ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ การออกแบบตัวเข้ารหัสและถอดรหัส DCT โดยการบวนการ ASIC

DCT Encode and Decode Design Using by ASIC Design Process

ชื่อนักศึกษา 1. นายธนันต์ชัย บรรเทิงจิตร รหัสประจำตัว 43035376

2. นางสาวนวรัตน์ ลิมาภิรักษ์ รหัสประจำตัว 43035613

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** อิเล็กทรอนิกส์และคอมพิวเตอร์

อาจารย์ที่ปรึกษา อาจารย์กิติพงค์ มะโน

อาจารย์ที่ปรึกษาร่วม ผศ.วิสุทธิ์ อธิพรธรรม

คณะกรรมการสอบปริญญานิพนธ์		ลายมือชื่อ
1. อาจารย์กิติพงค์	มะโน	.....
2. อาจารย์ปิยะ	จิตธรรมมาภิรมย์	.....
3. อาจารย์สุชิน	อาจหาญ	.....
4. อาจารย์ไพบูลย์	พวงวงศ์ตระกูล	.....
5. อาจารย์สุระชัย	พิมพ์สาลี	.....

วัน/เดือน/ปีที่สอบ วันเสาร์ที่ 10 พฤศจิกายน พ.ศ. 2544 เวลา 14.00 น.

สถานที่สอบ ห้อง ค.311 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.วิสุทธิ์ อธิพรธรรม)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 29 เดือน พ.ย. พ.ศ. 2544

<BT4402072>

การออกแบบตัวเข้ารหัสและถอดรหัส DCT โดยการบวนการ ASIC

ปริญญานิพนธ์

การออกแบบตัวเข้ารหัสและถอดรหัส DCT โดยกระบวนการ ASIC

DCT Encode and Decode Design using by ASIC Design Process

นายธนันต์ชัย บรรเทิงจิตร

นางสาวนวรัตน์ ลิมาภิรักษ์

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์

ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์

เรื่อง การออกแบบตัวเข้ารหัสและถอดรหัส DCT โดยกระบวนการแบบ ASIC

DCT Encode and Decode Design using by ASIC Design Process

วัตถุประสงค์

1. เพื่อศึกษาหลักการออกแบบวงจร ASIC, ทฤษฎีของ DCT และ IDCT หนึ่งมิติ
2. เพื่อออกแบบวงจร DCT และ IDCT
3. เพื่อสร้างวงจร DCT และ IDCT หนึ่งมิติระดับ Layoutโดยโปรแกรม L-Edit, MATLAB, Leonardo Spectrum, T-Spice
4. เพื่อทดลองเลียนแบบการทำงานของวงจร DCT และ IDCT แต่ละส่วน
5. เพื่อนำ Layout ต้นแบบที่ได้นำไปเจือสารสร้างเป็นชิพวงจรรวมในโอกาสต่อไป

ประโยชน์ที่คาดว่าจะได้รับ

1. มีความรู้เกี่ยวกับการออกแบบวงจรรวม ASIC, ทฤษฎีของ DCT และ IDCT หนึ่งมิติ
2. ได้โครงสร้างของวงจร DCT และ IDCT
3. ได้ Layout ต้นแบบ ของชิพรวม DCT และ IDCT
4. ได้ปรับปรุง และพัฒนาวงจรขนาดใหญ่ให้มีขนาดที่เล็กลง
5. ได้นำ Layout ต้นแบบ เพื่อไปเจือสารได้เป็นชิพวงจรรวมต่อไป

ชื่อหัวข้อ	การออกแบบตัวเข้ารหัสและถอดรหัส DCT โดยกระบวนการ ASIC
นักศึกษา	นายธนันต์ชัย บรรเทิงจิตร
	นางสาวนวรัตน์ ลิมาภิรักษ์
อาจารย์ที่ปรึกษา	อาจารย์กิติพงค์ มะโน
อาจารย์ที่ปรึกษาร่วม	ผศ.วิสุทธิ์ อธิพรธรรม
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชา	อิเล็กทรอนิกส์และคอมพิวเตอร์
ปีการศึกษา	2544

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอแนวทางการออกแบบวงจรรวมเฉพาะงานในระดับ Mask Layout โดยใช้ Tanner tools ซึ่งเป็นซอฟต์แวร์ที่สามารถออกแบบได้ทั้งในระดับ Netlist และในระดับ Mask Layout โดยได้นำงานออกแบบชิพวงจรเข้ารหัสและถอดรหัส DCT แบบหนึ่งมิติ ขนาด 8 บิต ที่ได้จากการออกแบบโดยใช้ภาษาบรรยายพฤติกรรมทางฮาร์ดแวร์(VHDL) มาทำการออกแบบ Mask Layout ซึ่งเริ่มต้นจากการทำ SPR และ BPR Place & Route จากใน EDIF Netlist ของวงจรที่ได้รับมาโดยใช้เทคโนโลยีซีมอสแสตนดาร์ดเซล 0.5 ไมครอน ของโรงงาน Alcatel Microelectronics เพื่อนำ Mask Layout ที่สร้างส่งไปยังโรงงานผลิตวงจรรวม

Thesis Title	DCT Encode and Decode Design using by ASIC Design Process
Students	Mr. Thanunchai Bunthoengjit
	Miss Navarat Limapiruk
Advisor	Mr. Kitipong Mano
Co-Adviser	Asst.Prof. Wisuit Atiporntum
Education Level	Bachelor of Science in Industrial Education
Program in	Electronic and computer
Academic Year	2001

ABSTRACT

This thesis illustrates partially the implementation of ASIC (Application Specific Integrated Circuit) layout design using Tanner tools, which are capable of doing design works at the netlist and mask layout levels for the 1D-DCT encoder and decoder. The thesis shows mask layout design technique. EDIF netlist file are using by SPR (Standard Cell Place and Rout) and BPR (Block Place and Route) to place and route a design. The chip was fabricated using Alcatel Microelectronics' 0.5 micron CMOS technology.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากความกรุณาของอาจารย์ที่ปรึกษาคือ อาจารย์กิติพงค์ มะโน ที่คอยให้คำปรึกษา คำแนะนำ แนวความคิด ความรู้ต่างๆ ตลอดจนแนวทางในการแก้ไขปัญหาต่างๆ ในการจัดทำปริญญานิพนธ์ และขอขอบพระคุณอาจารย์ในภาควิชาครุศาสตร์วิศวกรรมทุกท่านที่ให้ความช่วยเหลือตลอดการทำโครงงานอย่างเต็มที่

ขอขอบคุณห้องสมุดคณะครุศาสตร์อุตสาหกรรม ห้องสมุดคณะวิศวกรรมศาสตร์ ที่ช่วยอำนวยความสะดวกและเอื้อเฟื้อสถานที่ในการค้นคว้าหาข้อมูล สุดท้ายขอขอบพระคุณ บิดามารดาที่เป็นผู้ให้การสนับสนุนด้านการศึกษาและเป็นผู้ให้กำลังใจด้วยดีตลอดมา ตั้งแต่อดีตจนถึงปัจจุบัน

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปริญญานิพนธ์

การออกแบบวงจรและระบบดิจิตอลที่ใช้ IC มาตรฐานมักส่งผลกระทบมากมาย เช่น วงจรที่มีขนาดใหญ่สิ้นเปลืองกำลังงานมาก กระบวนการออกแบบใช้เวลานาน สามารถเลียนแบบและละเมิดลิขสิทธิ์ได้ง่าย ดังนั้นในปัจจุบันผู้ออกแบบจึงนิยมออกแบบวงจรดิจิตอลที่ใช้เฉพาะงานมากว่าการออกแบบด้วยอุปกรณ์มาตรฐาน และแนวโน้มของการออกแบบวงจรรวมเฉพาะงาน (ASIC : Application Specific Integrated Circuit) จะเข้ามามีบทบาทมากขึ้นในการใช้งานของนักออกแบบอุปกรณ์ทางด้านดิจิตอล ซึ่งการออกแบบวงจรรวมเฉพาะงานนี้สามารถแบ่งตามลักษณะการสร้างออกเป็น 2 กลุ่มใหญ่ๆ ได้แก่ Field Programmable เป็นการออกแบบโดยใช้ภาษา VHDL หรือ Verilog แล้วนำไปสังเคราะห์เพื่อสร้างด้วยอุปกรณ์ประเภท Field Programmable เช่น FPGAs CPLD ซึ่งมีข้อจำกัดในการใช้งานบางประการที่ต้องการประสิทธิภาพของวงจร เช่น ความเร็ว จำนวนบิทที่ใช้ในการประมวลผล เป็นต้น และอีกกลุ่ม คือ Mask Programmable ซึ่งการออกแบบลักษณะนี้จะส่งผลให้ประสิทธิภาพของวงจรดีมากยิ่งขึ้น เช่น ความผิดเพี้ยนของสัญญาณน้อยลง ใช้พลังงานในวงจรที่ต่ำ การทำงานของวงจรที่เร็วขึ้นหรือเกิดการหน่วงเวลา (Time Delay) น้อยที่สุดเสมือนกับการทำงานทันกับเวลาจริง (Real Time) สามารถออกแบบวงจรรวมได้มากขึ้นเพื่อตอบสนองความต้องการในการประมวลผลในระดับที่สูงขึ้นหรือการเพิ่มบิทของการประมวลผลให้สูงขึ้น เป็นต้น

ดังนั้นโครงงานนี้จึงให้ความสำคัญ ในการศึกษาการออกแบบวงจรรวมเฉพาะงานในระบบดิจิตอล ด้วยกระบวนการ ASIC โดยการนำวงจรเข้ารหัสและถอดรหัส DCT แบบหนึ่งมิติ มาเป็นกรณีศึกษา ซึ่งประมวลผลสัญญาณขนาด 8 บิท โดยอาศัยการออกแบบบรรยายพฤติกรรมการทำงานของฮาร์ดแวร์ที่แสดงถึงความสัมพันธ์ระหว่างสัญญาณอินพุตและสัญญาณเอาพุตด้วยภาษา VHDL เพื่อนำไปสร้างเป็น Layout โดยใช้ Tanner Tools ซึ่งเป็นซอฟแวร์ที่สามารถออกแบบทั้งในระดับ Netlist โดยการออกแบบ Schematic Diagram จำลองการทำงานในระดับเกต (Gate-Level Simulation) และในระดับวงจร (Circuit-Level Simulation) และในระดับ (Mask Layout) โดยเริ่มจากการทำ Library Standard Cell เพื่อการ Place & Route ของวงจรที่ได้ จากนั้นทำการ Layout จาก Netlist File ที่ได้จากการออกแบบวงจรด้วยใช้ภาษา VHDL โดยนำงานที่ออกแบบตัวเข้ารหัสและถอดรหัส DCT มาจำลองการทำงาน (Simulation) ด้วยโปรแกรม T-Spice ต่อไป

1.2 ขีดความสามารถของโครงงาน

โครงงานนี้มีขีดความสามารถดังนี้

1) มีวงจร DCT และ IDCT ขนาดหนึ่งมิติ

2) สามารถแสดงผลเลียนแบบการทำงานในระดับ VHDLของ DCT และ IDCTได้

3) สามารถแสดงผลเลียนแบบการทำงานในระดับ Layout โดยใช้โปรแกรม T-Spice

4) สามารถรับสัญญาณจากดิจิตอลขนาด 8 บิต เป็นอย่างต่ำ

5) ใช้เทคโนโลยี 0.5ไมครอนบริษัท MITEC

1.3 เนื้อหาโดยสังเขป

เนื้อหาของปริญญานิพนธ์ฉบับนี้ได้กล่าวถึงการออกแบบและสร้างตัวเข้ารหัสและถอดรหัส DCT โดยใช้ภาษา VHDL ใช้ในการออกแบบบรรยายพฤติกรรมทางด้านฮาร์ดแวร์ของวงจรเชิงเลข เพื่อนำ VHDL Code ไปทำกระบวนการสังเคราะห์ให้เป็นระดับเกต และระดับวงจร หลังจากนั้นนำผลที่ได้จากการสังเคราะห์ไปทำกระบวนการ Place & Route และนำไปสร้าง Layout หรือ Mask สำหรับส่งไปเจือสาร ซึ่งการทำโครงงานทั้งหมดนี้จะใช้ในทฤษฎีของการออกแบบวงจรรวมเฉพาะงานและการใช้โปรแกรมของบริษัท Tanner เป็นเครื่องมือที่ใช้ในการออกแบบ

บทที่ 2 ทฤษฎีและหลักการ การเข้ารหัสและถอดรหัส DCT การออกแบบ Top-Down Design โครงสร้างภาษา VHDL ลักษณะการบรรยายด้วยภาษา VHDL การบรรยายเชิงพฤติกรรม การบรรยายเชิงข้อมูล การบรรยายเชิงโครงสร้าง ทฤษฎีและหลักการของการออกแบบวงจร (ASIC : Application Specific Integrated Circuit)

บทที่ 3 การออกแบบ การสร้างและการทำงานของวงจรเข้ารหัสและถอดรหัส DCT ในแต่ละภาค ขั้นตอนการใช้เครื่องมือ โครงสร้าง สำหรับการออกแบบวงจรรวมเฉพาะงาน (ASIC) และการเลียนแบบการทำงานของวงจรเพื่อทดสอบ

บทที่ 4 การทดลองและผลการทดลอง นำ Layout ที่ได้จากในแต่ละส่วนย่อยมาเลียนแบบการทำงาน จากนั้นก็รวม Cell ในแต่ละส่วนที่ได้ไปเลียนแบบการทำงานของวงจรรวมทั้งหมด

บทที่ 5 บทสรุป ปัญหา แนวทางแก้ไข และพัฒนา การสรุปผลและอภิปรายเกี่ยวกับความสามารถ ประสิทธิภาพของตัวเข้ารหัสและถอดรหัส DCT โดยกระบวนการออกแบบวงจรรวมเฉพาะงาน (ASIC) ปัญหาที่เกิดขึ้นระหว่างทำโครงงาน แนวทางการแก้ไขปัญหาที่เกิดขึ้น และข้อเสนอแนะ แนวทางในการพัฒนาปรับปรุงโครงงานให้มีประสิทธิภาพมากยิ่งขึ้น

ภาคผนวก ก รายละเอียดของวงจร

ภาคผนวก ข Layout ของวงจร

ภาคผนวก ค ภาษา VHDL

บทที่ 2
ทฤษฎีและหลักการ

2.1 กล่าวนำ

สำหรับทฤษฎีที่ใช้ในการออกแบบโครงงานนี้ประกอบด้วย ทฤษฎีของการเข้ารหัสและถอดรหัส DCT กระบวนการออกแบบ Top-Down Design ทฤษฎีและหลักการออกแบบวงจรรวมเฉพาะงาน (ASIC) และหลักการใช้เครื่องมือต่างๆ (Tools) ที่เกี่ยวข้องกับการออกแบบทั้งหมด

2.2 ทฤษฎีการเข้ารหัสและถอดรหัส DCT

การเข้ารหัสและถอดรหัส DCT เป็นวิธีการหนึ่งของการเข้ารหัสการแปลง (Transform Coding) ซึ่งจะอธิบายถึงความสัมพันธ์และประโยชน์ ของการเข้ารหัสและถอดรหัส DCT ต่อการบีบอัดข้อมูลดังนี้

2.2.1 การจำแนกประเภทของการบีบอัดข้อมูล

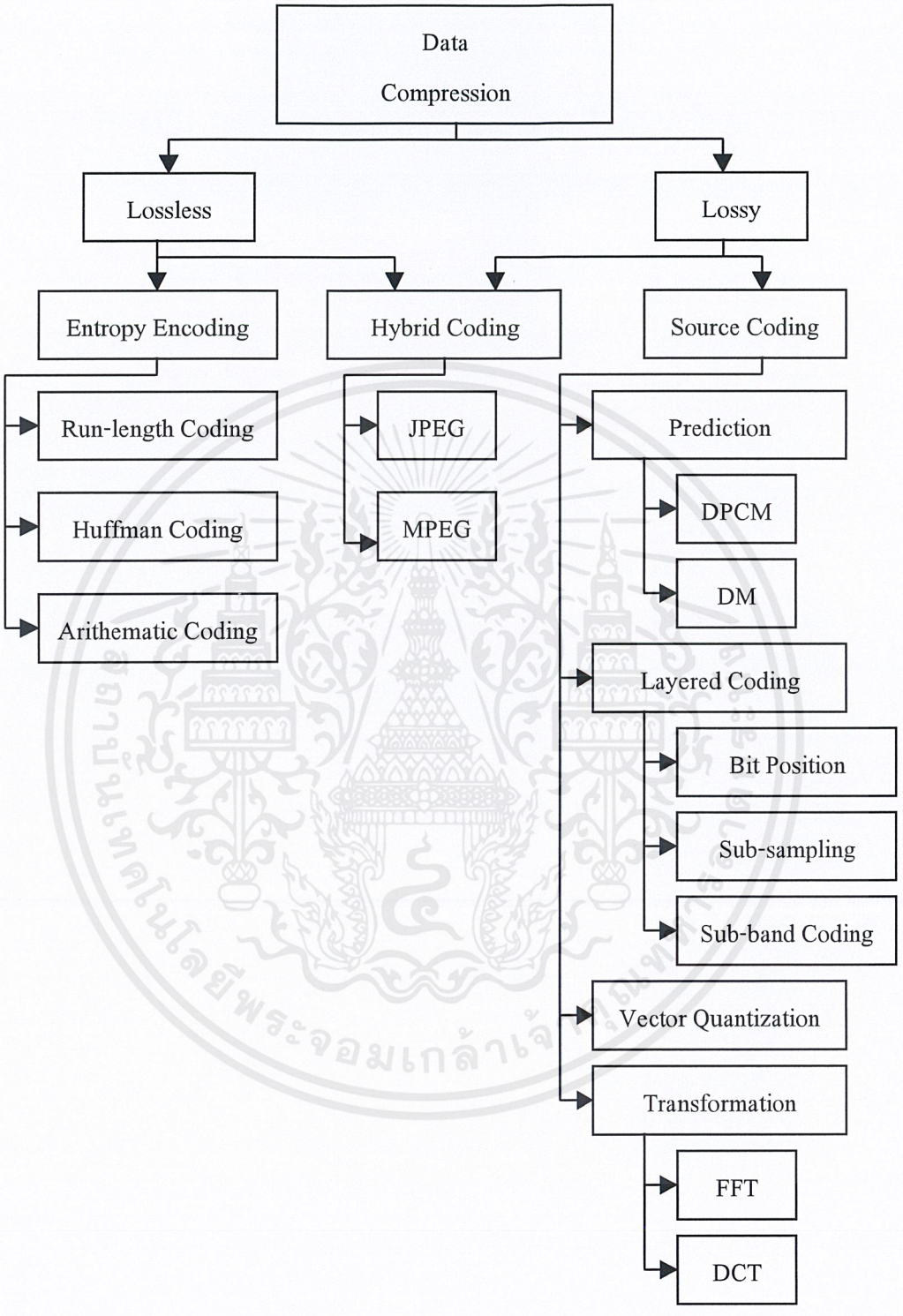
โดยทั่วไปรูปแบบการบีบอัดข้อมูล สามารถแบ่งได้เป็น 2 ประเภท ดังนี้

1) การบีบอัดข้อมูลแบบไม่มีการสูญเสีย (Lossless Compression) เป็นวิธีการบีบอัดข้อมูลเมื่อผ่านขั้นตอนการบีบอัดแล้วผลลัพธ์ของข้อมูล ที่ได้จะเหมือนกับต้นแบบ (Original) ทุกประการ ซึ่งมีอัตราการบีบอัดข้อมูล 3 : 1 นิยมใช้กับข้อมูลที่มีความสำคัญมากและสูญเสียไม่ได้ เช่น ข้อมูลภาพที่ใช้วินิจฉัยทางการแพทย์ เป็นต้น

2) การบีบอัดข้อมูลที่มีการสูญเสีย (Lossy Compression) เป็นวิธีการที่ผลลัพธ์ของข้อมูลที่ได้จากการขยายข้อมูลกลับมีความแตกต่างจากข้อมูลต้นแบบ โดยยอมให้มีการสูญเสียข้อมูลได้บ้าง นิยมใช้ในการประมวลภาพ และเสียงทั่วไปเป็นอย่างมาก เนื่องจากประสาทสัมผัสของคนเราไม่สามารถตรวจจับข้อมูลในส่วนที่มีการสูญเสียได้ ซึ่งมีอัตราการบีบอัดข้อมูลมากว่า 100 : 1

อย่างไรก็ตามคุณภาพของภาพหรือเสียงที่ได้มีความสัมพันธ์กับอัตราการบีบอัดข้อมูล (Compression Ratio) ด้วย นั่นคือข้อมูลที่มีอัตราการบีบอัดที่สูงจะได้ขนาดของข้อมูลที่มีขนาดเล็ก แต่ทำให้คุณภาพของภาพแย่ลงไปด้วย

วิธีการบีบอัดข้อมูลภาพและเสียงมีหลายวิธี แต่ละวิธีมีความเหมาะสมในลักษณะของการนำไปใช้งานที่แตกต่างกัน สามารถจำแนกความแตกต่างของวิธีการบีบอัดข้อมูลได้ดังรูปที่ 2.1



รูปที่ 2.1 การจำแนกการเข้ารหัสและวิธีการบีบอัดข้อมูลแบบต่างๆ

3) การเข้ารหัสเอนโทรปี (Entropy Encoding) การเข้ารหัสแบบนี้จะไม่คำนึงถึงคุณสมบัติเฉพาะของข้อมูลผ่านกระบวนการ โดยสายข้อมูลจะถูกบีบอัดข้อมูลโดยการใช้ลำดับของข้อมูลทางดิจิตอล การเข้ารหัสแบบนี้เป็นการเข้ารหัสแบบไม่มีการสูญเสีย ได้แก่

3.1) การเข้ารหัสแบบรันเลนจ์ (Run-length Coding) เป็นวิธีการเข้ารหัสที่ง่ายที่สุดในการเข้ารหัสข้อมูลภาพที่มีพิกเซล (Pixel) ที่อยู่ติดกันมีการเปลี่ยนแปลงของระดับความเข้มไม่มากและมีจำนวนที่ซ้ำกัน จะทำการเก็บจำนวนของพิกเซลที่ซ้ำกันแทนการเก็บค่าระดับความเข้มของพิกเซลเหล่านั้น

3.2) การเข้ารหัสฮัฟแมน (Huffman Coding) หลักการที่สำคัญคือข้อมูลที่มีความน่าจะเป็นสูงกว่าหรือข้อมูลที่มีความถี่สูงกว่าจะถูกเข้ารหัสด้วยจำนวนบิทที่น้อยกว่า การเข้ารหัสฮัฟแมนสามารถอธิบายได้ดังนี้

3.2.1) การนับความถี่ของข้อมูลทุกตัวและเรียงลำดับตามความถี่จากมากไปหาน้อย

3.2.2) นำค่าความถี่ที่ค่าต่ำสุด 2 ค่ามารวมกันเพื่อสร้างโหนดใหม่ นำค่าความถี่ใหม่ที่ได้มาเรียงลำดับกับค่าความถี่เดิมที่เหลือ

3.2.3) กำหนดค่าไบนารีให้โหนดทุกๆ คู่ซึ่งถูกกำหนดให้เป็นเส้นทางของโหนดใหม่ โดยที่โหนดหนึ่งมีค่าเป็น 0 ส่วนอีกโหนดหนึ่งมีค่าเป็น 1

3.2.4) กระทำซ้ำตามขั้นตอนข้อที่ 3.2.2) และ 3.2.3) จนกระทั่งได้ความถี่เท่ากันกับจำนวนพิกเซล (Picxel) ทั้งหมดของภาพ

3.2.5) การเข้ารหัสและถอดรหัสสามารถทำได้โดยใช้การเปิดตาราง (Look up table) และรหัสที่ได้สามารถทำการถอดรหัสได้ทันทีโดยไม่ต้องอ้างถึงข้อมูลอื่น

3.3) การเข้ารหัสเลขคณิต (Arithematic Coding) การเข้ารหัสเลขคณิตนั้น ข้อมูลจะถูกแทนที่ด้วยช่วงระยะห่างของจำนวนจริงระหว่าง 0 และ 1 โดยข้อมูลแต่ละตัวจะลดขนาดของระยะห่างตามความน่าจะเป็นของข้อมูลนั้น ข้อมูลที่มีความน่าจะเป็นสูงจะทำให้ระยะห่างถูกลดลงได้น้อยกว่าข้อมูลที่มีความน่าจะเป็นต่ำ ดังนั้นข้อมูลที่มีความน่าจะเป็นสูงจะใช้จำนวนบิทในการเข้ารหัสน้อยกว่า

4) การเข้ารหัสแบบไฮบริด (Hybrid Coding) เป็นการนำหลักการที่ได้จากวิธีการบีบอัดข้อมูลทั้งแบบที่มีการสูญเสียและไม่มีการสูญเสียมาผสมเข้าด้วยกัน เช่น การบีบอัดข้อมูลภาพนิ่ง (JPEG) โดยอาศัยหลักการพื้นฐานของการแปลงแบบดีสครีตโคไซน์ (Discrete Cosine) ร่วมกับการมอดูเลตรหัสพัลส์เชิงอนุพันธ์ เป็นต้น

5) การเข้ารหัสซอร์ส (Source Coding) เมื่อพิจารณาถึงความสัมพันธ์ของข้อมูลหรือองค์ประกอบของข้อมูลเริ่มต้นเป็นหลัก การเข้ารหัสนี้จะให้อัตราการบีบอัดข้อมูลที่สูงกว่าการเข้ารหัสแบบเอนโทรปี (Entropy Encoding) ได้แก่

5.1) การเข้ารหัสทำนาย (Prediction Coding) เป็นการใช้เทคนิคการทำนายข้อมูลที่เกิดขึ้นในปัจจุบัน โดยอาศัยข้อมูลจากอดีตที่ผ่านมา แล้วนำมาเปรียบเทียบกับค่าข้อมูลเริ่มต้นเพื่อหาค่าผลต่าง เช่น การมอดูเลตรหัสพัลส์เชิงอนุพันธ์ (DPCM : Differential Pulse Code Modulation) และการมอดูเลตข้อมูล (DM : Data Modulation)

5.2) การเข้ารหัสเลเยอร์ (Layer Coding)

5.3) การเข้ารหัสด้วยการควอนไตซ์แบบเวกเตอร์ (Vector Quantization)

5.4) การเข้ารหัสการแปลง (Transform Coding) เป็นการใช้เทคนิคการแปลงรูปจากโดเมน (Domain) หนึ่งไปอีกโดเมนหนึ่ง เช่น การแปลงฟูริเยร์ (FFT) การแปลงดีสครีตโคไซน์ (DCT) เป็นต้น

2.2.2 Discrete Cosine Transform Coding

Discrete Cosine Transform Coding เป็นวิธีการที่ใช้ในการลดข้อมูลภาพที่ได้รับความนิยมอย่างแพร่หลายทั้งนี้เพราะค่าของสัมประสิทธิ์ในโดเมนความถี่ (Frequency Domain) ที่ได้จะไม่มีส่วนของค่าจินตภาพ (Imaginary-part) แต่จะเป็นส่วนของค่าจริง (Real-part) เท่านั้นอีกทั้งยังสามารถคำนวณได้อย่างรวดเร็ว (Fast-algorithms) โดยกระบวนการทำงานของ DCT จะมีความสำคัญอยู่ที่การแบ่งข้อมูลที่เข้ามาออกเป็นบล็อคย่อยๆ และทำการแปลงข้อมูลซึ่งอยู่ในโดเมนเวลา (Time Domain) ให้เปลี่ยนไปอยู่ในโดเมนความถี่ (Frequency Domain) โดยสมการที่ (2.1) ซึ่งเป็นสมการที่ใช้ในการเข้ารหัส DCT โดยหาค่าสัมประสิทธิ์ออกมาแล้วนำข้อมูลที่เข้ามาไปผ่านกระบวนการทางคณิตศาสตร์จนได้ค่าข้อมูลใหม่ซึ่งจะอยู่ในรูปของเมตริกซ์ออกมา เพื่อความสะดวกในการทำขั้นตอนต่อไปในกระบวนการบีบอัดข้อมูล เช่น การควอนไตซ์ค่าสัมประสิทธิ์ การเข้ารหัสเอนโทรปี เป็นต้น

การแปลงข้อมูลเสียงแบบเมตริกซ์โดยใช้สัมประสิทธิ์ค่าคงที่ของ DCT ดังสมการที่ (2.2)

$$g = \mathrm{DCT(f)} \tag{2.1}$$

$$\mathrm{g} = \mathrm{AfA}^{\mathrm{T}} \tag{2.2}$$

เมื่อ g คือ สัมประสิทธิ์การเปลี่ยนแปลงของ DCT

A คือ เมตริกซ์ของสัมประสิทธิ์ DCT หรือเมตริกซ์ $X(k)$

A^{T} คือ Transpose ของเมตริกซ์ A

f คือ ข้อมูลอินพุต

จากสมการที่ (2.2) สามารถหาสัมประสิทธิ์ค่าคงที่ของ DCT ได้ดังสมการที่ (2.3)

$$X(k)=\frac{1}{2}C(k)\sum_{i=0}^{N-1}X(i)\cos\left[\frac{(2i+1)k\pi}{2N}\right] \quad ; \quad k=0,1,2,...,N-1 \qquad \textbf{(2.3)}$$

เมื่อ

$$\pi = 180$$

$$C(k)=\begin{cases}1/\sqrt{2} & ; \quad k=0\\ 1 & ; \quad k=0,1,2,...,N-1\end{cases}$$

2.2.3 Inverse Discrete Cosine Transform Coding

เป็นวิธีการที่ใช้การแปลงกลับหรือการถอดรหัสสัญญาณ DCT จากข้อมูลที่อยู่ในรูปโดเมนความถี่แปลงกลับมาอยู่ในรูปโดเมนเวลาโดยสมการที่ (2.2) ซึ่งหลักการทำงานในกระบวนการถอดรหัสสัญญาณนี้จะตรงข้ามกับการเข้ารหัส

การแปลงกลับข้อมูลแบบเมตริกซ์

$$\mathrm{h}=\mathrm{IDCT(g)} \qquad \textbf{(2.4)}$$

$$\mathrm{h}=\mathrm{A}^{\mathrm{T}}\mathrm{gA} \qquad \textbf{(2.5)}$$

เมื่อ g คือ สัมประสิทธิ์การเปลี่ยนแปลงของ DCT

A คือ เมตริกซ์ของสัมประสิทธิ์ IDCT หรือเมตริกซ์ $X(i)$

A^{T} คือ Transpose ของเมตริกซ์ A

h คือ สัมประสิทธ์การแปลงกลับของ DCT

จากสมการที่ (2.5) สามารถหาสัมประสิทธิ์ค่าคงที่ของ IDCT ได้ดังสมการที่ (2.6)

$$X(i)=\frac{1}{2}\sum_{k=0}^{N-1}C(k)X(k)\cos\left[\frac{(2i+1)k\pi}{2N}\right] \quad ; \quad i=0,1,2,...,N-1 \qquad \textbf{(2.6)}$$

เมื่อ

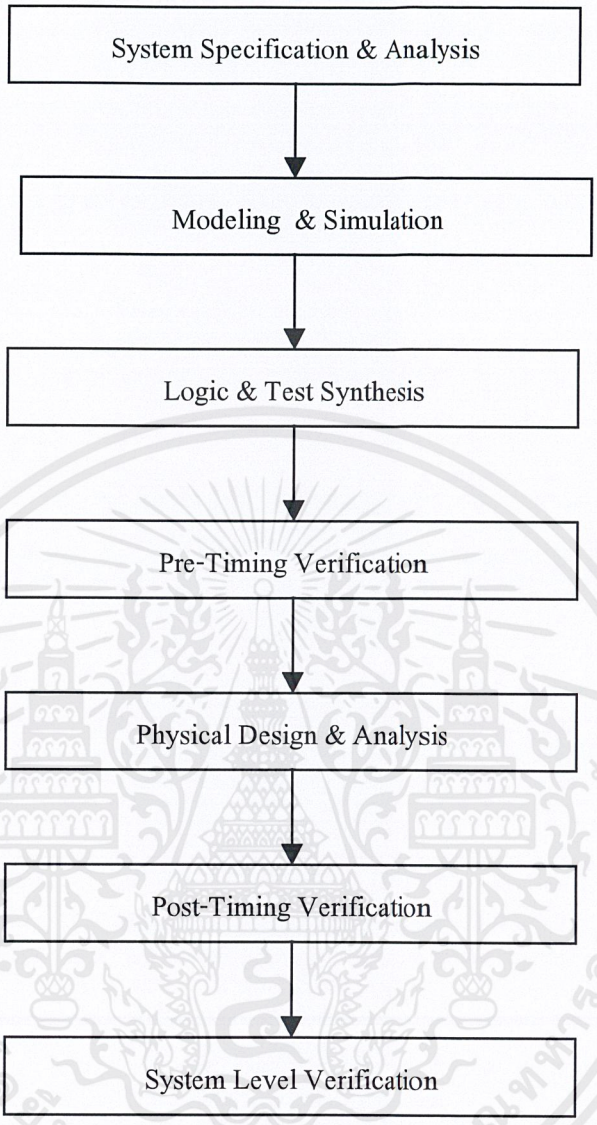
$$\pi = 180$$

$$C(k)=\begin{cases}1/\sqrt{2} & ; \quad k=0\\ 1 & ; \quad k=0,1,2,...,N-1\end{cases}$$

2.3 Top-Down Design

การนำภาษา VHDL มาใช้กำหนดและบรรยายพฤติกรรมฟังก์ชั่นการทำงานของฮาร์ดแวร์ในระบบดิจิตอลนั้นคือความยืดหยุ่นของภาษาซึ่งสามารถจำลองการทำงานจากหลักการของรูปแบบ (Simulate Conceptual Designs) แต่ในขณะเดียวกันนั้นสามารถจำลองการทำงานของฮาร์ดแวร์ที่ให้ลายละเอียดเกี่ยวกับเวลาอย่างถูกต้อง (Timing Based Simulation Levels) ความสามารถดังกล่าวนี้จึงช่วยให้การออกแบบสามารถที่จะเขียนรูปแบบบรรยายจากระดับบนสุดของวงจรที่อยู่ในรูปสังเขป (High Level of Abstraction) ลงสู่รายละเอียดในระดับล่างของวงจรได้ เช่น Gate Level เป็นต้น ในเวลานั้นวงการอุตสาหกรรม ไมโครอิเล็กทรอนิกส์ ตลอดจนสถาบันวิจัยและศึกษา กำลังพัฒนาภาษาที่ใช้สำหรับการสังเคราะห์วงจรแบบอัตโนมัติ เพื่อลดเวลาในการพัฒนาวงจรลง ภาษา VHDL จึงถูกนำเข้ามาพิจารณาในโครงการนี้ด้วย โดยเพิ่มขีดความสามารถของภาษาขึ้น นอกเหนือจากภาษาที่ใช้สำหรับสังเคราะห์วงจร (Synthesis Language)

