วงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจายด้วยอุปกรณ์ FPGA

DISTRUBUTED ARITHMETIC DIGITAL IMAGE FILTER USING FPGA

โดย

นายอภิรักษ์ ปานชื่น

นายอมรเทพ สนินัด

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

วงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจายด้วยอุปกรณ์ FPGA

DISTRUBUTED ARITHMETIC DIGITAL IMAGE FILTER USING FPGA

โดย

นายอภิรักษ์ ปานชื่น

นายอมรเทพ สนินัด

อาจารย์ที่ปรึกษา

รศ.ดร. กอบชัย เดชหาญ

อ. ศรวัฒน์ ชิวปรีชา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **วงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจายด้วยอุปกรณ์ FPGA**

DISTRIBUTED ARITHEMETIC DIGITAL IMAGE FILTER USING FPGA

ผู้จัดทำ

1. **นายอภิรักษ์ ปานชื่น** **44015044**
2. นายอมรเทพ สนินัด **44015045**

(_____) อาจารย์ที่ปรึกษา

รศ.ดร. กอบชัย เดชหาญ

(_____) อาจารย์ที่ปรึกษา

อาจารย์ ศรีรัตน์ ชิวปรีชา

วงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจายด้วยอุปกรณ์ FPGA

DISTRIBUTED ARITHMETIC DIGITAL IMAGE FILTER USING FPGA

โดย นายอภิรักษ์ ปานชื่น 44015044

นายอมรเทพ สนินัด 44015045

อาจารย์ที่ปรึกษา รศ.ดร. กอบชัย เดชหาญ

อาจารย์ ศรวัฒน์ ชิวปรีชา

บทคัดย่อ

โครงงานนี้นำเสนอเกี่ยวกับการออกแบบและการสร้างวงจรกรองสัญญาณภาพเชิงเลข (Digital Image Filter) โดยจะแสดงให้เห็นถึงการนำเอาฮาร์ดแวร์เข้าไปประยุกต์ใช้กับการกรองสัญญาณเชิงเลขสองมิติแบบกรองความถี่ต่ำผ่าน (Low-Pass Filtering) และการตรวจจับขอบภาพโดยแบบโซเบล (Sobel Edge Detector) โดยนำวิธีการของโครงสร้างเลขคณิตกระจาย (Distributed Arithmetic, DA) มาใช้ในการสร้างเป็นฮาร์ดแวร์ด้วยอุปกรณ์ FPGA ทำให้วงจรกรองสัญญาณภาพที่ได้ปราศจากตัวคูณ

ABSTRACT

This project presents a design and implementation of digital image filter. The hardware realization will be applied to two dimensions digital signal filtering base on low-pass filtering and sobel edge detector. For hardware implementation will be used DA (Distributed Arithmetic) to implement with FPGA (Field Programmable Gate Array). Consequently, will be obtained digital image filter that disappear multipliers.

สารบัญ

สารบัญ(ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญรูปภาพ(ต่อ)

หน้า

สารบัญรูปภาพ(ต่อ)

สารบัญรูปภาพ(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาของหัวข้อปริญญานิพนธ์

การประมวลผลสัญญาณดิจิตอลที่เรียกว่า ดิจิตอลซิกแนลโปรเซสซิ่ง (Digital Signal Processing) หรือใช้ชื่อย่อว่า DSP เป็นขบวนการที่ใช้ความรู้ทางคณิตศาสตร์เกี่ยวกับตัวเลขมาทำการจัดการกับสัญญาณต่างๆ ซึ่งรูปแบบของสมการคณิตศาสตร์ในการคำนวณหาค่าผลลัพธ์ที่ได้มักจะอยู่ในรูปของสมการผลบวกของผลคูณ (Sum of Product) ระหว่างสัญญาณ Input กับ ค่าสัมประสิทธ์ของวงจรดิจิตอล ซึ่งถ้านำสมการไปสร้างเป็นวงจรดิจิตอลที่มีจำนวนสัญญาณ Input มากๆ ทำให้ต้องใช้ตัวคูณมากขึ้นด้วย ทำให้การสร้างทางด้านฮาร์ดแวร์ต้องสิ้นเปลืองเกทจำนวนมาก มีผลทำให้เวลาในการคำนวณช้าลง จึงมีการคิดค้นวิธีการที่จะทำการลดจำนวนตัวคูณให้มีจำนวนน้อยลงหรือไม่ใช้ตัวคูณเลย เพื่อทำให้ลดจำนวนเกทที่ใช้และเพิ่มความเร็วการทำงานให้กับฮาร์ดแวร์ ซึ่งมีวิธีการหนึ่งที่สามารถสร้างได้ในทางด้านฮาร์ดแวร์เรียกว่า การกระจายทางคณิตศาสตร์ (Distributed Arithmetic) หรือเรียกย่อๆว่า DA ซึ่งจะทำการจัดสมการให้อยู่ในรูปแบบการคำนวณแบบบิท และใช้วิธีการคูณแบบเปิดตารางค่าสัมประสิทธิ์ที่ถูกเก็บไว้ในหน่วยความจำแบบอ่านได้อย่างเดียวซึ่งจะทำให้ไม่ต้องใช้ตัวคูณ

โครงงานนี้จึงทำการนำการกระจายทางคณิตศาสตร์มาทำการประยุกต์ใช้กับการประมวลผลภาพ (Image Processing) เพื่อนำมาสร้างเป็นวงจรดิจิตอลโดยใช้ภาษาบรรยายพฤติกรรมของวงจรดิจิตอล (VHDL) ในการสร้างเพื่อที่จะทำให้เวลาในการประมวลผลภาพเร็วขึ้น

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1.2.1 เพื่อศึกษาและประยุกต์ใช้งานการประมวลผลสัญญาณดิจิตอล

1.2.2 เพื่อศึกษาเกี่ยวกับภาษาที่ใช้บรรยายพฤติกรรมของวงจรดิจิตอล

1.2.3 เพื่อศึกษาหลักการของการกระจายทางคณิตศาสตร์

1.2.4 เพื่อนำหลักการของการกระจายทางคณิตศาสตร์มาประยุกต์ใช้กับการประมวลผลสัญญาณภาพ

