การ์ดประมวลผลดิจิตอลภาพแบบโปรแกรมได้

RE-PROGRAMMABLE DIGITAL IMAGE PROCESSING CARD

โดย

นายปรเมศวร์ ห่อแก้ว

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

การ์ดประมวลผลดิจิตอลภาพแบบโปรแกรมได้

RE-PROGRAMMABLE DIGITAL IMAGE PROCESSING CARD

โดย

นายปรเมศวร์ ห่อแก้ว 38014271

อาจารย์ที่ปรึกษา

ผศ.ดร.ยุทธพงษ์ รังสรรค์เสรี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

การ์ดประมวลผลภาพดิจิตอลแบบโปรแกรมได้

RE-PROGRAMMABLE DIGITAL IMAGE PROCESSING CARD

โดย นายปรเมศวร์ ห่อแก้ว 38014271

อาจารย์ที่ปรึกษา ผศ.ดร.ยุทธพงษ์ รังสรรค์เสรี

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอการออกแบบการ์ดประมวลผลภาพดิจิตอลแบบโปรแกรมได้ สำหรับการใช้ในงานประยุกต์แบบเวลาจริง การ์ดดังกล่าวประกอบด้วยสองส่วนได้แก่ ส่วนแปลงสัญญาณวิดีโอให้เป็นข้อมูลภาพดิจิตอล และส่วนประมวลผลภาพ ส่วนแปลงสัญญาณภาพแปลงสัญญาณวิดีโอมาตรฐานให้เป็นข้อมูลภาพดิจิตอล 256 ระดับขนาด 1 ฟิลด์ เอาท์พุทต่อเข้ากับส่วนประมวลผลซึ่งออกแบบโดยใช้ FPGA (Field Programmable Gates Array) สามารถโปรแกรมให้ทำการประมวลผลใดๆ ภายในแมสค์ขนาด 3x3 จุดภาพได้ ผลลัพธ์ที่ได้รับจะถูกส่งออกมาด้วยอัตราเร็วเดียวกับอัตราการสแกนภาพวิดีโอ ผลการทดลองแสดงการประยุกต์ใช้งานการ์ดนี้ด้วยการโปรแกรมแมสค์ใบโนเมียลลงบน FPGA สำหรับการกำจัดสัญญาณรบกวนภายในภาพ

Abstract

This thesis presents the design of a re-programmable digital image-processing card for the use in real-time applications. The card is composed of two significant parts: the digitizer and the image processor. The digitizer can convert each frame of the standard video signal into a digital image with 256 gray levels, size of 1 field. The output is fed into the processor part, which is implemented by using FPGA (Field Programmable Gates Array), and can be programmed to perform any operations within a 3 x 3 mask. The result is output at the same rate of the video scan rate. An experimental result is given, with the FPGA programmed as a binomial mask, for the noise reduction purpose.

สารบัญ

สารบัญ

หน้า

สารบัญ

สารบัญรูปภาพ

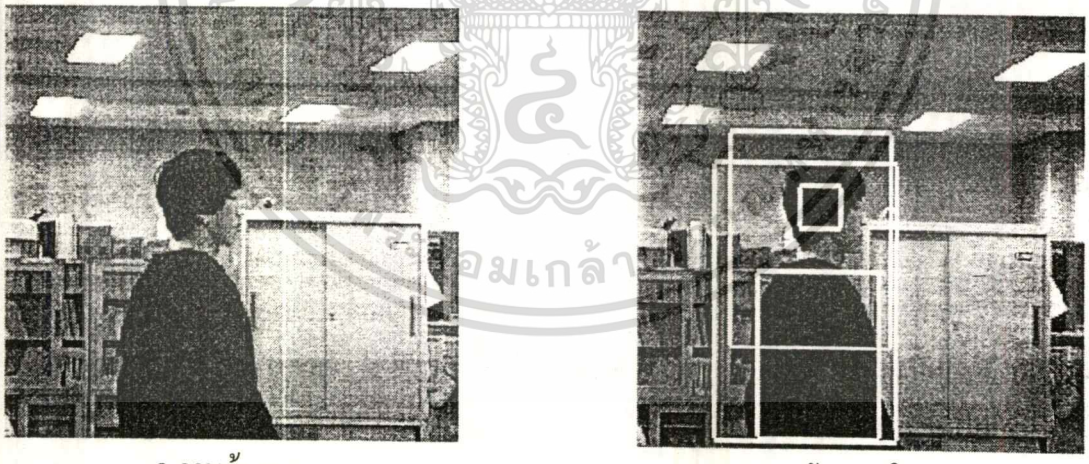
สารบัญตาราง

บทที่ 1
บทนำ

1.1 หลักการและเหตุผล

การประมวลผลภาพดิจิตอล ในบางกรณี อาทิเช่น การหาบริเวณของภาพเคลื่อนไหวเพื่อการระบุถึงวัตถุภายในภาพ (Object Identification) ในการประยุกต์ใช้งานประเภทการติดตามวัตถุ (Object Tracking) ดังแสดงในรูปที่ 1.1 จำเป็นจะต้องใช้การประมวลผลภาพแบบเวลาจริง ทั้งนี้เนื่องจากภาพดังกล่าวประกอบด้วยชุดของภาพ ซึ่งมีช่วงระยะเวลาระหว่างกันคงที่ และสัมพันธ์กัน ถึงแม้ว่าเทคโนโลยีซอฟต์แวร์ในปัจจุบันจะสามารถรองรับความต้องการดังกล่าวได้ในระดับหนึ่ง แต่การใช้ฮาร์ดแวร์ในการประมวลผลก็ยังเป็นที่นิยมกันอย่างแพร่หลาย เนื่องจากมีข้อดีคือ สามารถประมวลผลด้วยความเร็วสูงในระดับหลายสิบ MHz ซึ่งเหมาะกับการประยุกต์ใช้งานกับข้อมูลภาพซึ่งมีปริมาณมาก และความเร็วเป็นสิ่งสำคัญโดยเฉพาะอย่างยิ่งภาพเคลื่อนไหว, ระบบโดยรวมมีขนาดเล็ก เหมาะสำหรับการผลิตในเชิงอุตสาหกรรมทำให้ผลิตภัณฑ์มีราคาถูกลง

ปริญญานิพนธ์นี้เสนอวิธีการออกแบบการ์ดประมวลผลภาพดิจิตอลแบบโปรแกรมได้ ซึ่งแปลงสัญญาณวิดีโอมาตรฐานเป็นข้อมูลภาพดิจิตอลขนาด 384x288 จุดภาพ หรือ 1 ฟิลด์ 256 ระดับความเทา แล้วนำมาผ่านการประมวลผลด้วยคอนโวลเวอร์ (Convolver) ขนาด 3x3 จุดภาพ ซึ่งสัมประสิทธิ์ได้ถูกโปรแกรมไว้ใน FPGA บนการ์ด ผลลัพธ์ที่ได้จัดเก็บไว้ในบัฟเฟอร์ พร้อมที่จะอ่านและใช้ประโยชน์โดยเครื่องคอมพิวเตอร์ต่อไป ทั้งนี้ระบบสามารถทำงานได้แบบเวลาจริงระดับอัตราของภาพวิดีโอ

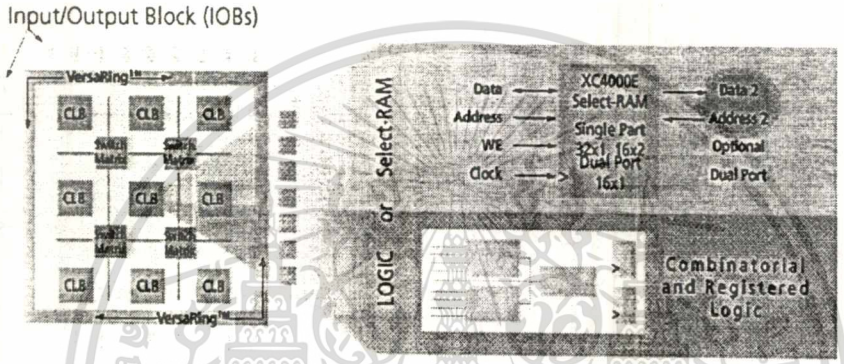


ก ภาพต้นแบบ ข ระบุวัตถุภายในภาพ

รูปที่ 1.1 การประยุกต์ใช้งานประเภทการติดตามวัตถุ

เทคโนโลยี FPGA (Field Programmable Gate Array) เป็นเทคโนโลยีของวงจรรวมขนาดใหญ่มาก (Very Large Scale Integrated Circuit: VLSI) ซึ่งได้มีการรวบรวมวงจรตรรกพื้นฐานจำนวนมากไว้ภายใน ดังแสดงในรูปที่ 1.2 และมีขาต่อออกมาใช้งานภายนอกอีกจำนวนหนึ่งขึ้นอยู่กับขนาดและชนิดของ FPGA ใน

การใช้งานจะโปรแกรมวงจร, พฤติกรรมของระบบ หรือ สมการตรรกะ (Logic Equation) เข้าไปในวงจรรวม ผ่านการเชื่อมต่อที่ได้กำหนดไว้โดยผู้ผลิต



รูปที่ 1.2 วงจรรวม FPGAและโครงสร้างภายใน

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. ศึกษาเทคโนโลยีของ FPGA ในแง่ของ ทฤษฎี โครงสร้าง และการทำงานพื้นฐาน
2. ประยุกต์ใช้ FPGA กับตัวดำเนินการคอนโวลเวอร์ ที่ต้องการการหน่วงจุดภาพไม่เกิน 2 เส้น โดยใช้วงจรตรรก
3. ออกแบบและสร้างการ์ดประมวลผลภาพดิจิตอลแบบโปรแกรมได้ ซึ่งประกอบด้วยวงจรรวม FPGA

1.3 ขอบเขตของปริญญานิพนธ์

เพื่อให้บรรลุวัตถุประสงค์ดังกล่าวข้างต้น จึงกำหนดขอบเขตของงานดังนี้

1. โปรแกรมคอนโวลเวอร์ที่นิยมใช้กันโดยทั่วไป ที่อยู่ภายในเมตริกซ์ ขนาดไม่เกิน 3x3 ลงไปบนวงจรรวม FPGA
2. ออกแบบและสร้างการ์ดประมวลผลภาพดิจิตอลแบบโปรแกรมได้ ซึ่งประกอบด้วยวงจรรวม FPGA โดยให้มีคุณสมบัติตามความต้องการดังตารางที่ 1.1

Item	*Specification*
Video Signal	
Standard	CCD
Captured Image	
Image size	388 x 284 pixels (single field)
Image depth	8-Bit gray scale
Processing	
Operator type	Any operator with 2-line delay, e.g. Convolver
External delay available	2 lines
Host Interface	
PC Interface type	8-bit ISA slot
Image transferring	8-bit I/O port

ตารางที่ 1.1 คุณสมบัติของการ์ดที่ต้องการออกแบบ

3. ทดลองประสิทธิภาพและการทำงานของการ์ดที่สร้างกับการลดสัญญาณรบกวนภายในภาพ โดยใช้คอนโวลเวอร์ แบบไบโนเมียล ขนาด 3x3

1.4 เนื้อหาของรายงานโดยสังเขป

บทที่ 1 บทนำ กล่าวถึงหลักการและเหตุผลของโครงงาน วัตถุประสงค์ และขอบเขตของโครงงาน

บทที่ 2 กล่าวถึงทฤษฎีเบื้องต้นที่ประยุกต์ใช้ในโครงงาน ได้แก่ แนวคิดของเครื่องจักรที่สามารถมองเห็นได้ (Vision Machine) เทคโนโลยีของ FPGA และการใช้งานโดยละเอียด

บทที่ 3 กล่าวถึงขั้นตอนในการออกแบบการ์ดประมวลผลภาพดิจิตอลแบบโปรแกรมได้ การออกแบบและใช้งานตัวดำเนินการคอนโวลเวอร์บน FPGA โดยละเอียด และการออกแบบขั้นตอนวิธีและการเขียนโปรแกรมควบคุมการทำงาน

บทที่ 4 การทดลองและผลการทดลอง แสดงผลการทดสอบวงจรของการ์ดประมวลผลภาพดิจิตอลในส่วนแปลงสัญญาณภาพ (Digitizer) และผลจากการใช้งานจริงของการ์ดเปรียบเทียบโดยใช้ผลของฮิสโตแกรม

บทที่ 5 สรุปและวิจารณ์ผลการทดลอง วิเคราะห์ถึงข้อเด่นข้อด้อยของโครงงาน และเสนอแนะแนวทางในการพัฒนาต่อไป

บทที่ 2
ทฤษฎีและหลักการพื้นฐาน

2.1 หลักการวิชั่นแมชชีน (Machine Vision)

2.1.1 เปรียบเทียบวิชั่นแมชชีนกับการมองเห็นของมนุษย์

วิชั่นแมชชีนคือเครื่องจักรทำหน้าที่แปลงข้อมูลภาพโดยไม่มีการสัมผัสและทำการวิเคราะห์แบบอัตโนมัติเพื่อใช้ในกระบวนการและกิจกรรมในการควบคุม

เมื่อเปรียบเทียบกับการมองเห็นของมนุษย์ซึ่งประกอบด้วย ตา, ประสาทตา, และสมอง ตาทำหน้าที่สร้างภาพที่เกิดจากแสงโดยเลนส์ และตรวจจับโดยเรตินา (Retina) ประสาทตาส่งข่าวสารของภาพไปยังสมองซึ่งจะวิเคราะห์และแยกเอาข่าวสารภาพออกมา เพื่อที่ว่าระบบสมองส่วนอื่นจะนำข้อมูลนี้ไปใช้ควบคุมกล้ามเนื้อต่อไป

แบบจำลองของวิชั่นแมชชีนอุปมาได้กับระบบดังกล่าว กล้องถ่ายภาพซึ่งมีเลนส์สร้างภาพที่เกิดจากแสงบนตัวตรวจจับภาพ สัญญาณวิดีโอถูกส่งไปยังคอมพิวเตอร์ผ่านทางสายเคเบิลเพื่อวิเคราะห์และดึงเอาข่าวสารภาพที่จำเป็นออกมา ข่าวสารดังกล่าวจะถูกส่งไปยังส่วนควบคุมเครื่องจักรส่วนอื่นๆ จุดประสงค์ของวิชั่นแมชชีนคือพยายามเลียนแบบระบบการมองเห็นของมนุษย์ให้มากที่สุด เพื่อแทนที่การทำงานที่ต้องเสี่ยงอันตรายและต้องปฏิบัติซ้ำๆ เช่นในสิ่งแวดล้อมที่ประกอบด้วยรังสีเอกซ์ (X-ray) เป็นต้น

2.1.2 นิยามของวิชั่นแมชชีน (Vision Machine Definition)

คำนิยามอย่างเป็นทางการของวิชั่นแมชชีนที่เป็นที่ยอมรับกันได้แก่

"เครื่องจักรที่ทำหน้าที่เก็บข้อมูลภาพโดยไม่มีการสัมผัสและมีการวิเคราะห์แบบอัตโนมัติเพื่อแยกเอาข้อมูลที่จำเป็นออกมาเพื่อจุดประสงค์ในการควบคุมกระบวนการหรือกิจกรรม"

คำสำคัญที่จำเป็นต้องกล่าวถึงได้แก่คำว่า "อัตโนมัติ" ซึ่งหมายถึงการทำงานได้ด้วยตัวเอง ระบบวิชั่นแมชชีนสามารถดำเนินงานได้โดยไม่ต้องมีการเกี่ยวข้องของมนุษย์ แต่เนื่องความต้องการความเชื่อถือได้ดังนั้นการประยุกต์ใช้งานของระบบอัตโนมัติดังกล่าวจึงถูกจำกัดโดยเทคโนโลยีปัจจุบัน

ส่วนคำว่า "การเก็บ" และ "การวิเคราะห์" แสดงถึงความแตกต่างจากทัศนอุปกรณ์ ทางอิเลกทรอนิกส์อื่น เช่น โทรทัศน์จะทำหน้าที่เก็บข้อมูลภาพอย่างเดียวโดยไม่มีการวิเคราะห์ ในขณะที่เครื่องคอมพิวเตอร์จะทำหน้าที่วิเคราะห์ภาพโดยไม่ทำการเก็บข้อมูลภาพโดยตรง

ส่วนคำว่า "ไม่สัมผัส" เพื่อแบ่งแยกออกจากเครื่องมือตรวจจับผลิตภัณฑ์ทางอุตสาหกรรมซึ่งต้องมีการสัมผัสวัตถุโดยตรง ข้อดีของการไม่สัมผัสคือ มีความเร็วสูง และสามารถเชื่อถือได้

เนื่องจากวิชั่นแมชชีนจะเก็บข้อมูลเฉพาะ "ข้อมูลที่ต้องการ" เพื่อให้บรรลุจุดประสงค์ที่ต้องการซึ่งข้อมูลที่ต้องการจะต้องนิยามให้เหมาะสมกับการประยุกต์ใช้งานที่ต้องการ ส่วน "การควบคุมกระบวนการหรือกิจกรรม" เป็นกระบวนการโดยปกติของการผลิต เช่นการควบคุมแขนหุ่นยนต์ให้ติดอุปกรณ์แบบเวลาจริง

2.1.3 องค์ประกอบของระบบวิชั่นแมชชีน (Vision Machine System Components)

ระบบวิชั่นแมชชีนส่วนมากประกอบด้วย ส่วนสำคัญสามประการได้แก่ แหล่งแสง, กล้องถ่ายภาพ, และตัวประมวลผล เพื่อวิเคราะห์ภาพ

บางระบบอุปกรณ์ดังกล่าวสามารถแบ่งแยกได้อย่างชัดเจน ในขณะที่บางระบบอุปกรณ์ทั้งหมดประกอบรวมกันอยู่ในชิ้นส่วนเดียว

2.1.4 การจำแนกตามการประยุกต์ใช้งาน (Application Categories)

การประยุกต์ใช้งานของวิชั่นแมชชีนได้แก่ การประกันคุณภาพ คือการแยกผลิตภัณฑ์ที่เสียออกจากผลิตภัณฑ์ที่ดี, การเรียบเรียง คือการจัดเรียงวัตถุตามลักษณะทางกายภาพ, การควบคุมกระบวนการ คือการตรวจจับผลิตภัณฑ์ในลักษณะเวลาจริงเพื่อนำข้อมูลไปใช้ในการควบคุม, การจับถือวัสดุ เพื่อทำงานอัตโนมัติด้วยความเร็วสูง, การแนะนำหุ่นยนต์ เช่น การทำงานอัตโนมัติโดยหุ่นยนต์ที่อาศัยการมองเห็น, การตรวจสอบและปรับแต่ง เช่นผลิตภัณฑ์ประเภทเครื่องมือตรวจจับความร้อน, การตรวจตราเครื่องจักร เพื่อปกป้องการลงทุนที่อาจเสียหายหากเครื่องจักรทำงานผิดพลาด และ ในด้านความปลอดภัยในการดูแลการทำงานของคนงาน เป็นต้น

2.1.5 การจำแนกตามหน้าที่การทำงาน (Function Categories)

การทำงานของวิชั่นแมชชีนได้แก่ การวัดโดยไม่มีการสัมผัสเพื่อการประยุกต์ใช้งานทางอุตสาหกรรมที่ต้องการความเร็วและต้องการความเชื่อถือได้, การตรวจสอบ เพื่อประกันคุณภาพของผลิตภัณฑ์ที่ออกมาจากกระบวนการผลิต, การบ่งชี้ โดยการพิจารณาฉลากระบุโดยอ่านสัญลักษณ์บนวัตถุ, การทำความเข้าใจ โดยพิจารณาจุดเด่นของวัตถุเพื่อระบุลักษณะเฉพาะตัว, หาตำแหน่ง โดยการหาตำแหน่งที่แม่นยำโดยสัมพันธ์กับการทำความเข้าใจวัตถุ และการติดตาม แบบเวลาจริงโดยการระบุตำแหน่งของวัตถุอย่างต่อเนื่อง

2.2 เทคโนโลยี FPGA (Field Programmable Gate Arrays)

2.2.1 กล่าวนำ

FPGA เป็นชิพ VLSI ซึ่งมีหน้าที่พิเศษส่วนมากคล้ายกับ ASIC อื่นๆ อย่างไรก็ดีสำหรับ FPGA การทำงานถูกโปรแกรมโดยผู้ใช้ซึ่งโดยทั่วไปมักจะเป็นผู้ออกแบบ หลังจากที่ได้โปรแกรมไปครั้งหนึ่งแล้วชิพ (เรียกว่าส่วน FPGA) จะกลายเป็นชิพซึ่งมีจุดประสงค์เฉพาะซึ่งตัว FPGA ถูกออกแบบและโปรแกรมสำหรับจุดประสงค์นั้น

ส่วน FPGA บรรจุด้วยชิ้นส่วนตรรกะที่โปรแกรมได้, เส้นทางเชื่อมต่อระหว่างกันที่โปรแกรมได้, และส่วนต่ออินพุท/เอาท์พุทที่โปรแกรมได้ซึ่งเชื่อมต่อส่วน FPGA เข้ากับวงจร อิเลกทรอนิกส์ซึ่งอยู่ภายนอกตัวถังดูรูปที่ 2.1 ทั้งหมดนี้สามารถโปรแกรมได้โดยวิศวกรภาคสนามโดยไม่ต้องการการเข้าถึงหรือการรองรับจากผู้ผลิต ผู้ใช้เพียงแต่ซื้อ FPGA ว่างมาจากผู้ขาย ตัว FPGA เองถูกสร้างมาสำหรับการกำหนดเองอย่างเต็มที่

FPGA ว่างอันหนึ่งมีราคาสูงกว่า Gate Arrays สมมูลกัน แต่ถ้ามีการค้นพบข้อผิดพลาดใดหลังจากที่ได้โปรแกรม FPGA ไปแล้วจะสามารถโปรแกรมส่วน FPGA ใหม่ได้ด้วยการออกแบบซึ่งแก้ไขแล้ว ทำให้ได้

ราคาที่ต่ำมากและสามารถทำได้ทันที ในทางตรงข้ามการแก้ไขข้อผิดพลาดของชิพที่ผลิตเองและสามารถกำหนดได้ทั้งบางส่วนหรือทั้งหมดหรือ Gate Arrays ต้องการการตรวจสอบข้อผิดพลาดและการนำกลับไปแก้ไขใหม่หลายครั้ง