การเริ่มต้นด้วยวิธีการเขียนรูปแบบจากแนวความคิดอย่างสังเขป พร้อมทั้งการจำลองการทำงานของรูปที่เขียนขึ้นเพื่อตรวจสอบความถูกต้อง ประกอบกับการกลั่นกรองเพื่อเพิ่มเติมรายละเอียดลงสู่ระบบดิจิตอลที่สมบูรณ์ในรูปของวงจรไฟฟ้า ซึ่งในแต่ละขั้นนั้นเป็นขบวนการของ Top-Down Design โดยที่ผู้ที่ออกแบบเริ่มต้นด้วยการเขียนรูปแบบในระดับวงจรในระดับบน (Test Environment) ได้ตั้งแต่ระยะแรกๆ ของการออกแบบ เพื่อใช้สำหรับการตรวจสอบความถูกต้องของแนวความคิดสำหรับสิ่งที่ต้องการจริงหรือ Specification ของงาน ดังนั้น จึงเป็นไปได้ยากที่ในระดับล่างลงมาโดยที่เพิ่มรายละเอียดของรูปแบบให้ได้มากขึ้นตามลำดับจะเกิดข้อพิดพลาด หรือเบี่ยงเบนไปจากจุดประสงค์เดิม เพราะในแต่ละขั้นตอนย่อยจะมีการสร้างสภาพแวดล้อมเพื่อตรวจสอบความใหม่ โดยอ้างอิงจากระดับที่สูงกว่าขึ้นไปเสมอจากรูปที่ 2.2 แสดงให้ขั้นตอนของการออกแบบในลักษณะของ Top-Down Design ทั้งนี้ในการปฏิบัติอาจจะมีข้อแตกต่างไปจากนี้บ้างเล็กน้อยก็เนื่องจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้หลายๆ เทคโนโลยี เช่น Programmable Logic Devices อันได้แก่ PLA, FPGAs หรือ CPLD เป็นต้น นอกจากนั้นยังมี Semi-Custom IC (Gate Array, Standard Cell) และ Full Custom IC



รูปที่ 2.2 ขั้นตอนของ Top-Down Design

2.3.1 กระบวนการออกแบบโดยวิธีใช้ 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 การเขียนรูปแบบของระบบ ที่ต้องการจะออกแบบ โดยภาษา VHDL จากแนวความคิดอย่างสังเขปที่ได้ สำหรับบรรยายพฤติกรรมการทำงานพร้อมทั้งจำลองการทำงานเพื่อเปรียบเทียบและทดสอบความถูกต้องกับข้อกำหนด

3) Logic and Test Synthesis หลังจากที่ได้หลักการขั้นต้นพร้อมทั้งแนวความคิดที่ผ่านการตรวจสอบแล้ว หลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมาเป็นระดับขั้นตอนที่เหมือนกัน คือ Modeling and Simulation จนกระทั่งอยู่ในระดับที่สามารถจะนำไปผลิตเป็นวงจร หรือสังเคราะห์ (Synthesis) ในขั้นตอนนี้เอง เทคโนโลยีที่จะมารองรับวงจรออกแบบจะถูกกำหนดขึ้นและระบบช่วยการออกแบบสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้น ให้อยู่ในรูปของวงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้น หรือไม่ก็อยู่ในรูปของ Netlist ที่สามารถนำไปผลิตลงบนอุปกรณ์อื่นได้ และนอกจากนั้นการผลิตบางเทคโนโลยี อาทิเช่น Gate Aray หรือ Standard Cell และ Full Custom IC อาจจะมีความจำเป็นที่ต้องสร้างโครงสร้างของวงจรใหม่หลังจากที่สังเคราะห์ครั้งแรกแล้ว เพื่อความสะดวกต่อการตรวจสอบการทำงานหลังจากที่ผลิตเป็นวงจรต้นแบบแล้ว หรือที่เรียกว่า "Design For Test" (DFT) พร้อมทั้งข้อมูลในการตรวจสอบจะถูกกำหนดในขั้นตอนนี้

4) Pre-Timing Verification หลังจากสังเคราะห์วงจรให้อยู่ในรูป Gate-Level หรือ Netlish แล้วข้อมูลที่ได้จากผู้ผลิตอุปกรณ์วงจรนั้น นอกจากจะเป็นข้อมูลสำหรับจำลองการทำงานในเรื่องของความถูกต้องของฟังก์ชัน (Functional Simulation) แล้ว ยังมีข้อมูลที่เกี่ยวกับเวลาด้วยซึ่งเป็นความจริงที่ว่าอุปกรณ์อิเล็กทรอนิกส์ทุกชิ้นจะมี Propagation Delay เสมอ ถึงแม้ว่าจะเป็นเวลาที่น้อยมากระดับ nanosecond (10^{-9}) แต่ถ้าภายในวงจรหนึ่งจะประกอบด้วยเกตของฟังก์ชันต่างๆ จำนวน 10,000 เกทขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้น จนอาจจะทำให้การทำงานของวงจรรวมทั้งหมดผิดไป หรือไม่สามารถทำงานในย่านความถี่สัญญาณนาฬิกาที่สูง

5) Physical Design and Analysis คือขั้นตอนของการผลิตเป็นวงจรจริง (Technology and Device Mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจจะอยู่ในรูปแบบของแผงวงจรไฟฟ้าที่ประกอบด้วยอุปกรณ์หลายๆ ชิ้น หรืออยู่มนรูปแบบของวงจรรวมเฉพาะงาน (ASIC)

6) Post-Timing Verification หลังจากที่ได้วงจรจริงมาเรียบร้อยแล้ว ยังต้องมีความจำเป็นที่ต้องตรวจสอบการทำงานที่คำนึงถึงเวลาด้วย เพื่อความถูกต้องของวงจรครั้งสุดท้ายก่อนที่จะนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เพราะในขั้นตอนนี้วงจรที่จะออกแบบ จะประกอบด้วย Input และ Output Pad ซึ่งเป็นจะเป็นจุดต่อสำหรับรับและส่งสัญญาณกับภายนอก

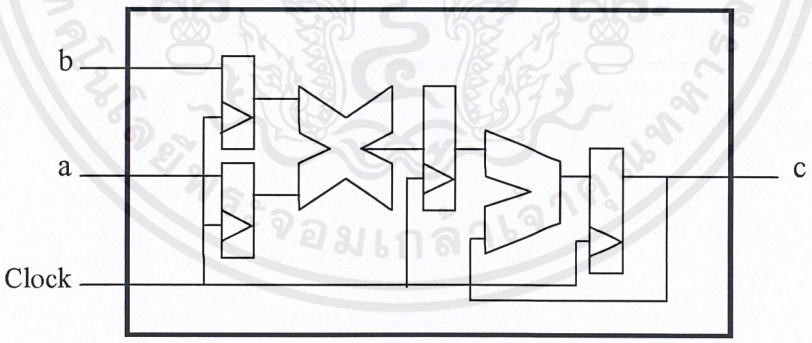
7) System Level Verification หลังจากนำวงจรที่ออกแบบรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอลแล้วนั้น จะต้องทดสอบการทำงานรวมทั้งระบบร่วมกับอุปกรณ์อื่นอีกครั้ง ซึ่งเป็นการควบคุมคุณภาพของผลิตภัณฑ์

ในตัวอย่างของ Multiply Accumulate Algorithm แสดงให้เห็นกระบวนการออกแบบในลักษณะของ Top-Down Design ได้โดยการใช้ Multiply Accumulate Function จุดประสงค์แรกก็คือการเขียนรูปแบบของฟังก์ชันในระดับบนสุดด้วยภาษา VHDL และจำลองการทำงานของรูปแบบที่ได้เพื่อตรวจสอบความถูกตอ้งซึ่งฟังก์ชันนี้อาจจะเป็น Discrete Fourier Transform (DFT) ก็ได้ หลัง

จากที่ฟังก์ชันที่ต้องการแล้วมีชื่อว่า “ multiply_accum ” สามารถนำไปใช้ได้ในรูปของ Function Callคือ

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

ถ้าฟังก์ชันเป็นที่พอใจแล้วในขั้นต่อไป จะเป็นการเริ่มของการเปลี่ยนแปลงแนวความคิดไปสู่การสร้างวงจรจริง กระบวนการดังกล่าวเกิดขึ้นโดยการเพิ่มรายละเอียดให้กับแนวความคิดเดิม รูปที่ 2.3 แสดงให้เห็นวิธีการบรรยายแบบ Dataflow (Dataflow Description) ของ Multiply Accumulate Function

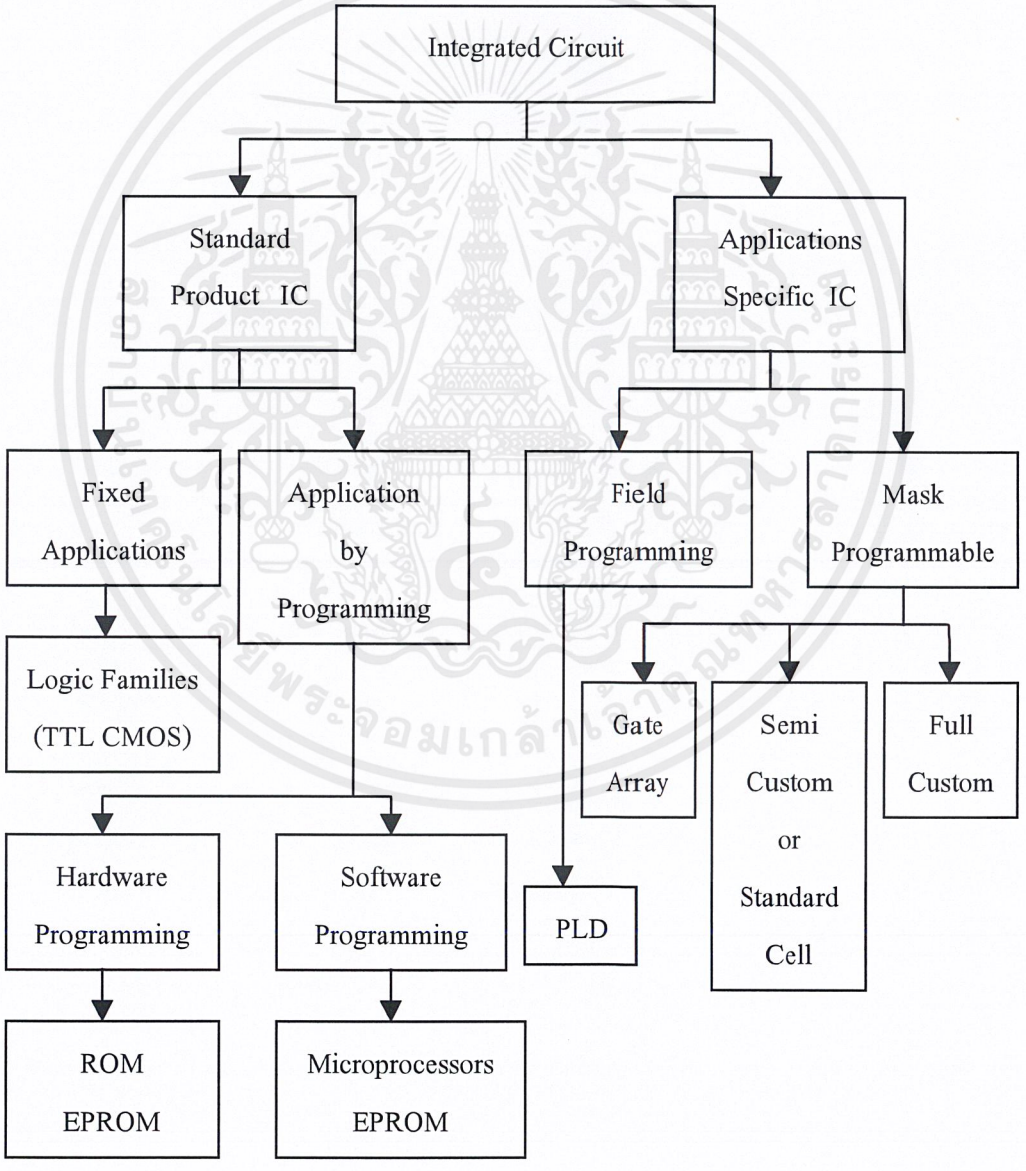


รูปที่ 2.3 Dataflow Description ของ Multiply Accumulation Function

จาก Dataflow Format ที่ได้นี้ กระบวนการต่อไปคือการสร้างวงจรให้อยู่ในรูปของอุปกรณ์พื้นฐานหรือ Gate-Level Implementation ซึ่งวิธีนี้อาจใช้วิธีการสังเคราะห์อัตโนมัติ ผลลัพธ์ที่ได้จากการสังเคราะห์จะเป็นการแสดงภาพวงจรด้วยเกตของฟังก์ชันต่างๆ หรือในรูปของ Netlish และจะเป็นการกำหนดเทคโนโลยีจำเพาะที่จะนำไปผลิตภายหลัง

2.4 หลักการออกแบบวงจรรวมเฉพาะงาน (ASIC : Application Specific Integrated Circuit)

นับตั้งแต่กลางคริสศตวรรษที่ 20 ได้มีการพัฒนาวงจรรวมขนาดเล็กในระดับ Small Scale Integrated (SSI) จนกระทั่งถึงวงจรรวมขนาดใหญ่มากระดับ Very Large Scale Integrated (VLSI) ในปัจจุบันได้มีการแบ่งวิธีการออกแบบและการใช้งานของวงจรรวมออกเป็น 2 แบบใหญ่ๆ คือ Standard Product Integrated Circuit (SPIC) และ Application Specific Integrated Circuit (ASIC) ดังรูปที่ 2.4

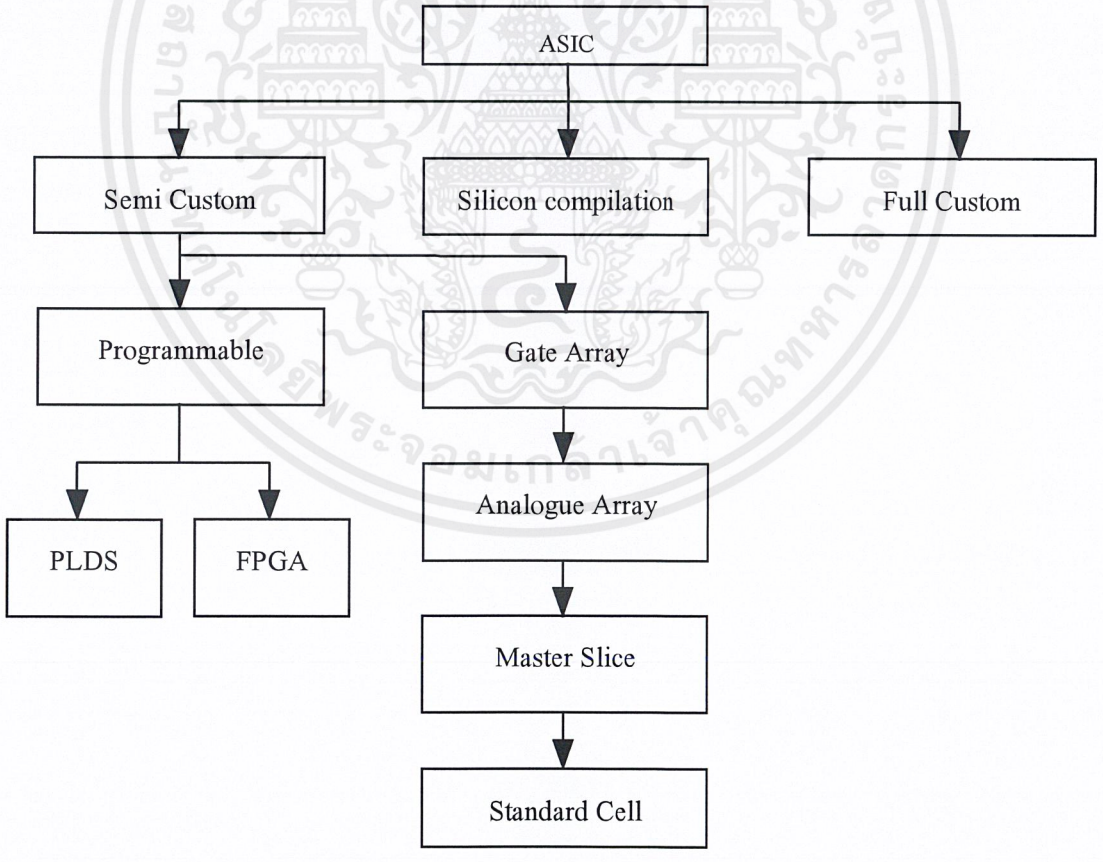


รูปที่ 2.4 วงจรรวมประเภทต่างๆ

2.4.1 หลักการออกแบบวงจรรวมเฉพาะงาน ASIC ชนิดต่างๆ

การใช้งานวงจรรวม ASIC ในเชิงพาณิชย์ จำเป็นต้องใช้วงจรรวม ASIC แบบ Mask Programmable เนื่องจากต้นทุนต่อ 1 ตัวต่ำกว่าวงจรรวมแบบ Standard Products ในกรณีที่มีปริมาณการผลิตสูงนับหมื่นตัวขึ้นไป ยกตัวอย่างเช่น วงจรรวม EPLD ตัวหนึ่งอาจมีราคาสูงถึงหนึ่งพันบาท ในขณะที่ถ้าผลิตวงจรรวมที่มีคุณสมบัติเหมือนกันทุกประการโดยใช้ Mask Programmable จึงมีบทบาทสำคัญในการผลิตสินค้าในเชิงพาณิชย์ในปัจจุบัน

วงจรรวม Mask Programmable ASIC นี้ หลังจากผู้ใช้ออกแบบและตรวจสอบการทำงานจนเป็นที่พอใจแล้ว ต้องส่งไปยังผู้ผลิตทำการเจือสาร ไม่สามารถโปรแกรมได้ด้วยตนเองเหมือนวงจรรวมแบบ Field Programmable ช่วงเวลาการผลิตออกมาใช้งานจึงใช้เวลานับเดือนและมีค่าใช้จ่ายเบื้องต้นในการเจือสารสูง วงจรรวมแบบ Mask Programmable ASIC ในปัจจุบันมีการออกแบบเป็น 3 กลุ่ม คือ Full Custom, Silicon Compilation และ Semi-Custom ในทั้ง 3 กลุ่มนี้มีเพียงกลุ่มของ Semi Custom เท่านั้นที่ยังมีการแบ่งแยกออกไปได้อีกดังแสดงในแผนผังรูปที่ 2.5



รูปที่ 2.5 แผนผังการแบ่งวงจรรวม

2.4.2 วงจรรวม Full Custom

วงจรรวม Full Custom นี้ผู้ใช้เป็นผู้ออกแบบเองทั้งหมด ตั้งแต่ระดับวงจรเชิงเลขจนกระทั่งถึงระดับล่างสุดจนทำเป็น Layout ออกมาแล้วจึงส่งข้อมูลการออกแบบไปยังผู้ผลิตเพื่อนำไปเจือสาร จะจัดอยู่ในรูปของไฟล์จุด GDS II หรือ CIF ซึ่งรูปแบบการเจือสารมี 3 รูปแบบดังนี้

1) Multi Project Wafer (MPW) การเจือวงจรรวม Full Custom แบบนี้แผ่น Wafer จะแบ่งออกเป็นส่วนๆ ที่เรียกว่า die ซึ่งแต่ละชนิดจะเจือสารต่างกัน

2) Multi Project Chip (MPC) เป็นการเจือสารออกแบบวงจรลงบน Die ชุดเดียวกันโดยที่ Die ทุกชุดบน Wafer เดียวกันจะมีลักษณะเหมือนกันการจัดวางวงจรที่ออกแบบลงบน Die เน้นการใช้พื้นที่ของซิลิคอนให้มีประโยชน์สูงสุดเป็นสำคัญ

3) Multi Project Reticle (MPR) เป็นการเจือสารในลักษณะการผสมระหว่าง Full Custom กับ Standard Cell กล่าวคือ Die จะแบ่งออกเป็นส่วนๆ อย่างสม่ำเสมอแต่ละส่วนบรรจุที่ออกแบบได้ประมาณ 8 วงจรและ Die ทุกชุดในแผ่น Wafer จะมีลักษณะเหมือนกัน

2.4.3 วงจรรวมประเภท Semi Custom

วงจรรวมในประเภทนี้ผู้ผลิตได้ทำการรวบรวมวงจรพื้นฐานต่างๆ ไว้ในวงจรรวมเพียงตัวเดียวกัน ซึ่งผู้ใช้จะเป็นผู้ที่ทำการเชื่อมโยงวงจรพื้นฐานที่มีอยู่ให้เป็นวงจรรวมที่มีขนาดใหญ่มากที่สามารถทำงานได้ตามความต้องการโดยที่จะใช้เครื่องคอมพิวเตอร์เป็นเครื่องมือที่ช่วยในการออกแบบซึ่งในวงจรรวม ASIC แบบนี้ สามารถแบ่งออกเป็น 3 กลุ่ม ได้แก่

1) Programmable Logic Devices (PLD) วงจรรวม ASIC ในกลุ่มนี้ผู้ผลิตได้บรรจุวงจรพื้นฐานประเภทเกต (Gate) แบบต่างๆ เช่น AND, OR, INVERTER ในวงจรรวมเพียงตัวเดียว ผู้ใช้สามารถออกแบบเชื่อมโยงวงจรประเภทนี้เข้าเป็นวงจรเชิงเลขขนาดใหญ่ และเมื่อออกแบบเสร็จแล้วก็สามารถผลิตขึ้นใช้เองได้โดยไม่ต้องสั่งให้ผู้ผลิตทำการเจือสารแต่อย่างใด ระยะเวลานับจากการออกแบบจนกระทั่งผลิตขึ้นใช้งานจึงสิ้นสุดเมื่อเทียบกับวงจรรวม ASIC ในกลุ่มอื่นๆ จึงเหมาะสมที่จะพัฒนาต้นแบบหรือผลิตสินค้าที่มีจำนวนขายสินค้าเป็นสิบเป็นร้อยตัวเท่านั้น

2) Gate Array วงจรรวม ASIC ประเภทนี้ผู้ผลิตจะบรรจุวงจรพื้นฐานเกตแบบต่างๆ ไว้ในชิปตัวเดียวและวงจรเกตบางส่วนจะต่อสายภายในไว้เป็นกลุ่ม วงจรเชิงเลขที่ใช้งานกันทั่วไปได้แก่ Latch, Decoder, Register เป็นต้น กลุ่มของวงจรเชิงเลขที่รวบรวมบรรจุไว้ในวงจร Gate Array นี้เรียกว่า Macrocell หรือ Logic-cell ที่ผู้ใช้มีหน้าที่ออกแบบต่อสายเชื่อมโยงระหว่าง Macrocell แต่ละตัวเข้าด้วยกันให้ผู้ผลิตทำการเจือสารต่อไป การออกแบบวงจร Gate Array นี้ผู้ใช้ต้องมีข้อมูลเกี่ยวกับวงจรรวมและ Macrocell ที่บรรจุภายในเสียก่อน ซึ่งตามปกติผู้ผลิตจะให้ข้อมูลมาโดยไม่คิดค่าใช้จ่าย แต่เนื่องจากวงจรรวมประเภทนี้ใช้ต้นทุนสูง ดังนั้นในการผลิตจึงเป็นไปในเชิงพาณิชย์

3) Standard Cell ปัญหาการใช้พื้นที่ Silicon อย่างมากมาย สำหรับการเชื่อมโยงของวงจร Gate Array ทำให้เกิดขีดจำกัดในความซับซ้อนของวงจรที่จะใช้ ประกอบกับผู้ใช้จำนวนมากต้องรวบรวมวงจรรวมมาตราฐาน อาทิ วงจรรวมตระกูล 7400 วงจรรวมตระกูล 4000 จำนวนหลายตัวเข้าเป็นวงจรรวม ASIC เพียงตัวเดียวเท่านั้น ทำให้เกิดวงจรรวม ASIC แบบ Standard Cell ขึ้นวงจรรวม Standard Cell นี้ผู้ใช้ เป็นผู้เลือกวงจรทำหน้าที่ต่างๆ เช่น เกต ฟลิปฟลอป ตัวนับ ตัวเลื่อน หน่วยความจำ หรือแม้กระทั่งไมโครโปรเซสเซอร์จากแฟ้มข้อมูล (Library) ของผู้ผลิต ซึ่งอาจจะเป็นแฟ้มข้อมูลคอมพิวเตอร์ หรือเหมาะสมสำหรับวงจรเชิงเลขที่มีความซับซ้อน การผลิตวงจรรวม Standard Cell จะมีต้นทุนสูงกว่าวงจรรวม Gate Array และใช้เวลาในการออกแบบและเจือสารยาวกว่าสองถึงสามเท่าตัว วงจรรวม Standard Cell จึงเหมาะสมกับการใช้งานในเชิงพาณิชย์ที่มีปริมาณการผลิตนับหมื่นตัวขึ้นไป บทบาทของวงจรรวม Programmable Gate Array สามารถบรรจุจำนวนเกตได้มากขึ้นจนกระทั่งทัดเทียมกับมวลรวม Gate Array ในปัจจุบัน รูปที่ 2.9 แสดงตัวอย่างจากแฟ้มข้อมูลของผู้ผลิตรายหนึ่ง ผู้ใช้จะนำข้อมูล Standard Cell มาออกแบบวงจรที่ตนต้องการ แล้วส่งผลการออกแบบให้ผู้ผลิตเจือสาร ซึ่งแผ่นซิลิคอนที่เจือสารแล้วจะประกอบด้วยพื้นที่ของ Cell ต่างๆ

2.4.4 Design Flow

ขั้นตอนในการออกแบบและสร้างวงจรรวม ASIC ในประเภท ที่เป็น Semi Custom นั้นจะมีการนำคอมพิวเตอร์เป็นเครื่องมือ มาช่วยในการออกแบบ ซึ่งในการออกแบบวงจรรวม ASIC ในประเภท Semi Custom ที่ใช้คอมพิวเตอร์มาช่วยจะประกอบด้วยขั้นตอนที่สำคัญดังแสดงให้เห็นในรูปที่ 2.10 ซึ่งมีรายละเอียดดังนี้

1) Design Entry ผู้ออกแบบจะเริ่มต้นด้วยการกำหนดรายละเอียด หน้าที่การทำงานของวงจรที่ต้องการ (Function Specification) แล้วจึงแปลงรายละเอียดนี้ให้อยู่ในลักษณะที่คอมพิวเตอร์เข้าใจซึ่งจะใช้ ภาษาบรรยายฮาร์ดแวร์ (Hardware Description Language : HDL)

2) Logic Synthesis จะใช้ภาษา VHDL หรือ Verilog หรืออาจจะใช้เป็น Schematic มาทำการสังเคราะห์ ออกมาให้ได้เป็นไฟล์ Netlist โดยใช้เครื่องมือในการสังเคราะห์

3) System Partitioning เป็นการตรวจสอบขนาดของวงจรรวมที่ได้มาการ Netlist

4) Prelayout Simulation เป็นการนำไฟล์ Nelist ที่ได้จากการแปลงของ VHDL มาเปรียบเทียบความถูกต้องของวงจรที่ได้ออกแบบโดยจำลองการทำงาน

5) Floorplannig เป็นการร่างการวาง Block ต่างๆ ที่ทำเป็นชิปให้มีความเหมาะสม

6) Placement & Routing เป็นขั้นตอนการจัดเรียงพื้นที่ต่างๆ ของ Cell และทำการเชื่อมต่อจุดต่างๆ

2.5 ภาษา VHDL

วิวัฒนาการของภาษา VHDL นับเริ่มต้นประมาณปี ค.ศ. 1981 โดยที่กระทรวงกลาโหมของสหรัฐอเมริกา (The United State Department of Defense : DoD) ได้ตระหนักถึงสภาพการของอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการทหารได้รับการพัฒนามาเมื่อประมาณ 20 ปีก่อนไม่เป็นที่ยอมรับในปัจจุบัน เพราะเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์ได้รับการพัฒนาไปอย่างรวดเร็ว แต่เดิมวงจรอิเล็กทรอนิกส์ถูกสร้างขึ้นมาจากชิ้นส่วนอุปกรณ์อิเล็กทรอนิกส์จำนวนหลายๆ ชิ้นนำมาประกอบกันอยู่บนแผงวงจรไฟฟ้า (Printed) ดังนั้น DoD ได้เริ่มต้นโครงการโดยได้มอบหมายให้บริษัท IBM, Texas Instruments และ Intermetrics เป็นผู้ศึกษาและพัฒนาดำเนินการไปอย่างต่อเนื่องและได้ผลเป็นที่น่าพอใจ จนกระทั่งปี ค.ศ. 1985 ITAR ได้ยกเลิกการถ่ายทอดเทคโนโลยีทางทหารออกจากโครงการ ด้วยเหตุนี้ภาษา VHDL จึงเริ่มเป็นที่รู้จักกันโดยทั่วไป หลังจากนั้น IEEE ได้นำภาษานี้มาศึกษาจนกระทั่ง ค.ศ. 1987 และกำหนดมาตราฐาน IEEE 1076-1987 โดยมีชื่อเรียกว่า VHDL ต่อมาได้รับการปรับปรุงจนในปัจจุบันเป็นมาตรฐาน IEEE 1076-1993 หรือ VHDL 1993

การที่ DoD ในขณะนั้นซึ่งเป็นลูกค้ารายใหญ่ของอุตสาหกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์จึงมีผู้สนใจรับโครงการต่างๆ จาก DoD ไปดำเนินการด้านวิจัยและพัฒนาเป็นอย่างมาก ดังนั้น เพื่อที่จะให้เป็นมาตราฐานเดียวกันหมด ทาง DoD จึงกำหนดว่า ในการส่งโครงการนั้นจะต้องเขียนอยู่ในรูปของภาษา VHDL เท่านั้น ซึ่งจะทำให้เกิดข้อดีต่อ DoD เองที่มาตราฐานเดียวกัน สามารถนำไปจำลองกับเครื่องคอมพิวเตอร์ได้หลายๆ ระบบ

2.5.1 แนะนำภาษา VHDL

VHDL เป็นภาษาบรรยายพฤติกรรมทางฮาร์ดแวร์สำหรับการออกแบบในระดับสูงทางอิเล็กทรอนิกส์ (Electronics) มีความสามารถในการเลียนแบบ (Simulation), การสังเคราะห์ (Synthesis), และการทดสอบ (Testing) ในการจำลองระบบอิเล็กทรอนิกส์ดิจิตอลลักษณะการทำงานของ VHDL จะทำงานไปพร้อมๆ กัน (Concurrent) และเป็นลำดับ (Sequential Statements) ซึ่งหมายถึงทุกๆ คำสั่ง องค์ประกอบเกตหรือวงจรตรรกจะนำมาปฏิบัติทั้งหมดจึงดูเหมือนว่าได้มีการปฏิบัติไปพร้อมๆ กัน

2.5.2 ขั้นตอนในการออกแบบ

1) Behavioral เป็นขั้นตอนในการสร้างแบบจำลองการบรรยายพฤติกรรมในขั้นตอนแรก

2) RTL (Registor Transfer Logic) การบรรยายการถ่ายโอนลอจิก

3) Logic เป็นขั้นตอนการสังเคราะห์ให้เป็นวงจรที่แสดงการเชื่อมต่อ ประกอบด้วยเกตต่างๆ

4) Layout เป็นขั้นตอนการเจือสารในส่วนของภาษา VHDL ครอบคลุมขั้นตอนการออกแบบBehavioral, RTL และ Logic

2.5.3 วัตถุประสงค์ในขั้นตอนของ Behavioral มาถึงขั้น RTL ก็เพื่อ

1) ช่วยในการกำหนดการทำงานและการแก้ไขจุดผิดพลาดของโปรแกรม

2) ช่วยในการวิเคราะห์ปัญหาของโปรแกรม

3) เป็นวิธีในการระบุรายละเอียดและการแยกส่วนการทำงานของโปรแกรม

4) เป็นการตรวจสอบและเปรียบเทียบผลลัพธ์ของโปแกรม

2.5.4 ข้อได้เปรียบของการใช้ VHDL

1) การออกแบบมีคุณภาพที่สูงกว่า

2) ออกแบบวงจรที่มีความซับซ้อนมากๆ ได้

3) ใช้เวลาในการออกแบบสั้นกว่า

4) ค้นหาข้อผิดพลาดและเปลี่ยนแปลงแก้ไขได้ง่าย

2.5.5 ภาษา VHDL สามารถใช้ในการบรรยายได้ 3 ลักษณะคือ

1) การบรรยายเชิงพฤติกรรม (Behavioral) เป็นการบรรยายลักษณะการทำงานของอุปกรณ์ฮาร์ดแวร์ในเชิงพฤติกรรม ซึ่งเป็นการบรรยายลักษณะการเปลี่ยนแปลงของข้อมูลในรูปแบบของข้อมูลที่เข้ามา โดยไม่คำนึงถึงว่าลักษณะโครงสร้าง หรือความสัมพันธ์ของอุปกรณ์ที่อยู่ภายใน

2) การบรรยายเชิงข้อมูล (Dataflow) เป็นการบรรยายลักษณะการเลื่อนไหลของข้อมูลผ่านรีจิสเตอร์และบัสของระบบ ซึ่งเป็นระดับขั้นของการบรรยายที่อยู่กึ่งกลางระหว่างการบรรยายเชิงพฤติกรรม และการบรรยายเชิงโครงสร้าง เครื่องมือที่ใช้ในการควบคุมการไหลเลื่อน ได้แก่ Conditional, Selected และ Guarded

3) การบรรยายเชิงโครงสร้าง (Structural) เป็นการบรรยายลักษณะการทำงานของระบบเชิงโครงสร้าง โดยกำหนดรายการของอุปกรณ์และการเชื่อมต่อระหว่างอุปกรณ์ต่างๆ ที่ใช้ในระบบ ซึ่งการบรรยายในระดับนี้เป็นการบรรยายที่ใกล้เคียงลักษณะของฮาร์ดแวร์มากที่สุด ภาษา VHDL ได้จัดเตรียมเครื่องมือ และลักษณะโครงสร้างของการบรรยายในลักษณะนี้ไว้ 4 ลักษณะคือ

3.1) ความสารถในการเลือกหรือกำหนดอุปกรณ์ที่ต้องการ ได้จากไลบรารี

3.2) การสร้างไลบรารีเพื่อเก็บอุปกรณ์ที่ผู้ใช้ออกแบบไว้เองได้

3.3) กลไกในการเชื่อมต่อระหว่างอุปกรณ์

3.4) โครงสร้างของการกำหนดอุปกรณ์ชนิดเดียวกันซ้ำๆ กัน

2.5.6 ส่วนประกอบหลักของภาษา VHDL

1) โครงสร้าง จากเอกสารของ DoD (The United State Department of Defense) ได้กล่าวว่า ภาษาบรรยายทางฮาร์ดแวร์เป็นภาษาที่ต้องการ การกำหนดรูปแบบทางโครงสร้างเพราะการกำหนดโครงสร้างนี้จะช่วยในการออกแบบทั้งระบบที่ง่ายไปจนถึงระบบที่ซับซ้อน ซึ่งระบบที่ว่านี้ถือเป็นมาตรฐานของภาษาบรรยายทางฮาร์ดแวร์

