การออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์สำหรับการประมวลผลหาเส้นตรงบนภาพด้วยเอฟพีจีเอ

Hardware/Software Co-design for Line Detection Algorithm on FPGA

วรุตม์ ขยันกิจ

Warut Khayankit

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Computer Engineering
Prince of Songkla University
2552

ชื่อวิทยานิพนธ์ การออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์สำหรับการประมวลผลหาเส้นตรงบนภาพด้วยเอฟพีจีเอ

ผู้เขียน นายวรุตม์ ขยันกิจ

สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

……..... (ดร.วรรณรัช สันติอมรทัต)

อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม

……..... (รองศาสตราจารย์ ดร.มนตรี กาญจนะเดชะ)

……..... (ผู้ช่วยศาสตราจารย์ ดร.ณัฏฐา จินดาเพ็ชร์)

คณะกรรมการสอบ

……..ประธานกรรมการ (ผู้ช่วยศาสตราจารย์ ดร.พรชัย พฤกษ์ภัทรานนต์)

……..กรรมการ (รองศาสตราจารย์ ดร.สมศักดิ์ มิตะถา)

……..กรรมการ (ดร.วรรณรัช สันติอมรทัต)

……..กรรมการ (รองศาสตราจารย์ ดร.มนตรี กาญจนะเดชะ)

……..กรรมการ (ผู้ช่วยศาสตราจารย์ ดร.ณัฏฐา จินดาเพ็ชร์)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

……

(รองศาสตราจารย์ ดร.เกริกชัย ทองหนู)

คณบดีบัณฑิตวิทยาลัย

ชื่อวิทยานิพนธ์ การออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์สำหรับการประมวลผลหาเส้นตรงบนภาพด้วยเอฟพีจีเอ
ผู้เขียน นายวรุตม์ ขยันกิจ
สาขาวิชา วิศวกรรมคอมพิวเตอร์
ปีการศึกษา 2551

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้นำเสนอการใช้เทคโนโลยีเอฟพีจีเอเข้ามาเป็นตัวช่วยในการประมวลผลร่วมกับหน่วยประมวลผลบนบอร์ดเอฟพีจีเอเพื่อเพิ่มประสิทธิภาพในการประมวลผลสัญญาณภาพให้ดียิ่งขึ้น โดยใช้ภาษา ImpulseC ในการพัฒนาระบบและซอฟต์แวร์ช่วยออกแบบระบบ Impulse CoDeveloper จะทำหน้าที่ในการแปลงให้อยู่ในรูปแบบของภาษาอธิบายพฤติกรรมของฮาร์ดแวร์ เพื่อให้สามารถสังเคราะห์เป็นวงจรที่ใช้บนเทคโนโลยีเอฟพีจีเอได้ ดังนั้น งานวิจัยนี้จึงเป็นการออกแบบและพัฒนาระบบร่วมฮาร์ดแวร์และซอฟต์แวร์โดยใช้การประมวลผลหาเส้นตรงบนภาพเป็นกรณีศึกษา ซึ่งสามารถแบ่งระบบออกได้เป็น 2 ส่วนหลัก คือ การหาระบบภาพโดยทำการเปรียบเทียบประสิทธิภาพของระบบการหาขอบภาพด้วยอัลกอริทึมต่างๆ ได้แก่ Robert, Sobel และ Prewitt ด้วยการประมวลแบบ single core และการหาเส้นตรงด้วยเทคนิค Hough Transform โดยกำหนดการทดสอบระบบด้วยการใช้ภาพในสกุล bmp และให้ระบบแสดงผลลัพธ์เป็นรูปแบบของ text file ผลที่ได้จากการทดสอบภาพขนาด 200x200 pixels พบว่าระบบสามารถประมวลผลได้เสร็จภายในเวลา 34.507 ns และใช้ทรัพยากรของ FPGA Xilinx Spartan3E 64%

คำสำคัญ: เอฟพีจีเอ, ImpulseC, Edge Detection, Hough Transform

Thesis Title	Hardware/Software Co-design for Line Detection Algorithm on FPGA
Author	Mr.Warut Khayankit
Major Program	Computer Engineering
Academic Year	2008

ABSTRACT

This thesis presents the usage of FPGA technology to help in line detection algorithm with co-design on FPGA for making image signal processing more efficient by using C-base programming named ImpulseC. By system and software development to help system design, Impulse co-developer has been employed to tell about hardware for synthesis the system on FPGA technology. This research shows the design methodology of hardware/software co-design on FPGA by using line detection algorithm to be the case study which is divided into 2 main components, the one is edge detection module, with various algorithm, Robert, Sobel and Prewitt in single core, the other one is Hough transform with input file picture bmp make the system show the result in text file. From the testing of 200x200 pixels, it is found that the system both in hardware/software for line detection on FPGA is simulated in 34.507 ns and use the resource on the FPGA Xilinx Spartan3E equal to 64%.

Keywords: FPGA, ImpulseC, Edge Detection, Hough Transform

กิตติกรรมประกาศ

ขอขอบพระคุณ ดร.วรรณรัช สันติอมรทัต อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก ที่ได้ให้คำปรึกษา ชี้แนะแนวทาง และให้ความรู้ในด้านต่างๆ รวมถึงการให้การสนับสนุนในเรื่องอุปกรณ์ในการทำวิจัย ตลอดจนช่วยตรวจและแก้ไขวิทยานิพนธ์ให้เป็นไปอย่างสมบูรณ์

ขอขอบพระคุณรองศาสตราจารย์ ดร.มนตรี กาญจนะเดชะ และ ผู้ช่วยศาสตราจารย์ ดร.ณัฏฐา จินดาเพ็ชร์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม ที่ได้ให้คำปรึกษา ชี้แนะแนวทาง และให้ความรู้ในด้านต่างๆ รวมถึงการให้การสนับสนุนในเรื่องอุปกรณ์ในการทำวิจัย ตลอดจนช่วยตรวจและแก้ไขวิทยานิพนธ์ให้เป็นไปอย่างสมบูรณ์

ขอขอบพระคุณรองศาสตราจารย์ ดร.สมศักดิ์ มิตะถา และ ผู้ช่วยศาสตราจารย์ ดร.พรชัย พฤกษ์ภัทรานนต์ ที่กรุณาให้เกียรติเป็นกรรมการสอบวิทยานิพนธ์ ให้คำแนะนำที่มีประโยชน์ต่อการวิจัยตลอดจนตรวจแก้ไขวิทยานิพนธ์ให้ดำเนินไปอย่างสมบูรณ์

ขอขอบพระคุณคณาจารย์ และบุคลากรในภาควิชาวิศวกรรมคอมพิวเตอร์ทุกท่าน ที่ให้คำปรึกษาและความช่วยเหลือในระหว่างการทำวิทยานิพนธ์

ขอขอบพระคุณบัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ ที่ได้มอบทุนสนับสนุนในการทำวิจัย

ขอขอบคุณนักศึกษาปริญญาโทสาขาวิชาวิศวกรรมคอมพิวเตอร์ทุกท่านที่ได้ให้คำแนะนำและเป็นกำลังใจมาโดยตลอด

และสุดท้ายนี้ ขอน้อมรำลึกถึงพระคุณบิดา มารดา และครอบครัว ที่ส่งเสริมให้กำลังใจ และให้การสนับสนุนในเรื่องต่างๆ จนกระทั่งข้าพเจ้าประสบความสำเร็จในการศึกษา

วรุตม์ ขยันกิจ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

รายการตาราง

ตาราง **หน้า**

รายการภาพประกอบ

ภาพประกอบ **หน้า**

รายการภาพประกอบ (ต่อ)

ภาพประกอบ **หน้า**

สัญลักษณ์คำย่อและตัวย่อ

FPGA	Field Programmable Gate Arrays
VHDL	VHSIC Hardware Description Language
VHSIC	Very High-Speed Integrated Circuit
BMP	Bitmap
CLB	Configurable Logic Block
DSP	Digital Signal Processing

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของหัวข้อวิจัย

เนื่องจากในปัจจุบัน การแข่งขันทางเทคโนโลยีนั้นมีสูงมาก เทคโนโลยีใดที่ให้ประสิทธิภาพสูง เช่น ถูกต้องแม่นยำ ประมวลผลรวดเร็ว ต้นทุนต่ำ เป็นต้น ย่อมได้เปรียบอย่างมาก ไม่ว่าจะเป็นในทางธุรกิจหรืออื่นๆ

การประมวลผลสัญญาณภาพดิจิตอล (digital image processing) [1][2] แบบเรียลไทม์ ได้ถูกนำมาประยุกต์ใช้งานหลากหลายในปัจจุบัน อาทิเช่น การเขียนโปรแกรมติดต่อกับเว็บแคม โปรแกรมตรวจจับและหาตำแหน่งวัตถุ โปรแกรมแยกวัตถุออกจากภาพ โปรแกรมจดจำตัวอักษร (Optical Character Recognition) แต่เนื่องจากการประมวลผลที่มีอยู่ในปัจจุบันส่วนใหญ่นั้น ยังคงต้องพึ่งพาการประมวลผลด้วยหน่วยประมวลผลบนเครื่องคอมพิวเตอร์ทั่วไป ซึ่งมีราคาสูง ขนาดใหญ่ และใช้พลังงานสูง ดังนั้นจึงได้สนใจนำการประมวลผลสัญญาณภาพมาใช้บนระบบสมองกลฝังตัว [3] ซึ่งราคาไม่สูง ประหยัดพลังงาน และมีการประมวลผลที่รวดเร็วใกล้เคียงกับเครื่องคอมพิวเตอร์ทั่วไป งานวิจัยนี้เป็นการใช้เทคโนโลยีเอฟพีจีเอเข้ามาช่วยในการประมวลผลร่วมกับหน่วยประมวลผลบนบอร์ดสมองกลฝังตัวเพื่อเพิ่มประสิทธิภาพในการประมวลผลสัญญาณภาพให้ดียิ่งขึ้น

แม้ว่าคอมพิวเตอร์ในปัจจุบันจะสามารถประมวลผลการทำงานได้รวดเร็ว แต่การประมวลผลสัญญาณภาพ ให้ระบบตอบสนองแบบเรียลไทม์ ยกตัวอย่างเช่น real-time biometric (face, retina, and/or fingerprint) recognition นั้นยังคงต้องใช้คอมพิวเตอร์ที่มีตัวประมวลผลที่ราคาสูง ขนาดใหญ่ และยังสิ้นเปลืองพลังงาน ดังนั้นหากนำเทคโนโลยีทางด้านเอฟพีจีเอมาใช้ในการพัฒนางานแทนหน่วยประมวลผลทั่วไป ก็จะช่วยลดขนาด ราคา และประหยัดพลังงานมากยิ่งขึ้น

ปัจจุบันเทคโนโลยีทางด้านการประมวลผลสัญญาณภาพนั้นถูกพัฒนาบนหน่วยประมวลผลของบริษัท Intel หรือโพรเซสเซอร์ประมวลผลสัญญาณดิจิตอล Digital Signal Processor (DSPs) สำหรับในงานวิจัยนี้จะเสนอการพัฒนาหน่วยประมวลผลภาพบนเทคโนโลยีของ field programmable gate arrays (FPGAs) เนื่องจาก FPGA มีความสามารถที่ยืดหยุ่นปรับเปลี่ยน

โครงสร้างของวงจร ราคาต่ำ และเหมาะกับการออกแบบให้ทำงานแบบขนานเพื่อช่วยเพิ่มประสิทธิภาพของระบบ

1.2 การตรวจเอกสาร

1. Accelerated Image Processing on FPGAs [21]

บทความนี้ได้แสดงถึงโครงงาน Cameron ได้พัฒนาภาษา และรวบรวม สำหรับการปะติดปะต่อรูปภาพลงบน FPGAs เปเปอร์นี้ทดสอบเทคโนโลยีบนหลายโปรแกรม และค้นพบว่า FPGAs เวลาระหว่าง 8 และ 800 จะเร็วกว่าเมื่อเปรียบเทียบกับ Pentiums สำหรับรูปภาพพื้นฐาน

2. Developing FPGA Coprocessors for Performance-Accelerated Spacecraft Image Processing [22]

บทความนี้ได้แสดงถึง FPAGs กับตัวประมวลผลสมองกลฝังตัวกำลังแสดงระดับของการทำงานและประสิทธิภาพ ซึ่งไม่สามารถเป็นไปได้ก่อนหน้าด้วยการใช้ตัวประมวลผลดั้งเดิม และแนะนำภาษา ImpulseC ว่าสามารถที่จะสร้างฮาร์ดแวร์ในรูปของภาษา VHDL หรือ Verilog โดยฮาร์ดแวร์สามารถที่จะถูกสังเคราะห์ด้วยเครื่องมือ FPGA เช่น Xilinx ISE

3. Combining Impulse C™ with uClinux™ for MicroBlaze™-based FPGAs [23]

บทความนี้ได้แสดงถึง uClinux สามารถเพิ่มพลังงานและความยืดหยุ่นเปลี่ยนแปลงของ FPGA ที่ฝังตัวประมวลผล ระบบปฏิบัติการบนสมองกลฝังตัว “soft ” สามารถจัดเตรียมทางเข้าสู่อุปกรณ์ฮาร์ดแวร์มาตรฐานภายใต้เงื่อนไขในรูปแบบพื้นฐาน FPGA (รวมถึงส่วนติดต่อเครือข่ายและอุปกรณ์อื่นๆ) ดีเท่าๆกับการจัดการความสามารถของงานหลายๆอย่าง โดยซอฟแวร์ที่รวมกันทำงานภายใต้หน่วยควบคุมของระบบปฏิบัติการ กับการออกแบบฮาร์ดแวร์ที่เหมาะสมเพื่อเร่งความเร็วอุปกรณ์ที่อาศัยอยู่ใน FPGAขณะที่ส่วนประกอบซอฟแวร์ noncritical ฝังอยู่ในตัวประมวลผล โปรแกรมนี้บรรยาย ImpulseC และเครื่องมือ CoDeveloper เพื่อแก้ไขปัญหา โดยปราศจากการเขียนอธิบายฮาร์ดแวร์ระดับต่ำ

4. Algorithms For Edge Detection [24]

บทความนี้ได้แสดงถึงการแนะนำหลักเบื้องต้นของการหาขอบภาพ โดยจำแนกวิธีการหาขอบภาพออกเป็น 2 กลุ่มหลัก คือ Gradient method และ Laplacian method และได้อธิบายรายละเอียดของเทคนิคการหาขอบในแต่ละวิธี ได้แก่ Sobel, Robert, Prewitt และ Canny’s edge detection algorithm

5. Hough Transform [25]

บทความนี้ได้แสดงถึง Hough transform คือ เทคนิคซึ่งสามารถแยกลักษณะพิเศษเฉพาะที่เจาะจงภายในรูปภาพ เพราะว่าสามารถจำแนกลักษณะที่ต้องการตามปัจจัยเฉพาะ Hough transform ส่วนมากนั้นจะตรวจสอบเส้นโค้งปกติ เช่น เส้นตรง, วงกลม, วงรี และอื่นๆ ผลประโยชน์หลักของ Hough transform คือความทนทานของช่องว่างในคำอธิบายเส้นเขต ความสามารถ และค่อนข้างไม่มีผลกระทบโดยสิ่งรบกวนรูปภาพ

6. Application of The Hough Transform [26]

บทความนี้ได้แสดงถึง Hough transform คือ วิธีการตรวจสอบเส้นโดย Paul Hough โดยแบ่งออกเป็น 2 วิธีหลัก คือ 1) The Classic Hough Transform Using the Accumulator Array และ 2) The Generalized Hough Transform ซึ่งจะเน้นรายละเอียดที่การหาเส้นตรงด้วยวิธีหลังเป็นส่วนใหญ่

7. Equation of the straight line [27]

บทความนี้ได้แสดงถึงสมการเส้นตรง (Equation of the straight line) สมการของเส้นตรง l คือ ความสัมพันธ์ระหว่าง x กับ y เมื่อ P(x,y) เป็นจุดใดๆบนเส้นตรง l

การเขียนเส้นตรง คือ การหาความสัมพันธ์ระหว่าง x กับ y ตามลักษณะหรือเงื่อนไขที่กำหนด ซึ่งอาจเขียนได้หลายแบบ ดังนี้ 1) สมการเส้นตรงที่ขนานกับแกนพิกัด 2) สมการเส้นตรงแบบจุด-ความชัน 3) สมการเส้นตรงแบบสองจุด 4) สมการเส้นตรงแบบความชัน- จุดตัดแกน 5) สมการทั่วไปของเส้นตรง

1.3 วัตถุประสงค์

1. เพื่อนำเสนอการออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์สำหรับการประมวลผลหาเส้นตรงในภาพ
2. เพื่อเพิ่มประสิทธิภาพการทำงานของการประมวลผลหาเส้นตรงด้วยการใช้คุณสมบัติการประมวลผลแบบขนานของเอฟพีจีเอ
3. เพื่อนำเสนอวิธีการสร้าง image processing algorithm ลงบน FPGA ด้วยภาษา ImpulseC
4. เพื่อนำเสนอสถาปัตยกรรมของวงจรที่ใช้การประมวลผลหาขอบภาพ

1.4 ขอบเขตของการวิจัย

1. สร้างต้นแบบในการออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์สำหรับการประมวลผลหาเส้นตรงโดยการใช้เทคโนโลยีเอฟพีจีเอและทดสอบบน Simulator
2. ทดสอบและวิเคราะห์ผลของโมดูล Edge Detection และ Hough Transform ที่ออกแบบด้วย Impulse C
3. ทดสอบระบบต้นแบบด้วยภาพเส้นกึ่งกลางถนน

1.5 ขั้นตอนและวิธีการวิจัย

1. ศึกษาการประมวลผล edge detection และ Hough transform
2. ศึกษาเทคนิคของการออกแบบวงจรให้มีประสิทธิภาพความเร็วสูง
3. ออกแบบการตรวจจับ (detection) สำหรับใช้ในระบบประมวลผลภาพการหาเส้นตรง
4. ทำการสร้างและพัฒนาวงจรสำหรับการตรวจจับ (detection) บนเทคโนโลยี FPGA
5. ทดสอบ และแก้ไขเพื่อให้สามารถทำงานได้ถูกต้องตรงตามฟังก์ชันที่ต้องการ
6. ทำการทดสอบเพื่อวิเคราะห์ และเก็บผล
7. จัดทำวิทยานิพนธ์

1.6 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถสร้างตัว co-processor สำหรับการประมวลผลภาพประสิทธิภาพสูงบนเทคโนโลยี FPGA
2. ได้วิธีการออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์สำหรับการประมวลผลหาเส้นตรงด้วยภาษา Impulse C
3. ได้รับความรู้หลักการทำงานของการประมวลผลสัญญาณภาพโดยใช้เทคนิค Edge detection และ Hough Transform
4. ได้เทคนิคการออกแบบวงจรให้มีความเร็วสูงในการประมวลผลหาขอบภาพ

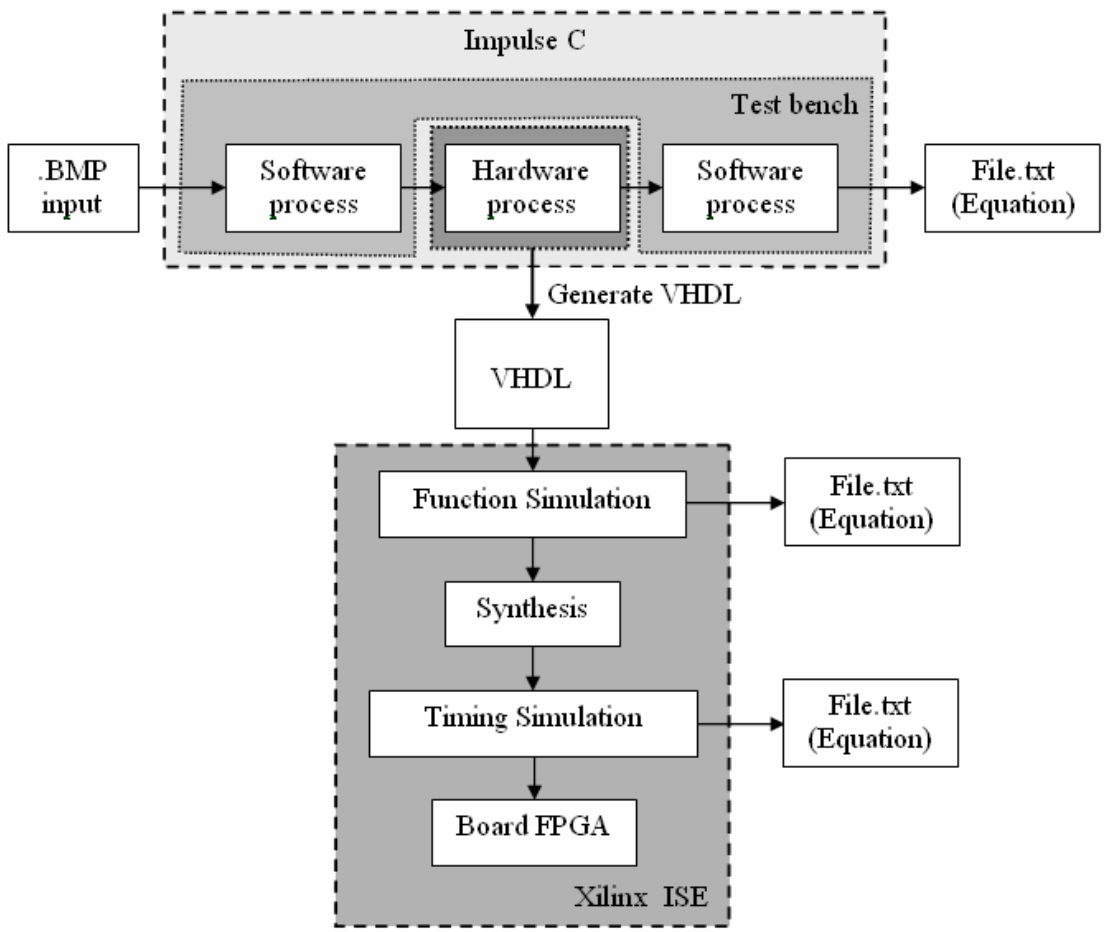
1.7 ทรัพยากรที่ใช้ในระบบ

ในการวิจัยนี้เป็นการใช้เทคโนโลยีเอฟพีจีเอเข้ามาช่วยในการประมวลผลร่วมกับหน่วยประมวลผลบนบอร์ดสมองกลฝังตัวเพื่อทำการหาสมการเส้นตรงจากภาพอินพุตที่ได้รับ ซึ่งมีซอฟต์แวร์และอุปกรณ์ที่ใช้ในการออกแบบและทดสอบดังนี้

1. คอมพิวเตอร์ส่วนบุคคล (Personal Computer) ที่ใช้ในการทำงานวิจัยนี้เป็นคอมพิวเตอร์ที่มีหน่วยประมวลกลาง AMD Athlon(TM)64 Processor 3000+ อัตราเร็ว 1.81 GHz และหน่วยความจำ RAM ขนาด 1.50 GB
2. โปรแกรม Impulse CoDeveloper Application Manager Universal Edition สำหรับใช้ในการออกแบบโปรแกรมของการประมวลผลสัญญาณภาพ และสร้างไฟล์ VHDL ขึ้นมาเพื่อนำไปใช้งานกับบอร์ด FPGA
3. โปรแกรม Xilinx ISE WebPACK เป็นโปรแกรมฟรี (Free software) ของบริษัท Xilinx ที่สามารถดาวน์โหลดได้ที่ http://www.xilinx.com/webpack/index.htm โปรแกรมนี้สามารถรองรับการออกแบบวงจรแบบ High Level Design ที่ใช้ภาษา ABEL หรือ HDL (VHDL, Verilog HDL) ในการออกแบบ และยังสามารถสังเคราะห์โมเดลวงจรจากภาษาดังกล่าวให้อยู่ในรูปของวงจรลอจิก รวมกระทั่งการโปรแกรมที่ช่วยในการโปรแกรมชิปได้ทั้งตระกูล CPLD และ FPGA และนอกเหนือจากนี้ภายในชุดโปรแกรม Xilinx ISE WebPACK ยังมีโปรแกรมช่วยเหลือในการออกแบบอีกมากมาย อาทิเช่น ModelSim Xilinx Edition (MXE) Starter Version เป็นโปรแกรมใช้จำลองการทำงานโมเดลที่ออกแบบจากภาษา VHDL, Verilog HDL ได้ทั้งระดับฟังก์ชัน (Functional) และไทม์มิ่ง (Timing)

1.8 ภาพรวมของระบบ

ภาพรวมของระบบที่ใช้ในการทำวิทยานิพนธ์นี้แสดงได้ดังภาพประกอบ ซึ่งสามารถอธิบายการทำงานได้ดังภาพประกอบ 1-1 ระบบจะรับภาพสกุล BMP เข้ามาทำการประมวลผลด้วยระบบการหาเส้นตรงภาพที่พัฒนาด้วยภาษา ImpulseC [4] โดยภาษานี้จะแบ่งออกเป็นสองส่วนหลัก คือ Software process และ Hardware process ซึ่งระบบการหาเส้นตรงของภาพนี้ประกอบไปด้วยฟังก์ชัน Edge Detection และ Hough Transform อยู่ในส่วนของ Hardware process และสามารถสร้างในรูปของภาษาทางฮาร์ดแวร์ ในที่นี้ใช้เป็นภาษา VHDL ออกมาได้ เพื่อนำไป simulation วิเคราะห์ค่าต่างๆบนซอฟต์แวร์ Xilinx ISEก่อนนำลงบอร์ด FPGA โดยผลลัพธ์ที่ได้ของระบบจะอยู่ในรูปของไฟล์ .txt ซึ่งมีรายละเอียดของสมการเส้นตรงของภาพ



ภาพประกอบ 1-1 ภาพรวมของระบบ

บทที่ 2

ทฤษฎีและหลักการ

2.1 การประมวลผลสัญญาณภาพดิจิตอล

การประมวลผลสัญญาณภาพดิจิตอล หรือ Digital Image Processing เป็นการนำภาพเข้าสู่การแปลงข้อมูลภาพให้อยู่ในรูปแบบข้อมูลดิจิตอลที่สามารถนำข้อมูลนี้ผ่านกระบวนการต่างๆ เพื่อให้ได้ผลลัพธ์แบบใหม่ที่บ่งบอกถึงลักษณะและคุณสมบัติของภาพ เช่น นำภาพสี RGB [5][6] แปลงเป็นภาพเฉดขาวดำ (Gray level), การหาขอบภาพ, การแยกชนิดสี, การดูช่วงค่าความกระจายของสี (Histogram) เป็นต้น กระบวนการต่างๆที่ยกตัวอย่างมานี้ เรียกว่า การกรอง (Filter)

2.2 รูปร่างของภาพ

วัตถุที่มีอยู่ตามธรรมชาติและที่มนุษย์สร้างขึ้นมีรูปร่างที่แตกต่างกันไป ทั้งที่เป็นรูปทรงเรขาคณิตและไม่เป็นรูปทรงเรขาคณิต ในศาสตร์ของการประมวลผลภาพนั้น การกำหนดขอบเขตของภาพทุกภาพให้อยู่ในรูปสี่เหลี่ยม (Rectangular Image Model) เป็นวิธีที่นิยมใช้กันมากที่สุด เนื่องจากทำให้การอ่านภาพ การจัดเก็บข้อมูลภาพในหน่วยความจำและการแสดงภาพออกทางอุปกรณ์ต่าง ๆ เป็นไปได้อย่างมีประสิทธิภาพ

การเก็บข้อมูลภาพลงหน่วยความจำของคอมพิวเตอร์สามารถทำได้โดยการจองหน่วยความจำของเครื่องไว้ในรูปของตัวแปรอะเรย์ (Array) โดยค่าในแต่ละช่องของอะเรย์แสดงถึงคุณสมบัติของจุดภาพ (Pixel) และตำแหน่งของช่องอะเรย์เป็นตัวกำหนดตำแหน่งของจุดภาพ

สมมุติให้ ภาพ เป็นตัวแปรแบบอะเรย์ขนาด M x N (M แถว และ N คอลัมน์) ที่ใช้เก็บภาพขนาด M x N จุด (M จุดในแนวนอน และ N จุดในแนวตั้ง) ค่าสี (หรือความสว่าง ในกรณีที่เป็นภาพ grey level) ของจุดภาพในแถวที่ 5 คอลัมน์ที่ 4 จะตรงกับค่าของ ภาพ(5,4) จะเห็นว่าเราใช้ตำแหน่งของจุดภาพทั้งสองแกนเป็นตัวชี้ค่าข้อมูลในอะเรย์

จากการใช้หน่วยความจำเพื่อการเก็บภาพในลักษณะที่กล่าวมา เนื้อที่ในการเก็บภาพสามารถคำนวณได้จาก M x N x g เมื่อ g เป็นจำนวนเต็มที่แทนจำนวนบิตของข้อมูลในแต่ละจุดภาพ ตัวอย่างถ้า g มีค่าเท่ากับ 8 บิต เราจะสามารถเก็บความแตกต่างของระดับสีที่เป็นไปสูงสุด

256 ระดับ ค่า M และ N จะเป็นตัวบอกถึงความละเอียดของภาพ สำหรับคอมพิวเตอร์ทั่วไปในระบบ VGA (Video Graphic Array) จะมีขนาด 640x480, 800x600 และ 1024x768 จุด เป็นต้น การกำหนดความละเอียดจะขึ้นอยู่กับงานที่จะใช้

ปกติแล้วในการเก็บข้อมูลภาพโดยเครื่องมือต่าง ๆ จะเก็บตามมาตรฐานของโทรทัศน์ ซึ่งมีอัตราส่วนแนวนอนต่อแนวตั้งเท่ากับ 4:3 สำหรับเครื่องมือเก็บข้อมูลภาพที่ไม่เป็นไปตามอัตราส่วน 4:3 เมื่อนำภาพนี้ไปแสดงในจอภาพมาตรฐาน จะทำให้ภาพที่แสดงนั้นมีขนาดของจุดภาพไม่เป็นสีเหลี่ยมจัตุรัส เช่นในบางระบบอาจจะใช้ความละเอียดในการแสดงเท่ากับ 640x520 ซึ่งจะทำให้ขนาดของจุดภาพที่ได้มีขนาดของด้านกว้างมีความยาวมากกว่าด้านสูง ซึ่งลักษณะดังกล่าวนี้เป็นหัวข้อที่ต้องสนใจสำหรับการเขียนโปรแกรมทางด้านกราฟฟิกและการจัดการข้อมูล

สำหรับการแสดงข้อมูลภาพที่มีขนาด 1 บิตและ 8 บิตนั้นจะมีการทำงานที่จะใกล้เคียงกันเนื่องจากหน่วยประมวลผลจะไม่สามารถจัดการกับข้อมูลที่เป็นบิตเดี่ยว ๆ ได้ดังนั้นในการแสดงข้อมูลออกทางจอภาพตัวโปรเซสเซอร์จะทำการคัดลอกข้อมูลทั้ง 8 บิต ส่งให้กับจอภาพซึ่งในกรณีที่ จุดภาพมีขนาด 1 บิต เมื่อโปรเซสเซอร์จะทำงานกับบิตแรกที่ต้องการแล้วก็จะทำการคัดลอกข้อมูลชุดใหม่ทันทีโดยที่ไม่เกี่ยวกับข้อมูลอีก 7 บิตที่เหลือส่วนในกรณี จุดภาพที่มีขนาด 8 บิต โปรเซสเซอร์จะทำการคัดลอกข้อมูลจุดใหม่ก็ต่อเมื่อโปรเซสเซอร์ทำงานกับทุกบิตแล้ว

ตัวอย่างสำหรับระบบที่มีความละเอียดเท่ากับ 640x480 และมีขนาด 16 บิตต่อจุดภาพจะสามารถแสดงสีได้ทั้งหมด 65536 ระดับและต้องใช้เนื้อที่ในการเก็บเท่ากับ 640x480x16 บิต