ก่อนที่จะมีการคิดค้น FPGA ขึ้นมานั้นการออกแบบเหล่านี้ไม่สามารถสร้างได้จริง หรืออาจจะเลียนแบบการทำงานโดยใช้ไมโครคอนโทรลเลอร์ (Microcontroller) หรืออุปกรณ์ประเภท SSI ซึ่งต้องการกำลังงานจำนวนมาก และไม่มีความน่าเชื่อถือ (นั่นคือสามารถทำงานล้มเหลวได้หลังจากปฏิบัติงานไปได้ระยะเวลาหนึ่ง) และไม่ทนทาน (นั่นคือไม่สามารถทำงานได้ภายใต้ภาวะแวดล้อมที่มีการกระทบกระเทือนทางกลและทางไฟฟ้า) และต้องการชิ้นส่วนติดตั้งขนาดใหญ่ (ในนัยของพื้นที่บนบอร์ด, กล่องบรรจุ, หรือวงจรย่อย) เมื่อเปรียบเทียบกับการออกแบบที่สมมูลกันซึ่งสร้างบนชิพเดี่ยว ราคาค่าผลิตสำหรับชิพ (ซึ่งมีราคาสูงขึ้นเรื่อยๆทุกครั้งที่ชิพถูกสร้างขึ้นโดยไม่สามารถเชื่อถือได้และไม่ผ่านการตรวจสอบ) น้อยกว่าผลิตภัณฑ์ซึ่งตั้งบนพื้นฐานของไมโครคอนโทรลเลอร์หรือ SSI มากๆ ซึ่งมีการออกแบบในแนวคิดสำคัญเหมือนกัน

ดังนั้น FPGA รวมข้อดีของการสร้างในระดับชิพซึ่งมีความยืดหยุ่นของการออกแบบที่ตั้งอยู่บนพื้นฐานของไมโครคอนโทรลเลอร์ โดยไม่ต้องมีการตรวจสอบข้อผิดพลาดและส่งกลับไปแก้ไขซึ่งมีในการออกแบบแบบกำหนดเองและ Gate Arrays เมื่อผู้ออกแบบส่วนมากมีความเชื่อมั่นในแนวทางนี้ ตลาด FPGA ถูกคาดหวังให้ขยายตัวอย่างต่อเนื่อง ในขณะที่ไมโครคอนโทรลเลอร์และการออกแบบแบบกำหนดเองทั้งหมดอยู่ในช่วงชะลอการเติบโต

บล็อกตรรกะแบบโปรแกรมได้ (Programmable Logic Block) ใน FPGA สามารถสร้างได้เป็นทั้งการดำเนินการทางตรรกะแบบการรวมกันโดยไม่คำนึงถึงลำดับ (Combination) และแบบลำดับ (Sequential) และแต่ละบล็อกสามารถโปรแกรมได้อิสระต่อกัน การเชื่อมต่อที่สามารถโปรแกรมได้เสนอความต้านทานเปิด (ON-resistance) มากกว่าที่จะเป็นการเชื่อมต่อทางโลหะซึ่งใช้ในการผลิตแบบกำหนดเองทั้งหมดทำให้ทำงานได้ช้ากว่าชิพที่ออกแบบเองทั้งหมด บล็อกอินพุท/เอาท์พุทสามารถเป็นกันชนสัญญาณไฟฟ้าทั้งขาเข้า, ขาออก, หรือทั้งสองทิศทาง และยังสามารถทำให้เป็นสถานะความต้านทานสูงได้ โดยที่ไม่ต้องการการทำงานของวงจรอื่นที่ต่ออยู่ด้วย ลักษณะเด่นเหล่านี้สามารถโปรแกรมได้โดยผู้ใช้งานและสามารถออกแบบสิ่งที่คิดค้นใหม่ได้หลายอย่างโดยใช้เครื่องมือการออกแบบระดับสูงซึ่งแปลงข้อกำหนดทางพฤติกรรมที่ป้อนโดยผู้ออกแบบในภาษาที่เหมาะสมในลักษณะรูปแบบของบิตซึ่งสามารถส่งไปยัง FPGA เพื่อที่จะสร้างชิ้นส่วนที่ทำหน้าที่เฉพาะอย่างดังเช่นชิพ ASIC เครื่องมือเหล่านี้มีราคาไม่สูงนัก (เมื่อเปรียบเทียบกับเครื่องมือที่ใช้กับอุปกรณ์ประเภทกำหนดเองและ Gate Arrays) และมักจะทำงานบนเครื่องคอมพิวเตอร์ส่วนบุคคลที่มีราคาไม่แพง

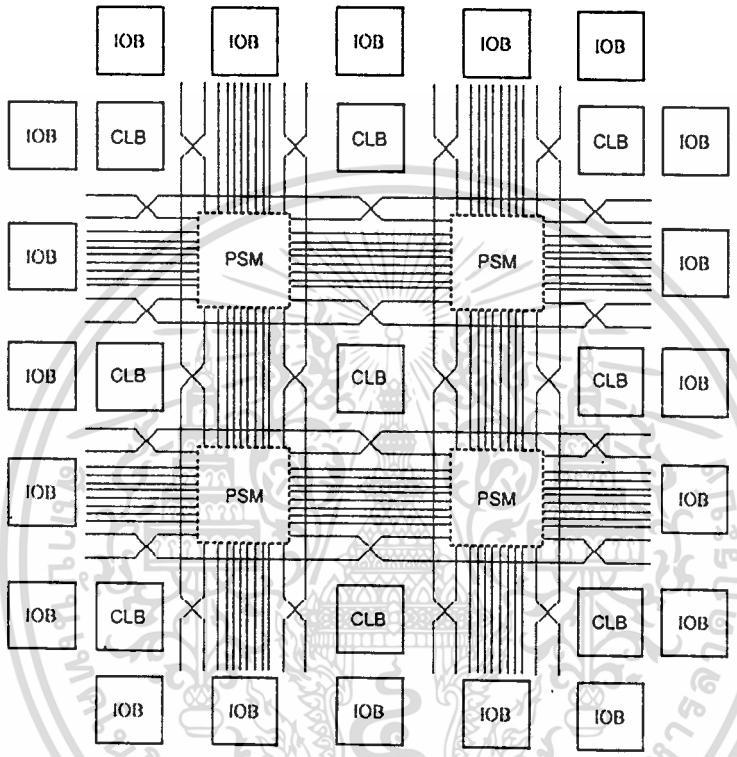
ชนิดที่สำคัญของเทคโนโลยีแบ่งออกได้เป็น

- FPGA ที่อาศัย Static RAM (RAM)
- FPGA ที่อาศัย Anti-fuse

FPGA ที่อาศัย SRAM มีบล็อกตรรกที่ค่อนข้างซับซ้อน โปรแกรมที่ป้อนเข้าไปถูกบรรจุอยู่ภายในโดยใช้ SRAM ซึ่งควบคุมทรานซิสเตอร์ส่งผ่านที่ขนถ่ายสัญญาณไฟฟ้าภายใน FPGA สำหรับโครงงานนี้ใช้เทคโนโลยี FPGA ชนิดนี้

ชิ้นส่วนโปรแกรมซึ่งเริ่มต้นนำกระแสเมื่อถูกทำลายถูกเรียกว่า Anti-fuse ซึ่งตรงกันข้ามกับฟิวส์ประเภทที่หยุดนำกระแสเมื่อถูกทำลาย สำหรับ FPGA ที่อาศัย Anti-fuse มี Anti-fuse เล็กๆ ที่เชื่อมต่อ บล็อกตรรกที่โปรแกรมได้และบล็อกอินพุท/เอาท์พุท

แต่ละเทคโนโลยีมีข้อดีและข้อด้อยของตัวเองซึ่งทั้งสองก่อให้เกิดการพัฒนาของ VLSI ที่รุดหน้า เพิ่มศักยภาพของแนวความคิดของผู้ใช้ และคืนมูลค่ากลับไปสู่ผู้ใช้เอง ซึ่งจะลดราคาของผลิตภัณฑ์ที่สร้างจากผู้ใช้งาน FPGA



รูปที่ 2.1 สถาปัตยกรรมของ FPGA

2.2.2 FPGA ตระกูล XC4000E และ XC4000X

FPGA (Filed Programmable Gate Arrays) อนุกรม XC4000 ของบริษัท Xilinx เป็น FPGA ประสิทธิภาพสูง และความจุสูง ซึ่งให้ข้อดีของ CMOS VLSI (Very Large Scale Integrated circuit) แม้ว่ายังคงหลีกเลี่ยงค่าใช้จ่ายเริ่มต้น วัฏจักรการพัฒนาที่นาน และความเสี่ยงแฝง ของ เกทอะเรย์ (Gate Array) แบบดั้งเดิม

FPGA ดังกล่าวประกอบด้วย ความคล่องตัวทางสถาปัตยกรรม หน่วยความจำแบบ RAM ที่ฝังตัวอยู่ในชิพซึ่งแบบ สองพอร์ท (Dual Port) และมีการทริกโดยใช้ขอบ ความเร็วที่เพิ่มขึ้น และมีทรัพยากรในการจัดหาเส้นทางมากมาย นอกจากนี้ยังมีซอฟต์แวร์ใช้ในการออกแบบที่ซับซ้อน ความจุ และสมรรถนะสูง อนุกรม XC4000E และ XC4000 ประกอบด้วย FPGA 20 เบอร์ ดังแสดงในตาราง 2.1

Device	Logic Cells	Max Logic Gates (NO RAM)	Max RAM Bits (No Logic)	Typical Gate Range (Logic and RAM)	CLB Matrix	Total CLB	Number of Flip-Flops	Max User I/O
XC4003E	238	3000	3200	2000-5000	10X10	100	360	80
XC4005E/XL	466	5000	6272	3000-9000	14X14	196	616	112
XC4006E	608	6000	8192	4000-12000	16X16	256	768	128
XC4008E	770	8000	10368	6000-15000	18X18	324	936	144
XC4010E/XL	950	10000	12800	7000-20000	20X20	400	1120	160
XC4013E/XL	1368	13000	18432	10000-30000	24X24	576	1536	192
XC4020E/XL	1862	20000	25088	13000-40000	28X28	784	2016	224
XC4025E	2432	25000	32768	15000-45000	32X32	1024	2560	256
XC4028EX/XL	2432	28000	32768	18000-50000	32X32	1024	2560	256
XC4036EX/XL	3078	36000	41472	22000-65000	36X36	1296	3168	288
XC4044XL	3800	44000	51200	27000-80000	40X40	1600	3840	320
XC4052XL	4598	52000	61952	33000-100000	44X44	1936	4576	352
XC4062XL	5472	62000	73728	40000-130000	48X48	2304	5376	384
XC4085XL	7448	85000	100352	55000-180000	56X56	3136	7168	448

ตารางที่ 2.1 FPGA ในตระกูล XC4000E และ XC4000X

2.2.3 ลักษณะของ FPGA อนุกรม XC4000E/XL

อนุกรม XC4000 ถูกสร้างขึ้นด้วยสถาปัตยกรรมแบบโปรแกรมได้ของ CLB (Configurable Logic Block) ที่เที่ยงตรงและยืดหยุ่น มีการเชื่อต่อระหว่างกันด้วยลำดับชั้นของทรัพยากรการจัดหาเส้นทางที่คล่องตัวและมีประสิทธิภาพ และแวดล้อมด้วย IOB (Input/Output Block) ที่โปรแกรมได้โดยรอบ ซึ่งประกอบกันเป็นทรัพยากรการจัดหาเส้นทางที่กว้างขวางเพื่อสร้างเป็นรูปแบบการเชื่อมต่อที่ซับซ้อน

อุปกรณ์เหล่านี้สามารถแก้ไขดัดแปลงได้โดยถ่ายข้อมูลสัณฐานเข้าไปภายในชิ้นส่วนหน่วยความจำภายใน FPGA สามารถอ่านข้อมูลด้วยตัวเองได้ทั้งจาก PROM ภายนอกแบบขนานหรืออนุกรม (Master Mode) หรือข้อมูลสัณฐานสามารถเขียนเข้าไปใน FPGA จากอุปกรณ์ภายนอก (Slave and Preipheral Mode)

เนื่องจาก FPGA ของ Xilinx สามารถโปรแกรมใหม่ได้ไม่จำกัดจำนวนครั้ง และสามารถใช้ในการออกแบบที่มีการปรับเปลี่ยนซึ่งฮาร์ดแวร์มีการเปลี่ยนแปลงแบบพลวัต หรือเมื่อฮาร์ดแวร์มีการปรับตัวสำหรับการประยุกต์ใช้งานของผู้ใช้ที่แตกต่างกัน

2.2.4 กลุ่มโครงร่างพื้นฐาน (Basic Buliding Block)

FPGA ของ Xilinx ประกอบด้วยชิ้นส่วนหลักที่โปรแกรมได้สองส่วน ก่าวคือ กลุ่มตรรกะแบบโปรแกรมได้ (Configurable Logig Block : CLB) และ กลุ่มอินพุทเอาท์พุท (Input/Output Block)

– CLB จัดเตรียมชิ้นส่วนการทำงานสำหรับสร้างตรรกะของผู้ใช้

– IOB จัดเตรียมการเชื่อมต่อระหว่างขาของชิ้นส่วนกับเส้นสัญญาณภายใน

นอกจากนี้ยังมีวงจรชนิดอื่นอีก 3 แบบ

– กันชนสามสถานะ (Tristate Buffer : TBUF) ซึ่งขับส่วนที่อยู่บนระนาบอยู่ภายในแต่ละ CLB

– ตัวถอดรหัสขอบกว้าง (Wide Edge Decoder) อยู่รอบนอกของแต่ละอุปกรณ์

– ตัวสร้างกำเนิดความถี่ภายในชิพ

ทรัพยากรการเชื่อมต่อระหว่างกันแบบโปรแกรมได้จัดหาการหาเส้นทางในการเชื่อมต่ออินพุทและเอาท์พุทของชิ้นส่วนที่โปรแกรมได้เหล่านี้ด้วยโครงข่ายที่เหมาะสม

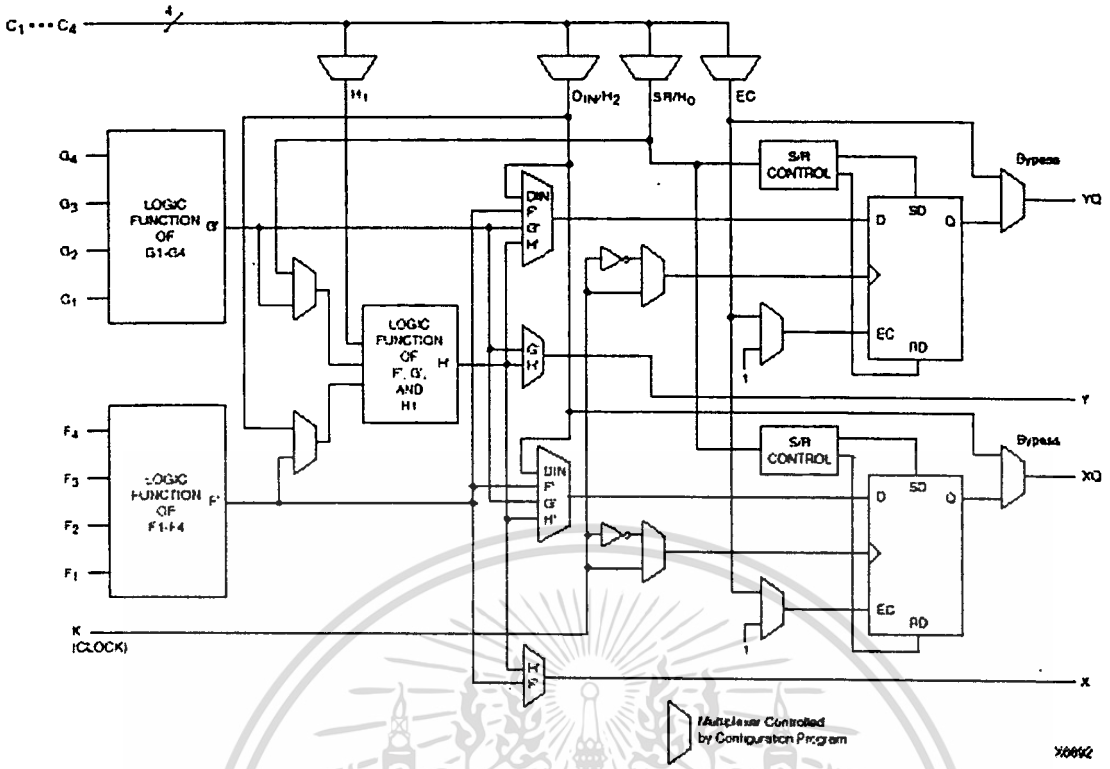
หน้าที่การทำงานของแต่ละกลุ่มสามารถปรับเปลี่ยนได้ระหว่างการจัดสัณฐานโดยการโปรแกรมชิ้นส่วนหน่วยความจำสถิตย์ภายใน ค่าดังกล่าวถูกเก็บไว้ในหน่วยความจำเหล่านี้เพื่อใช้ในการพิจารณาหน้าที่ทางตรรกและการเชื่อมต่อที่สร้างขึ้นใน FPGA ในหัวข้อนี้จะเป็นการอธิบายรายละเอียดของกลุ่มการทำงานเหล่านี้โดยละเอียด

2.2.4.1 กลุ่มตรรกะแบบจัดสัณฐานได้ (Configurable Logic Block: CLB)

กลุ่มตรรกะแบบโปรแกรมได้สร้างตรรกะส่วนใหญ่ใน FPGA ชิ้นส่วนหลักใน CLB แสดงในรูปที่ 2.2 ตัวกำเนิดฟังก์ชัน 4 อินพุท (4-Input Function Generator) F และ G เสนอความคล่องตัวได้ไม่จำกัด การรวมกันของฟังก์ชันทางตรรกส่วนมากต้องการไม่เกิน 4 อินพุท อย่างไรก็ดี ตัวกำเนิดฟังก์ชันที่สามคือ H ได้ถูกเตรียมขึ้น ตัวกำเนิดฟังก์ชัน H มีสามอินพุท อินพุทเหล่านี้ไม่ว่าจะเป็น 0 1 หรือ 2 สามารถเป็นเอาท์พุทของ F และ G ส่วนอินพุทที่เหลือมาจากภายนอก CLB ดังนั้น CLB สามารถสร้างฟังก์ชันได้ถึง 9 อินพุท เช่นการตรวจสอบพาริตี้ (Parity) หรือการเปรียบเทียบค่าแบบขยายของสี่อินพุท สองกลุ่ม

แต่ละ CLB ประกอบด้วยชิ้นส่วนบรรจุที่สามารถใช้เก็บเอาท์พุทของตัวกำเนิดฟังก์ชัน อย่างไรก็ตามชิ้นส่วนบรรจุและตัวกำเนิดฟังเหล่านี้สามารถใช้งานเป็นอิสระต่อกันได้ ชิ้นส่วนบรรจุเหล่านี้สามารถจัดสัณฐานให้เป็น Flip-Flop ได้ในทั้งอุปกรณ์ XC4000E และ XC4000XL ในส่วนของ XC4000X ยังสามารถเลือกได้ว่าจะจัดสัณฐานให้เป็น Latch ได้ด้วย DIN สามารถใช้เป็นอินพุทโดยตรงไปยังหนึ่งในสองชิ้นส่วนจัดเก็บทั้งสอง H1 สามารถขับอีกส่วนหนึ่งผ่านทางตัวกำเนิดฟังก์ชัน H เอาท์พุทของตัวกำเนิดฟังก์ชันยังสามารถขับเอาท์พุทเป็นอิสระจากเอาท์พุทของชิ้นส่วนจัดเก็บ ความคล่องตัวนี้เพิ่มความจุทางตรรกะและการจัดหาเส้นทางที่ไม่ซับซ้อน 13 CLB อินพุทและ 4 CLB เอาท์พุท จัดเตรียมการเข้าถึงตัวกำเนิดฟังก์ชันและชิ้นส่วนจัดเก็บ อินพุทและเอาท์พุทเหล่านี้เชื่อมต่อไปยังทรัพยากรในการเชื่อมต่อระหว่างกันภายนอกกลุ่ม แบ่งออกเป็นส่วนประกอบย่อยๆ ได้แก่

ตัวกำเนิดฟังก์ชัน (Function Generator) ประกอบด้วยอินพุททั้งสี่ที่เป็นอิสระต่อกันได้ถูกจัดเตรียมให้ในแต่ละตัวกำเนิดฟังก์ชันทั้งสอง (F1-F4 และ G1-G4) ตัวกำเนิดฟังก์ชันแต่ละตัวเหล่านี้ ซึ่งมีเอาท์พุทได้แก่ F' และ G' มีความสามารถในการสร้างฟังก์ค่าความจริง (Boolean) ที่ถูกนิยามใดๆที่มี 4 อินพุท ตัวกำเนิดฟังก์ชันถูกสร้างขึ้นเป็นหน่วยความจำตารางเสาะหา (Look-up Table) ดังนั้นความล่าช้าในการส่งผ่านขึ้นอยู่กับการสร้างฟังก์ชัน



รูปที่ 2.2 แผนผังอย่างง่ายของ CLB อนุกรม XC4000 (ไม่แสดง RAM และตรรกะ)

ตัวกำเนิดฟังก์ชันที่สาม ได้แก่ H' สามารถสร้างฟังก์ชันค่าความจริงด้วยอินพุททั้งสาม สองในสามอินพุทเหล่านี้สามารถเลือกได้ว่าจะเป็นเอาท์พุทของ F' และ G' หรืออีกทางเลือกหนึ่ง หนึ่งหรือสองของอินพุทเหล่านี้สามารถมาจากภายนอก CLB (H1, H2) อินพุทที่สาม จะต้องมาจากข้างนอกกลุ่ม (H1) สัญญาณจกาตัวกำเนิดฟังก์ชันสามารถออกจาก CLB ทางเอาท์พุททั้งสอง F' หรือ H'สามารถเชื่อต่อไปยังเอาท์พุท X G' หรือ H' สามารถเชื่อมต่อไปยังเอาท์พุท Y

CLB สามารถใช้ในการสร้างฟังก์ชันต่อไปนี้

- ฟังก์ชันใดๆที่มีสี่ตัวแปร รวมกับอีกสองฟังก์ชันใดๆที่มีสี่ตัวตัวแปรที่ไม่สัมพันธ์กัน รวมกับอีกสามฟังก์ชันใดๆที่มีสามตัวแปรที่ไม่เกี่ยวเนื่องกัน
- ฟังก์ชันใดๆที่มีห้าตัวแปร
- ฟังก์ชันใดๆที่มีสี่ตัวแปรร่วมกับบางฟังก์ชันที่มีหกตัวแปร
- บางฟังก์ชันที่มีเก้าตัวแปร

การสร้างฟังก์ชันกว้างในกลุ่มเดียวลดทั้งจำนวนของกลุ่มที่ต้องการและความล่าช้าในเส้นทางผ่านของสัญญาณ ยังผลให้มีการเพิ่มของความจุและความเร็ว ความคล่องตัวของตัวกำเนิดฟังก์ชัน CLB ปรับปรุงความเร็วของระบบอย่างมีนัยสำคัญ ยิ่งไปกว่านั้นซอฟต์แวร์เครื่องมือออกแบบสามารถทำงานร่วมกับแต่ละตัวกำเนิดฟังก์ชันได้อย่างเป็นอิสระ ความยืดหยุ่นนี้ปรับปรุงการใช้งานของหน่วย