1.3 ขอบเขตของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้เป็นการออกแบบและการสร้างวงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจายด้วยอุปกรณ์ FPGA ซึ่งแสดงให้เห็นถึงการประยุกต์ใช้การกระจายทางคณิตศาสตร์ในการนำมาใช้ในวงจรดิจิตอลทางด้านการประมวลผลสัญญาณภาพ โดยในขั้นตอนการการสร้างจะทำการเขียนบรรยายพฤติกรรมของวงจรด้วยภาษา VHDL โดยโปรแกรม Max+Plus II

1.4 เนื้อหาของปริญญานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎีที่นำมาใช้ในการออกแบบและสร้างวงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจาย ซึ่งประกอบด้วยพื้นฐานการประมวลข้อมูลภาพ ทฤษฎีการกระจายทางคณิตศาสตร์ การนำการกระจายทางคณิตศาสตร์มาประยุกต์ใช้กับการกรองสัญญาณภาพแบบ Low-pass filter และ Sobel edge Detector

บทที่ 3 กล่าวถึงภาษาที่ใช้บรรยายพฤติกรรมของวงจรดิจิตอล (VHDL) และวิธีการออกแบบวงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจาย

บทที่ 4 กล่าวถึงการทดลองและผลการทดลองที่ได้มาจากการจำลองการทำงานของวงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจาย

บทที่ 5 เป็นการวิจารณ์ วิเคราะห์และสรุปผล

บทที่ 2
ทฤษฎีหรือหลักการ

2.1 การประมวลผลสัญญาณภาพ (Image Processing)

ในที่นี้จะกล่าวถึงการประมวลผลสัญญาณเชิงเลขสองมิติซึ่งเป็นข้อมูลภาพ (Image Processing) และองค์ประกอบพื้นฐานในการประมวลผลภาพข้อมูล รูปแบบของอิมเมจ การแทนภาพในระบบดิจิตอล การเพิ่มความชัดเจนของภาพ

2.1.1 การแทนภาพในระบบดิจิตอล (Digital Image Representation)

ในระบบภาพโดยทั่วไปสามารถแทนด้วยฟังก์ชั่นของความเข้มแสงในระนาบสองมิติ f(x,y) โดยที่ x และ y แทนตำแหน่งคู่ลำดับที่เกิดขึ้นในภาพจริงและค่าฟังก์ชั่น f ณ จุด (x,y) ใดๆจะเป็นสัดส่วนโดยตรงกับความสว่างหรือระดับสีเทา (Gray level) ของภาพที่จุดนั้นๆแสดงดังรูปที่ 2.1

รูปที่ 2.1 ค่าของฟังก์ชั่น f(x,y) ในระบบภาพดิจิตอล

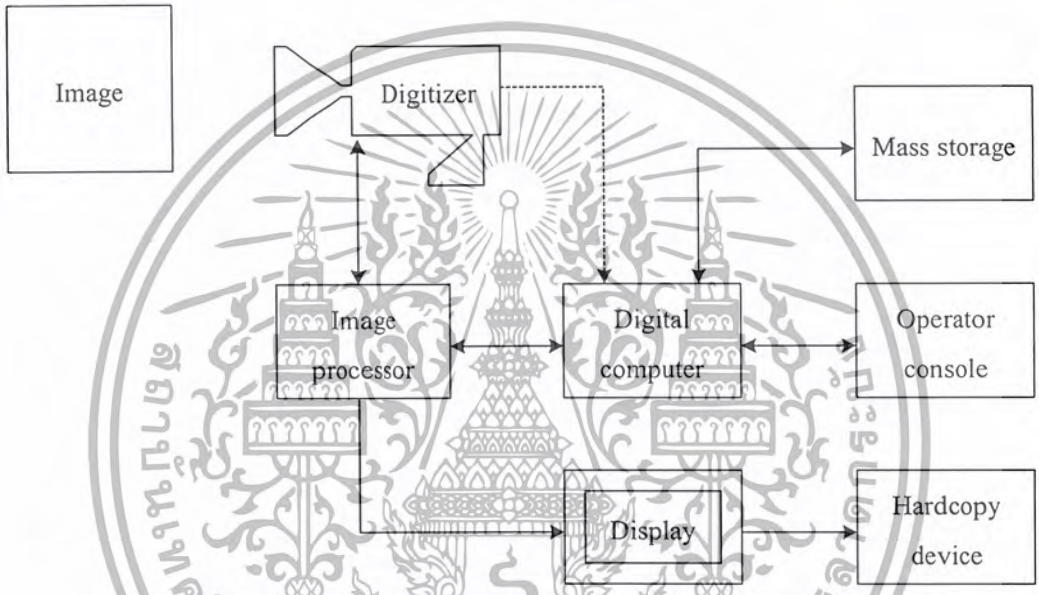
สำหรับข้อมูลภาพแบบดิจิตอล f(x,y) มีลักษณะที่ไม่ต่อเนื่องทั้งในระนาบสเปเชียล (Spatial) ดังนั้นจึงพิจารณาให้ข้อมูลภาพแบบดิจิตอลแทนอยู่ในรูปของเมตริกซ์ ซึ่งเมตริกซ์จะมีแถวและหลักที่มีลักษณะเป็นเอกลักษณ์ และมีความสัมพันธ์กับค่าสมาชิกในแถวและหลัก ที่แทนด้วยระดับสีเทาของภาพที่จุดนั้นๆ

ถึงแม้ว่าขนาดของข้อมูลภาพแบบดิจิตอลจะเปลี่ยนไปตามการประยุกต์ใช้งาน แต่จะมีข้อดีที่ได้เปรียบถ้ากำหนดให้มีลักษณะเฉพาะเป็นอะเรย์แบบสมมาตร (Square array) และจำนวนของระดับสีเทาเป็นจำนวนเต็มกำลังสอง