2) การทำงานที่พร้อมกัน ในวงจรอิเล็กทรอนิกส์ อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอและจะมีเรื่องของเวลาเข้ามาควบคุมการทำงานเสมอ ภาษา VHDL ได้รับการออกแบบมาเพื่อบรรยายรูปแบบทางดิจิตอล และการป้องกันของเวลา สำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วนของสถาปัตยกรรม (Architechture) จะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซส (process) ซึ่งมีการทำงานภายในเป็นลำดับคำสั่งก็ตาม หากมีหลายๆ โปรเซสอยู่ภายในโครงสร้างเดียวกันก็จะทำงานไปพร้อมกันด้วย

3) ลำดับของคำสั่ง ถึงแม้ว่าลักษณะเฉพาะของภาษาบรรยายทางฮาร์ดแวร์จะสนับสนุนการปฏิบัติตามคำสั่งอย่างเป็นลำดับเป็นกระบวนการ ตัวภาษา VHDL ก็ยังจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งเอาไว้ และเมื่อนักออกแบบได้บรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดที่อยู่ในแต่ละองค์ประกอบได้ในลักษณะเดียวกันกับการเขียนโปรแกรม ซึ่งจะประกอบด้วยโครงสร้างแบบ if-then-else และ loop การบรรยายแบบลำดับคำสั่ง ทำให้การออกแบบหน้าที่การทำงานของภาษา VHDL ก็ยังเป็นการทำงานแบบพร้อมเพรียงกันอยู่

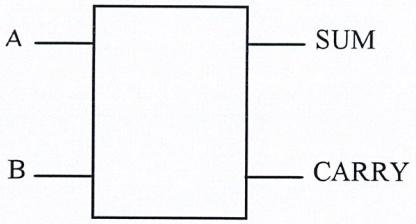
4) เวลาที่ใช้ในการควบคุม ภาษา VHDL เป็นภาษาที่อนุญาตให้ผู้ออกแบบ สามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ การตรวจสอบการออกแบบเกต หรือการหน่วงเวลาสามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอยเหตุการณ์ นอกจากนี้ก็ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้

2.5.7 องค์ประกอบในการเขียนภาษา VHDL

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

1) Entity เป็นส่วนที่ใช้ในการประกาศ อธิบายการเชื่อมต่อกับองค์ประกอบภายนอกแต่ยังมิได้กำหนดการกระทำใดๆ ทั้งสิ้น ซึ่งในส่วนประกอบนี้จะต้องมีการกำหนดชื่อ (ห้ามซ้ำกับชื่อสงวนของภาษา VHDL) แล้วตามด้วย is ตัวอักษรในภาษา VHDL ทั้งตัวใหญ่ตัวเล็กมีค่าเท่ากันดังตัวอย่างแสดงในรูปที่ 2.6

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

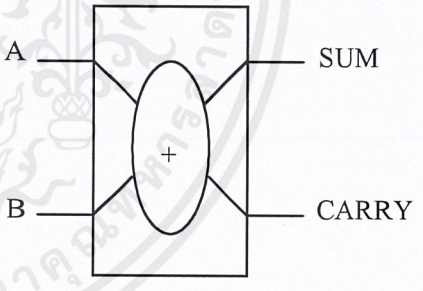


รูปที่ 2.6 โปรแกรมการใช้งานและลักษณะการติดต่อใช้งานของ entity

จากรูปที่ 2.6 จะเห็นว่ามีการประกาศชื่อ entity HALFADD ตาม is แล้วมีการกำหนดสัญญาณอินพุตและเอาต์พุต โดยมีวงเล็บเปิดและวงเล็บปิด ในแต่ละบรรทัดจะถูกปิดท้ายด้วย (;) และใช้ในการกำหนดสัญญาณให้กับตัวแปรและจบ entity ด้วยคำสั่ง end ตามด้วยชื่อของ entity

2) Architecture เป็นส่วนที่ใช้การกำหนดการทำงานภายในของ entity ต้องเป็นการเชื่อมโยงกันของ entity จำเพาะซึ่ง entity เดียวสามารถมีได้หลาย Architecture จะต้องมีการกำหนดชื่อของ Architecture นั้นๆ ด้วยว่าเป็นของ entity ไหน การเขียน Architecture จะตามหลัง begin เสมอและจบด้วยคำสั่ง end ตามด้วยชื่อของ Architecture นั้น ดังตัวอย่างแสดงในรูปที่ 2.7

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



รูปที่ 2.7 โปรแกรมการใช้งานและลักษณะการติดต่อใช้งานของ Architecture

3) Configuration ในภาษา VHDL สามารถสร้างอุปกรณ์เก็บเอาไว้ได้ และสามารถจำลอง Socket ของอุปกรณ์ในแต่ละตัว เพื่อความสะดวกเหมือนกับการใช้งานจริงในการเชื่อมต่ออุปกรณ์ ดังรูปที่ 2.7 ถูกสร้างขึ้นเป็น Socket ให้กับอุปกรณ์ โดยในการนำไปใช้งานนั้นจะอาศัย Component ตามด้วยชื่ออุปกรณ์มีการกำหนดสัญญาณอินพุตเป็นแบบบิตให้กับตัวแปร A และ B ส่วนของเอาต์พุตก็เป็นสัญญาณแบบแบบบิตที่กำหนดค่าให้กับตัวแปร SUM และ CARRY เหมือนกับ entity HALFADD ทุกประการ และจบด้วยคำสั่ง end component; สำหรับการเชื่อมต่อกันนั้นจะใช้คำสั่ง

port-map โดยเป็นการให้กับทางขวามือ คือ u1 : HALFADD port map (A, B, N_SUM, N_CARRY) ; โดยการเขียนนี้ต้องอยู่หลังจาก begin ดังตัวอย่างแสดงในรูปที่ 2.8

```
component HALFADD
      port (A,B : in bit ;
            SUM, CARRY : out bit);
      end component
begin
      u1 : HALFADD port map (A, B, N_SUM, N_CARRY);
DEFAULT
      CONFIGURATION
      Entity HALFADD is
      Port (A, B : in bit ;
```

รูปที่ 2.8 โปรแกรมและการติดต่อใช้งานของ Configuration

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

รูปที่ 2.9 โปรแกรมการเชื่อมต่ออุปกรณ์ภายในและการประกาศสายสัญญาณ

จากรูปที่ 2.9 มีการประกาศสัญญาณภายในอุปกรณ์คือ Signal N_CARRY1, N_CARRY2 ให้มีสัญญาณเป็นแบบบิต จะสังเกตเห็นว่าวงจรมีอุปกรณ์ภายใน 3 ตัว คือ HALFADD2 ตัวและ ORGATE 1 ตัวจึงประกอบด้วย u1, u2, u3 และสัญญาณที่เขียนใน Port map จะเรียงสัญญาณเข้าจน

ถึงสัญญาณออกตามลำดับและจบโปรแกรมด้วยคำสั่ง end STRUCTURAL ตามด้วยชื่อของ Architecture เสมอ ส่วนของ component เป็นส่วนหนึ่งของ Architecture

4) Process ส่วนนี้เป็นการบรรจุคำสั่งลำดับที่ต้องเขียนอยู่ภายใน Architecture ซึ่ง Process ในหลายๆ Process จะมีการทำงานที่พร้อมกัน (Concurrent) ตัวแปรทางขวามือจะถูกกำหนดค่าให้จากสัญญาณทางซ้ายมือ (<=) ในการเริ่มโปแกรมใช้คำสั่ง process ละจบด้วยคำสั่ง ene process ส่วนประกอบของการบรรยายโปรเซสประกอบด้วยส่วนของการประกาศตัวแปรที่ต้องใช้และส่วนของการปฏิบัติตามคำสั่งเพื่อให้ได้ผลลัพธ์ตามที่ต้องการ ดังรูปที่ 2.10

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

รูปที่ 2.10 โปรแกรมการทำงานของ Process

จากรูปที่ 2.10 ชื่อของ Process คือ AND_OR_FUNC มีสัญญาณอินพุตเป็น A และ B โดยใช้คำสั่ง if-then-else ในการลำดับการทำงานดังนี้ ถ้า A=1 หรือ B=1 จะให้ Z_OR มีค่าเป็น 1 เงื่อนไขนอกจากนั้น Z_OR จะมีค่าเป็น 0 จบการทำงานด้วย end if ; และถ้า A=1 และ B=1 ดังนั้น Z_AND จะมีค่าเป็น 1 เงื่อนไขนอกจากนั้น Z_AND จะมีค่าเป็น 0 และจบ end process AND_OR_FUNC ; และจบ Architecture ด้วยคำสั่ง end BEHAVE ;

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

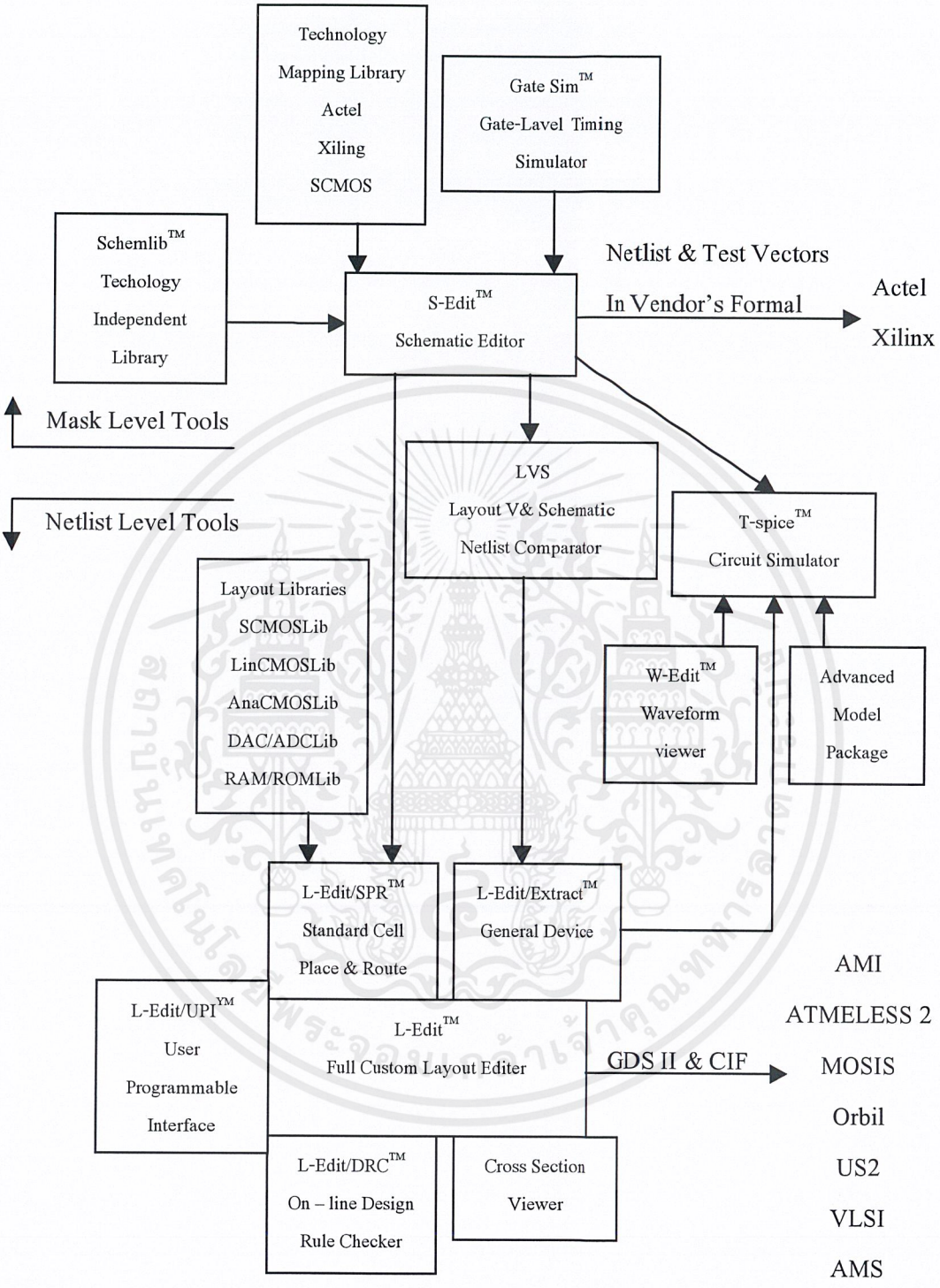
6) Library เป็นส่วนของการเก็บโปรแกรมต่างๆ ที่สมบูรณ์ซึ่งภายในไลบรารีจะประกอบด้วย entity, architecture, package, configulation

2.6 เครื่องมือที่ใช้ออกแบบวงจรรวม

การออกแบบวงจรรวมโดยใช้ Tanner Tools ซึ่งเป็นซอฟแวร์ที่สามารถออกแบบได้ทั้งในระดับ Netlist โดยการออกแบบ Schematic Diagram จำลองการทำงานระดับเกต (Gate-Level Simulation) กับในระดับวงจร (Circuit-Level Simulation) และในระดับ Mask Layout โดยเริ่มจากทำ Library Standard Cell เพื่อการ Place & Route ของวงจรที่ได้ จากนั้นทำ Layout จาก Netlist File ที่ได้จากการออกแบบวงจรจากการใช้ภาษา VHDL โดยนำงานออกแบบวงจรเฉพาะงานตัวเข้ารหัสและถอดรหัสข้อมูลเสียงที่ใช้วิธีการของ DCT มาทำการ Simulation กับ T-Spice ปัจจุบันสามารถทำการออกแบบด้วยวิธีการออกแบบขั้นสูง โดยใช้ VHDL หรือ Verilog แล้วทำการสังเคราะห์ลงบน FPGA แล้ว แต่ยังไม่เหมาะสมที่จะใช้งานจริง เนื่องจาก FPGA มีข้อจำกัดในการนำไปใช้งานบางอย่างที่ต้องการประสิทธิภาพสูงของวงจร เช่น ความเร็ว การสร้างวงจรรวม ซึ่งเป็นสิ่งที่จำเป็น แต่ปัจจุบันสามารถทำได้โดยเครื่องมือที่มีราคาถูก และประสิทธิภาพเพียงพอที่จะใช้ในการออกแบบอย่างเช่น Tanner Tools™ [3]

2.6.1 การใช้ Tanner Tools™ System

Tanner Tools เป็นซอฟแวร์ที่ช่วยในการออกแบบวงจรรวมได้อย่างสมบูรณ์บนเครื่องคอมพิวเตอร์ PC ประกอบด้วย 3 ส่วนหลักๆ แสดงดังรูปที่ 2.11



รูปที่ 2.11 แผนผังการทำงาน Tanner Tools

ส่วนที่ 1 จำลองการทำงานของวงจร

T-Spice เป็นส่วนจำลองการทำงานของวงจรแอนาลอกและดิจิตอล

W-Edit เป็นตัวแสดงสัญญาณ waveform ที่ได้จาก T-Spice

Gate-Sim ใช้จำลองการทำงานของวงจรดิจิตอลร่วมกับค่า Timing Delays

ส่วนที่ 2 ส่วนออกแบบในระดับ Netlist

S-Edit เป็นซอฟแวร์ที่ช่วยใช้สร้างและแก้ไข Circuit Schematics ของวงจรได้ทั้งแอนาลอกและดิจิตอลโดยมีความสัมพันธ์กับซอฟแวร์อื่นๆ

LVS เป็นเครื่องมือที่ใช้เปรียบเทียบ Spice Netlist ที่ได้จาก Schematics กับ Spice Netlist ที่ถูก Extract จาก Layout เพื่อให้ทราบว่า Layout ที่ออกแบบเป็นวงจรจรเดียวกัน

ส่วนที่ 3 ส่วนออกแบบในระดับ Layout

L-Edit เป็น Low-Level, Full Custom, Mask Editor ซึ่งไม่สามารถทำการแปลง Layer ของ Layout ได้โดยอัตโนมัติ (Automatic Layer Transformations) ได้ L-Edit สามารถอ่านและบันทึกในรูปแบบของ Standard Mask Layout Interchange Formats (GDS II และ CIF) หรือโครงสร้างข้อมูลของ Tanner (TDS = Tanner Database) ซึ่งสามารถทำได้เร็วกว่า 2 แบบข้างต้นโดยโครงสร้างของข้อมูล TDB จะใช้ในระหว่างกระบวนการ การออกแบบเพื่อเรียกใช้เก็บข้อมูลจากการออกแบบส่วน CIF กับ GDS II จะใช้ในการถ่ายทอดข้อมูลไปสู่โรงงานผลิตชิป หรือ CAD Tools ตัวอื่นๆ

L-Edit/SPR จะสร้าง Layout สำหรับ Standard Cell Design รวมถึงการทำ Placement, Routing และ Pad Routing

L-Edit/Extract จะสร้าง SPICE-Compatible Circuit Netlist จาก L-Edit Layout

L-Edit/DRC จะมีส่วนของ User Programmable Rules เช่น Minimum Width, Exact Width เป็นต้น และมีส่วนของ Error Browser, Object Browser สำหรับทำ Rule-Checking Error และสนับสนุนโครงสร้างแบบ 45 องศา และ 90 องศา

นอกจากนี้ยังมี ASIC Library ของ MOSIS และ ORBIT[4] ให้ใช้ในการออกแบบซึ่งไลบรารี ทั้งหมดประกอบด้วย

1) SCMOSLib เป็นไลบรารีของ ดิจิตอลลอจิกและ I/O Pads

2) DAC/ADCLib เป็นไลบรารีของ วงจร DAC และ ADC

3) Lin CMOSLib เป็นไลบรารีของ วงจรเชิงเส้น

4) Ana CMOSLib เป็นไลบรารีของ วงจรแอนาลอก

5) RAM/ROMLib เป็นไลบรารีของ ใช้สร้าง Layout ของ RAM หรือ ROM โดยไลบรารีทั้งหมดสามารถใช้กับเทคโนโลยี 2.0, 1.2, 1.0 และ 0.8 ไมครอน

บทที่ 3
การออกแบบ การสร้างและการทำงาน

3.1 กล่าวนำ

ในกระบวนการออกแบบและการสร้างวงจรรวมเฉพาะกิจ ตัวเข้ารหัสและถอดรหัส DCT เป็นวิธีการนำสัมประสิทธิ์ DCT และ IDCT มาใช้การออกแบบโดยใช้ภาษา VHDL เป็นตัวอธิบายการทำงานของระบบทั้งหมด เพื่อนำไปสู่การสังเคราะห์ให้เป็นวงจรในระดับเกทโดย Sofeware ที่ใช้ในการสังเคราะห์ ดังนั้นในบทนี้จึงนำส่วนที่ได้มีการออกแบบจากสัมประสิทธิ์การแปลง DCT และ IDCT ไว้แล้วมาสร้าง Mask Layout ด้วยเทคโนโลยีการผลิต CMOS 0.5 ไมครอน ของโรงงาน Alcatel Microelectronics

3.2 การคำนวณหาค่าสัมประสิทธิ์ของการแปลง DCT และ IDCT

ในการคำนวณหาค่าสัมประสิทธิ์การแปลงสัญญาณนั้นจะทำการคำนวณค่าสมการทางคณิตศาสตร์ของการแปลงสัญญาณ

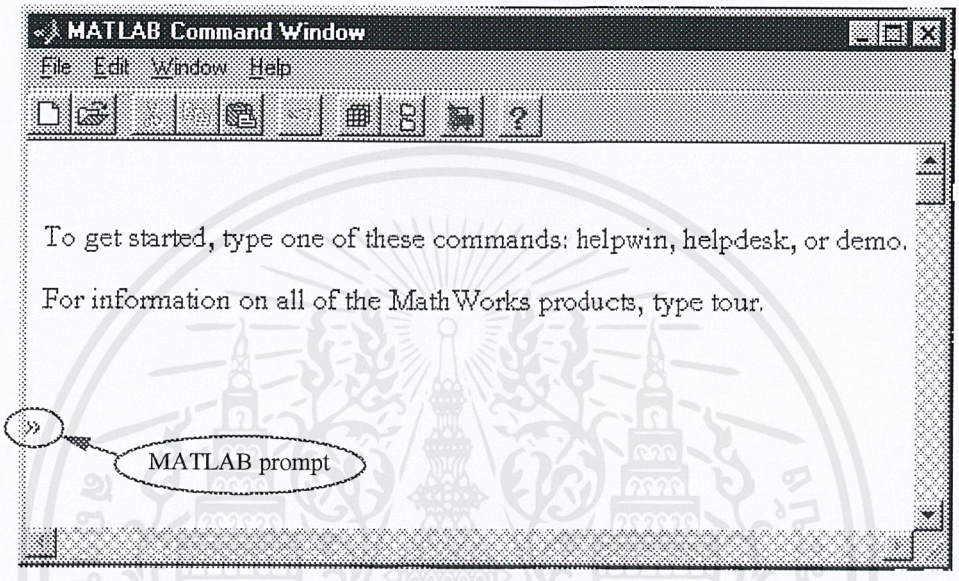
3.2.1 การใช้งาน MATLAB

โปรแกรม MATLAB เป็นโปรแกรมที่ออกแบบมาเพื่ออำนวยความสะดวกในการคำนวณทางคณิตศาสตร์โดยเฉพาะอย่างยิ่งการคำนวณทางเวกเตอร์ และเมตริกซ์ทั้งในระบบจำนวนจริงและระบบจำวนเชิงซ้อนดังนั้นโปรแกรม MATLAB จะมีการกำหนดฟังก์ชั่นทางคณิตศาสตร์ ค่าตัวแปร และคำสั่งที่ใช้ในการจัดการไฟล์ และไดเรคทอรี่ทั้งนี้ก็เพื่ออำนวยความสะดวกแก่ผู้ใช้งาน

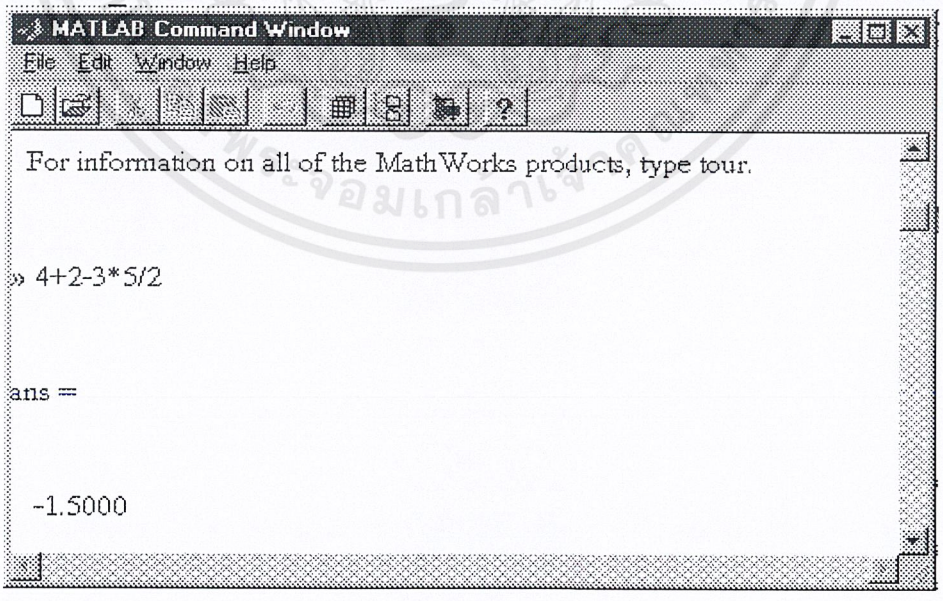
การใช้งานโปรแกรม MATLAB เบื้องต้น

เมื่อเริ่มต้นใช้งานโปรแกรม MATLAB หน้าแรกที่พบก็คือหน้าต่างคำสั่ง บนหน้าต่างคำสั่งจะพบเครื่องหมาย MATLAB prompt (>>) ที่ทำการคำนวณและประมวลผลโดยตำแหน่งเคอร์เซอร์จะอยู่ทางด้านขวามือของเครื่องหมาย “>>” ของหน้าต่างดังนี้

เนื่องจากโปรแกรม MATLAB ได้มีการแสดงในลักษณะ Interrective mode ทำให้เมื่อพิมพ์คำสั่งในการคำนวณและประมวลผลโดยตรงที่เครื่องหมาย ">>" บน Command Windows แล้วกดที่ปุ่ม (Enter) บนคีย์บอร์ด โปรแกรม MATLAB ก็จะคำนวณและให้คำตอบทันทีบนหน้าต่างคำสั่งดังรูปที่ 3.1



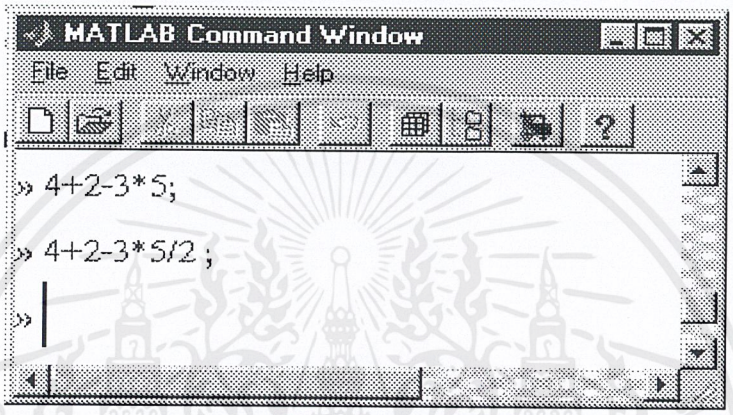
รูปที่ 3.1 หน้าต่างคำสั่งและเครื่องหมาย MATLAB prompt



รูปที่ 3.2 ตัวอย่างการพิมพ์คำสั่งและคำตอบที่โปรแกรมคำนวณได้

จากรูปที่3.2 ถ้าไม่ต้องการให้มีการพิมพ์ผลลัพธ์ในการคำนวณออกมาบนหน้าต่างคำสั่งก็สามารถกระทำได้โดยพิมพ์เครื่องหมาย เซมิโคลอน (;) ต่อท้ายคำสั่ง

เมื่อไม่ต้องการที่จะทำหารป้อนคำสั่งทีละคำสั่งบนเส้นคำสั่ง(Command Line) ก็สามารถทำได้โดยการเขียนคำสั่งบน Editor หรือ Word Processor ใดๆ ในการเขียนซึ่งคำสั่งลักษณะนี้เรียกว่า Script M-File ซึ่งทำการบันทึกไว้ใน ไฟล์ *.m และนำกลับมาใช้ในภายหลังได้ ดังรูปที่ 3.3



รูปที่ 3.3 ตัวอย่างคำสั่งที่ไม่ต้องการให้มีการพิมพ์ผลลัพธ์ในการคำนวณ

3.2.2 การคำนวณหาค่าสัมประสิทธิ์ของการแปลง DCT

การแปลงข้อมูลที่อยู่ในโดเมนเวลา (Time Domain) ให้เปลี่ยนไปอยู่ในโดเมนความถี่ (Frequency Domain) โดยสมการที่ (3.1) เป็นสมการที่ใช้ในการเข้ารหัส DCT โดยใช้วิธีการหาค่าสัมประสิทธิ์การเปลี่ยนแปลง ข้อมูลที่ได้นำไปสู่กระบวนการทางคณิตศาสตร์จะได้ข้อมูลใหม่ที่มีการแปลงข้อมูลแบบเมตริกออกมาดังสมการที่ (3.3) ผลที่ได้จะได้ค่าสัมประสิทธิ์การเปลี่ยนแปลงที่อยู่ในรูปของ เมตริกซ์ ในกระบวนการหาค่าและคำนวณได้ใช้โปรแกรมสำเร็จรูป MATLAB เป็นโปรแกรมที่ช่วยในการหาค่าสัมประสิทธิ์การเปลี่ยนแปลงที่อยู่ในรูปสมการเมตริกซ์ สมการ DCT

$$X(k) = \frac{1}{2} C(k) \sum_{i=0}^{N-1} X(i) \cos\left[\frac{(2i+1)k\pi}{2N}\right] \quad ; \quad k = 0, 1, 2, ..., N-1 \qquad \textbf{(3.1)}$$

เมื่อ

$$\pi = 180$$

$$C(k) = \begin{cases} 1/\sqrt{2} & ; \quad k = 0 \\ 1 & ; \quad k = 0, 1, 2, ..., N-1 \end{cases}$$

การแปลงข้อมูลแบบเมตริกซ์

$$g = \mathrm{DCT}(f) \tag{3.2}$$

$$g = AfA^{T} \tag{3.3}$$

เมื่อ g คือ สัมประสิทธิ์การเปลี่ยนแปลงของ DCT

A คือ เมตริกซ์ของสัมประสิทธิ์ DCT หรือเมตริกซ์ $X(k)$

A^{T} คือ Transpose ของเมตริกซ์ A

f คือ ข้อมูลอินพุต

จากสมการการแปลงสัญญาณ DCT เมื่อกำหนดให้ความยาวของสัญญาณซึ่งมีขนาด 8 บิต ดังนั้น N = 8 และเมื่อใช้ข้อมูลเริ่มต้นตั้งแต่ข้อมูลที่ 0 ถึง 7 หรือกำหนดให้ i = 0, 1, 2, …, 7 ดังนั้น สัมประสิทธิ์ของการแปลง $X(k)$ ก็จะได้เป็นเมตริกซ์ขนาด 8×8 ดังในตารางที่ (3.1)

เมื่อ $X(k)$ เป็นสัมประสิทธิ์ของการแปลง DCT การแปลงข้อมูลเมตริกซ์จากสัมประสิทธิ์ขนาด 8×8 โดยมีสมการหรือรูปแบบของ DCT จากสมการที่ (3.2) จะเห็นว่าข้อมูลที่ออกมาจะอยู่ในรูปของ เมตริกซ์

ตารางที่ 3.1 ค่าสัมประสิทธิ์ของการแปลง DCT

i/k	0	1	2	3	4	5	6	7
0	0.70710	0.98078	0.92387	0.83146	0.70710	0.55557	0.38268	0.19509
1	0.70710	0.83146	0.38268	-0.19509	-0.70710	-0.98078	-0.98078	-0.55557
2	0.70710	0.55557	-0.38268	-0.98078	-0.70710	0.19509	0.92387	0.83146
3	0.70710	0.19509	-0.92387	-0.55557	0.70710	0.83146	-0.38268	-0.98078
4	0.70710	-0.91509	-0.92387	0.55557	0.70710	-0.83146	-0.38268	0.98078
5	0.70710	-0.55557	-0.38268	0.98078	-0.70710	-0.19509	0.92387	-0.83146
6	0.70710	-0.83146	0.38268	0.19509	-0.70710	0.98708	-0.92387	0.55557
7	0.70710	-0.98078	0.92387	-0.83146	0.70710	-0.55557	0.38268	-0.19509

3.2.3 การคำนวณหาค่าสัมประสิทธิ์ของการแปลง IDCT

การแปลงข้อมูลที่อยู่ในโดเมนความถี่ (Frequency Domain) ให้เปลี่ยนไปอยู่ในโดเมนเวลา (Time Domain) โดยสมการที่ (3.4) เป็นสมการที่ใช้ในการถอดรหัส DCT โดยที่ใช้วิธีการหาค่าของสัมประสิทธิ์การเปลี่ยนแปลง ซึ่งข้อมูลที่ได้นำไปสู่กระบวนการทางคณิตศาสตร์จะได้ข้อมูลใหม่ที่มีการแปลงข้อมูลแบบเมตริกออกมาดังสมการที่ (3.6) ผลที่ได้จะได้ค่าสัมประสิทธิ์การเปลี่ยนแปลงที่อยู่ในรูปของเมตริกซ์ ในกระบวนการหาค่าและคำนวณได้ใช้โปรแกรมสำเร็จรูป MATLAB เป็นโปรแกรมที่ช่วยในการหาค่าสัมประสิทธิ์การเปลี่ยนแปลงที่อยู่ในรูปสมการเมตริกซ์ สมการ IDCT

$$X(i) = \frac{1}{2}\sum_{k=0}^{N-1} C(k)X(k)\cos\left[\frac{(2i+1)k\pi}{2N}\right] \quad ; \quad i = 0,1,2,...,N-1 \qquad \textbf{(3.4)}$$

เมื่อ

$$\pi = 180$$

$$C(k) = \begin{cases} 1/\sqrt{2} & ; \quad k = 0 \\ 1 & ; \quad k = 0,1,2,...,N-1 \end{cases}$$

การแปลงกลับข้อมูลแบบเมตริกซ์

$$h = IDCT(f) \qquad \textbf{(3.5)}$$

$$h = A^{T} gA \qquad \textbf{(3.6)}$$

เมื่อ g คือ สัมประสิทธิ์การเปลี่ยนแปลงของ DCT

A คือ เมตริกซ์ของสัมประสิทธิ์ DCT หรือเมตริกซ์ $X(i)$

A^{T} คือ Transpose ของเมตริกซ์ A

h คือ สัมประสิทธ์การแปลงกลับของ DCT

จากสมการการแปลงสัญญาณ IDCT เมื่อกำหนดให้ ความยาวของสัญญาณมีขนาด 8 บิต ดังนั้น N = 8 สัมประสิทธิ์ของการแปลงมีค่าเริ่มจาก k = 0, 1, 2, …, 7 ดังนั้นข้อมูลจึงเริ่มต้นตั้งแต่ข้อมูลที่ 0 ถึง 7 ดังในตารางที่ (3.2)

เมื่อ $X(i)$ เป็นค่าข้อมูลเริ่มต้นการแปลง DCT การแปลงข้อมูลเมตริกซ์จากขนาด 8 x 8 โดยมีสมการหรือรูปแบบของ DCT จากสมการที่ (3.6) จะเห็นว่าข้อมูลที่ออกมาจะอยู่ในรูปของเมตริกซ์ซึ่งเป็นค่าข้อมูลที่ได้จากการสุ่มตัวอย่างข้อมูลเสียงที่อยู่ในรูปของเมตริกซ์

ตารางที่ 3.2 ค่าสัมประสิทธิ์ของการแปลง IDCT

i/k	0	1	2	3	4	5	6	7
0	0.70710	0.70710	0.70710	0.70710	0.70710	0.70710	0.70710	0.70710
1	0.98078	0.83146	0.55557	0.19509	-0.91509	-0.55557	-0.83146	-0.98078
2	0.92387	0.38268	-0.38268	-0.92387	-0.92387	-0.38268	0.38268	0.92387
3	0.83146	-0.19509	-0.98078	-0.55557	0.55557	0.98078	0.19509	-0.83146
4	0.70710	0.70710	-0.70710	-0.70710	0.70710	0.70710	-0.70710	-0.70710
5	0.55557	-0.98078	0.19509	0.83146	-0.83146	-0.19509	0.98708	-0.55557
6	0.38268	-0.98078	0.92387	-0.38268	-0.38268	0.92387	-0.92387	0.38268
7	0.19509	-0.55557	0.83146	-0.98078	0.98078	-0.83146	0.55557	-0.19509