ฟลิปฟลอป (Flip-Flop) CLB สามารถผ่านเอาท์พุทรวมไปยังโครงข่ายที่เชื่อต่อระหว่างกันได้ แต่ก็ยังคงสามารถเก็บผลลัพธ์รวมหรือข้อมูลที่เข้ามาอื่นๆในฟลิปฟลอปหนึ่งหรือสองตัว และเชื่อมต่อเอาท์พุทของฟลิปฟลอปเหล่านั้นไปยังโครงข่ายที่เชื่อต่อกันได้เช่นกัน ฟลิปฟลอปชนิด D แบบทริกโดยใช้ขอบสองตัวมีอินพุทสัญญาณนาฬิกา (K) และตัวยอมให้สัญญาณนาฬิกาผ่าน (EC) ร่วมกัน อินพุทสัญญาณนาฬิกาตัวใดตัวหนึ่งหรือทั้งสองตัวสามารถทำให้ผ่านได้แบบถาวรก็ได้

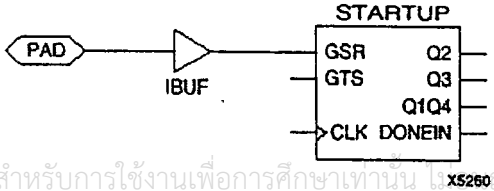
ตัวค้างข้อมูล หรือแลตช์ (Latch : XC4000X only) ชิ้นส่วนจัดเก็บของ CLB สามารถจัดสัณฐานให้เป็นแลตช์ แลตช์ทั้งสองมีอินพุทสัญญาณนาฬิกาและตัวยอมให้สัญญาณนาฬิกาผ่านร่วมกัน

อินพุทสัญญาณนาฬิกา (Clock Input) ฟลิปฟลอปแต่ละตัวสามารถทริกบนขอบขาขึ้นหรือขาลงของสัญญาณนาฬิกา ขอสัญญาณนาฬิกาใช้ร่วมกันโดยชิ้นส่วนจัดเก็บทั้งสอง อย่างไรก็ดีสัญญาณนาฬิกาถูกกลับค่าอย่างเป็นอิสระสำหรับแต่ละชิ้นส่วนจัดเก็บ ตัวกลับค่าแต่ละตัวที่อยู่ที่อินพุทสัญญาณนาฬิกาถูกรับเอาไว้ใน CLB อย่างอัตโนมัติ

ตัวยอมให้สัญญาณนาฬิกาผ่าน (Clock Enable) สัญญาณยอมให้สัญญาณนาฬิกาผ่าน (EC) ทำงานที่สถานะสูง (High) ขา EC ถูกใช้ร่วมกันสำหรับชิ้นส่วนจัดเก็บทั้งสอง ถ้าไม่ต่อขาทั้งสองสัญญาณนาฬิกาจะผ่านไปได้โดยปริยายในขณะทำงาน EC ไม่สามารถกลับค่าได้ภายใน CLB

เซ็ตและรีเซต (Set/Reset) อินพุทแบบไม่เข้าจังหวะกัน (Asynchronous) ของชิ้นส่วนจัดเก็บ (SR) สามารถจัดสัณฐานให้เป็นได้ทั้งเซ็ตและรีเซต ทางเลือกการจัดสัณฐานนี้พิจารณาสถานะที่ฟลิปฟลอบปฏิบัติงานหลังจากการจัดสัณฐาน และยังพิจารณาผลกระทบของ พัลส์เซ็ตและรีเซตรวม (Global Set/Reset) ระหว่างการทำงานตามปกติ และผลกระทบของพัลส์ที่ขา SR ของ CLB เซ็ตและรีเซตทั้งสามทำหน้าที่สำหรับฟลิปฟลอปเดี่ยวใดๆถูกควบคุมโดยบิตข้อมูลบิตสัณฐานเดียวกัน สถานะเซ็ตและรีเซ็ตสามารถระบุได้อิสระต่อกันสำหรับแต่ละฟลิปฟลอป อินพุทนี้ยังสามารถยับยั้งได้อิสระต่อกันสำหรับแต่ละฟลิปฟลอปอีกด้วย สถานะเซ็ตและรีเซ็ตถูกระบุโดยการใช้สมบัติของ INIT หรือโดยการจัดวางสัญญลักษณ์ฟลิปฟลอปเซ็ตหรือรีเซ็ตในห้องสมุดที่เหมาะสม SR ทำงานที่สถานะสูง และไม่สามารถกลับค่าได้ภายใน CLB

เซ็ตรีเซ็ตรวม (Global Set/Reset) สายเซ็ต/รีเซ็ตที่แยกออกมาต่างหาก (ไม่ได้แสดงในรูปที่ 2.2) ตั้งหรือล้างชิ้นส่วนจัดเก็บระหว่างการเริ่มต้นจ่ายไฟ (Power-Up) การจัดสัณฐานใหม่ หรือเมื่อข่ายเซ็ตเฉพาะถูกขับให้ทำงาน ข่ายร่วมนี้ (GSR) ไม่ได้เกี่ยวข้องกับทรัพยากรจัดหาเส้นทางอื่น แต่ใช้โครงข่ายกระจายเฉพาะ แต่ละฟลิปฟลอปถูกจัดสัณฐานให้เป็นได้ทั้ง เซ็ตและรีเซตรวมในทำนองเดียวกันกับสัญญาณเซ็ตและรีเซตท้องถิ่น (SR) ถูกระบุ ดังนั้นถ้าฟลิปฟลอปถูกเซ็ตโดย SR จะถูกเซ็ตหรือรีเซ็ตด้วย GSR ด้วย ในทำนองเดียวกันฟลิปฟลอปที่ถูกรีเซ็ตจะถูกรีเซ็ตทั้งโดย SR และ GSR



รูปที่ 2.3 สัญลักษณ์แผนทางไฟของเซ็ต/รีเซ็ตรวม

GSR สามารถถูกขับจากขาใดๆที่โปรแกรมโดยผู้ใช้ให้เป็นอินพุทของเซ็ต/รีเซ็ตรวมได้ เพื่อที่จะใช้ข่ายรวมดังกล่าว จัดวางสัญญลักษณ์ PAD และกันชนอินพุทในแผนทางไฟฟ้าหรือรหัส HDL ขับขา GSR ด้วยสัญญลักษณ์ STARTUP ดังแสดงในรูปที่ 2.3 ตำแหน่งของขาที่เฉพาะเจาะจงสามารถกำหนดให้กับอินพุทนี้ได้โดยใช้สมบัติหรืออ้างถึง LOC ดังเช่น PAD ที่ผู้ใช้สามารถโปรแกรมได้อื่นๆตัวกลับค่าสามารถเป็นทางเลือกที่จะแทรกหลังกันชนอินพุทเพื่อที่จะกลับค่าที่รับได้ของสัญญาณเซ็ตและรีเซ็ตรวม อีกทางเลือกหนึ่ง GSR สามารถขับจากโหนด (Node) ภายในใดๆ

อินพุทและเอาท์พุทของข้อมูล (Data Input and Output) แหล่งกำเนิดข้อมูลอินพุทของชิ้นส่วนจัดเก็บสามารถโปรแกรมได้ และถูกขับโดยฟังก์ชัน F' G' และ H' ใดๆ หรือโดยอินพุทไปยังกลุ่มโดยตรง (Direct In Block : DIN) ฟลิปฟลอปหรือแลตช์ขับเอาท์พุท XQ และ YQ ของ CLB เส้นทางป้อนผ่านแบบความเร็วสูงสองเส้นทางแสดงดังรูปที่ 2.2 มัลติเพลกเซอร์ (Multiplexer) แบบสองไปหนึ่งบนแต่ละเอาท์พุท XQ และ YQ เลือกระหว่างเอาท์พุทของชิ้นส่วนจัดเก็บหรืออินพุทควบคุมใดๆ เส้นทางข้ามผ่าน (Bypass) นี้บางครั้งถูกใช้โดยตัวจัดหาเส้นทางอัตโนมัติเพื่อที่จะจัดกำลังของสัญญาณภายในใหม่

สัญญาณควบคุม (Control Signal) มัลติเพลกเซอร์ใน CLV ทำแผนที่ สำหรับสี่อินพุทควบคุม (C1-C4 ในรูปที่ 2.2) ไปยังสี่สัญญาณควบคุมภายใน (H1, DIN/H2, SR/H0, และ EC) อินพุทใดๆเหล่านี้สามารถขับสัญญาณควบคุมภายในทั้งสี่ใดๆได้ เมื่อฟังก์ชันตรรกะถูกจัดให้ทำงาน อินพุททั้งสี่คือ

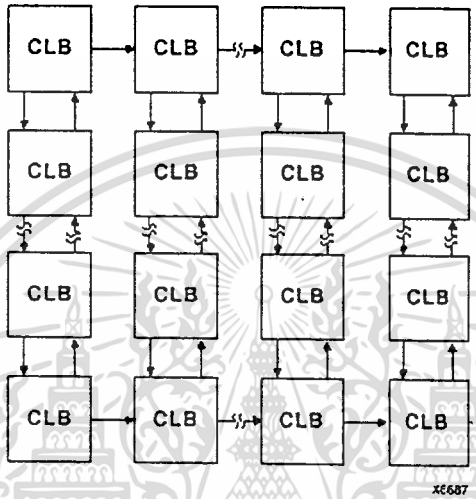
- EC - สัญญาณยอมให้สัญญาณนาฬิกาผ่าน
- SR/H0 - สัญญาณเซ็ตและรีเซ็ตแบบไม่เข้าจังหวะ หรือตัวกำเนิดฟังก์ชัน H อินพุท 0
- DIN/H2 - เส้นทางโดยตรงหรือตัวกำเนิดฟังก์ชัน H อินพุท 2
- H1 - ตัวกำเนิดฟังก์ชัน H อินพุท 1

เมื่อฟังก์ชันหน่วยความจำถูกจัดให้ทำงาน อินพุททั้งสี่คือ

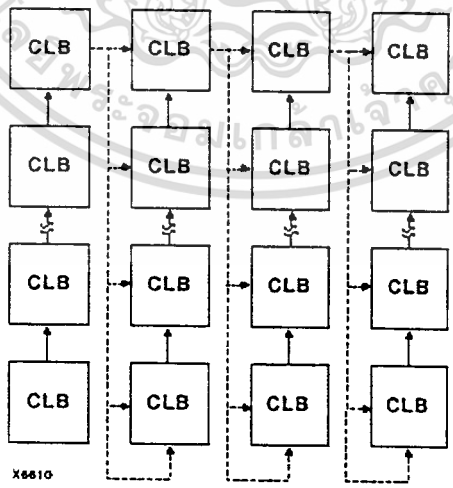
- EC - สัญญาณยอมให้สัญญาณนาฬิกาผ่าน
- WE - สัญญาณยอมให้มีการเขียน
- D0 - อินพุทข้อมูลไปยัง ตัวกำเนิดฟังก์ชัน F และหรือ G
- D1 - อินพุทข้อมูลไปยัง ตัวกำเนิดฟังก์ชัน G (ในแบบ 16x1 และ 16x2) หรือบิตที่ 5 ของตำแหน่ง (ในแบบ 32x1)

ตรรกะตัวทดความเร็วสูง (Fast Carry Logic) ตัวกำเนิดฟังก์ชัน F และ G ใน CLB แต่ละตัวบรรจุตรรกะทางพีชคณิตไว้สำหรับการสร้างสัญญาณตัวทดและตัวยืมความเร็วสูง เอาท์พุทที่เพิ่มขึ้นมานี้ส่งผ่านไปบนตัวกับเนิดฟังก์ชันของ CLB ที่ติดกัน ลูกโซ่ตัวทดเป็นอิสระต่อทรัพยากรการจัดหาเส้นทางตามปกติ ตรรกะตัวทดความเร็วสูงโดยเฉพาะนี้เพิ่มประสิทธิภาพและสมรรถนะของตัวบวก (Adder) ตัวลบ (Subtractor) ตัวสะสม (Accumulator) ตัวเปรียบเทียบ (Comparator) และตัวนับ (Counter) ทำให้สามารถประยุกต์ใช้ในการใช้งานที่ไม่สามารถทำได้โดย FPGA รุ่นก่อนหน้าซึ่งมีความเร็วไม่เพียงพอ อาทิเช่นการใช้งานโดยทั่วไปสองประเภทได้แก่ การคำนวนค่าเลื่อน (Offset) ของตำแหน่งความเร็วสูงในไมโครโปรเซสเซอร์หรือในระบบกราฟฟิก และการบวกความเร็วสูงในการประมวลผลสัญญาณดิจิตอล (Digital Signal Processing) ตัวกำเนิดฟังก์ชันสี่อินพุทสองตัวสามารถถูกจัดสัณฐานให้เป็นตัวบวกสองบิตที่มีตัวทดแบบซ่อนภายในซึ่งสามารถ

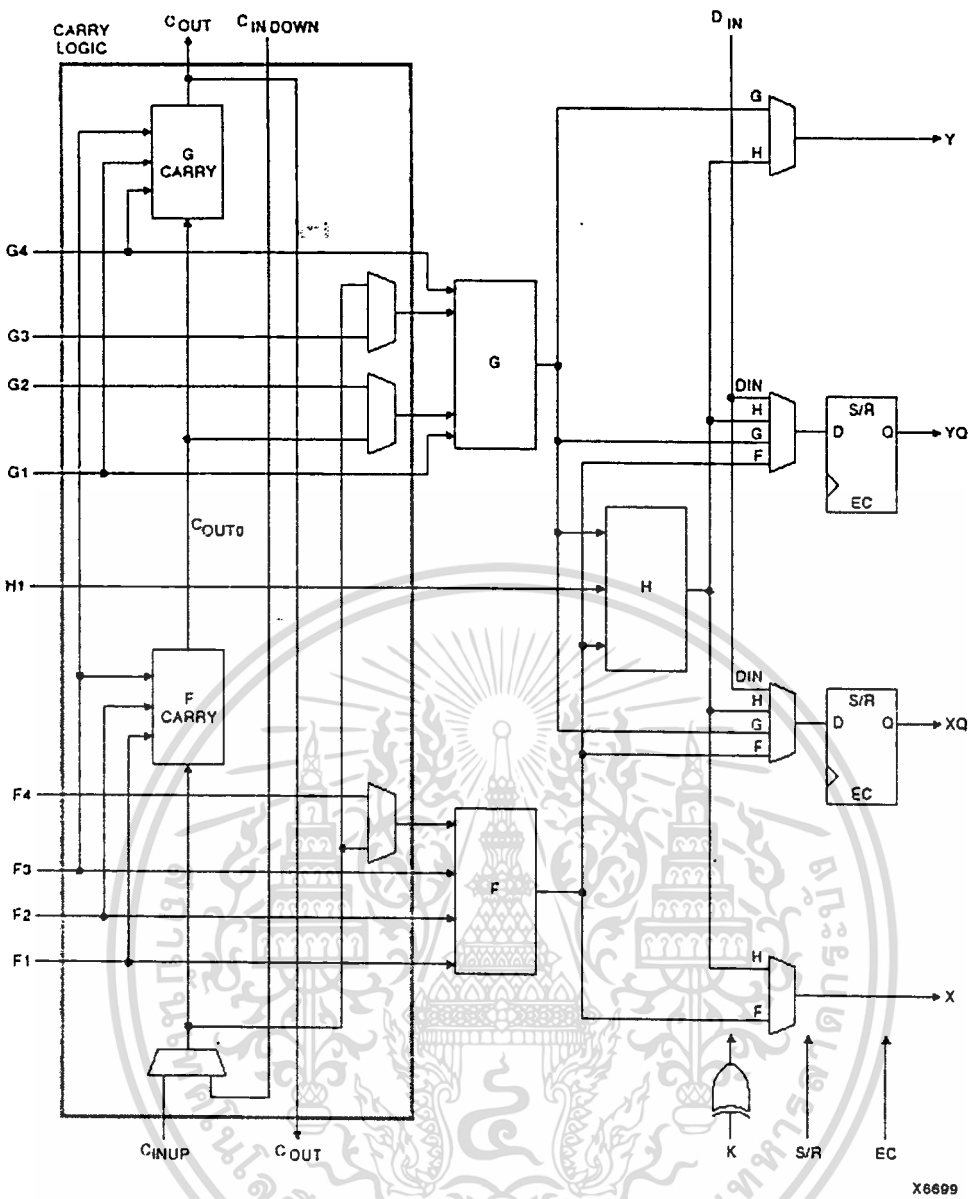
ขยายให้เป็นความยาวใดๆ วงจรตัวทดโดยเฉพาะเจาะจงนี้เร็วและมีประสิทธิภาพมากพอที่จะสามารถละทิ้งวิธีการเพิ่มความเร็วแบบดั้งเดิม เช่นการสร้างและส่งผ่านตัวทด ถึงแม้ว่าจะเป็นที่ระดับ 16 บิต หรือสูงสุดที่ระดับ 32 บิต ตรรกะตัวทดความเร็วสูงนี้เป็นหนึ่งในจุดเด่นที่สำคัญซึ่งมีมากกว่านี้ของอนุกรม XC4000 เพิ่มความเร็วการนับและการคำนวณทางพีชคณิตสูงถึงพิกัด 70 MHz ลูกโซ่ตัวทดในอุปกรณ์ XC4000E สามารถวิ่งขึ้นหรือลงได้ ที่หัวและท้ายสดมภ์ที่ไม่มี CLB อยู่ข้างบนและข้างล่าง ตัวทดถูกส่งผ่านไปทางขวา ดังแสดงในรูปที่ 2.4 เพื่อที่จะปรับปรุงความเร็วในอุปกรณ์ความจุสูงอย่างเช่น XC4000X ซึ่งมีศักยภาพพอสำหรับลูกโซ่ที่ยาวมาก ลูกโซ่ตัวทดจะเดินทางขึ้นบนเท่านั้น ดังแสดงในรูปที่ 2.5



รูปที่ 2.4 เส้นทางเดินของตัวทดใน XC4000E



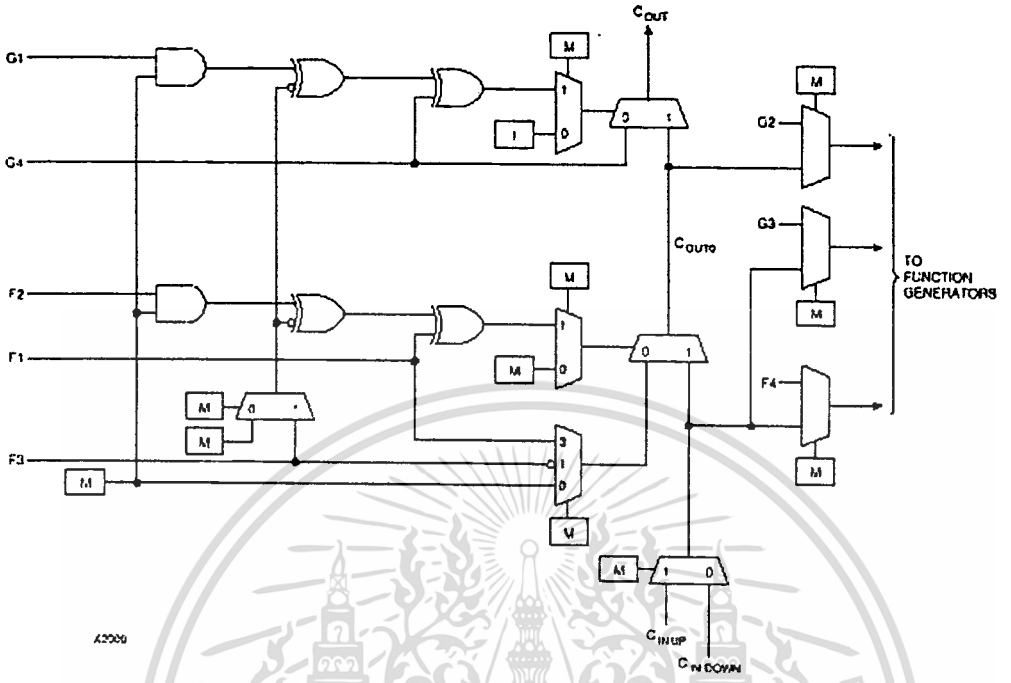
รูปที่ 2.5 เส้นทางเดินของตัวทดใน XC4000X (เส้นประใช้การเชื่อมต่อกันแบบธรรมดา)



รูปที่ 2.6 ตรรกะตัวทดความเร็วสูง (บริเวณที่แรเงาไม่มีใน XC4000X)

ยิ่งไปกว่านั้นการเชื่อมต่อตามมาตรฐานสามารถใช้ในการจัดหาเส้นทางให้กับสัญญาณตัวทดในทิศทางลงได้ด้วย รูปที่ 2.6 แสดง CLB ที่มีตรรกะตัวทดความเร็วสูงเฉพาะเจาะจง ตรรกตัวทดใน XC4000X มีความคล้ายคลึงกันยกเว้นที่มี COUT อยู่ที่ด้านบนเท่านั้นและสัญญาณ CINDOWN ไม่มีดังแสดงในรูปที่ 2.6 ตรรกะตัวทดใช้ตัวถูกดำเนินการ (Operand) และอินพุทควบคุมร่วมกันกับตัวกำเนิดฟังก์ชัน เอาท์พุทตัวทดนี้ต่อไปยังตัวกำเนินฟังก์ชัน ที่เอาท์พุทเหล่านั้นรวมอยู่กับตัวถูกดำเนินการเพื่อที่จะสร้างผลบวก รูปที่ 2.7 แสดงรายละเอียดของตรรกะตัวทดของ XC4000E แผนผังแสดงสิ่งที่บรรจุอยู่ในกรอบชื่อว่า CARRY LOGIC ในรูปที่ 2.6 ตรรกะตัวทดมีความคล้ายกันมาก แต่มัลติเพลกเซอร์ในลูกโซ่ตัวทดส่งผ่านถูกยับยั้งเพื่อลดความล่าช้า ยิ่งไปกว่านั้นในมัลติเพลกเซอร์บนเส้นทาง G4 มีหน่วยความจำที่โปรแกรมได้ 0 อินพุทซึ่งยอมให้ G4 เชื่อมต่อไปยัง COUT ดังนั้น G4 จึงกลายเป็นเส้นทางเริ่มต้นความเร็วสูงที่เพิ่มขึ้นมาสำหรับตัวทดเข้า ตรรกะตัวทดเข้า

สามารถเข้าถึงได้โดยจัดวางสัญญลักษณ์ฐานสองพิเศษ หรือโดยใช้ Xilinx Relationally Placed Macros (RPMs) ซึ่งรวมอยู่แล้วในสัญญลักษณ์เหล่านี้



รูปที่ 2.7 ตรรกะตัวทดเฉพาะโดยละเอียดของ XC4000E

2.2.4.2 กลุ่มอินพุทเอาท์พุท (Input/Output Block : IOB)

กลุ่มอินพุทเอาท์พุทที่ผู้ใช้สามารถจัดสัณฐานได้จัดเตรียมการเชื่อมต่อระหว่างขาภายนอกชิพกับตรรกะภายใน แต่ละ IOB ควบคุมขาหนึ่งของชิพ และสามารถจัดสัณฐานให้เป็นสัญญาณอินพุท เอาท์พุทหรือสองทิศทางได้