2.3 ทฤษฎีสีและเงา

สี คือ ลักษณะความเข้มของแสงที่ปรากฏแก่สายตาให้เห็นเป็นสี โดยผ่านกระบวนการรับรู้ด้วยตา โดยที่ตาได้ผ่านกระบวนการวิเคราะห์ข้อมูลพลังงานแสงมาแล้ว ผ่านประสาทสัมผัสการมองเห็น ผ่านศูนย์สับเปลี่ยนในสมองไปสู่ศูนย์การเห็นภาพ การสร้างภาพหรือการมองเห็นก็คือการที่ข้อมูลได้ผ่านการวิเคราะห์ แยกแยะให้เรารับรู้ถึงสรรพสิ่งรอบตัว โดยการมองเห็นภาพจากคอมพิวเตอร์นั้นเป็นการประกอบขึ้นของจุดภาพ (pixel) ที่มีค่าของสีอยู่ในจุดภาพนั้น จำนวนสีสูงสุดที่เป็นไปได้ของแต่ละจุดภาพขึ้นอยู่กับจำนวนบิตที่ใช้เมื่อมีการกำหนดให้ขนาดของบิตต่อจุดมากขึ้นจะทำให้จำนวนของสีมากขึ้นด้วย ตัวอย่างเช่น

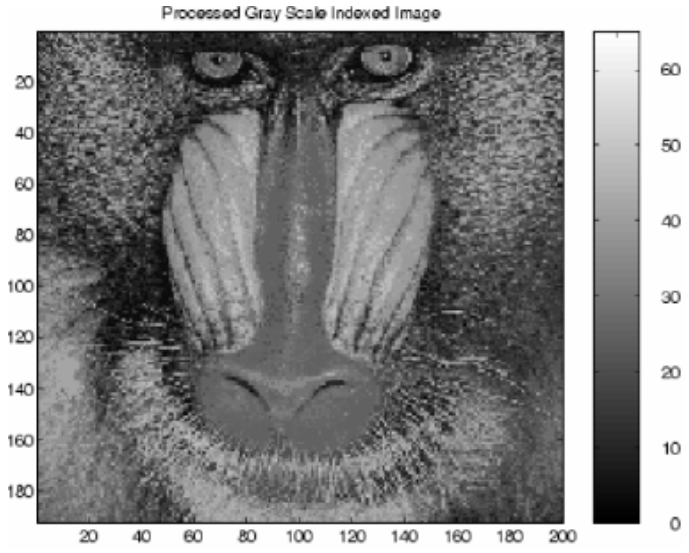
1 บิต	=	2^1	=	2	สี
2 บิต	=	2^2	=	4	สี
4 บิต	=	2^4	=	16	สี
8 บิต	=	2^8	=	256	สี
16 บิต	=	2^{16}	=	65536	สี

2.3.1 มาตรฐานของสี

มาตรฐานของสีที่ใช้อยู่ในปัจจุบันมีอยู่หลายระบบด้วยกัน ทั้งนี้จะขึ้นอยู่กับการนำไปใช้ แต่โดยทั่วไปแล้วทุกมาตรฐานจะมีแนวคิดเดียวกันคือ การแทนจุดสีด้วยจุดที่อยู่ภายในพิกัด 3 มิติ โดยจะมีแกนอ้างอิงสำหรับจุดสีนั้นในระนาบซึ่งแต่ละแกนจะมีความเป็นอิสระต่อกัน

2.3.1.1 ระบบสีแบบ Gray Scale

ภาพโทนสีขาวดำหรือภาพที่มีการเปลี่ยนแปลงตามความเข้มของแสง (Intensity Image) ค่าในแต่ละพิกเซลคือค่าความเข้มของแสง ณ แต่ละตำแหน่งของพิกเซล ซึ่งอยู่ในรูปแบบของระดับค่าขาวดำ (Gray Scale หรือ Gray Level) [7] ดังภาพประกอบ 2-1 ค่าที่เป็นไปได้ของระดับขาวดำ จะขึ้นอยู่กับจำนวนบิตที่ใช้ ตัวอย่างเช่น 8-Bit จะมีระดับของโทนสีขาวดำทั้งหมด 256 ระดับ



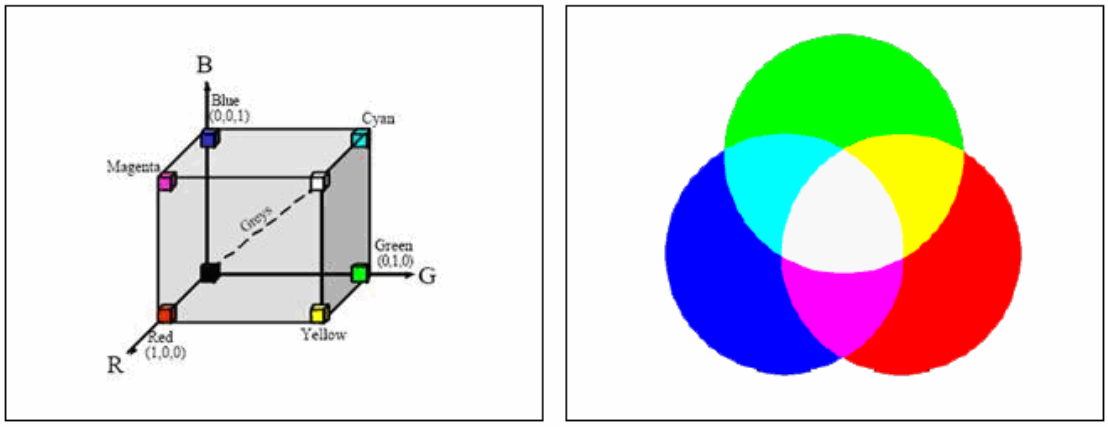
ภาพประกอบ 2-1 ตัวอย่างภาพโทนสีขาวดำแสดงค่าระดับความเข้มแสง

2.3.1.2 ระบบสีแบบ RGB

เป็นระบบสีที่เกิดจากการรวมกันของแสงสีแดง เขียวและน้ำเงิน รวมกันเป็นแสงสีขาว โดยมีการรวมกันแบบบวก ใช้ในการแสดงผลทางหน้าจอคอมพิวเตอร์ และหน้าจอโทรทัศน์

ในระบบพิกัด Color Space [8] ดังภาพประกอบ 2-2 (ก) โดยแต่ละสีจะมีค่าตั้งแต่ศูนย์จนถึงหนึ่ง โดยที่ศูนย์หมายถึง สีนั้นมีความเข้มมากจึงดูมืด และหนึ่งหมายถึง สีนั้นมีความเข้มน้อยจึงดูสว่าง จะได้ภาพการผสมสีทางแสงหรือการบวกแม่สี (Additive Primary Color) เข้าด้วยกัน ดังภาพประกอบ 2-2 (ข)

โดยทั่วไปจำนวนบิตข้อมูลที่ใช้ในการแทนความเข้มของแม่สีแต่ละสีมี 256 ระดับ (0-255) จำนวน 8 บิต รวมแม่สีทั้งสามแล้วใช้จำนวน 8x3 เท่ากับ 24 บิต ซึ่งสามารถใช้สร้างสีได้ถึง 256x256x256 เท่ากับ 16,777,216 สี



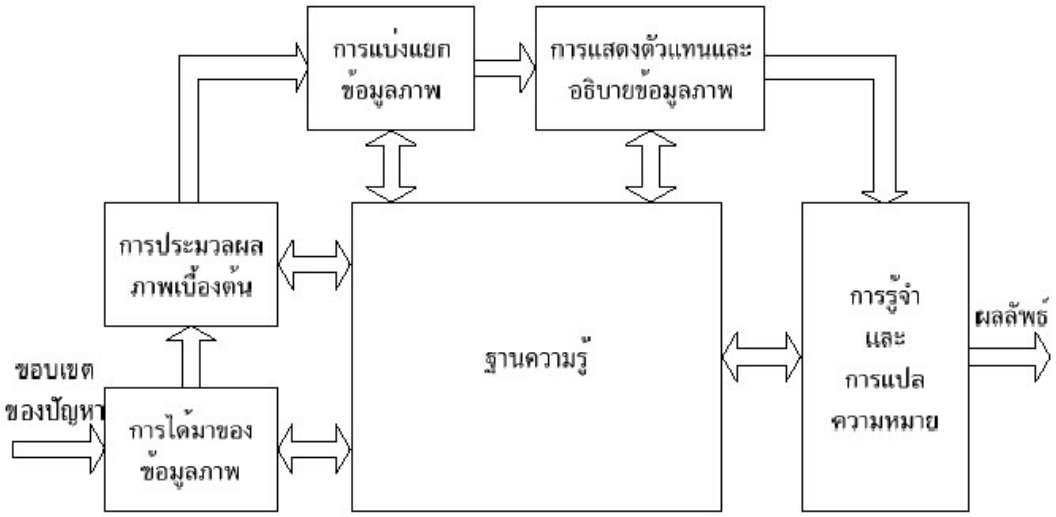
ภาพประกอบ 2-2 (ก) แสดงโมเดลในระบบพิกัด Color Space
(ข) แสดงการผสมสีทางแสง (Additive Primary Color) [9]

2.4 กระบวนการทางด้านการประมวลผลภาพด้วยคอมพิวเตอร์

การประมวลผลภาพด้วยคอมพิวเตอร์หรือที่นิยมเรียกกันว่า การประมวลผลภาพดิจิตอล (Digital Image Processing) เป็นกระบวนการที่มีเทคนิควิธีในการประมวลผลข้อมูลตัวเลขของภาพที่มีหลากหลายวิธี ซึ่งสามารถเลือกไปประยุกต์ใช้งานให้เหมาะสมกับข้อมูลภาพที่นำเข้ามาประมวลผล โดยปกติแล้วข้อมูลภาพจะมีลักษณะเด่นทางด้านรูปร่าง พื้นผิว สีสัน และโครงสร้างต่างๆ ที่แตกต่างกันไปขึ้นอยู่วัตถุและสภาพแวดล้อมโดยรอบของวัตถุ

2.4.1 ขั้นตอนพื้นฐานของการประมวลผลภาพดิจิตอล

การประมวลผลภาพดิจิตอลสามารถทำงานในรูปแบบของฮาร์ดแวร์หรือซอฟต์แวร์ได้ หลักทฤษฎีการประมวลผลภาพ มีการทำงานตามขั้นตอนอย่างเหมาะสมและเป็นไปตามเทคนิคทางด้านการประมวลผลภาพ ดังภาพประกอบ 2-3



ภาพประกอบ 2-3 ขั้นตอนพื้นฐานของการประมวลผลภาพดิจิตอล [10]

2.4.1.1 การได้มาของข้อมูลภาพ (Image Acquisition) เป็นการนำข้อมูลภาพเข้าสู่คอมพิวเตอร์ โดยอาศัยตัวรับรู้สัญญาณภาพและสามารถแปลงให้เป็นสัญญาณระบบดิจิตอลด้วยตัวรับรู้ เช่น กล้องถ่ายภาพดิจิตอล กล้องวิดีทัศน์ กล้องเว็บแคม เครื่องสแกน หรืออุปกรณ์รับสัญญาณภาพอื่นๆ ที่เหมาะสมกับระบบงานแต่ละระบบ อย่างไรก็ตามรูปแบบของข้อมูลจะถูกจัดเก็บให้อยู่ในลักษณะของภาพ 2 มิติ ที่มีความสว่างของแสงหรือความคมชัดแตกต่างกันของแต่ละจุดภาพในตำแหน่งต่างๆ

2.4.1.2 การประมวลผลภาพเบื้องต้น (Image Preprocessing) เป็นเทคนิควิธีของการปรับปรุงคุณภาพของข้อมูลภาพดิจิตอลที่ได้จากขั้นตอนการนำเข้าภาพ เพื่อให้ข้อมูลภาพมีความถูกต้องสมบูรณ์ตามความเป็นจริงก่อนนำไปประมวลผล โดยปกติแล้วการปรับปรุงคุณภาพของข้อมูลภาพดิจิตอลมีหลากหลายเทคนิค เช่น การปรับความคมชัด, การปรับความสว่าง, การกำจัดสัญญาณรบกวน, การหมุนและการกรองช่วงความถี่ของภาพ เป็นต้น

2.4.1.3 การแบ่งแยกข้อมูลภาพ (Image Segmentation) เป็นวิธีการแบ่งแยกข้อมูลภาพออกเป็นส่วนๆ เพื่อทำให้ได้ข้อมูลที่ต้องการออกจากพื้นหลัง โดยทั่วไปผลลัพธ์ของการ

แบ่งแยกข้อมูลภาพจะได้เป็นข้อมูลดิบของจุดภาพที่ประกอบด้วยขอบภาพของแต่ละบริเวณหรือจุดภาพภายในบริเวณนั้น ในแต่ละกรณีจะต้องทำการแปลงข้อมูลให้มีรูปแบบที่เหมาะสมสำหรับการประมวลผลที่บังคับไว้ จะทำให้การตัดสินใจข้อมูลมีการแสดงตัวแทนของขอบภาพหรือบริเวณนั้นๆ อย่างสมบูรณ์

2.4.1.4 การแสดงตัวแทนและอธิบายข้อมูล (Representation and Description) สำหรับการแสดงภาพหลังจากการแบ่งแยกข้อมูลภาพแล้ว เพื่อให้เห็นถึงลักษณะเด่นและอธิบายข้อมูลภาพของบริเวณต่างๆ ของภาพนำเข้า การเลือกตัวแทนสำหรับแสดงข้อมูลเป็นส่วนเดียวของการแก้ปัญหาสำหรับการแปลงข้อมูลดิบเป็นรูปแบบที่เหมาะสมสำหรับการประมวลผลของคอมพิวเตอร์ต่อไป วิธีการที่จะอธิบายลักษณะเด่นของข้อมูลที่สนใจถือเป็นสิ่งสำคัญ ซึ่งเรียกว่าการเลือกลักษณะเด่น (Feature Extraction) ผลลัพธ์ที่ได้จากการแยกลักษณะเด่นหรือความแตกต่างของข้อมูลที่สนใจออกจากข้อมูลอื่นๆ ก็คือ กลุ่มของวัตถุ (Class of Object) ที่ต้องการนั้นเอง

2.4.1.5 การรู้จำและการแปลความหมาย (Recognition and Interpretation) เป็นขั้นตอนสุดท้ายของการประมวลผลภาพดิจิตอลหลังจากขั้นตอนการแสดงตัวแทนและอธิบายข้อมูล ก็คือ การรู้จำภาพ (Image Recognition) ซึ่งเป็นแขนงหนึ่งของการรู้จำแบบรูป (Pattern Recognition) โดยการรู้จำภาพจะต้องรู้จำแบบรูปของแต่ละภาพเป้าหมายเพื่อการให้คำตอบว่าแบบรูปของภาพนำเข้ามีความคล้ายกับแบบรูปของภาพอ้างอิงภาพใดมากที่สุด และการแปลความหมายนำไปสู่การกำหนดความหมายของชุดข้อมูลรู้จำวัตถุ การได้มาของแบบรูปอ้างอิงนั้นสามารถทำได้หลายวิธี เช่น รูปแบบอ้างอิงอาจอยู่ในรูปแบบจำลองทางคณิตศาสตร์ ซึ่งจะต้องมีวิธีเฉพาะในการเปรียบเทียบ การสร้างแบบจำลองทางคณิตศาสตร์สามารถทำได้จากขั้นตอนการฝึกฝน (Training Phrase) ซึ่งโดยทั่วไปแล้วจะต้องมีตัวอย่างภาพที่มีลักษณะเดียวกันหลายๆภาพ จากนั้นจะทำการคำนวณหาค่าลักษณะเด่นของแต่ละภาพ ซึ่งผลลัพธ์ที่ได้ก็คือ แบบรูปของภาพเหล่านั้นนั่นเอง แบบจำลองของภาพในแต่ละกลุ่มสามารถคำนวณได้จากค่าสถิติต่างๆ ของแบบรูปของภาพในกลุ่มเดียวกัน บางครั้งอาจจะอยู่ในรูปของฐานความรู้ (Knowledge Base) จำนวนมากจนมีกลุ่มข้อมูลที่เก็บไว้เป็นฐานความรู้ในรูปแบบของฐานข้อมูลความรู้ (Knowledge Database)

2.4.1.6 ฐานความรู้ (Knowledge base) ระบบฐานความรู้(Knowledge Base) คือ การสร้างระบบคอมพิวเตอร์ที่สามารถรับรู้และมีความเชี่ยวชาญในด้านใดด้านหนึ่ง โดยนำความรู้จากบุคคลผู้เชี่ยวชาญในสาขาความรู้นั้นมาสร้างเป็นระบบฐานความรู้ (Knowledge base) ขึ้น นอกจากนี้ยังมีโปรแกรมที่ควบคุมการค้นหาความรู้ที่ต้องการ และโปรแกรมที่ตรวจสอบกฎเกณฑ์และทฤษฎีเพื่อให้ได้คำตอบของปัญหา เพื่อให้คอมพิวเตอร์สามารถแก้ปัญหาตามกฎเกณฑ์และเงื่อนไขที่ได้รับได้

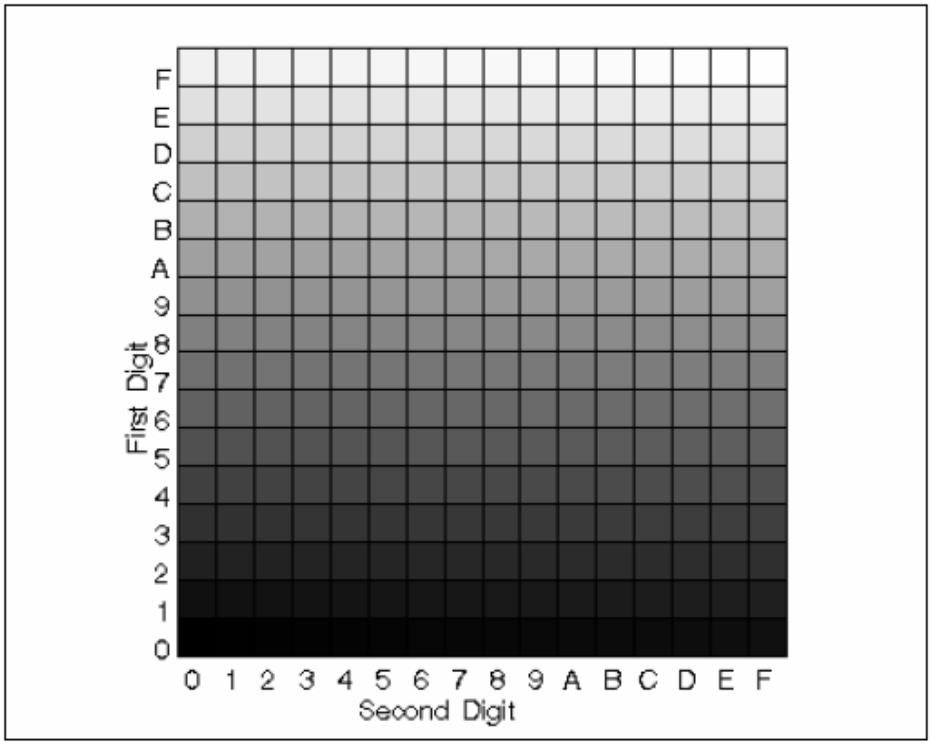
2.5 การแปลงภาพสีเป็นภาพระดับสีเทา (Gray-Scale Image Transform)

เป็นการแปลงค่าข้อมูลภาพสีให้แสดงถึงค่าความสว่างของภาพเพียงอย่างเดียว โดยปราศจากค่าข้อมูลของสีภาพ โดยทั่วไปภาพระดับสีเทาจะประกอบด้วยค่าความสว่างที่แตกต่างกัน 256 ระดับ มีค่าตั้งแต่ 0 ถึง 255 นั้นคือไล่ระดับความสว่างจากมืดไปจนขาว

การแปลงภาพสีเป็นขาวดำได้นั้นทำได้หลายวิธี เช่น การใช้สมการ 2-1 แปลงค่า RGB ให้เป็นค่าเฉลี่ยแล้วแทนลงไปในพิกเซลนั้นๆ ซึ่งจะได้ความลึกของภาพเท่ากับ 24 บิตเหมือนเดิม หรืออีกวิธีโดยการเปลี่ยนจากภาพสี RGB เป็นภาพ Gray Scale ซึ่งความลึกของภาพจะเหลือ 8 บิต โดยจะมีการคูณด้วยค่าคงที่ไปที่แต่ละสีของ RGB ซึ่งค่าคงที่นั้นโดยความจริงแล้วอาจจะไม่ใช่ตัวเลขที่ตายตัวเสมอไป ที่นิยมจะใช้สมการ 2-1

$$Y = 0.299R + 0.587G + 0.114B \qquad (2\text{-}1)$$

โดยที่ Y คือค่าของความเข้มมีค่าอยู่ในช่วงตั้งแต่ 0-255 (สีดำ-สีขาว) ได้ดังภาพประกอบ 2-4



ภาพประกอบ 2-4 เกรย์สเกล (Gray scale) [11]

2.6 วิธีการหาขอบภาพ (Edge Detection Methods)

การหาขอบภาพ [12][13] คือการตรวจสอบว่าเส้นขอบลากผ่านหรือใกล้เคียงกับจุดใด โดยวัดจากการเปลี่ยนแปลงของความเข้มในตำแหน่งที่ใกล้เคียงกับจุดดังกล่าว ซึ่งวิธีการหาขอบนั้นแบ่งได้เป็น 2 กลุ่มหลักๆ คือ Gradient method และ Laplacian method โดยในแต่ละวิธีมีรายละเอียดดังต่อไปนี้

- **Gradient method**

วิธีการค้นหาขอบภาพโดยใช้อนุพันธ์อันดับหนึ่ง (Gradient Operator: ∇) มีหลักการคือบริเวณขอบของวัตถุในภาพจะมีค่าเกรเดียนต์ที่สูง การพิจารณาขนาดของเกรเดียนต์(Gradient Magnitude, $|P \nabla|$) เปรียบเทียบกับค่าอ้างอิง (Threshold, T) ที่กำหนดขึ้น เมื่อค่าของเกรเดียนต์มีค่ามากกว่าค่าอ้างอิง แสดงว่าจุดดังกล่าวคือขอบของวัตถุที่ปรากฏในภาพที่จุด $P(x,y)$ การค้นหาขอบของวัตถุโดยใช้ อนุพันธ์อันดับหนึ่ง เป็นวิธีแยกส่วนประกอบของภาพและเมื่อความไม่ต่อเนื่องของค่าพิกเซลบริเวณรอยต่อระหว่างวัตถุกับพื้นหลังและค่าอนุพันธ์ย่อยที่ไม่ต่อเนื่องตามทิศทางของเกรเดียนต์ของแนวแกน x และแกน y กำหนดค่าได้ตามสมการ 2-2 และ 2-3

$$\nabla_x P(x,y) = P(x,y) - P(x-1,y) \tag{2-2}$$

และ
$$\nabla_y P(x,y) = P(x,y) - P(x,y-1) \tag{2-3}$$

เมื่อ $P(x,y)$ คือ ฟังก์ชันของภาพ

โดยที่ขนาดของเกรเดียนต์ของ $P(x,y)$ กำหนดค่าได้จากสมการ 2-4

$$\left|\nabla P(x,y)\right| = \sqrt{(\nabla_x P(x,y))^2 + (\nabla_y P(x,y))^2} \tag{2-4}$$

เพื่อให้ง่ายต่อการคำนวณ ประมาณค่าขนาดของเกรเดียนต์ ได้จากสมการ 2-5

$$\left|\nabla P(x,y)\right| = \left|\nabla_x P(x,y)\right| + \left|\nabla_y P(x,y)\right| \tag{2-5}$$

การค้นหาขอบภาพที่มีองค์ประกอบของเส้นตรงเป็นวิธีการแบ่งออกเป็นส่วนย่อยในแต่ละพิกเซล กำหนดรูปแบบการเปลี่ยนแปลงภาพให้มีความเรียบด้วยตัวกรองเกาเซียนก่อนคำนวณหาขนาดและทิศทางของเกรเดียนต์และค่าของ Mask กำหนดค่าได้ดังสมการ 2-6 และ 2-7 [3]

$$\text{Mask}(E_x) = \begin{bmatrix} Z_{x1} & Z_{x2} & Z_{x3} \\ Z_{x4} & Z_{x5} & Z_{x6} \\ Z_{x7} & Z_{x8} & Z_{x9} \end{bmatrix} = E_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (2\text{-}6)$$

$$\text{Mask}(E_y) = \begin{bmatrix} Z_{y1} & Z_{y2} & Z_{y3} \\ Z_{y4} & Z_{y5} & Z_{y6} \\ Z_{y7} & Z_{y8} & Z_{y9} \end{bmatrix} = E_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (2\text{-}7)$$

ให้ Z คือ ตำแหน่งของแต่ละพิกเซลทั้ง 9 จุด

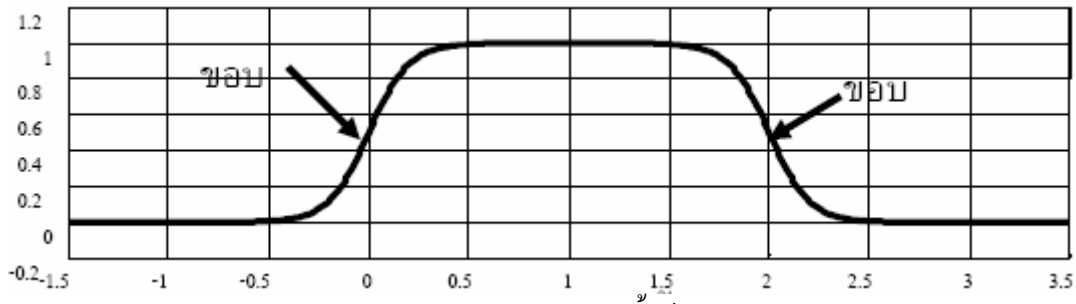
กำหนดให้ขนาดของ Mask เท่ากับ 3x3 และมีค่าเท่ากับ Ex และ Ey หาค่าอนุพันธ์อันดับหนึ่งของส่วน $\frac{\partial P}{\partial x}$ และของส่วน $\frac{\partial P}{\partial y}$ ขนาดและทิศทางของเกรเดียนต์หาได้จากสมการ 2-8 และ 2-9

$$M(i,j) = \sqrt{E_x^2(i,j) + E_y^2(i,j)} \quad ; \quad M(i,j) \text{ คือ ขนาดของ } \nabla \quad (2\text{-}8)$$

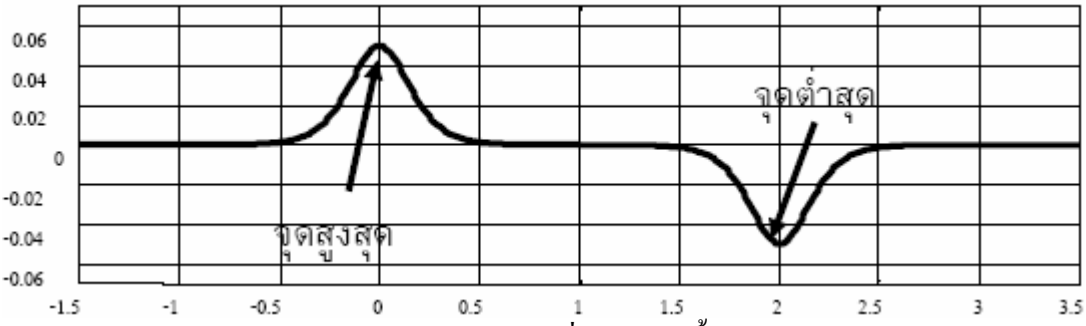
$$\theta(i,j) = \tan^{-1}(E_x^2(i,j) + E_y^2(i,j)) \quad ; \quad \theta(i,j) \text{ คือ ทิศทางของ } \nabla \quad (2\text{-}9)$$

การกำหนดค่า Mask ของแนวแกน x และแกน y ทำให้ขนาดและทิศทางของเกรเดียนต์เปลี่ยนแปลงและลดขนาดของขอบในแต่ละพิกเซลของขอบภาพได้ พิกเซลรอบข้างของทิศทางของเกรเดียนต์ใดที่ไม่ใช่ค่าสูงสุดไม่ถือว่าเป็นขอบ ดังนั้นจะมีเพียงพิกเซลหนึ่งเดียวที่เป็นขอบ

ตัวอย่างวิธีการหาขอบโดยใช้อนุพันธ์อันดับหนึ่ง ได้แก่ Roberts, Prewitt, Sobel และ Canny เป็นต้น



(a) สัญญาณของภาพตั้งต้น



(b) อนุพันธ์อันดับหนึ่งของภาพตั้งต้น

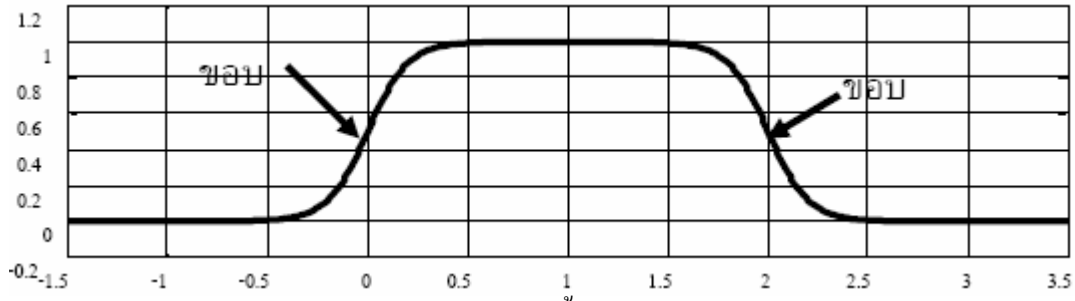
ภาพประกอบ 2-5 การหาขอบภาพโดยใช้อนุพันธ์อันดับหนึ่ง

- **Laplacian method**

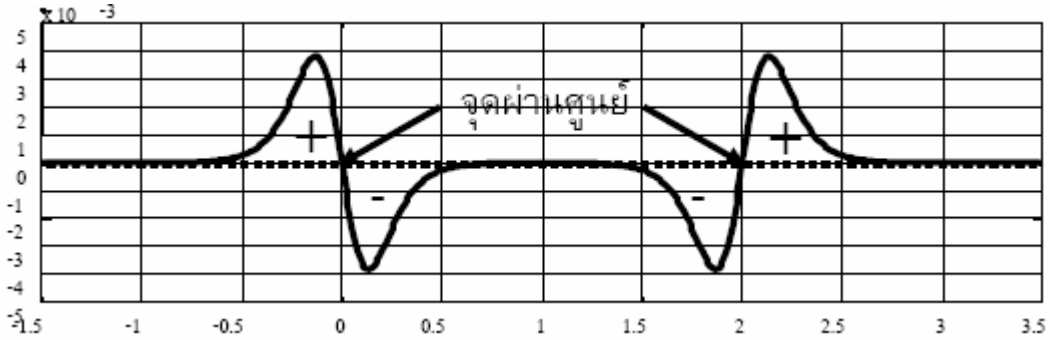
การค้นหาโดยใช้อนุพันธ์อันดับสอง เปรียบได้กับการหาอนุพันธ์อันดับสองของภาพ เพื่อให้ได้ขอบของภาพ โดยภาพที่ผ่านการหาอนุพันธ์อันดับสอง (Laplacian Operator ($\nabla^2 P$)) บริเวณที่เป็นส่วนขอบจะเด่นชัดขึ้น การประมาณค่าของการหาอนุพันธ์อันดับสองโดยใช้ Mask ของสมการ 2-6 และ 2-7 ทำให้ตำแหน่งของบริเวณขอบของวัตถุในภาพคือค่าจุดผ่านศูนย์ (Zero Crossing) ของการหาอนุพันธ์อันดับสอง สามารถหาได้จากสมการ 2-10

$$\nabla^2 P = \frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2} \tag{2-10}$$

การค้นหาขอบภาพด้วยอนุพันธ์อันดับสอง เมื่อทำ $\nabla^2 P$ บริเวณขอบจะมีตำแหน่งเดียวกับค่าจุดผ่านศูนย์ของค่าจาก $\nabla^2 P$ พิจารณาได้จากตำแหน่งที่พิกเซลเปลี่ยนแปลงจากค่าที่เป็นบวกเป็นค่าที่เป็นลบหรือจากค่าที่เป็นลบไปเป็นค่าที่เป็นบวก ดังภาพประกอบ 2-6