3.2.4 การปรับปรุงค่าสัมประสิทธิ์ของ DCT และ IDCT

จากผลการคำนวณค่าสัมประสิทธิ์ของการแปลง DCT และ IDCT ดังตารางที่ 3.1 และ 3.2 ตามลำดับจะเห็นได้ว่าผลที่ได้นั้นจะอยู่ในรูปของเมตริกซ์ ขนาด 8×8 และค่าที่ได้ทั้งหมดนั้นจะเป็นเลขทศนิยมที่มีค่าไม่เกิน 1 ซึ่งค่าที่อยู่ในรูปแบบของจำนวนที่เรียกว่า Floating Point แต่เนื่องจากกระบวนการในการทำงานต่อไปจำเป็นที่จะต้องนำค่าจำนวนที่เป็นรูปแบบของ Fixed Point ไปใช้ในการคำนวณทั้งหมด ดังนั้นจึงจำเป็นที่จะต้องทำการปรับปรุงค่าของสัมประสิทธิ์ที่ได้ให้อยู่ในรูปของจำนวนเต็ม โดยใช้วิธีในการคูณสัมประสิทธิ์ทั้งหมดด้วยค่า 10 แล้วปัดจำนวนทศนิยมให้เป็นจำนวนเต็ม ซึ่งในการแปลงค่าสัมประสิทธิ์ของการแปลง DCT และ IDCT นั้น เอกลักษณ์ของแต่ละค่ายังคงเหมือนเดิมและยังสัมพันธ์กันอยู่

ค่าสัมประสิทธิ์ DCT และ IDCT ที่ได้ทำการปรับปรุงแล้วแสดงดังตารางที่ 3.3 และ 3.4 ตามลำดับ

ตารางที่ 3.3 ค่าสัมประสิทธิ์ของการแปลง DCT หลังการปรับปรุงค่า

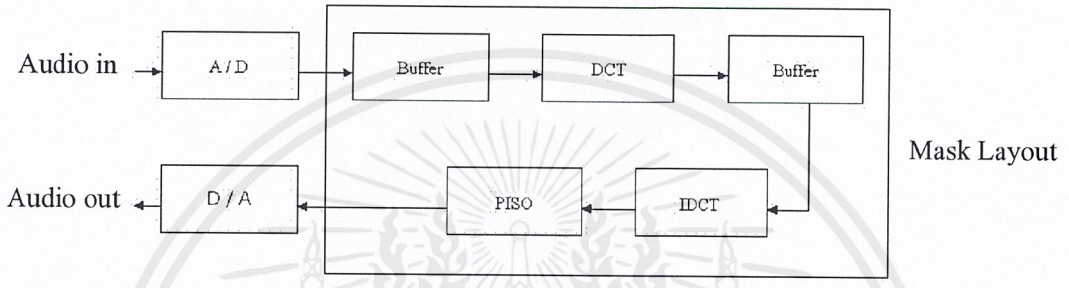
i/k	0	1	2	3	4	5	6	7
0	7	10	9	8	7	5	3	1
1	7	8	3	-1	-7	-10	-9	-5
2	7	5	-3	-10	-7	1	9	8
3	7	1	-9	-5	7	-8	-3	10
4	7	-1	-9	5	5	-8	-3	10
5	7	-5	-3	10	-7	-1	9	-8
6	7	-8	3	1	-7	10	-9	5
7	7	-10	9	-8	7	-5	3	-1

ตารางที่ 3.4 ค่าสัมประสิทธิ์ของการแปลง IDCT หลังการปรับปรุงค่า

i/k	0	1	2	3	4	5	6	7
0	7	7	7	7	7	7	7	7
1	10	8	5	1	-1	-5	-8	-10
2	9	3	-3	-9	-9	-3	3	9
3	8	-1	-10	-5	5	10	1	-8
4	7	-7	-7	7	7	-7	-7	7
5	5	-10	1	8	-8	-1	10	-5
6	3	-9	9	-3	-3	9	-9	3
7	1	-5	8	-10	10	-8	5	-1

3.3 การเขียนชุดคำสั่งในการประมวลผล

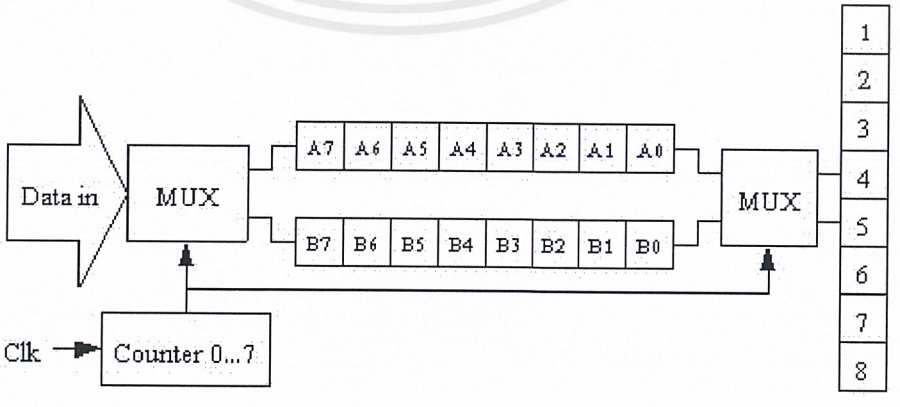
การเขียนโปรแกรมในส่วนของการทำงาน เพื่อนำไปทำการสังเคราะห์ และ Mask Layout ใช้ภาษา VHDL ในการเขียนโปรแกรมบรรยายพฤติกรรมทางฮาร์ดแวร์ของภาคต่างๆ ในวงจรรวมการเข้ารหัสและถอดรหัส DCT โดยได้แบ่งส่วนการเขียนโปรแกรมออกเป็นส่วนๆ เพื่อที่ง่ายต่อการออกแบบ ไว้ทั้งหมด 4 ภาคการทำงานใหญ่ๆ ดังรูปที่ 3.4



รูปที่ 3.4 แผนผังการทำงานของโปรแกรมทั้งหมด

3.3.1 การเขียนโปรแกรมส่วน Buffer

Buffer เป็นส่วนที่ทำการเตรียมข้อมูลนำเข้า มีลักษณะสัญญาณแบบดิจิตอล ขนาดความยาวของข้อมูล 8 บิต มีลักษณะการส่งสัญญาณอนุกรม ซึ่งโปรแกรม Buffer นี้จะเป็นส่วนที่จัดข้อมูลให้มีลักษณะเป็นแบบเมตริกซ์ ขนาด 8×1 โดยจะทำการนับข้อมูลที่เข้ามาทางด้านอินพุตได้ครบจำนวน 8 บิต (0 - 7) เมื่อข้อมูลที่มาสู่ Buffer ครบ Buffer จะทำการส่งข้อมูลออกไปยังวงจร DCT โดยมีลักษณะเป็นข้อมูลแบบขนาน ขนาด 8 ช่องสัญญาณ ดังรูปที่ 3.5



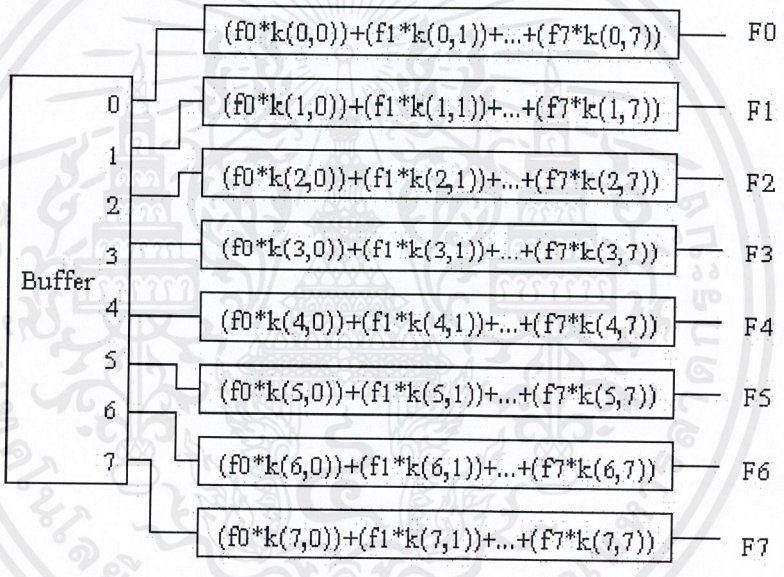
รูปที่ 3.5 แผนผังการทำงานของโปรแกรม Buffer

3.3.2 การเขียนโปรแกรมส่วน Parallel In Serial Out (PISO)

การทำงานในส่วนนี้จะเป็นส่วนการทำงานหลังจากส่วนของ IDCT ซึ่งทำงานโดยจัดข้อมูลที่ขนานกันอยู่ในลักษณะของเมตริกซ์ ขนาด 8×8 ที่เป็นข้อมูลอนุกรมเพื่อต่อเข้าไปยังวงจรภายนอกต่อไป

หลักการเขียนโปรแกรมในส่วนนี้ จะใช้หลักการนำข้อมูลขนานซึ่งไม่มีสัญญาณนาฬิกาเข้าจังหวะสัญญาณนำมาซิงโครไนซ์รวมกับสัญญาณนาฬิกาที่กำหนดขึ้นและจะแยกข้อมูลออกเป็นห้วงๆ จึงได้ข้อมูลกลับเป็นข้อมูลอนุกรมนั่นเอง

3.3.3 การเขียนโปรแกรมส่วน DCT



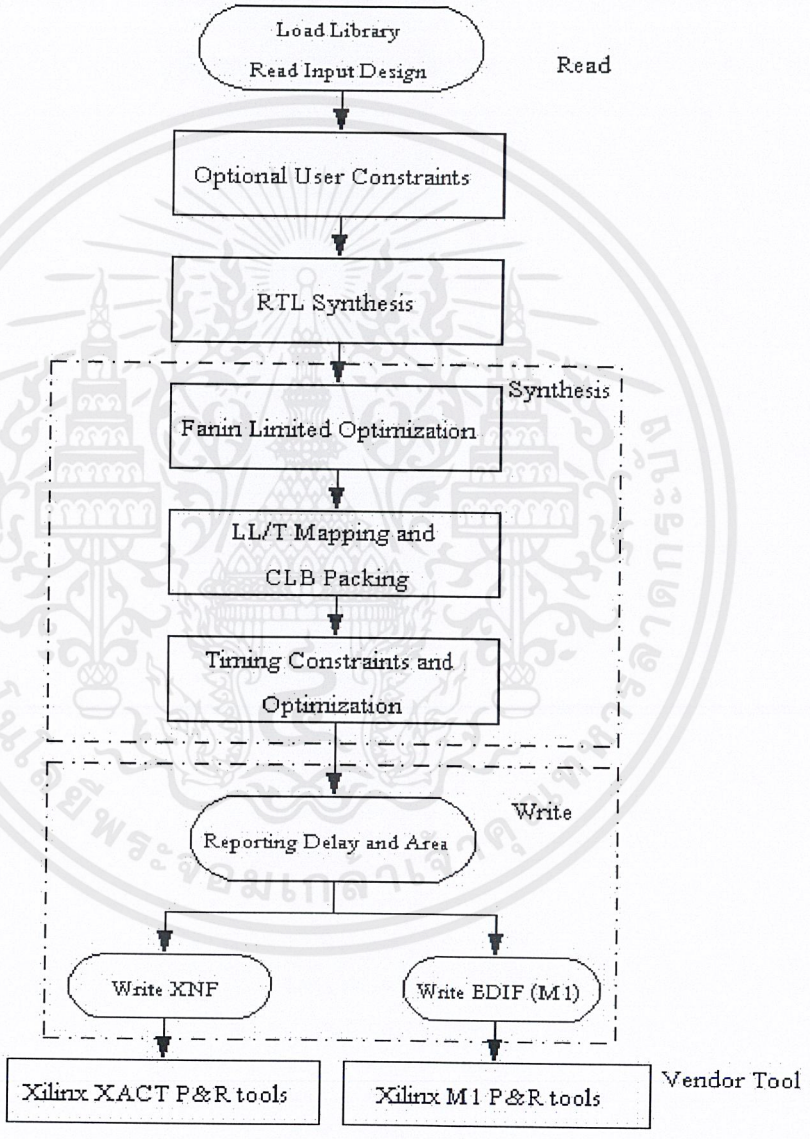
รูปที่ 3.6 แผนผังการทำงานโปรแกรมส่วน DCT

3.3.4 การเขียนโปรแกรมส่วนของ IDCT

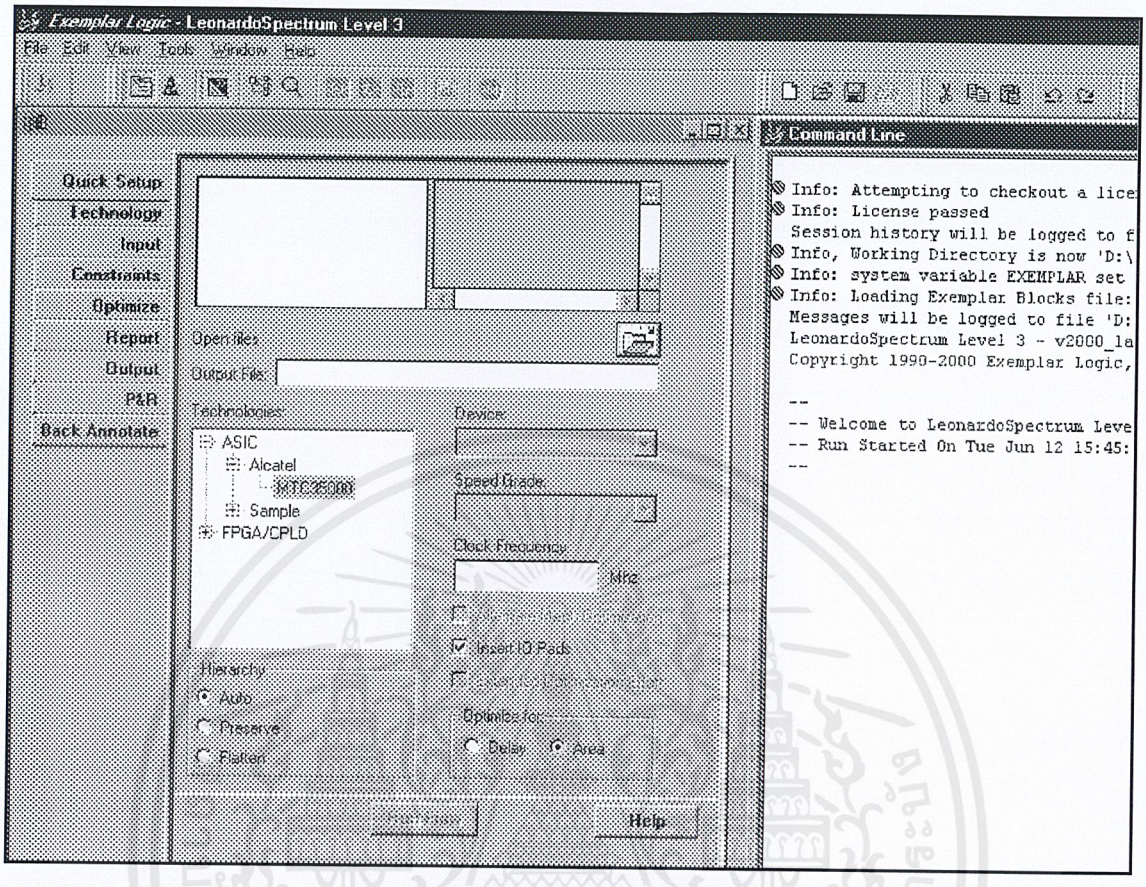
ในส่วนของการเขียนโปรแกรมส่วนของ IDCT นั้น โครงสร้างหลักของโปรแกรมจะเหมือนกับโปรแกรม DCT เพียงแต่ค่าสัมประสิทธิ์ที่นำมาคูณเข้ากับข้อมูลนั้น จะเป็นค่าสัมประสิทธิ์ที่ได้จากการคำนวณสมการ IDCT ในตาราง ที่ 3.4

3.4 การใช้โปรแกรม Leonardo Spectrum เพื่อการสังเคราะห์วงจรต่างๆ

เมื่อได้ออกแบบวงจรดิจิตอลโดยใช้ภาษา VHDL และรวมถึงการตรวจสอบวงจรในระดับฟังก์ชั่น (Function) เรียบร้อยแล้วขั้นต่อไปก็คือการนำไฟล์ภาษา VHDL ที่ออกแบบไปสังเคราะห์โดยใช้โปรแกรม Leonardo Spectrum ซึ่งมีขั้นตอนต่างๆ ดังในรูปที่ 3.7 และเมนูในรูปที่ 3.8



รูปที่ 3.7 Synthesis Design Flows



รูปที่ 3.8 หน้าต่างของโปรแกรม Leonardo Spectrum

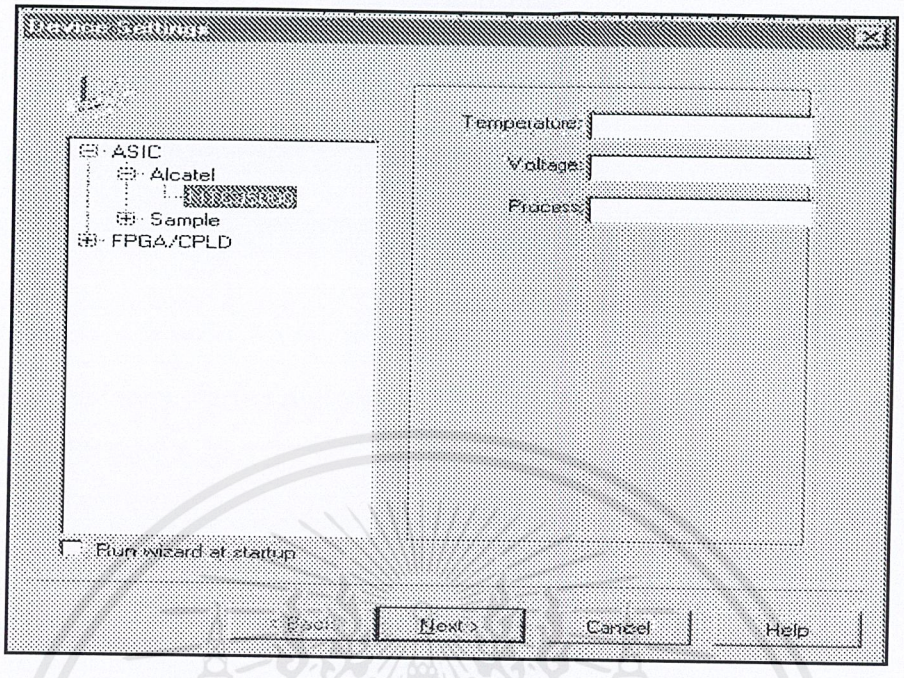
3.4.1 การ Synthesis Wizard หรือ Flows Tabs Technology

เพื่อเลือกอุปกรณ์(Device)จากไลบรารีซึ่งจะมีให้เลือก 2 ไลบรารี คือ

1) ASIC Library เป็นไลบรารี ของอุปกรณ์ ASIC(Application Specific Integrated Circuits) เทคโนโลยี

2) FPGA/CPLD Library เป็นไลบรารีอุปกรณ์ FPGA(Field Programmable Gate Array) หรือ CPLD(Complex Programmable Logic Device)ตระกูลต่างๆ ดังรูปที่ 3.8

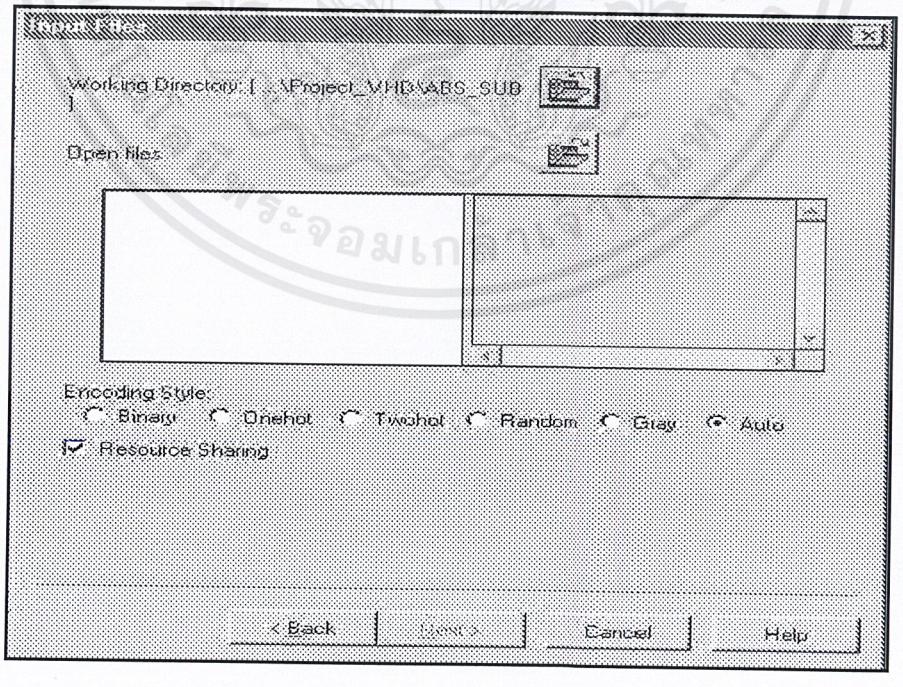
จากนั้น ก็ให้ใส่ค่า Temperature, Voltage, Process (ในรูปที่ 3.9 นี้ค่าต่างๆ เป็นของ MTC35000)



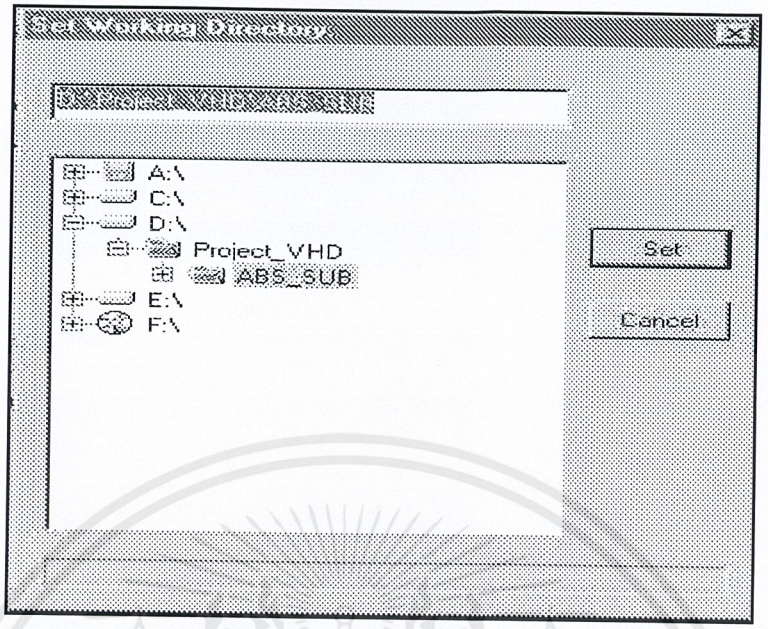
รูปที่ 3.9 หน้าต่างการเลือกเทคโนโลยีในการสังเคราะห์

3.4.2 การเลือก Working Directory และไฟล์ VHDLที่ต้องการสังเคราะห์

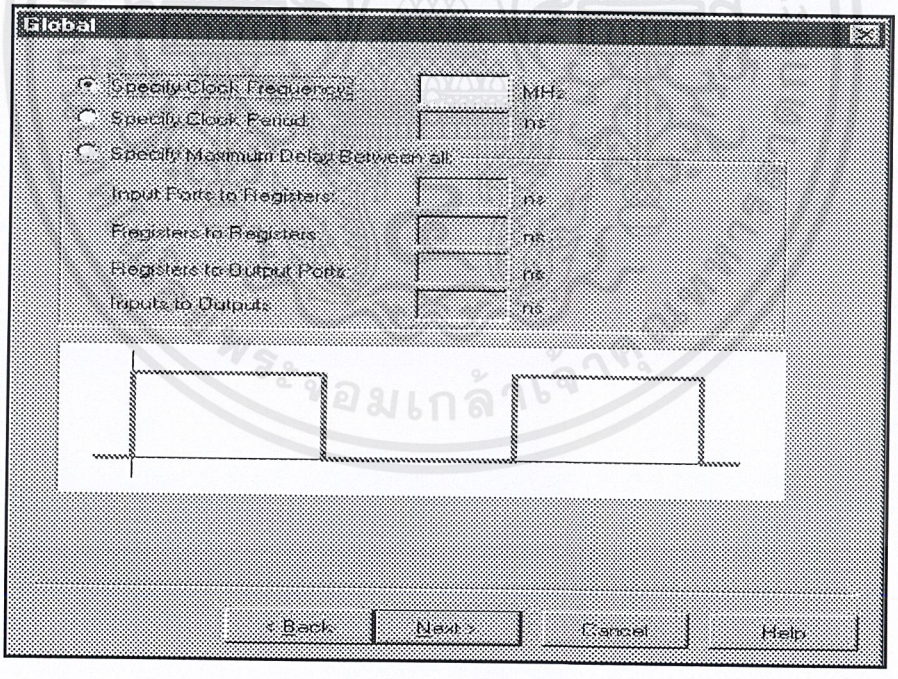
ขั้นตอนต่างๆ ดังรูปที่ 3.10 – 3.11



รูปที่ 3.10 หน้าต่างการเลือก Input File



รูปที่ 3.11 หน้าต่างการเลือก Working Directory

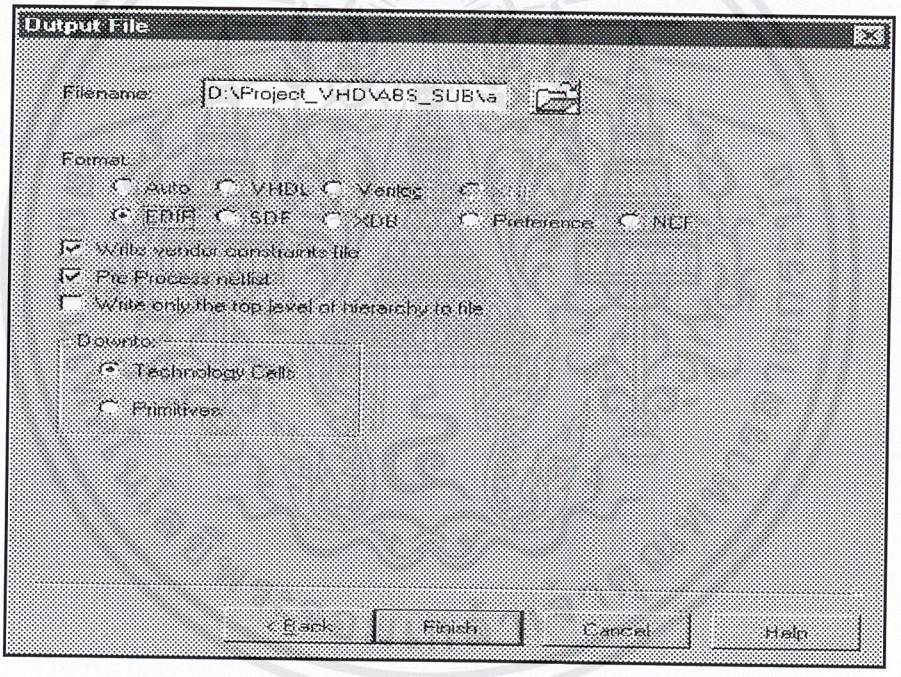


รูปที่ 3.12 หน้าต่างการกำหนดความเร็วของวงจร

หลังจากนั้นก็ให้คลิก ที่ Open File ก็จะพบหน้าต่าง Set Input File และก็ให้เลือกไฟล์ที่ต้องการสังเคราะห์ หลังจากนั้นก็ให้คลิก Next > ก็จะพบเมนูการกำหนดความเร็วของวงจรที่จะสังเคราะห์ ดังรูปที่ 3.12

3.4.3 การกำหนด Format ของ Output File

คลิก Next > ก็จะพบเมนู เพื่อกำหนด Output File โดยเลือก Format เป็น EDIF ไฟล์ เพื่อนำไปทำขั้นตอนของการทำ Standard Place & Route และ Block Cells Place & Route โดยถ้าเลือกใช้ Option ของ Downto: เป็น Primitives ในการเลือก Format แบบ VHDL โดยถ้าเลือกใช้ Option ของ Downto: เป็น Technology Cells ในการเลือก Format แบบ EDIF XNF เมื่อเลือกค่าได้ครบถูกต้องแล้วก็ คลิก Finish ดังรูปที่ 3.13



รูปที่ 3.13 หน้าต่างการเลือกเทคโนโลยีในการสังเคราะห์

ถ้าผลของการสังเคราะห์ที่ออกแบบไม่พบปัญหาใด โปรแกรมจะรายงานการสังเคราะห์ที่ได้เช่น ขนาดของวงจรที่สังเคราะห์ ความเร็วของวงจร(ถ้ามี) แต่ถ้าพบปัญหาในการสังเคราะห์ โปรแกรมจะทำการแจ้งและหยุดทันทีและให้เราทำการแก้ไขต่อไป

3.4.4 การดูผลวงจรในระดับ RTL

คลิก  เพื่อดูรูปของวงจรในระดับ RTLได้ ซึ่งรูปของวงจรต่างๆ (อยู่ในภาคผนวก ก) เมื่อทำกระบวนการทั้งหมดจนทุกขั้นตอนแล้วเราก็จะได้ไฟล์ EDIFขึ้น เพื่อไปใช้ในกระบวนการทำ Standard Place & Route และ Block Cells Place & Route ต่อไป

3.5 การใช้โปรแกรม L-Edit ในการทำ Place & Route เพื่อทำ IC Layout

ในการสร้าง Layout นั้นจำเป็นต้องสร้างไฟล์ EDIF ก่อนเพื่อที่จะนำไป Mapping กับ Standard Cells และในส่วนของการสร้างไฟล์ EDIF นั้นเราก็จะกระทำสร้างไฟล์ได้โดยใช้โปรแกรม Leonardo Spectrum การออกแบบวงจรรวมโดยใช้ Tanner Tools ซึ่งเป็นโปรแกรมที่สามารถออกแบบได้ทั้งในระดับ Netlist โดยการออกแบบ Schematic Diagram จำลองการทำงานระดับเกต (Gate-Level Simulation) กับในระดับวงจร (Circuit-Level Simulation) และในระดับ Mask Layout โดยเริ่มจากทำ Library Standard Cell เพื่อการ Place&Route ของวงจรที่ ได้ จากนั้นทำ Layout จาก Netlist File ที่ได้จากการออกแบบวงจรจากการใช้ภาษา VHDL

ปัจจุบันสามารถทำการออกแบบ ด้วยวิธีการออกแบบขั้นสูง โดยใช้ VHDL หรือ Verilog แล้วทำการสังเคราะห์ลงบน FPGA แล้ว แต่ยังไม่เหมาะสมที่จะใช้งานจริง เนื่องจาก FPGA นั้นมีข้อจำกัดในการนำไปใช้งานบางประการที่ต้องการประสิทธิภาพของวงจร เช่น ความเร็ว การสร้างวงจรรวมซึ่งเป็นสิ่งที่จำเป็นมาก แต่ในปัจจุบันสามารถทำได้โดยใช้เครื่องมือที่มีราคาถูกอีกทั้งมีประสิทธิภาพเพียงพอที่จะใช้ในการออกแบบอย่างเช่น Tanner Tools ™ [3]

3.5.1 การใช้ Tanner Tools™ System

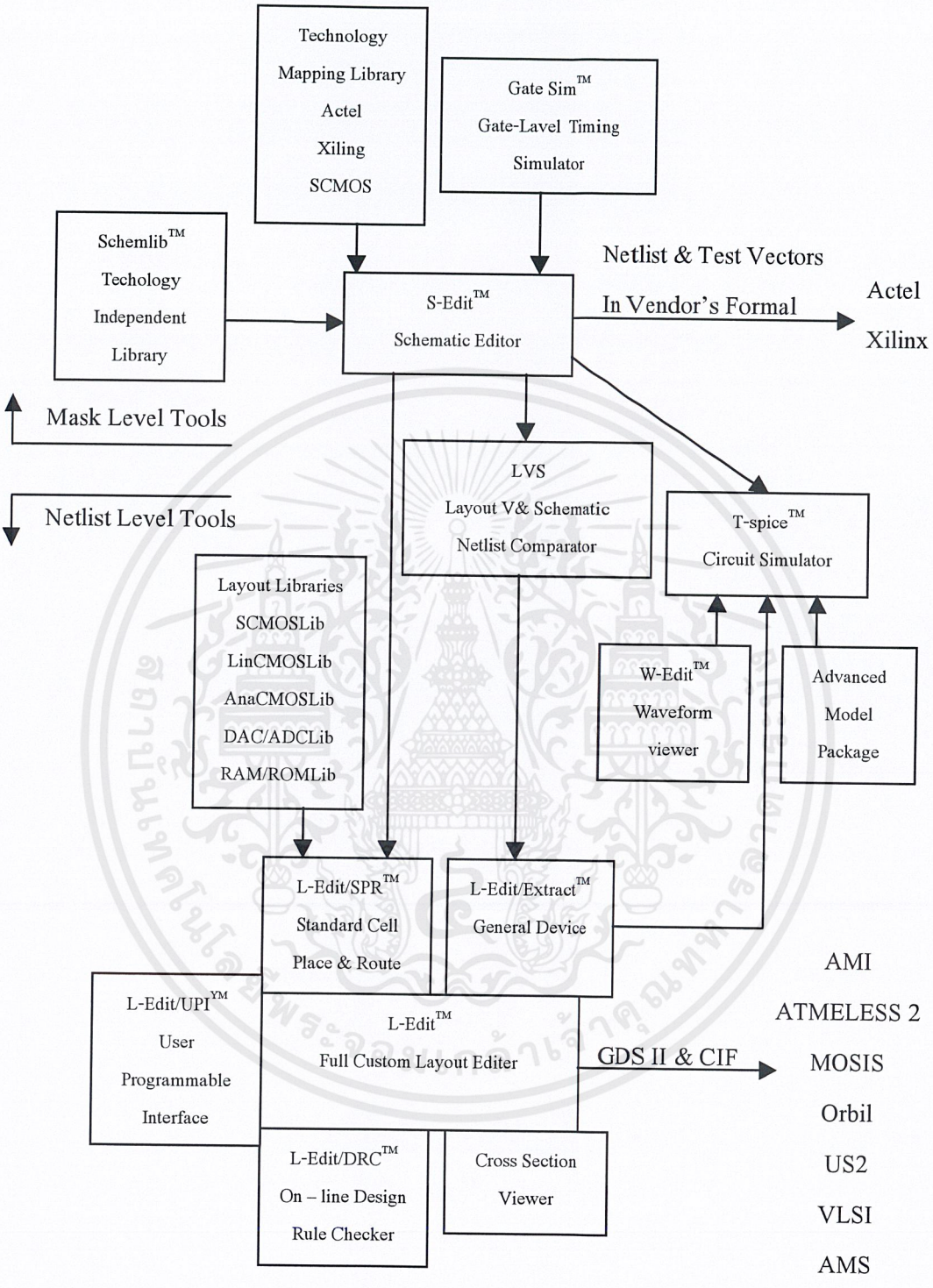
Tanner Tools เป็นโปรแกรมที่ช่วยในการออกแบบวงจรรวมได้อย่างสมบูรณ์บนเครื่องคอมพิวเตอร์ PC ประกอบด้วย 3 ส่วนหลักๆ แสดงดังรูปที่ 3.14

1) ส่วนจำลองการทำงานของวงจร

T-Spice เป็นส่วนจำลองการทำงานของวงจรแอนาลอกและดิจิตอล

W-Edit เป็นตัวแสดงสัญญาณ Wavefrom ที่ได้จาก T-Spice

Gate-Sim ใช้จำลองการทำงานของวงจรดิจิตอลร่วมกับค่า Timing Delays ในการทำ Static Timing Analysis โดยสามารถจำลองการทำงานของวงจรที่มีขนาดใหญ่