รูปที่ 2.8 แสดงผังง่ายๆของ IOB ของ XC4000E สำหรับ IOB ของ XC4000X บรรจุจุดเด่นพิเศษบางประการที่ไม่ได้รวมอยู่ใน IOB ของ XC4000E ฟังก์ชันเหล่านี้ถูกเน้นในแผนผังในรูปที่ 2.9 และจะมีการกล่าวถึงในหัวข้อนี้ จุดเด่นดังกล่าวจะถูกนิยามอย่างเด่นชัด ถ้าไม่มีการระบุเป็นอย่างอื่นจะมีอยู่ในทั้ง อุปกรณ์ XC4000E และ XC4000X ประกอบด้วยส่วนสำคัญดังต่อไปนี้

สัญญาณอินพุท IOB (IOB Input Signal) สองเส้นทางได้แก่ I1 และ I2 ในรูปที่ 2.8 และ 2.9 จะนำสัญญาณอินพุทไปยังอาเรย์ อินพุทยังต่อไปยังรีจิสเตอร์อินพุทที่สามารถโปรแกรมให้เป็นได้ทั้งฟลิปฟลอปที่ทริกโดยใช้ขอบหรือแลตช์ที่ตอบสนองต่อระดับ การเลือกสามารถทำได้โดยวางสัญลักษณ์ที่เหมาะสม ตัวอย่างเช่น IFD เป็นอินพุทฟลิปฟลอปพื้นฐาน (ทริกโดยใช้ขอบขาขึ้น) และ ILD เป็นแลตช์อินพุทพื้นฐาน (ตอบสนองระดับสูง) สามารถสร้างความแตกต่างด้วยการกลับค่าสัญญาณนาฬิกา และการรวมกันของแลตช์ และฟลิปฟลอปบางประการสามารถสร้างได้ภายใน IOB เดียว ดังที่อธิบายไว้ใน XACT Libraries Guide

รีจิสเตอร์อินพุท (Register Input) สัญญาณ I1 และ I2 ที่ออกจากกลุ่มสามารถนำได้ทั้งสัญญาณอินพุทโดยตรงหรือเป็นรีจิสเตอร์ได้ ชิ้นส่วนจัดเก็บที่ฝั่งอินพุทและเอาท์พุทในแต่ละ IOB จะมีอินพุทสัญญาณ

2.2.5.2.1 รูปแบบ Master

รูปแบบ Master สามรูปแบบใช้ตัวกำเนิดความถี่ภายในในการสร้างสัญญาณนาฬิกาจัดสัณฐาน (CCLK) เพื่อที่จะขับอุปกรณ์ Slave และยังสร้างตำแหน่งและ Timing สำหรับ PROM ภายนอกที่บรรจุข้อมูลสัณฐาน

รูปแบบ Master Parallel (Up หรือ Down) สร้างสัญญาณนาฬิกา CCLK และตำแหน่ง PROM และรับข้อมูลเป็นไบต์ขนาน ข้อมูลถูกส่งเรียงเข้าไปใน FPGA ในรูปแบบของกรอบข้อมูล การเลือก Up หรือ Down เป็นการสร้างตำแหน่งเริ่มต้น 0 หรือ 3FFFF เพื่อให้เข้ากันได้กับระบบการอ้างอิงตำแหน่งของไมโครโปรเซสเซอร์ที่ต่างกัน รูปแบบ Master Serial สร้าง CCLK และรับข้อมูลแบบอนุกรมจาก PROM อนุกรม

2.2.5.2.2 รูปแบบ Peripheral

รูปแบบ Peripheral ทั้งสองแบบรับข้อมูลกว้าง 1 ไบต์มาจากบัส สถานะ RDY/BUSY มีไว้เพื่อเป็นสัญญาณตรวจสอบ ในรูปแบบอะซิงโครนัส (Asynchronous) ตัวกำเนิดความถี่ภายในจะสร้าง CCLK จำนวนมากเพื่อถ่ายเรียงข้อมูล CCLK ยังขับอุปกรณ์ Slave อื่นได้อีก ในรูปแบบซิงโครนัส (Asynchronous) สัญญาณนาฬิกาจะถูกส่งมาจากภายนอกไปยัง CCLK เพื่อถ่ายเรียงข้อมูล

2.2.5.2.3 รูปแบบ Slave Serial

ในรูปแบบ Slave Serial FPGA รับข้อมูลการจัดสัณฐานที่ขอบขาขึ้นของ CCLK และหลังจากถ่ายข้อมูลเรียบร้อยแล้ว จะส่งผ่านข้อมูลออกเพิ่มเติมเพื่อเข้าจังหวะอีกครั้งหนึ่ง ที่ขอบขาลงถัดไปของ CCLK อุปกรณ์ Slave หลายตัวสามารถเชื่อมโยงเข้าด้วยกันได้ด้วยอินพุท DIN โดยที่อุปกรณ์สามารถจัดสัณฐานไปพร้อมๆกันได้

2.2.5.3 การตั้งความถี่ CCLK

สำหรับรูปแบบ Master CCLK สามารถสร้างได้ความถี่ใดความถี่หนึ่งใน 2 ความถี่ โดยปริยายอยู่ในแบบช้า ความถี่ตกอยู่ในช่วง 0.5 MHz ถึง 1.25 MHz สำหรับ XC400E และ XC4000EX สำหรับ XC4000XL จะอยู่ในช่วง 0.6 MHz ถึง 1.8 MHzใน CCLK รูปแบบเร็ว ความถี่อยู่ในช่วง 4 MHz และ 10 MHz สำหรับ XC4000EX และอยู่ในช่วง 5 MHz ถึง 15 MHz สำหรับ XC4000XL ความถี่ถูกเลือกโดยใช้ทางเลือกเมื่อใช้โปรแกรมสร้างสายบิต ถ้า Master อนุกรม XC4000 ขับ Slave ตระกูล XC3000 หรือ XC2000 ต้องใช้ CCLK รูปแบบช้า ยิ่งไปกว่านั้น Master XC4000XL ถ้าขับ XC4000E หรือ XC4000EX ควรใช้ CCLK แบบช้า โดยปริยายจะเป็นแบบช้า

2.2.5.4 รูปแบบสตรีมข้อมูล (Data Stream Format)

รูปแบบสตรีมข้อมูล (บิตสตรีม) เหมือนกันสำหรับการจัดสัณฐานทุกรูปแบบ รูปแบบของสตรีมข้อมูลแสดงในตารางที่ 2.3 ข้อมูลบิตอนุกรมถูกอ่านจากซ้ายไปขวา และไบต์ขนานจะถูกประกอบขึ้นจากบิตเหล่านี้ โดยที่บิตแรกของแต่ละไบต์จะกำหนดให้เป็น D0

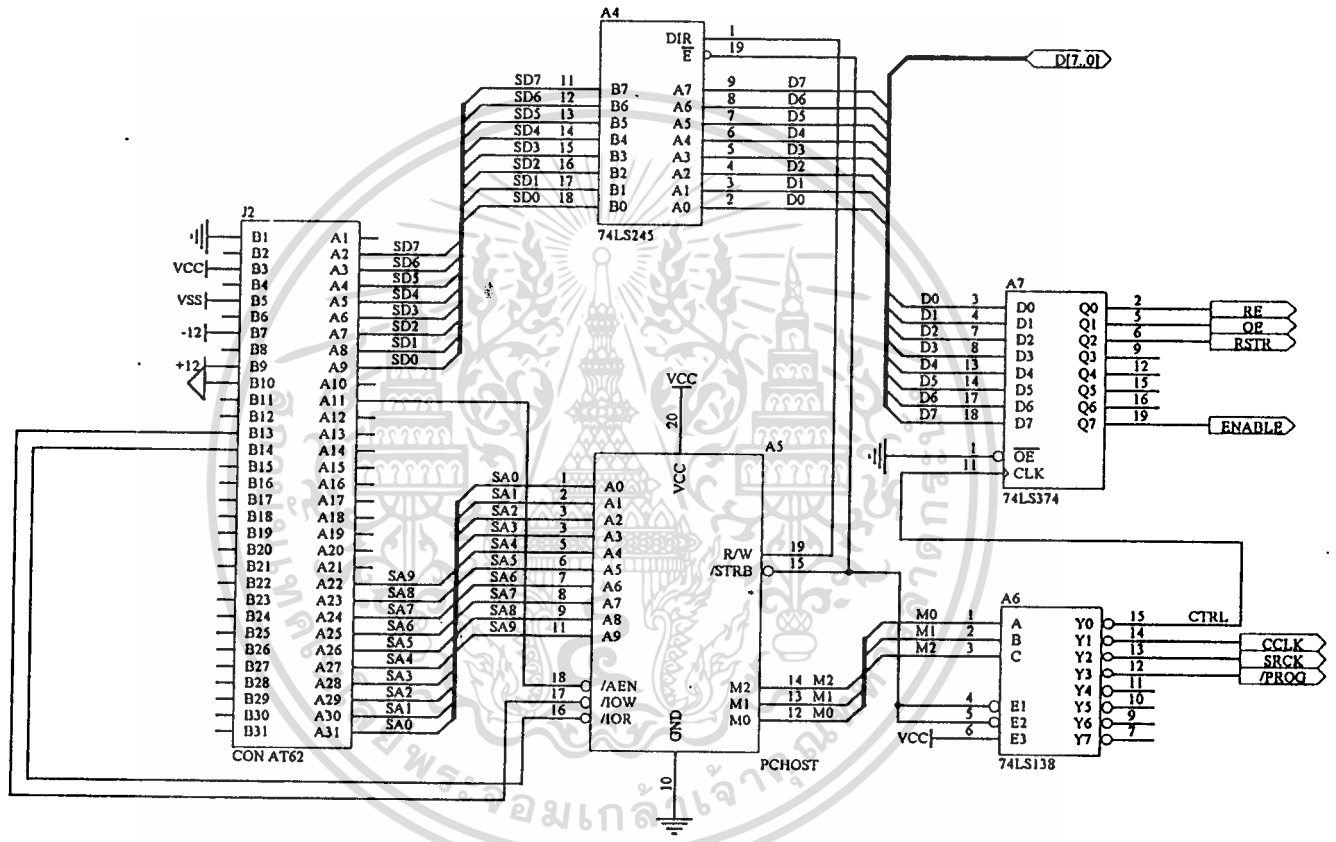
สตรีมข้อมูลจัดสัณฐานเริ่มต้นด้วยสายของ หนึ่งแปดตัว รหัสพรีเอมเบิล (Preamble) ตามด้วยค่านับของความยาว 24 บิต และแยกแต่ละฟิลด์ด้วยหนึ่ง ส่วนหัวนี้ตามด้วยข้อมูลจัดสัณฐานจริงในกรอบข้อมูล ความยาวและจำนวนของกรอบขึ้นอยู่กับชนิดของอุปกรณ์ แต่ละกรอบเริ่มด้วยฟิลด์เริ่มต้น และจบด้วยการตรวจสอบข้อผิดพลาด รหัสโพสแตมเบิล (Poatamble) จำเป็นต้องมีเพื่อเป็นสัญญาณว่าถึงจุดสุดท้ายของข้อมูลของแต่ละอุปกรณ์ ในทุกกรณีไบต์เริ่มต้น (Start-up) ของข้อมูลจำเป็นต้องมีเพื่อจัดเตรียมสัญญาณนาฬิกาสี่ลูกเพื่อใช้เป็นลำดับเริ่มต้นที่จุดสิ้นสุดของการจัดสัณฐาน Daisy Chain ยาวๆจำเป็นต้องมีไบต์เริ่มต้นเพิ่มเติมเพื่อที่จะเลื่อนข้อมูลสุดท้ายผ่านไปยังลูกโซ่ ไบต์เริ่มต้นทั้งหมดจะไม่พิจารณา ไบต์เหล่านี้จะไม่รวมอยู่ในบิตสตรีมที่สร้างขึ้นโดยซอฟต์แวร์ของ Xilinx

การเลือกการตรวจสอบความผิดพลาดแบบ CRC หรือ non-CRC ยอมให้มีขึ้นโดยซอฟต์แวร์สร้างบิตสตรีม การตรวจสอบความผิดพลาดแบบ non-CRC ตรวจสอบจุดสิ้นสุดของกรอบที่ต้องการสำหรับแต่ละกรอบ สำหรับการตรวจสอบความผิดพลาดแบบ CRC ซอฟต์แวร์จะคำนวณ CRC วิ่งและแทรกเศษย่อยตรวจสอบ 4 บิตหนึ่งเดียวที่จุดสุดท้ายของแต่ละกรอบ การตรวจสอบ CRC แบบ 11 บิตของกรอบสุดท้ายของ FPGA รวมถึง 7 บิตสุดท้ายของบิตข้อมูล

ผลการตรวจจับความผิดพลาดยังผลให้เกิดการขับยั้งการถ่ายข้อมูลและดึงขา /INIT ลง ในรูปแบบ Master CCLK และสัญญาณตำแหน่งจะยังคงปฏิบัติงานอยู่ภายนอก ผู้ใช้ต้องตรวจสอบขา /INIT และเริ่มต้นจัดสัณฐานใหม่โดยการจ่ายพัลส์ลบให้กับขา /PROGRAM หรือเข้าสู่วัฏจักรของ Vcc

Data Type	All Other Modes (D0...)
Fill Byte	11111111b
Preamble Code	0010b
Length Count	COUNT(23:0)
Fill Bits	1111b
Start Filed	0b
Data Frame	DATA(n-1:0)
CRC or Constant Field Check	xxxx (CRC) หรือ 0010b
Extend Write Cycle	-
Postamble Code	01111111b
Start-Up Bytes	xxh

ตารางที่ 2.3 รูปแบบสตรีมข้อมูลสำหรับอนุกรม XC4000



Title

Size	Number	Revision
A4		

Date:	29-Mar-1999	Sheet of
File:	D:\AOW\THESIS2\CAD\PCINP.SCH	Drawn By:

รูปที่ 3.2 แสดงการต่อวงจรพอร์ตขนาน

ตำแหน่งบิต [ชื่อ]	หน้าที่
D0 [RE]	ควบคุมการอ่านหน่วยความจำฟิลด์
D1 [OE]	ควบคุมอิมพิแดนซ์ของหน่วยความจำฟิลด์
D2 [RSTR]	ควบคุมให้อ่านหน่วยความจำฟิลด์ที่ตำแหน่งแรก
D7 [ENABLE]	ควบคุมการเริ่มต้นแปลงและเก็บข้อมูลภาพ

ตารางที่ 3.1 รีจิสเตอร์ควบคุมการทำงาน

3.1.2 ส่วนแปลงสัญญาณภาพ (Image Digitizer)

ส่วนแปลงสัญญาณภาพทำหน้าที่ปรับแต่งสัญญาณวิดีโอให้อยู่ในลักษณะที่เหมาะสม (Signal Conditioning) และแปลงสัญญาณภาพวิดีโอ (Video Signal Acquisition) ให้เป็นข้อมูลภาพดิจิตอลเพื่อใช้ในการประมวลผล โดยประกอบด้วยส่วนสำคัญดังนี้

A2 เบอร์ LF357 และอุปกรณ์แวดล้อมได้แก่ C8 R2 R3 และ D1 ทำหน้าที่เป็นวงจรคืนกระแสตรง (DC Restorer) เมื่อสัญญาณวิดีโอเข้า มาที่พอร์ต VIDEO จะชาร์ต (Charge) ตัวเก็บประจุ C8 ไปยังค่าแรงดันที่ยกระดับสัญญาณอินพุทให้มีค่าต่ำสุดเท่ากับระดับสัญญาณที่ขา 3 ของ A2 ซึ่งในที่นี้มีค่าเท่ากับ 0 โวลท์ เมื่อสัญญาณอินพุทแกว่งไปมีค่าลบ (ต่ำกว่าระดับสัญญาณที่ขา 3) A2 จะขับไดโอด D1 ทำให้เกิดเส้นทางที่มีอิมพิแดนซ์ต่ำไปชาร์ตตัวเก็บประจุ C8 หลังจากสัญญาณต่ำสุดผ่านไปและ D1 ไม่นำกระแสอีกครั้งหนึ่ง ประจุยังคงค้างอยู่ที่ C8 เนื่องจากเส้นทางที่ประจุจะคายได้มีเพียงขา 2 ของ A2 ผ่านทาง R2 เท่านั้นซึ่งเป็นเส้นทางที่มีกระแสไหลน้อยมาก R3 ทำหน้าที่เป็นโหลดอิมพิแดนซ์ของสัญญาณวิดีโออินพุทซึ่งมีค่าเท่ากับ 75 โอห์ม

A3 เบอร์ LF357 และอุปกรณ์แวดล้อมได้แก่ R4 และ VR1 ทำหน้าที่เป็นวงจรขยายแบบไม่กลับเฟส (Non-Inverting Amplifier) เนื่องจากสัญญาณวิดีโอมีระดับสัญญาณจากยอดถึงยอดประมาณ 1 โวลท์ แต่วงจรแปลงสัญญาณต้องการระดับแรงดันจากยอดถึงยอดประมาณ 1.7 โวลท์ ดังนั้น R4 และ VR1 ทำหน้าที่กำหนดอัตราขยายของวงจร

D2 และ R5 ทำหน้าที่เป็นวงจรลิมิตเตอร์ (Limitter) ทางด้านลบ เพื่อป้องกันแรงดันลบค่าสูงที่อาจเกิดขึ้นจาก สัญญาณรบกวนกระชาก (Spike Noise) หรือสัญญาณรบกวนอื่นจากภาคก่อนหน้าไปทำลายวงจรแปลงสัญญาณ โดยทีสัญญาณที่แคโทด(Cathode)ของ D2 ต้องมีค่ามากกว่า -0.7 โวลท์เท่านั้นจึงจะสามารถผ่านไปได้

D3 และ R6 ในทำนองเดียวกันทำหน้าที่เป็นวงจรลิมิตเตอร์ทางด้านบวก เพื่อป้องกันแรงดันบวกค่าสูงที่อาจเกิดจาก สัญญาณรบกวนกระชาก หรือสัญญาณรบกวนอื่นจากภาคก่อนหน้าไปทำลายวงจรแปลงสัญญาณ โดยวงจรแปลงสัญญาณได้กำหนดระดับสัญญาณที่สามารถแปลงได้อยู่ในช่วง 1.5-3.2 โวลท์ เมื่อสัญญาณที่อาโนด (Anode) ของ D3 มีค่ามากกว่า 0.7 โวลท์บวกกับแรงดันที่ขาแคโทดของ D4 ซึ่งทำหน้าที่เป็นวงจรกำเนิดแรงดันอ้างอิง 2.50 ซึ่งมีค่าเท่ากับ 3.20 โวลท์เท่านั้นจึงจะผ่านไปได้ ซึ่งมีค่าเท่ากับค่าสูงสุดที่วงจรแปลงสัญญาณจะสามารถแปลงได้ ระดับสัญญาณที่สูงกว่านี้จะถูกละทิ้งไป (ปรับให้มีค่า 3.20 โวลท์โดยอัตโนมัติ)

A1 เบอร์ TDA8703 และอุปกรณ์แวดล้อมได้แก่ C1 C2 C3 C4 C5 C6 C7 และ R1 ทำหน้าที่เป็นวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลความเร็วสูง (Flash Analog to Digital Converter) ซึ่งมีความเร็วในการแปลงสัญญาณสูงถึง 40 MSPS มากพอสำหรับสัญญาณวิดีโอ สัญญาณที่ได้จากการปรับปรุงจากภาคก่อนหน้าจะส่งมายังวงจรแปลงสัญญาณทางขา 8 ของ A1 ข้อมูลที่แปลงได้จะถูกส่งออกทางบัสข้อมูล D0-D7 ของ A1 (AD [0..7]) สัญญาณนาฬิกาที่ใช้ในการแปลงได้จากคริสตอลออสซิลเลเตอร์ (Crystal Oscillator) X1 วงจรแสดงดังรูปที่ 3.3

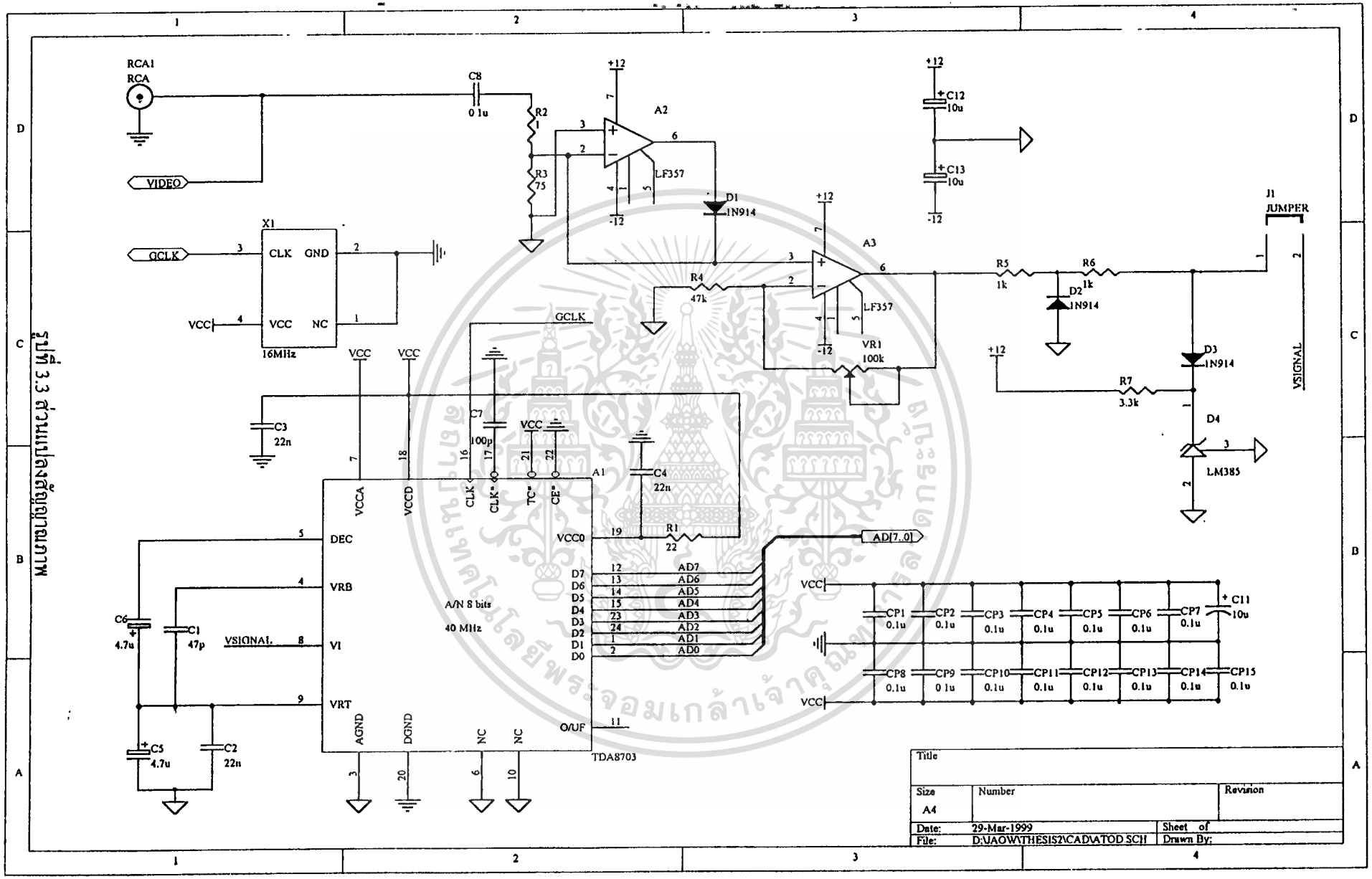
3.1.3 ส่วนสร้างสัญญาณควบคุม (Control Signal Generator)