(a) ภาพตั้งต้น



(b) อนุพันธ์อันดับสองของภาพตั้งต้น

ภาพประกอบ 2-6 การหาขอบภาพโดยใช้อนุพันธ์อันดับสอง

การหาขอบโดยใช้อนุพันธ์อันดับสอง ไม่สนใจทิศทางของภาพในแนวแกน x และแกน y กำหนดจุดที่ค่า y เป็นจุดผ่านศูนย์ วิธีนี้ใช้เวลาในการคำนวณมากกว่าการค้นหาขอบโดยใช้อนุพันธ์อันดับหนึ่ง กล่าวคือเราสามารถตรวจจับความไม่ต่อเนื่องของพิกเซลในโดเมนรูปภาพได้โดยใช้อนุพันธ์ของภาพนั่นเอง

เนื่องจากการจัดเก็บรูปภาพเป็นการจัดเก็บในรูปแบบพิกเซล ฉะนั้นการที่จะแยกวัตถุ 2 วัตถุออกจากกันหรือแยกวัตถุออกจากพื้นหลังจึงต้องอาศัยขอบ (Edge) ของวัตถุ การหาขอบของวัตถุสามารถทำได้โดย การดูความแตกต่างของสีของพิกเซลใกล้เคียง ถ้าสีแตกต่างกัน แสดงว่ามีขอบอยู่ระหว่างพิกเซลนั้น

2.6.1 Roberts Edge Detection

การหาขอบด้วยวิธีนี้ก็เป็นอีกตัวอย่างหนึ่งของการใช้เทคนิคการปรับปรุงขอบที่ไม่ต่อเนื่อง ให้ $a \in R^x$ เป็นภาพต้นฉบับ และขอบที่ได้เป็น $b \in R^x$ วิธีการของ Roberts ก็คือ

$$b(i,j) = \sqrt{((a(i,j) - a(i+1,j+1))^2 + (a(i,j+1) - a(i+1,j))^2} \qquad (2\text{-}11)$$

ให้ S คือ mask ของแนวแกน x

T คือ mask ของแนวแกน y

$$S = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \qquad T = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

ภาพประกอบ 2-7 เทมเพลตของ S และ T

เมื่อทำการคำนวณหา edge ของทุก pixel ของทั้งภาพแล้ว จะได้ภาพของ edge ซึ่งสีขาว (High Intensity) จะเป็นขอบภาพ

ภาพประกอบ 2-8 ภาพที่ได้จากการหาขอบด้วยวิธีของ Roberts [16]

2.6.2 Prewitt Edge Detection

วิธีการนี้จะคำนวณ ขอบที่เป็นเกรเดียนต์เวคเตอร์ของทุกจุดบนภาพที่เป็นภาพต้นฉบับ ขอบที่ผ่านการปรับปรุงแล้วนั้นมาจากขนาดของเกรเดียนต์เวคเตอร์ มาสค์ที่ใช้แทนอนุพันธ์จะเกี่ยวข้องกับ x และ y ให้ $a \in R^x$ เป็นภาพต้นฉบับและ a1,a2,…,a7 เป็นค่าของแต่ละพิกเซล 8 จุดที่ตำแหน่ง (i,j) ตามทิศทางทวนเข็มนาฬิกา ดังภาพประกอบ 2-9

a_3	a_2	a_1
a_4	**I(i,j)**	a_0
a_5	a_6	a_7

ภาพประกอบ 2-9 แสดงตำแหน่งของตัวแปรด้วยวิธี Prewitt

ให้ $u = (a_5 + a_6 + a_7) - (a_1 + a_2 + a_3)$ และ $v = (a_0 + a_1 + a_7) - (a_3 + a_4 + a_5)$

ขอบของภาพเป็น $b \in R^x$ ให้ $b(i,j) = \sqrt{u^2 + v^2}$ (2-12)

และให้ทิศทางของขอบภาพ $d \in R^x$ คือ $d(i,j) = \arctan(\frac{v}{u})$ (2-13)

ให้ S คือ mask ของแนวแกน x

T คือ mask ของแนวแกน y

$$S = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad T = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}$$

ภาพประกอบ 2-10 เทมเพลตของ Prewitt

ภาพประกอบ 2-11 ภาพที่ได้จากการหาของด้วยวิธีของ Prewitt [16]

2.6.3 Sobel Edge Detection

วิธีนี้เป็นการหาขอบที่ไม่เป็นเชิงเส้น สามารถเปลี่ยนแปลงค่าความไม่ต่อเนื่องได้ตามการปรับปรุงขอบให้ $a \in R^x$ เป็นภาพต้นฉบับ และ a0, a1, a2, …, a7 แสดงถึงตำแหน่งของแต่ละพิกเซลทั้ง 8 จุด ทวนเข็มนาฬิกา ดังภาพประกอบ 2-9

ให้ $u = (a_5 + 2a_6 + a_7) - (a_1 + 2a_2 + a_3)$ และ $v = (2a_0 + a_1 + a_7) - (a_3 + 2a_4 + a_5)$

ขนาดขอบของภาพ Sobel เป็น $m \in R^x$ ให้ $m(i,j) = \sqrt{u^2 + v^2}$ (2-14)

และให้ทิศทางของเกรเดียนต์ของภาพ d คือ $d(i,j) = \arctan(\frac{u}{v})$ (2-15)

ให้ S คือ mask ของแนวแกน x

T คือ mask ของแนวแกน y

$$S = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad T = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

ภาพประกอบ 2-12 เทมเพลตของ Sobel

ภาพประกอบ 2-13 ภาพที่ได้จากการหาของด้วยวิธีของ Sobel [16]

2.6.4 Canny Edge Detection

ขั้นตอนการค้นหาขอบภาพโดยวิธีของ Canny ประกอบด้วย 4 ขั้นตอน เริ่มต้นจากการปรับภาพให้เรียบ (Smoothing) ด้วยตัวกรองเกาเซียน เพื่อกำจัดสัญญาณรบกวน หลังจากนั้นหาอนุพันธ์อันดับหนึ่ง คำนวณค่าขนาดและทิศทางของเกรเดียนต์ นำค่าที่ได้มาคำนวณค่าของ Non-maxima Suppression กับค่าขนาดของเกรเดียนต์เพื่อทำให้ได้ขอบที่บางลงและในขั้นตอนสุดท้ายใช้การกำหนดจุดอ้างอิงสองระดับ (Double Thresholding) เพื่อระบุค่าของพิกเซลที่เป็นขอบและช่วยเชื่อมต่อขอบ [17] โดยในแต่ละขั้นตอนมีรายละเอียดดังต่อไปนี้

2.6.4.1 การปรับภาพให้เรียบ (Smoothing)

ขั้นตอนแรกการค้นหาขอบโดยอัลกอริทึมของ Canny ต้องกำจัดสัญญาณรบกวน (Noise) ออกก่อน ด้วยวิธีการใช้ตัวกรองเกาเซียน กำหนดกรอบ (Masks) เป็นเมตริกซ์ขนาด 3x3 หรือมีขนาดเท่ากับ 9 พิกเซล การกำหนดขนาดของตัวกรองเกาเซียน หากมีขนาดกว้างมาก จะมีผลทำให้ลดสัญญาณรบกวนได้มาก ถ้าขนาดกรอบกว้างมากเกินไปมีผลทำให้ขอบย่อยๆที่เป็นส่วน

รายละเอียดหายไป ผลของภาพที่ผ่านการปรับภาพให้เรียบด้วยตัวกรองเกาเซียนหาได้จากสมการ 2-16

$$S_{(i,j)} = G_{(i,j,\sigma)} \bullet I_{(i,j)} \quad (2\text{-}16)$$

กำหนดให้

$I_{(i,j)}$	คือ ภาพที่ต้องการหาขอบ
$G_{(i,j,\sigma)}$	คือ Gaussian Smoothing Filter
σ	คือ ควบคุมระดับของการ Smoothing
$\bullet$	คือ โอเปอเรชันการคูณ

2.6.4.2 การคำนวณค่าของเกรเดียนต์ (Gradient Calculation)

ขั้นแรกปรับภาพ $I_{(i,j)}$ ให้มีความเรียบ ผลลัพธ์ที่ได้คือค่าของภาพในฟังก์ชัน $S_{(i,j)}$ ขั้นตอนที่สอง การหาค่าของเกรเดียนต์ในทิศทางของแกน x และแกน y และกำหนดขนาดของอนุพันธ์อันดับหนึ่งของ $Px_{(i,j)}$ และ $Qy_{(i,j)}$ ตามลำดับดังสมการ 2-17 และ 2-18

$$Px_{(i,j)} \approx (S_{(i,j+1)} - S_{(i,j)} + S_{(i+1,j+1)} - S_{(i+1,j)}) / 2 \quad (2\text{-}17)$$

$$Qy_{(i,j)} \approx (S_{(i,j)} - S_{(i+1,j)} + S_{(i,j+1)} - S_{(i+1,j+1)}) / 2 \quad (2\text{-}18)$$

นำค่า $Px_{(i,j)}$ และ $Qy_{(i,j)}$ ที่ผ่านการหาอนุพันธ์อันดับหนึ่งเมื่อคำนวณการแปลงรูปแบบจากระนาบของระบบพิกัดฉาก (Rectangular Form) ไปเป็นระนาบพิกัดเชิงขั้ว (Polar Form) เพื่อหาขนาดและทิศทางของเกรเดียนต์แทนค่าตามสมการ 2-17 และ 2-18 ได้ค่าขนาดเกรเดียนต์ดังนี้คือ $M(i,j) = \sqrt{P_x^2(i,j) + Q_y^2(i,j)}$ และทิศทางของเกรเดียนต์ (Gradient Orientation) เท่ากับ $\theta(i,j) = \tan^{-1}(Q_y(i,j)/P_x(i,j))$ และสามารถหาค่ามุม θ ออกมาได้เมื่อแทนค่าตัวแปรในฟังก์ชั่น $\theta = \tan^{-1}(x,y)$

2.6.4.3 Non-maxima Suppression

การค้นหาขอบภาพด้วยวิธีการของ Canny จุดที่ถือว่าเป็นเส้นขอบของภาพได้นั้นต้องเป็นจุดที่ให้ค่าสูงสุดเฉพาะที่และเป็นทิศทางเดียวกับเกรเดียนต์ การค้นหาขอบภาพโดยใช้อนุพันธ์อันดับหนึ่งทำให้ได้ขอบที่บางเพียง 1 พิกเซล ภาพที่ได้หลังการทำ Non-maxima Suppression จะให้ค่าเป็นศูนย์ในทุกจุดยกเว้นจุดที่เป็น local Maxima Point ซึ่งจะยังคงค่าเดิมไว้

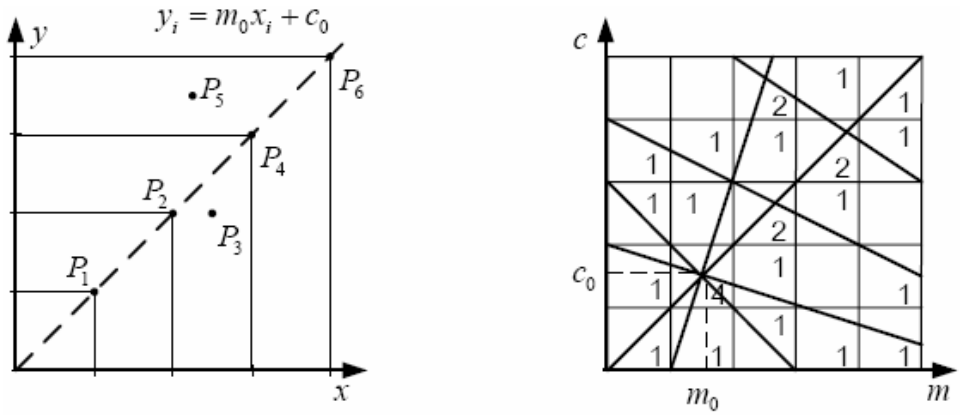
2.6.4.4 Thresholding

แม้ว่าภาพจะผ่านการ Smoothing ในขั้นตอนแรกแล้วก็ตาม ภาพที่ได้อาจยังมีเส้นขอบที่ไม่ใช่ขอบที่แท้จริงปรากฏอยู่ อันเนื่องจากสัญญาณรบกวนหรือลักษณะของวัตถุในภาพเป็นพื้นผิวที่มีลวดลายหรือมีรายละเอียดภายในมาก ดังนั้นเพื่อลดปัญหาดังกล่าวจึงได้มีการกำหนดค่า Threshold ขึ้นมา 2 ค่าคือ High Threshold (T_1) และ Low Threshold (T_2) โดยพิกเซลที่มีค่ามากกว่า T_1 จะถูกปรับเป็น '1' เป็นพิกเซลที่เป็นขอบ แต่ถ้าน้อยกว่า T_2 จะถูกปรับเป็น '0' ส่วนค่าที่อยู่ระหว่างค่า Threshold ทั้งสอง การปรับเป็นค่า '0' หรือ '1' นั้นขึ้นอยู่กับพิกเซลที่อยู่รอบข้าง หากพบว่าพิกเซลที่อยู่รอบข้างของพิกเซลที่เป็นขอบ (ขอบค่า > T_1) มีค่ามากกว่า T_2 แล้ว จะปรับค่าพิกเซลดังกล่าวให้มีค่าเป็น '1' และถือเป็นสมาชิกหนึ่งในภาพขอบด้วยเช่นกัน ดังนั้นการทำ Threshold จะทำให้ภาพที่มีขอบหนาหรือบางนั่นเอง

ภาพประกอบ 2-14 ภาพที่ได้จากการหาของด้วยวิธีของ Canny

2.7 Hough Transform

วิธีการของ Hough Transform คือการค้นหาเส้นตรงและวงกลมจากจุดต่างๆ โดยแต่ละจุดจะโหวตว่าจุดนั้นๆอยู่บนเส้นใดบ้าง เมื่อทุกจุดโหวตแล้ว สมการที่ถูกโหวตมากที่สุดจะเป็นเส้นที่ผ่านจุดมากที่สุด เช่น ต้องการหาสมการเส้นตรงที่ผ่านจุด (x,y) จะมีเส้นตรงมากมายเป็นอนันต์ผ่านจุดดังกล่าว เมื่อพิจารณาเส้นตรง $y_0 = mx_0 + c$ เส้นตรงที่ผ่านจุด (x,y) มีค่าพารามิเตอร์คงที่ (m,c) ซึ่งค่า $c_0 = y - m_0 x$ ดังนั้นจุดหนึ่งๆจะโหวตให้กับสมการเส้นตรงที่มี Parameter ต่างๆกันได้หลายสมการ เมื่อจุดทุกจุดได้โหวตเสร็จเรียบร้อยแล้ว สมการเส้นตรงที่ถูกโหวตมากที่สุด จะเป็นเส้นตรงที่ผ่านจุดที่กำหนดให้มากที่สุด [18]

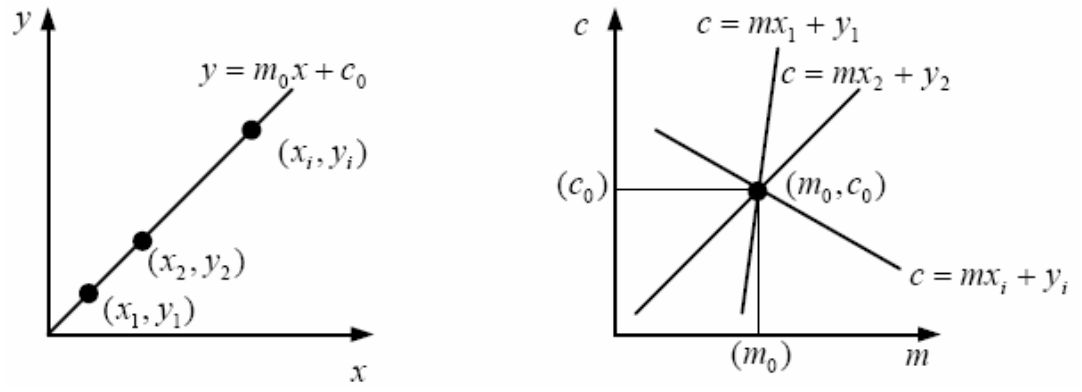


(a) จุดของเส้นตรงใน Image Space (b) จำนวนเส้นตรงใน Parameter Space

ภาพประกอบ 2-15 การนับจำนวนเส้นตรงของการเปลี่ยนแปลง Hough Transform

2.7.1 การแปลงรูปแบบจาก Image Space ไปสู่ Parameter Space

จากสมการของเส้นตรง $y_i = mx_i + c$ เมื่อจุด (x_i, y_i) ดังแสดงในภาพประกอบ 2-15 (a) ค่าความชันและจุดแกน y ของสมการเป็นค่าคงที่ (m_0, c_0) หรือเรียกว่าส่วนของ Parameter Space ดังนั้นการเปลี่ยนจาก Image Space ไปสู่ Parameter Space สมการของการเปลี่ยนแปลงคือที่จุด (m_0, c_0) จะเท่ากับ $c_i = y - m_i x$ ดังแสดงในภาพประกอบ 2-15 (b)

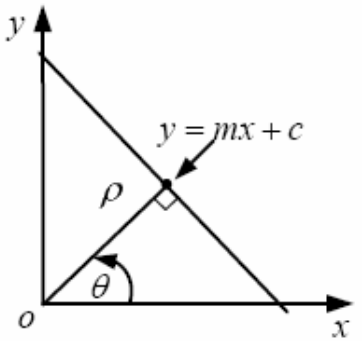


(a) Image Space (b) Parameter Space

ภาพประกอบ 2-16 การแปลงรูปแบบระหว่าง Image Space กับ Parameter Space

จากภาพประกอบ 2-16 (a) มีจำนวนจุดทั้งสามจุดที่เส้นตรงที่มีค่าของความชันและจุดตัดแกน y ที่ตำแหน่ง (m_0, c_0) ดังนั้นเมื่อพิจารณาในรูปแบบของ Parameter Space ที่จุด (m_0, c_0) ก็จะมีเส้นตรงที่เกิดขึ้นจากสมการเส้นตรงได้ทั้งหมดสามสมการที่ลากผ่านจุดดังกล่าว

ดังนั้นเมื่อกำหนดจุด (x,y) บนระนาบของ Image Space และทำการเปลี่ยนเป็นระนาบ Parameter Space หรือเรียกว่า Hough Space ดังแสดงในภาพประกอบ 2-17



ภาพประกอบ 2-17 ความสัมพันธ์ระหว่างค่าของเวกเตอร์ ρ กับจุดของเส้นตรง

ความสัมพันธ์ระหว่างการเปลี่ยนรูปแบบคือที่จุด (ρ,θ) บน Parameter Spaces จะเป็นจุดที่อยู่บนสมการเส้นตรงที่ลากผ่านจุด (x,y) เมื่อพิจารณาค่าของเวกเตอร์ ρ ที่ตั้งฉากกับจุดดังกล่าวและทำมุมกับแกน x เท่ากับมุม θ ดังนั้นการแปลงรูปแบบจาก Image Space ไปสู่ Parameter Space สามารถหาค่าของเวกเตอร์ ρ ได้จากสูตร

$$\rho = x\cos\theta + y\sin\theta \tag{2-19}$$

เมื่อกำหนดให้ ρ คือ ระยะที่วัดจากจุดกำเนิดไปตั้งฉากกับเส้นตรง

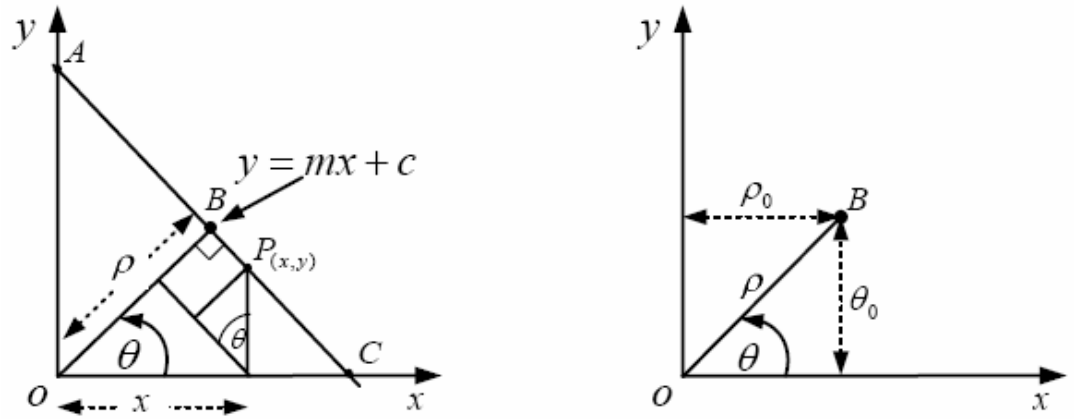
θ คือ ค่าของมุมระหว่างเวกเตอร์ ρ กับแกน x

2.7.2 การค้นหาเส้นตรงใช้ Hough Transform ในภาพสองมิติ

หลักการของ Hough Transform เพื่อการค้นหาองค์ประกอบของภาพสองมิติที่มีส่วนเส้นตรงปรากฏในภาพนั้น เป็นการหาเส้นตรงของภาพจากฟังก์ชันของสมการ 2-20

$$f(x,y,\rho_0,\theta_0) = x\cos\theta_0 + y\sin\theta_0 \tag{2-20}$$

และกำหนดให้จุดของภาพสองมิติมีค่าเท่ากับ (ρ_0,θ_0) เมื่อ ρ_0 ระยะที่วัดจากจุดกำเนิดไปตั้งฉากกับเส้นตรง และจุด θ_0 เป็นค่าของมุมระหว่างเวกเตอร์ ρ กับแกน x ดังแสดงในภาพประกอบ 2-18



(a) Image Space (b) Parameter Space

ภาพประกอบ 2-18 การหาเส้นตรงของภาพสองมิติโดยใช้ Hough Transform

วิธีการ Hough Transform คือ จากสมการ 2-19 มีการกำหนดจุด (x_0, y_0) ของ Image Space ดังนั้นค่าของ ρ จะเท่ากับสมการ 2-21

$$\rho = \sqrt{x_0^2 + y_0^2}\left(\frac{x_0}{\sqrt{x_0^2 + y_0^2}}\cos\theta + \frac{y_0}{\sqrt{x_0^2 + y_0^2}}\sin\theta\right) \qquad (2\text{-}21)$$

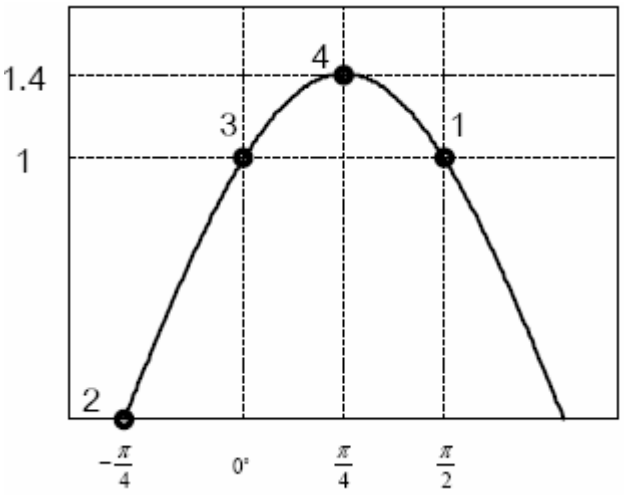
กำหนดให้ $r_0 \cong \sqrt{x_0^2 + y_0^2}$ และ $\alpha_0 \cong \tan^{-1}(\frac{y_0}{x_0})$ และเมื่อแทนค่าของสมการ 2-20 จะได้ค่าของ ρ ตามสมการ 2-22

$$\rho = r_0(\cos\alpha_0\cos\theta + \sin\alpha_0\sin\theta) = r_0(\cos\alpha_0 - \theta) \qquad (2\text{-}22)$$

การแปลงรูปแบบของ HT พบว่า $\rho = x\cos\theta + y\sin\theta$ ของจุด (x_0, y_0) ใน Image Space เป็นการแปลงรูปแบบไปสู่เส้นโค้งรูปซายด์ (Simusoidal Curve) ใน Parameter Space ทิศทางทวนเข็มนาฬิกาและจุด (ρ_0, θ_0) ของรูปเส้นโค้งรูปซายด์นี้ แสดงให้เห็นแทนเส้นตรงที่ลากผ่านจุด (x_0, y_0) ใน Image Space เมื่อทดลองแทนค่า θ และ ρ ตามตาราง 2-1 ดังนั้นผลที่ได้จะเป็นการทำ Hough Transform ระหว่างค่าของเส้นตรงใน Image Space ไปสู่ค่าของจุดใน Hough Space ดังแสดงในภาพประกอบ 2-19

ตาราง 2-1 ค่าความสัมพันธ์ระหว่างค่าของ θ และ ρ เมื่อแทนค่าในสูตร 2-19

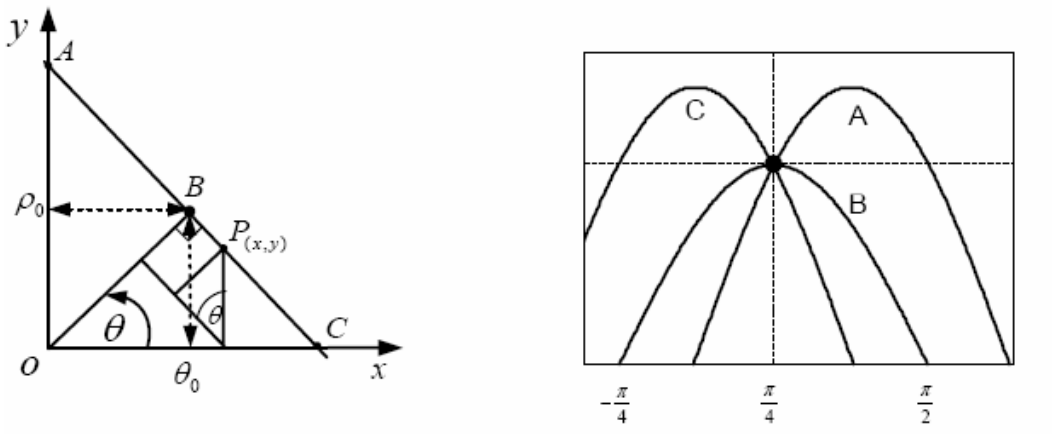
	จุดที่ 1	**จุดที่** 2	**จุดที่** 3	**จุดที่** 4
θ	$\pi/2 = 1.57$	$-\pi/4 = -0.785$	0	$\pi/4 = 0.785$
ρ	1	0	1	1.414



ภาพประกอบ 2-19 ตัวอย่างการทำ Hough Transform

เมื่อทั้ง 4 จุดมีที่เกิดขึ้นใน Hough Space ค่าของจุด 2 3 4 และ 1 ในภาพประกอบ 2-20 (b) จะมีค่าตรงกันในแต่ละเส้นของในภาพประกอบ 2-20 (a) Image Space ลักษณะของภาพแบบการแปลงบทกลับของ Hough Transform ค่าของจุด (ρ_0, θ_0) ใน Parameter Space สามารถอธิบายได้ด้วยบทกลับของการแปลงรูปแบบจาก Spatial Domain ไปสู่การแทนค่าของเส้นตรงและอธิบายได้ด้วยโดยสมการ 2-23 [19]

$$\rho = x\cos\theta + y\sin\theta - \rho_0 = 0 \qquad (2\text{-}23)$$



(a) Image Space (b) Hough Space

ภาพประกอบ 2-20 การแปลงค่าของ Hough Space เมื่อมีการเปลี่ยนแปลงจุดในเส้นตรง

จากภาพประกอบ 2-20 ตัวอย่างของ 3 จุดของตำแหน่ง A(0,2) B(1,1) และ C(2,0) ของทุกจุดบนเส้นตรง จุดดังกล่าวนี้เป็นจุดที่ตรงใน Hough Space ในความเป็นจริงแล้วจุดเป็นค่าหนึ่งของสมการเส้นตรงเส้นที่สองและสามและมีค่าสอดคล้องกับเส้นโค้งที่ร่วมอยู่ในจุดของค่าเวกเตอร์ ρ เท่ากับ 1.414 และค่าของมุม θ เท่ากับ 0.79 เมื่อนำค่าของ (ρ,θ) แทนค่าในสมการ 2-19 ได้ดังนี้

$$1.414 = x\cos(0.79) + y\sin(0.79) \qquad (2\text{-}24)$$

ดังนั้นค่าของ $x + y = 2$

เมื่อเราทำ Hough Transform แล้วจุดที่ $P(x,y)$ ที่ปรากฏอยู่ในภาพจะมีเส้นตรงจำนวนมากมายที่ลากผ่านได้ ดังนั้นวิธีการของ Hough transform คือการนับค่าว่าจุด $P(x,y)$ ดังกล่าวมีจำนวนเส้นของเส้นตรงลากผ่านจุดนี้จำนวนเท่าไร และถ้าพิจารณาใน Hough Space ก็จะมองเห็นเป็นจุดที่มีค่าของเส้นโค้งรูปซายด์ตัดผ่าน แสดงว่าจุดนั้นคือจุดเด่นของเส้นตรงที่ผ่านจุด $P(x,y)$ มีค่ามากที่สุด จุดดังกล่าวมีค่าระยะห่างจากจุดกำเนิดมากที่สุดก็ต่อเมื่อเวกเตอร์ ρ ตั้งฉากกับเส้นตรงที่ผ่านจุด $P(x,y)$ ดังกล่าวนั่นเอง

ในบทนี้ได้กล่าวถึงหลักการพื้นฐานและเทคนิควิธีการประมวลผลภาพ โดยมุ่งเน้นไปที่การอธิบายทฤษฎีที่ใช้ในการประมวลผลสัญญาณภาพดิจิตอลตั้งแต่ รูปร่างของภาพ มาตรฐานสี การประมวลผลภาพเบื้องต้นด้วยหลักการตัวกรองประเภทต่างๆ การแปลงภาพสีเป็นภาพ

ระดับสีเทา นอกจากนั้นได้กล่าวถึงเทคนิคขั้นตอนอื่นๆที่จะใช้ในงานวิจัยนี้ นั้นคือ การเรียนรู้วิธีการหาขอบภาพด้วยอัลกอริทึมต่างๆ และเทคนิคการหาเส้นตรงด้วยการใช้เทคนิคของฮัฟทรานส์ฟอร์ม ซึ่งข้อมูลเหล่านี้จะได้นำไปใช้ในการพัฒนาการออกแบบภาษาระดับสูงเพื่อให้สามารถนำไปใช้งานบนบอร์ดเอฟพีจีเอได้