2.1.2 องค์ประกอบของระบบประมวลผลข้อมูลภาพดิจิตอล (Elements of A Digital Image Processing System)

องค์ประกอบพื้นฐานโดยทั่วในระบบการประมวลข้อมูลภาพ ประกอบด้วย 5 ส่วนใหญ่ๆ คือ

1. ส่วนเปลี่ยนสัญญาณอนาล็อกเป็นสัญญาณดิจิตอล (Digitizer)
2. เครื่องประมวลผลข้อมูลภาพ (Image Processor)
3. เครื่องคอมพิวเตอร์แบบดิจิตอล (Digital Computer)
4. อุปกรณ์ที่ใช้ในการแสดงผลและบันทึกผล (Display and Recording Devices)
5. อุปกรณ์ที่ใช้ในการเก็บรักษาข้อมูล (Storage Devices)



รูปที่ 2.2 องค์ประกอบของระบบการประมวลผลแบบดิจิตอล

1.) ส่วนเปลี่ยนสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอล (Digitizer)

ส่วนของดิจิไตเซอร์ ทำหน้าที่เปลี่ยนสัญญาณภาพให้อยู่ในรูปของตัวเลขเพื่อใช้เป็นข้อมูลป้อนเข้าดิจิตอลคอมพิวเตอร์ ส่วนนี้ได้แก่ กล้องโทรทัศน์ ดิจิไตเซอร์ ภายในประกอบด้วยหลอดวิดิคอน ทำหน้าที่เป็นสื่อนำไฟฟ้าทางแสง ภาพจะถูกโฟกัสลงบนผิวของหลอด ซึ่งถูกเปลี่ยนให้เป็นสัญญาณไฟฟ้า ซึ่งหลังจากการควอนไตซิ่งสัญญาณนี้จะได้ภาพดิจิตอลเกิดขึ้น

2.) เครื่องประมวลข้อมูลภาพ (Image Processor)

เครื่องประมวลข้อมูลภาพแบบดิจิตอลเป็นหัวใจของระบบการประมวลข้อมูลภาพ เครื่องประมวลข้อมูลภาพประกอบด้วยชุดฮาร์ดแวร์ที่มี 4 ฟังก์ชั่นพื้นฐานคือ การหยุดภาพนิ่ง (Image Acquisition) การเก็บข้อมูลภาพ (Storage) การประมวลภาพในระดับต่ำ (แต่รวดเร็ว) (Low Level (fast) Processing) และการแสดงผล (Display) โดยทั่วไปโมดูลของการหยุดภาพนิ่งจะมีสัญญาณโทรทัศน์เป็นหน่วยอินพุทและทำการเปลี่ยนสัญญาณนี้ให้เป็นข้อมูลแบบดิจิตอล เครื่องประมวลข้อมูลภาพสมัยใหม่

ส่วนมากจะใช้การดิจิไตซ์ภาพโทรทัศน์ในเวลา 1 เฟรม (One frame-time) เช่น ในเวลา 1/30 วินาที จึงมักเรียกว่าเครื่องจับภาพนิ่ง (Frame Grabber)

โมดูลของการเก็บภาพแบบที่เรียกว่า เฟรมบัฟเฟอร์ (Frame Buffer) จะทำหน้าที่เก็บรักษาหน่วยความจำของภาพแบบดิจิตอล โดยทั่วไปจะมีเครื่องมือ 2-3 ชุดที่รวมไว้ในเครื่องประมวลข้อมูลภาพ คุณสมบัติการหยุดภาพของโมดูลการเก็บภาพประกอบด้วยหน่วยความจำที่สามารถอ่านข้อมูลจากอัตราของโทรทัศน์ปกติมีอัตรา 30 ภาพต่อวินาที ลักษณะนี้เป็นไปตามการทำงานของโมดูลการหยุดภาพนิ่งที่ประกอบไปด้วยภาพที่สมบูรณ์ในการเก็บภาพที่เร็วเท่าที่ภาพถูกจับให้นิ่ง หน่วยความจำสามารถเปลี่ยนให้เป็นแอดเดรสของโทรทัศน์ได้โดยชุดแสดงผลของภาพที่จอมอนิเตอร์

เครื่องประมวลข้อมูลภาพยังมีหน้าที่ในฟังก์ชั่นระดับต่ำอีกด้วย (Low-level Function) เช่นการประมวลผลทางคณิตศาสตร์ และทางตรรกศาสตร์หรือที่เรียกว่าหน่วยตรรกเลขคณิต (Arithmetic Logic Unit : ALU) โดยถูกออกแบบพิเศษเพื่อที่จะเพิ่มความเร็วในการประมวลผลแบบขนานส่วนโมดูลแสดงผล (Display Module) ทำหน้าที่อ่านหน่วยความจำของภาพแล้วทำการเปลี่ยนข้อมูลของภาพที่เป็นแบบดิจิตอลให้เป็นสัญญาณวีดีโอแบบอนาล็อกและแสดงผลสัญญาณที่แปลงได้นั้นให้ออกทางมอนิเตอร์โทรทัศน์หรืออุปกรณ์วีดีโอชนิดอื่น

3.) เครื่องคอมพิวเตอร์แบบดิจิตอล (Digital Computer)

ถึงแม้ว่าเครื่องประมวลข้อมูลภาพอาจเปรียบได้กับความสามารในการประมวลผลภายในดังที่ได้กล่าวมาแล้วแต่ว่าระดับของการประมวลผลยังอยู่ในระดับค่อนข้างต่ำ ในด้านความทันสมัยซึ่งโดยทั่วไปพบว่าเครื่องประมวลข้อมูลภาพนี้จะถูกต่อเชื่อม (Interface) กับเครื่องคอมพิวเตอร์ระบบคอมพิวเตอร์ที่ใช้สำหรับการประมวลผลของข้อมูลภาพมีตั้งแต่อุปกรณ์จำพวกไมโครโปรเซสเซอร์จนถึงระบบคอมพิวเตอร์ขนาดใหญ่ที่มีความสามารถในการประมวลผลข้อมูลภาพ ซึ่งจะต้องมีอะเรย์ใหญ่ๆด้วย ส่วนพารามิเตอร์ที่สำคัญนั้นคือการเพิ่มการประยุกต์ใช้งานและความต้องการของขนาดข้อมูลในการประมวลผล สำหรับการประยุกต์ใช้ที่พบเห็นได้โดยทั่วไปจะเป็นเครื่องคอมพิวเตอร์ที่มีหน่วยความจำของแอดเดรสจำนวนมากและมีความเร็วสูงก็จะเป็นข้อได้เปรียบ