ส่วนนี้ทำหน้าที่สังเคราะห์สัญญาณควบคุมการทำงานของอุปกรณ์ภายในการ์ดให้ทำงานสัมพันธ์กันกับสัญญาณต่างๆ อาทิเช่นสัญญาณเข้าจังหวะวิดีโอ, สัญญาณนาฬิกาของวงจรแปลงสัญญาณภาพ เป็นต้น รายละเอียดการทำงานของแต่ละอุปกรณ์มีดังนี้

A14 เบอร์ LM1881 และอุปกรณ์แวดล้อม ได้แก่ C9 C10 และ R8 ทำหน้าที่เป็นวงจรแยกสัญญาณเข้าจังหวะวิดีโอ (Video Synchronizing Separator) สัญญาณที่ได้จะเป็นพัลส์ที่แสดงถึงสัญญาณเข้าจังหวะในแนวนอน (Horizontal Synchronizing Pulse) และสัญญาณเข้าจังหวะในแนวตั้ง (Vertical Synchronizing Pulse) ทางขา 1 และ ขา 3 ตามลำดับ และสัญญาณระบุชนิดของฟิลด์ทางขา 7

พัลส์เข้าจังหวะในแนวนอนจะใช้เป็นสัญญาณนาฬิกาให้กับวงนับ 9 บิต (หาร 512) ซึ่งประกอบด้วย A15A, A15B และ A16A เบอร์ 74LS393 เพื่อระบุตำแหน่งของเส้นสแกนในขณะนั้นเพื่อใช้สร้างสัญญาณควบคุมที่สัมพันธ์กัน ค่าที่นับได้ต่อไปยังบัส VCT[8..0] สัญญาณระบุชนิดของฟิลด์จากขา 7 ของ A14 ต่อไปยังขา 2,12 และ 2 (MR) ของ A15A, A15B, A16A ตามลำดับซึ่งจะมีค่าเป็น High และ Low เมื่อฟิลด์ขณะนั้นเป็นฟิลด์คู่และฟิลด์คี่ตามลำดับ ทั้งนี้เนื่องจากในการประมวลผลภาพจะใช้ข้อมูลภาพเพียงแค่ฟิลด์เดียว ในที่นี้เลือกใช้ฟิลด์คี่ ด้วยการต่อเช่นนี้จะทำให้วงจรนับดังกล่าวเริ่มนับใหม่ทุกครั้งที่มีการเปลี่ยนจากฟิลด์คู่ไปฟิลด์คี่ และหยุดนับเมื่อเปลี่ยนจากฟิลด์คี่ไปฟิลด์คู่

สัญญาณที่นับได้จากวงจรนับจากบัส VCT[8..0] จะต่อไปเข้า A18 ซึ่งเป็น PAL(Programmable Array Logic) เบอร์ PAL16L8 โดยเขียนโปรแกรมให้รับค่าเส้นสแกนขณะนั้นเข้ามา เพื่อสร้างสัญญาณควบคุมที่เกี่ยวข้อง กล่าวคือ



รูปที่ 3.3 ตัวแปลงสัญญาณภาพเป็นดิจิตอล

1.สัญญาณ /RESET จะใช้สำหรับรีเซตตำแหน่งของไลน์ดีเลย์บัฟเฟอร์ (Line Delay Buffer) ซึ่งจะทำการรีเซตในช่วงเวลาของเส้นสแกนที่ 1 เท่านั้น ดังนั้นสัญญาณ /RESET จะเป็น Low เมื่อขณะนั้นเป็นเส้นสแกนที่1 นอกจากนั้นสัญญาณนี้จะเป็น High

2. สัญญาณ CAPT จะใช้ระบุบริเวณที่ทำการแปลงและจัดเก็บข้อมูลภาพ ในที่นี้จะทำการแปลงสัญญาณภาพตั้งแต่เส้นสแกนที่ 12 ถึงเส้นสแกนที่ 299 หรือคิดเป็น 288 เส้นทั้งนี้เนื่องจากบริเวณดังกล่าวเป็นบริเวณข้อมูลภาพที่ปรากฏจริง ไม่รวมถึงบริเวณที่ใช้สร้างสัญญาณเข้าจังหวะ อาทิเช่นสัญญาณ Equalizing Pulse และสัญญาณ Serration Pulse เป็นต้น สัญญาณ CAPT จะเป็น High เฉพาะเส้นสแกนที่ 12 ถึง 299 เท่านั้น นอกจากนั้นสัญญาณนี้จะเป็น Low

3. สัญญาณ CS เป็นสัญญาณเลือก ให้มีการทำงานของ A18 จากเงื่อนไขข้างต้นสามารถเขียนโปรแกรม PAL ด้วยภาษา ABEL สำหรับ U1 ได้ดังนี้

```
Module cropv
title 'Composite Sync Information extractor Location U002
Paramate    HORKAEW       9 June 1998'

        cropv                   device 'P16L8';
        D0..D8                  pin 1..9;
        ChipSel                 pin 11;

        !Reset                  pin 19 istype 'com';
        Capture                 pin 18 istype 'com';

        Counter = [D8..D0];

Equations

        Capture = ChipSel & (Counter>=12) & (Counter<=299);
        Reset   = (Counter==1);

Test_Vectors
       ([ChipSel, Counter] -> [Capture, Reset])
        [1,  0] -> [ 0, 0];
        [1,  1] -> [ 0, 1];
        [1,  2] -> [ 0, 0];
        [1, 11] -> [ 0, 0];
        [1, 12] -> [ 1, 0];
        [1, 13] -> [ 1, 0];
        [1, 14] -> [ 1, 0];
        [1, 15] -> [ 1, 0];
        [0, 50] -> [ 0, 0];
        [1,100] -> [ 1, 0];
        [1,295] -> [ 1, 0];
        [1,296] -> [ 1, 0];
        [1,297] -> [ 1, 0];
        [1,298] -> [ 1, 0];
        [1,299] -> [ 1, 0];
        [1,314] -> [ 0, 0];
        [1,315] -> [ 0, 0];
        [1,320] -> [ 0, 0];
        [1,400] -> [ 0, 0];
        [1,500] -> [ 0, 0];
End
```

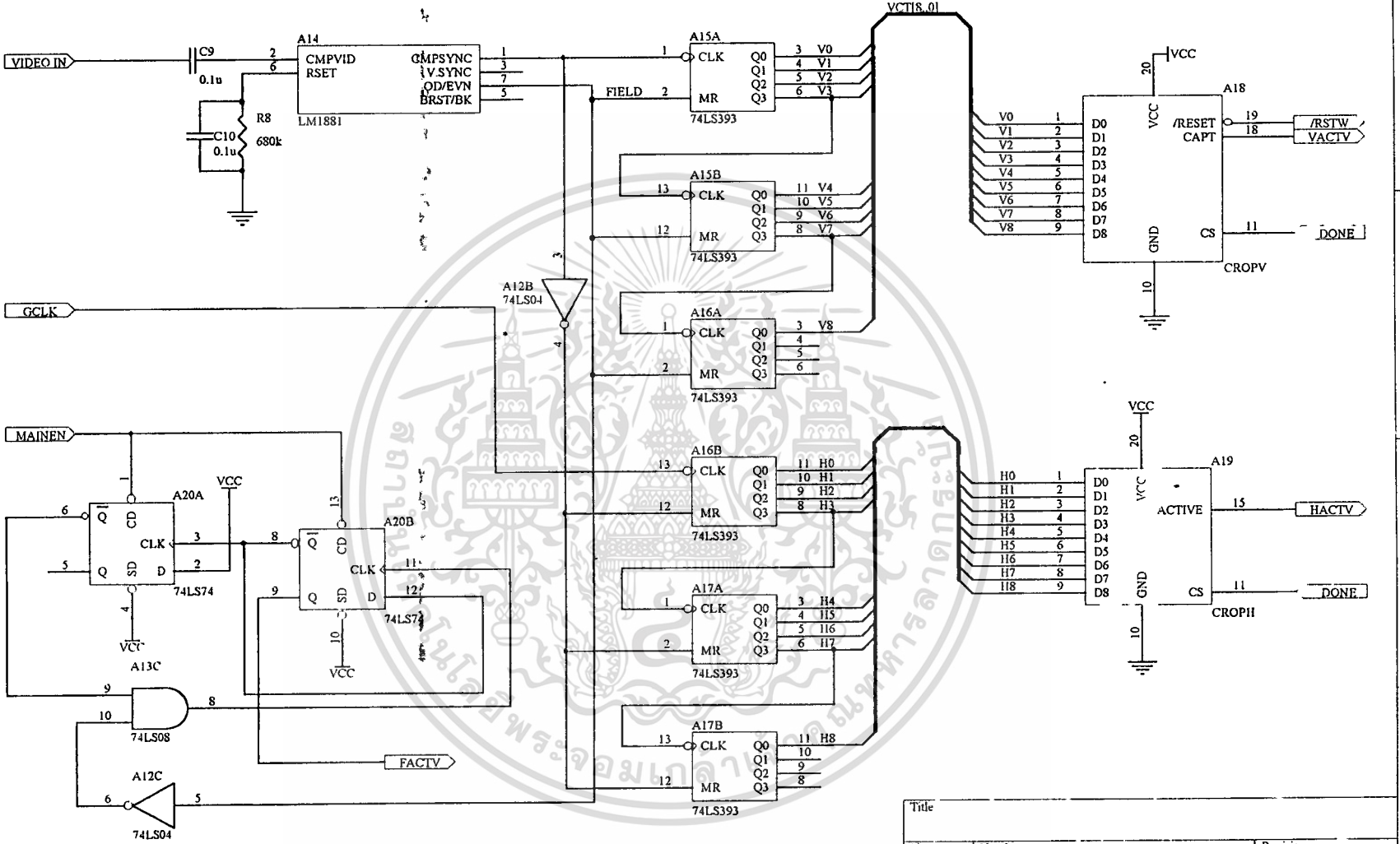
สัญญาณนาฬิกาของระบบจากพอร์ต GCLK ต่อไปยังขา 13 ของ A16B เบอร์ 74LS393 ซึ่งทำหน้าที่เป็นวงจรนับ 9 บิต ร่วมกับ A17A และ A17B เบอร์ 74LS393 เพื่อระบุตำแหน่งของจุดภาพในเส้นสแกนขณะนั้นเพื่อใช้สร้างสัญญาณควบคุมที่สัมพันธ์กัน ค่าที่นับได้ต่อไปยังบัส HCT[8..0] สัญญาณเข้าจังหวะในแนวนอนจากขา 1 ของ A14 ต่อผ่านอินเวอร์เตอร์ (Inverter) ไปเข้าขา 12, 2 และ 2 ของ A16B, A17A, A17B ตามลำดับ ดังนั้นวงจรนับจะเริ่มนับใหม่ทุกครั้งที่เริ่มต้นเส้นสแกนใหม่

สัญญาณที่นับได้จากวงจรนับจากบัส HCT[8..0] จะต่อไปเข้า A19 ซึ่งเป็น PAL โดยเขียนโปรแกรมให้รับค่าตำแหน่งจุดภาพขณะนั้นเข้ามา เพื่อสร้างสัญญาณควบคุมที่เกี่ยวข้องได้แก่สัญญาณ ACTIVE เนื่องจากจำนวนเส้นสแกนในแนวตั้งที่ทำการแปลงสัญญาณภาพเป็น 288 เส้น ดังนั้นเพื่อให้ค่าอัตราส่วนยังผล (Aspect Ratio) ของภาพยังเป็น 4:3 ดังนั้นบริเวณที่ทำการแปลงสัญญาณในแนวนอนควรจะเป็น (4x288)/3 หรือประมาณ 384 จุดภาพ ถ้าสัญญาณนาฬิกาเป็น 8 MHz (สำหรับแปลงสัญญาณวิดีโอที่มีแบนด์วิดท์ 4 MHz ตามเงื่อนไขของ Nyquist) และช่วงเวลาของสัญญาณวิดีโอหนึ่งเส้นเป็น 64 ไมโครวินาที ถ้าไม่นับช่วงเวลาสัญญาณเข้าจังหวะ 12 ไมโครวินาทีจะได้ว่าจำนวนจุดภาพที่สามารถเห็นได้เป็น

$$(64\times10^{-6}-12\times10^{-6})\times(8\times10^{6})=416\ Pixels$$

แต่เราต้องการเพียง 384 จุดภาพ ดังนั้นกำหนดสัญญาณ CS เป็นสัญญาณเลือก ให้มีการทำงานของ A19 จากเงื่อนไขข้างต้นสามารถเขียนโปรแกรม PAL สำหรับ A19 เบอร์ PAL16L8 ได้ดังนี้

```
Module croph
title 'Composite Sync Information extractor Location U002
Paramate   HORKAEW      9 June 1998'

        croph                   device 'P16L8';
        D0..D8                  pin 1..9;
        ChipSel                 pin 11;
        Active                  pin 15 istype 'com';

        Counter = [D8..D0];

Equations

        Active  = ChipSel & (Counter>=16) & (Counter<=399);

Test_Vectors
       ([ChipSel, Counter] -> [Active])
        [0,  0] -> [ 0 ];
        [1,  0] -> [ 0 ];
        [1,  1] -> [ 0 ];
        [1, 15] -> [ 0 ];
        [1, 16] -> [ 1 ];
        [1, 17] -> [ 1 ];
        [1, 18] -> [ 1 ];
        [1, 19] -> [ 1 ];
        [1, 20] -> [ 1 ];
        [1, 50] -> [ 1 ];
        [1,100] -> [ 1 ];
        [1,300] -> [ 1 ];
        [1,310] -> [ 1 ];
        [1,395] -> [ 1 ];
        [1,396] -> [ 1 ];
        [1,397] -> [ 1 ];
```



รูปที่ 3.4 ส่วนสร้างสัญญาณควบคุม

การอ่านข้อมูลจาก A11 ทำผ่านบัส D[7..0] โดยมีสัญญาณควบคุม RE, OE และ RSTW ที่โปรแกรมจาก PC ผ่านทางรีจิสเตอร์ A7 และสัญญาณนาฬิกาในการอ่าน SRCK จากวงจรถอดรหัส A6 วงจรส่วนนี้แสดงดังรูป 3.5

3.2 การออกแบบตัวกรองดิจิตอล (Digital Filter)

ในขั้นตอนนี้จะเป็นการออกแบบวงจรกรองดิจิตอลบน FPGA โดยจะเป็นการยกตัวอย่างตัวกรองดิจิตอลแบบทวินาม (Binomial) ซึ่งมีผลตอบสนองอิมพัลส์ในรูปของเมตริกซ์ดังนี้

$$1/16\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

สามารถหาผลตอบสนองเชิงความถี่ได้ดังนี้

$$H(\omega_1,\omega_2)=\frac{1}{16}\begin{bmatrix}\exp(j(-\omega_1+\omega_2))+2\exp(j\omega_2)+\exp(j(\omega_1+\omega_2))+\\ 2\exp(-j\omega_1)+4+2\exp(j\omega_1)+\\ \exp(j(-\omega_1-\omega_2))+2\exp(-j\omega_2)+\exp(j(\omega_1-\omega_2))\end{bmatrix} \quad \cdots(3.1)$$

3.2.1 เทคนิคในการลดขั้นตอนการคำนวณ (Computation Optimization)

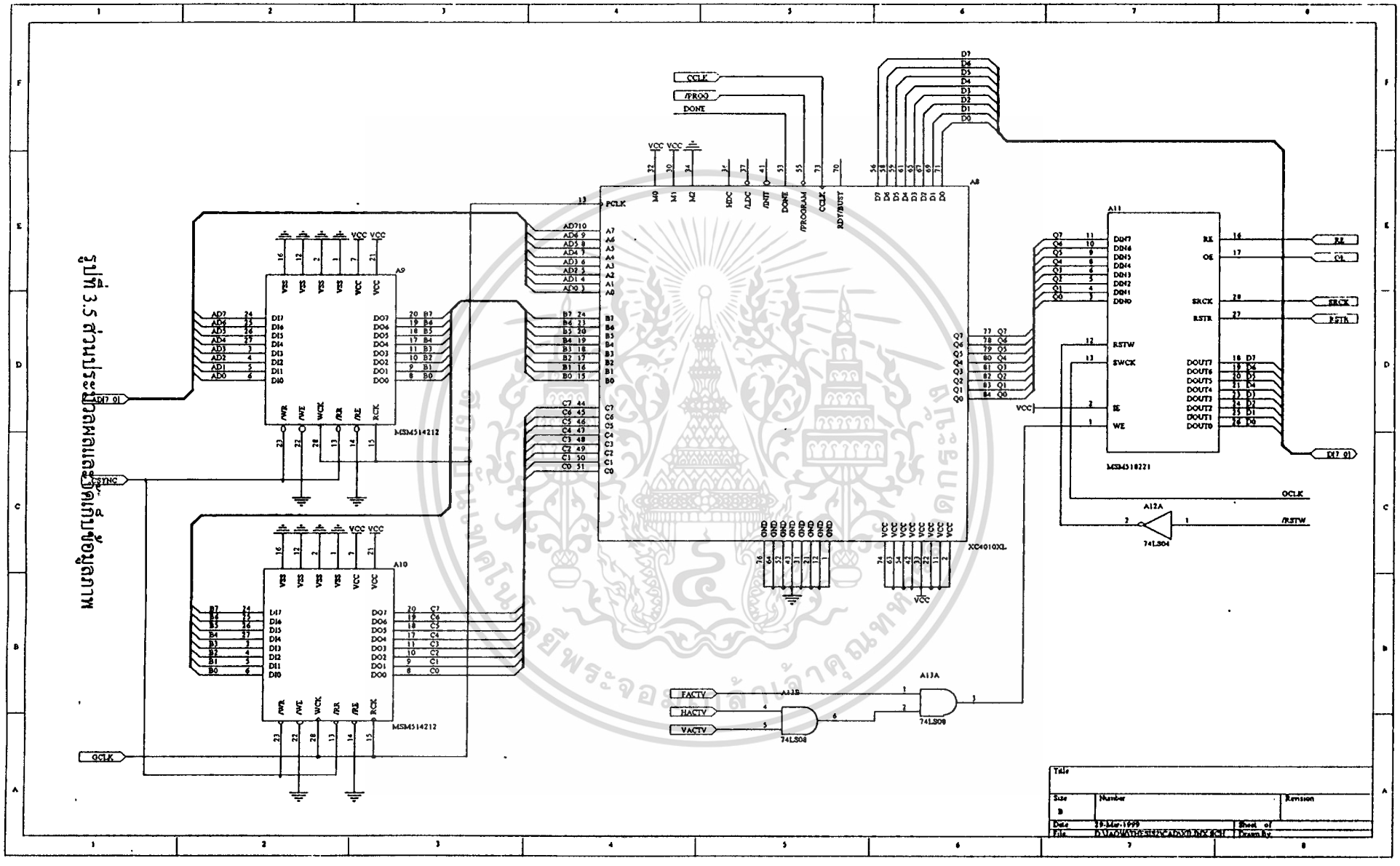
สำหรับการคำนวณการคอนโวลูชั่นระหว่างตัวดำเนินการ *h(i,j)* ใดๆ กับภาพดิจิตอล *f(i,j)* ซึ่งมี สมการดังนี้

$$h(i,j)\otimes f(i,j)=\sum_{k=1}^{m}\sum_{l=1}^{n}h(k,l)f(i-k,j-l) \quad \cdots(3.2)$$

ถ้าหากสามารถพิสูจน์ได้ว่า

$$h(i,j)\otimes f(i,j)=\sum_{k=1}^{m}h_1(k)\left\{\sum_{l=1}^{n}h_2(l)f(i-k,j-l)\right\} \quad \ldots(3.3)$$

โดยที่ค่าในวงเล็บเป็นการคอนโวลูชั่นระหว่างภาพดิจิตอล *f(i,j)* กับผลตอบสนองอิมพัลส์ในแนวตั้ง *h(l)* ผลลัพธ์ที่ได้นำไปทำคอนโวลูชั่นกับผลตอบสนองอิมพัลส์ในแนวนอน *h(k)* เนื่องจากการคอนโวลูชั่นมีคุณสมบัติการจัดกลุ่มและการสลับที่ ลำดับของการทำคอนโวลูชั่นสามารถทำกลับกันได้โดยทำคอนโวลูชั่นในแนวนอนก่อน แล้วจึงนำผลลัพธ์ที่ได้มาทำคอนโวลูชั่นในแนวตั้ง

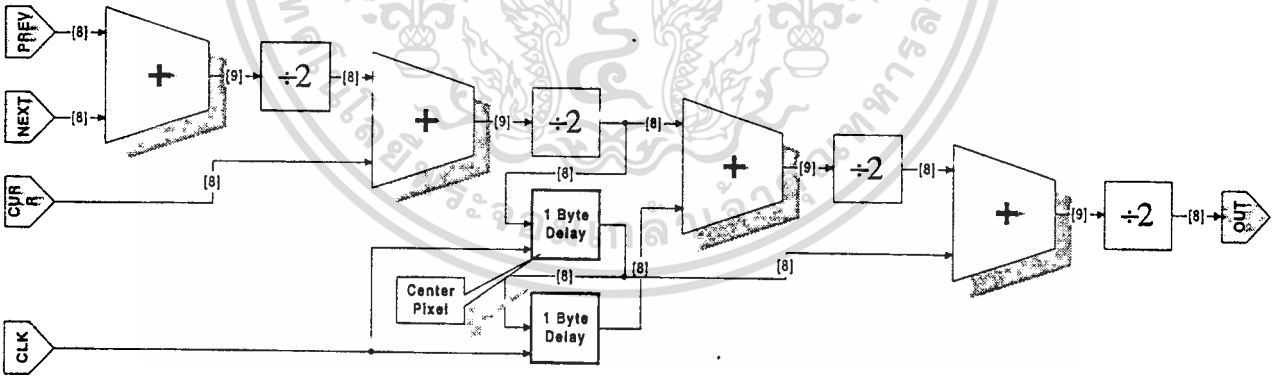


รูปที่ 3.5 ส่วนประกอบของหน่วยความจำภาพ

วิธีการนี้จะสามารถสร้างได้จากการทำคอนโวลูชั่นในแนวนอนก่อนแล้วผลลัพธ์ชั่วคราวเก็บในตำแหน่งทรานสโพส (Transpose) ของตำแหน่งเดิม แล้วจึงนำผลลัพธ์ชั่วคราวไปทำคอนโวลูชั่นด้วยวิธีการเดียวกัน ซึ่งเป็นการทำคอนโวลูชั่นในแนวตั้งจากคอนโวลูชั่นในแนวนอน ข้อมูลผลลัพธ์จากการทำคอนโวลูชั่นครั้งที่สองก็จะเป็นการทำทรานสโพส (Transpose) อีกครั้งทำให้ข้อมูลเก็บในตำแหน่งที่ถูกต้อง จากผลตอบสนองอิมพัลส์ของตัวกรองแบบทวินามและ เมตริกซ์ในแนวตั้ง $\begin{bmatrix} 1/4 \\ 1/2 \\ 1/4 \end{bmatrix}$ กับเมตริกซ์ในแนวนอน $\begin{bmatrix} 1/4 & 1/2 & 1/4 \end{bmatrix}$ ซึ่งเมื่อแทนค่าลงไปในสมการ (3.3) สามารถพิสูจน์ได้ว่าสมการเป็นจริง ดังนั้นจึงนำความรู้นี้ไปสร้างเป็นวงจรตรรกต่อไป