รูปที่ 3.14 แผนผังการทำงาน Tanner Tools

2) ส่วนออกแบบในระดับ Netlist

S-Edit เป็นโปรแกรมที่ช่วยใช้สร้างและแก้ไข Circuit Schematics ของวงจรได้ทั้งในแบบแอนาลอกและดิจิตอล โดยมีความสัมพันธ์กับโปรแกรมอื่น ๆ

LVS เป็นเครื่องมือที่ใช้เปรียบเทียบ SPICE Netlist ที่ได้จาก Schematics กับ SPICE Netlist ที่ถูก Extract จาก Layout เพื่อให้ทราบว่า Layout ที่ออกแบบเป็นวงจรเดียวกัน

3) ส่วนออกแบบในระดับ Layout

L-Edit เป็น Low-Level, Full Custom Mask Editor ซึ่งไม่สามารถทำแปลง Layer ของ Layout ได้โดยอัตโนมัติ (Automatic Layer Transformations) ได้ L-Edit สามารถอ่านและบันทึกในรูปแบบของ Standard Mask Layout Interchange Formats (GDS II และ CIF) หรือโครงสร้างข้อมูลของ Tanner (TDB = Tanner Database) สามารถทำได้เร็วกว่าสองแบบข้างต้น โดยโครงสร้างข้อมูลของ TDB จะถูกใช้ในระหว่างกระบวนการออกแบบเพื่อเรียกใช้และเก็บข้อมูลจากการออกแบบ ส่วน CIF กับ GDS II จะใช้ในการถ่ายทอดข้อมูลไปยังโรงงานผลิตชิป หรือ CAD Tools ตัวอื่น ๆ ในส่วนของการกำหนดขนาดของช่องไฟใน L-Edit Grid Option สามารถใช้ได้กับกฎการออกแบบได้ทั้งแบบแลมด้าและไมครอน

L-Edit/SPR จะสร้าง Layout สำหรับ Standard Cell Design รวมถึงการทำ Placement, Routing และ Pad Routing

L-Edit/Extract จะสร้าง SPICE-Compatible Circuit Netlist จาก L-Edit Layout

L-Edit/DRC จะมีส่วนของ User Programmable Rules เช่น Minimum Width, Exact Width เป็นต้น และมีส่วนของ Error Browser, Object Browser สำหรับทำ Rule-Checking Error และสนับสนุนโครงสร้างแบบ 45 องศา และ 90 องศา

นอกจากนี้ยังมี ASIC Library ของ MOSIS และ ORBIT [4] ให้ใช้ในการออกแบบ ซึ่ง Library ทั้งหมดประกอบด้วย

3.1) SCMOSLib เป็นไลบรารีของ ดิจิตอลลอจิกและ I/O Pads

3.2) DAC/ADCLib เป็นไลบรารีของ วงจร DAC และ ADC

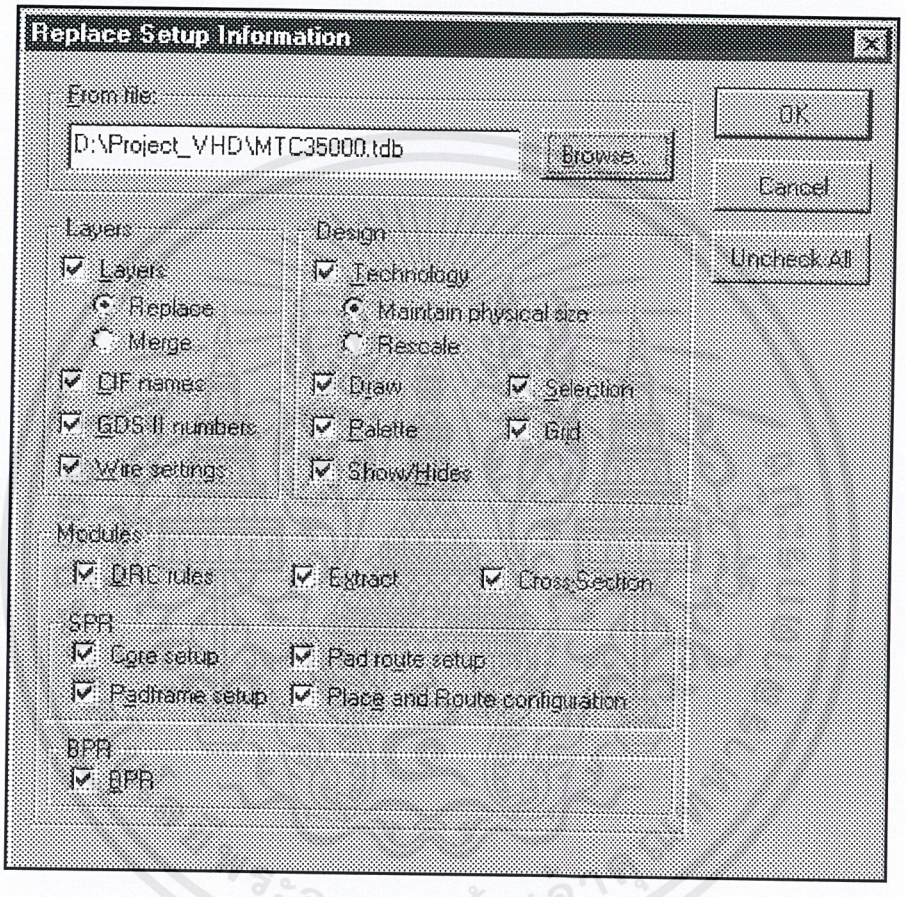
3.3) Lin CMOSLib เป็นไลบรารีของ วงจรเชิงเส้น

3.3) Ana CMOSLib เป็นไลบรารีของ วงจรแอนาลอก

3.4) RAM/ROMLib เป็นไลบรารี ใช้สร้าง Layout ของ RAM หรือ ROM โดยที่ Library ทั้งหมดสามารถใช้ กับเทคโนโลยี 2.0, 1.2, 1.0 และ 0.8 ไมครอน

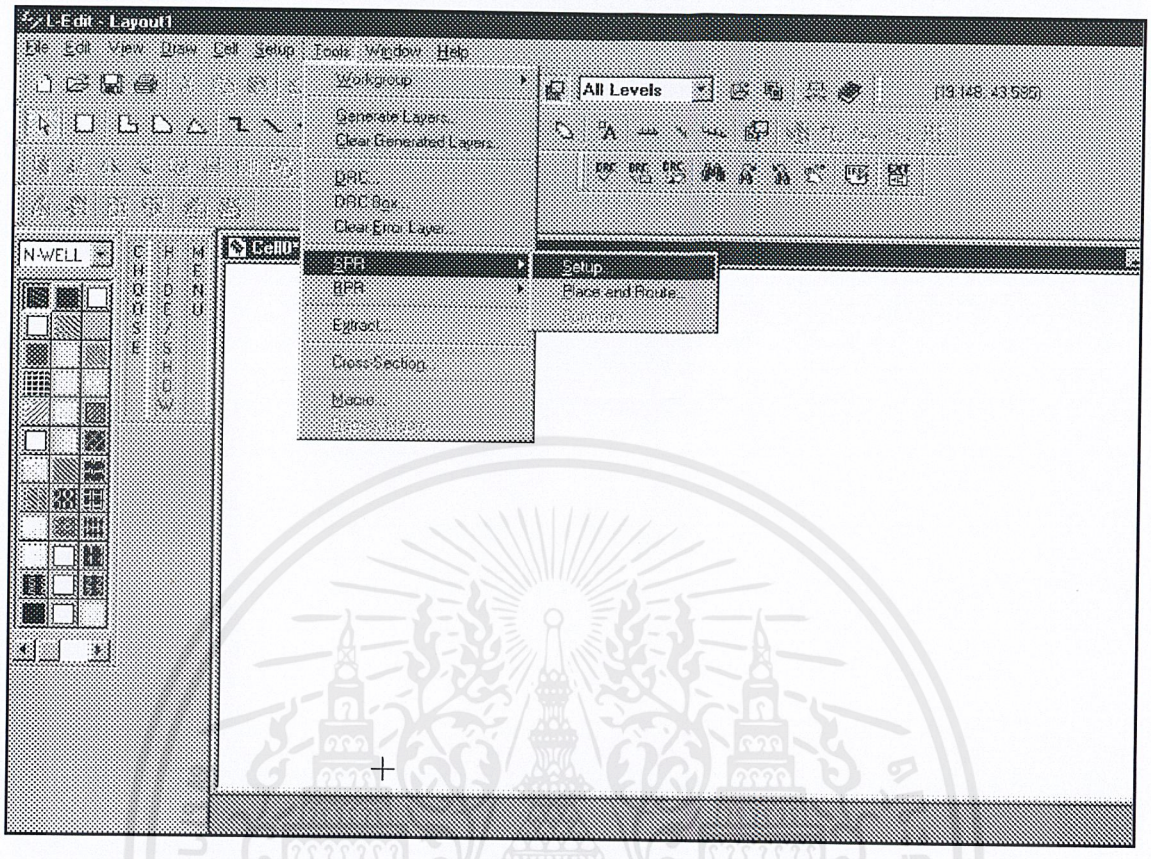
3.5.2 ขั้นตอนในการสร้าง Layout โดยทำจากเมนู SPR

1) การเลือกใช้เทคโนโลยี เริ่มด้วย RUN โปรแกรมL-Edit และไปที่ FILE >REPLACE SETUP ก็จะพบหน้าต่าง Replace Setup Information และตรง From file นั้นให้เลือกเทคโนโลยีของ Standard Cell (MTC35000) ดังรูปที่ 3.15

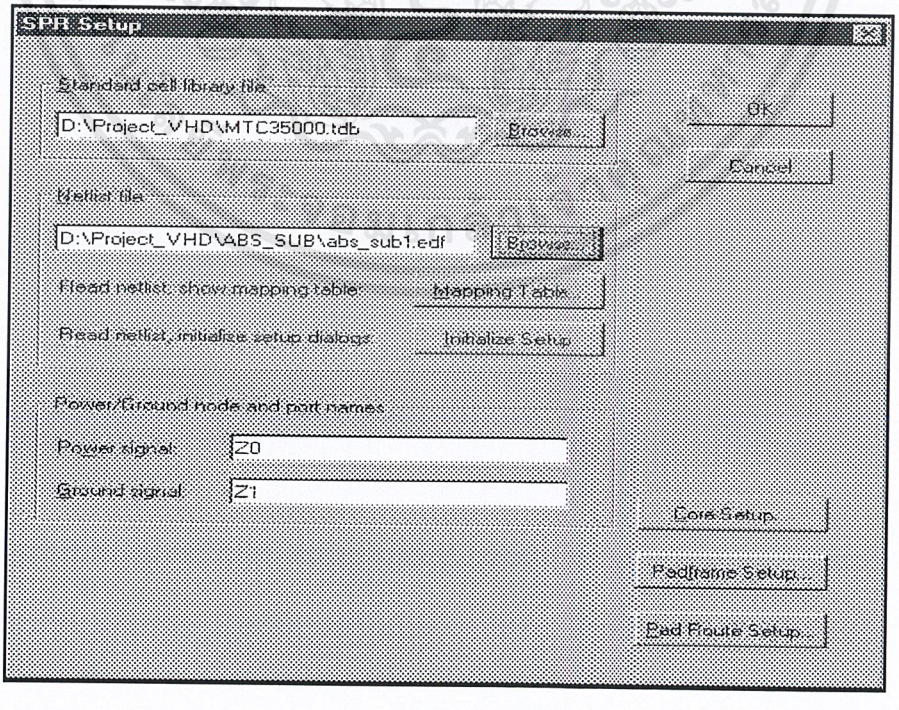


รูปที่ 3.15 การเลือกใช้เทคโนโลยี

2) การทำ Standard Cell Place&Route โดยนำ Netlist ของวงจรแต่ละส่วนที่ได้มาทำ Standard Cell Place&Routeวิธีการนำ Netlist ของวงจรแต่ละส่วนที่ได้มาทำ Standard Cell Place&Route โดยเริ่มจากไปที่เมนู Tool > SPR > SETUP ดังรูปที่ 3.16 – 3.17



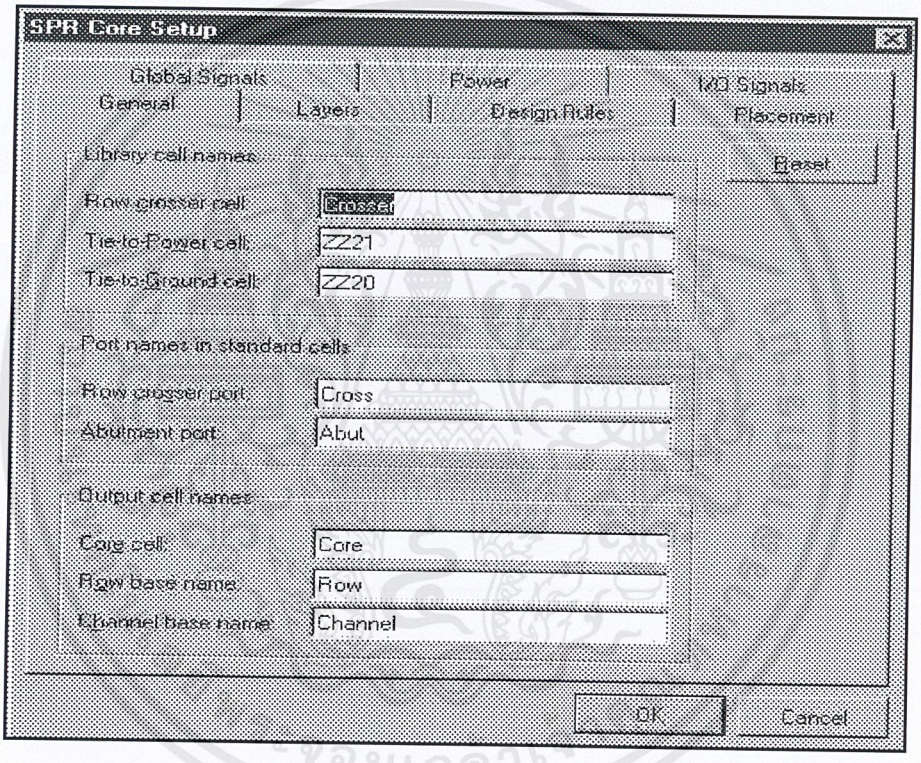
รูปที่ 3.16 การทำ Standard Cell Place&Route



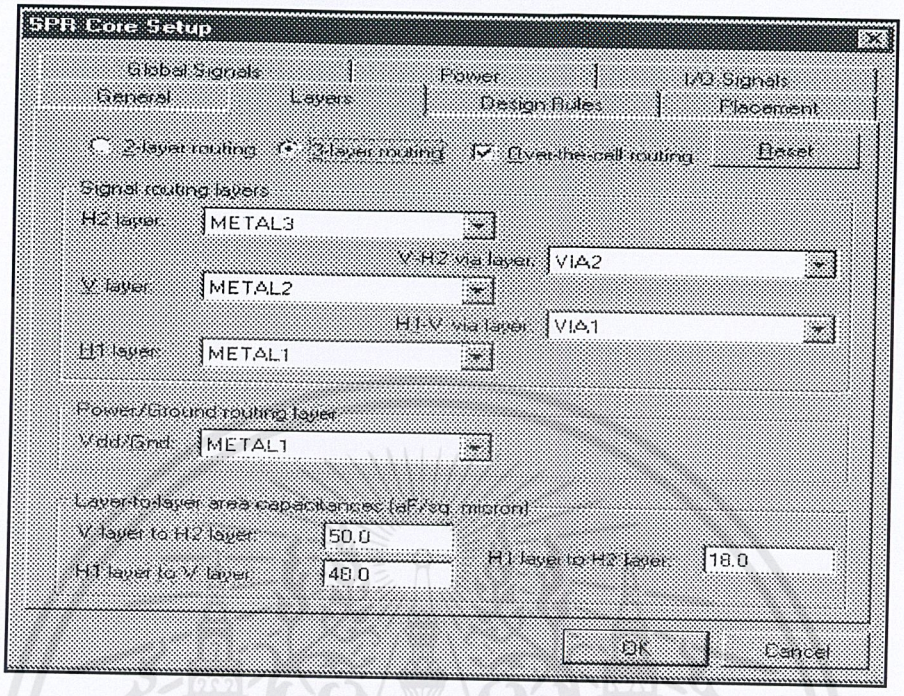
รูปที่ 3.17 การกำหนดค่าต่างๆ ใน SPR Setup

ที่ช่อง Standard Cell library fileให้ใส่ Standard Cell ที่ต้องการใช้ ในที่นี้เลือก MTC35000.tdb และในช่อง Netlist file ให้ใส่ไฟล์ EDF ที่ ได้จากการสังเคราะห์ด้วยโปรแกรม Leonardo Spectrum

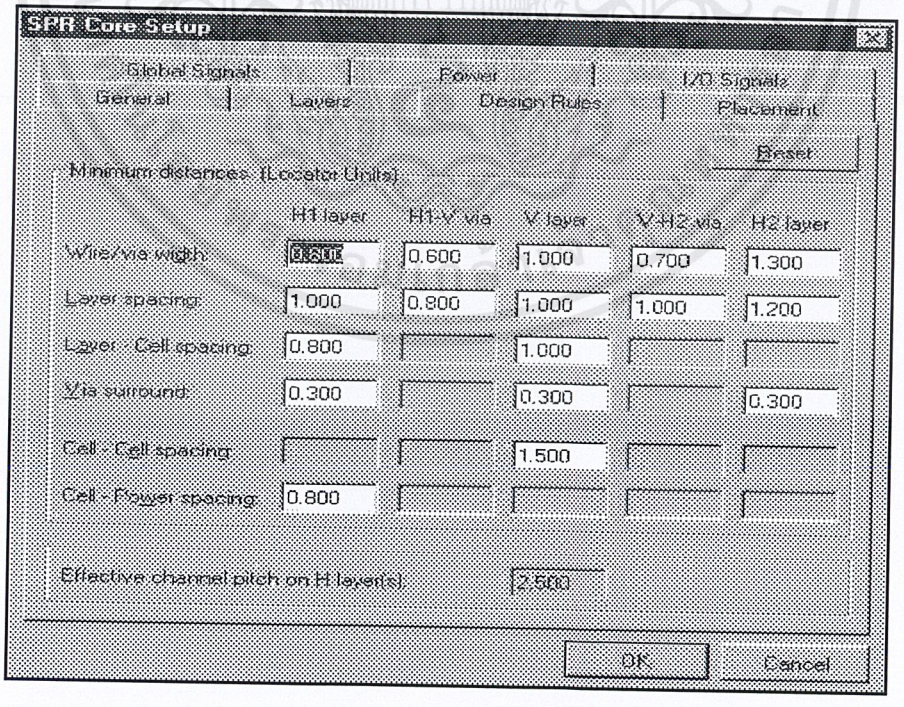
3) การกำหนดค่าใน SPR Core Setup โดย Power/Ground node and port name ในช่องของ Power signal ให้ใส่ Z0ในช่อง Ground signal ให้ใส่ Z1 (ค่าต่างๆ ที่ใส่นั้นเป็นค่าเฉพาะของเทคโนโลยีแต่ที่ใช้คือ CMOS 0.5 ไมครอน) เมื่อใส่ค่าเสร็จแล้วก็ให้กำหนดค่าใน SPR Core Setup ค่าพารามิเตอร์ต่างๆ ที่เป็นของ MTC35000.tdb จะมีค่าดังรูปที่ 3.18 – 3.24



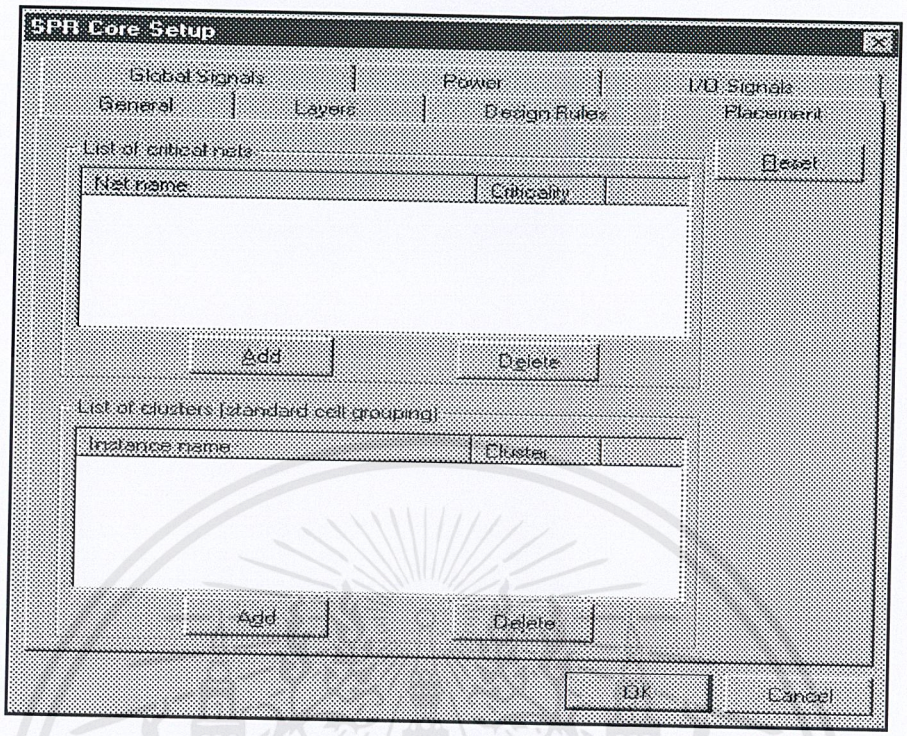
รูปที่ 3.18 การกำหนดค่า General



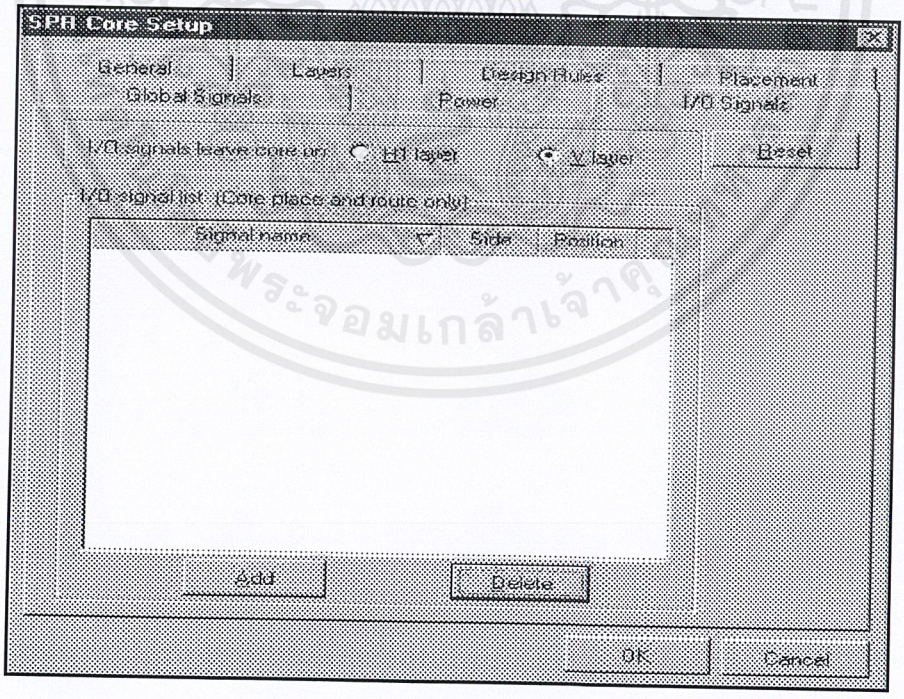
รูปที่ 3.19 การกำหนดค่า Layers



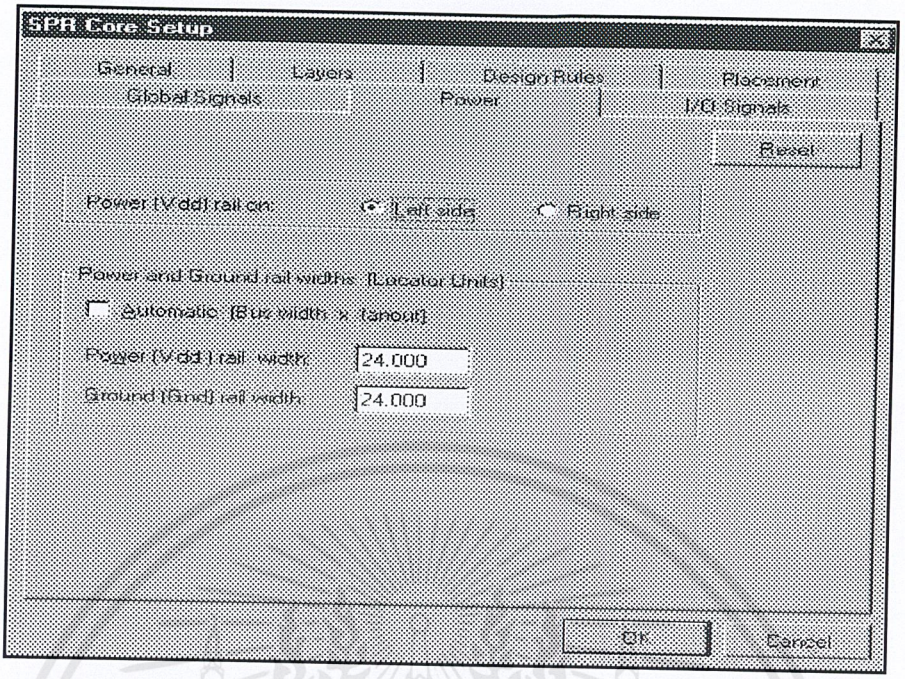
รูปที่ 3.20 การกำหนดค่า Design Rules



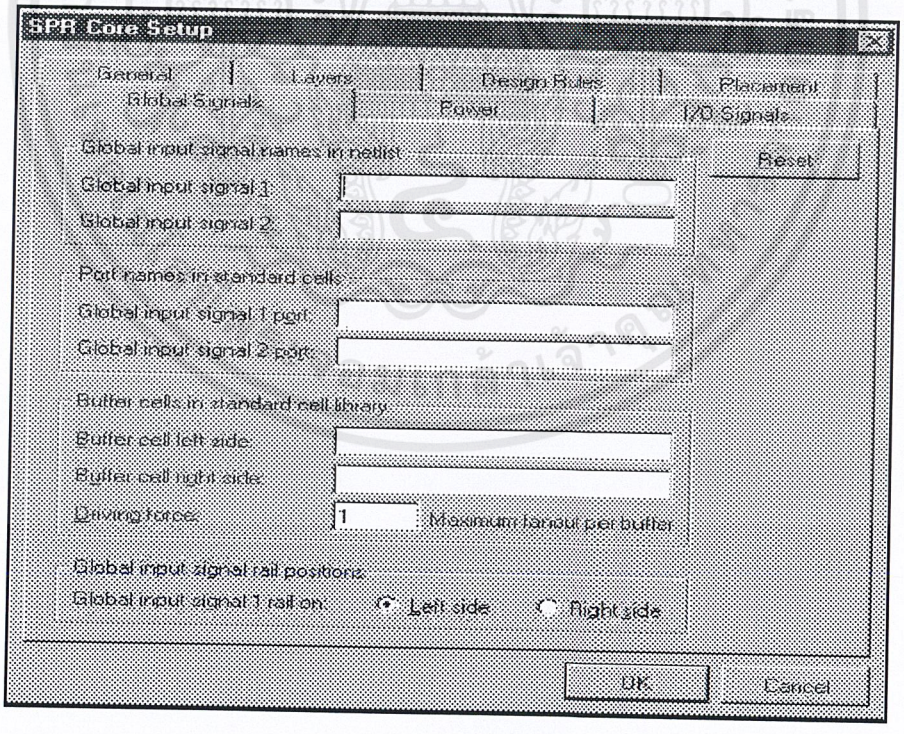
รูปที่ 3.21 การกำหนดค่า Design Rules



รูปที่ 3.22 การกำหนดค่า I/O Signals

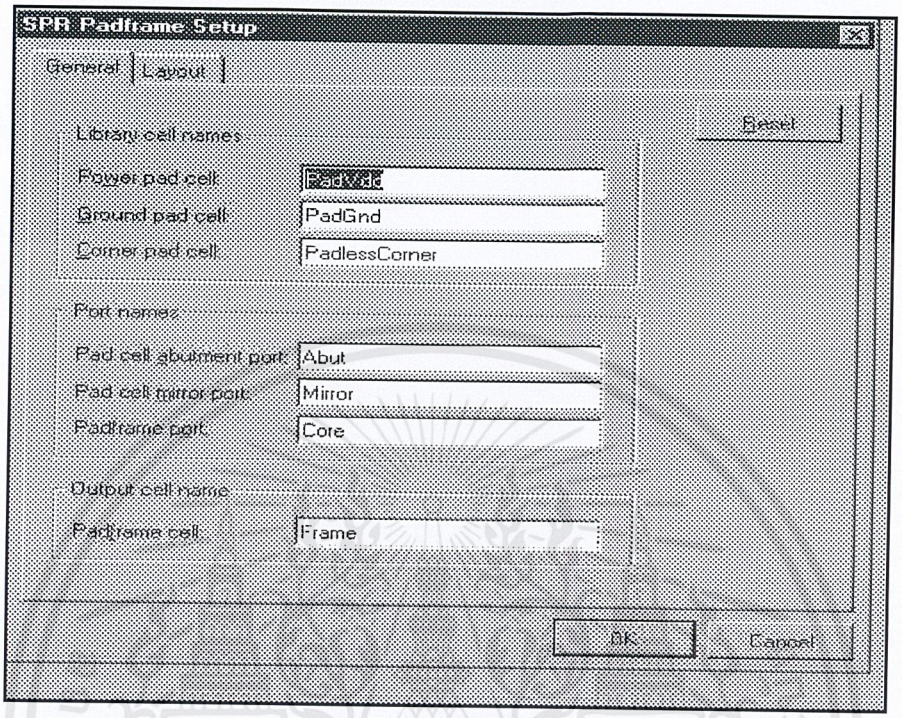


รูปที่ 3.23 การกำหนดค่า Power

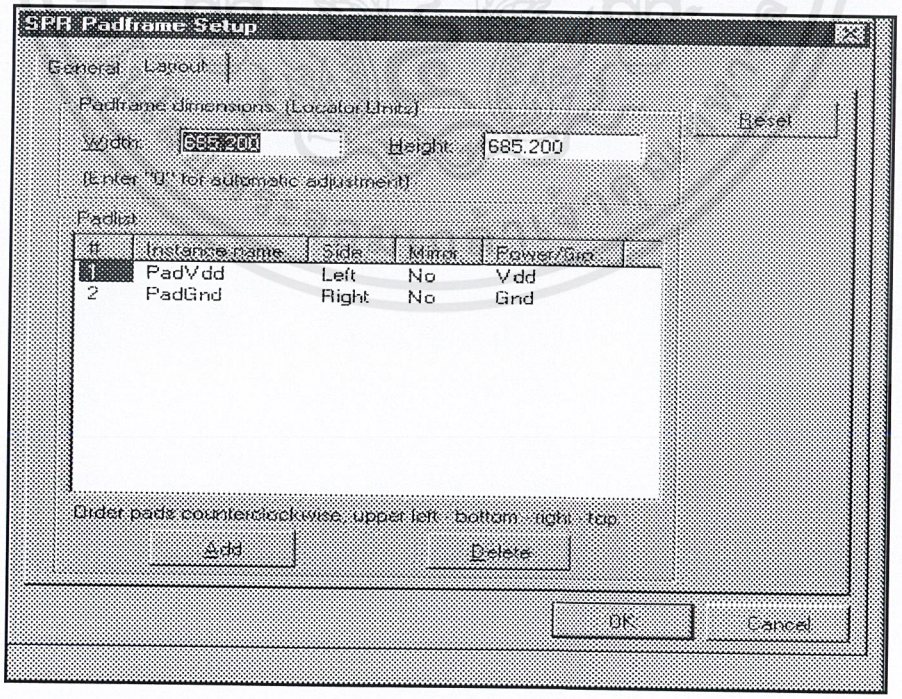


รูปที่ 3.24 การกำหนดค่า Global Signals

4) การกำหนดค่าต่างๆ ใน **SPR Padframe Setup** ดังรูปที่ 3.25 – 3.26

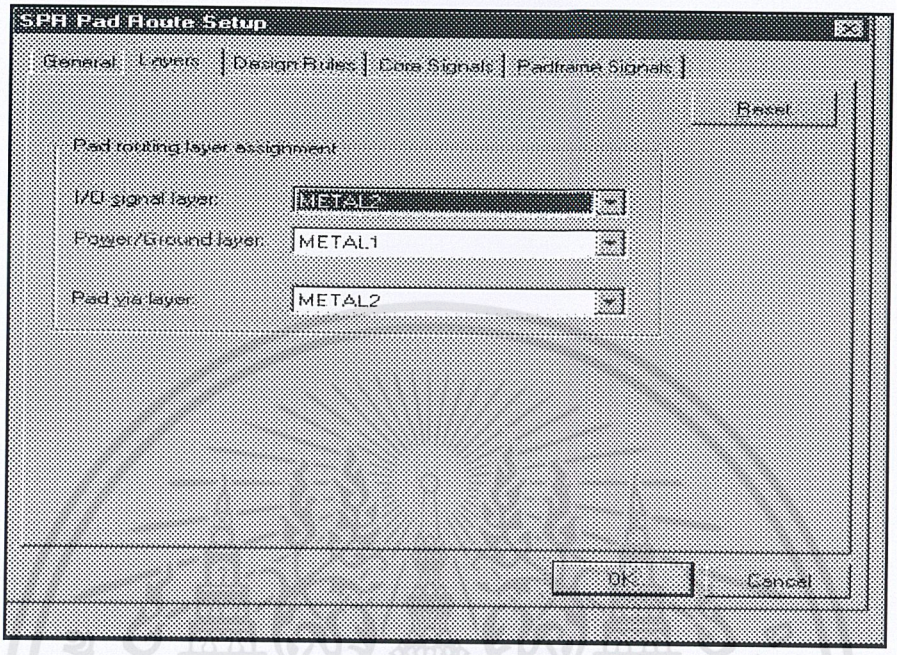


รูปที่ 3.25 การกำหนดค่า General ของ SPR Padframe Setup

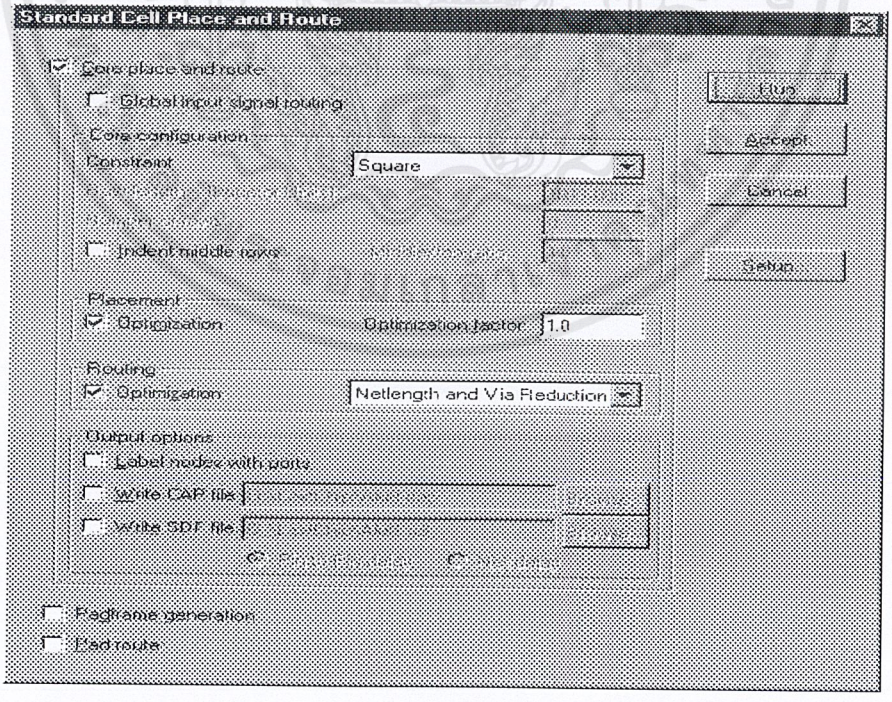


รูปที่ 3.26 การกำหนดค่า Layout ของ SPR Padfrae Setup

5) การกำหนดค่า Layers ของ SPR Pad Route Setup ดังรูปที่ 3.27



รูปที่ 3.27 การกำหนดค่า Layers ของ SPR Pad Route Setup



รูปที่ 3.28 การกำหนดค่าของ Place&Route

6) การกำหนดค่าของ Place&Route เมื่อกำหนดค่าทั้งหมดแล้วก็ให้ กลับไปที่ หน้าต่างของ L-Edit > Tool > SPR > Place&Route ดังรูปที่ 3.28 ในส่วนช่อง Core configuration ในส่วนค่า Constraint นั้นเราสามารถเลือกได้ เพราะเป็นการเลือกว่าจะให้ Place & Route แบบ Square, Width , Height, Width and Height และที่ช่องของ Routing เป็นการกำหนดค่าที่ดีที่สุดเพื่อลด Via และ Net เมื่อทำทุกขั้นตอนเสร็จแล้วให้คลิกที่ RUN ถ้าผลถูกต้องก็จะได้รูปของ Layout ออกมา