บทที่ 3

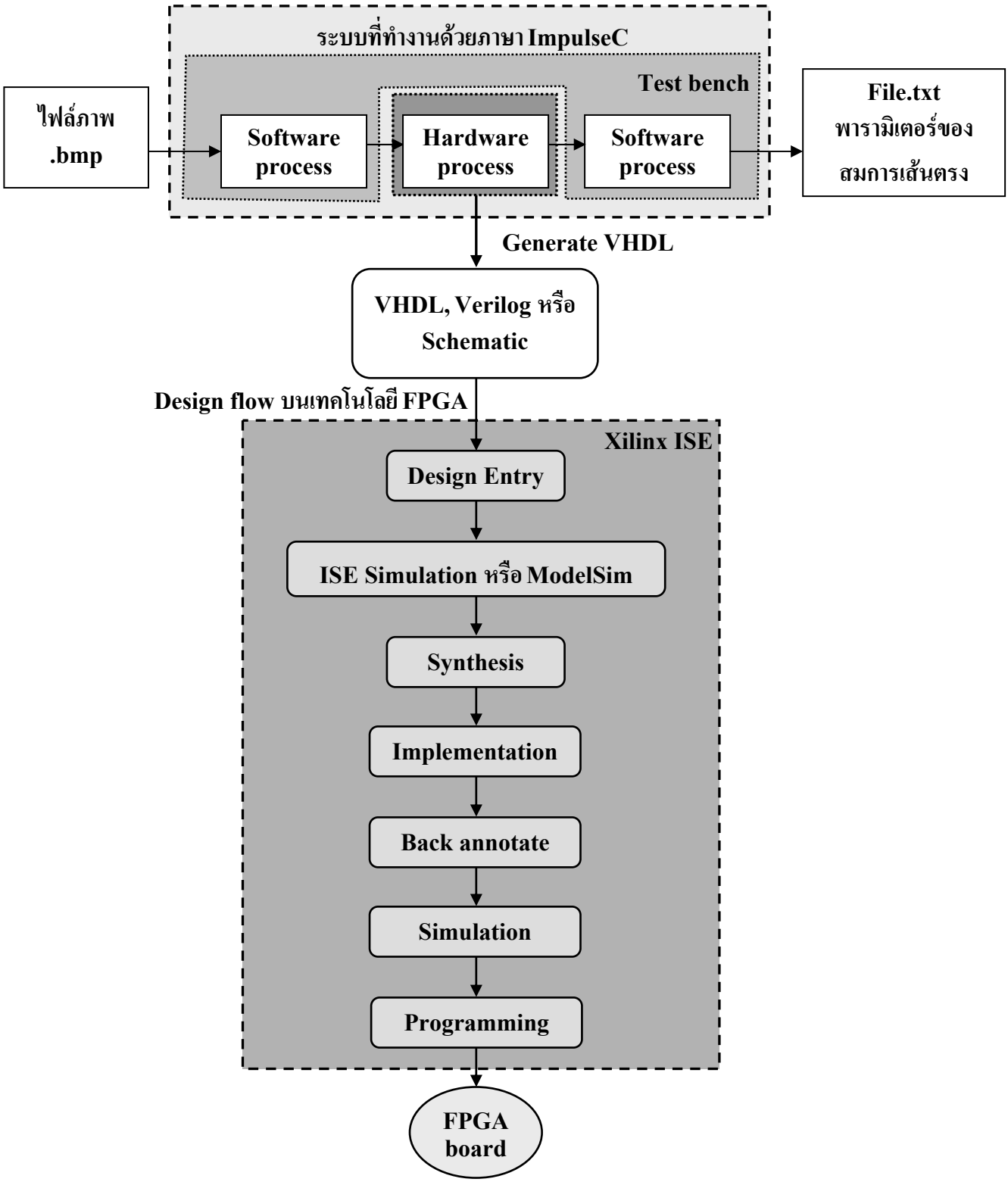
การออกแบบระบบ

งานวิจัยนี้มีจุดประสงค์เพื่อพัฒนาวงจรด้วยภาษาระดับสูงที่ชื่อว่า ImpulseC ให้สามารถทำการประมวลผลเพื่อหาขอบภาพและสมการเส้นตรงที่สามารถนำไปใช้งานร่วมกับเอฟพีจีเอ โดยกระบวนการที่สนใจนั้น ได้แก่ การหาโครงสร้างสถาปัตยกรรมของการประมวลผลหาขอบของภาพที่ให้ประสิทธิภาพทางด้านความเร็วที่ดี และการพัฒนาวงจรที่ใช้ในการหาเส้นตรงจากภาพ ทั้งนี้เพื่อลดภาระการประมวลผลสัญญาณภาพบนเครื่องคอมพิวเตอร์ทั่วไป โดยใช้ฮาร์ดแวร์แทนการทำงานของซอฟต์แวร์ที่ใช้ในการหาขอบภาพและสมการเส้นตรง จะเป็นการช่วยเพิ่มประสิทธิภาพในการประมวลผลสัญญาณภาพ ในบทนี้จะกล่าวถึงส่วนของการออกแบบและพัฒนาระบบเป็นหลัก

3.1 ขั้นตอนในการออกแบบระบบ

แนวความคิดในการออกแบบระบบแบ่งออกเป็น 2 ส่วนหลัก คือ การพัฒนาโปรแกรมแทนการทำงานการประมวลผลสัญญาณภาพด้วยภาษาระดับสูง “ImpulseC” และการนำไฟล์ VHDL หรือ Verilog ซึ่งเป็นผลลัพธ์ที่ได้จากซอฟต์แวร์ช่วยออกแบบในส่วนแรกไปทำการจำลองการทำงานของวงจรบนซอฟต์แวร์จำลองการทำงานและดำเนินการตามขั้นตอนของการออกแบบระบบ ดังภาพประกอบ 3-1

การประมวลผลภาพใช้ทรัพยากรของคอมพิวเตอร์ในการประมวลผลค่อนข้างมาก ดังนั้นจึงมีความพยายามในการสร้างวงจรเฉพาะทางสำหรับประมวลผลภาพเพื่อเพิ่มความเร็วแทนการใช้ซอฟต์แวร์ที่ทำงานอยู่บนเครื่องคอมพิวเตอร์ทั่วไป แต่นักวิจัยหรือผู้พัฒนางานทางด้านประมวลผลภาพอาจจะไม่มีความถนัดในการออกแบบวงจร ดังนั้น จึงมีการใช้ภาษาระดับสูงที่ใกล้เคียงกับภาษาซีหรือ C++ ที่ผู้พัฒนางานทางการประมวลผลภาพใช้กันทั่วไป มาช่วยในการออกแบบวงจร ในที่นี้เลือกใช้ภาษา ImpulseC เนื่องจากมีซอฟต์แวร์ช่วยออกแบบมารองรับในการแปลงให้ได้เป็นพฤติกรรมของวงจรในรูปแบบของภาษา VHDL หรือ Verilog และมีราคาของ license ไม่แพงมากเมื่อเปรียบเทียบกับภาษาอื่นๆ เช่น System C หรือ Handle C เป็นต้น



ภาพประกอบ 3-1 ขั้นตอนการออกแบบระบบ

กระบวนการในการหาขอบภาพมีด้วยกันหลายอัลกอริทึมให้เลือกใช้ ซึ่งในงานวิจัยนี้เลือกใช้อัลกอริทึม Robert, Sobel และ Prewitt Edge Detection โดยนำเอาอัลกอริทึมเหล่านี้มาเปรียบเทียบหาผลลัพธ์ของการหาขอบภาพที่เหมาะสมและดีที่สุดเพียงอัลกอริทึมเดียว ด้วยการออกแบบระบบประมวลผลในแบบ Single Core

กระบวนการที่จะนำมาใช้ในการหาเส้นตรงของภาพ คือ Hough Transform โดยค่าอินพุตที่ใช้สำหรับกระบวนการของฮัฟนั้นต้องการข้อมูลขนาด 8 บิต/พิกเซลก็เพียงพอแล้ว ดังนั้นภาพอินพุตสำหรับงานวิจัยนี้ที่เป็น .bmp จึงต้องทำการแปลงภาพสีขนาด 24 บิตให้เหลือเพียง 8 บิต ด้วยการใช้วิธีการแปลงภาพสีเป็นภาพระดับสีเทาก่อนจะนำไปผ่านฟังก์ชันของการหาขอบภาพและการหาเส้นตรงของภาพตามลำดับ โดยสุดท้ายแล้วผลลัพธ์ที่ได้คือไฟล์ .txt ซึ่งจะมีรายละเอียดของสมการเส้นตรงที่ได้จากภาพอินพุต

ในขั้นตอนสุดท้าย จะทำการสร้างไฟล์ VHDL จากซอฟต์แวร์ช่วยออกแบบภาษา ImpulseC นำไฟล์ VHDL ที่ได้นั้นไปสังเคราะห์และจำลองการทำงานบนซอฟต์แวร์ “ISE Simulator” ของบริษัท Xilinx เพื่อวิเคราะห์ผลลัพธ์และตรวจสอบความถูกต้องเทียบกับผลลัพธ์จาก Matlab โดยที่สามารถใช้ระบบและชุดทดสอบเดียวกันกับ ImpulseC ได้

3.2 การออกแบบร่วมระหว่างฮาร์ดแวร์-ซอฟต์แวร์ (Hardware-software codesign) [20]

ในการออกแบบระบบที่มีทั้งส่วนประกอบของทั้งตัวโปรเซสเซอร์ และอุปกรณ์ต่อร่วม เช่น วงจรลอจิกที่อาจสร้างอยู่ในไอซีที่ใช้งานเฉพาะอย่าง (ASIC: Application specific IC) หรือ FPGA (Field Programmable Gate Array) นอกจากนี้ในระบบอาจจะมีตัวเซนเซอร์คอยตรวจจับ รวมทั้งอุปกรณ์ทางกลที่สามารถตอบสนองสภาวะการทำงานในตอนนั้นเพื่อให้ทั้งระบบมีการทำงานร่วมกัน ในการออกแบบ จำเป็นจะต้องมีการเขียนทั้งส่วนของโปรแกรมที่จะต้องประมวลผลในตัวโปรเซสเซอร์ และส่วนของการออกแบบลอจิกในอุปกรณ์ต่อร่วมต่างๆ ซึ่งถ้าหากมองดูผิวเผินแล้วอาจจะง่าย แต่การออกแบบระบบนี้ จะเกิดปัญหาระหว่างอุปกรณ์ต่างๆที่ต้องการให้ทำงานร่วมกัน เช่น เรื่องของความเร็วในการทำงานของอุปกรณ์แต่ละตัวที่ไม่เท่ากัน การซิงค์โครไนซ์ในระหว่างการทำงาน รวมถึงการแก้ไข (Debugging) ระบบ

3.2.1 ฮาร์ดแวร์-ซอฟต์แวร์ (Hardware-software)

ฮาร์ดแวร์ในลักษณะของระบบที่เป็น Codesign จะเป็นอุปกรณ์ที่สามารถทำงานได้อย่างอิสระ อาจประกอบด้วยส่วนของวงจรดิจิตอล หรืออะนาลอกหลายๆส่วนที่สามารถทำงาน

พร้อมๆกันได้ (Concurrency) ทำให้มีความเร็วในการทำงานสูง เช่น ASIC ที่เป็นฮาร์ดแวร์ที่ใช้งานเฉพาะอย่าง หรือ FPGA ที่เป็นฮาร์ดแวร์ที่สามารถโปรแกรมหรือเปลี่ยนแปลงได้ ที่เรียกว่า รีคอนฟิกกูเรเบิลฮาร์ดแวร์ (Reconfigurable hardware) สำหรับส่วนของซอฟต์แวร์นั้นจะหมายถึงตัวประมวลผลที่ต้องทำงานโดยใช้โปรแกรมหรือคำสั่ง (Instruction codes) โดยมีการทำงานเป็นแบบเรียง ลำดับทีละคำสั่ง (Sequential) ตัวอย่างเช่น (ไมโคร)โปรเซสเซอร์ที่ใช้งานทั่วไป (General purpose processor) ตัวอย่างที่ง่ายและชัดเจนที่สุดก็คือ เครื่องคอมพิวเตอร์ส่วนบุคคล ซึ่งมีส่วนของซอฟต์แวร์ คือ CPU ส่วนของฮาร์ดแวร์อาจจะเป็นกราฟฟิกการ์ดที่ช่วยเร่งความเร็วในการแสดงผลภาพ โดยที่มีการติดต่อกันผ่าน AGP พอร์ต หรือ PCI เป็นต้น

3.2.2 วิธีการออกแบบระบบดิจิตอล

ในการออกแบบระบบทางดิจิตอลเพื่อการประยุกต์ใช้งาน Application ในด้านต่างๆนั้น มีหลายๆวิธีการเริ่มต้นตั้งแต่การใช้วิธีการทางด้านซอฟต์แวร์ที่มีการเขียนโปรแกรมบอกลักษณะขั้นตอนการทำงานของระบบ ให้ไปทำงานบนโปรเซสเซอร์ที่มีความยืดหยุ่นในการใช้งาน เนื่องจากสามารถโปรแกรมได้ใหม่ตลอดเวลาและยังมีราคาที่ไม่แพง แต่ข้อเสียก็คือทำงานได้ช้า เนื่องจากกระบวนการประมวลผลต้องกระทำทีละบรรทัดของโปรแกรม การปรุบปรุงอาจทำได้โดยการใช้โครงสร้างโปรเซสเซอร์ที่มีประสิทธิภาพมากขึ้น เช่น Parallel processor, DSP (Digital signal processor), RISC (Reduced instruction set computer), หรือเพิ่มความถี่สัญญาณนาฬิกาที่จะทำได้เมื่อทีเทคโนโลยีการผลิตไอซีที่ก้าวหน้าขึ้นในแต่ละปี

สำหรับการใช้งานเฉพาะอย่าง การใช้วิธีการทางด้านฮาร์ดแวร์ทั้งหมด เช่น การสร้าง ASIC ชิป สามารถเพิ่มประสิทธิภาพของการทำงานได้อย่างมากมาย เนื่องจากสามารถทำงานได้เร็ว รวมทั้งการกินไฟต่ำ แต่ก็มีข้อเสีย นั่นคือ มีความยืดหยุ่นในการใช้งานน้อยเพราะใช้ในงานเฉพาะและต้องผลิตเป็นจำนวนมากๆ เนื่องจากค่าใช้จ่ายในการออกแบบและผลิตที่สูง

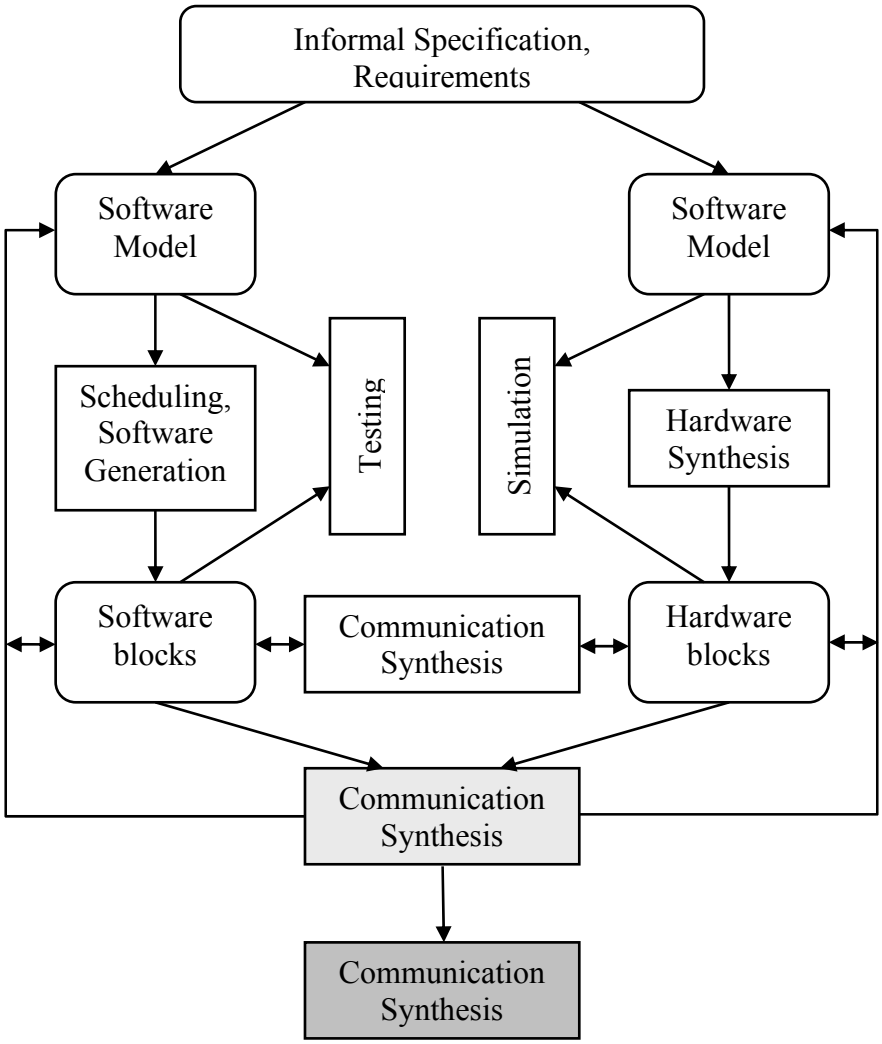
บางครั้งในการรักษาดุลยภาพของระบบทั้งทางด้านประสิทธิภาพและราคา ระบบที่ทำงานโดยใช้โปรเซสเซอร์อย่างเดียวอาจจะมีราคาถูก มีความยืดหยุ่นในการใช้งานสูง แต่จะทำงานได้ช้าเมื่อเทียบกับระบบที่มีฮาร์ดแวร์ที่มีวงจรลอจิกที่สามารถทำงานได้พร้อมๆกัน จะสามารถทำงานได้เร็วกว่ามาก แต่ข้อเสียคือราคาแพง ดังนั้นทางสายกลางที่จะมาพบกันได้ทั้งราคาและประสิทธิภาพก็คือใช้ทั้งสองอย่าง หรือเป็นระบบ Codesign ตามความเหมาะสมในการออกแบบใช้งาน ที่จะให้การทำงานส่วนไหนไปอยู่บนตัวอุปกรณ์อะไร เช่นส่วนที่ต้องการการคำนวณหรือประมวลผลมากๆก็ควรที่จะให้อยู่ในฮาร์ดแวร์

จะเห็นได้ว่าแต่ละลักษณะของการออกแบบมีข้อดี-ข้อเสียแตกต่างกัน บางครั้งเมื่อต้องการทั้งความยืดหยุ่นที่สูงเหมือนซอฟต์แวร์และทำงานได้รวดเร็วเหมือนฮาร์ดแวร์ ซึ่ง FPGA ดูเหมือนจะเป็นคำตอบสำหรับความต้องการทั้งสองนี้ นั่นคือเป็นฮาร์ดแวร์ที่สามารถโปรแกรมได้ จะเห็นได้ว่ามีการใช้งานใน Application ต่างๆอย่างมากมาย และมีความนิยมเพิ่มขึ้นอย่างรวดเร็วในปัจจุบัน การออกแบบสามารถทำได้ง่ายโดยใช้ภาษาขั้นสูง เช่น VHDL หรือ Verilog การโปรแกรมก็สามารถทำได้เอง โดยไม่มีค่าใช้จ่ายเพิ่มเติม และไม่มีความเสียงใดๆทั้งสิ้น เนื่องจากสามารถโปรแกรมใช้งานใหม่ได้ แต่ราคาค่อนข้างแพงเมื่อเทียบกับไมโครโปรเซสเซอร์

3.2.3 นิยามของ Codesign

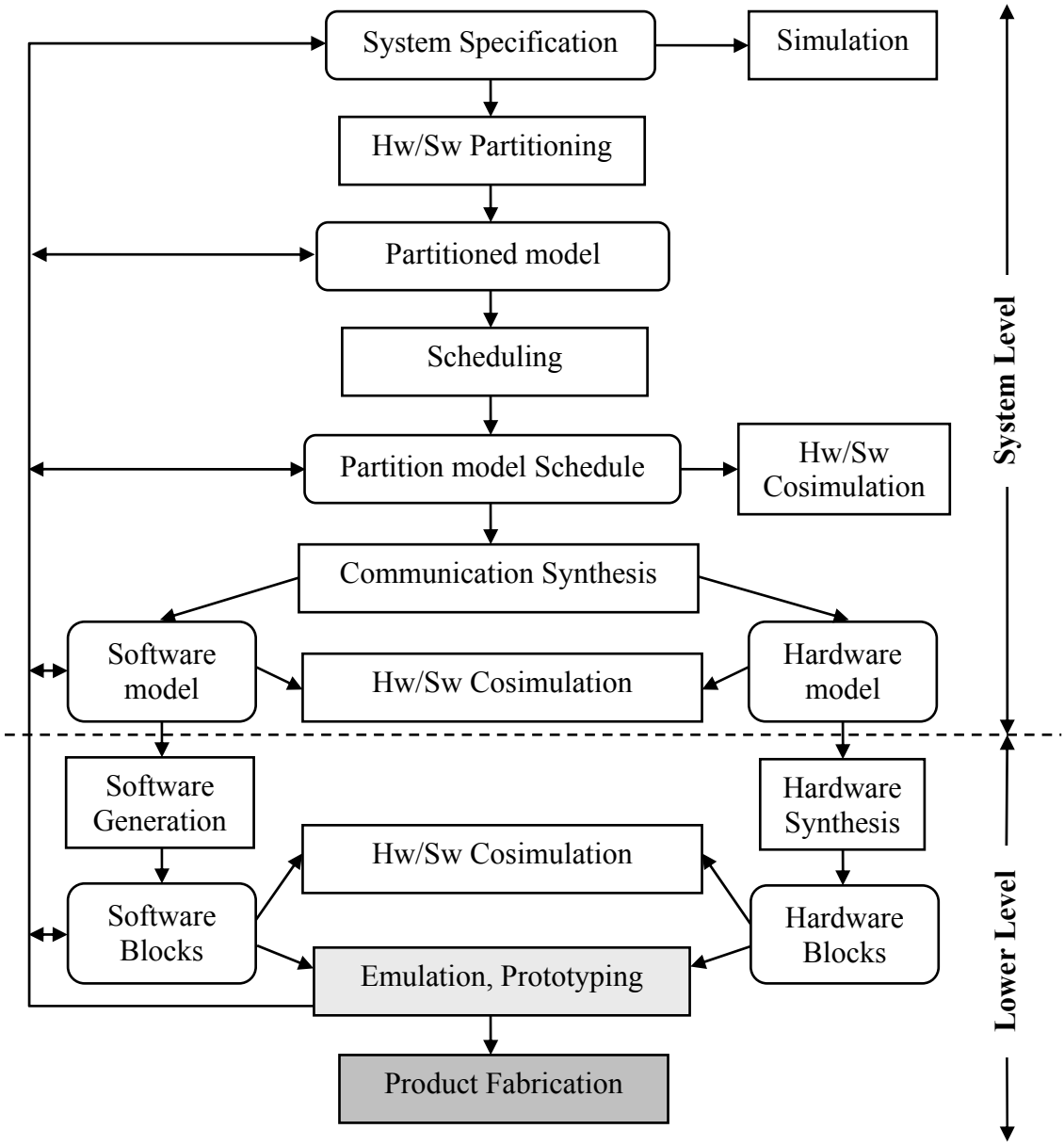
นิยามของ Codesign ที่ให้โดย Franke และ Purvis ในบทความของเขาในปี 1991 ว่า “เป็นการออกแบบระบบที่มีทั้งฮาร์ดแวร์และซอฟต์แวร์พร้อมๆกัน ตั้งแต่จุดเริ่มต้นของการออกแบบ เพื่อความยืดหยุ่นในการออกแบบ และประสิทธิภาพของการจัดการฟังก์ชันการทำงานในระบบอย่างเหมาะสม” Wayne Woft ให้นิยามเพิ่มเติมว่า “ระบบจะต้องออกแบบร่วมกันทั้งในส่วนของฮาร์ดแวร์และซอฟต์แวร์ เพื่อเป็นการประกันว่าสามารถทำงานได้ในตอนท้ายที่สุด ซึ่งมีประสิทธิภาพ ความเชื่อถือได้ และราคาตามที่ต้องการ”

ดังนั้นการออกแบบร่วมกันระหว่างฮาร์ดแวร์-ซอฟต์แวร์เป็นการสร้างระบบที่มีความสมดุลกันระหว่างประสิทธิภาพ (System performance) ที่ได้มาจากฮาร์ดแวร์ และราคาที่ได้มาจากซอฟต์แวร์ หลักการของ Codesign คือ การหลีกเลี่ยงการแยกกันของการออกแบบฮาร์ดแวร์ และการออกแบบซอฟต์แวร์ โดยได้ผลลัพธ์เป็นระบบที่มีประสิทธิภาพ ภายในระยะเวลาที่สั้น ยิ่งถ้าสามารถที่จะทำให้ฮาร์ดแวร์ในระบบมีความยืดหยุ่นได้โดยใช้ FPGA ก็จะทำให้มีความน่าสนใจในการใช้งานเพิ่มขึ้นอย่างมาก



ภาพประกอบ 3-2 การออกแบบระบบแบบเดิมทั่วไปที่แยกกันออกแบบ

จากภาพประกอบ 3-2 การออกแบบระบบที่มีทั้งฮาร์ดแวร์และซอฟต์แวร์แบบเก่าที่นิยมทำกันโดยทั่วไป จะเป็นการออกแบบและทดสอบที่แยกกันโดยสิ้นเชิง ระหว่างผู้ออกแบบโปรแกรมซอฟต์แวร์และผู้ออกแบบวงจรฮาร์ดแวร์ โดยปราศจากการพยายามที่จะทดสอบร่วมกันหรือ Cosimulation ระหว่างการออกแบบ ซึ่งจะเสี่ยงอย่างมากในการที่จะได้ระบบที่สมบูรณ์แบบในตอนท้ายของการสร้างใช้งานจริง ซึ่งผู้ออกแบบจะต้องกลับมาเสียเวลาออกแบบ เพื่อเป็นการปรับแต่งการทำงานให้เข้ากันได้ระหว่างซอฟต์แวร์และฮาร์ดแวร์

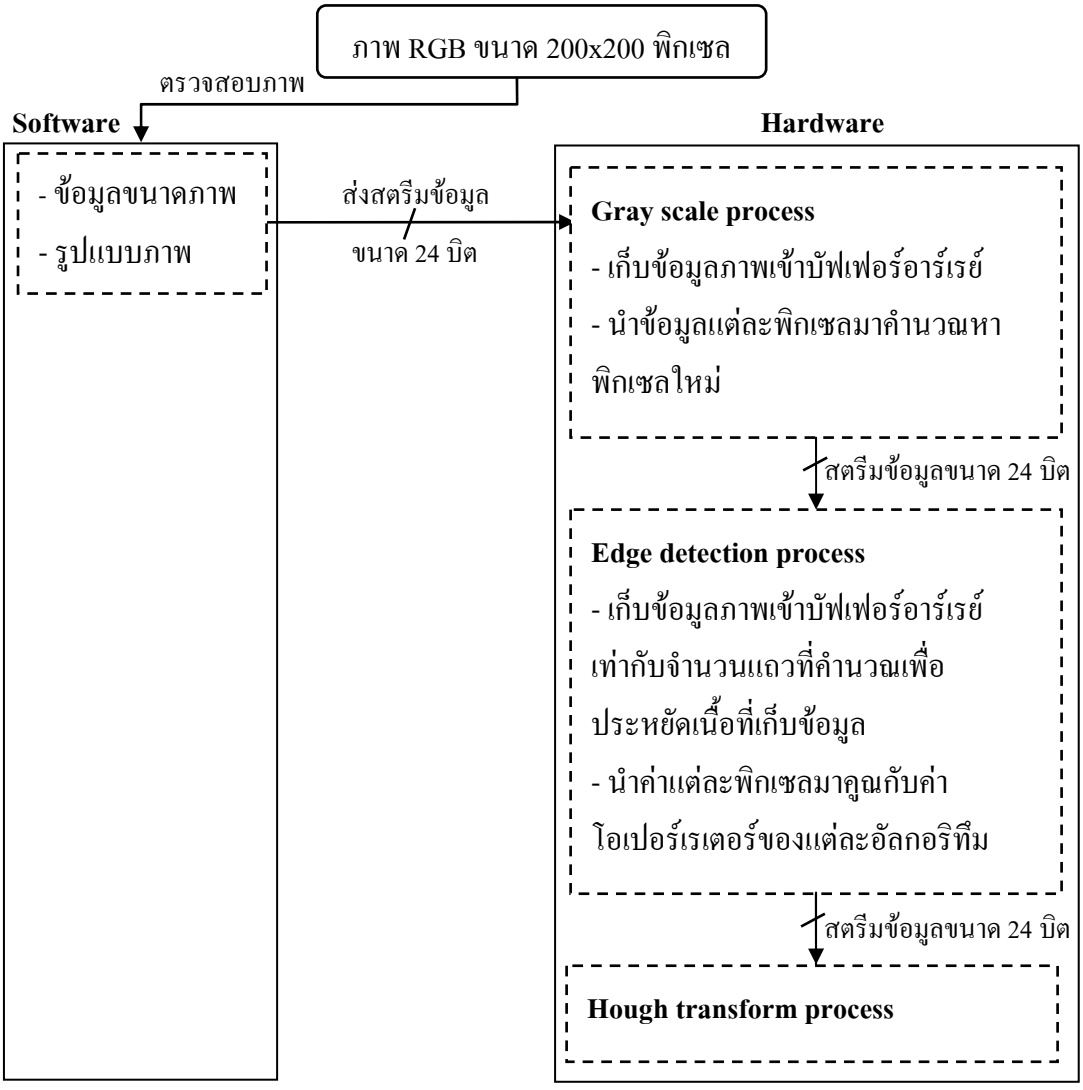


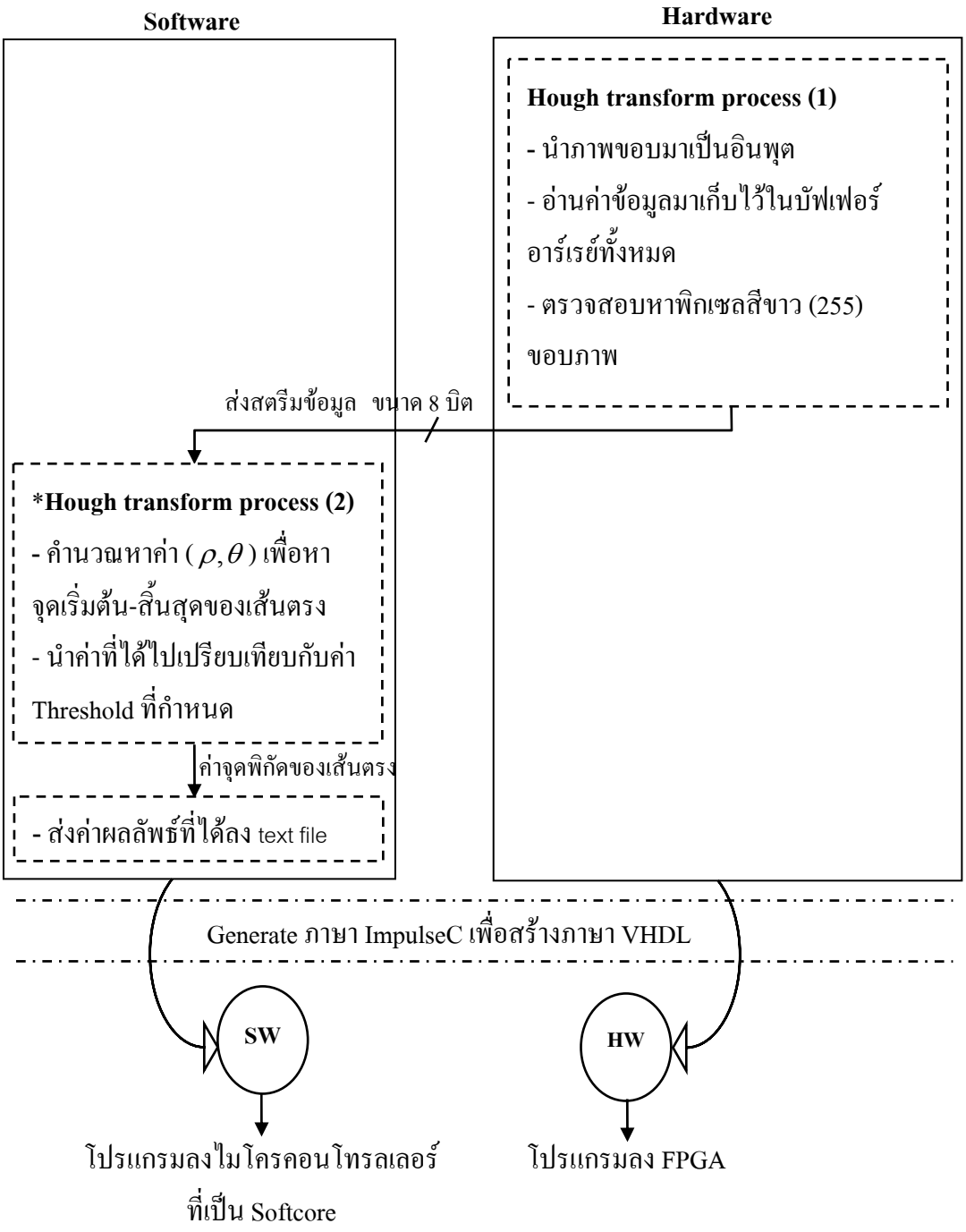
ภาพประกอบ 3-3 การออกแบบร่วมกันแบบ Codesign

จากภาพประกอบ 3-3 สำหรับระบบที่เรียกว่า Hardware/Software codesign นั้น ผู้ออกแบบเริ่มต้นออกแบบจากภาษาระดับสูง (High-level design language) ในการกำหนดความสามารถของระบบ หลังจากนั้นก็พยายามออกแบบทั้งในส่วนของฮาร์ดแวร์และซอฟต์แวร์ไปพร้อมๆกัน โดย Codesign tools เช่น Partitioner ที่แบ่งส่วนงานให้ไปทำที่ฮาร์ดแวร์หรือซอฟต์แวร์ Scheduler ที่เรียงลำดับการทำงานในส่วนต่างๆของระบบทั้งที่อยู่ในฮาร์ดแวร์และซอฟต์แวร์ โดยมีการกระทำที่เรียกว่า Cosimulation เพื่อทดสอบการทำงานร่วมกันตลอดเวลา ซึ่งทำให้ผู้ออกแบบมีความมั่นใจมากขึ้นก่อนที่จะไปสร้าง (Implement) จริง