3.2.2 ตัวกรองดิจิตอลในรูปของวงจรตรรก (Digital Filter in Boolean Circuit Form)

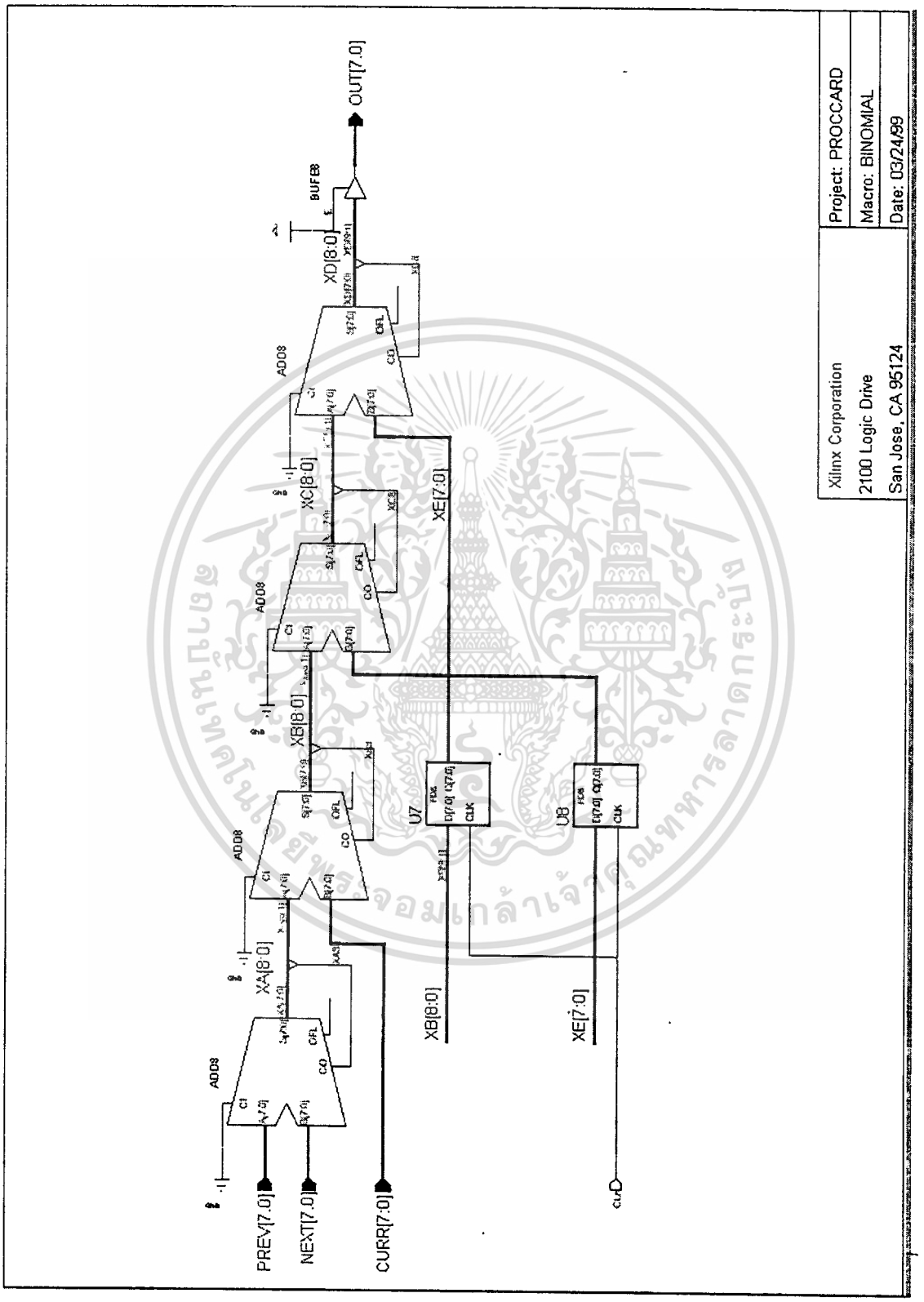
จากหัวข้อที่ผ่านมาเห็นได้ว่าสามารถนำข้อมูลภาพไปทำการคอนโวลูชั่นกับเมตริกซ์ในแนวตั้งก่อนแล้วจึง นำผลลัพธ์ที่ได้ไปทำคอนโวลูชั่นกับเมตริกซ์ในแนวนอน แต่เพื่อความสะดวกในการออกแบบวงจรจะทำการดึงตัวร่วม ¼ ของทั้งสองเมตริกซ์ออกมาไว้ข้างนอกสุด ดังนั้นจะได้เมตริกซ์ในแนวตั้งเป็น $\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$ และเมตริกซ์ในแนวนอนเป็น $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$ เมื่อทำคอนโวลูชั่นแล้วจึงหารด้วย 1/16 แผนผังวงจรกรองดิจิตอลแบบทวินามแสดงดังรูปที่ 3.6



รูปที่ 3.6 แผนผังวงจรกรองดิจิตอลแบบทวินาม

จากแผนผังวงจรในรูปที่ 3.6 วงจรบวก 8 บิตทำการบวกค่าจุดภาพที่เส้นที่ n-1 และเส้นที่ n+1 แทนด้วยสัญลักษณ์ PREV และ NEXT ตามลำดับ แล้วจึงนำผลลัพธ์ที่ได้ไปหารด้วยสองแล้วบวกกับจุดภาพเส้นที่ n แทนด้วยสัญลักษณ์ CURR ทั้งหมดนี้คือการทำคอนโวลูชั่นในแนวตั้ง หลังจากนั้นผลลัพธ์จะถูกหน่วงไป 1 และ 2 จุดภาพ ในทำนองเดียวกัน ค่าจุดภาพที่ m-1 และค่าจุดภาพที่ m+1 จะบวกกัน ผลลัพธ์ที่ได้ไปหารด้วย

สองแล้วบวกกับค่าจุดภาพที่ m ผลลัพธ์ที่ได้จะหารด้วยสองอีกครั้ง ได้เป็นค่าจุดภาพที่ต้องการที่พิกัด (m, n) ของภาพ ซึ่งสามารถนำไปสร้างเป็นวงจรที่สมมูลย์กันได้ดังรูปที่ 3.7



รูปที่ 3.7 วงจรที่สมมูลย์กับแผนผังในรูปที่ 3.6

3.3 การออกแบบโปรแกรมควบคุมการทำงาน

โปรแกรมสำหรับควบคุมการทำงานของการ์ดประมวลผลภาพดิจิตอลแบบโปรแกรมได้เป็นโปรแกรมที่ทำงานภายใต้ระบบปฏิบัติการ DOS มีคุณสมบัติ ดังต่อไปนี้

1. สามารถติดต่อกับผู้ใช้ผ่านทางแป้นพิมพ์
2. แสดงผลภาพในโหมดกราฟฟิกโมโนโครม (Monochrome)
3. สามารถเลือกคอนโวลเวอร์ (Convolver) ที่จะโปรแกรม FPGA ได้และสามารถโปรแกรม FPGA ผ่านทางพอร์ตเอาท์พุท
4. สามารถอ่านข้อมูลภาพที่เก็บในหน่วยความจำหลักของการ์ดผ่านทางพอร์ตอินพุทมาแสดงผลได้
5. สามารถจัดเก็บภาพเป็นแฟ้มในรูปข้อมูลดิบได้ (*.RAW : ข้อมูลประกอบด้วยค่าความสว่างของจุดภาพแต่เพียงอย่างเดียว)

เพื่อให้โปรแกรมสามารถทำได้ตามคุณสมบัติข้างต้น ในที่นี้จะพัฒนาโปรแกรมโดยใช้ภาษา C โดยแบ่งออกเป็นสองส่วนแยกออกจากกัน กล่าวคือส่วน การทำงานหลัก และ ส่วนดูแลการแสดงผล อยู่ในแฟ้มข้อมูลในภาคผนวกชื่อ IMGMAIN.C และ IMGGRAPH.C ตามลำดับ ในที่นี้จะอธิบายการออกแบบเฉพาะส่วนการทำงานหลักเท่านั้น (สำหรับส่วนดูแลการแสดงผลนั้นจะเป็นรูปแบบที่นิยมใช้กันทั่วไป สำหรับผู้สนใจสามารถศึกษาได้จากภาคผนวก) ซึ่งแบ่งเป็นโมดูล (Module) ย่อยที่สำคัญได้ดังนี้

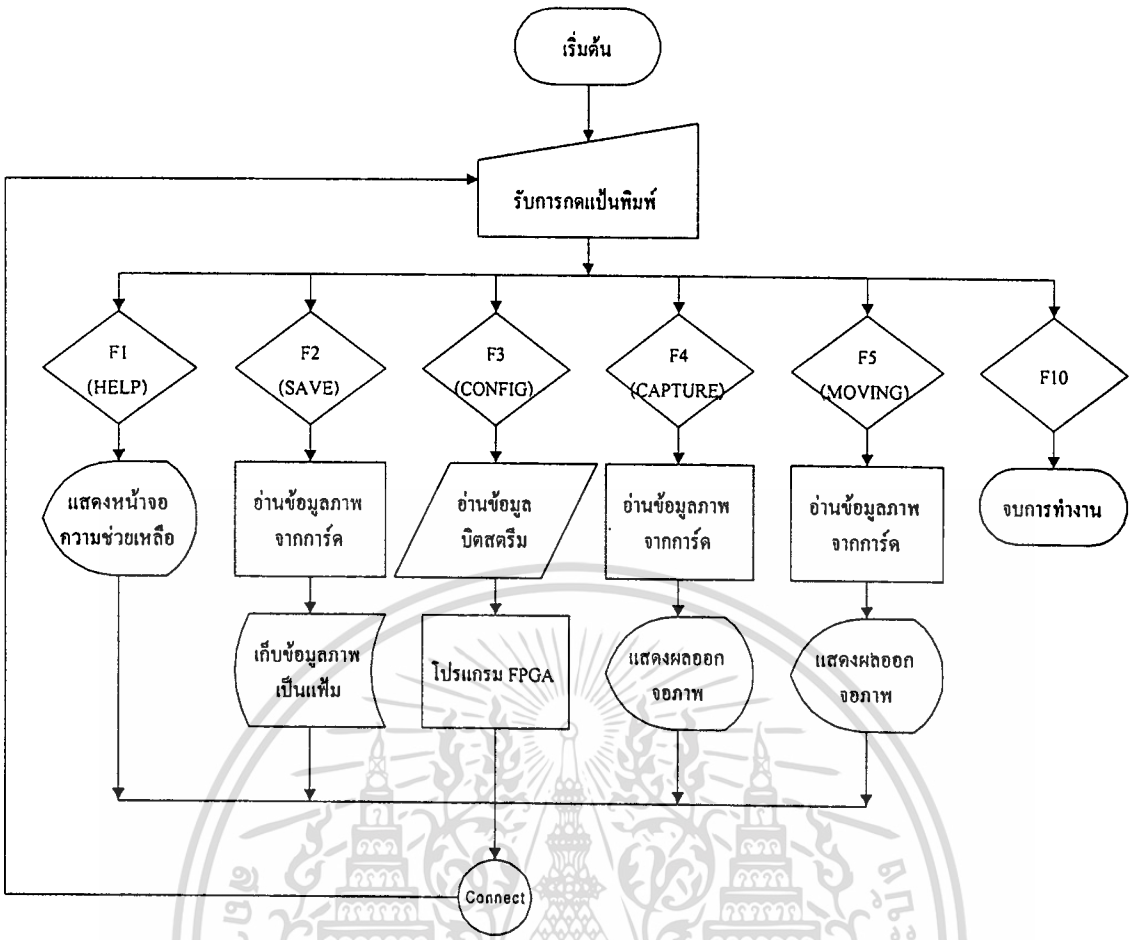
3.3.1 ส่วนติดต่อกับผู้ใช้

ส่วนนี้จะทำงานโดยรอรับการกดแป้นพิมพ์จากผู้ใช้ และพิจารณาลักษณะของคำสั่ง ได้แก่เรียกหน้าจอความช่วยเหลือ จัดเก็บข้อมูลภาพ โปรแกรม FPGA อ่านข้อมูลภาพมาแสดงผล และจบการทำงาน และรับผิดชอบเรียกฟังก์ชันเพื่อทำตามคำสั่งนั้น มีแผนผังการทำงานดังรูปที่ 3.8

3.3.2 ส่วนโปรแกรม FPGA (CONFIG)

ส่วนนี้จะทำงานอ่านแฟ้มข้อมูลที่เก็บบิตสตรีม (Bit stream) ของคอนโวลเวอร์ที่ผู้ใช้เลือกมาโปรแกรม FPGA ผ่านทางพอร์ตเอาท์พุต ก่อนจะเริ่มทำการโปรแกรมควรจะดำเนินการดังต่อไปนี้

- ยกเลิกการอินเตอร์รัพท์ทั้งหมด เพื่อป้องการการอ่านข้อมูลผิดพลาดของ FPGA ถ้าหากมีการเขียนข้อมูลโดย DMA (Direct Memory Access)
- ทำให้บัสมีอิมพิแดนซ์สูง โดยการตัดการเชื่อมต่อบัสข้อมูลของการ์ดกับพอร์ตเอาท์พุทของหน่วยความจำหลัก เพื่อป้องกันการสับสนในการตีความข้อมูล



รูปที่ 3.8 แผนผังส่วนติดต่อกับผู้ใช้

เมื่อโปรแกรม FPGA เสร็จแล้วให้อินเตอร์รัพท์กลับสู่สถานะเดิม ส่วนนี้มีแผนผังการทำงานดังรูปที่ 3.9

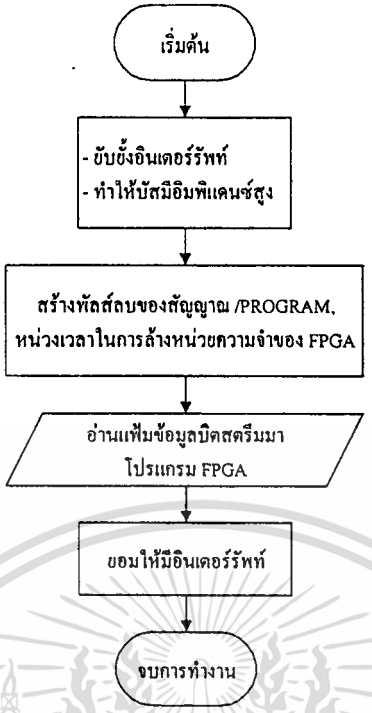
3.3.3 ส่วนควบคุมการแปลงภาพและอ่านข้อมูลภาพจากการ์ดมาแสดงผล

ส่วนนี้จะควบคุมการแปลงสัญญาณภาพ และ อ่านข้อมูลภาพที่พักอยู่ในหน่วยความจำฟิลด์ที่อยู่บนการ์ดมาแสดงผลทางจอภาพ มีแผนผังการทำงานดังรูปที่ 3.10

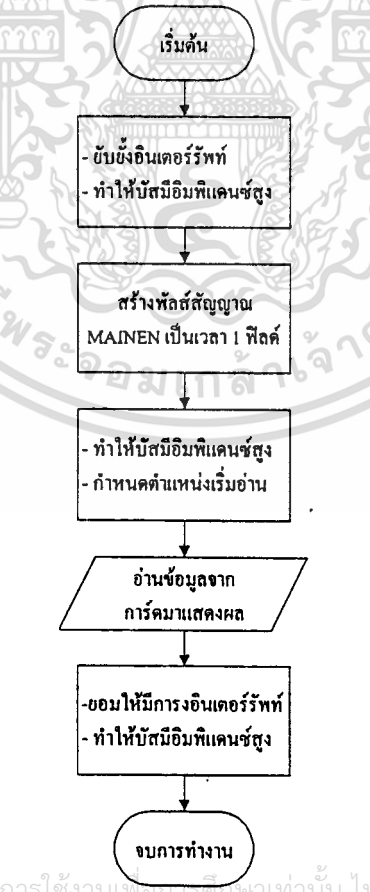
หลักการทำงานในส่วนนี้ สามารถนำไปอธิบายการทำงานในอีกสองส่วนที่เหลือ คือ

- ส่วนแสดงผลแบบต่อเนื่อง (MOVING)
- ส่วนอ่านและจัดเก็บข้อมูลภาพเป็นแฟ้มข้อมูล (SAVE)

โดยที่ขั้นตอนเบื้องต้นของทั้งสามส่วนเหมือนกัน จึงขอไม่อธิบายการทำงานในส่วนดังกล่าวในที่นี้ สำหรับผู้ที่สนใจสามารถศึกษาโปรแกรมโดยละเอียดได้จากภาคผนวก



รูปที่ 3.9 ส่วนโปรแกรม FPGA



รูปที่ 3.10 ส่วนควบคุมการแปลงภาพและอ่านข้อมูลภาพมาแสดงผล

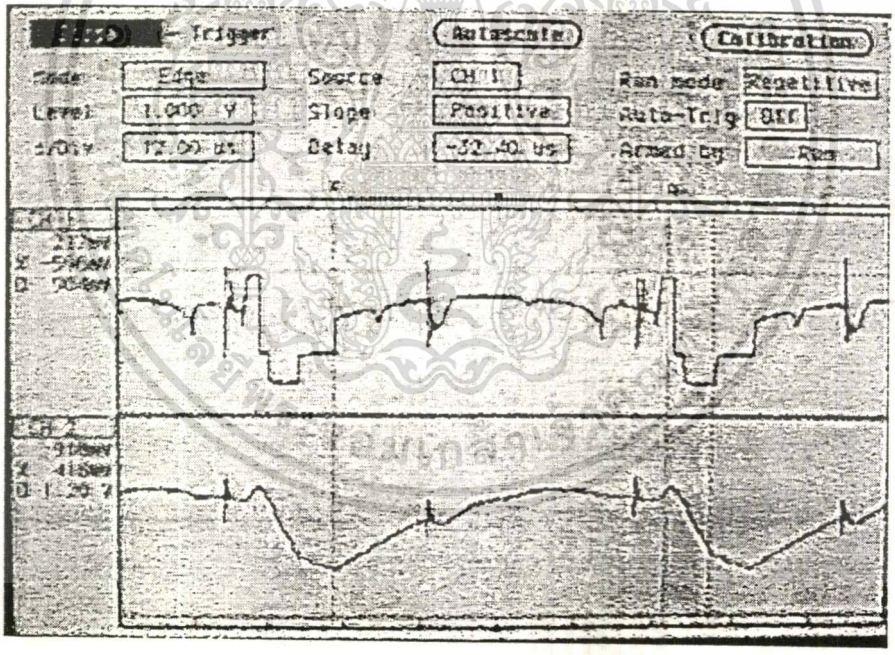
บทที่ 4
การทดลองและผลการทดลอง

ในบทนี้จะกล่าวถึงการทดลองและผลการทดลองในปริญญานิพนธ์นี้ โดยแบ่งเป็น 3 ส่วนกล่าว คือ การทดลองส่วนการ์ดประมวลผลภาพดิจิตอล การสร้างคอนโวลเวอร์ขนาด 3x3 บน FPGA และการทดลองใช้งานจริงของการ์ดประวลผลภาพดิจิตอลที่ได้ออกแบบไว้

4.1 การทดลองส่วนการ์ดประมวลผลภาพดิจิตอล

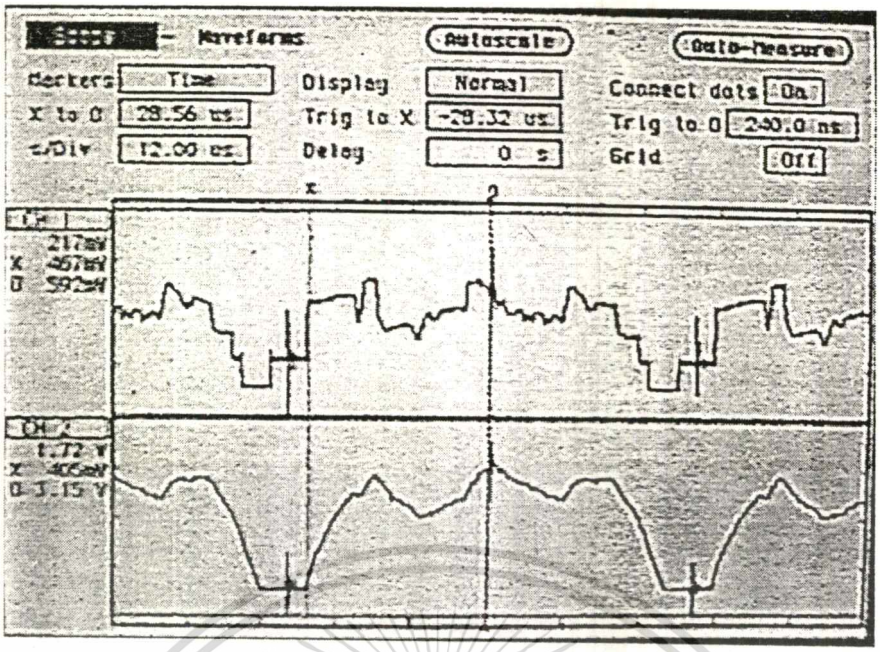
การทดลองในส่วนนี้จะเป็นการวัดสัญญาณที่จุดต่างๆ โดยใช้เครื่องมือวัด Logic Analyzer รุ่น HP1652B โดยจะทำการวัดในส่วนแปลงสัญญาณภาพที่ทำหน้าปรับรูปแบบของสัญญาณให้เหมาะสม

พิจารณาวงจรในส่วนแปลงสัญญาณภาพวัดแรงดันที่ผ่านจากวงจรคืนกระแสตรง (DC-Restorer) A2 โดยใช้สโคปช่องที่ 1 วัดสัญญาณวิดีโออินพุทที่ C8 และช่องที่สองวัดสัญญาณเอาท์พุทที่ขา 2 ของ A2 ได้รูปสัญญาณดังรูปที่ 4.1 กล่าวคือสัญญาณจะถูกยกระดับมาอยู่ซีกบวก