3.5.3 ขั้นตอนการออกแบบและการสร้าง Layout

ขั้นตอนการออกแบบและการสร้าง Layout ในแต่ละส่วน

1) สร้าง Standard Cell สำหรับ Place&Route จาก Black Box GDSII File ที่อยู่ใน Design Kit ของโรงงานด้วยโปรแกรม L-Edit

2) นำ Netlist ของวงจรแต่ละส่วนที่ได้มาทำ Standard Cell Place&Route

3) การ Extract Layout ของ Block ที่ได้จากการ Place&Route

4) การตรวจสอบความถูกต้องของ Layout ที่ได้จากการทำ DRC

5) สร้าง Standard Cell สำหรับ Place&Routeจาก Black Box GDSII File ที่อยู่ใน Design Kit ของโรงงานด้วยโปรแกรม L-Edit

6) นำ Netlist ของวงจรแต่ละส่วนที่ได้มาทำ Standard Cell Place&Route

7) การ Extract Layout ของBlock ที่ได้จากการ Place&Route

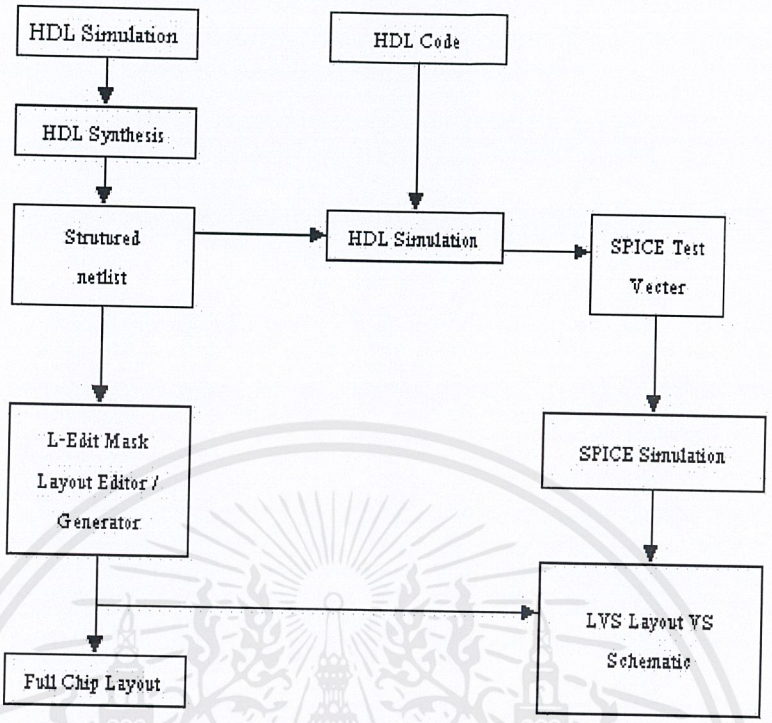
8) การตรวจสอบความถูกต้องของ Layout ที่ได้จากการทำ DRC

9) การสร้าง Padframe และทำ Block Place&Route(BPR)

10) การ Extract Layout เพื่อทำการตรวจสอบหลังหารทำ Block Routing

11) การตรวจสอบความถูกต้องของ Layout ด้วยการ DRC&ERC

12) ขั้นตอนการทำงานตั้งแต่การรับ Netlist ของไฟล์ที่มีนามสกุล VHD แล้วแปลงให้เป็น File ที่มีนามสกุล EDF นั้น เพื่อดำเนินการ Place&Route สร้าง Mask Layout โดยใช้ L-Edit

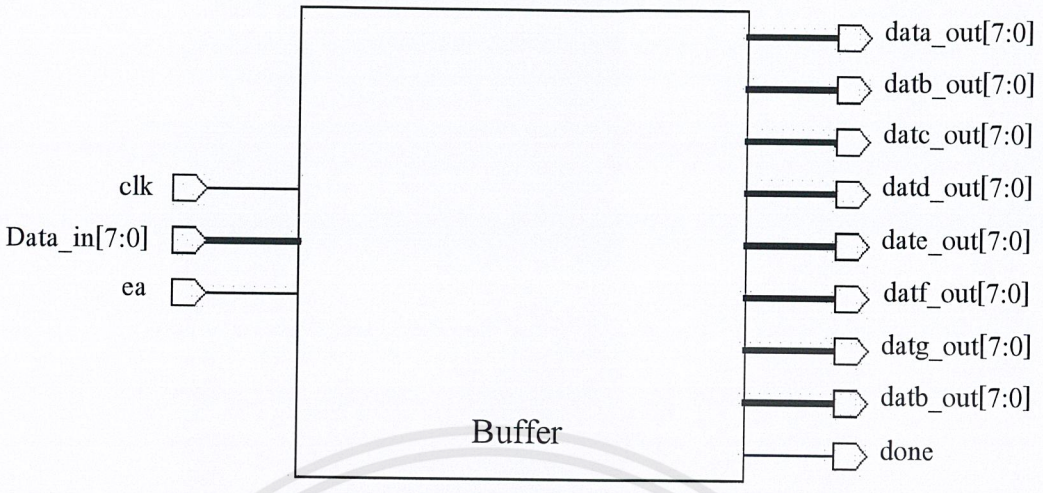


รูปที่ 3.29 ขั้นตอนการดำเนินการทำ Mask Layout

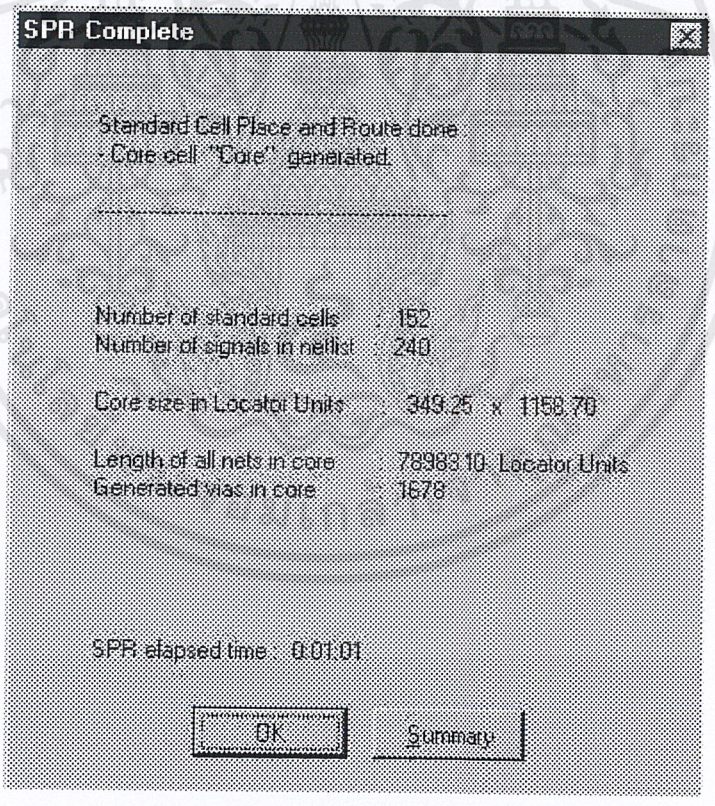
3.6 ส่วนประกอบต่างๆ ของตัวเข้ารหัสและถอดรหัส DCT

3.6.1 ภาค Buffer

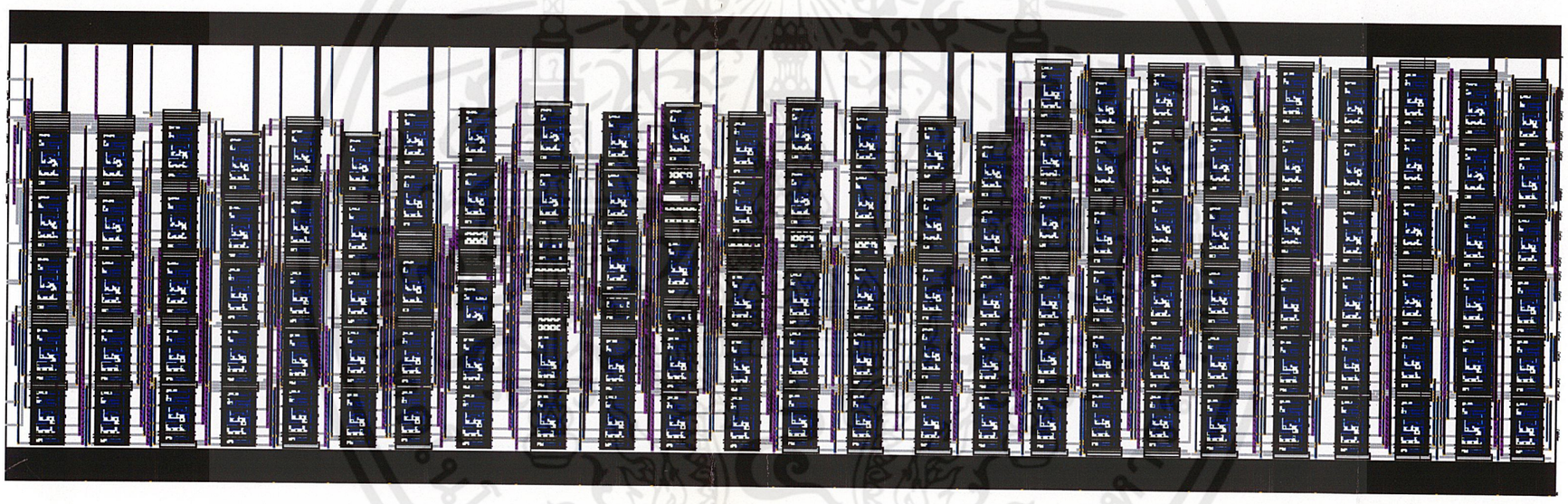
เป็นตัวรับสัญญาณดิจิตอลอินพุต(อาจผ่านมาจากวงจรA/Dภายนอก)ที่เข้ามาแบบอนุกรมกัน (Serial) เข้าสู่ภาค Buffer ที่ขา data_in โดยมีหลักการทำงานคือ จะรับสัญญาณดิจิตอลเข้ามาครั้งละ 8 Bit จะนับเป็น 1 ชุด (เมตริกส์ 1×8) เมื่อครบ 1 ชุดแล้วภาค Buffer ก็จะส่งสัญญาณออกไปที่ขา done เพื่อส่งไปควบคุมภาคต่อไปและสัญญาณที่เอาท์พุตที่ขา data_out0 ถึง data_out7 ซึ่งส่งออกแบบขนานกัน (Parallel) (เมตริกส์ 8×1) ส่วนขา ea เป็นส่วนที่ควบคุมการทำงานของภาค Buffer ทั้งหมด และสัญญาณนาฬิกาเข้ามาที่ขา clk แผนผังการทำงานของ Buffer ดังรูปที่ 3.30 ผลการทำ SPR ของ Buffer ดังรูปที่ 3.31 และเมื่อทำการMask Layout แล้วจะได้ Layout ดังรูปที่ 3.32



รูปที่ 3.30 แผนผังการทำงานของ Buffer



รูปที่ 3.31 ผลการทำ SPR ของ Buffer

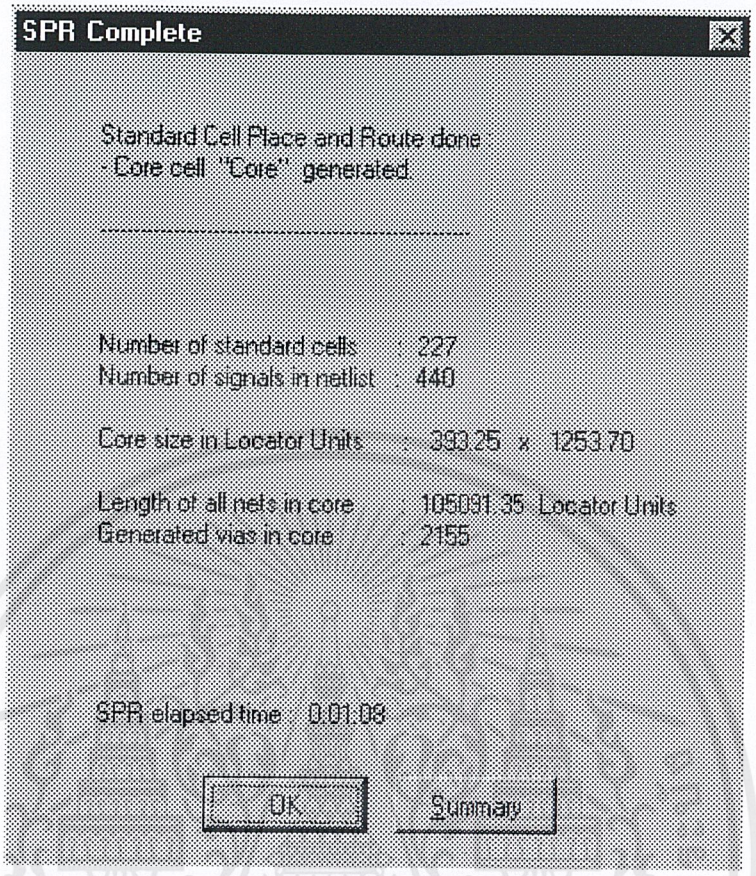


รูปที่3.32 Layout ของ Buffer

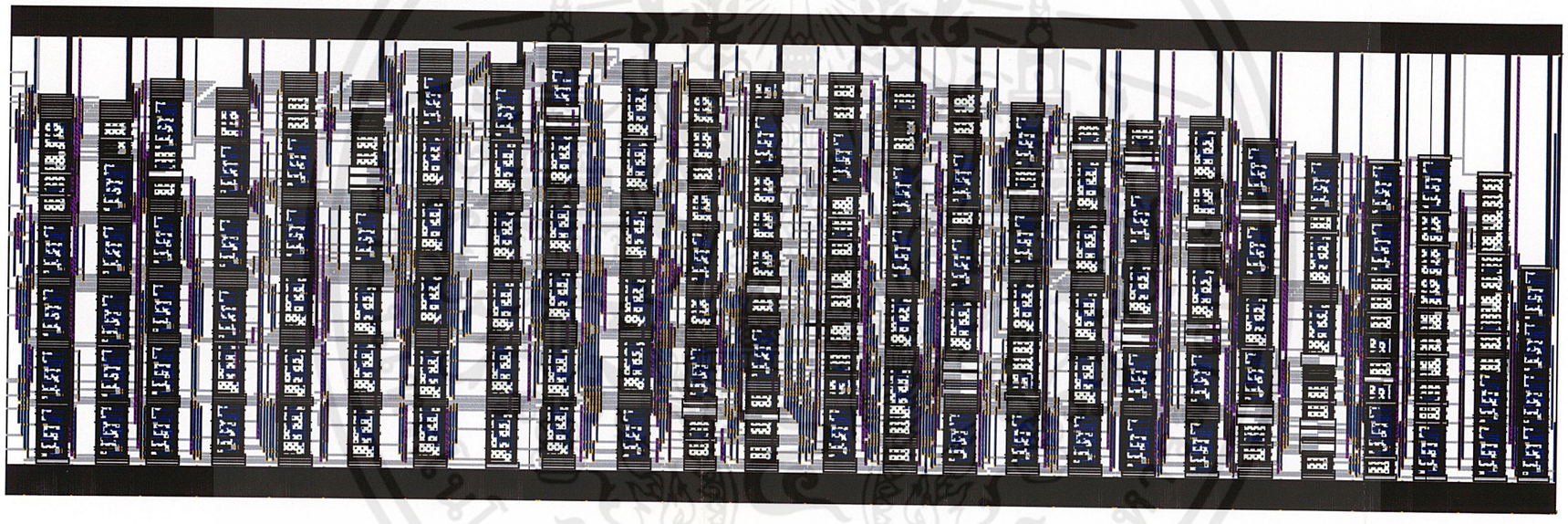
3.6.2 ภาค DCT

ส่วนของภาค DCT จะรับข้อมูลเข้ามาเป็นแบบขนานที่ขา data ถึง dath ผ่านกระบวนการของ DCT แล้วออกมาเป็นแบบขนานที่ขา dcta ถึง dcth โดยมีขา done_in ทำหน้าที่รับคำสั่งควบคุมจังหวะในการทำงานภายใน ส่วนที่ขา ea เป็นส่วนที่ควบคุมการทำงานของภาค DCT ทั้งหมด โดยแผนผังการทำงานของ DCT ดังรูปที่ 3.33 ผลการทำ SPR ของ Buffer ดังรูปที่ 3.34 และเมื่อมาทำกระบวนการสร้าง Mask Layout แล้วจะได้ Layout ดังรูปที่ 3.35

รูปที่ 3.33 แผนผังการทำงานของ DCT



รูปที่ **3.34** ผลการทำ SPR ของ DCT

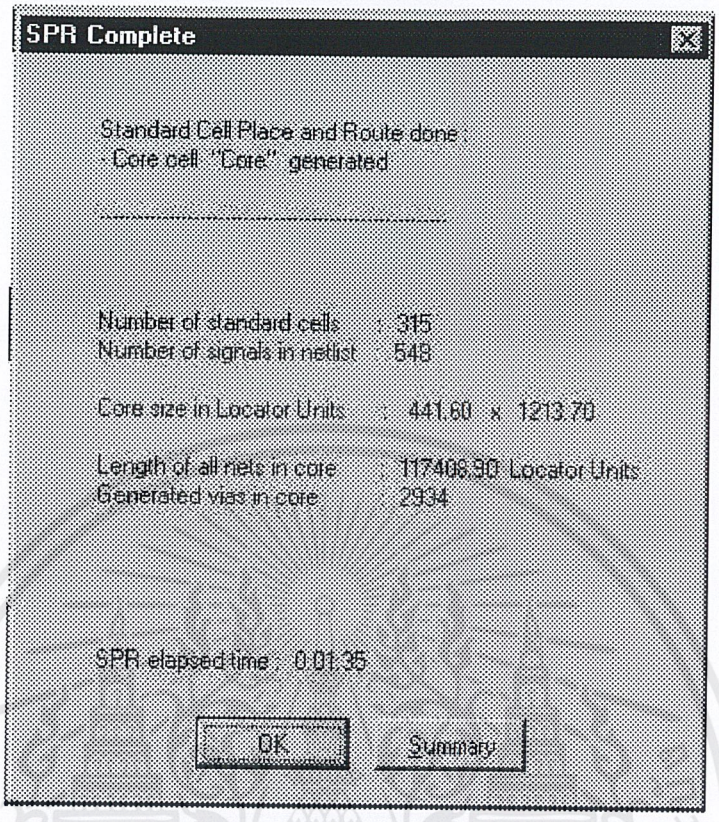


รูปที่3.35 Layout ของ DCT

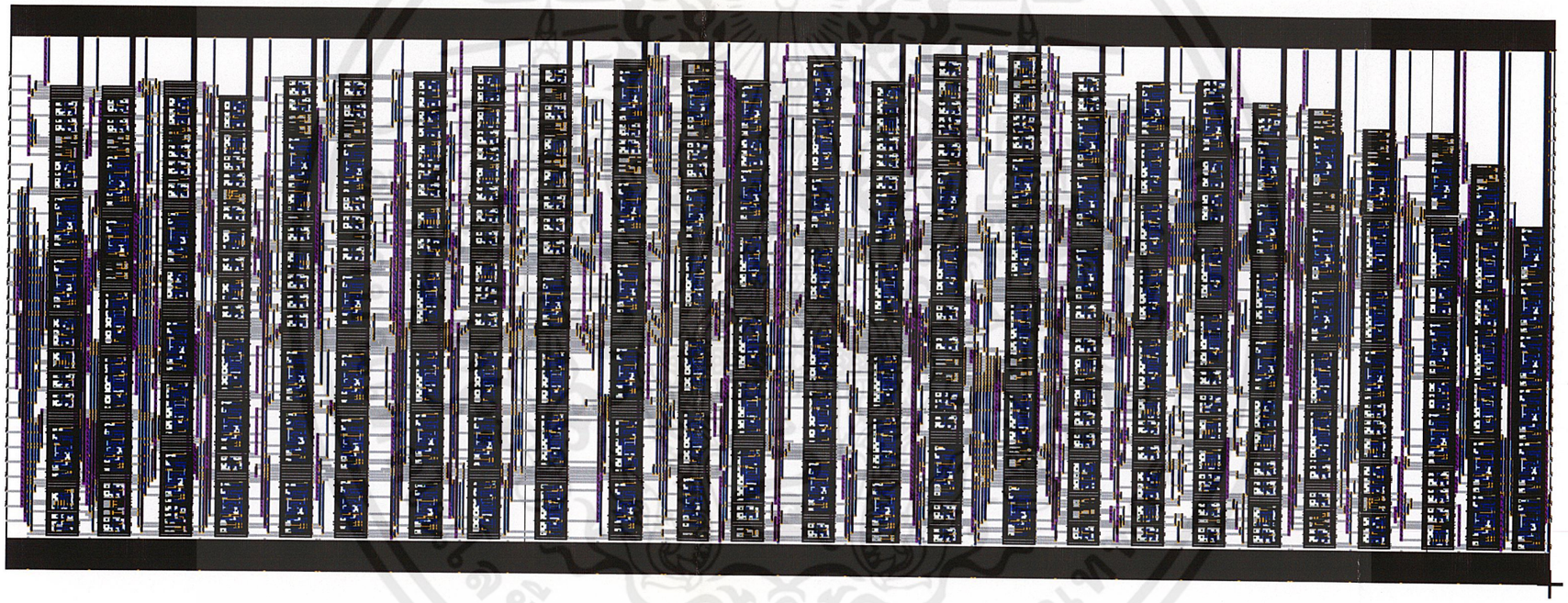
3.6.3 ภาค IDCT

ส่วนของภาค IDCT จะรับข้อมูลเข้ามาเป็นแบบขนานที่ขา dcta ถึง dcth ผ่านกระบวนการของ IDCT แล้วออกมาเป็นแบบขนานที่ขา data ถึง dath โดยมีขา done_in ที่ทำหน้าที่รับคำสั่งการควบคุมจังหวะการทำงานภายในเพื่อให้สัมพันธ์กับการทำงานในภาคอื่นๆ ส่วนที่ขา ea เป็นส่วนที่ควบคุมการทำงานของภาค IDCT ทั้งหมด แผนผังการทำงานของ IDCT ดังรูปที่ 3.36 ผลของการทำ SPR ของ IDCT ดังรูปที่ 3.37 และเมื่อมาทำกระบวนการสร้าง Mask Layout แล้วจะได้ Layout ดังรูปที่ 3.38

รูปที่ 3.36 แผนผังการทำงานของ IDCT



รูปที่ 3.37 ผลการทำ SPR ของ IDCT

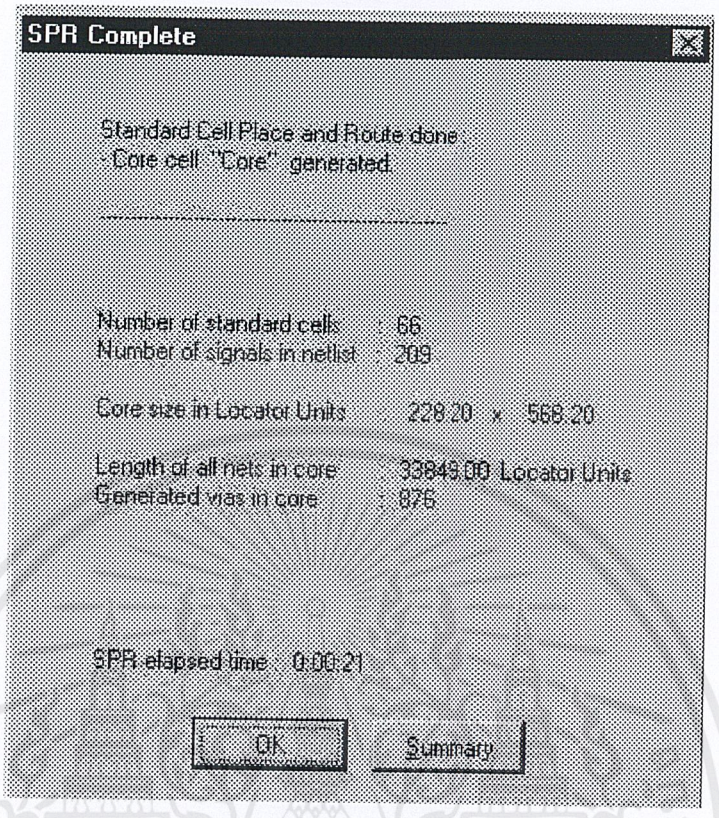


รูปที่3.38 Layout ของ IDCT

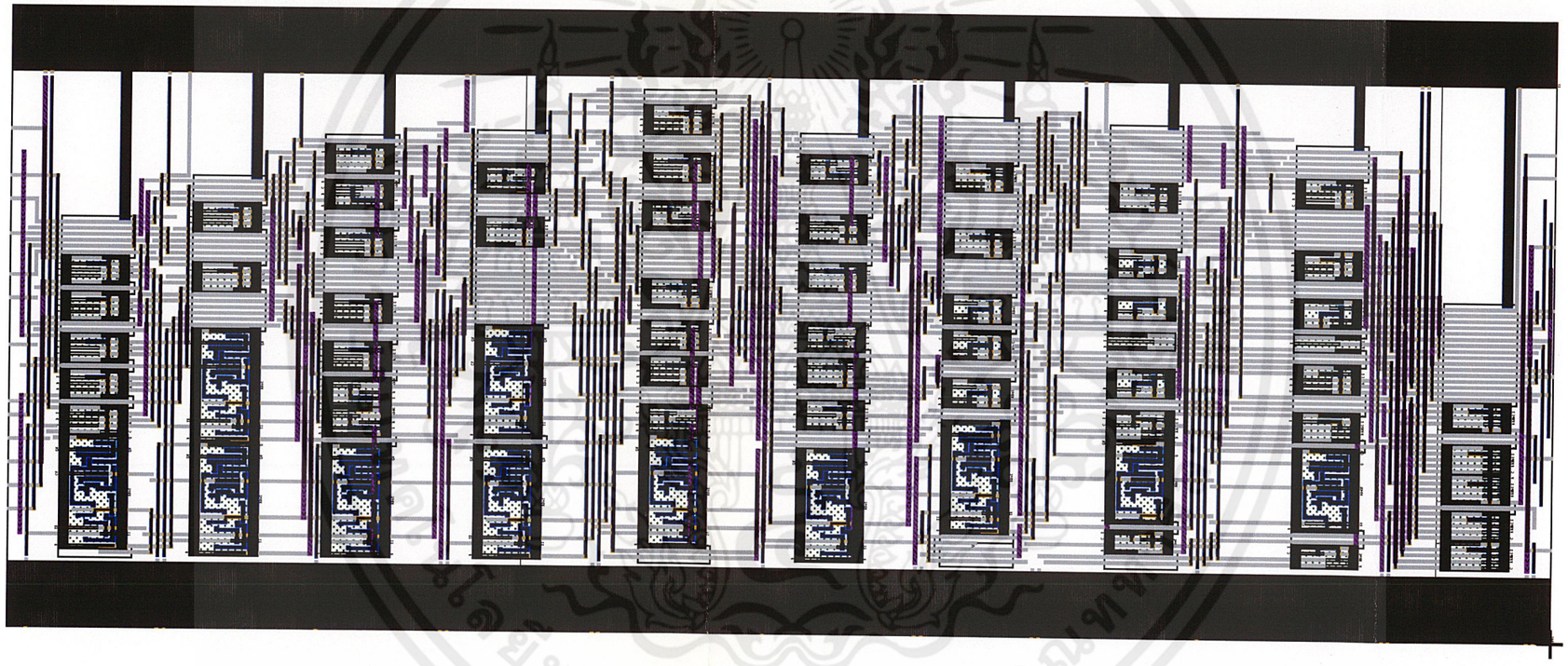
3.6.4 ภาค PISO (Parallel In Serial Out)

เป็นตัวรับสัญญาณดิจิตอลอินพุตที่เข้ามาแบบขนาน (Parallel) เข้าสู่ภาค PISO ที่ขา p0 ถึง p7 โดยมีหลักการทำงานคือ จะรับสัญญาณดิจิตอลเข้ามาครั้งละ 1 Bit ทั้งรวมทั้ง 8 ขารวมเป็น 1 ชุด (เมตริกซ์ 8x1) เมื่อครบ 1 ชุด แล้วภาค PISO ก็จะส่งสัญญาณออกไปที่ขา s_out จะออกเป็นลักษณะอนุกรม โดยข้อมูลจะออกไปครั้งละ 1 ชุด (เมตริกส์ 1x8) ส่วนขา ca เป็นส่วนที่ควบคุมการทำงานของภาค PISO ทั้งหมด และสัญญาณนาฬิกาเข้ามาที่ขา clk แผนผังการทำงานของ PISO ดังรูปที่ 3.39 ผลการทำ SPR ของ PISO ดังรูปที่ 3.40 และเมื่อมาเข้ากระบวนการสร้าง Mask Layout เมื่อทำการMask Layout แล้วจะได้ Layout ดังรูปที่ 3.41

รูปที่ 3.39 แผนผังการทำงานของ PISO



รูปที่ 3.40 ผลการทำ SPR ของ PISO



รูปที่3.41 Layout ของ PISO

3.7 การสร้าง Padframe

ก่อนการสร้าง Padframe ผู้ออกแบบจำเป็นต้องทราบชนิดและวิธีการใช้งานของ Pad ต่างๆ ของโรงงานเจือสารก่อน ผู้ออกแบบสามารถเลือกลักษณะของ Padframe ได้ตามความต้องการซึ่งจะประกอบด้วย Pad ต่างๆ ดังนี้

3.7.1 Pad

Pad ที่จะใช้กับชิพต้องการแรงดันคงที่ ในบางกรณีที่ชิพต้องการแรงดันที่มีความเที่ยงตรงสูง (Noise ต่ำ) เช่น วงจรแอนาลอก หรือชิพบางตัวที่มีความไวต่อการเปลี่ยนแปลง

1) **VDDCO** ใช้เฉพาะจ่ายแรงดันเข้าไปภายใน

2) **VSSCO** ใช้เฉพาะเป็น Ground ของวงจรภายใน

3.7.2 Pad ชนิดพิเศษ

ได้แก่ Pad ดังต่อไปนี้

1) **VDD** จะรวมไฟเลี้ยงทั้ง 3 รวมกัน

2) **VSS** จะรวม Ground ทั้ง 3 วงจรรวมกัน รวมไปถึง Ground Core และ Substrate Biasing

3) **VDDI1** ต่อมาจากไฟเลี้ยงภายในวงจรที่มีความกว้าง

4) **VSSI1** ต่อมาจาก Ground ภายในวงจรที่มีความกว้าง

5) **VDDI2** ต่อมาจากไฟเลี้ยงภายในวงจรที่มีขนาดความแคบ

6) **VSSI2** ต่อมาจาก Ground ภายในวงจรที่มีขนาดความแคบ

3.7.3 ขั้นตอนการสร้าง Padframe

เมื่อได้ Floorplan ของชิพก็จะทำการเลือก Pad ที่จะใช้งาน โดยเลือก Pad Output ทั้งหมดที่สามารถจ่ายกระแสได้ 8 mA เพื่อง่ายต่อการใช้งาน รายละเอียดของ Pad ที่สามารถเลือกใช้มีดังนี้

1) **B8CR** เป็น Pad Output แบบ CMOS

2) **BD8R** เป็น Pad Input / Output แบบ CMOS

3) **IBUF** เป็น Pad Input

4) **CORNER** เป็น Pad ที่ใช้ต่อมุมของ Pad Frame

5) **VDD** เป็น Pad ที่ใช้ต่อกับ Vdd ภายนอก เลี้ยงวงจรทั้งหมด

6) **VSS** เป็น Pad ที่ใช้ต่อกับ Ground ภายนอก

3.8 การทำ Block Place & Route (BPR)

3.8.1 Initialization

เมื่อทำกำหนดพารามิเตอร์เรียบร้อยแล้ว โปรแกรม L-Edit จะอ่าน Netlist เพื่อทำ Initialize ข้อมูลเพื่อทำ BPR จะได้บล็อกต่างๆ ซึ่งถูกจัดวางที่ขอบด้านล่างของ Pad Frame มี Routing Guide ปรากฏขึ้นเพื่อแสดงการเชื่อมต่อของสายสัญญาณภายในชิพเพื่อใช้เป็นแนวทางในการปรับแต่งบล็อกเพื่อใช้เชื่อมต่อสัญญาณอย่างเหมาะสม ดังรูป ข.1