3.3 ภาพรวมการออกแบบโปรแกรม

ImpulseC เป็นภาษาที่เหมาะสำหรับการออกแบบการทำงานร่วมระหว่างฮาร์ดแวร์และซอฟต์แวร์ สามารถพัฒนาได้ง่ายเพราะใช้ภาษาซีเป็นภาษาหลักในการพัฒนา อาจจะแตกต่างจากภาษาซีทั่วไปเล็กน้อยตรงที่มีฟังก์ชันเฉพาะให้เรียกใช้งาน และสามารถแปลงภาษาได้อัตโนมัติ เช่น VHDL, Verilog หรือสามารถสร้างระบบให้เข้ากับ embedded FPGA เช่น Microblaze เป็นต้น โดยโครงของภาษา ImpulseC จะแบ่ง Source File ออกเป็น 2 ส่วนหลัก คือ Hardware Process และ Software Process ซึ่งฟังก์ชันการประมวลผลหลักของโปรแกรมที่ต้องการสร้างเป็นวงจรจะถูกจัดไว้ในส่วนของ Hardware Process ดังภาพประกอบ 3-4





ภาพประกอบ 3-4 แผนภาพการออกแบบโปรแกรมโดยรวมของระบบ

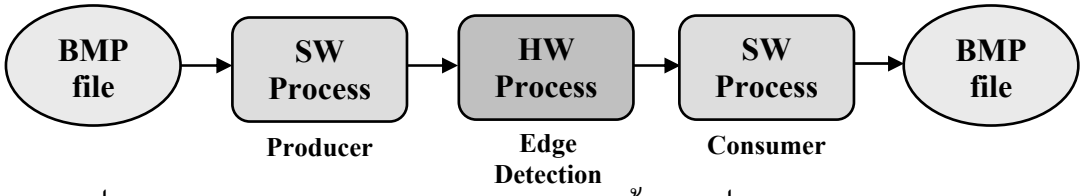
*สาเหตุที่ต้องนำส่วนการประมวลผลของ Hough transform process บางส่วนมาคำนวณใน Software process เพื่อลดการใช้พื้นที่ในการจองอาร์เรย์ เพราะอาร์เรย์เหล่านี้จะถูกแปลงให้เป็น Ram เมื่อนำไปใช้งานบน FPGA

ในส่วนนี้จะกล่าวถึงภาพรวมการออกแบบระบบด้วยภาษา ImpulseC ที่ใช้ในงานวิจัยนี้ โดยมีรายละเอียด ดังนี้

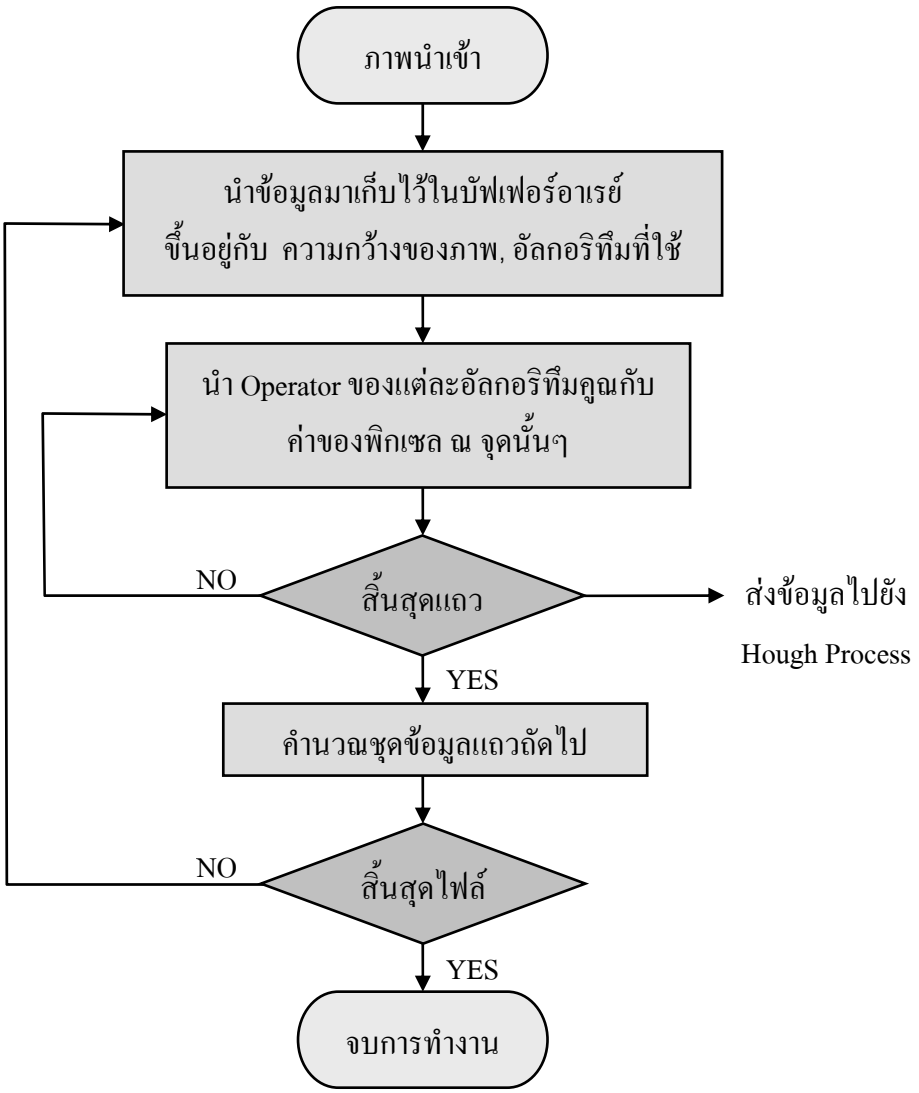
3.3.1 การพัฒนาระบบด้วยภาษา ImpulseC

กระบวนการที่นำมาทดสอบคือการหาขอบภาพ (Edge Detection Methods) ซึ่งมีหลายอัลกอริทึมในการหาขอบภาพ แต่ในงานวิจัยนี้จะนำอัลกอริทึมเพียง 3 แบบมาทำการทดสอบ ได้แก่ Robert, Sobel และ Prewitt Edge Detection

ส่วนที่หนึ่ง เขียนอธิบายการทำงานของอัลกอริทึมทั้ง 3 แบบด้วยภาษา ImpulseC ในรูปแบบโครงสร้างแบบ Single Core ดังภาพประกอบ 3-5 เพื่อหาอัลกอริทึมที่ให้ผลลัพธ์ขอบภาพที่มีความคมชัดและเหมาะสมที่สุดเพียงอัลกอริทึมเดียว

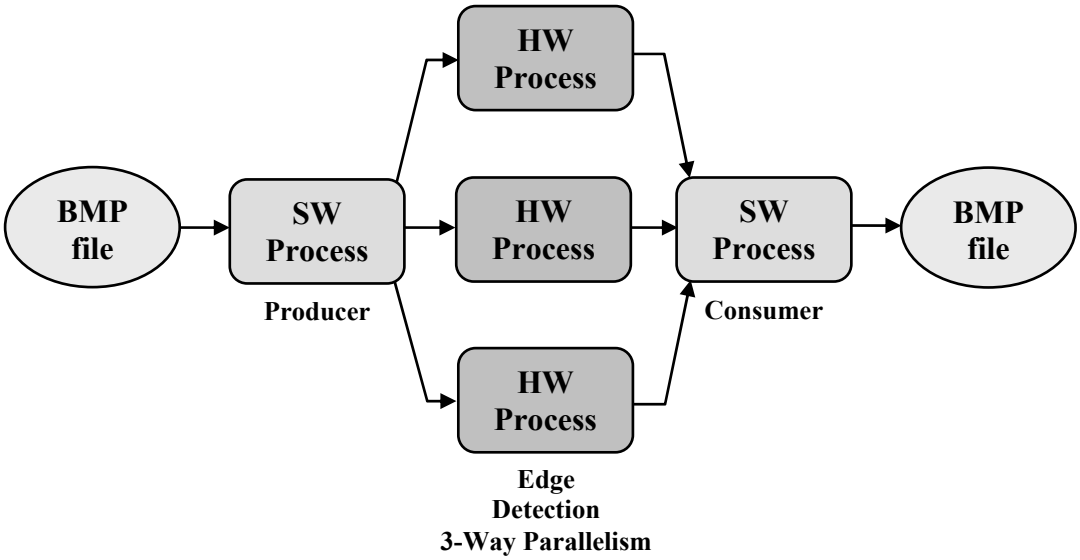


ภาพประกอบ 3-5 แผนภาพกระบวนการหาขอบภาพด้วยการประมวลผลแบบ Single Core



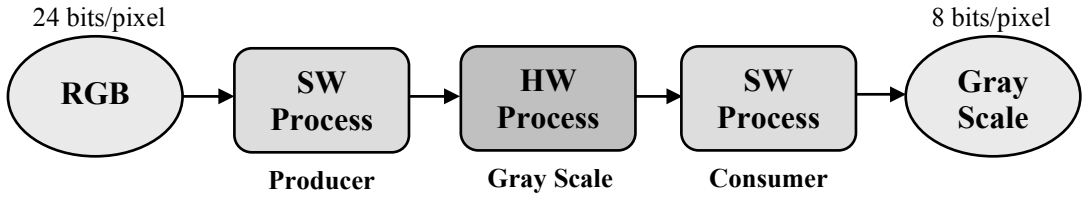
ภาพประกอบ 3-6 ผังการทำงานของฟังก์ชัน Edge Detection ในส่วนของ HW Process

นอกจากนี้ยังสามารถออกแบบโครงสร้างการทำงานของการหาขอบภาพให้เป็นแบบขนาน 3-way parallelism ได้ดังภาพประกอบ 3-7 โดยแบ่งการประมวลผลเป็น 3 ส่วนพร้อมๆกันตามสีภาพ R, G, B แต่ผลลัพธ์ของระบบภาพที่ได้ไม่ได้มีความแตกต่างจากการใช้ภาพแบบ Gray Scale ดังนั้นจึงไม่เลือกใช้โครงสร้างแบบขนานที่มีขนาดวงจรใหญ่มากกว่า

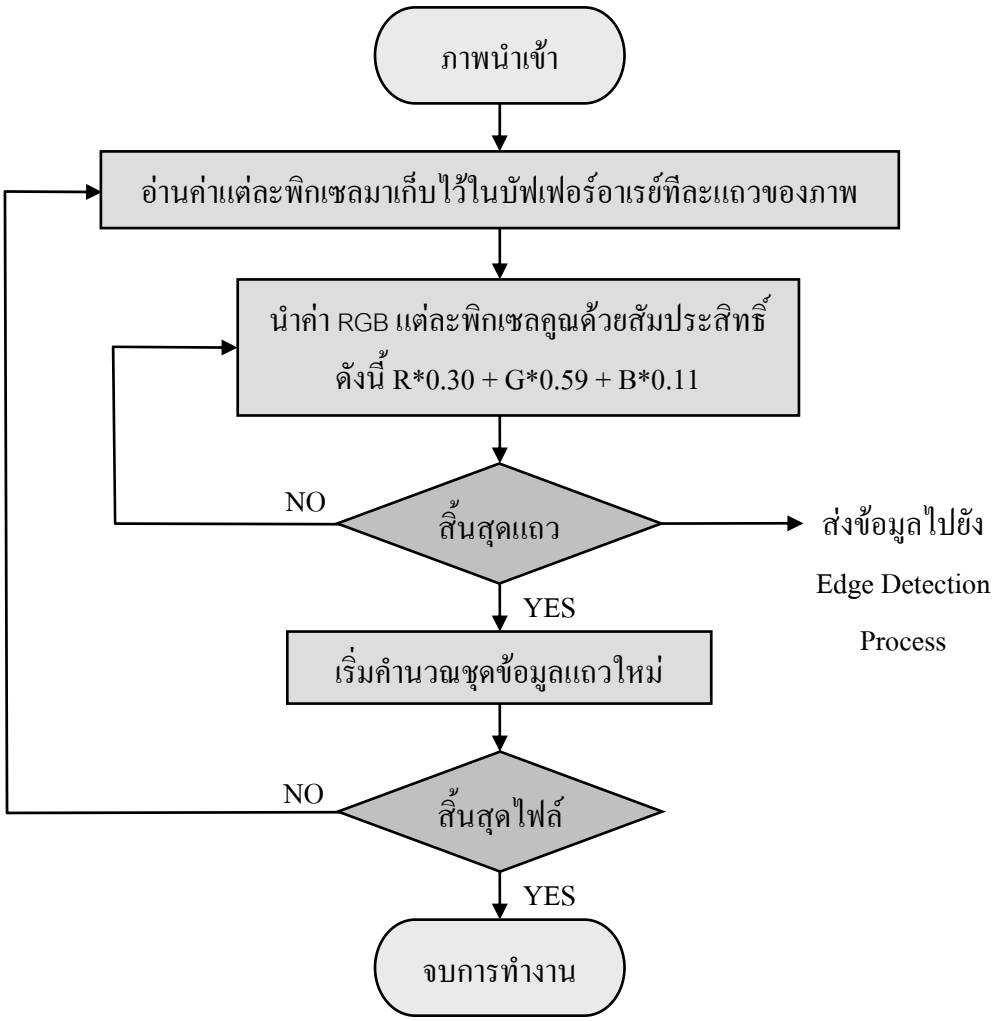


ภาพประกอบ 3-7 แผนภาพกระบวนการหาขอบภาพด้วยการประมวลผลแบบขนาน

ส่วนที่สอง เป็นการนำผลลัพธ์ขอบภาพจากในส่วนแรกมาใช้เพื่อหาเส้นตรงของภาพด้วยการใช้วิธีของฮัฟ แต่เนื่องจากการหาเส้นตรงด้วยวิธีฮัฟนั้นต้องการเพียงข้อมูลขนาด 8 บิต/พิกเซล แต่ภาพอินพุตที่รับเข้ามาเป็นภาพสีขนาด 24 บิต/พิกเซล ดังนั้นเพื่อเพิ่มความเร็วในการประมวลผลให้กับระบบที่ออกแบบ จึงใช้วิธีการแปลงภาพสีเป็นภาพระดับสีเทาก่อนเป็นอันดับแรกดังภาพประกอบ 3-8

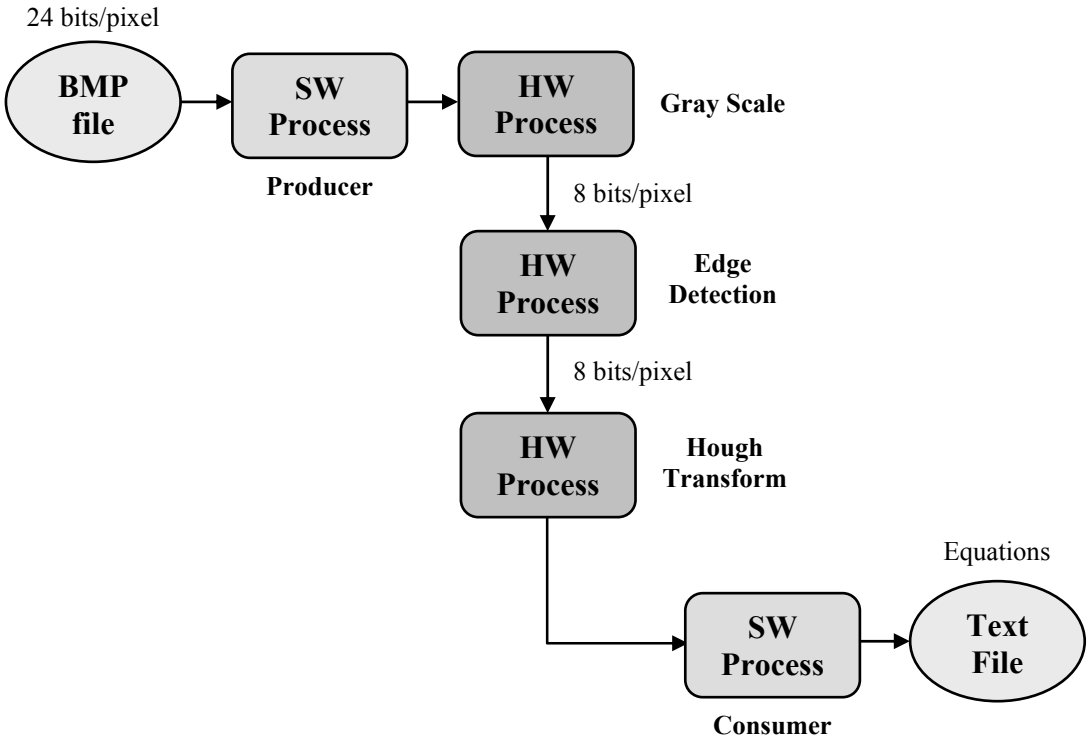


ภาพประกอบ 3-8 แผนภาพกระบวนการแปลงภาพสีเป็นภาพระดับสีเทา

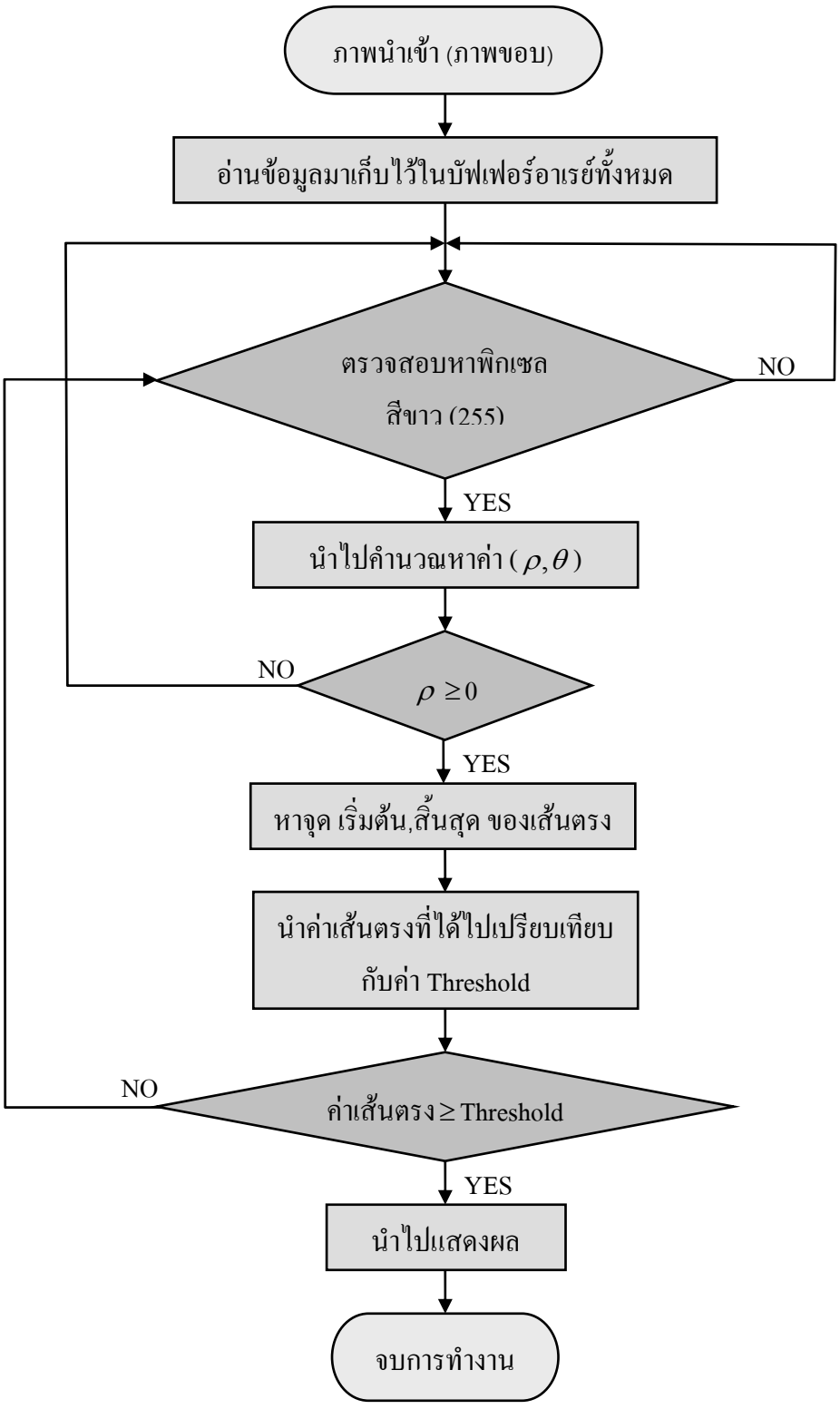


ภาพประกอบ 3-9 ผังการทำงานของฟังก์ชัน Gray Scale ในส่วนของ HW Process

เมื่อได้ภาพระดับสีเทาซึ่งเป็นข้อมูลขนาด 8 บิต/พิกเซล ก็นำไปเข้ากระบวนการหาขอบภาพ จากนั้นเมื่อได้เส้นขอบภาพมาแล้ว จึงจะนำไปเข้ากระบวนการหาเส้นตรงของภาพในขั้นตอนสุดท้ายดังภาพประกอบ 3-10



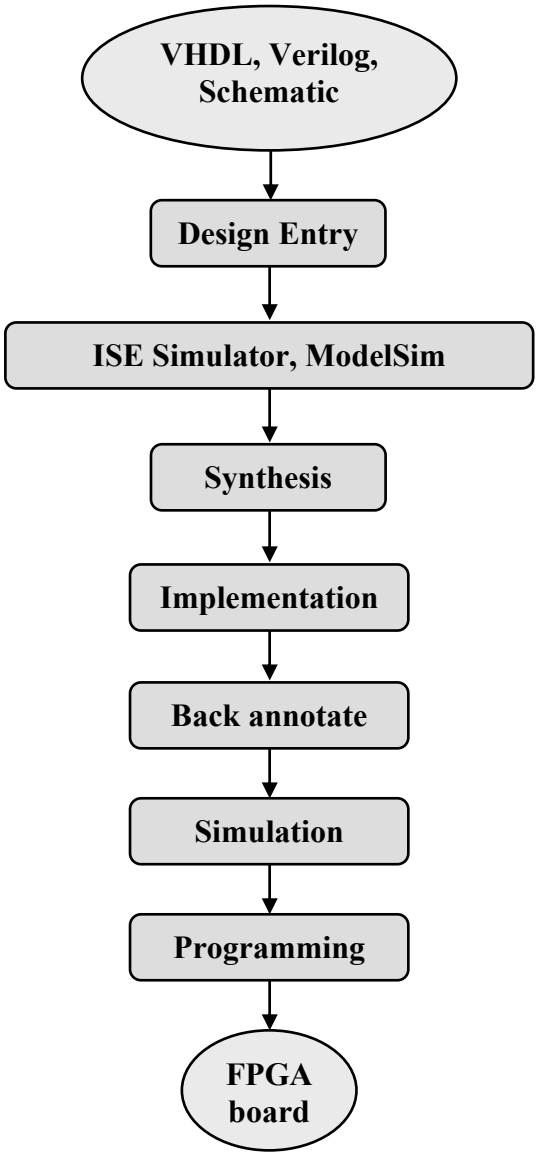
ภาพประกอบ 3-10 แผนภาพแสดงกระบวนการทั้งหมดของการหาเส้นตรงของภาพ



ภาพประกอบ 3-11 ผังการทำงานของฟังก์ชัน Hough Transform Process ในส่วนของ HW Process

3.3.2 ขั้นตอนการพัฒนาวงจรบนเทคโนโลยีเอฟพีจีเอ

ในการออกแบบวงจรดิจิตอล ผู้ออกแบบวงจรไม่จำเป็นต้องสร้างไอซีที่มีค่าใช้จ่ายสูง เพื่อให้ได้วงจรที่ต้องการ เนื่องจากปัจจุบันมีเทคโนโลยีเอฟพีจีเอ (FPGA) ย่อมาจาก Field Programmable Gate Array ที่สามารถจำลองวงจรที่ออกแบบด้วยภาษา VHDL หรือ Verilog ในการออกแบบวงจรด้วยเอฟพีจีเอนั้นสามารถออกแบบด้วยซอฟต์แวร์ที่ทางผู้ผลิตไอซีพัฒนาขึ้น ตัวอย่างเช่น FPGA ของบริษัท Xilinx ซอฟต์แวร์ที่มีชื่อเรียกว่า ISE และซอฟต์แวร์ที่ช่วยในการจำลองการทำงาน สามารถใช้งานร่วมกับ ISE ก็คือ ModelSim XE ขั้นตอนการออกแบบวงจรบนเทคโนโลยีเอฟพีจีเอ สามารถอธิบายได้ดังภาพประกอบ 3-12



ภาพประกอบ 3-12 ขั้นตอนการออกแบบวงจรบนเทคโนโลยีเอฟพีจีเอ

3.2.2.1 Design Entry

เป็นขั้นตอนแรกในการออกแบบ Concept design ซึ่งสามารถออกแบบได้จากการเขียนภาษาอธิบายการทำงานของฮาร์ดแวร์ (Hardware Description Language, HDL) มีทั้งแบบภาษา VHDL หรือ Verilog HDL นอกจากนี้ยังสามารถใช้การวาดวงจร (Schematic) แทนการอธิบายการทำงานของฮาร์ดแวร์ ในงานวิจัยนี้จะใช้ภาษา VHDL ที่สร้างขึ้นอัตโนมัติจากซอฟต์แวร์ช่วยออกแบบ ImpulseC

3.2.2.2 Simulation

เมื่อได้ทำการออกแบบในขั้นตอนแรก ขั้นตอนต่อไปคือการตรวจสอบ Concept design ในขั้นตอนแรกว่ามีการทำงานถูกต้องหรือไม่ โดยทำการจำลองการทำงาน (Simulation) ซึ่งการจำลองการทำงานจะต้องสร้างโปรแกรมที่สร้างชุดข้อมูลในการทดสอบวงจรที่เรียกว่า Test Bench ซึ่งมีซอฟต์แวร์ HDL Bencher ช่วยในการสร้างชุดข้อมูลทดสอบอัตโนมัติของวงจรที่ออกแบบในขั้นตอนแรก โดยโปรแกรมที่ทำหน้าที่จำลองการทำงานของวงจรที่ออกแบบ ได้แก่ MXE ซึ่งสามารถจำลองการทำงานของวงจรได้ทั้งระดับฟังก์ชันการทำงาน (Functional Simulation) ตลอดถึงระดับไทม์มิ่ง (Timing Simulation) ที่มีค่าล่าช้าของเกต (Gate Delay) และค่าล่าช้าของเส้นเชื่อมต่อต่างๆ (Routing Delay) ของวงจรทั้งหมด

3.2.2.3 Synthesis

เมื่อจำลองการทำงานของวงจรในระดับฟังก์ชันเป็นที่ถูกต้องแล้วจะเข้าสู่ขั้นตอนต่อไป คือนำโค้ดที่เขียนไปสังเคราะห์ (Synthesis) เพื่อสร้างเป็นผังวงจร (Schematic) โดยอาศัยโปรแกรมช่วยในการสังเคราะห์วงจร ได้แก่ ซอฟต์แวร์ XST (Xilinx Synthesis Technology) โดยขั้นตอนตอนดังกล่าวจะต้องเลือกเทคโนโลยีที่ต้องการสร้างให้เป็นวงจร โดยซอฟต์แวร์นี้จะมีอยู่ด้วยกัน 2 เทคโนโลยี คือ CPLD (Complex Programmable Logic Device) หรือ FPGA (Field Programmable Gate Array) ซึ่งแล้วแต่ผู้ออกแบบต้องการเลือกใช้ชิปประเภทใด ซึ่งแตกต่างกันตรงที่ CPLD ทำจาก EEPROM ส่วน FPGA ส่วนใหญ่ทำจาก SRAM ทำให้เวลาโปรแกรมวงจรลงในชิปแล้ว CPLD จะสามารถจดจำค่าที่ถูกโปรแกรมลงไปได้ ส่วน FPGA จะไม่สามารถจดจำวงจรได้เมื่อไม่มีไฟเลี้ยง แต่ข้อดีของ FPGA คือจะมีขนาดหรือความจุของเกตที่ให้ออกแบบมากกว่า CPLD ทำให้สามารถออกแบบวงจรที่ซับซ้อนสูงได้ดีกว่า CPLD ซึ่งในงานวิจัยนี้จะเลือกใช้ชิปประเภท FPGA

3.2.2.4 Implementation

เมื่อผ่านการสังเคราะห์เรียบร้อยแล้ว จะได้ไฟล์โครงสร้างของวงจรที่เรียกว่า เนตลิสท์ (Netlist) ซึ่งเป็นวงจรที่ถูกสังเคราะห์ขึ้นจากเทคโนโลยีที่เลือกใช้ในขั้นตอนการสังเคราะห์ หลังจากนั้นเข้าสู่ขั้นตอนการนำเอาไฟล์เนตลิสท์ที่ได้ทำการ Filter หรือที่เรียกว่า Implementation ลงบนเซลล์ภายในอุปกรณ์ FPGA โดยในขั้นตอนนี้ สามารถกำหนดขาอินพุตและเอาท์พุตของวงจรกับขาอุปกรณ์ FPGA ได้ จะเห็นได้ว่าในการใช้ FPGA ในการออกแบบวงจร สามารถที่จะทำ PCB บอร์ดของวงจรได้พร้อมกับการออกแบบวงจรไปพร้อมๆกันได้เลย โดยไม่ต้องรอให้ออกแบบวงจรเสร็จเสียก่อน ทำให้ลดเวลาในการออกแบบได้มาก ผลลัพธ์ที่ได้จากขั้นตอนนี้ เรียกว่าการทำกระบวนการย้อนกลับ (Back annotate) เพื่อให้ได้ VHDL ในรูปแบบโครงสร้างที่รวมถึงค่า delay ของวงจรจริงบนเอฟพีจีเอ นำไปจำลองการทำงานอีกครั้ง เพื่อเป็นการทดสอบว่าวงจรที่ได้สามารถทำงานได้จริงเมื่อบรรจุลงบนเอฟพีจีเอ

3.2.2.5 Programming

เป็นขั้นตอนสุดท้ายของการออกแบบ คือเป็นการนำไฟล์ที่ผ่านการ Implementation เรียบร้อยแล้วไปโปรแกรมลงสู่ชิป FPGA ที่มี โดยซอฟต์แวร์ชื่อว่า iMPACT ซึ่งเป็นซอฟต์แวร์ที่ใช้ในการดาวน์โหลดวงจรที่ออกแบบลงสู่ชิป FPGA ผ่านทางสาย JTAG Cable ได้โดยตรงไม่ต้องอาศัยเครื่องมืออื่นใด เนื่องจากชิปดังกล่าวมีระบบ ISP (In-System Programming) อยู่ภายใน ทำให้สะดวกในการโปรแกรมและสามารถโปรแกรมซ้ำๆได้ไม่จำกัดจำนวนครั้ง

ในบทนี้ได้แสดงถึงแนวคิดและแสดงขั้นตอนในการออกแบบระบบ ประกอบไปด้วย 2 ส่วนหลักๆ คือ ในส่วนแรกเป็นการพัฒนาระบบให้สามารถประมวลผลสัญญาณภาพด้วยภาษา “ImpulseC” ซึ่งในส่วนนี้ก็จะมีอัลกอริทึมที่ใช้อยู่ 2 อัลกอริทึมหลัก คือ การหาขอบภาพ (Edge Detection), การหาเส้นตรงบนภาพ (Hough Transform) ส่วนที่สองเป็นการนำผลลัพธ์วงจร VHDL จากส่วนแรกมาทำการจำลองการทำงานเพื่อทดสอบฟังก์ชันของระบบ แล้วสังเคราะห์วงจรจริงบนเทคโนโลยีเอฟพีจีเอ สุดท้ายนำวงจรจริงที่ได้ไปทำการจำลองผลอีกครั้งเพื่อยืนยันว่าระบบที่ออกแบบและพัฒนาสามารถทำงานได้ถูกต้อง