4.) อุปกรณ์ที่ใช้ในการประมวลผลและบันทึกผล (Display and Recording Device)

อุปกรณ์ที่ใช้ในการแสดงผลที่สำคัญและนิยมใช้กันมากในระบบการประมวลข้อมูลภาพได้แก่ จอโมโนโครม หรือ จอโทรทัศน์สี มอนิเตอร์จะถูกขับโดยเอาท์พุทของชุดแสดงผล ภาพในเครื่องประมวลผลข้อมูลภาพ สัญญาณที่ได้รับนี้สามารถเข้าสู่ระบบการบันทึกภาพที่อาจใช้ฮาร์ดก๊อปปี้ได้ (Hard copy) เช่น สไลด์สี รูปภาพ และแผ่นใส หรือ ในขณะที่กำลังดูภาพจากมอนิเตอร์ก็ได้ ส่วนสื่อที่ใช้ในการแสดงผลและบันทึกผลที่พบกันโดยทั่วไปคือ จอภาพแบบหลอดภาพหลอดภาพรังสีคาโธด ตำแหน่งของภาพทั้งในแนวนอน (Horizontal) และแนวตั้ง (Vertical) ของแต่ละสมาชิก ในอะเรย์ของภาพจะถูกเปลี่ยนให้เป็นโวลเตจ ซึ่งถูกยิงจากลำแสงอิเลคตรอนของหลอดภาพรังสีคาโธด โดยหารขับลำแสงใน 2 มิติ ที่จำเป็นต่อการผลิตภาพเอาท์พุทในแต่ละจุดจะมีการยิงลำแสงอิเลคตรอน ความเข้มของแสงจะถูกเปลี่ยนโดยการใช้โวลเตจที่เป็นสัดส่วนโดยตรงต่อค่าของจุด ที่มีความสัมพันธ์กับในอะเรย์เชิงเลข

(Numerical array) โดยการเปลี่ยนความเข้มน้อยสุดให้เป็นสีดำและความเข้มข้นมากที่สุดให้เป็นสีขาว ผลการเปลี่ยนแปลงความเข้มของแสงจะถูกบันทึกด้วยกล้องถ่ายภาพแบบกราฟฟิก (Photographic camera) ที่โฟกัสของจอภาพแบบหลอดภาพรังสีคาโธด

ส่วนอุปกรณ์แสดงผลภาพพิมพ์ที่ใช้ในขั้นต้นจะมีความละเอียดต่ำ ซึ่งสามารถแสดงระดับสีเทาได้ด้วยการพิมพ์จำนวนและความเข้มของอักขระที่พิมพ์ในตำแหน่งนั้นๆ ในการเลือกอักขระที่เหมาะสมกับการกระจายในแบบระดับสีเทาที่ดี ก็จะทำให้เครื่องพิมพ์สามารถแสดงภาพระดับสีเทาได้

5.) อุปกรณ์ที่ใช้ในการเก็บรักษาข้อมูล (Storage Devices)

ข้อมูลภาพดิจิตอลที่มีขนาด 256*256 พิกเซลแต่ละควอนไตซ์มีข้อมูลขนาด 8 บิท โดยมีสื่อที่ใช้ในการเก็บรักษาข้อมูลที่สำคัญ คือ แผ่นจานแม่เหล็ก (Magnetic disks) เทปแม่เหล็ก (Magnetic tapes) และแผ่นจานนำแสง (Optical disks) แต่สื่อที่นิยมใช้ในการเก็บข้อมูลในระดับทั่วไปคือ แผ่นจานแม่เหล็ก

2.1.3 การเพิ่มความคมชัดของภาพ (Image Enhancement)

จุดประสงค์ของเทคนิคการเพิ่มความชัดเจนของภาพนั้นคือ ต้องทำภาพให้มีผลลัพธ์ที่เหมาะสมเฉพาะงานที่จะนำไปประยุกต์ใช้มากกว่าภาพต้นแบบเดิมซึ่งมีความสำคัญมาก เพราะว่าการใช้เทคนิคเดียวกันนี้มาใช้ร่วมกันได้หมด ในหัวข้อนี้จะกล่าวถึงรูปแบบอิมเมจ วิธีการฟิลเตอร์

1.) โมเดลข้อมูลภาพ (An Image Model)

ในที่นี้คำว่าข้อมูลภาพหมายถึง ฟังก์ชั่นของความเข้มแสง 2 มิติโดยมีรูปแบบ f(x,y) ซึ่งขนาดหรือแอมปลิจูด (Amplitude) ของฟังก์ชั่นที่พิกัดสเปเชี่ยล (Spatial Coordinates) ที่จุด (x,y) จะแสดงค่าความสว่าง (Brightness) ของข้อมูลภาพที่จุดนั้นๆเมื่อแสงเป็นรูปแบบพลังงานแบบหนึ่ง ดังนั้น f(x,y) จึงต้องมีขนาดไม่เท่ากับศูนย์และจะต้องมีขนาดไม่เท่ากับอนันต์ (Finite) ดังนั้นจะได้ว่า

$$0 < f(x,y) < \alpha \tag{2.1}$$

พื้นฐานโดยทางธรรมชาติของ f(x,y) อาจจะได้รับการพิจารณาเกี่ยวกับคุณสมบัติสองประการดังต่อไปนี้ คือประการแรกจะพิจารณาเกี่ยวกับแหล่งกำเนิดแสงทั้งหมดที่เกิดขึ้นเองบนฉาก ประการที่สองตะพิจารณาเกี่ยวกับแสงสะท้อนทั้งหมดที่เกิดขึ้นจากการสะท้อนของวัตถุในฉากเอง คุณสมบัติทั้งสองประการนี้จะถูกเรียกว่า การส่องสว่าง (Illumination) และการสะท้อน (Reflectance) ซึ่งมีรูปแบบฟังก์ชั่นดังนี้คือ i(x,y) และ r(x,y) ตามลำดับ ทั้งสองฟังก์ชั่นนี้รวมกันจะกลายเป็นรูปแบบของ f(x,y)