3.8.2 Block Placement (Floorplan)

Block Placement (Floorplan) คือการวางบล็อกต่างๆ ให้อยู่ในกรอบภายในของ Pad Frame ปรับด้านและตำแหน่งของบล็อกต่างๆ โดยคำนึงถึงพื้นที่การเชื่อมต่อสัญญาณระหว่างบล็อกกับ Pad และปัจจัยที่กำหนดเพื่อให้ได้ Floorplan ที่เหมาะสมกับการทำ Block Placement หรือ Floorplan เกี่ยวข้องกับขั้นตอน SPR ของบล็อกต่างๆ เนื่องจากถ้าไม่สามารถสร้าง Floorplan ที่เหมาะสมจากขนาดของ Block ที่มีอยู่ได้ก็ต้องกลับไปทำขั้นตอน SPR เพื่อปรับขนาดของบล็อกใหม่ ซึ่งจำนวนการทำซ้ำในแต่ละขั้นตอนการออกแบบวงจรจะขึ้นอยู่กับเงื่อนไขของพื้นที่ของชิพว่าถูกจำกัดไว้ที่ขนาดเท่าไหร่และอาจต้องย้อนกลับไปถึงขั้นตอนการออกแบบวงจรถ้าไม่สามารถปรับให้เป็นไปตามต้องการแสดงดังรูปที่ ข.2

3.8.3 Block Routing

หลังจากการ Floorplan จะเป็นขั้นตอนของสัญญาณต่างๆ เข้าด้วยกันโดยพิจารณาจากประเภทและคุณสมบัติของสัญญาณแต่ละเส้นเพื่อใช้เป็นข้อมูลในการกำหนดชนิดของเลเยอร์รวมทั้งขนาดของสัญญาณ เนื่องจากคุณสมบัติทางไฟฟ้าที่แตกต่างกันเช่น ค่าของ Capacitance และ Resistance ที่มีผลกับ Delay ของสัญญาณทุกเส้น Maximum Current Density ซึ่งเป็นปัจจัยที่กำหนดขนาดสัญญาณของ Power เป็นต้น ในการเชื่อมต่อสัญญาณ (Routing) สามารถทำได้ทั้งแบบอัตโนมัติโดยใช้ Automatic Routing ซึ่งใช้กับสัญญาณที่ไม่มีเงื่อนไขในการบังคับมากนักและสามารถทำเองได้ โดยใช้ Assist Manual Route กับสัญญาณที่สำคัญ เช่น Clock, Bus และ Power

ขั้นตอน Block Routing จะมีความสำคัญมากกับชิพที่มีเงื่อนไขของเวลามาทำการบังคับมากๆ เช่น ชิพที่มีความเร็วในการทำงานสูงและยิ่งสำคัญมากขึ้นเมื่อใช้เทคโนโลยี การผลิตที่เล็กลง ซึ่งการทำซ้ำในขั้นตอน SPR อาจจะต้องทำรวมมาถึงขั้นตอนนี้ด้วยถ้าไม่สามารถเป็นไปตามเงื่อนไขของเวลาได้

บทที่ 4
การทดลองและผลการทดลอง

4.1 กล่าวนำ

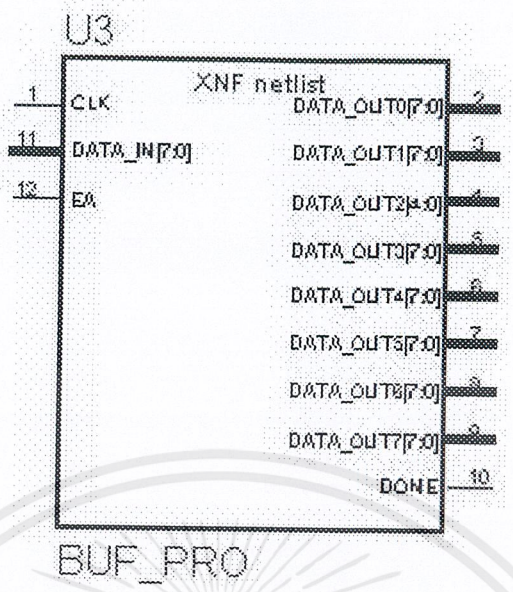
กระบวนการออกแบบและการสร้างวงจรรวมเฉพาะกิจนั้นได้สร้างขึ้นโดยใช้หลักการของการแปลงสัญญาณ 1D-DCT เพื่อใช้ในการเข้ารหัสและถอดรหัสสัญญาณ ตามหลักการและทฤษฎีในบทที่ 1 – 3 โดยการใช้ภาษา VHDL และโปรแกรม ช่วยในการออกแบบจากลักษณะการออกแบบด้วยภาษา VHDL นั้นสามารถจำลองการทำงานของวงจรได้โดยการใช้โปรแกรมเลียนแบบการทำงานสัญญาณต่างๆ โดยใช้โปรแกรม Logic Simulator ของ Xilinx Foundation เป็นการทำงานในส่วนของ Extract Layout

4.2 การจำลองการทำงานของตัวเข้ารหัสและถอดรหัส DCT

จากการออกแบบการสร้างวงจรรวมเฉพาะงานตัวเข้ารหัสและถอดรหัส DCT ได้แบ่งการทำงานออกเป็นบล็อกต่างๆ ดังนี้ คือ Buffer, DCT, IDCT, และ PISO โดยการทำงานและผลการเลียนแบบของแต่ละส่วนจะมีการทำงานที่แตกต่างกันไปขึ้นอยู่กับการใช้งาน

4.2.1 ผลการจำลองการทำงานของสัญญาณส่วนของ Buffer

การเลียนแบบการทำงานส่วนของ Buffer โดยใช้โปรแกรม Logic Simulator ของ Xilinx Foundation จากรูปที่ 4.1



รูปที่ 4.1 แผนผังการทำงานของ Buffer

การจำลองการทำงานในส่วนนี้สามารถทำการป้อนสัญญาณอินพุตชนิดลอจิกขนาด 8 บิต เป็นการนับขึ้นให้กับส่วนของ Buffer การส่วนของโปรแกรม จะทำการเรียงข้อมูลอินพุตให้จนกว่าจะครบ 8 ข้อมูลหรือ 1 ชุด จึงจะทำการส่งข้อมูลออกมาสู่วงจรในภาคต่อไปในการรับข้อมูลเข้ามานั้นรับเข้ามาแบบอนุกรมและจะส่งข้อมูลเดิมออกไปในแบบขนาน และเมื่อมีการรับข้อมูลออกมาจนครบ 1 ชุดแล้วจะส่งค่า ลอจิก “1” ไปยังขา DONE เพื่อควบคุมการทำงานของภาคต่อไป

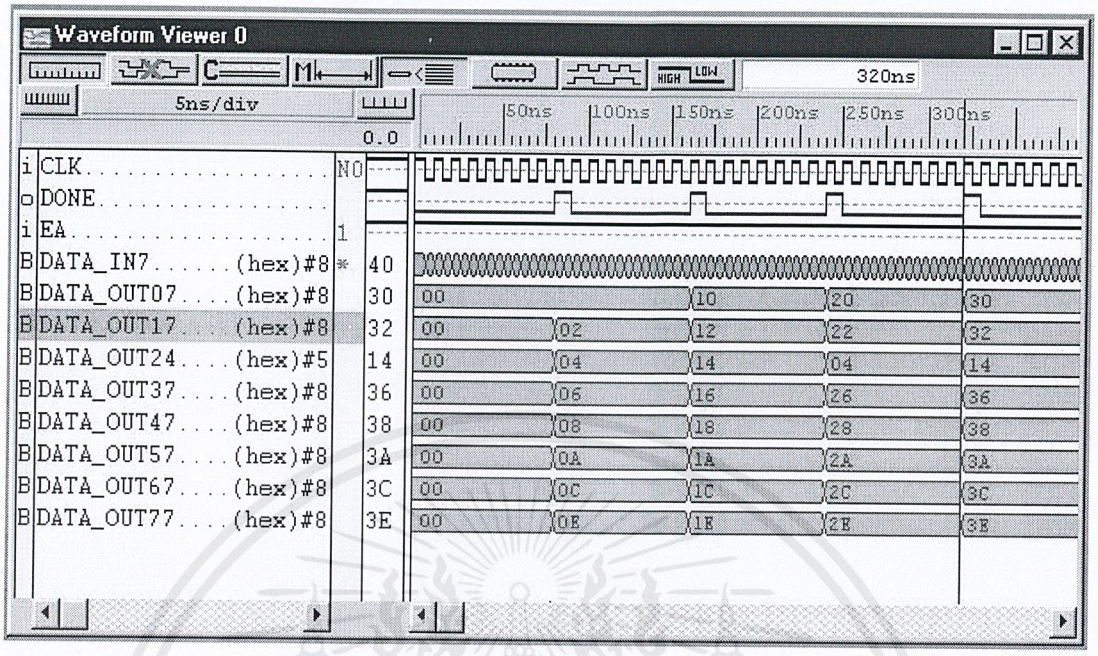
การจำลองการทำงานของวงจรสามารถทำการป้อนสัญญาณอินพุตให้กับวงจรโดยสามารถจำลองสัญญาณได้ดังตารางที่ 4.1

ตารางที่ 4.1 การป้อนสัญญาณจำลองการทำงานของโปรแกรม Buffer

สัญญาณอินพุต	
ชื่อขาสัญญาณ	สัญญาณที่ป้อน
CLK	Nbe0
EA	1
DATA_IN7	Be1
DATA_IN6	Be2
DATA_IN5	Be3
DATA_IN4	Be4
DATA_IN3	Be5
DATA_IN2	Be6
DATA_IN1	Be7
DATA_IN0	Be8

ผลการทดลอง

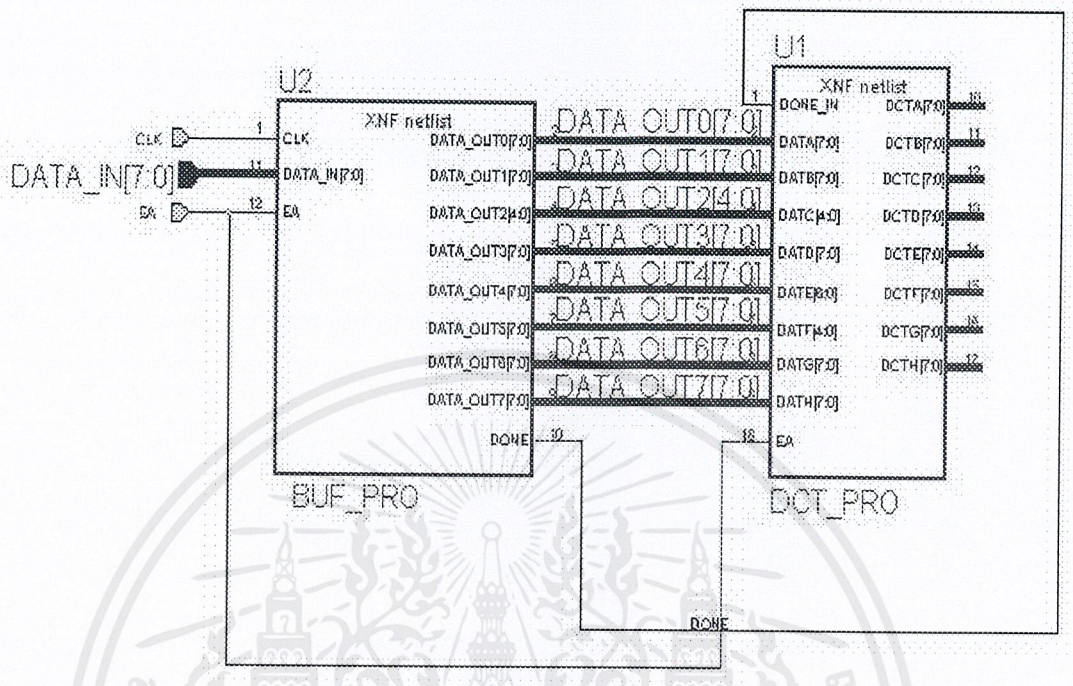
จากผลการจำลองการทำงานของวงจร โปรแกรมสามารถเลียนแบบการทำงานของวงจร Buffer ได้จริง คือข้อมูลที่เข้ามาเป็นแบบอนุกรมผลที่ได้เอาต์พุตออกมาเป็นข้อมูลแบบขนาดและผลที่ได้ข้อมูลอินพุตและเอาต์พุตมีค่าเป็นค่าเดิม ดังผลการทดลองรูปที่ 4.2



รูปที่ 4.2 ผลการจำลองการทำงานของภาค Buffer

4.2.2 ผลการจำลองการทำงาน ของสัญญาณส่วนของ DCT

ผลที่ได้จากการจำลองการทำงานของวงจร DCT นั้นที่ทำการเชื่อมต่อการทำงานเข้ากับส่วนของ Buffer เพื่อให้ง่ายต่อการป้อนสัญญาณทางด้านอินพุต ดังรูปที่ 4.3



รูปที่ 4.3 แผนผังการทำงานของ DCT

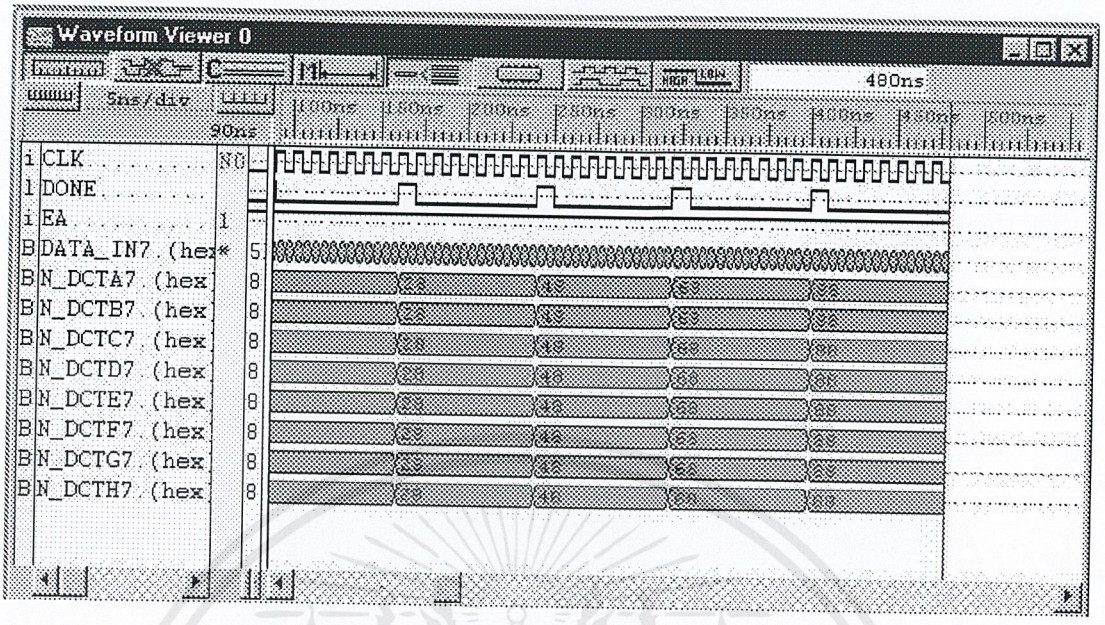
การทำงานของวงจร DCT นั้นสามารถจำลองการทำงานของวงจรได้จริง คือ เมื่อมีการป้อนสัญญาณที่อินพุตของ Buffer ครบทั้ง 8 บิต แล้วขา DONE ของ Buffer จะถูกกำหนดให้เป็น “1” เพื่อป้อนสู่ขา DONE_IN ของ DCT เมื่อขา DONE_IN ได้รับสัญญาณจากอินพุตที่เข้ามาส่วนของ DCT จะแปลงสัญญาณให้เป็นตามที่ผู้ออกแบบได้ออกแบบโดยมีสัญญาณเป็นอินพุต ดังตารางที่ 4.2

ตารางที่ 4.2 การป้อนสัญญาณจำลองการทำงานของโปรแกรม DCT

สัญญาณอินพุต	
ชื่อขาสัญญาณ	สัญญาณที่ป้อน
CLK	Nbe0
EA	1
DATA_IN7	Be1
DATA_IN6	Be2
DATA_IN5	Be3
DATA_IN4	Be4
DATA_IN3	Be5
DATA_IN2	Be6
DATA_IN1	Be7
DATA_IN0	Be8

ผลการทดลอง

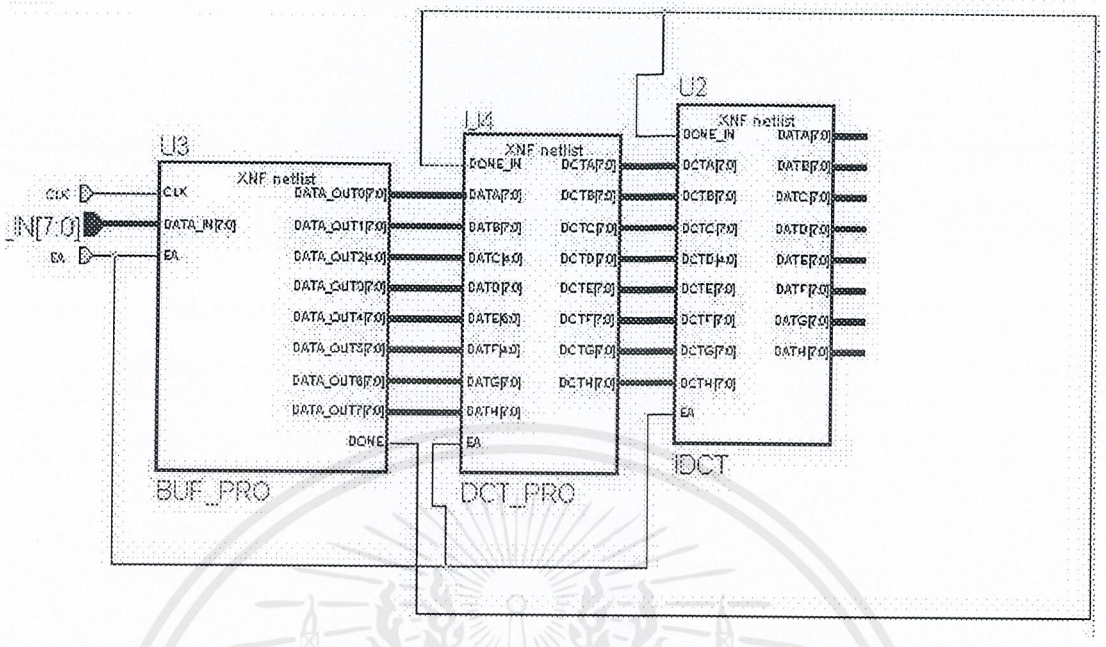
เนื่องจากการป้อนโปรแกรมเป็นแบบการทำงานเชื่อมต่อระหว่างการทำงานของภาค Buffer และ DCT การป้อนสัญญาณจึงเป็นการป้อนเข้าที่อินพุตของภาค Buffer ทางด้าน เอาต์พุตจะได้ออกมาทางภาคของ DCT ผลที่ได้จากการทำงานของส่วน DCT จะได้การจำลองการทำงานโดยการใช้โปรแกรม Logic Simulator ของ Xilinx Foundation เป็นโปรแกรมที่ช่วยในการจำลองการทำงานของวงจรรวม เฉพาะงาน ดังรูปที่ 4.4



รูปที่ 4.4 ผลการจำลองการทำงานโปรแกรม DCT

4.2.3 ผลการจำลองการทำงานของสัญญาณส่วนของ IDCT

ผลที่ได้จากการจำลองการทำงานของวงจร IDCT นั้นที่ทำการเชื่อมต่อการทำงานเข้ากับส่วนของ Buffer เพื่อให้ง่ายต่อการป้อนสัญญาณทางด้านอินพุต ดังรูปที่ 4.5



รูปที่ 4.5 แผนผังการทำงานของ IDCT

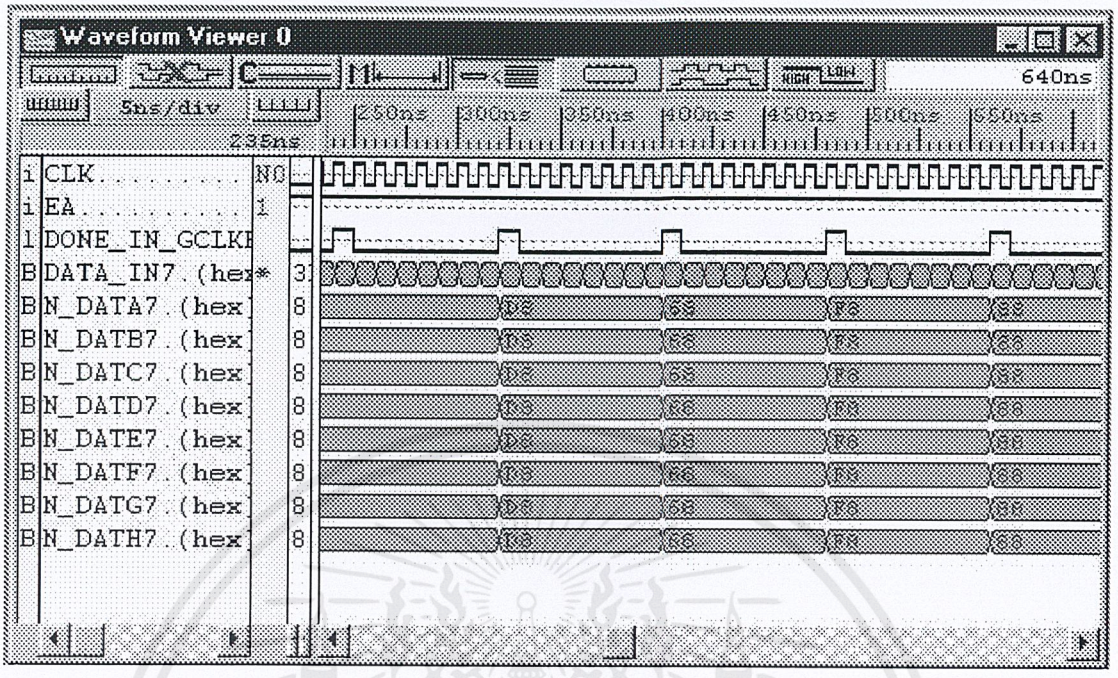
การทำงานของวงจร IDCT คือ เมื่อมีการป้อนสัญญาณที่อินพุตของ Buffer ครบทั้ง 8 บิต แล้วขา DONE ของ Buffer จะถูกกำหนดให้เป็น “1” เพื่อป้อนสู่ขา DONE_IN ของ IDCT เมื่อขา DONE_IN ได้รับสัญญาณจากอินพุตที่เข้ามา ส่วนของ IDCT จะทำการแปลงสัญญาณให้เป็นตามที่ได้ทำการออกแบบโดยมีสัญญาณเป็นอินพุต ดังตารางที่ 4.2

ตารางที่ 4.3 การป้อนสัญญาณจำลองการทำงานของโปรแกรม IDCT

สัญญาณอินพุต	
ชื่อขาสัญญาณ	สัญญาณที่ป้อน
CLK	Nbe0
EA	1
DATA_IN7	Be1
DATA_IN6	Be2
DATA_IN5	Be3
DATA_IN4	Be4
DATA_IN3	Be5
DATA_IN2	Be6
DATA_IN1	Be7
DATA_IN0	Be8

ผลการทดลอง

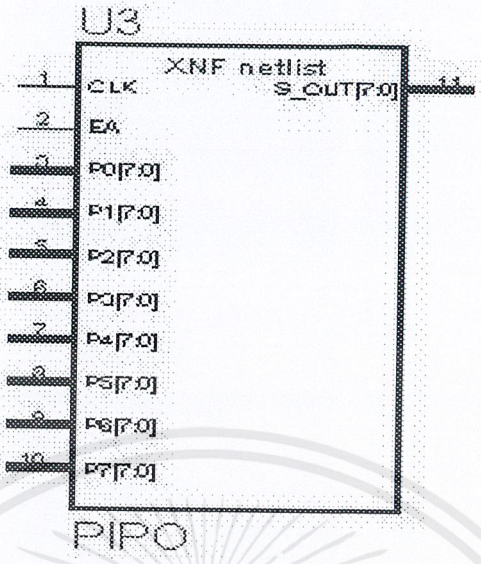
เนื่องจากการป้อนโปรแกรมเป็นแบบการทำงานเชื่อมต่อระหว่างการทำงานของภาค Buffer และ IDCT การป้อนสัญญาณจึงเป็นการป้อนเข้าที่อินพุตของ Buffer ทางด้าน เอาต์พุตจะได้ออกมาทางภาคของ IDCT ผลที่ได้จากการทำงานของส่วน DCT ทางด้านเอาต์พุต DCT จะเป็นส่วนของ อินพุต ของ IDCT เนื่องจากต้องมีการเปรียบเทียบสัญญาณทางด้านอินพุตและเอาต์พุตของทั้งสองตัว จะได้การจำลองการทำงานโดยการใช้โปรแกรม Logic Simulator ของ Xilinx Foundation เป็นโปรแกรมที่ช่วยในการจำลองการทำงานของวงจรรวมเฉพาะงาน ดังรูปที่ 4.4



รูปที่ 4.6 ผลการจำลองการทำงานโปรแกรม IDCT

4.2.4 ผลการจำลองการทำงานของสัญญาณส่วนของ PISO

การทำงานส่วนของวงจร PISO นั้นเป็นการรับผลของสัญญาณที่ได้จากการจำลองการทำงานส่วนของวงจร IDCT ซึ่งผลของสัญญาณที่ได้นั้นผลที่ได้เป็นสัญญาณที่อยู่ในรูปของสัญญาณแบบขนาดเนื่องจากวงจรการทำงานในภาคอื่นๆ นั้นจำเป็นต้องมีการรับข้อมูลแบบอนุกรม ดังนั้นในส่วนของวงจร PISO นั้นเป็นการแปลงข้อมูลหรือการจัดเรียงข้อมูลให้อยู่ในรูปของสัญญาณแบบอนุกรมเพื่อส่งไปยังวงจรการทำงานในส่วนอื่นๆ ต่อไป วงจรการทำงานในส่วนของ PISO ได้ดังรูปที่ 4.7



รูปที่ 4.7 แผนผังการทำงานของ PISO

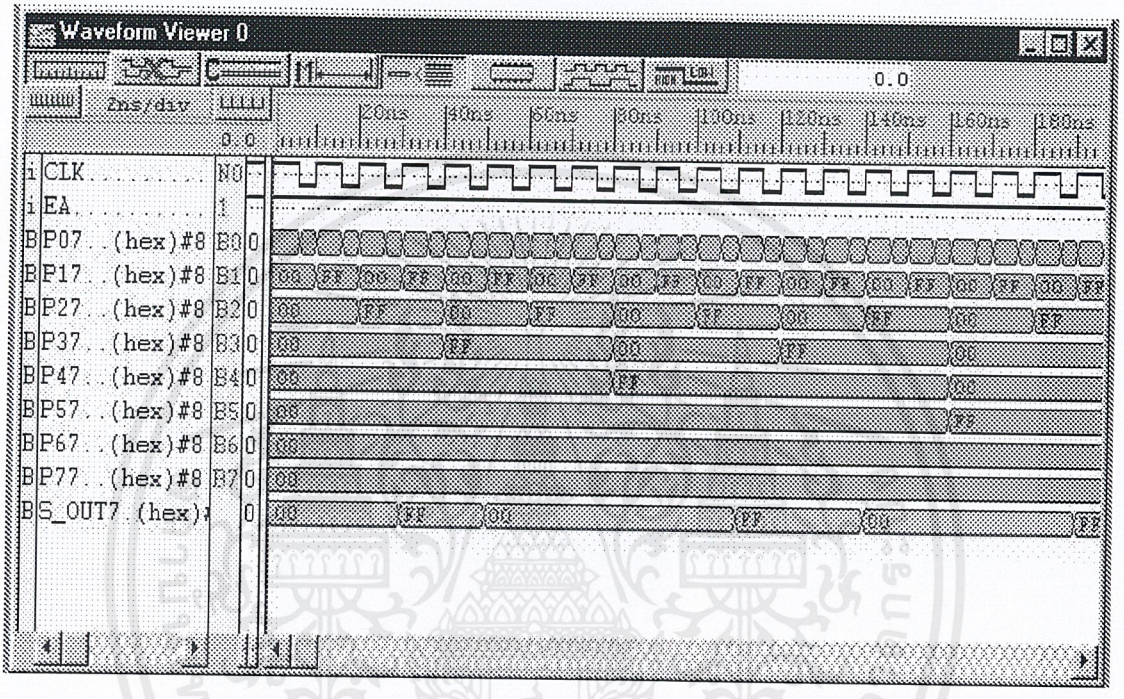
จากแผนผังการทำงานของวงจร PISO สามารถป้อนสัญญาณอินพุตในการจำลองการทำงานได้ตามตารางที่ 4.4 ดังนี้

ตารางที่ 4.4 การป้อนสัญญาณจำลองการทำงานของโปรแกรม PISO

สัญญาณอินพุต	
ชื่อขาสัญญาณ	สัญญาณที่ป้อน
CLK	Nbe0
EA	1
P0	Be1
P1	Be2
P2	Be3
P3	Be4
P4	Be5
P5	Be6
P6	Be7
P7	Be8

ผลการทดลอง

เมื่อมีการจำลองสัญญาณที่เข้ามาทางด้านอินพุตของส่วนของวงจร PISO เป็นรูปสัญญาณแบบขนานนั้นจะเห็นได้ว่าผลการจำลองการทำงานเอาท์พุตที่ออกมาจะส่งสัญญาณออกมาในรูปแบบของสัญญาณอนุกรมตามที่ได้ออกแบบมาได้ดังรูปผลการจำลองการทำงานที่ 4.8



รูปที่ 4.8 ผลการจำลองการทำงานโปรแกรม PISO

บทที่ 5
บทสรุป แนวทางการแก้ไขและพัฒนา

5.1 บทสรุป

การจัดทำโครงงานนี้เป็นการสร้าง Layout ต้นแบบของวงจรเข้ารหัสและถอดรหัส DCT โดยใช้กระบวนการ ASIC Design ที่ใช้ Standard Cell การออกแบบเริ่มจากการใช้ภาษา VHDL เป็นตัวบรรยายพฤติกรรมทางด้านฮาร์ดแวร์ของวงจรเพื่อนำไปสังเคราะห์วงจรโดยใช้โปรแกรม Leonardo Spectrum เป็นตัวสังเคราะห์วงจรออกมา เมื่อได้ File.Edif จากการสังเคราะห์แล้ว สามารถนำไปสร้าง Mask Layout โดยกระบวนการของ Tanner tools โดยโปรแกรม L-Edit เพื่อให้ได้ Layout ต้นแบบของวงจรเข้ารหัสและถอดรหัสของ DCT ซึ่งจะประกอบด้วยบล็อกย่อยของตัววงจร Buffer, DCT, IDCT และ PISO เป็นชิพวงจรรวมเฉพาะงานและสามารถจำลองการทำงานได้ในระดับ VHDL

5.2 ปัญหาที่เกิดขึ้นในการจัดทำโครงงาน

1. เนื่องจากข้อมูลในส่วนของกระบวนการ ASIC Design นี้เป็นยังเป็นข้อมูลที่ใหม่ อีกทั้งมีผู้รู้ในส่วนของกระบวนการนี้น้อยมาก ทำให้ต้องใช้เวลาศึกษาและความพยายามในการแก้ปัญหาต่างๆ ที่เกิดขึ้นในกระบวนการทำงาน

2. เนื่องจากกระบวนการทำ SPR และ BPR ยังเกิดปัญหาในการ Extract อันเนื่องมาจาก Standard Cell ที่มีอยู่นั้นเป็น SubCircuit ซึ่งผู้ทดลองมีประสบการณ์ไม่เพียงพอ จึงทำให้เกิดปัญหาขึ้นในหลายๆ จุดโดยเฉพาะในส่วนของการทดสอบ Layout ซึ่งเมื่อทำการ Extract แล้วพบว่า Port ของวงจรจะถูกเรียงจากลำดับตัวอักษร ซึ่งจะทำให้แตกต่างจาก Netlist ที่แปลงจาก VHDLจึงทำให้ไม่สามารถทำ LVS ได้อย่างถูกต้อง

3. จากการออกแบบโดยใช้ภาษา VHDL มีข้อจำกัดในการใช้งานในลักษณะความซับซ้อนของโปรแกรม จึงทำให้ผลการสังเคราะห์ไม่ผ่าน ทำให้ต้องมีการปรับปรุงแก้ไขบ่อยครั้ง

4. ผลที่ได้จากการสังเคราะห์ได้ File.Edif ออกมานั้น Port ที่ได้มาไม่ครบตามที่ทำการออกแบบไว้ เมื่อดูจาก Text file (Text.Edif) ทำให้ไม่สามารถเชื่อมต่อกับบล็อกในส่วนอื่นได้และไม่สามารถทำการ Mapping ได้เมื่อนำมาใช้การ Mask Layout ติดปัญหาที่ cell บางตัวไม่สามารถทำ

การ Mapping ได้ อาจเกิดข้อผิดพลาดในการเขียนโปรแกรมควบคุมการทำงานในส่วนของ VHDL ทำให้การสังเคราะห์เกิดการผิดเพี้ยนไปจึงไม่สามารถทำการ Mapping ได้

5. จากการใช้งานของโปรแกรมในกระบวนการ ASIC นั้น โดยเฉพาะ โปรแกรม Leonardo Spectrum และ L-Edit ที่ใช้ในการสังเคราะห์และ Mask Layout นั้น Library ที่ใช้งานนั้นไม่เพียงพอและไม่รองรับกับการออกแบบจึงทำให้เกิดปัญหาในการทำงาน

