเรียนวัดที่ประดิษฐ์
- มัธยมศึกษาตอนต้น	โรงเรียนปากพลีวิทยาคาร
- ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคนครนายก
- ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคนครนายก
- ปริญญาตรี	สาขาวิชาเทคโนโลยีการวัดคุมทางอุตสาหกรรม ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
คติพจน์	-