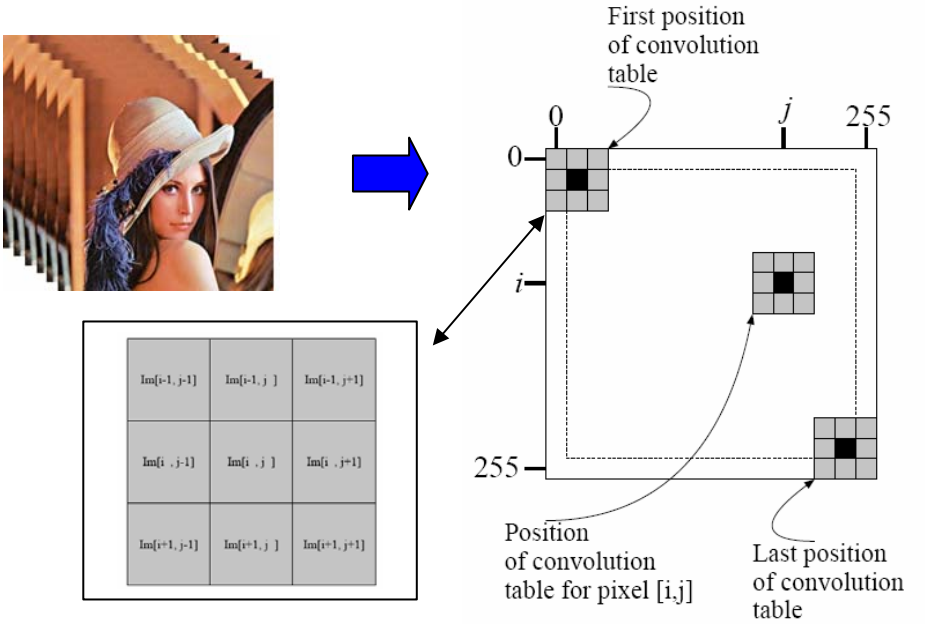
บทที่ 4

ผลและการอภิปรายผลการทดลอง

ในบทนี้อธิบายถึงผลการทดสอบกระบวนการหาขอบภาพด้วยอัลกอริทึมต่างๆด้วยการนำภาพสีเข้ามาทดสอบแล้วพิจารณาว่าให้ผลลัพธ์เป็นอย่างไร และนำอัลกอริทึมที่ดีที่สุดมาเปรียบเทียบกันด้วยวิธีการประมวลผลแบบอนุกรมว่าทรัพยากรที่ใช้ในระบบและความเร็วของการประมวลผลเป็นอย่างไร ผลการทดสอบกระบวนการหาเส้นตรงด้วยเทคนิคของฮัฟ โดยพิจารณาสมการเส้นตรงที่เป็นผลลัพธ์ว่ามีความสัมพันธ์กับภาพต้นแบบหรือไม่ ซึ่งผลการจำลองการทำงานของระบบทั้งหมดจะเป็นไปตามขั้นตอนการออกแบบที่ได้กล่าวไว้ในบทที่ 3

4.1 ผลการทดสอบกระบวนการหาขอบภาพ (Edge Detection)

อัลกอริทึมที่นำมาใช้ในการหาขอบภาพในงานวิจัยชิ้นนี้ ได้แก่ Robert, Sobel และ Prewitt ซึ่งวิธีการทดสอบจะใช้ภาพ RGB ในสกุล .BMP มาเป็นภาพอินพุต ในที่นี้หากนำภาพเข้าประมวลผลอย่างต่อเนื่องก็จะเสมือนเป็นการรับภาพเข้าแบบวีดีโอสตรีม ดังภาพประกอบ 4-1



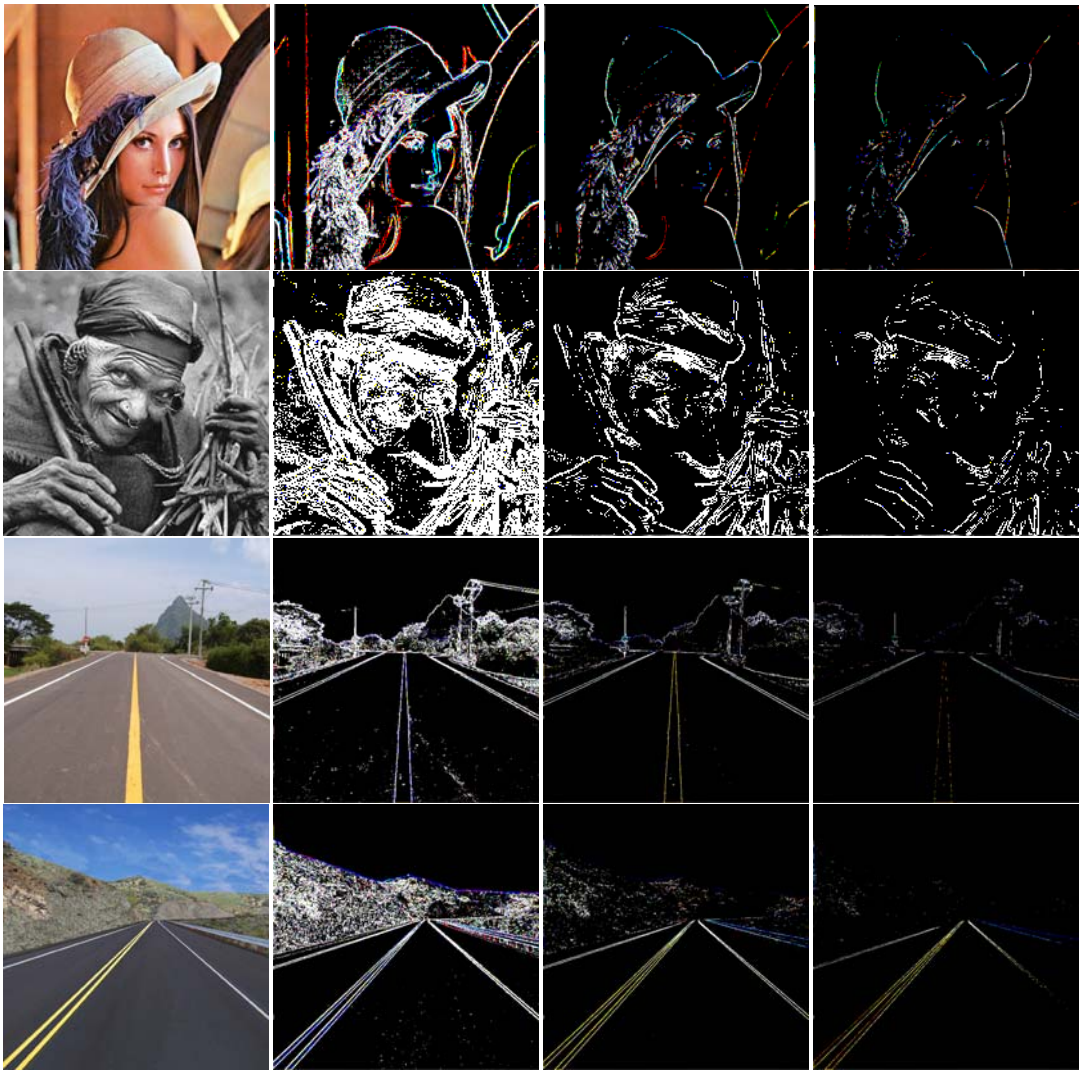
ภาพประกอบ 4-1 การรับภาพอินพุตเข้าทำการคำนวณกับ Convolution Mask

จากภาพประกอบ 4-1 แสดงการนำภาพเข้าประมวลผลด้วยการนำพิกเซลรอบจุดที่สนใจคูณกับ Convolution Mask ณ ตำแหน่งนั้นๆ เพื่อให้ได้ค่าพิกเซลใหม่ขึ้นมา โดยการทำเช่นนี้จนครบทั้งไฟล์ภาพ จะทำให้ได้ข้อมูลภาพใหม่ซึ่งแสดงขอบภาพให้เห็นตาม Convolution Mask ของอัลกอริทึมการหาขอบนั้นๆ

อัลกอริทึม Robert มีโอเปอเรเตอร์ขนาด 2x2 ดังภาพประกอบ 4-2 มีผลลัพธ์ดังภาพประกอบ 4-3

$$Rx = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \qquad Ry = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

ภาพประกอบ 4-2 Robert Operator



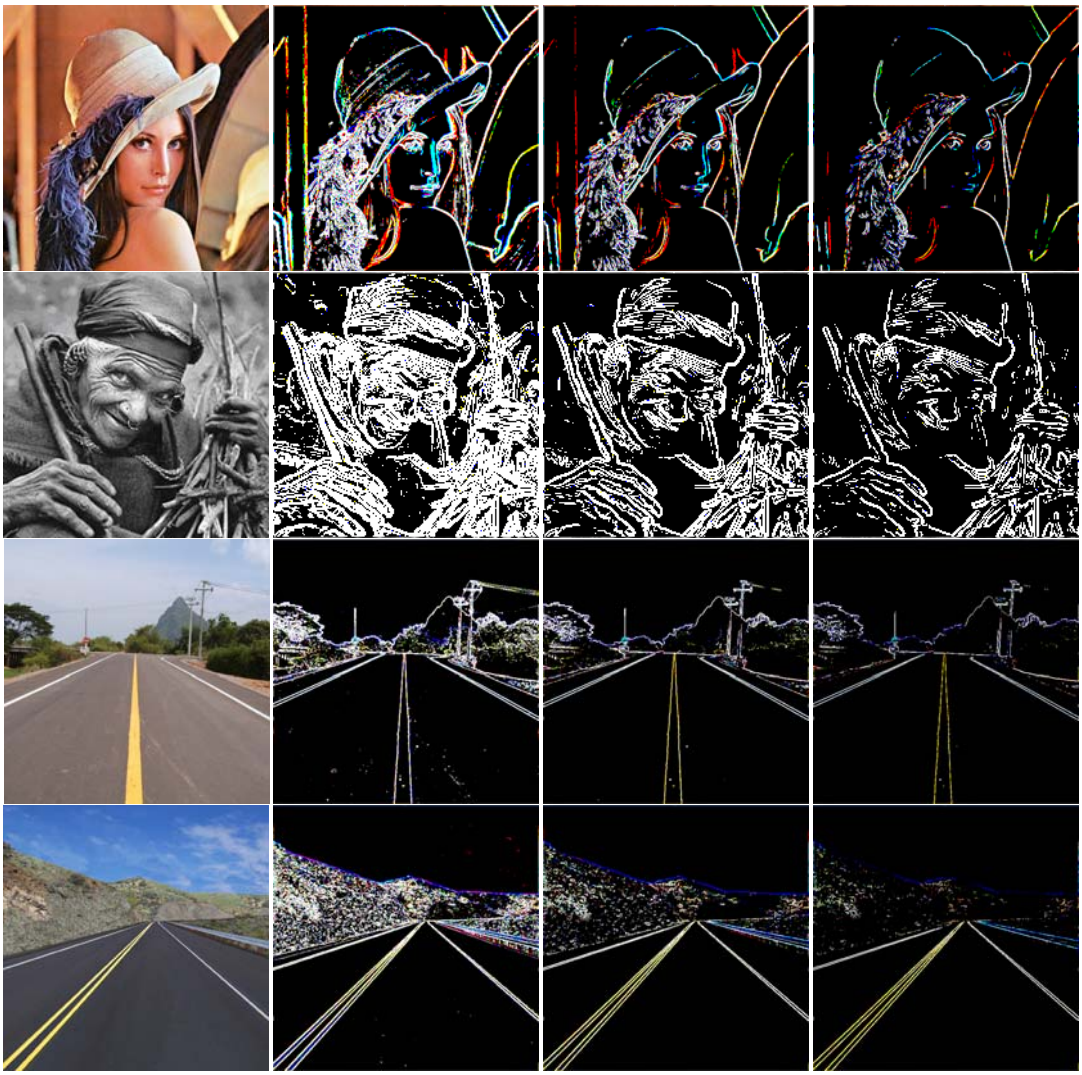
input threshold = 20 threshold = 50 threshold = 80

ภาพประกอบ 4-3 ผลลัพธ์จากอัลกอริทึมของ Robert

ทดลองด้วยวิธีเดียวกันกับอัลกอริทึม Sobel แต่ต่างกันตรงที่อัลกอริทึมนี้มีโอเปอเรเตอร์ขนาด 3x3 ดังภาพประกอบ 4-4 ดังนั้นค่า Threshold ที่ใช้ทดสอบจึงต้องปรับเปลี่ยนให้เหมาะสม โดยยังคงใช้ 3 ระดับเหมือนเดิม ได้ผลลัพธ์ดังภาพประกอบ 4-5

$$Sx = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \qquad Sy = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

ภาพประกอบ 4-4 Sobel Operator



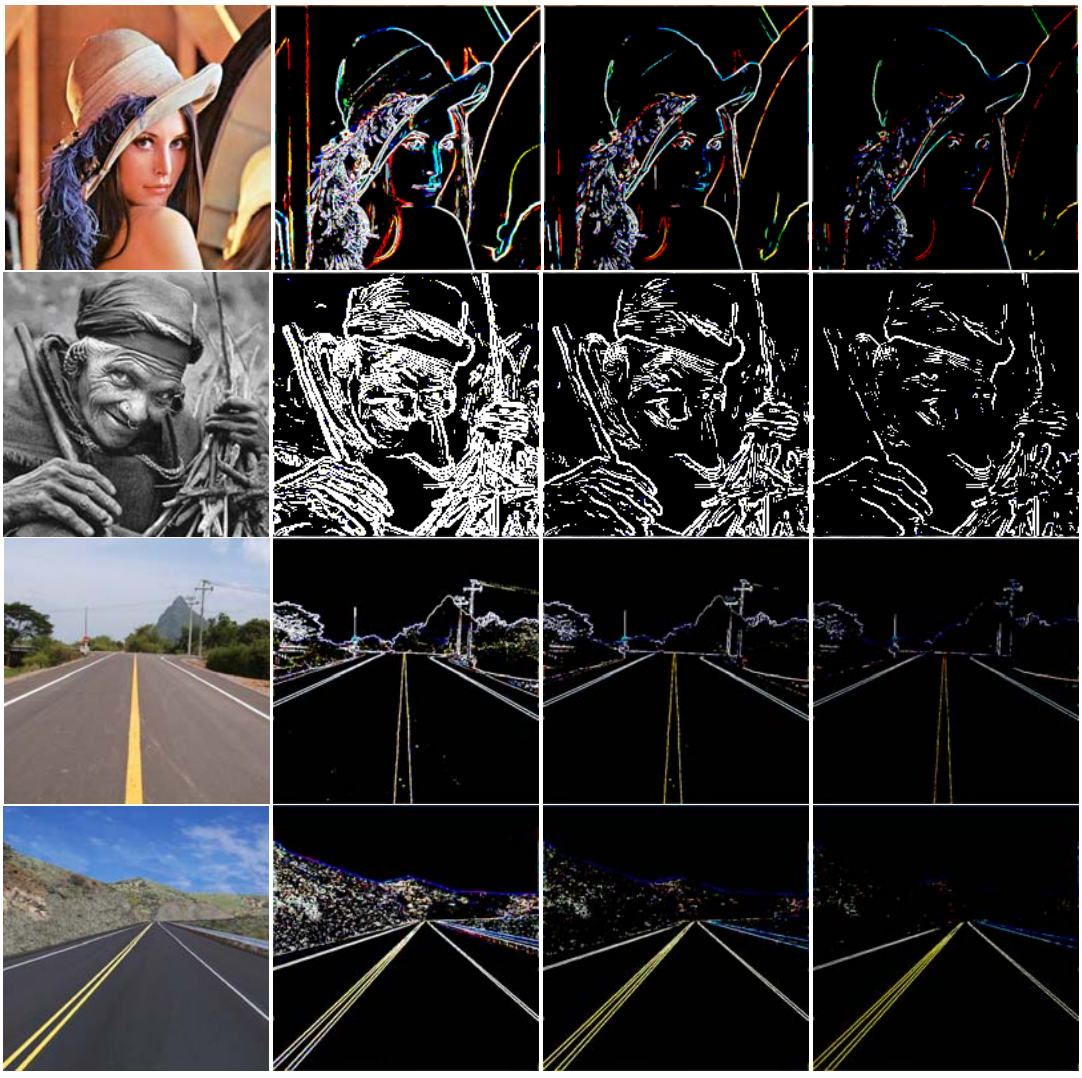
input threshold = 100 threshold = 180 threshold = 260

ภาพประกอบ 4-5 ผลลัพธ์จากอัลกอริทึมของ Sobel

Prewitt เป็นอัลกอริทึมที่มีโอเปอเรเตอร์ขนาด 3x3 เช่นเดียวกันกับ Sobel แต่แตกต่างกันตรงที่ค่าโอเปอเรเตอร์ดังภาพประกอบ 4-6 และได้ผลลัพธ์ดังภาพประกอบ 4-7

$$Px = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad Py = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}$$

ภาพประกอบ 4-6 Prewitt Operator



input threshold = 100 threshold = 180 threshold = 260

ภาพประกอบ 4-7 ผลลัพธ์จากอัลกอริทึมของ Prewitt

จากนั้นนำผลการทดลองของทั้ง 3 อัลกอริทึมมาตรวจสอบหาค่าความคลาดเคลื่อน Root Mean Square Error (RMSE) ซึ่งมีสูตรคำนวณ ดังสมการ 4-1

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(WL_{Observe} - WL_{Forecast})^2} \quad (4\text{-}1)$$

เมื่อ RMSE = ค่าความคลาดเคลื่อนกำลังสอง

$WL_{Observe}$ = ค่าที่ได้จากการตรวจวัด

$WL_{Forecast}$ = ค่าที่ได้จากการทำนาย

N = จำนวนคู่ของข้อมูลที่ใช้ในการวิเคราะห์ทั้งหมด

Root Mean Square Error		
Robert	**Sobel**	**Prewitt**
RMSE = 1.173	RMSE = 1.436	RMSE = 1.250

ตาราง 4-1 เปรียบเทียบค่าความคลาดเคลื่อนจากผลลัพธ์ของอัลกอริทึม Robert, Sobel และ Prewitt ด้วย Root Mean Square Error (RMSE)

จากตาราง 4-1 พบว่าค่าความคลาดเคลื่อนของทั้ง 3 อัลกอริทึมมีค่าที่ใกล้เคียงกันมาก ซึ่งในงานวิจัยนี้ใช้ภาพถนนที่มีเส้นกึ่งกลางถนนเป็นภาพหลักในการทดสอบ ดังนั้นจึงนำผลลัพธ์ของทั้ง 3 อัลกอริทึมนี้ไปเปรียบเทียบกันด้วยตัวชี้วัดอื่นๆอีก เช่น ความเร็วที่ใช้ในการประมวลผล และทรัพยากรที่ระบบต้องการใช้งาน ด้วยการนำโปรแกรมการหาขอบด้วยอัลกอริทึมทั้ง 3 ไปจำลองการทำงานบนเทคโนโลยีเอฟพีจีเอด้วยซอฟต์แวร์ Xilinx ISE เพื่อเปรียบเทียบผลลัพธ์ของแต่ละอัลกอริทึม ดังตาราง 4-2

- จำลองการทำงานบนบอร์ดเอฟพีจีเอ Spartan3E (XC3S1600E)

Algorithm / Resources	Robert	Sobel	Prewitt
Number of External IOBs	**22.8%**	**22.8%**	**22.8%**
Number of BUFMUXs	**8.3%**	**8.3%**	**8.3%**
Number of MULT18X18SIOs	**25.0%**	36.1%	36.1%
Number of RAMB16s	**13.8%**	16.6%	16.6%
Number of Slices	**9%**	10.8%	10.9%
Number of SLICEMs	**0.8%**	**0.8%**	**0.8%**
Clock net clk_BUFGP(ns)	**24.037**	32.840	33.650

ตาราง 4-2 เปรียบเทียบผลลัพธ์ของอัลกอริทึม Robert, Sobel และ Prewitt ด้วยขั้นตอนการพัฒนาวงจรบนเทคโนโลยีเอฟพีจีเอด้วยบอร์ด Spartan3E (XC3S1600E)

จากตาราง 4-2 พิจาณาจากทรัพยากรที่ใช้ของแต่ละอัลกอริทึม ทำให้สามารถเลือกได้ว่าอัลกอริทึม Robert มีความเหมาะสมสำหรับงานวิจัยชิ้นนี้ เพราะใช้เวลาและทรัพยากรในการประมวลผลน้อยกว่าอัลกอริทึมแบบอื่นๆ

4.2 ผลการทดสอบกระบวนการหาเส้นตรงบนภาพ (Hough Transform)

ในส่วนนี้เป็นผลการทดสอบกระบวนการหาเส้นตรงบนภาพด้วยเทคนิคฮัฟ จะเน้นไปที่ภาพซึ่งจะประกอบไปด้วยเส้นกึ่งกลางถนนเป็นหลัก เพื่อการทดสอบผลลัพธ์ที่ได้กับภาพอินพุตว่ามีความถูกต้อง และสามารถที่จะนำไปใช้งานได้จริง

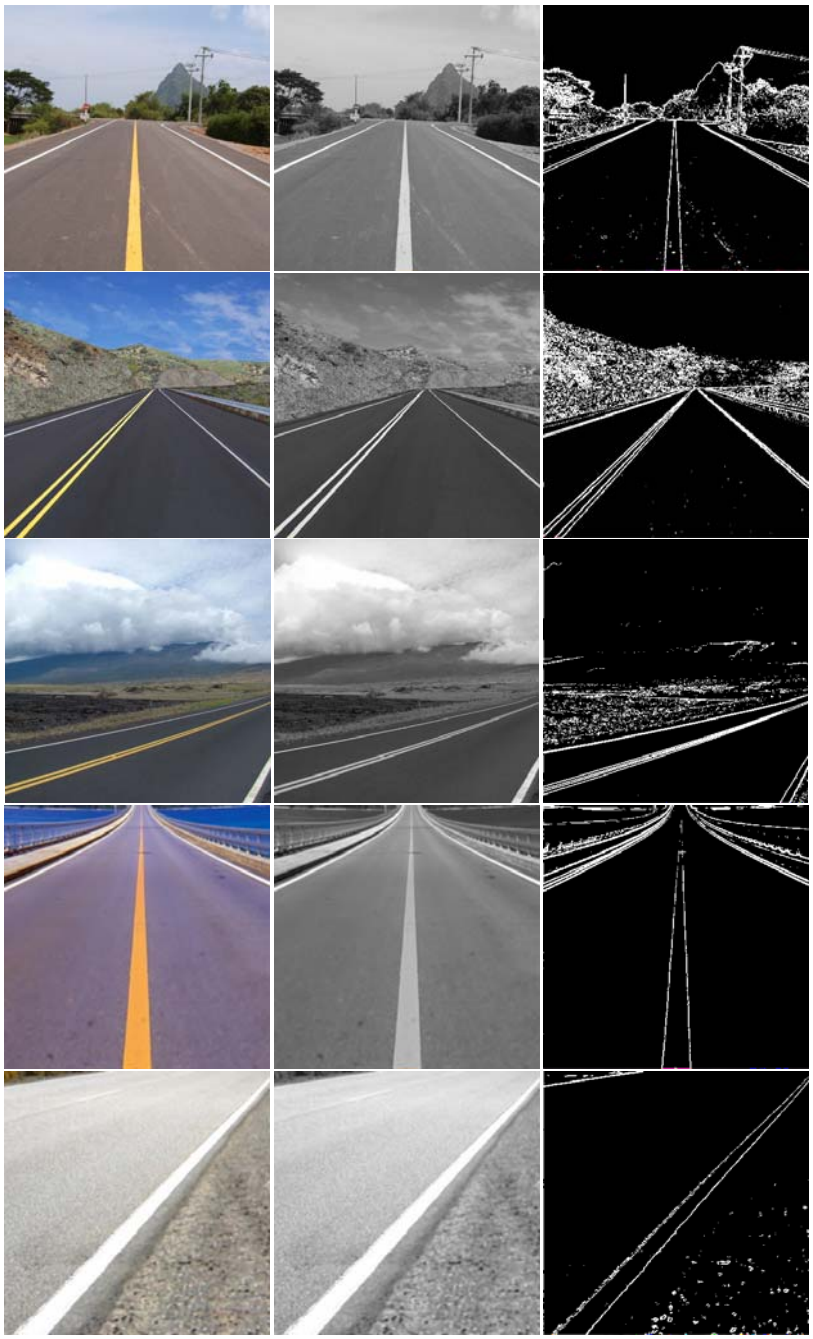
4.2.1 Gray Scale Process

เนื่องจากการหาเส้นตรงด้วยวิธีฮัฟนั้นต้องการข้อมูลขนาด 8 บิต/พิกเซล ก็เพียงพอแล้ว แต่ภาพอินพุตที่รับเข้ามาเป็นภาพสี (RGB) ขนาด 24 บิต/พิกเซล ดังนั้นเพื่อเพิ่มความเร็วในการประมวลผลให้กับระบบที่ออกแบบ จึงใช้วิธีการแปลงภาพสีเป็นภาพระดับสีเทาก่อนเป็นอันดับแรกดังภาพประกอบ 3-4 และได้ผลลัพธ์ ดังภาพประกอบ 4-8

ภาพประกอบ 4-8 ผลลัพธ์จาก Gray Scale Process

4.2.2 Edge Detection Process

ในที่นี้จะขอใช้ภาพถนนหลายๆรูปแบบซึ่งมีเส้นกึ่งกลางถนนในลักษณะที่เป็นเส้นตรงอย่างเดียว(ไม่รวมถึงเส้นโค้ง และเส้นประ) ซึ่งอยู่ภายในขอบเขตของงานวิจัยนี้ ดังภาพประกอบ 4-9

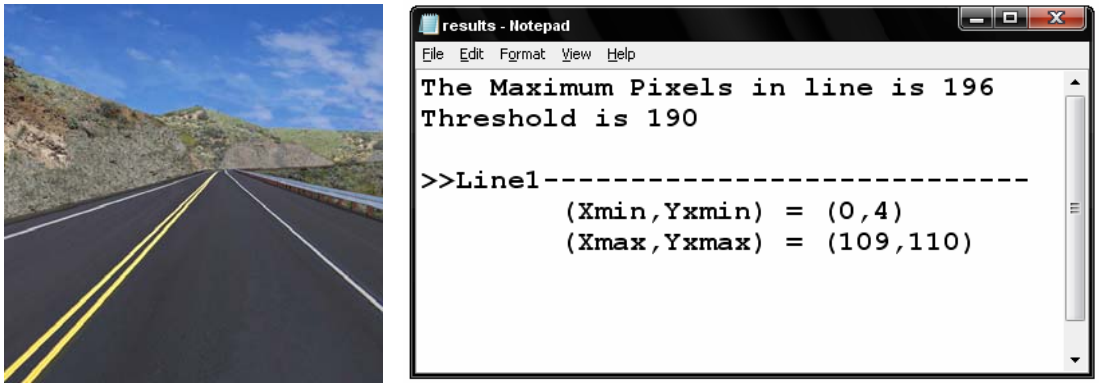


ภาพประกอบ 4-9 ผลลัพธ์จาก Edge Detection Process

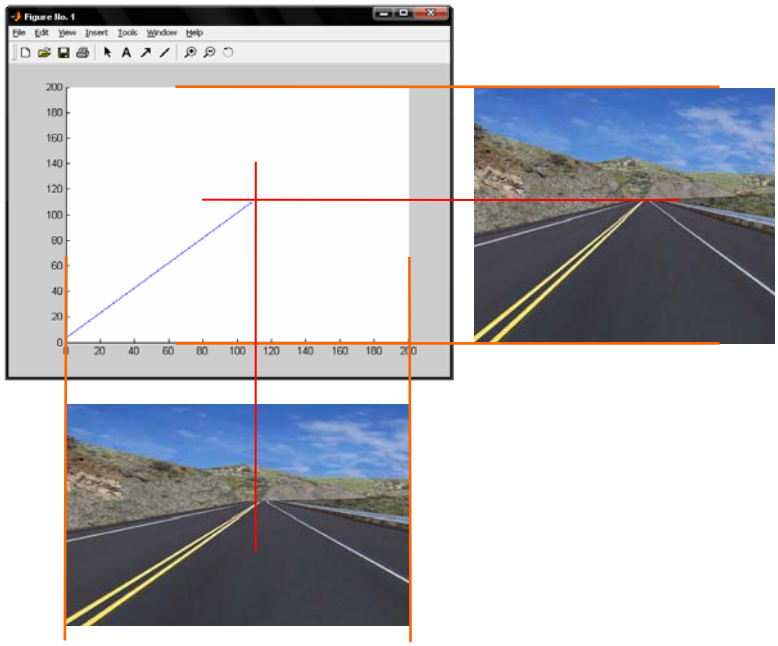
4.2.3 Hough Transform Process

ในส่วนนี้เป็นกระบวนการหาเส้นตรงของภาพ โดยรับอินพุตจากกระบวนการหาขอบภาพมาประมวลผลหาสมการเส้นตรง ผลลัพธ์ที่ได้นั้นต้องการออกมาเป็นค่าพิกัดจุดเริ่มต้นและจุดสิ้นสุดของเส้นตรงต่างๆบนภาพตามค่า Threshold ที่กำหนด

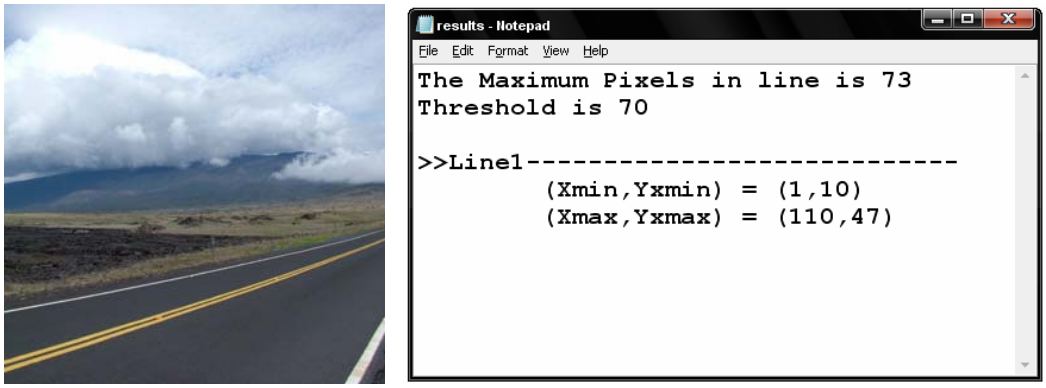
เนื่องจากกระบวนการทดสอบความถูกต้องของภาพอินพุตมีวิธีการเหมือนกัน จึงขอนำภาพบางส่วนจากภาพอินพุตมาทดสอบค่าสมการเส้นตรงที่ได้ โดยมีผลลัพธ์ของกระบวนการหาเส้นตรงของภาพ และการตรวจสอบผลลัพธ์ของสมการเส้นตรง ดังภาพประกอบ 4-10 และ 4-11



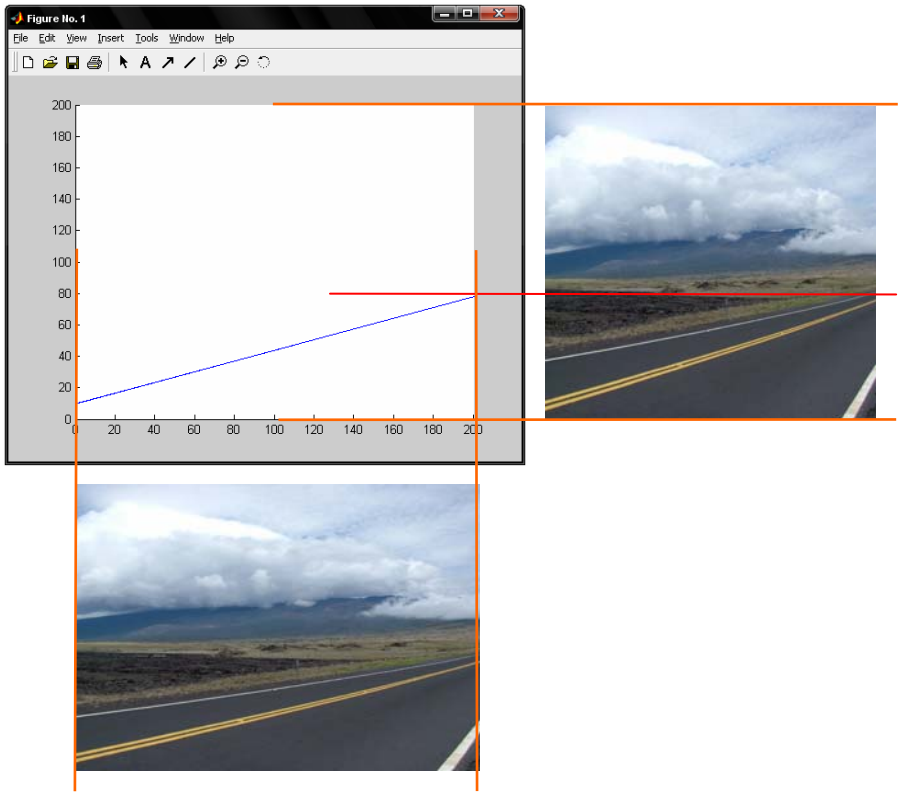
ภาพประกอบ 4-10 ผลลัพธ์ของภาพเส้นถนนจาก Hough Transform Process