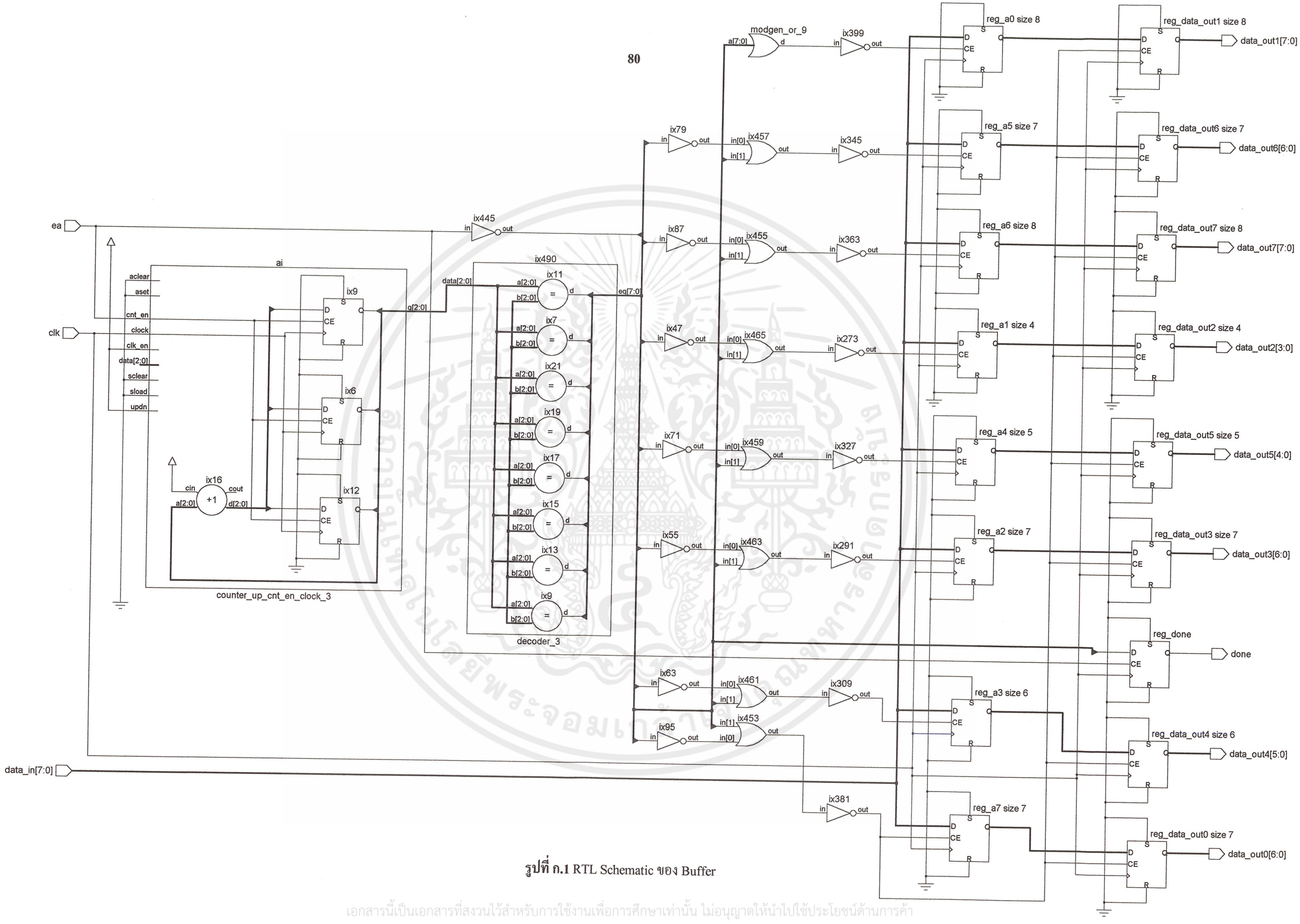
5.3 แนวทางการแก้ไข และพัฒนา

1. ในส่วนของการประมวลผลของสัญญาณส่วนของ DCT และ IDCT ได้ใช้การประมวลผลโดยการใช้จำนวนเต็ม (Fixed-Point) แทนการคำนวณแบบจุดทศนิยม (Floating-Point) แต่ผลการคำนวณยังมีผลที่เหมือนเดิมเพียงแต่จะเกิดความเพี้ยนของสัญญาณบ้างแต่เป็นส่วนที่เล็กน้อย เพื่อให้การออกแบบทาง VHDL เป็นการใช้งานคำสั่งที่ไม่ซับซ้อน

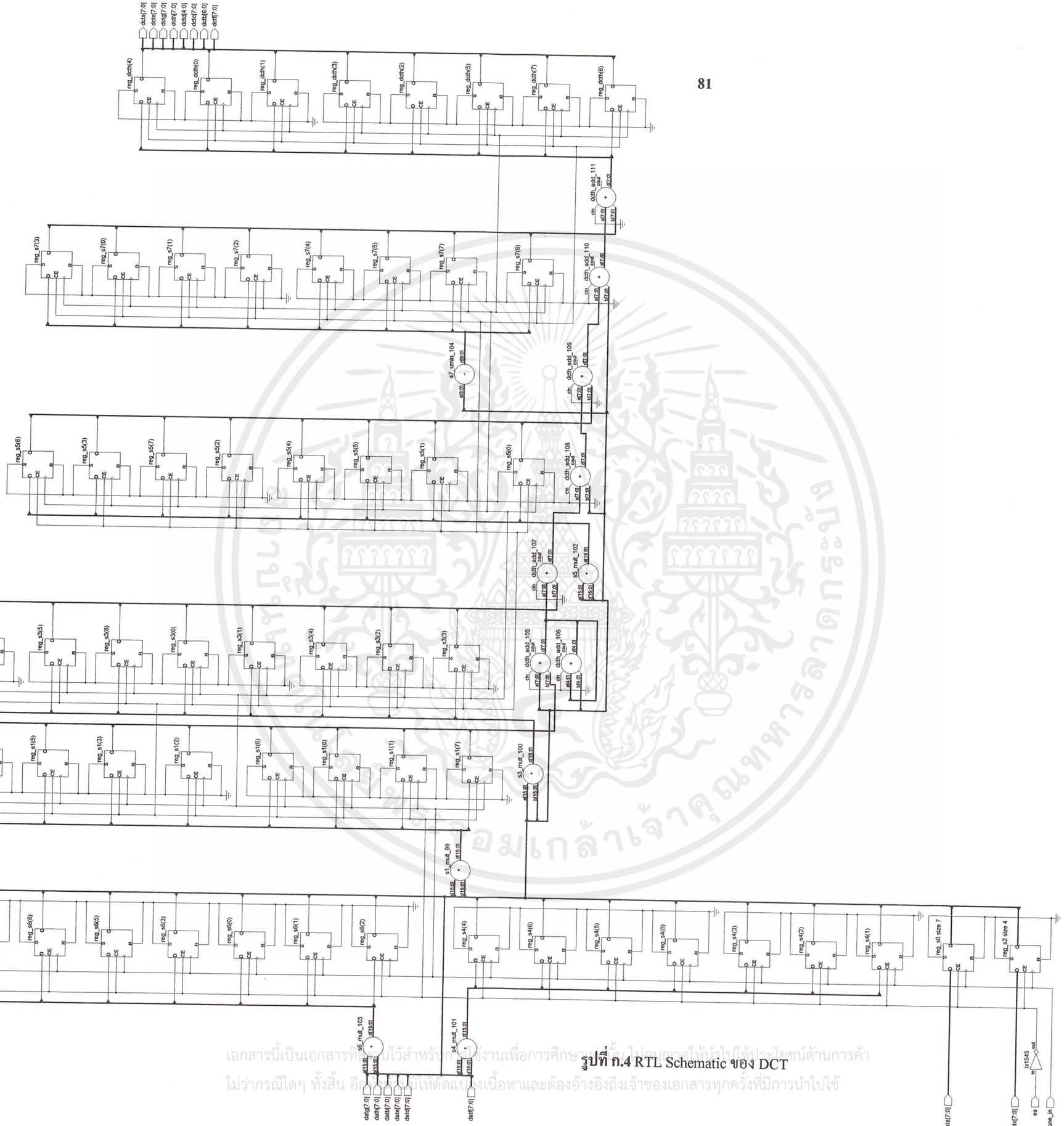
2. แก้ไขการออกแบบภาษา VHDL ใหม่ในรูปแบบของโครงสร้างโปรแกรมรูปแบบใหม่ เพื่อให้โปรแกรมที่ใช้ในการสังเคราะห์รองรับการทำงานได้มากขึ้น

3. จำเป็นต้องใช้โปรแกรมที่มี License ที่ถูกต้องและ Library ที่ได้จากโรงงานเจือสารซึ่งมี Library ที่ครบและเพียงพอในการใช้งานซึ่งขึ้นอยู่กับลักษณะของงานด้วย แต่ทั้งนี้ทั้งนั้นเนื่องจากการทำโครงงานนี้เป็นการศึกษากระบวนการ ASIC Design โดยนำหลักการของตัวเข้ารหัสและถอดรหัส DCT มาเป็นกรณีศึกษา จึงเห็นควรว่าไม่จำเป็นต้องจัดหาโปรแกรมดังกล่าวซึ่งจะติดปัญหาในหลายๆ เรื่องโดยเฉพาะเรื่องค่าใช้จ่ายที่ค่อนข้างสูงจึงได้มีการศึกษาหาแนวทางอื่น โดยการออกแบบใช้งาน Library ในส่วนที่มีอยู่ ทำให้ได้ศึกษาขั้นตอนกระบวนการสร้างและออกแบบที่ถูกต้องเพื่อเป็นพื้นฐานของการออกแบบในระดับสูงต่อไปในอนาคต

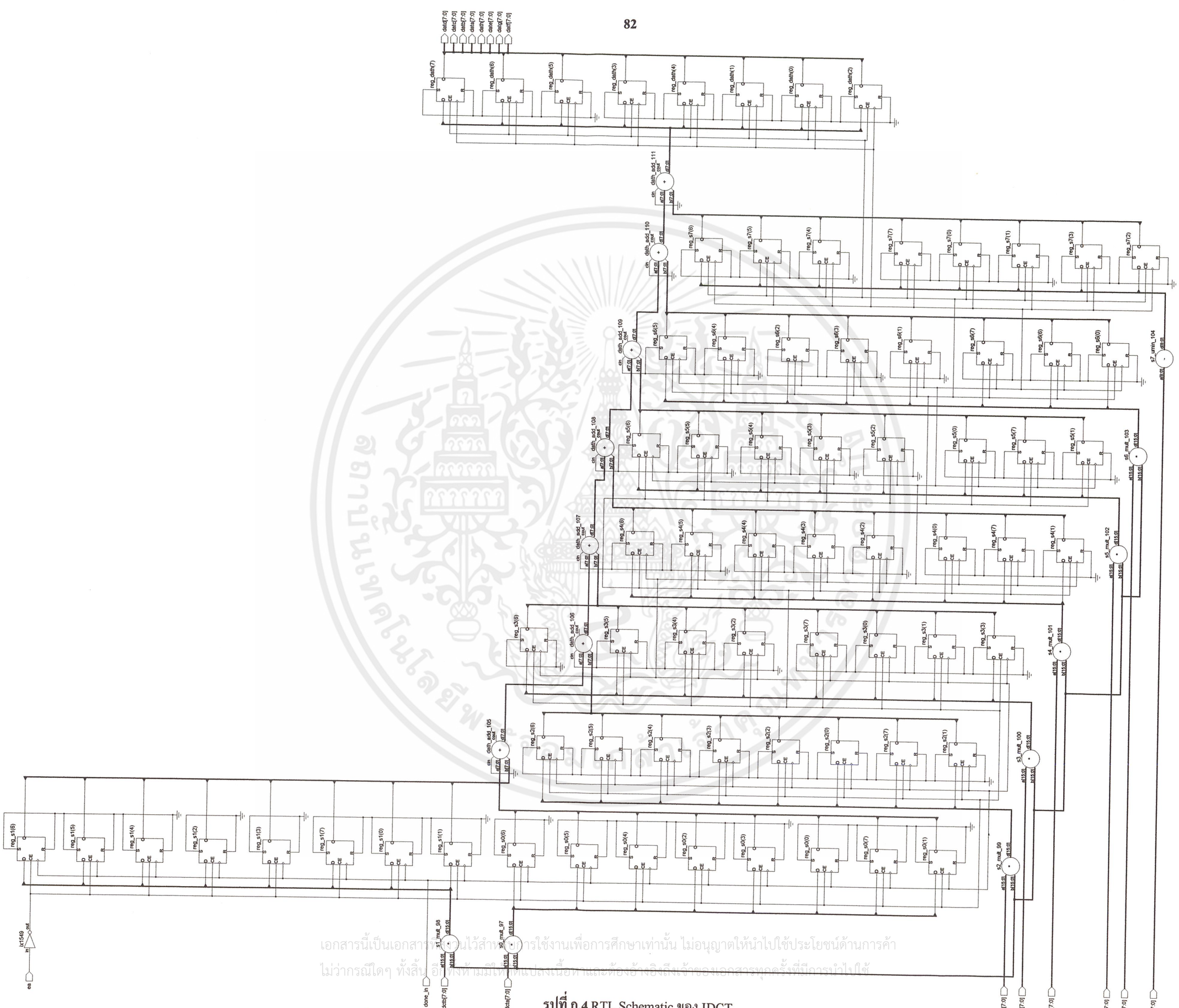
ภาคผนวก ก
รายละเอียดของวงจร



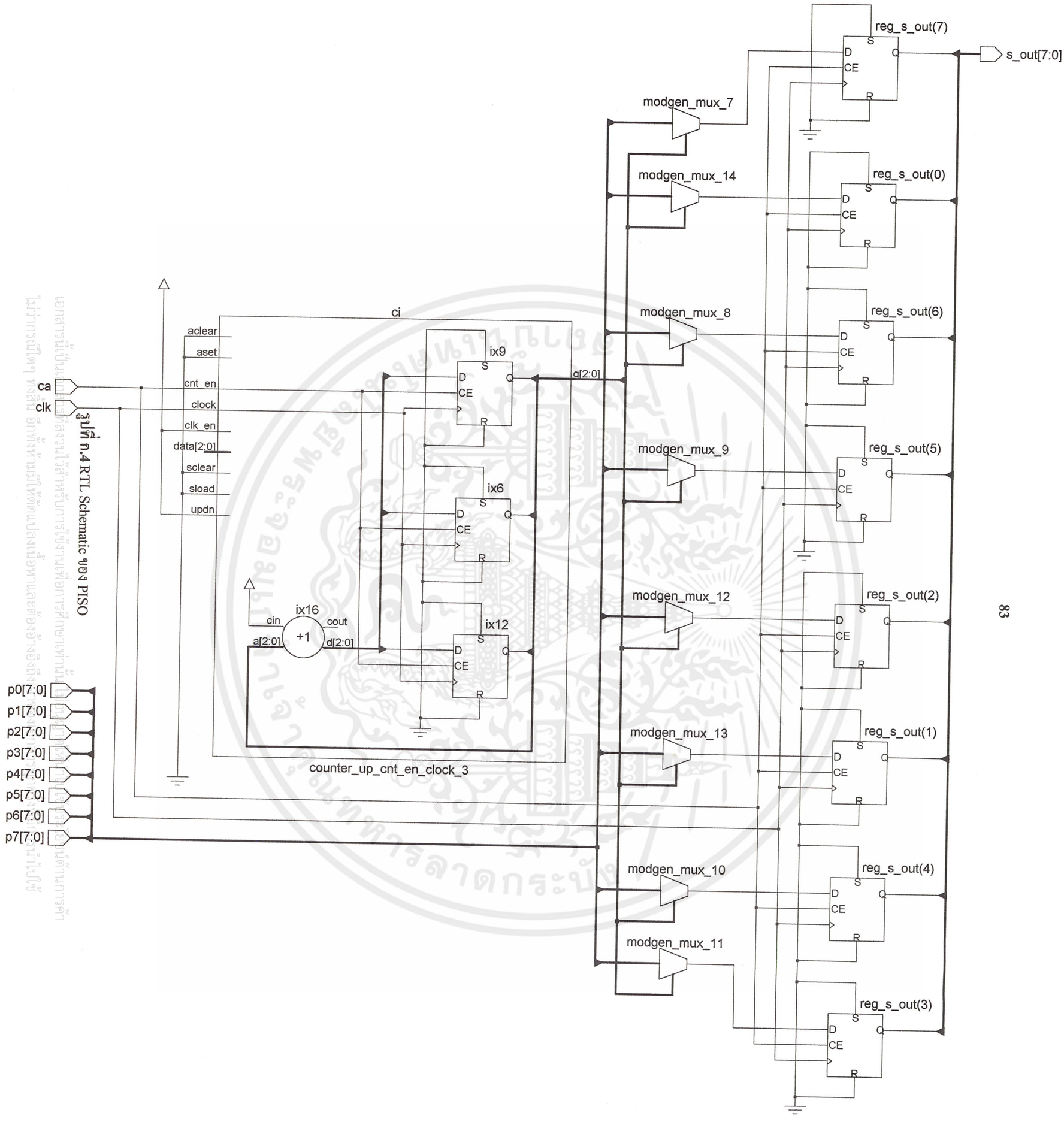
รูปที่ ก.1 RTL Schematic ของ Buffer



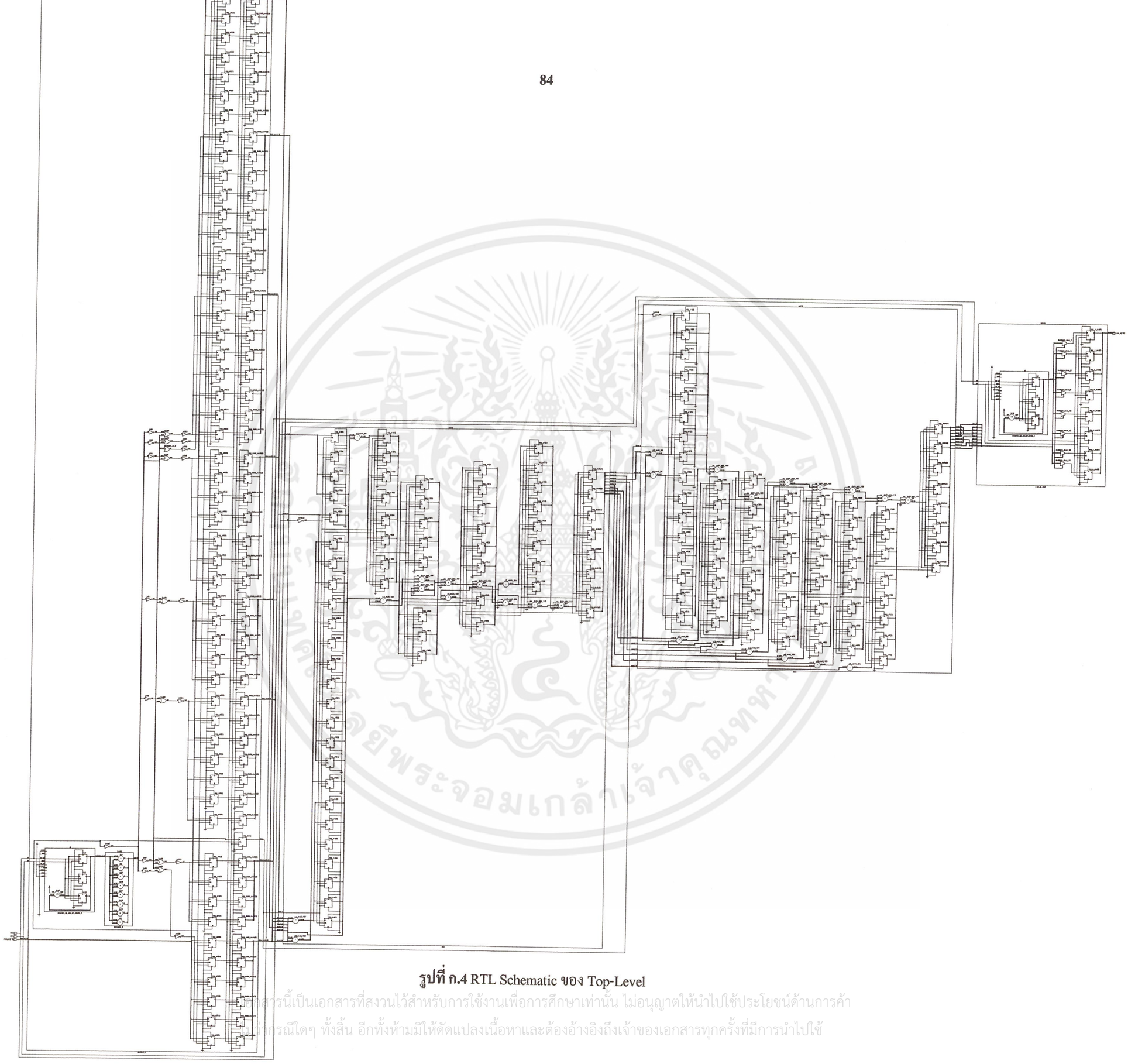
รูปที่ ก.4 RTL Schematic ของ DCT



รูปที่ ก.4 RTL Schematic ของ IDCT



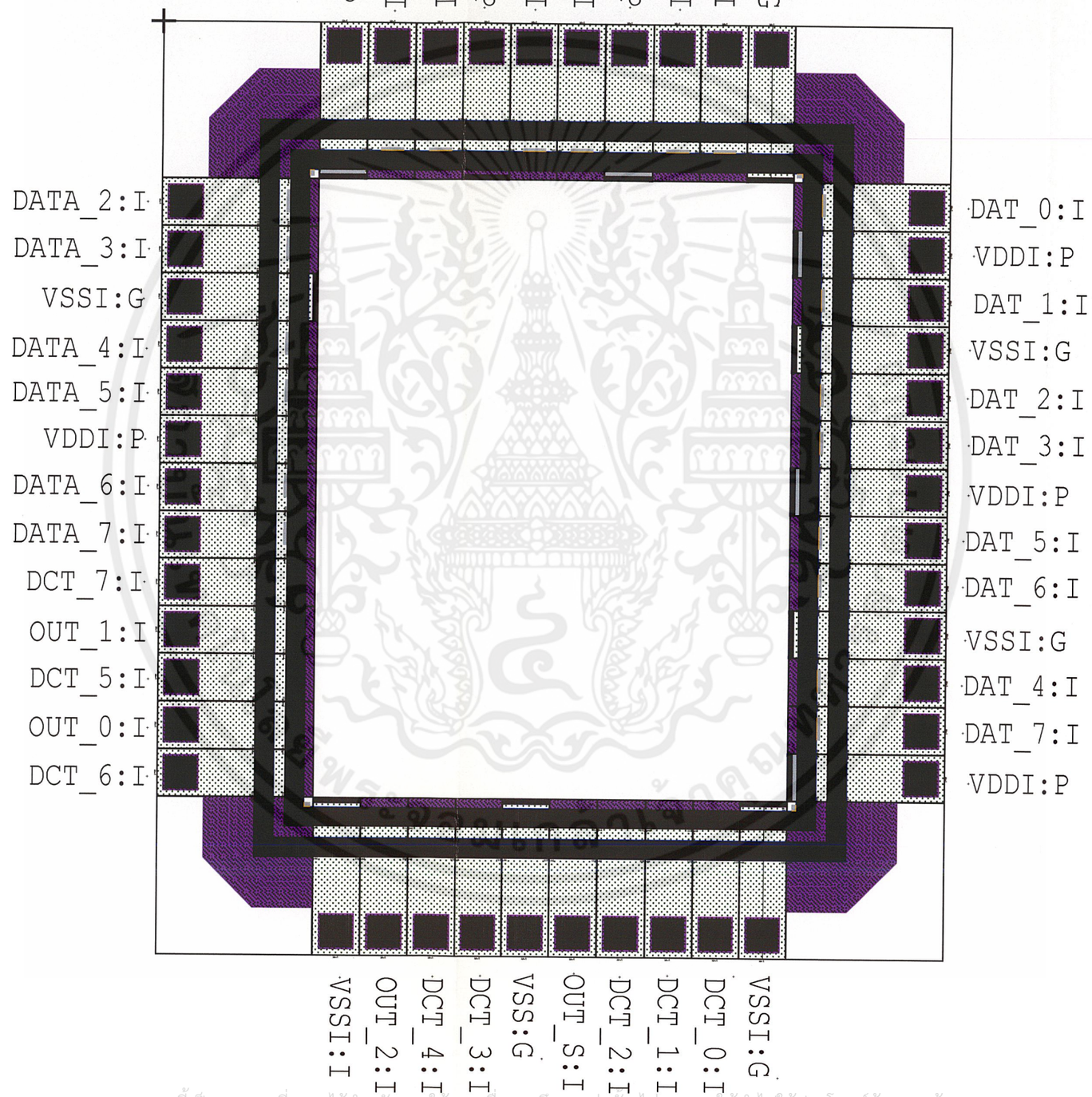
รูปที่ ก.4 RTL Schematic ของวงจร PISO



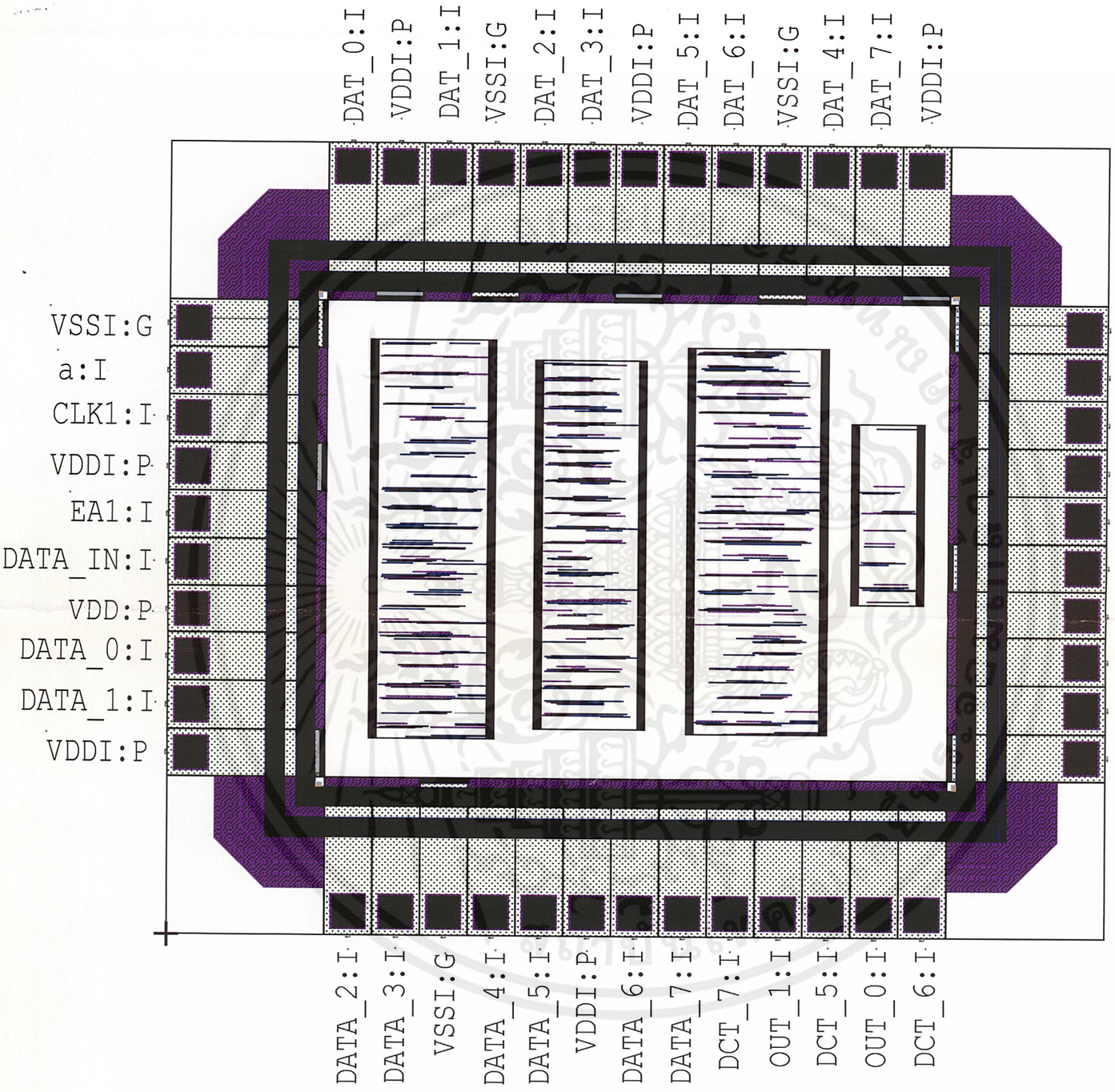
รูปที่ ก.4 RTL Schematic ของ Top-Level

ภาคผนวก ข

Layout Padframe



รูปที่ ข.1 Layout ของ Padframe



รูปที่ ข.2 Layout ของวงจรรวม

ภาคผนวก ค

ภาษา VHDL

```
>> type DCT
function b=dct(a,n)
error(nargchk(1,2,nargin));
if min(size(a))==1
   if size(a,2)>1
      do_trans = 1;
   else
      do_trans = 0;
   end
   a = a(:);
else
   do_trans = 0;
end
if nargin==1,
 n = size(a,1);
end
m = size(a,2);
% Pad or truncate a if necessary
if size(a,1)<n,
 aa = zeros(n,m);
 aa(1:size(a,1),:) = a;
else
 aa = a(1:n,:);
end
if rem(n,2)==1 | ~isreal(a), % odd case
% Form intermediate even-symmetric matrix.
 y = zeros(2*n,m);
 y(1:n,:) = aa;
 y(n+1:n+n,:) = flipud(aa);
```

```
% Perform FFT
  yy = fft(y);
  % Compute DCT coefficients
  ww = (exp(-i*(0:n-1)*pi/(2*n))/sqrt(2*n)).';
  ww(1) = ww(1) / sqrt(2);
  b = ww(:,ones(1,m)).*yy(1:n,:);
else % even case
  % Re-order the elements of the columns of x
  y = [ aa(1:2:n,:); aa(n:-2:2,:) ];
  % Compute weights to multiply DFT coefficients
  ww = 2*exp(-i*(0:n-1)'*pi/(2*n))/sqrt(2*n);
  ww(1) = ww(1) / sqrt(2);
  W = ww(:,ones(1,m));
  % Compute DCT using equation (5.92) in Jain
  b = W .* fft(y);
end
if isreal(a), b = real(b); end
if do_trans, b = b.'; end
```

รูปที่ ค.1 โปรแกรมการคำนวณค่าสัมประสิทธิ์ DCT โดยโปรแกรม MATLAB

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
entity project_H is
        port (data_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 25;
              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.2 โปรแกรมส่วน Buffer

```
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
        else 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

```
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;
                ca : 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(ca,clk,ci)
begin
        if clk'event and clk ='1' then
        if ca ='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.5 โปรแกรมส่วน PISO

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_signed.all;
entity buffer_dct is
        port (data_in : in INTEGER range 0 to 255;
                clk1 : in STD_LOGIC;
                ea1 : in STD_LOGIC;
                out_s : out integer range 0 to 255);
end buffer_dct ;
architecture buffer_dct_arch of buffer_dct is
 component project_H
                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 127;
                data_out1 : out INTEGER range 0 to 255;
                data_out2 : out INTEGER range 0 to 15;
                data_out3 : out INTEGER range 0 to 127;
                data_out4 : out INTEGER range 0 to 63;
                data_out5 : out INTEGER range 0 to 31;
                data_out6 : out INTEGER range 0 to 127;
                data_out7 : out INTEGER range 0 to 255);
end component;
component DCT
                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 -64 to 63;
                    dctc : out integer range -128 to 127;
                    dctd : out integer range -32 to 31;
                    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 component;
component IDCT
            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 component;
component s_in_p_out
                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;
                    ca : in STD_LOGIC;
                    s_out : out integer range 0 to 255);
end component;
signal  a : STD_LOGIC;
signal  z0,z1,z2,z3,z4,z5,z6,z7  :integer range 0 to 255;
signal  k0,k1,k2,k3,k4,k5,k6,k7  :integer range -128 to 127;
signal  y0,y1,y2,y3,y4,y5,y6,y7  :integer range 0 to 255;
```

```
begin
unit1 : project_H port map ( data_in=>data_in,
                    clk => clk1,
            EA =>ea1,
            done=>a,
           data_out0=>z0,
           data_out1=>z1,
           data_out2=>z2,
           data_out3=>z3,
           data_out4=>z4,
           data_out5=>z5,
           data_out6=>z6,
           data_out7=>z7);
unit2 : DCT port map (ea=>ea1,
           done_in=>a,
          data=>z0,
          datb=>z1,
          datc=>z2,
          datd=>z3,
          date=>z4,
          datf=>z5,
          datg=>z6,
          dath=>z7,
          dcta=>k0,
          dctb=>k1,
          dctc=>k2,
```

```
                dctd=>k3,
                dcte=>k4,
                dctf=>k5,
                dctg=>k6,
                dcth=>k7 );
unit3 : IDCT port map (ea => ea1,
                done_in=>a,
                dcta=>k0,
                dctb=>k1,
                dctc=>k2,
                dctd=>k3,
                dcte=>k4,
                dctf=>k5,
                dctg=>k6,
                dcth=>k7,
                data=>y0,
                datb=>y1,
                datc=>y2,
                datd=>y3,
                date=>y4,
                datf=>y5,
                datg=>y6,
                dath=>y7);
unit4 : s_in_p_out port map (p0=>y0,
                p1=>y1,
                p2=>y2,
                p3=>y3,
```

```
        p4=>y4,
                    p5=>y5,
                    p6=>y6,
                    p7=>y7,
                    clk=>clk1,
                    ca=>ea1,
                    s_out=>out_s);

    end buffer_dct_arch;
```

รูปที่ ค.6 โปรแกรม Top-Level

บรรณานุกรม

เทอดศักดิ์ ธนกิจประภาและคณะ. “โปรแกรมการบีบอัดภาพนิ่งด้วยวิธี JPEG.” หน้า 1012 - 1019. ใน **การประชุมทางวิศวกรรมไฟฟ้า ครั้งที่ 18.** กรุงเทพฯ : คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2538

มนัส สังวรศิลป์ และ วรรัตน์ ภัทรอมรกุล. คู่มือโปรแกรม MATLAB ฉบับสมบูรณ์. กรุงเทพฯ : อินโฟร์เพรส. 2543

ชัยรัตน์ ฤทธิรงค์. “การบีบอัดข้อมูลทางการแพทย์โดยการมอดูเลตแบบรหัสพัลส์เชิงอนุพันธ์.” วิทยานิพนธ์วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า บัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2539

ชาญวิทย์ เคนอัศวนง. “การออกแบบวงจรรวมซีมอสสำหรับถอดรหัสบัตรแม่เหล็ก.” วิทยานิพนธ์วิศวกรรมศาสตร์มหาบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า บัณฑิตวิทยาลัยสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง. 2541

Michael, J.S.S. Application Specific Integrated Circuit. Massachusetts : Addison - Wesley Longman, Inc. 1998

DanClein. CMOS IC LAYOUT Concept, Methologies, and Tools. Boston : Butterworth - Heinemann, Inc. 1999

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายธนันต์ชัย บรรเทิงจิตร
วัน เดือน ปีเกิด	วันที่ 11 ธันวาคม 2521
สถานที่เกิด	โรงพยาบาลสุราษฎรธานี
ภูมิเลาเนาเดิม	สุราษฎรธานี
ที่อยู่ปัจจุบัน	111 ถ. พูลศิริ ต. นาสาร อ. บ้านนาสาร
	จ. สุราษฎรธานี 84120
โทรศัพท์	(077)341729
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนนาสาร
มัธยมศึกษาตอนต้น	โรงเรียนบ้านนาสาร
ประกาศนียบัตรวิชาชีพ (ปวช.)	โรงเรียนสุราษฎร์เทคโนโลยีช่างอุตสาหกรรม
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคสุราษฎร์ธานี
ปริญญาตรี	สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์
	ภาควิชาครุศาสตร์วิศวกรรม
	คณะครุศาสตร์อุตสาหกรรม
	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร
	ลาดกระบัง
คติพจน์	สจิตฺตปริโยทปนำ

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นางสาวนวรัตน์ ลิมาภิรักษ์
วัน เดือน ปีเกิด	วันที่ 23 เมษายน 2522
สถานที่เกิด	โรงพยาบาลมหาสารคาม
ภูมิเลาเนาเดิม	มหาสารคาม
ที่อยู่ปัจจุบัน	93/3 หมู่ 3 ต.โคกพระ อ. กันทรวิชัย จ. มหาสารคาม 44150
โทรศัพท์	(043)789199
ประวัติการศึกษา	
ประถมศึกษา	โรงเรียนอนุบาลมหาสารคาม
มัธยมศึกษาตอนต้น	โรงเรียนผดุงนารี
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคมหาสารคาม
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	สถาบันเทคโนโลยีราชมงคล วิทยาเขตขอนแก่น
ปริญญาตรี	สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
คติพจน์	อตฺตาหิอตฺตโนนาโถ