จะได้ว่า

$$f(x,y) = i(x,y) \bullet r(x,y) \tag{2.2}$$

เมื่อ

$$0 < i(x,y) < \alpha \tag{2.3}$$

และ

$$0 < r(x,y) < \alpha \tag{2.4}$$

จากสมการที่ (2.4) ชี้ให้เห็นถึงข้อเท็จจริงว่าขอบล่างที่มีค่าเป็น "ศูนย์" คือการดูดกลืนแสงทั้งหมด (Total absorption) ขอบบนที่มีค่าเป็น "หนึ่ง" คือการสะท้อนของแสงทั้งหมด (Total reflectance) ที่เกิดขึ้น ดังนั้นธรรมชาติของฟังก์ชั่น f(x,y) จะถูกกำหนดโดยแหล่งกำเนิดของแสงขณะที่ฟังก์ชั่น r(x,y) จะถูกกำหนดโดยคุณสมบัติของตัววัตถุในฉากนั้นเอง ที่ความเข้มของฟังก์ชั่นข้อมูลภาพสีเดียว (Monochrome image) ที่จุด (x,y) จะถูกเรียกว่าระดับสีเทา (Gray level) หรือ (I) ของภาพที่จุดๆนั้น

2.) ฟิลเตอร์ (Filter)

ฟิลเตอร์เป็นตัวกรองสัญญาณภาพเบื้องต้นให้ได้ภาพที่มีความคมชัดมากยิ่งขึ้นเพื่อที่จะนำภาพไปประยุกต์ใช้ได้ดียิ่งขึ้น ซึ่งมีมากหลายชนิดแต่ในที่นี้จะกล่าวเฉพาะที่นิยมใช้กันมากเท่านั้น

2.1.) การเฉลี่ยค่ารอบย่าน (Neighbourhood Average)

การเฉลี่ยค่ารอบย่านเป็นเทคนิคที่ใช้กันในสเปเชียลโดเมนโดยตรงสำหรับการทำภาพให้ชัดเจนยิ่งขึ้น ให้ภาพอินพุท f(x,y) มีขนาด n*n และ g(n,y) เป็นผลของภาพที่ได้จากการปฏิบัติของระดับสีเทาทุกๆจุด (x,y) ได้จากการเฉลี่ยค่าระดับสีเทาของพิกเซลในภาพ f(x,y) ซึ่งสามารถอธิบายได้ดังนี้

$$g(x,y) = \frac{1}{M}\sum_{n,m} f(n,m) \qquad (2.5)$$

สำหรับ x,y = 0,1,...,N-1 และ M คือผลรวมของจำนวนจุดในย่านนั้นๆ

2.2.) ฟิลเตอร์แบบมัธยฐาน (Median Filter)

สิ่งสำคัญประการหนึ่งจากการเฉลี่ยค่ารอบย่าน คือ การเกิดความเบลอร์ที่ขอบของภาพ และมีรายละเอียดสัญญาณรบกวนที่ชัดเจน อาจแก้ไขได้โดยการใช้เทรสโฮลด์ (Threshold) ซึ่งโดยทั่วไปแล้วค่าเทรสโฮลด์จะได้จากการลองผิดลองถูก แต่ถ้าใช้ฟิลเตอร์แบบมัธยฐานแทนระดับสีเทาของแต่ละพิกเซลโดยการใช้ค่าเฉลี่ยมัธยฐานของระดับสีเทาในรอบย่านนั้นๆของพิกเซลจะทำให้ภาพที่ได้มีประสิทธิภาพที่ได้มีประสิทธิภาพดีขึ้น การหาค่ามัธยฐาน (M) คือเซ็ตที่มีค่าในเซ็ตครึ่งหนึ่งมากกว่า M และอีกครึ่งหนึ่งน้อยกว่า M การหาค่ามัธยฐานสามารถหาได้โดยทำการเรียงข้อมูลจากน้อยไปหามากแล้วนำค่าที่อยู่ตรงกลางนั้นมาใช้แทนค่า M

ตัวอย่าง

สมมติว่ามีย่านขนาด 3*3

(10, 20, 20, 20, 15, 20, 20, 25, 100)

สามารถเรียงข้อมูลใหม่ได้เป็น

(10, 15, 20, 20, 20, 20, 20, 25, 100)

ดังนั้นค่าที่อยู่ตรงกลางเป็นค่ามัธยฐานมีค่าเท่ากับ 20 นั่นเอง ความคิดที่มีอิทธิพลต่อฟังก์ชั่นของฟิลเตอร์แบบมัธยฐานคือจะทำให้จุดที่มีความเข้มที่แตกต่างมากกว่าย่านรอบๆของมัน

2.3.) ฟิลเตอร์แบบความถี่ต่ำผ่าน (Low-pass Filter)