ภาพประกอบ 4-11 ตรวจสอบผลลัพธ์สมการของรูปภาพเส้นถนนด้วย Matlab 7.0



ภาพประกอบ 4-12 ผลลัพธ์ของภาพเส้นถนนจาก Hough Transform Process



ภาพประกอบ 4-13 ตรวจสอบผลลัพธ์สมการของภาพเส้นถนนด้วย Matlab 7.0

จากผลการตรวจสอบผลลัพธ์ของสมการเส้นตรงที่ได้ด้วยซอฟต์แวร์ Matlab 7.0 ทำให้เห็นว่าสมการที่ได้มีความถูกต้องอยู่ในเกณฑ์ที่ดี

จากนั้นได้นำวงจร HDL ที่สร้างได้จากซอฟต์แวร์ Impulse CoDeveloper ไปจำลองการทำงานบนเอฟพีจีเอด้วยซอฟต์แวร์ Xilinx ISE เพื่อวิเคราะห์ผลลัพธ์ที่ได้ ดังตาราง 4-3 โดยในการทดสอบได้ใช้ภาพ RGB ในสกุล bmp ขนาด 200x200 พิกเซล เป็นภาพอินพุต

FPGA board / **Resources**	**Spartan3E (XC3S1600E)**	
Number of External IOBs	40 out of 250	16.0%
Number of BUFGMUXs	1 out of 24	4.2%
Number of MULT18X18SIOs	28 out of 36	77.7%
Number of RAMB16s	3 out of 36	8.3%
Number of Slices	9448 out of 14752	64.0%
Number of SLICEMs	3850 out of 7376	52.2%
Clock net clk_BUFGP	34.507 ns	

ตาราง 4-3 ผลลัพธ์จากขั้นตอน
การพัฒนาวงจรบนเทคโนโลยีเอฟพีจีเอด้วยบอร์ด Spartan3E (XC3S1600E)

จากตาราง 4-3 พบว่า เวลาในส่วนของฮาร์ดแวร์ที่ใช้ประมวลผล 1 พิกเซลนั้นใช้เวลาเท่ากับ 34.507 ns ซึ่งในการทดสอบใช้ภาพขนาด 200x200 พิกเซล เพราะฉะนั้นเวลาทั้งหมดในส่วนของฮาร์ดแวร์ที่ใช้ประมวลผลเท่ากับ 40,000x34.507 = 1,380,280 ns ≈1.4 ms และเวลาในส่วนของซอฟต์แวร์ที่ใช้ประมวลผลจะขึ้นอยู่กับความเร็วของ Core ที่ใช้ ในที่นี้จำลองการทำงานด้วย Core PowerPC 400 MHz โดยโปรแกรมมีคำสั่งประมาณ 900,000 โอเปอร์เรชัน เพราะฉะนั้นเวลาประมวลผลในส่วนของซอฟต์แวร์เท่ากับ 900,000/400MHz ≈ 2.25 ms ดังนั้นเวลารวมที่ใช้ในการประมวลผลทั้งหมดเท่ากับ เวลาในส่วนของฮาร์ดแวร์ + เวลาในส่วนของซอฟต์แวร์ และนำมาเปรียบเทียบกับความเร็วในการประมวลผลบนคอมพิวเตอร์ทั่วไป ได้ผลลัพธ์ดังตาราง 4-4

ความเร็วในการประมวลผลเพื่อหาเส้นตรงในภาพ	
การออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์ FPGA board Spartan3E (XC3S1600E)	คอมพิวเตอร์ส่วนบุคคล AMD Athlon(TM)64 Processor 3000+ 1.81 GHz RAM 1.50 GB
Hw + Sw 1.4 ms + 2.25 ms ≈ 4 ms	2.359 s

ตาราง 4-4 แสดงการเปรียบเทียบความเร็วในการประมวลผลเพื่อหาเส้นตรงในภาพระหว่างการออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์กับคอมพิวเตอร์ส่วนบุคคล

ในบทนี้แสดงถึงผลการทดสอบกระบวนการหาขอบภาพและเส้นตรงบนภาพ ตามวิธีดำเนินการวิจัยที่ได้กล่าวไว้ในบทที่ 3 ซึ่งจากการทดสอบพบว่าระบบสามารถจำลองการทำงานบนบอร์ดเอฟพีจีเอและพร้อมที่จะนำไปใช้งานได้จริง

บทที่ 5

บทสรุปและข้อเสนอแนะ

ในบทนี้จะกล่าวสรุปผลการวิจัยที่ได้ดำเนินการสำหรับวิทยานิพนธ์นี้ รวมทั้งข้อเสนอแนะต่างๆ ที่จะเป็นประโยชน์ต่อการทำวิจัยด้านการออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์และเอฟพีจีเอ

5.1 บทสรุป

วิทยานิพนธ์ฉบับนี้ได้นำเสนอการออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์สำหรับการประมวลผลภาพเพื่อหาเส้นตรง โดยใช้เทคโนโลยีเอฟพีจีเอเข้ามาเป็นตัวช่วยในการประมวลผลร่วมกับหน่วยประมวลผลบนบอร์ดสมองกลฝังตัว

จากผลการทดสอบกระบวนการหาขอบภาพซึ่งนำอัลกอริทึม 3 แบบ ได้แก่ Robert, Sobel และ Prewitt มาทดสอบประมวลผลด้วยโครงสร้างแบบเดี่ยว พบว่าค่าความคลาดเคลื่อน (RMSE) ของผลลัพธ์ขอบภาพที่ได้มีค่าใกล้เคียงกันมาก ดังนั้นจึงนำผลลัพธ์ทั้ง 3 ไปพิจารณาต่อในขั้นตอนการพัฒนาวงจรบนเทคโนโลยีเอฟพีจีเอ พบว่า Robert Edge Detection มีความเหมาะสมสำหรับงานวิจัยชิ้นนี้ เพราะใช้เวลาและทรัพยากรในการประมวลผลน้อยกว่าอัลกอริทึมแบบอื่นๆ

จากนั้นนำกระบวนการหาขอบภาพ (Edge Detection) มาใช้งานร่วมกับกระบวนการหาเส้นตรงบนภาพ (Hough Transform) และทดสอบด้วยการใช้ภาพตัวอย่างถนนในลักษณะต่างๆ มาเป็นภาพอินพุต ผลลัพธ์ที่ได้จากกระบวนการหาเส้นตรงบนภาพ คือ พิกัด (X,Y) ของจุดตั้งต้นและจุดสิ้นสุดของเส้นตรงต่างๆที่ตรวจพบบนภาพ ซึ่งสามารถนำพิกัดเหล่านี้ไปสร้างเป็นสมการเส้นตรง นำพิกัดเหล่านี้ไปตรวจสอบความถูกต้องด้วยซอฟต์แวร์ Matlab 7.0 พบว่าพิกัดที่นำไปพล็อตกราฟทดสอบนั้นมีความถูกต้องแม่นยำอยู่ในเกณฑ์ที่ดี สามารถนำไปใช้งานได้

สุดท้ายนำวงจร HDL ที่สร้างขึ้นมาจากซอฟต์แวร์ ImpulseC ไปทดสอบด้วยขั้นตอนการพัฒนาวงจรบนเทคโนโลยีเอฟพีจีเอ เพื่อทำการจำลองผลการทำงานบนบอร์ดเอฟพีจีเอ ในงานวิจัยชิ้นนี้ใช้สิ่งแวดล้อมการจำลองการทำงานบนบอร์ด Spartan3E เป็นบอร์ดในการใช้ทดสอบ พบว่าสามารถจำลองผลการทำงานบนบอร์ดได้จริง และจากตาราง 4-4 พบว่าเวลาที่ใช้ในการ

ประมวลผลหาเส้นตรงในภาพจากการออกแบบร่วมฮาร์ดแวร์/ซอฟต์แวร์น้อยกว่าการประมวลผลบนคอมพิวเตอร์ส่วนบุคคลทั่วไป

5.2 ข้อเสนอแนะ

1. ในงานวิจัยนี้ ช่วยเพิ่มประสิทธิภาพการประมวลผลภาพด้วยการนำเทคนิคการออกแบบร่วมฮาร์ดแวร์และซอฟต์แวร์ ทำให้ใช้เวลาในการพัฒนาระบบการประมวลผลภาพรวดเร็วขึ้น

2. ในงานวิจัยนี้ ใช้ซอฟต์แวร์ที่ชื่อว่า “ Impulse CoDeveloper “ ในการพัฒนาโปรแกรม เพราะซอฟต์แวร์ตัวนี้มีคุณสมบัติเด่นคือสามารถแปลงภาษาซีเป็นภาษา VHDL หรือ Verilog เพื่อให้สามารถนำไปใช้งานบนบอร์ดเอฟพีจีเอได้จริง แต่ทั้งนี้ขึ้นกับขนาดของวงจรที่ได้จะต้องไม่มีขนาดใหญ่เกินความสามารถของเอฟพีจีเอ

3. ในงานวิจัยนี้ ใช้เทคนิคฮัฟในการหาเฉพาะเส้นตรงเท่านั้น แต่เทคนิคนี้สามารถนำไปประยุกต์หาวงกลม วงรี ฯลฯ ซึ่งขึ้นอยู่กับความต้องการของผู้ใช้

4. นักวิจัยที่สนใจสามารถนำระบบการหาขอบภาพและสมการเส้นตรงทั้งในรูปแบบของโค้ดภาษา ImpulseC หรือภาษา HDL ไปใช้งานกับการประมวลผลภาพอื่นๆได้

เอกสารอ้างอิง

[1] ดร.เกียรติศักดิ์ ศรีพิมานวัฒน์, วุฒิกรณ์ ตรัยศิลานันท์, พรวิภา ทัศยาพันธุ์, “การสำรวจการประมวลผลภาพและการใช้รหัสเพื่อควบคุมความผิดพลาด”, http://www.kmitl.ac.th/dslabs/ksripima/readme/49/technical_Report/intern_49/Pornwipa/Report%20Pornvipha%20Tassayaphan.pdf

[2] วศิน สินธุภิญโญ, “Digital Image Processing and Digital Signal Processing”, กิจกรรมการบรรยายพิเศษทางด้านวิทยาศาสตร์ คณิตศาสตร์ และเทคโนโลยี, 8 มกราคม 2550

[3] Japan System House Association, “เทคโนโลยีสมองกลฝังตัว Embedded Technology”, 1/2549

[4] http://www.impulsec.com

[5] http://www.prc.ac.th/newart/webart/colour08.html

[6] http://multimedia.udru.ac.th/homecs1/konlaphan/rgb.html

[7] http://www.bobpowell.net/grayscale.htm

[8] http://dx.sheridan.com/advisor/cmyk_color.html

[9] J.D. Foley, A.V. Dam, S.K. Feiner, and J.F. Hughes., “Computer Graphics Principles and Practice 2nd ed.”, USA, Addison Wesley, 1996 http://www.colourtherapyhealing.com/colour/primary_colours.php

[10] R.C. Gonzalez, and R.E. Woods., “Digital Image Processing”, USA, Addison Wesley. Lowell January 2006

[11] http://en.wikipedia.org/wiki/Grayscale

[12] http://www.pages.drexel.edu/~weg22/edge.html

[13] Ellen C. Hildreth, “Theory of edge detection”, Journal, 1980, vol. 207 : 187-217

[14] I. Sobel. “An isotropic 3x3 image gradient operator”. In H. Freeman, editor, “Machine Vision for Three-Dimensional Scenes”, Academic Press, 1990 : 376-379

[15] J.M.S. Prewitt. “Object enhancement and extraction”. In B.S. Lipkin and A. Rosenfeld, editors, “Picture Processing and Psychopictorics”. Academic Press, 1970.

[16] http://web.en.rmutt.ac.th/cp/Project/p_27/chapter2.htm#7

[17] วิโรจน์ องอาจ, “การประมวลผลภาพวัดขนาดเส้นผ่านศูนย์กลางเส้นใยโพลิเมอร์จากการปั่นด้วยไฟฟ้าสถิต”, วิศวกรรมไฟฟ้า สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ, ปีการศึกษา 2549

[18] Duda R.O. and Hart P.E. “Use Hough transformation to detect lines and curves in pictures.” Commun. Ass. Comput. (1972): 465-468

[19] Hamarneh G., Althoff K. and Abu-Gharbieh R. Automatic Line Detection. 1999.

[20] ธีรยศ เวียงทอง, “รู้จักกับการออกแบบร่วมกันระหว่างฮาร์ดแวร์-ซอฟต์แวร์เบื้องต้น”, ภาควิชาวิศวกรรมอิเล็กทรอนิกส์ มหาวิทยาลัยเทคโนโลยีมหานคร

[21] Bruce A. Draper, J. Ross Beveridge, A.P. Willem Bohm, Charles Ross, Monica Chawathe, Jeffrey Hammes, “Accelerated Image Processing on FPGAs ”, IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL.12, NO.12, DECEMBER 2003

[22] Paula J. Pingree, Lucas J. Scharenbroich, Thomas A. Wenne and David Pellerin, “Developing FPGA Coprocessors for Performance-Accelerated Spacecraft Image Processing” *Xcell Journal,* 2nd quarter, 2008 : 23-26

[23] David Pellerin, “Combining Impulse C™ with uClinux™ for MicroBlaze™-based FPGAs”, CTO, Impulse Accelerated Technologies, Inc.

[24] Srikanth Rangarajan and RC Gonzalez and RE Woods, “Algorithms For Edge Detection”, Prentice-Hall, 2002

[25] http://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm

[26] Kye Wook Lee, “Application of The Hough Transform”, University of Massachusetts, Lowell, January 2006

[27] http://www.nrru.ac.th/preelearning/songsak/couse6.htm

ภาคผนวก

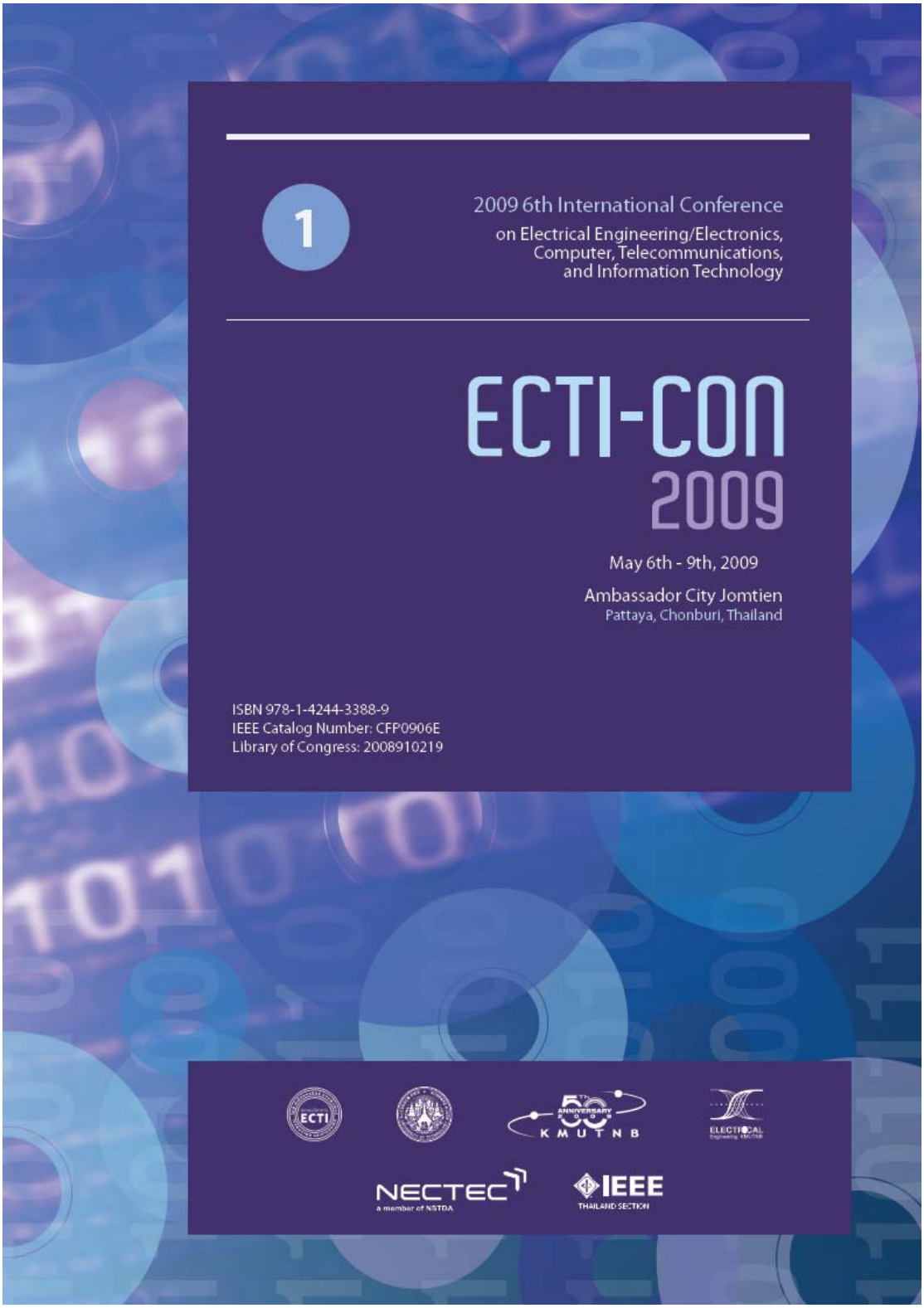
ภาคผนวก ก.

รายละเอียดของโปรแกรมและซอฟต์แวร์ช่วยออกแบบ

รายละเอียดของโปรแกรมและซอฟต์แวร์ช่วยออกแบบ
อยู่ในซีดีที่แนบมากับวิทยานิพนธ์ท้ายเล่ม

ภาคผนวก ข.

ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์


1
2009 6th International Conference
on Electrical Engineering/Electronics,
Computer, Telecommunications,
and Information Technology
ECTI-CON
2009
May 6th - 9th, 2009
Ambassador City Jomtien
Pattaya, Chonburi, Thailand
ISBN 978-1-4244-3388-9
IEEE Catalog Number: CFP0906E
Library of Congress: 2008910219
ECTI
KMUTNB
NECTEC
a member of NSTDA
IEEE
THAILAND SECTION

Hardware/Software Co-design for Line Detection Algorithm on FPGA

W. KayankitW. Suntiamorntut
Department of Computer Engineering, Faculty of Engineering
Prince of Songkla University, Hatyai Songkhla 90112 Thailand
{warut.k@hotmail.com, wannarat@coe.psu.ac.th}

***Abstract*- This paper presents the design methodology of hardware/software co-design on FPGA. In this research work uses line detection algorithm to be our case study. C-based programming named ImpulseC has been employed. By using this, we could reduce the time of the design. Two main components are involved. Edge detection module is the first component. Various algorithms, Robert, Sobel and Prewitt, are compared. The best algorithm in term of performance and area is selected. The second component is hough transform. This module is divided into hardware and software due to the limit of the available logic block. At the end, the system both in hardware/software for line detection on FPGA is simulated.**

I. INTRODUCTION

FPGA-based reconfigurable computers have become viable computing device. It could compute as super-computer by performing both in directly on FPGA hardware and on processor[1][2][3]. In [4], they present the operating system designed specially for reconfigurable computers. The concept of hardware process designed on FPGA hardware is introduced and replaced on software program. The OS kernel employs to maintain the hardware/software interface.

The line detection was mainly performed on software due to its large hardware requirement. By using hardware/software co-design, line detection system could be implemented on FPGA.

In this paper a FPGA implementation of the line detection has proposed. The choice of Robert edge detection is selected among Robert, Sobel and Prewitt, due to its low area and high speed. While a Hough transform will be divided into software and hardware process. Software platform uses ISE Xilinx 8.2i with simulation on ModelsimSE6.2.

II. EDGE DETECTION[5] AND HOUGH TRANSFORM

The line detection contains two major image processing routine. One is edge detection. Another is hough transform.

A. Robert edge detection

The Roberts Cross operator performs a simple, quick to compute, 2-D spatial gradient measurement on an image. The operator consists of a pair of 2×2 convolution kernels as shown in Figure 1. One kernel is simply the other rotated by 90°. This is very similar to the Sobel operator. The input to the operator commonly uses a grayscale image.

+1	0
0	-1

Gx

0	+1
-1	0

Gy

Figure 1 2×2 convolution kernels in Robert

The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

although typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy|$$

which is much faster to compute.

The angle of orientation of the edge giving rise to the spatial gradient (relative to the pixel grid orientation) is given by:

$$\theta = \arctan(Gy / Gx) - 3\pi / 4$$

B. Sobel edge detection

The operator consists of a pair of 3×3 convolution kernels as shown in Figure 2. One kernel is simply the other rotated by 90°. Sobel filter is a simple approximation to the concept of gradient with smoothing.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Figure 2 3×3 convolution kernels in Sobel

The gradient magnitude is given by:

$$|G| = \sqrt{Gx^2 + Gy^2}$$

Typically, an approximate magnitude is computed using:

$$|G| = |Gx| + |Gy|$$

which is much faster to compute.

The angle of orientation of the edge (relative to the pixel grid)

giving rise to the spatial gradient is given by:

$$\theta = \arctan(Gy / Gx)$$

C. *Prewitt edge detection*

Prewitt filter is very similar to Sobel filter. The difference with respect to Sobel filter is the spectral response. It is only suitable for well-contrasted noiseless image. Prewitt operator is used for detecting vertical and horizontal edges in images as shown in Figure 3.

+1	+1	+1
0	0	0
-1	-1	-1

Gx

-1	0	+1
-1	0	+1
-1	0	+1

Gy

Figure 3 Prewitt mask

D. *Hough transform*

The Hough Transform is a technique which can be used to isolate features of a particular shape within an image. Because it requires that the desired features be specified in some parametric form, the classical Hough Transform is commonly used for the detection of regular curves such as lines, circles, ellipses, etc.

The Hough Transform is an algorithm that can provide a significant solution to the problems associated with line detection and definition. It is defined by the parametric representation used to describe lines in the picture plane. It was introduced by Paul Hough in 1962 and patented by IBM. The transform parameterizes a description of a feature at any given location in the original image space. Hough used slope-intercept parameters exclusively.

Hough transform is commonly used to establish whether pixels line on a curve of a specific shape. A point with coordinates (xi,yi) can have an infinite number of straight line passing through it. These line can be described in polar coordinates in the (ρ,θ) space by the equation:

$$\rho = xi \cdot \sin\theta + yi \cdot \cos\theta$$

III. SYSTEM DEVELOPMENT BY IMPULSEC

E. *ImpuseC*

With the advantage of ImpulseC based FPGA design tools, the complete hardware/software design is easily compiled, simulated or debugged with a single tool set named ImpulseC CoDeveloper as shown in figure4.

The ImpulseC tools include a software-to-hardware compiler that converts individual ImpulseC processes to functionally equivalent hardware descriptions and that generates the necessary process-to-process interface logic.

C language applications, image processing algorithms are compiled and optimized to FPGA hardware represented by hardware description language (HDL). Compiler also generates hardware/software interfacing and the link between host processor and FPGA hardware.

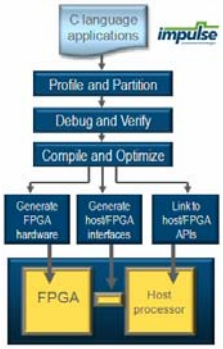


Figure 4 ImpulseC CoDeveloper

The standard C language does not contain parallel processing and the programming of parallel systems. The parallelism can be exploited at two distinct levels: at the application system level and the level of statements within a specific subroutine or loop.

ImpulseC programming model are processes and streams. Processes are independently synchronized. This makes it possible to create high-performance, mixed hardware/software applications for FPGA-based platforms without the need to write low-level VHDL or Verilog.

F. *Hardware/Software co-design process*

There are many different methods to design and implement hardware and software co-design. This research chooses ImpulseC design process.

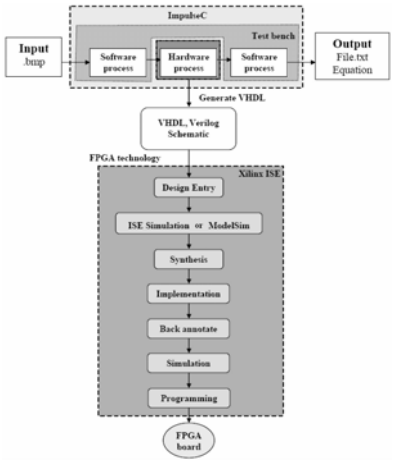


Figure 5 Hardware/Software co-design process

As shown in Figure5, we could separate the design process to be two sections: the system developed by ImpulseC and hardware implementation on FPGA.

IV. EXPERIMENTAL RESULTS

The edge detection and hough transform algorithm were modeled in ImpulseC on Spartan3E16000 based platform. The bmp file format is brought to the system as the input file. Then later, the image file is transformed into the gray scale image.

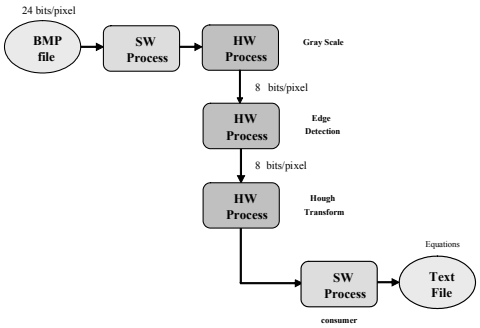


Figure 6 Line detection design process using ImpulseC

When the gray scale image is completed, edge detection is activated. After the edge detection process is done in one row, the result will forwarded directly to hough transform process. This technique will increase the throughput of line detection system. However, we cannot implement sine or cosine function onto the hardware because it consumes a large area of memory. Therefore, this function will be implemented on software process instead. The final output is generated as a text file showing the (Xmin, Ymin) and (Xmax, Ymax) results.

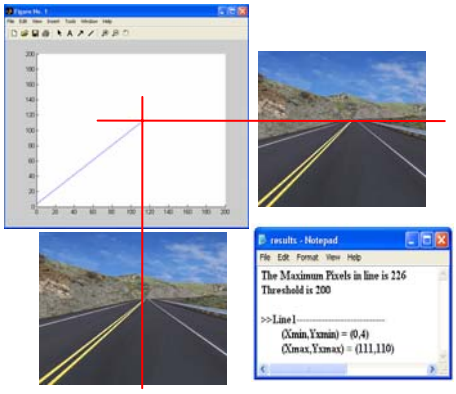


Figure 7 Line detection result compared to Mathlab7.0

The FPGA resource usage from simulation is presented in Table1. We found that the line detection algorithm could be solved within a second.

FPGA board / Resources	Spartan3E (XC3S1600E)
Number of External IOBs	16.0%
Number of BUFGMUXs	4.2%
Number of MULT18X18SIOs	77.7%
Number of RAMB16s	8.3%
Number of Slices	64.0%
Number of SLICEMs	52.2%
Clock net clk_BUFGP	34.507 ns

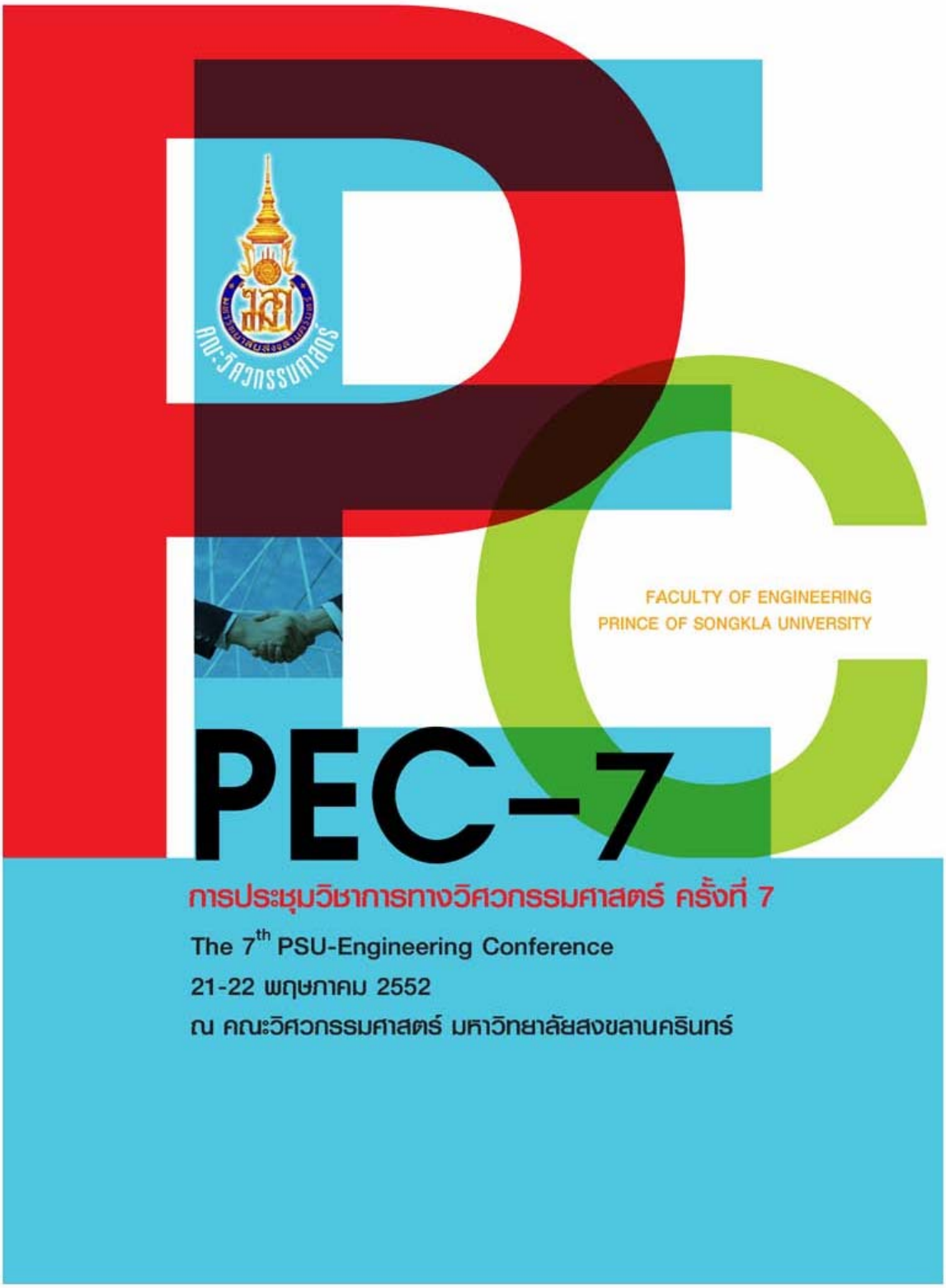
TABLE I FPGA resources for line detection

V. CONCLUSION AND FUTURE WORK

The hardware/software co-design is useful and reasonable to improve the performance of intensive computing such as image processing. The line detection is the case study of using C-language based programming to implement the hardware without the logic design knowledge.

REFERENCES

[1] http://www.cray.com/products/xd1/.
[2] C. Chang, J. Wawrzynek and R Brodersen. BEE2: A high-end reconfigurable computing system. *IEEE Des. Test. Comput.*, pp. 114-125, March, 2005.
[3] T. Hamada et al. Prograpе-1: A Programmable special-purpose computer for many-body simulations. In 6th *IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'98)*, 15-17 April 1998, Napa Valley, CA, USA, pp. 256-257, 1998 .
[4] Hayden Kwok-Hay So. *BORPH: An Operating System for FPGA-Based Reconfigurable Computers*, PhD thesis, University of California, Berkeley, 2008..
[5] http://www.uweb.ucsb.edu/~shahnam/.