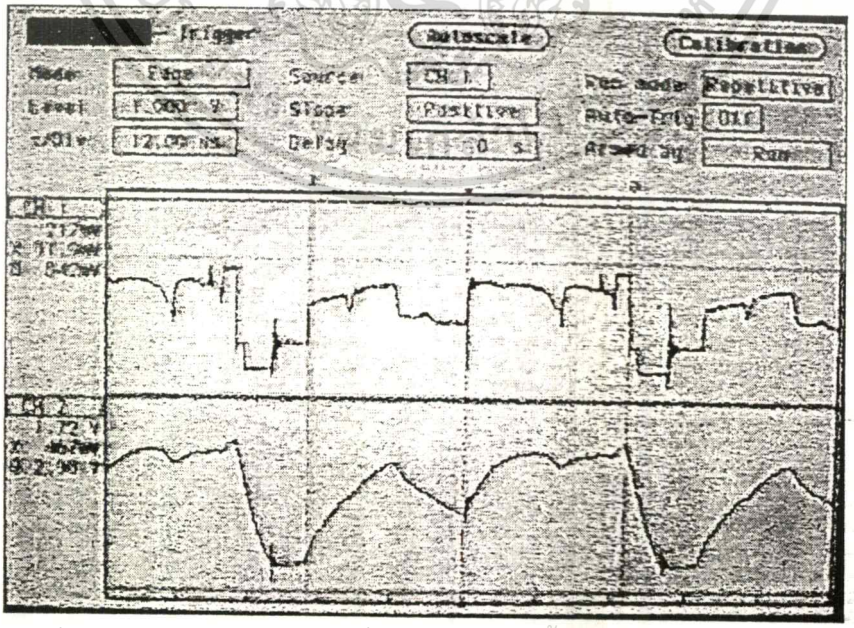
รูปที่ 4.1 สัญญาณที่ได้จากวงจรคืนกระแสตรง

เปลี่ยนช่องที่สองไปวัดที่เอาท์พุทของวงจรขยาย A3 โดยวัดที่ขา 6 ได้รูปสัญญาณ ดังรูปที่ 4.2



รูปที่ 4.2 สัญญาณวิดีโอหลังจากผ่านวงจรขยาย

เปลี่ยนสัญญาณในช่องที่สองไปวัดที่ขาแอโนดของ D3 ซึ่งทำงานเป็นวงจรจำกัดระดับสัญญาณ (Limitter) ได้รูปสัญญาณดังรูปที่ 4.3



รูปที่ 4.3 สัญญาณหลังจากผ่านวงจรจำกัดระดับสัญญาณ

4.2 การสร้างคอนโวลเวอร์ขนาด 3x3 บน FPGA

สำหรับปริญญานิพนธ์นี้จะทำการสร้างคอนโวลเวอร์ขนาด 3x3 ลงบน FPGA ของบริษัท Xilinx โดยใช้โปรแกรมชื่อ Xilinx Foundation Series โดยมีขั้นตอนการสร้างดังต่อไปนี้

4.2.1 โปรแกรมวงจรลงบน FPGA

ใช้หลักการออกแบบลำดับขั้น (Hierarchy) โดยออกผังการทำงานโดยรวมซึ่งจะแสดงถึงการเชื่อมต่อกันระหว่างส่วนย่อยต่างๆ และการเชื่อมต่อกับขาภายนอกตัว FPGA จากรูปจะเห็นได้ว่า ข้อมูลภาพจะเข้าสู่ FPGA ทางบัส A B C ผ่านแลต์ (Latch) ไปเข้าส่วนประมวลผลแบบทวินาม และ สัญญาณนาฬิกาเข้ามาทางพอร์ต PCLK ผ่านบัฟเฟอร์ไปยังส่วนต่างๆของระบบ ผลลัพธ์ที่ได้จะถูกส่งออกทางบัส Q ผ่านทางแลตช์ ซึ่งจะเห็นได้ว่าสมมูลย์กับขาที่ได้ออกแบบไว้ในบทที่ 3 ผังโดยรวมแสดงได้ดังรูปที่ 4.4 หลังจากนั้นจึงออกแบบส่วนที่ทำการประมวลผลการคอนโวลูชั่นแบบทวินาม ซึ่งคือส่วน BINOMIAL ในรูป ที่ 4.4 ใช้วงจรในรูปที่ 3.7

4.2.2 การนำผลที่ได้ไปใช้งาน

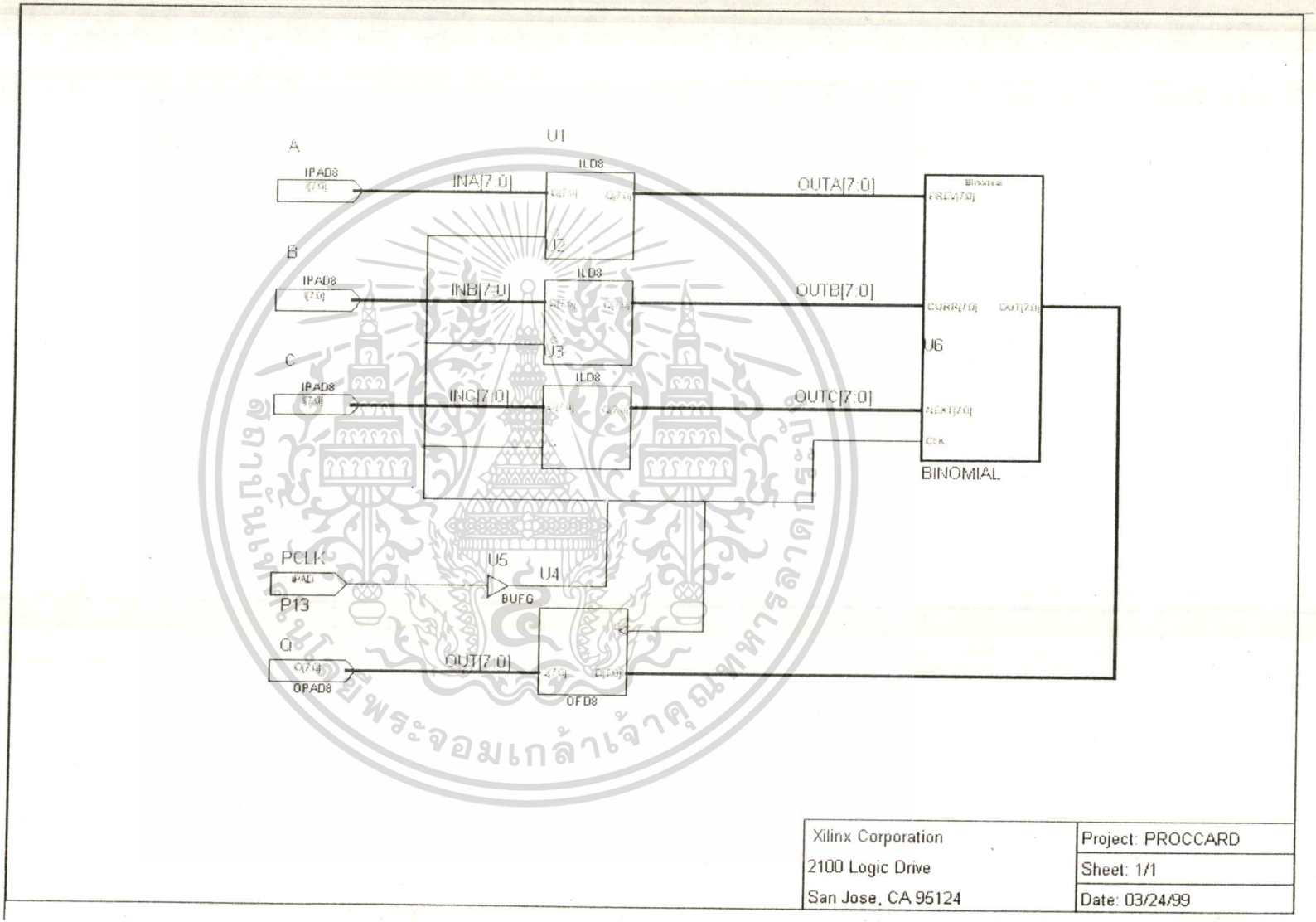
แฟ้มข้อมูลที่จะนำไปใช้คือแฟ้มที่มีส่วนขยาย .BIT ซึ่งจะนำไปโปรแกรมลงบน FPGA โดยใช้ร่วมกับโปรแกรมที่ได้ออกแบบในบทที่ 3 อยู่ในรูปแบบไบนารี (Binary) และแฟ้มที่มีส่วนขยาย .RBT เป็นแฟ้มที่ใช้อ้างอิงตรวจสอบบิตซึ่งอยู่ในรูปไฟล์แอสกี้ (ASCII) ดังแสดงรูป 4.5

4.3 การทดลองการใช้งานจริง

สำหรับปริญญานิพนธ์นี้จะนำเสนอการใช้งานการ์ดประมวลผลภาพแบบโปรแกรมได้ที่สร้างขึ้นไปประยุกต์ใช้ในการลดสัญญาณรบกวนภายในภาพวิดีโอ ขั้นตอนการทดลองมีดังนี้

4.3.1 แปลงสัญญาณภาพวิดีโอดั้งเดิม

ทำการสร้างระบบในทำนองเดียวกับในข้อ 4.2 แต่แทนที่จะผ่านแมสค์ทวินามก็ทำการเชื่อมต่อพอร์ต A (อินพุท) เข้ากับพอร์ต Q (เอาท์พุท) ผ่านทางแลตช์ที่ IOB เท่านั้น เมื่อโปรแกรมบิตสตรีมที่ได้ลงบน FPGA แล้วทดลองเก็บภาพจริงได้ผลดังรูปที่ 4.6 ทำการทดลองในทำนองเดียวกันแต่เปลี่ยนบิตสตรีมเป็นแบบที่ใช้คอนโวลเวอร์ทวินามที่ได้สร้างขึ้น ได้ผลดังรูปที่ 4.7 เมื่อพิจารณาจากผลฮิสโตรแกรม (Histrogram) จะเห็นได้ว่าสามารถลดสัญญาณรบกวนได้อย่างมีนัยสำคัญ

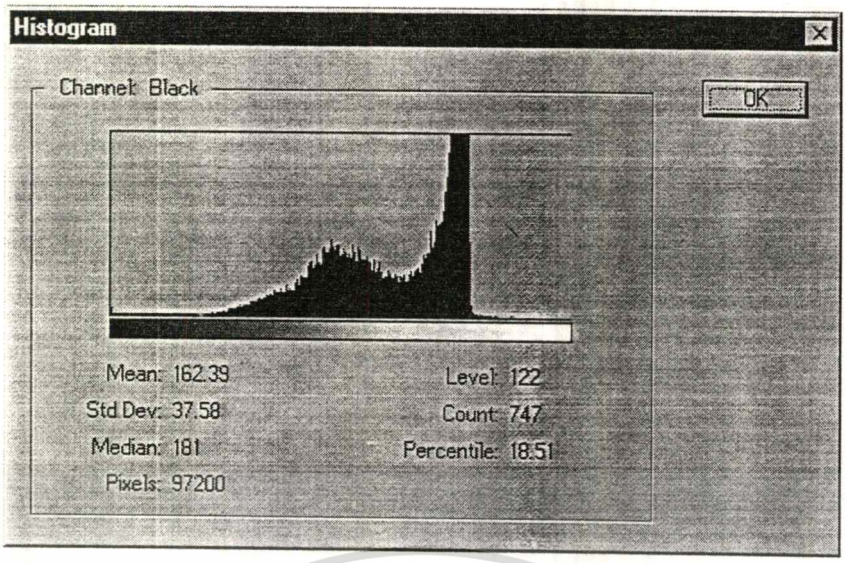


รูปที่ 4.4 ผังโครงสร้างของระบบ

```
Xilinx ASCII Bitstream
Created by Bitstream M1.4.12
Design name:     proccard.ncd
Architecture:    xc4000e
Part:            4010epc84
Date:            Wed Mar 24 22:15:20 1999
Bits:            178144
11111111001000000001010110111110110011111
00010111111111110111111110101111111010111111101011111110101111111010111111101011111110101111111
01111111111111101111111110111111111011111111101111111110111111111011111111101111111110111111111
01111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
01111111111111110111111111011111111101111111110111111111011111111101111111110111111111011111111
01111111111111111010111111101011111110101111111010111111101011111110101111111010111111101011111
01111111111111111010111111101011111110101111111010111111101011111110101111111010111111101011111
00111111111111111100111111110011111111001111111100111111110011111111001111111100111111110011111
01111111111111111011111111101111111110111111111011111111101111111110111111111011111111101111111
01111111111111111010111111101011111110101111111010111111101011111110101111111010111111101011111
00111111111111111101111111110111111111011111111101111111110111111111011111111101111111110111111
01111111111111111101111111110111111111011111111101111111110111111111011111111101111111110111111
01111111111111111011011111101101111110110111111011011111101101111110110111111011011111101101111
01111111111111111011111111101111111110111111111011111111101111111110111111111011111111101111111
01111111111111111011111111101111111110111111111011111111101111111110111111111011111111101111111
01101111111111110011111111100111111111001111111100111111110011111111001111111100111111110011111
01111111111111111011111111101111111110111111111011111111101111111110111111111011111111101111111
01101110111111100111111110011111111001111111100111111110011111111001111111100111111110011111111
01111111111111111011111111101111111110111111111011111111101111111110111111111011111111101111111
01111111111111111011111111101111111110111111111011111111101111111110111111111011111111101111111
01110111111111101101111111011011111101101111110110111111011011111101101111110110111111011011111
01111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
01111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
01111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
01111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
01111111111111111011111111101111111110111111111011111111101111111110111111111011111111101111111
01111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
01111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
01111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111111
```

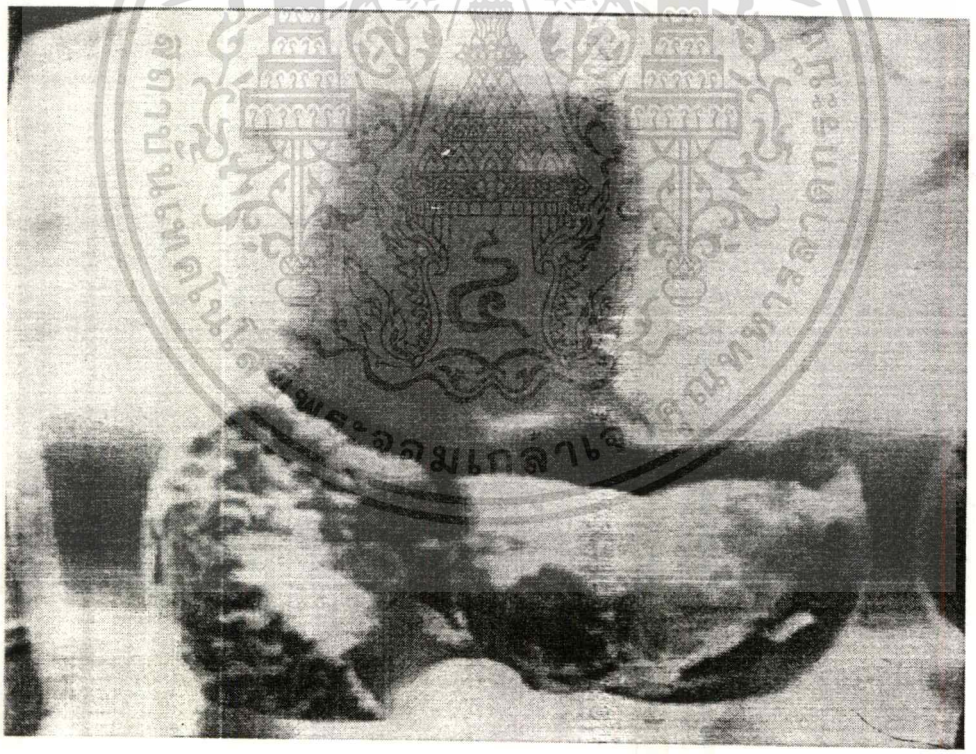
รูปที่ 4.5 แฟ้มข้อมูล RBT สำหรับการอ้างอิงตรวจสอบ

ก. ภาพดั้งเดิมจากกล้อง CCD

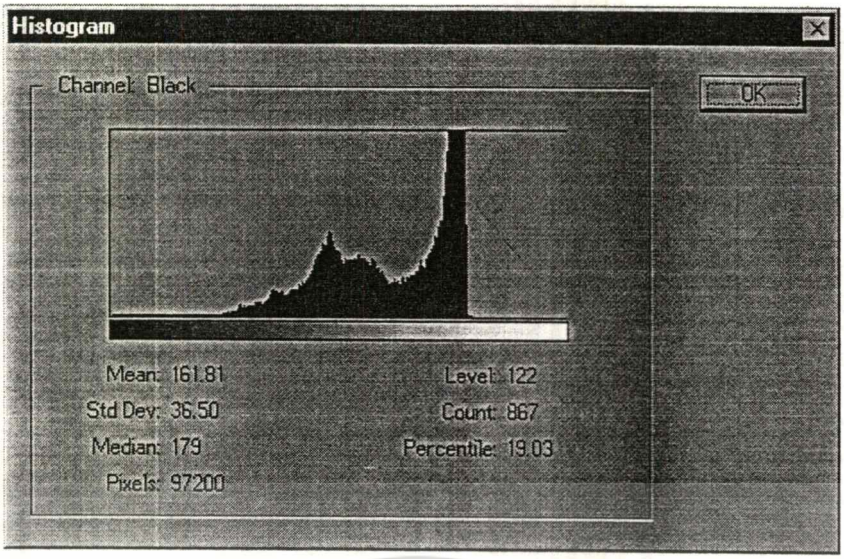


ข. ฮิสโตรแกรมของภาพดั้งเดิมจากกล้อง CCD

รูปที่ 4.6 ภาพดั้งเดิมจากกล้อง CCD เมื่อเชื่อมต่อพอร์ต A โดยตรงกับพอร์ต Q และฮิสโตรแกรม



ก. ภาพเมื่อผ่านคอนโวลเวอร์แบบทวินาม



ข. ฮิสโตรแกรมของภาพเมื่อผ่านคอลโวลเวอร์แบบทวินาม

รูปที่ 4.7 ภาพที่ได้หลังจากผ่านคอนโวลเวอร์แบบทวินาม

บทที่ 5
สรุปและวิจารณ์

5.1 วิเคราะห์ผลการทดลอง

การ์ดประมวลผลภาพดิจิตอลสำหรับโครงงานนี้ออกแบบเพื่อรองรับการประมวลผลใดๆ ภายในเมตริกซ์ขนาด 3x3 เนื่องจากเป็นขนาดที่ทำให้ระบบประหยัดที่สุดในขณะที่ยังคงสามารถทำงานให้ผลเป็นที่น่าพอใจ ซึ่งจะเห็นได้ว่าภาพผลลัพธ์หลังจากผ่านระบบที่ออกแบบให้ผลลัพธ์ตามจุดประสงค์คือสามารถลดผลของสัญญาณรบกวนได้ และมีลักษณะใกล้เคียงกับภาพต้นฉบับ

สำหรับในส่วนของวงจร โดยเฉพาะอย่างยิ่งส่วนที่ทำหน้าที่ปรับรูปแบบสัญญาณให้เหมาะสม จากผลการทดลองจะเห็นได้ว่า ถึงแม้ว่ารูปสัญญาณจะมีความใกล้เคียงกับสัญญาณเดิม แต่ยังเห็นความผิดเพี้ยนอยู่ จากการสังเกตพบว่า ลักษณะของสัญญาณเป็นความเพี้ยนเนื่องจากแถบความถี่สูงถูกลดทอนลงไป ซึ่งอาจเกิดขึ้นจาก แถบความถี่-อัตราขยายของออปแอมป์มีค่าไม่เพียงพอ, หรือไดโอดทำงานไม่ทันกับอัตราการเปลี่ยนแปลงแรงดันของสัญญาณ ซึ่งทั้งนี้จะได้ทำการวิเคราะห์หาสาเหตุที่แท้จริงและทำการแก้ไขต่อไป

5.2 ข้อดีของโครงงาน

ข้อดีของโครงงานสามารถจำแนกได้เป็นข้อๆ ดังนี้

- ระบบสามารถทำงานได้ในกับการประยุกต์ใช้งานแบบเวลาจริง
- ระบบมีความเชื่อถือได้สูงเนื่องจากจำนวนอุปกรณ์มีขนาดเล็กลง
- การเชื่อมต่อกับคอมพิวเตอร์ทำได้ง่าย
- การเปลี่ยนแปลงแก้ไขกระบวนการสามารถทำได้ผ่านทางเครื่องคอมพิวเตอร์ ทำให้เกิดความยืดหยุ่นในการใช้งาน

5.3 ข้อด้อยของโครงงาน

ข้อด้อยของโครงงานสามารถจำแนกไดเป็นข้อๆ ดังนี้

- ระบบเริ่มต้นมีราคาสูง เหมาะสำหรับการทำเครื่องต้นแบบ หรือผลิตจำนวนน้อย
- ประสิทธิภาพถูกจำกัดอยู่ที่อุปกรณ์แวดล้อมอื่นๆด้วย

5.4 แนวทางในการพัฒนาต่อ

แนวทางการพัฒนาต่อคือนำไปประยุกต์ใช้ในขั้นตอนประมวลผลเริ่มแรก (Pre-processing) ในกระบวนการติดตามการเคลื่อนไหว (Motion Tracking) หรือ การลดผลของสัญญาณรบกวนแบบเวลาจริง เป็นต้น

ภาคผนวก ก

โปรแกรมควบคุมการทำงานของการ์ดประมวลผลภาพดิจิตอลแบบโปรแกรมได้

แฟ้มข้อมูล ImgPro.H

```
/*********************************************************************
 * File           : IMGPRO.H : Image Processing Core Header          *
 * Project        : IMGPRO.PRJ : Digital Image Processing Card Driver*
 * Programmer     : Mr Paramate HORKAEW, KMITL                       *
 *********************************************************************/

#ifndef     IMGPRO_HEADER
#define     IMGPRO_HEADER

#define     IMG_WIDTH   384
#define     IMG_HEIGHT  288

#define     SOFT_TITLE  "Reprogrammable Digital Image Processing Card"
#define     SOFT_NAME   "RealTimeMask -Lite (RTM for DOS)"

#define     COMMAND             0x310
#define     CONFIG              0x311
#define     PIXDATA             0x312
#define     PROGTRIG            0x313
#define     WAITPORT            0x314

#define     DEVICE_DELAY        900000L

#endif      //    IMGPRO_HEADER
```

แฟ้มข้อมูล ImgMain.C

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <stdlib.h>
#include <dos.h>

#include "imgpro.h"
#include "imggraph.h"

#define F1      59
#define F2      60
#define F3      61
#define F4      62
#define F5      63
#define F6      64
#define F7      65
#define F8      66
#define F9      67
#define F10     68

void    Help(void);
void    Save(void);
void    Config(void);
void    Capture(void);
void    Live(void);
void    SetMask(int Namber);
void    WaitProcess(long Delay);
unsigned char     Traverse(unsigned char iData);
```

```
void    beep(void);

void    UserActivate(void);

static int Mask = 1;
static int bLive        = 1;

void    main(void)
{
      InitGraphic();
      DrawScreen();
      UserActivate();
      CloseGraphic();
}

void    UserActivate(void)
{
      char    ch;
      int     func;

      do
      {
            ch = getch();
            if (ch==0)                    //      Function keys
            {
                  func = 1;
                  ch = getch();
                  switch (ch)
                  {
                        case    F1      :       Help();
                                                break;
                        case    F2      :       Save();
                                                break;
                        case    F3      :       Config();
                                                break;
                        case    F4      :       Capture();
                                                break;
                        case    F5      :       Live();
                                                break;
                        case    F10     :       break;
                  }
            }
            else                                //      Normal keys
            {
                  func = 0;
                  SetMask(ch-0x30);
            }
      } while (!(func==1 && ch==F10));
}

void    Help(void)
{
      Message("Help");
}

void    Save(void)
{
      int         Row, Col;
      unsigned char cData;
```

```
        FILE*       fp;