สัญญาณรบกวนในระดับสีเทาของภาพมักจะมีส่วนของสัญญาณความถี่สูงของการแปลงฟูเรียร์อย่างมากซึ่งจะทำให้เกิดการเบลอร์ของภาพ ดังนั้นจึงต้องหาวิธีการกำจัดเฉพาะช่วงความถี่สูงที่ไม่ต้องการออกไป จากทฤษฎีการทำคอนโวลูชั่นจะได้ว่า

$$G(u,v) = H(u,v)F(u,v) \tag{2.6}$$

เมื่อ F(u,v) คือการแปลงฟูเรียร์ของภาพที่ต้องการให้เรียบ ปัญหาก็คือการเลือกทรานส์เฟอร์ฟังก์ชั่น H(u,v) ที่ให้ได้ผลลัพธ์ G(u,v) โดยที่สามารถกำหนดจุดช่วงที่ความถี่สูงของ F(u,v) ออกไปได้การแปลงอินเวอร์สลาปาส (Inverse Laplace) ของ G(u,v)

$$g(x,y) = L^{-1}[H(u,v)F(u,v)] \tag{2.7}$$

จะทำให้ได้ g(x,y) เรียบขึ้นเพราะว่าส่วนประกอบความถี่สูงไม่สามารถผ่านออกไปได้ แต่ช่วงความถี่ต่ำจะสามารถผ่านไปได้โดยปราศจากการลดทอนของสัญญาณซึ่งวิธีนี้เรียกว่า ฟิลเตอร์แบบความถี่ต่ำผ่าน ฟังก์ชั่น H(u,v) ถูกเรียกว่าทรานส์เฟอร์ฟังก์ชันของฟิลเตอร์ (Filter Transfer Function) ซึ่งในการคำนวณจะพิจารณาเฉพาะ H(u,v) เท่านั้น ฟิลเตอร์แบบความถี่ต่ำผ่านนี้มีอยู่หลายชนิดด้วยกัน โดยที่แต่ละฟิลเตอร์จะทำหน้าที่กรองความถี่สูงออกไปให้เหลือเฉพาะความถี่ต่ำเท่านั้น

2.2 หลักการเบื้องต้นของโครงสร้างเลขคณิตกระจาย

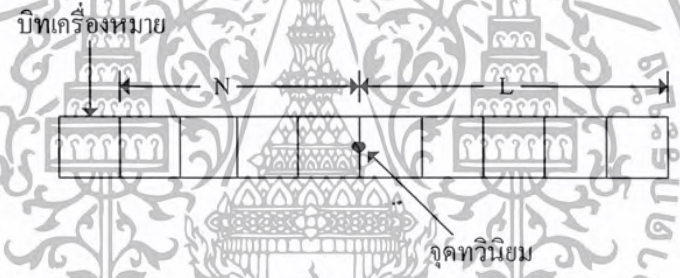
โครงสร้างเลขคณิตกระจาย (Distributed Arithmetic) หรือเรียกอย่างย่อๆว่า "DA" เป็นการจัดรูปแบบทางคณิตศาสตร์ของสมการที่อยู่ในรูปของผลบวกของผลคูณ (Sum of Product) ของเลขฐานสิบให้กระจายออกเป็นบิทหรือในรูปแบบของเลขฐานสอง เพื่อให้การคำนวณทางคณิตศาสตร์สามารถแปลงเป็นวงจรดิจิตอลอิเล็กทรอนิกส์ได้ โดยจะปรากฏอยู่ในงานทางด้านการประมวลผลสัญญาณเชิงเลข หลักการเบื้องต้นของโครงสร้างเลขคณิตกระจายที่นำมาใช้งานด้านการประมวลผลสัญญาณเชิงเลข คือการแปลงฟังก์ชันถ่ายโอน (Transfer Function) ซึ่งเป็นสมการที่อยู่ในรูปผลบวกของผลคูณระหว่างสัญญาณอินพุทกับค่าสัมประสิทธิ์ของวงจรกรอง โดยในการคูณกันระหว่างค่าสัมประสิทธิ์ของวงจรกรองกับสัญญาณอินพุท จะใช้การคูณเลขฐานสองแบบส่วนเติมเต็มสอง(2's complement) และการคูณจะใช้แบบเปิดตาราง (Look-up table) โดยค่าผลบวกของผลคูณระหว่างสัมประสิทธิ์และสัญญาณอินพุทจะถูกเก็บไว้ในหน่วยความจำ EPROM และจะใช้สัญญาณอินพุทเป็นแอดเดรสของ EPROM โดยตรง ทั้งค่าสัมประสิทธิ์ของวงจรกรองและสัญญาณอินพุทจะถูกแทนด้วยเลขส่วนเติมเต็มสอง ดังนั้นโครงสร้างเลขคณิตกระจายจึงมีทฤษฎีพื้นฐานอยู่บนการคูณแบบเลขส่วนเติมเต็มสอง (2's complement Multiplication)

2.2.1 ระบบตัวเลข

สำหรับระบบเชิงเลข ตัวเลขต่างๆจะถูกแทนด้วยเลขฐานสอง ซึ่งโดยทั่วไปมีรูปแบบที่นิยมใช้กันอยู่ 2 รูปแบบ คือ รูปแบบจำนวนโดยตรง (Fixed point format) และ รูปแบบจำนวนอิงดรรชนี (Floating point format) ซึ่งรูปแบบจำนวนโดยตรงจะมีวงจรฮาร์ดแวร์ที่ใช้ในการคำนวณที่ง่ายกว่า แต่ให้ค่าจากการคูณค่อนข้างจำกัด ส่วนรูปแบบจำนวนอิงดรรชนีจะสามารถแทนค่าของสัญญาณ คือให้ย่านพลวัตร (Dynmic range) ได้มากกว่า แต่ต้องใช้วงจรฮาร์ดแวร์ที่สลับซับซ้อน แพงกว่า และให้ความเร็วในการประมวลผลที่ลดลง

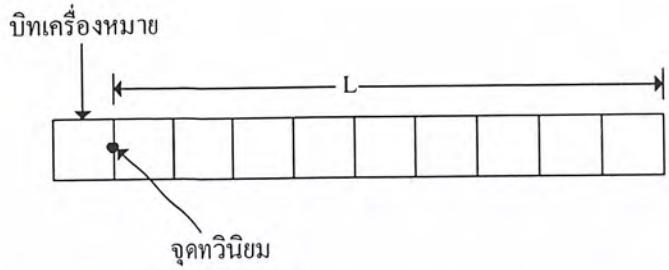
1.) รูปแบบจำนวนโดยตรง

รูปแบบจำนวนโดยตรงปกติจะประกอบไปด้วย 3 ส่วน คือ บิทเครื่องหมาย (Sign bit) 1 บิท บิทจำนวนเต็ม (Integer bit) N บิท และบิทเศษส่วน (Fractional bit) L บิท โดยจะมีจุดทวินิยม (Binary point) อยู่ระหว่างบิทจำนวนเต็มและบิทเครื่องหมายดังแสดงในรูปที่ 2.3