คณะวิศวกรรมศาสตร์
FACULTY OF ENGINEERING
PRINCE OF SONGKLA UNIVERSITY
PEC-7
การประชุมวิชาการทางวิศวกรรมศาสตร์ ครั้งที่ 7
The 7th PSU-Engineering Conference
21-22 พฤษภาคม 2552
ณ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์

การประชุมวิชาการทางวิศวกรรมศาสตร์มหาวิทยาลัยสงขลานครินทร์ ครั้งที่ 7
21-22 พฤษภาคม 2552

การออกแบบและพัฒนาวงจรหาขอบภาพด้วยภาษาระดับสูง ImpulseC

วรุตม์ ขยันกิจ[1*] วรรณรัช สันติอมรทัต[2]
[1,2] ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ อ.หาดใหญ่ จ.สงขลา 90112
E-mail: warut.k@hotmail.com*

Warut Khayankit[1*] Wannarat Santiamorntut[2]
[1,2] Department of Computer Engineering, Faculty of Engineering
Prince of Songkla University, Hat Yai, Songkhla 90112
E-mail: warut.k@hotmail.com*

บทคัดย่อ

บทความนี้นำเสนอขั้นตอนการออกแบบวงจรสำหรับหาเส้นขอบภาพด้วยภาษาระดับสูง ImpuseC วงจรที่ได้สามารถพัฒนาบนเทคโนโลยีเอฟพีจีเอเพื่อเพิ่มประสิทธิภาพในการประมวลผลสัญญาณภาพให้ดียิ่งขึ้น ด้วยการใช้งานภาษาระดับสูงนี้ทำให้นักออกแบบสัญญาณภาพสามารถสร้างวงจรขึ้นได้เองอย่างรวดเร็ว โดยบทความนี้ได้ทำการทดสอบประสิทธิภาพของระบบการหาขอบภาพของอัลกอริทึมแบบต่างๆเช่น Robert, Sobel และ Prewitt โดยใช้โครงสร้างการทำงานแบบเดี่ยวซึ่งกำหนดให้ไฟล์อินพุตเป็นภาพในสกุล bmp พบว่าการหาเส้นขอบภาพด้วยวิธีการทั้งสามสามารถจำลองการทำงานได้จริงบนเทคโนโลยีเอฟพีจีเอก่อนนำไปใช้งานบนบอร์ด Spartan3E

คำหลัก ImpulseC, การหาเส้นขอบภาพ, เอฟพีจีเอ

Abstract

This paper presents the design methodology of the edge detection circuits on Field Programmable Gate Array (FPGA) using high level language named ImpulseC. By using this, it could accelerate signal processing system and also reduce the design cycle time. Various edge detection algorithms such as Robert, Sobel and Prewitt, are implemented and well compared. We use the BMP image format as the test input. It's found that edge detection of 3 algorithms can simulate completely on FPGA technology before applying on the Spartan3E board.

Keywords: ImpulseC, Edge Detection, FPGA

1. บทนำ

ปัจจุบันการแข่งขันทางเทคโนโลยีมีสูงมากและจำเป็นที่จะต้องมีประสิทธิภาพสูง เช่น ถูกต้องแม่นยำ ประมวลผลรวดเร็ว ต้นทุนต่ำ เป็นต้น การประมวลผลสัญญาณภาพดิจิตอล (digital image processing) [1][2] แบบเรียลไทม์เป็นที่ต้องการและได้ถูกนำมาประยุกต์ใช้งานหลากหลาย เช่น การตรวจจับวัตถุเคลื่อนที่ การเขียน โปรแกรมตรวจหาตำแหน่งวัตถุ โปรแกรมแยกวัตถุออกจากภาพ โปรแกรมจดจำตัวอักษร(Optical Character Recognition) การประมวลผลภาพที่มีอยู่ในปัจจุบันส่วนใหญ่นั้น ยังคงต้องพึ่งพาการประมวลผลด้วยหน่วยประมวลผลบนเครื่องคอมพิวเตอร์ทั่วไป ซึ่งมีราคาสูง ขนาดใหญ่ และใช้พลังงานมาก ดังนั้นจึงได้ทำการออกแบบวงจรประมวลผลสัญญาณภาพบนเทคโนโลยีเอฟพีจีเอมาใช้กับระบบสมองกลฝังตัว ซึ่งราคาไม่สูง ประหยัดพลังงาน และมีการประมวลผลที่รวดเร็วใกล้เคียงกับเครื่องคอมพิวเตอร์ทั่วไปเมื่อใช้เทคโนโลยีเอฟพีจีเอ

การออกแบบและพัฒนาวงจรดิจิตอลบนเทคโนโลยีเอฟพีจีเอสามารถทำได้โดยการใช้ภาษาอธิบายการทำงานของฮาร์ดแวร์ (Hardware Description Language, HDL) แต่นักออกแบบจำเป็นที่จะต้องมีความรู้และเรียนรู้การใช้ภาษาของฮาร์ดแวร์เพื่อเขียนอธิบายทานการทำงานของวงจร ก่อนที่จะให้ซอฟต์แวร์ช่วยออกแบบทำการสังเคราะห์เป็นวงจรดิจิตอลที่สามารถใช้งานบนเอฟพีจีเอ แต่ขั้นตอนเหล่านี้จะใช้เวลามากและส่วนใหญ่แล้วนักออกแบบและพัฒนาที่เกี่ยวกับการประมวลผลสัญญาณดิจิตอลจะใช้ภาษาในระดับสูงเช่น C หรือ C++ ในการพัฒนางาน ดังนั้นเมื่อ

ต้องการจะเพิ่มความเร็วของการประมวลผลสัญญาณภาพด้วยฮาร์ดแวร์จึงเป็นไปด้วยความลำบากและใช้เวลานาน

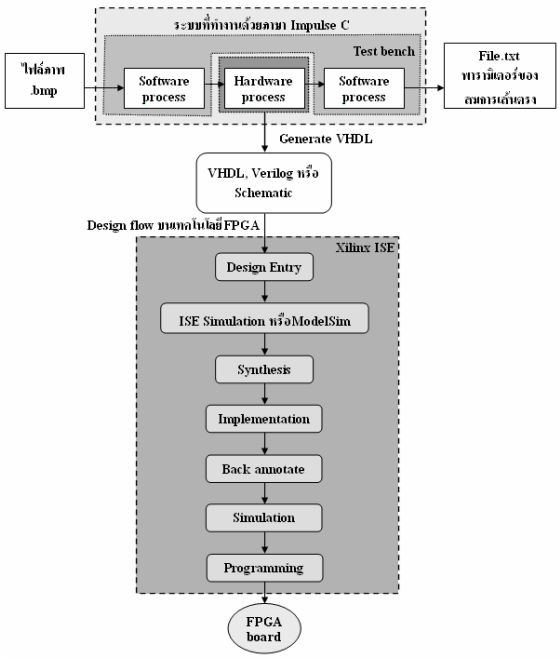
บทความนี้ใช้การหาเส้นขอบภาพเป็นกรณีศึกษาการออกแบบและพัฒนาวงจรด้วยภาษาระดับสูงที่ชื่อ ImpulseC ซึ่งเป็นภาษาที่อยู่บนพื้นฐานของ C++ นอกจากนี้แล้วจะได้ทำการเปรียบเทียบโครงสร้างสถาปัตยกรรมแบบเดี่ยวและขนานของวงจรหาเส้นขอบภาพ รวมถึงการวิเคราะห์หาอัลกอริทึมหาเส้นขอบภาพที่เหมาะสมสำหรับการออกแบบและพัฒนาวงจรบนเอฟพีจีเอ

2. งานและทฤษฎีที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องกับการเพิ่มความเร็วของการประมวลผลภาพด้วยการใช้เทคโนโลยีเอฟพีจีเอ สามารถพบได้ในงานวิจัยหลายชิ้น เช่นการนำเอฟพีจีเอเข้ามาช่วยในการประมวลผลร่วมกับหน่วยประมวลผลบนบอร์ดสมองกลฝังตัว [5] เพื่อเพิ่มประสิทธิภาพในการประมวลผลสัญญาณภาพให้ดียิ่งขึ้น ในงานวิจัย [3] เป็นการออกแบบร่วมฮาร์ดแวร์ซอฟต์แวร์ด้วยภาษาระดับสูง ImpulseC เพื่อเพิ่มประสิทธิภาพในการประมวลผลสัญญาณดิจิตอล ซึ่งเป็นจุดเด่นของภาษา ImpulseC ที่ทำให้ผู้เขียนเลือกใช้ สำหรับในงานวิจัย [4] เป็นตัวอย่างการเพิ่มความเร็วการประมวลผลภาพด้วยฮาร์ดแวร์บนเอฟพีจีเอแต่ใช้วิธีการสร้างด้วยภาษา SA-C ซึ่งไม่ได้รองรับการออกแบบในลักษณะระบบร่วมฮาร์ดแวร์และซอฟต์แวร์

3. ขั้นตอนการออกแบบและพัฒนา

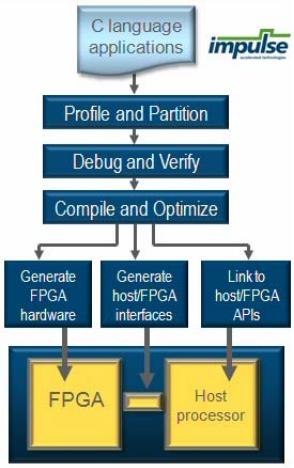
ขั้นตอนการออกแบบและพัฒนาการออกแบบระบบร่วมฮาร์ดแวร์/ซอฟต์แวร์สามารถทำได้หลายวิธี ในงานวิจัยนี้ใช้ภาษา ImpulseC และมีขั้นตอนการออกแบบดังรูปที่ 1 โดยสามารถแบ่งออกได้เป็น 2 ส่วนหลักคือ 1. การพัฒนาฮาร์ดแวร์ด้วย ImpulseC ที่สามารถรองรับการออกแบบร่วมระหว่างฮาร์ดแวร์และซอฟต์แวร์ในที่นี้จะเน้นการออกแบบฮาร์ดแวร์ 2. การขั้นตอนพัฒนาวงจรหรือฮาร์ดแวร์ลงบนเอฟพีจีเอ



รูปที่ 1 ขั้นตอนการออกแบบระบบร่วมฮาร์ดแวร์และซอฟต์แวร์

3.1 การพัฒนาฮาร์ดแวร์ด้วย Impulse C

ImpulseC [6] เป็นภาษาที่อยู่บนพื้นฐานเดียวกับภาษา C มีตัวแปลภาษาที่ชื่อ Impulse CoDeveloper เป็นภาษาอธิบายการทำงานของฮาร์ดแวร์ทั้ง VHDL และ Verilog ตัวแปลของ ImpulseC ยังมีความสามารถเพิ่มความเร็วของระบบด้วยการสร้างวงจรแบบขนานหรือไปป์ไลน์ โดยขั้นตอนการทำงานแสดงได้ดังรูปที่ 2



รูปที่ 2 การทำงานของตัวแปลภาษา Impulse CoDeveloper

3.2 การพัฒนาวงจรบนเทคโนโลยีเอฟพีจีเอ

เมื่อได้วงจรที่อธิบายการทำงานด้วยภาษา VHDL หรือ Verilog จากนั้นใช้ซอฟต์แวร์ที่มีชื่อเรียกว่า Xilinx ISE ช่วยในการสังเคราะห์วงจรและจำลองการทำงาน ซึ่งเป็นขั้นตอนการออกแบบวงจรบนเทคโนโลยีเอฟพีจีเอทั่วไป สามารถอธิบายได้ดังรูปที่ 1 (ในส่วนของ Design flow บนเทคโนโลยีเอฟพีจีเอ)

4. รายละเอียดการออกแบบและพัฒนา

การหาขอบภาพ (Edge Detection Methods) [7][8] ซึ่งมีหลายอัลกอริทึมในการหาขอบภาพ ได้แก่ Robert, Sobel, Prewitt, Robinson, Kirsch และ Canny จากการศึกษางานวิจัยที่เกี่ยวข้อง อัลกอริทึมของ Canny edge detection จะได้รับการยอมรับค่อนข้างสูงว่ามีประสิทธิภาพในการหาขอบภาพที่ดี แต่สาเหตุที่ไม่นำอัลกอริทึม Canny มาใช้เพราะขั้นตอนก่อนที่จะได้ผลลัพธ์นั้นซับซ้อนกว่าแบบอื่นๆมาก ซึ่งจะทำให้วงจรที่สร้างขึ้นนั้นใหญ่เกินความจำเป็น ดังนั้นในงานวิจัยนี้จะนำอัลกอริทึมเพียง 3 แบบมาทำการทดสอบ ได้แก่ Robert, Sobel และ Prewitt เพื่อทำการออกแบบและพัฒนาสร้างวงจรเพื่อเพิ่มประสิทธิภาพการทำงานของระบบ แบ่งการพัฒนาออกเป็น 2 ส่วน ส่วนแรกคือการนำอัลกอริทึมทั้งสามมาออกแบบด้วยภาษา ImpulseC เพื่อหาผลลัพธ์ขอบภาพด้วยโครงสร้างสถาปัตยกรรมแบบเดี่ยว ส่วนที่สองเป็นการสร้างวงจรที่ได้จากภาษา ImpulseC บนเทคโนโลยีเอฟพีจีเอ

4.1 เปรียบเทียบอัลกอริทึมการหาขอบภาพ

4.1.1 การหาขอบภาพด้วยอัลกอริทึม Roberts

การหาขอบด้วยวิธีนี้ก็เป็นอีกตัวอย่างหนึ่งของการใช้เทคนิคการปรับปรุงขอบที่ไม่ต่อเนื่อง ให้ $a \in R^x$ เป็นภาพต้นฉบับ และขอบที่ได้เป็น $b \in R^x$ วิธีการของ Roberts ก็คือ

b(i,j) =

$$\sqrt{((a(i,j)-a(i+1,j+1))^2+(a(i,j+1)-a(i+1,j))^2}$$

ให้ S คือ mask ของแนวแกน x

T คือ mask ของแนวแกน y

$$S=\begin{bmatrix}+1 & 0\\ 0 & -1\end{bmatrix} \quad T=\begin{bmatrix}0 & +1\\ -1 & 0\end{bmatrix}$$

เทมเพลตของ S และ T

เมื่อทำการคำนวณหา edge ของทุก pixel ของทั้งภาพแล้ว จะได้ภาพของ edge ซึ่งสีขาว (High Intensity) จะเป็นขอบภาพ

4.1.2 การหาขอบภาพด้วยอัลกอริทึม Sobel

วิธีนี้เป็นการหาขอบที่ไม่เป็นเชิงเส้น สามารถเปลี่ยนแปลงค่าความไม่ต่อเนื่องได้ตามการปรับปรุงขอบให้ $a \in R^x$ เป็นภาพต้นฉบับ และ a0, a1, a2, ..., a7 แสดงถึงตำแหน่งของแต่ละพิกเซลทั้ง 8 จุด ทวนเข็มนาฬิกา

ให้ $u = (a_5 + 2a_6 + a_7) - (a_1 + 2a_2 + a_3)$ และ

$v = (2a_0 + a_1 + a_7) - (a_3 + 2a_4 + a_5)$

ขนาดขอบของภาพ Sobel เป็น $m \in R^x$ ให้

$$m(i,j) = \sqrt{u^2+v^2}$$

และให้ทิศทางของเกรเดียนต์ของภาพ d คือ

$$d(i,j) = \arctan(\frac{u}{v})$$

ให้ S คือ mask ของแนวแกน x

T คือ mask ของแนวแกน y

$$S=\begin{bmatrix}-1 & 0 & +1\\ -2 & 0 & +2\\ -1 & 0 & +1\end{bmatrix} \quad T=\begin{bmatrix}+1 & +2 & +1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{bmatrix}$$

เทมเพลตของ Sobel

4.1.3 การหาขอบภาพด้วยอัลกอริทึม Prewitt

วิธีการนี้จะคำนวณ ขอบที่เป็นเกรเดียนต์เวคเตอร์ของทุกจุดบนภาพที่เป็นภาพต้นฉบับ ขอบที่ผ่านการปรับปรุงแล้วนั้นมาจากขนาดของเกรเดียนต์เวคเตอร์ มาสค์ที่ใช้แทนอนุพันธ์จะเกี่ยวข้องกับ x และ y ให้ $a \in R^x$ เป็นภาพต้นฉบับและ a1,a2,...,a7 เป็นค่าของแต่ละพิกเซล 8 จุดที่ตำแหน่ง (i,j) ตามทิศทางทวนเข็มนาฬิกาดังรูปที่ 3

a_3	a_2	a_1
a_4	I(i,j)	a_0
a_5	a_6	a_7

รูปที่ 3 แสดงตำแหน่งของตัวแปรด้วยวิธี Prewitt

ให้ $u = (a_5 + a_6 + a_7) - (a_1 + a_2 + a_3)$ และ

$v = (a_0 + a_1 + a_7) - (a_3 + a_4 + a_5)$

ขอบของภาพเป็น $b \in R^x$ ให้

$$b(i,j) = \sqrt{u^2 + v^2}$$

และให้ทิศทางของขอบภาพ $d \in R^x$ คือ

$$d(i,j) = \arctan(\frac{v}{u})$$

ให้ S คือ mask ของแนวแกน x

T คือ mask ของแนวแกน y

$$S = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad T = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}$$

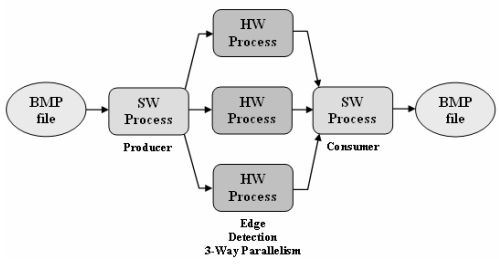
เทมเพลตของ Prewitt

เขียนอธิบายการทำงานของอัลกอริทึมทั้ง 3 แบบด้วยภาษา ImpulseC ในรูปแบบโครงสร้างแบบเดี่ยวดังรูปที่ 4 เพื่อหาผลลัพธ์ของแต่ละอัลกอริทึมงานวิจัยชิ้นนี้

รูปที่ 4 การหาขอบภาพด้วยโครงสร้างแบบเดี่ยว

จากนั้นทำการเปรียบเทียบความเร็วและทรัพยากรที่ใช้ของการประมวลผลระหว่างอัลกอริทึมทั้ง 3 แบบ รูปภาพที่เป็นอินพุตจะเป็นภาพสีในรูปแบบ bmp (24 bits/pixel) ขนาดภาพ 200x200 pixels จึงมีการสร้างวงจรในส่วนของแปลงเป็นภาพระดับสีเทา ก่อนที่จะเข้าสู่วงจรการหาขอบภาพ

นอกจากนี้ยังสามารถออกแบบโครงสร้างการทำงานของการหาขอบภาพให้เป็นแบบขนาน 3-way parallelism ได้ดังรูปที่ 5 โดยแบ่งการประมวลผลเป็น 3 ส่วนพร้อมๆกันตามสีภาพ R, G, B แต่ผลลัพธ์ของระบบภาพที่ได้ไม่ได้มีความแตกต่างจากการใช้ภาพแบบ Gray Scale ดังนั้นจึงไม่เลือกใช้โครงสร้างแบบขนานที่มีขนาดวงจรใหญ่มากกว่า



รูปที่ 5 การหาขอบภาพด้วยโครงสร้างแบบขนาน

4.2 ออกแบบและพัฒนาวงจรการหาขอบภาพ

เมื่อพัฒนาการแปลงภาพและการหาขอบภาพด้วยภาษา ImpulseC แล้ว จากนั้นใช้โปรแกรม Impulse CoDeveloper แปลภาษาเพื่อให้ได้วงจรที่อธิบายด้วยภาษาทางฮาร์ดแวร์ ในที่นี้เลือกใช้เป็นภาษา VHDL จากนั้นก็ผ่านเข้าสู่กระบวนการในการออกแบบวงจรด้วยเทคโนโลยีเอฟพีจีเอตามปกติดังรูปที่ 1 (ในส่วนของ Design flow บนเทคโนโลยีเอฟพีจีเอ)

5. ผลการทดลอง

ในส่วนนี้เป็นการทดสอบประสิทธิภาพของอัลกอริทึมที่ได้นำเสนอ โดยมีผลการทดสอบขั้นแรกที่เป็นการแปลงระดับสีภาพดังในรูปที่ 6 และผลของการหาขอบภาพดังรูปที่ 7

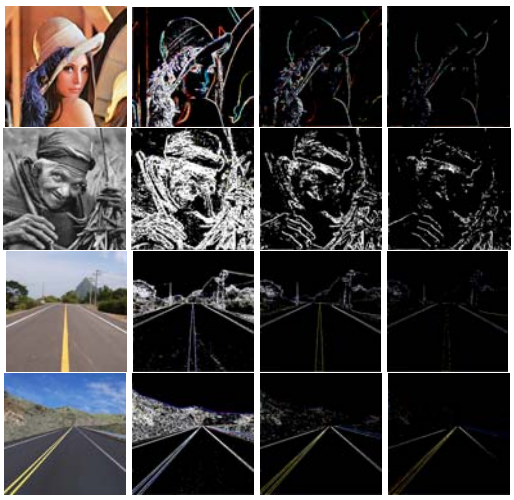
รูปที่ 6 ผลลัพธ์จาก Gray Scale process

รูปที่ 7 ผลลัพธ์จาก Edge Detection process

ผลลัพธ์ของการประมวลผลหาขอบภาพด้วยโครงสร้างแบบเดี่ยว ของอัลกอริทึมทั้ง 3 แบบ ได้แก่ Robert ดังรูปที่ 8 และ 9, Sobel ดังรูปที่ 10 และ 11 สุดท้าย Prewitt ดังรูปที่ 12 และ 13

$$Rx = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \qquad Ry = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

รูปที่ 8 Robert Operator

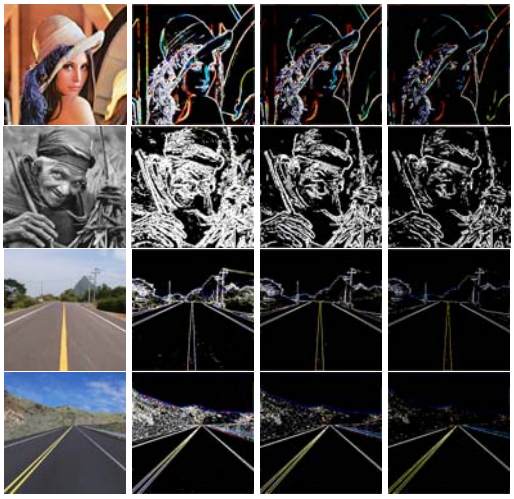


input threshold = 20 threshold = 50 threshold = 80

รูปที่ 9 ผลลัพธ์จากอัลกอริทึมของ Robert

$$Sx = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad Sy = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

รูปที่ 10 Sobel Operator

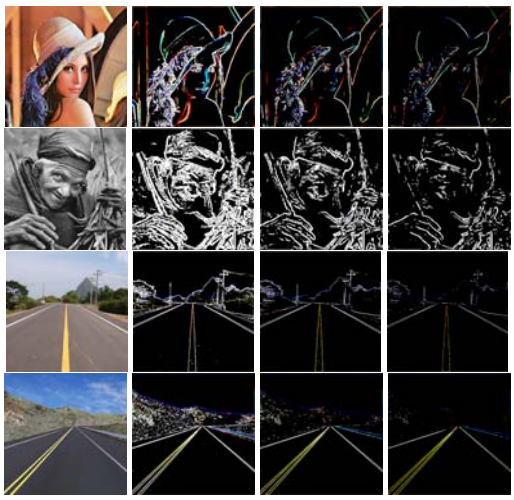


input threshold=100 threshold=180 threshold=260

รูปที่ 11 ผลลัพธ์จากอัลกอริทึมของ Sobel

$$Px = \begin{bmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad Py = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}$$

รูปที่ 12 Prewitt Operator



input threshold=100 threshold=180 threshold=260

รูปที่ 13 ผลลัพธ์จากอัลกอริทึมของ Prewitt

นำโปรแกรมการหาขอบด้วยอัลกอริทึมทั้ง 3 ไปจำลองการทำงานบนเทคโนโลยีเอฟพีจีเอด้วยซอฟต์แวร์ Xilinx ISE เพื่อเปรียบเทียบผลลัพธ์ของแต่ละอัลกอริทึม ดังตารางที่ 1

จำลองการทำงานบนบอร์ดเอฟพีจีเอ Spartan3E (XC3S1600E)

Algorithm / Resources	Robert	Sobel	Prewitt
Number of BUFMUXs	8.3%	8.3%	8.3%
Number of MULT18X18SIOs	25.0%	36.1%	36.1%
Number of RAMB16s	13.8%	16.6%	16.6%
Number of Slices	8.8%	10.8%	10.9%
Number of SLICEMs	0.7%	0.7%	0.7%
Clock net clk_BUFGP(ns)	23.063	32.556	30.781
Clock net sclk_BUFGP(ns)	9.555	8.600	8.805

ตารางที่ 1 เปรียบเทียบผลลัพธ์ของอัลกอริทึม Robert, Sobel และ Prewitt ด้วยขั้นตอนการพัฒนาวงจรบนเทคโนโลยีเอฟพีจีเอ

จากตารางที่ 1 แสดงจำนวนทรัพยากรที่จะใช้งานในแต่ละอัลกอริทึมบนบอร์ด Spartan3E และ Number of Slices จะเป็นตัวบอกถึงพื้นที่โดยรวมที่จะใช้งาน และความเร็วที่ใช้ในการประมวลผล 1 พิกเซล คือค่า clk_BUFGP โดยมีหน่วยเป็น ns

จากผลการทดลอง พบว่า อัลกอริทึมทั้ง 3 ให้ผลลัพธ์ขอบภาพที่มีคุณภาพใกล้เคียงกัน และเมื่อพิจาณาจากทรัพยากรที่ใช้ของแต่ละอัลกอริทึม ทำให้สามารถเลือกใช้งานอัลกอริทึมให้เหมาะสมกับงานที่ต้องการความเร็วและความถูกต้องแม่นยำในระดับต่างๆได้

เมื่อตรวจสอบความถูกต้องของผลลัพธ์แล้ว จะเข้าสู่ขั้นตอนการพัฒนาวงจรบนเทคโนโลยีเอฟพีจีเอตามรูปที่ 1 บนโปรแกรม Xilinx ISE เพื่อทำการ Simulation ก่อนนำไปใช้งานจริงบนบอร์ดเอฟพีจีเอ

6. บทสรุป

ในบทความนี้ได้เสนอการใช้เทคโนโลยีเอฟพีจีเอเข้ามาเป็นตัวช่วยในการประมวลผลร่วมกับหน่วยประมวลผลบนบอร์ดสมองกลฝังตัวเพื่อเพิ่มประสิทธิภาพในการประมวลผลสัญญาณภาพให้ดียิ่งขึ้น โดยงานวิจัยชิ้นนี้ได้ทำการเปรียบเทียบประสิทธิภาพของระบบการหาขอบภาพด้วยอัลกอริทึมทั้ง 3 แบบ ได้แก่ Robert, Sobel และ Prewitt มาประมวลผลแบบเดี่ยว เพื่อหาอัลกอริทึมที่ให้ผลลัพธ์ของการประมวลผลเหมาะสมกับงานวิจัยที่สุด โดยจะนำผลลัพธ์ที่ได้จากขั้นตอนการหาขอบภาพเพื่อนำไปหาเส้นตรงของภาพในขั้นตอนต่อไป ผลที่ได้คือ ทุกอัลกอริทึมที่ทดสอบให้ผลลัพธ์ในขั้นตอนการหาเส้นตรงที่ถูกต้องทั้งหมด ดังนั้น ถึงแม้ Robert จะเป็นวิธีที่ง่าย โดยมีโอเปอเรเตอร์เพียง 2x2 เท่านั้น แต่ให้ผลลัพธ์ขอบภาพซึ่งนำไปสู่การหาเส้นตรงที่ถูกต้องแม่นยำได้ จึงเลือกอัลกอริทึมการหาขอบแบบ Robert เพราะให้ผลลัพธ์ถูกต้องและใช้ทรัพยากรน้อยที่สุดในการจำลองการทำงานบนเอฟพีจีเอ

6.1 แนวทางการพัฒนาต่อ

เมื่อได้การหาขอบภาพแล้วสามารถนำมาใช้งานร่วมกับกระบวนการหาเส้นตรงของภาพ (Hough Transform) ผลลัพธ์ที่ได้จากกระบวนการนี้คือ พิกัด (X,Y) ของจุดตั้งต้นและจุดสิ้นสุดของเส้นตรงต่างๆที่ตรวจพบบนภาพ ซึ่งสามารถนำพิกัดเหล่านี้ไปสร้างเป็นสมการเส้นตรงได้ โดยระบบจะถูกออกแบบและพัฒนาแบบร่วมระหว่างฮาร์ดแวร์และซอฟต์แวร์ด้วยภาษา ImpulseC ซึ่งการทำงานส่วนของวงจรจะถูกโปรแกรมและใช้งานบนพื้นที่ส่วนหนึ่งของ เอฟพีจีเอ ส่วนของซอฟต์แวร์จะใช้งานบนไมโครคอนโทรลเลอร์ที่ฝังอยู่ในเอฟพีจีเอเช่น MicroBlaze หรือ PowerPC

7. เอกสารอ้างอิง

[1] ดร.เกียรติศักดิ์ ศรีพิมานวัฒน์, วุฒิกรณ์ ตรัยศิลานันท์, พรวิภา ทัศยาพันธุ์, “การสำรวจการประมวลผลภาพและการใช้รหัสเพื่อควบคุมความผิดพลาด”, http://www.kmitl.ac.th/dslabs/ksripima/readme/49/technical_Report/intern_49/Pornwipa/Report%20Pornviphaa%20Tassayaphan.pdf

[2] วศิน สินธุภิญโญ, “Digital Image Processing and Digital Signal Processing”, กิจกรรมการบรรยายพิเศษทางด้านวิทยาศาสตร์ คณิตศาสตร์ และเทคโนโลยี, 8 มกราคม 2550

[3] Paula J. Pingree, Lucas J. Scharenbroich, Thomas A. Wenne and David Pellerin, “Developing FPGA Coprocessors for Performance-Accelerated Spacecraft Image Processing” *Xcell Journal,* 2nd quarter, pp.23-26, 2008

[4] Bruce A. Draper, J. Ross Beveridge, A.P. Willem Bohm, Charles Ross, Monica Chawathe, Jeffrey Hammes, “Accelerated Image Processing on FPGAs”, *IEEE Transactions* on *Image Processing*, Volume 12, Issue 12, pp.1543-1551, December, 2003.

[5] Japan System House Association, “เทคโนโลยีสมองกลฝังตัว Embedded Technology”, 1/2549

[6] http://www.impulsec.com

[7] http://www.pages.drexel.edu/~weg22/edge.html

[8] Ellen C. Hildreth, “Theory of edge detection”, *Journal*, vol. 207, pp. 187-217, 1980.

ประวัติผู้เขียน

ชื่อ สกุล นายวรุตม์ ขยันกิจ

รหัสประจำตัวนักศึกษา 4910120081

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิศวกรรมศาสตรบัณฑิต (วิศวกรรมคอมพิวเตอร์)	มหาวิทยาลัยสงขลานครินทร์	2549

ทุนการศึกษา (ที่ได้รับในระหว่างการศึกษา)

ทุนศิษย์ก้นกุฏิ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์

ทุนการศึกษาระดับบัณฑิตศึกษา บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์

การตีพิมพ์เผยแพร่ผลงาน

วรุตม์ ขยันกิจ, วรรณรัช สันติอมรทัต, "Hardware/Software Co-design for Line Detection Algorithm on FPGA", ECTI-CON 2009 6th, 6-9 พฤษภาคม 2552, 604-606.

วรุตม์ ขยันกิจ, วรรณรัช สันติอมรทัต, "การออกแบบและพัฒนาวงจรหาขอบภาพด้วยภาษาระดับสูง ImpulseC", การประชุมวิชาการทางวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ครั้งที่ 7, 21-22 พฤษภาคม 2552, 120-125.