        Message("Save image to : UNTITLED.RAW");

        //      Disable Interupt/DMA
        asm     cli;

        fp = fopen("UNTITLED.RAW", "wb");

        //      Read frame from port
        //      Float data bus
        outportb(COMMAND, 0xF1);        //      RSTR = 0, OE = 0, RE = 1, Oths 1
        inportb(PIXDATA);
        //      Reset RAM Address
        outportb(COMMAND, 0xF4);        //      RSTR = 1, OE = 0, RE = 0, Oths 1
        inportb(PIXDATA);
        outportb(COMMAND, 0xF4);        //      RSTR = 1, OE = 0, RE = 0, Oths 1
        inportb(PIXDATA);

        //    Enable reading
        outportb(COMMAND, 0xF3);        //      RSTR = 0, OE = 1, RE = 1, Oths 1

        for (Row=0; Row<IMG_HEIGHT; Row++)
        {
              for (Col=0; Col<IMG_WIDTH; Col++)
              {     //      Read Pixel
                    char  str[25];
                    cData = inportb(PIXDATA);
                    fputc(Traverse(cData), fp);
              }
        }

        //      Float data bus
        outportb(COMMAND, 0xF1);        //      RSTR = 0, OE = 0, RE = 1, Oths 1
        inportb(PIXDATA);

        fclose(fp);
        //      Enable Interupt/DMA
        asm     sti;
        Message("Save complete");
        beep();
}

void    WaitProcess(long Delay)
{
        long    wait;

        for (wait=0L; wait<Delay; wait++)
              inportb(WAITPORT);
}

void    Config(void)
{
        char*   szConfigName = "MASK_.BIT";
        unsigned char   cData;
        int     iData;
        int     i;
        FILE*   fp;
        long    bcnt;
```

```
        szConfigName[4] = Mask+0x30;

        if ((fp=fopen(szConfigName, "rb"))==NULL)
              return;
        //      Disable Interupt/DMA
        asm     cli;

        //      Float data bus
        outportb(COMMAND, 0xF1);      //      RSTR = 0, OE = 0, RE = 1, Oths 1-
        inportb(PIXDATA);
        //      Pulse /PROGRAM "LOW" -> "HIGH"
        inportb(PROGTRIG);

        Message("Clear config memory, wait...");
        WaitProcess(DEVICE_DELAY);
        //      Ready to config
        Message("Configuration, wait...");

        //    Initialize shifting clock
        outportb(CONFIG, cData);
        do
        {
              char  str[15];
              iData = fgetc(fp);
              cData = Traverse((unsigned char) iData);
              //      Config Xilinx through port
              outportb(CONFIG, cData);
              for (i=0; i<7; i++)
              {
                    outportb(CONFIG, cData);          //    7 shifting clock
              }
        }
        while (iData!=EOF);

        for (i=0; i<10; i++)
        {
              outportb(CONFIG, 0xFF);       //    10 more shifting clock
        }
        //      Enable Interupt/DMA
        asm     sti;
        fclose(fp);

        Message("Configuration complete");
        beep();
}

void    Capture(void)
{
        int     Row, Col;
        unsigned char cData;

        RefreshScreen();

        //      Disable Interupt/DMA
        asm     cli;

        //    Float data bus
        outportb(COMMAND, 0xF1);      //  RSTR = 0, OE = 0, RE = 1, Oths 1
        inportb(PIXDATA);
        //    Capture frame
```

```
      Message("Capture frame, wait...");
      outportb(COMMAND, 0x7F);      //  MAINEN = 0, Oths 1
      //    Frame ready
      outportb(COMMAND, 0xFF);      //  MAINEN = 1, Oths 1
      WaitProcess(270000);
      //      Read frame from port
      //      Float data bus
      outportb(COMMAND, 0xF1);      //      RSTR = 0, OE = 0, RE = 1, Oths 1
      inportb(PIXDATA);
      //      Reset RAM Address
      outportb(COMMAND, 0xF4);      //      RSTR = 1, OE = 0, RE = 0, Oths 1
      inportb(PIXDATA);
      outportb(COMMAND, 0xF4);      //      RSTR = 1, OE = 0, RE = 0, Oths 1
      inportb(PIXDATA);

      //    Enable reading
      outportb(COMMAND, 0xF3);      //      RSTR = 0, OE = 1, RE = 1, Oths 1

      for (Row=0; Row<IMG_HEIGHT; Row++)
      {
            for (Col=0; Col<IMG_WIDTH; Col++)
            {
//      Read Pixel
                  char  str[25];
                  cData = inportb(PIXDATA);
                  DispPixel(Col, Row, Traverse(cData));
            }
      }

      //      Float data bus
      outportb(COMMAND, 0xF1);      //      RSTR = 0, OE = 0, RE = 1, Oths 1
      inportb(PIXDATA);

      //      Enable Interupt/DMA
      asm     sti;
      Message("Capture complete");
      beep();
}

void    Live(void)
{
      //      Loop : Read frame from port
      while (!kbhit())
      {
            Capture();
            delay(100);
      }
      //      Clear key buffer
      if (getch()==0) getch();
}

void    SetMask(int Number)
{
      if (Number>0 && Number<9)
      {
            char*   szMtrxName = "MASK_.MAT";
            char*   msg = "Select mask number _";
            FILE*   fp;
            int     h[9], i;
            char    szNum[5];
```

```
            Mask = Number;
            szMtrxName[4] = Mask+0x30;
            msg[19] = Mask+0x30;

            Message(szMtrxName);
            if ((fp=fopen(szMtrxName, "rt"))==NULL)
                  return;

            Message(msg);
            for (i=0; i<9; i++)
            {
                  fgets(szNum, 5, fp);
                  h[i] = atoi(szNum);
            }

            fclose(fp);
            DrawMask(h);
            beep();
      }
}

void    beep(void)
{
      sound(1000);
      delay(50);
      nosound();
}

unsigned char     Traverse(unsigned char iData)
{
      unsigned char     oData, iMark, oMark;
      int   i = 0;

      oData = 0x00;
      iMark = 0x01;
      oMark = 0x80;
      do
      {
            if (iMark&iData)
                  oData |= oMark;
            iMark<<=1;
            oMark>>=1;
            i++;
      } while (i<8);
      return oData;
}
```

แฟ้มข้อมูล ImgGraph.H

```
/*************************************************************************
 * File           : IMGGRAPH.H : Image Processing Graphic Header     *
 * Project        : IMGPRO.PRJ : Digital Image Processing Card Driver*
 * Programmer     : Mr Paramate HORKAEW, KMITL                       *
 *************************************************************************/

#ifndef     IMGGRAPH_HEADER
#define     IMGGRAPH_HEADER

#define     LOGO_NAME   "LOGO.RAW"
```

```
#define     LOGO_LEFT   540
#define     LOGO_TOP          339

// super vga 256 colors
typedef unsigned char DacPalette256[256][3];
extern int far _Cdecl Svga256_fdriver[];
int   InitGraphic(void);
void  CloseGraphic(void);
void  SetPalette(int sform);
void  DrawScreen(void);
void  DrawMask(int* h);
void  DispPixel(int x, int y, int data256);
void  RefreshScreen(void);
void  Message(char* text);

#endif      //    IMGGRAPH_HEADER
```

แฟ้มข้อมูล ImgGraph.C

```
#include <stdio.h>
#include <graphics.h>
#include <dos.h>
#include <string.h>
#include <stdlib.h>
#include "imgpro.h"
#include "imggraph.h"

void    setvgapalette256(DacPalette256 * PalBuf);
int huge DetectVGA256(void);
void    DrawButton(int x, int y, int width, int height, char* text);
void    DrawLogo(void);

int     left, top;

int InitGraphic(void)
{
      int gr_driver=DETECT, gr_mode, errorcode;

      installuserdriver("Svga256", DetectVGA256);
      registerfarbgidriver(Svga256_fdriver);
      initgraph(&gr_driver, &gr_mode, "");
      errorcode = graphresult();    // read result of initialization
      if (errorcode != grOk)        // an error occurred
      {
            printf("Graphics initialization");
            return -1;
      }
      settextstyle(DEFAULT_FONT, HORIZ_DIR, 1);
      settextjustify(LEFT_TEXT, CENTER_TEXT);
      setfillstyle(SOLID_FILL, BLACK);
      SetPalette(1);   // set gray palette full range

      left    = (getmaxx()-IMG_WIDTH)/2;
      top     = (getmaxy()-IMG_HEIGHT)/2;

      return 0;
}
```

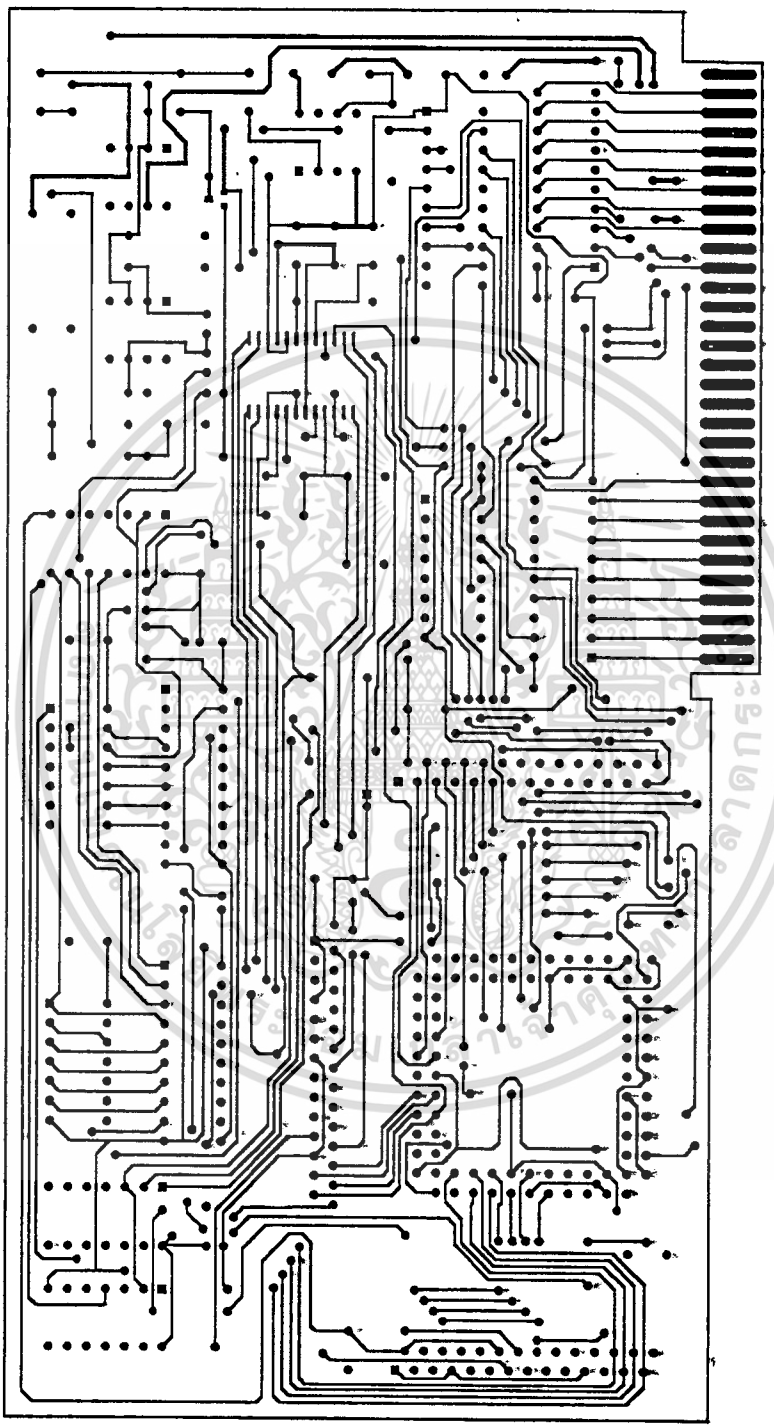
```
void CloseGraphic(void)
{
      closegraph();
}

int huge DetectVGA256(void)
{
      return 2;          // Svga 640 x 480 x 256
}

void setvgapalette256(DacPalette256 * PalBuf)
{
      struct REGPACK  reg;

      reg.r_ax = 0x1012;
      reg.r_bx = 0;
      reg.r_cx = 256;
      reg.r_es = FP_SEG(PalBuf);
      reg.r_dx = FP_OFF(PalBuf);
      intr(0x10, &reg);
}

void SetPalette(int sform) /* 0-adjuest, 1-reset */
{
      int i, NWL, NWH, OWL, OWH, sp, ep;
      static unsigned char palette[256][3];
      float delta, ps;

      if(sform)
      {
            for (i = 0; i < 64; i++)
            {
                  palette[i][0] = palette[i][1] = palette[i][2] = i;
            }
      }
      setvgapalette256(&palette);
      setcolor(63);
}

void    DrawButton(int x, int y, int width, int height, char* text)
{
      setcolor(48);
      rectangle(x, y, x+width, y+height);
      setcolor(63);
      outtextxy(x+(width-(strlen(text)*8))/2, y+height/2, text);
}

void    DrawScreen(void)
{
      char*   szSoftTitle = SOFT_TITLE;
      char*   szSoftName      = SOFT_NAME;

      setcolor(63);
      outtextxy((getmaxx()-strlen(szSoftTitle)*8)/2, 35, szSoftTitle);
      outtextxy((getmaxx()-strlen(szSoftName)*8)/2, 50, szSoftName);

      //      Draw frame
      rectangle(left-1, top-1, left+IMG_WIDTH, top+IMG_HEIGHT);
```

```
	//	Draw function buttons
	DrawButton( 20, 430, 90, 30, "F1-HELP");
	DrawButton(120, 430, 90, 30, "F2-SAVE");
	DrawButton(220, 430, 90, 30, "F3-CONFIG");
	DrawButton(320, 430, 90, 30, "F4-CAPTURE");
	DrawButton(420, 430, 90, 30, "F5-LIVE");
	DrawButton(520, 430, 90, 30, "F10-QUIT");

	//	Draw mask button
	DrawButton(10, top, 30, 30, "1");
	DrawButton(50, top, 30, 30, "2");
	DrawButton(10, top+40, 30, 30, "3");
	DrawButton(50, top+40, 30, 30, "4");

	//	Draw Mask
	DrawMask(NULL);

	DrawLogo();
}

void	DrawMask(int* h)
{
	int	i, j;

	bar(10, 304, 100, 394);
	for (i=0; i<3; i++)
		for (j=0; j<3; j++)
		{
			if (h!=NULL)
			{
				char	element[4];
				itoa(h[3*i+j], element, 10);
				DrawButton(10+(30*j), 304+(30*i), 30, 30, element);
			}
			else
				DrawButton(10+(30*j), 304+(30*i), 30, 30, "#");
		}
}

void	DrawLogo()
{
	unsigned char	cWidth, cHeight, i, j;
	unsigned	char*	aLine;
	FILE*	fp;

	if ((fp=fopen(LOGO_NAME, "rb"))==NULL)
		return;

	fread(&cWidth, sizeof(cWidth), 1, fp);
	fread(&cHeight, sizeof(cHeight), 1, fp);

	if ((aLine=(unsigned char *) malloc(cWidth))==NULL)
		return;

	for (i=0; i<cHeight; i++)
	{
		fread(aLine, sizeof(unsigned char), cWidth, fp);
		for (j=0; j<cWidth; j++)
		{
			int	pixel = aLine[j]/4;
```

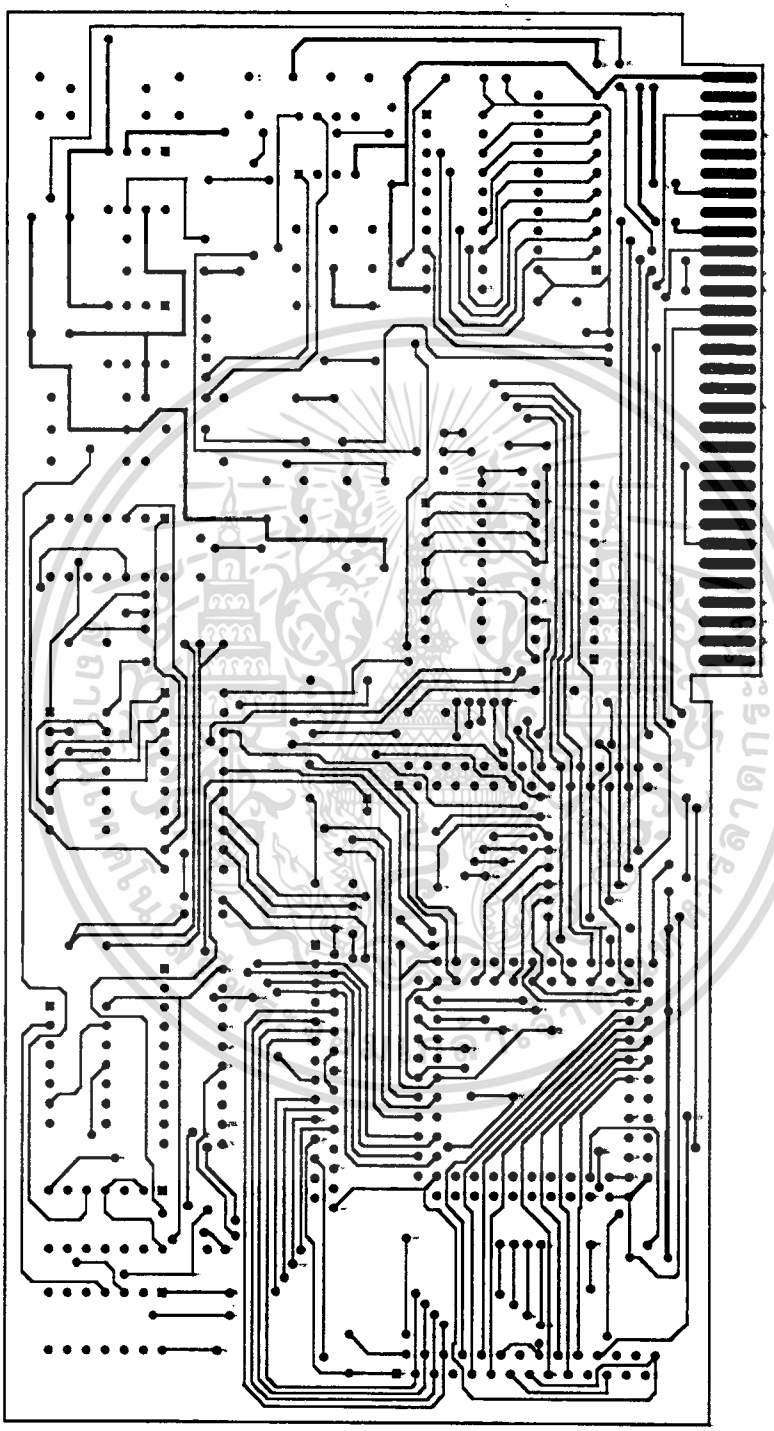
```
                        putpixel(LOGO_LEFT+j, LOGO_TOP+i, pixel);
                }
        }

        if (aLine)
                free(aLine);
        fclose(fp);
}

void    DispPixel(int x, int y, int data256)
{
        if (x>22 && y>7)
                putpixel(left+x-22, top+y-7, data256/4);
}

void    RefreshScreen()
{
        bar(left-1, top-1, left+IMG_WIDTH, top+IMG_HEIGHT);
        rectangle(left-1, top-1, left+IMG_WIDTH, top+IMG_HEIGHT);
}

void    Message(char* text)
{
        bar(left, top+IMG_HEIGHT+10, left+IMG_WIDTH, top+IMG_HEIGHT+20);
        setcolor(63);
        outtextxy(left, top+IMG_HEIGHT+15, "^");
        setcolor(56);
        outtextxy(left+16, top+IMG_HEIGHT+15, text);
}
```

ภาคผนวก ข

แผงวงจรพิมพ์ของการ์ดประมวลผลภาพดิจิตอลแบบโปรแกรมได้



ด้านบน



ด้านล่าง

กิตติกรรมประกาศ

ในโอกาสนี้ขอขอบพระคุณ ดร.ยุทธพงศ์ รังสรรค์เสรี ซึ่งเป็นผู้แนะนำให้คำปรึกษาเทคนิคและวิธีการออกแบบโครงงาน ดร.สุธี ผู้เจริญชนะชัย ที่ให้โอกาสและแนะแนวทางในการวิจัยที่ถูกต้อง นพ.ราเมศร์ วชิรสินธุ์ ผู้จุดประกายเริ่มต้นสำหรับโครงงานนี้ และสนับสนุนสถานที่พร้อมทั้งอุปกรณ์ในการทดสอบ คุณสมชาย เกรียงอารีกุล ที่อบรมและฝึกฝนการเขียนโปรแกรมให้ข้าพเจ้า และขอขอบคุณ คุณมณีสุดา ชะมด ซึ่งเป็นกำลังใจให้เสมอมาจนโครงงานนี้สำเร็จลุล่วงไปด้วยดี สุดท้ายนี้ขอกราบขอบพระคุณ คุณแคล้ว และคุณบัวงาม ห่อแก้ว ที่คอยสอบถามความก้าวหน้า และห่วงใยอยู่เสมอ

เอกสารอ้างอิง

1. Ronald N. Bracewell, TWO-DIMENSIONAL IMAGING, 1995, Prentice-Hall International, Inc.
2. Robert J. Schalkoff, DIGITAL IMAGE PROCESSINF AND COMPUTER VISION, 1989, John Wiley & Sons, Inc.
3. John C. Russ, THE IMAGE PROCESSING HANDBOOK 2nd Edition, 1994, CRC Press
4. D. M. Etter, ENGINEERING PROBLEM SOLVING WITH MATLAB, 1993, Prentice-Hall International, Inc.
5. Adrian Brian/Moshe Breiner ,MATLAB FOR ENGINEERING, 1995, Addison-Wesley
6. David F. Stout/Milton Kaufman, HANDBOOK OF OPERATIONAL AMPLIFIER CIRCUIT DESIGN, 1976,McGraw-Hill Book Company
7. David Pellerin/Michael Holley, DIGITAL DESIGN USING ABEL, 1994, Prentice-Hall International, Inc.
8. Vinai Piyatanasirikul and Somchai Jitapunkul, Application of Digital Image Processing for Cracking on Road Surface, 2540, การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 20
9. ดร.ไพรัช ธัชยพงษ์, การประมวลสัญญาณดิจิตอล ตอนการออกแบบวงจรกรองดิจิตอล, 2535, ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ
10. ปรเมศวร์ ห่อแก้ว และ ผศ.ดร.ยุทธพงษ์ รังสรรค์เสรี, การ์ดประมวลผลภาพดิจิตอลแบบโปรแกรมได้, 2542, การประชุมวิชาการแห่งมหาวิทยาลัยเกษตรศาสตร์ ครั้งที่ 37