รูปที่ 2.3 การจัดรูปแบบจำนวนโดยตรงที่ประกอบด้วยบิทจำนวนเต็มและบิทเศษส่วน

จำนวนบิท N เป็นตัวกำหนดย่านพลวัตรที่ต้องการ โดยถ้าเลือกให้มีจำนวนน้อยอาจทำให้เกิดการล้น (Overflow) จากการคำนวณได้ แต่ถ้าเลือกให้มีจำนวนมากความเที่ยงตรงก็จะน้อยลง ซึ่งในการสร้างวงจรกรองสัญญาณเชิงเลขโดยการแทนด้วยรูปแบบจำนวนโดยตรงนั้น นิยมที่จะทำมาตราส่วน (Scaling) เพื่อให้ขนาดของสัญญาณมีค่าอยู่ระหว่าง $-1 \le x < 1$ คือมีบิทเครื่องหมาย 1 บิท และบิทเศษส่วน L บิท ดังแสดงในรูปที่ 2.4



รูปที่ 2.4 การจัดรูปแบบจำนวนโดยตรงที่มีแต่บิทเศษส่วน

โดยทั่วไปเลขฐานสองแบบจำนวนโดยตรงแบ่งออกได้เป็น 3 รูปแบบด้วยกัน คือ (1) แบบขนาดและเครื่องหมาย (Sign magnitude) (2) แบบส่วนเติมเต็มหนึ่ง (1's complement) (3) แบบส่วนเติมเต็มสอง (2's complement) โดยคุณลักษณะที่สำคัญบางประการของการแทนตัวเลขด้วยเลขฐานสองแบบจำนวนโดยตรงทั้ง 3 รูปแบบสามารถสรุปได้ดังตารางที่ 2.1

ตารางที่ 2.1 คุณสมบัติที่สำคัญของรูปแบบจำนวนโดยตรง

Features	Sign and magnitude	2' complement	1'complement
Range	$-(1-2^{-L}) \le x \le (1-2^{-L})$	$-1 \le x \le (1-2^{-L})$	$-(1-2^{-L}) \le x \le (1-2^{-L})$
Representation of zero	0.000 and 1.000	0.000	0.000 and 1.111
Arithmetic rules	Simple must be kept track of, separately	Simple; negative numbers elegantly handled	Simple, but "end around carry" should be carefully handled
Suitability for serial arithmetic	Not so good	Excellent	Good

ใน 3 รูปแบบนี้ตัวเลขแบบส่วนเติมเต็มสองเป็นที่นิยมใช้กันมากในระบบการประมวลผลสัญญาณเชิงเลข ทั้งนี้เนื่องมาจาก

1. มีการแทนค่าเลขศูนย์ได้เพียงค่าเดียว

2. การสร้างวงจรฮาร์ดแวร์สำหรับการบวก ลบ และคูณ ของเลขส่วนเติมเต็มสองทำได้ง่ายโดยในการคูณสามารถใช้หลักการเลื่อนและบวก (Shift and add)

3. ในระหว่างผลการบวกย่อย (Partial sum) ของการบวกเลขส่วนเติมเต็มสอง สามหรือสี่จำนวน ถึงแม้อาจจะเกิดการล้น (ตัวทดจากผลการบวกล้นข้ามไปทับบิทเครื่องหมาย) แต่ผลลัพธ์สุดท้ายมักให้ค่าถูกต้องเสมอ ถ้าผลบวกอยู่ในช่วง -1 ถึง $1-2^{-L}$ ดังตัวอย่าง

7/8	0.111	
+4/8	0.100	
11/8	1.011	ผลบวกย่อยที่ผิดเนื่องจากเกิดการล้น
6/8	1.010	
5/8	0.101	ผลบวกที่ถูกต้อง

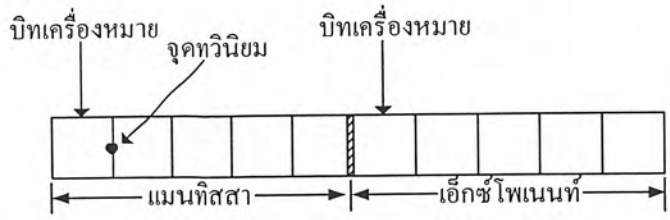
2.) รูปแบบจำนวนอิงดรรชนี

รูปแบบจำนวนโดยตรงมีข้อเสียที่สำคัญ 2 ประการ คือ (1) ย่านพลวัตรของตัวเลขมีค่าน้อย เช่น การแทนด้วยเลขส่วนเติมเต็มสอง ค่าที่น้อยที่สุดคือ -1 และค่าที่มากที่สุด คือ $1-2^{-L}$ เปอร์เซ็นต์ความผิดพลาดที่เกิดจากการตัด (Truncation) หรือการปัด (Rounding) จะเพิ่มมากขึ้นเมื่อขนาดของตัวเลขมีค่าลดลง ตัวอย่างเช่น ถ้าจำนวน 0.11011010 และ 0.000110101 ถูกตัดให้จำนวนบิทเศษส่วนเหลือเพียง 4 บิท เปอร์เซ็นต์ความผิดพลาดจะเป็น 4.59 % และ 39.6 % ตามลำดับ โดยข้อเสียนี้สามารถแก้ไขได้โดยการใช้รูปแบบจำนวนอิงดรรชนี ซึ่งตัวเลข X แสดงได้โดย

$$X = M \times 2^{e} \tag{2.8}$$

โดย e เป็นจำนวนเต็ม และ $\frac{1}{2} \le |M| < 1$

M และ e เรียกว่า แมนทิสสา (Mantissa) และ เอ็กซ์โพเนนท์ (Exponent) ตามลำดับ ตัวอย่างเช่น จำนวน 0.00110101 และ 01001.11 สามารถแทนได้โดย 0.110101×2^{-2} และ 0.100111×2^{4} ตามลำดับ ส่วนจำนวนที่มีค่าเป็นลบก็ทำในลักษณะเดียวกัน รูปแบบจำนวนอิงดรรชนีสามารถแสดงได้ดังรูปที่ 2.5 โดยแบ่งเป็น 2 ส่วน คือส่วนหนึ่งสำหรับแมนทิสสา และอีกส่วนสำหรับเอ็กซ์โพเนนท์



รูปที่ 2.5 การจัดรูปแบบจำนวนอิงดรรชนี

ข้อดีของการใช้จำนวนอิงดรรชนี คือแทนค่าของสัญญาณได้ละเอียดกว่า และแม่นยำกว่าแบบจำนวนโดยตรง แต่การบวก ลบ หรือคูณจะยุ่งยากกว่ามาก วงจรจึงซับซ้อนและแพงกว่าแบบจำนวนโดยตรงมาก นอกจากนี้ความเร็วในการประมวลผลยังช้ากว่าด้วย ดังนั้นสำหรับการประมวลผลแบบเวลาจริง (Real time) จึงนิยมใช้ระบบตัวเลขแบบจำนวนโดยตรง

2.2.2 ทฤษฎีเลขคณิตกระจาย

จากที่ได้กล่าวมาแล้วว่า โครงสร้างเลขคณิตกระจายมีพื้นฐานอยู่บนการคูณแบบเลขส่วนเติมเต็มสอง ดังนั้นในหัวข้อนี้จะได้อธิบายถึงหลักการของการคูณเลขส่วนเติมเต็มสอง

ให้เลขส่วนเติมเต็มสองของ X ซึ่งแทนด้วย $\overline{X}$ และนิยามโดย

$$\overline{X} = \begin{cases} X & \text{ถ้า } X \geq 0 \\ 2 - |X| & \text{ถ้า } X < 0 \end{cases} \tag{2.9}$$

โดย X เป็นเลขที่เป็นเศษส่วน (Fractional number)

ในระบบเลขส่วนเติมเต็มสองจะใช้บิทที่มีนัยสำคัญสูงสุด (MSB) เป็นบิทแสดงเครื่องหมาย โดยถ้าเป็นบวกแทนด้วย “0” และถ้าเป็นลบแทนด้วย “1” ถ้าให้ X แทนด้วยเลขฐานสองขนาด L+1 บิท ดังนั้นรูปแบบของเลขส่วนเติมเต็มสองจะเขียนได้ดังนี้

$$\overline{X} = X_0.X_1.X_2...X_L \tag{2.10}$$

ค่าของ $\overline{X}$ ในรูปของเลขฐานสิบสามารถหาได้ดังนี้

$$X = -X_0 + \sum_{i=1}^{L} X_i 2^{-i} \tag{2.11}$$

จากนั้นพิจารณาผลคูณต่อไปนี้

$$Y = X \times m \tag{2.12}$$

ให้ $\overline{Y}$, $\overline{X}$ และ $\overline{m}$ เป็นเลขส่วนเติมเต็มสองของ Y, X และ m ตามลำดับ จากนั้นพิจารณาจากสมการที่ (2.11) และ สมการที่ (2.12) จะได้

$$\begin{aligned} Y &= -Y_0 + \sum_{i=1}^{L} Y_i 2^{-i} \\ &= -X_0 m + \sum_{i=1}^{L} X_i m 2^{-i} \end{aligned} \tag{2.13}$$

ดังนั้น

$$\begin{aligned} \overline{Y} &= \text{ส่วนเติมเต็มสองของ} \left(-X_0 m + 2^{-1} X_1 m + 2^{-2} X_2 m + 2^{-3} X_3 m + \ldots + 2^{-L} X_L m \right) \\ &= \text{ส่วนเติมเต็มสองของ} \left(-X_0 m + 2^{-1} (X_1 m + \ldots + 2^{-1} (X_{L-1} m + 2^{-1} (X_L m))) \right) \end{aligned} \tag{2.14}$$

ต่อไปพิจารณาส่วนเติมเต็มสองของ $2^{-1}U$ โดย

$\overline{U} = U_0.U_1U_2...U_M$

สำหรับ $U \geq 0$ (หรือ $U_0 = 0$)

ส่วนเติมเต็มสองของ $(2^{-1}U) = 2^{-1}\overline{U}$

และสำหรับ $U < 0$ (หรือ $U_0 = 1$)

ส่วนเติมเต็มสองของ $(2^{-1}U) = 2 - \left| 2^{-1}U \right| = 1 + 2^{-1}\left(2 - |U|\right) = 1 + 2^{-1}\overline{U}$

ดังนั้นสรุปได้ว่า

$$\text{ส่วนเติมเต็มสองของ} (2^{-1}U) = \begin{cases} 2^{-1}\overline{U} & \text{ถ้า } U_0 = 0 \\ 1 + 2^{-1}\overline{U} & \text{ถ้า } U_0 = 1 \end{cases} \tag{2.15}$$

สมการที่ (2.15) นี้แสดงให้เห็นได้ว่า ส่วนเติมเต็มสองของ $(2^{-1}U)$ เป็นการเลื่อนข้อมูลของ $\overline{U}$ ไปทางขวา 1 บิท

$$\therefore \text{ส่วนเติมเต็มสองของ} (2^{-1}U) = 2_2^{-1}\overline{U} \tag{2.16}$$

โดย $2_2^{-1}\overline{U}$ แสดงถึงการเลื่อนข้อมูลของ $\overline{U}$ ไปทางขวา 1 บิท แบบเลขส่วนเติมเต็มสอง ซึ่งสัญลักษณ์ 2_2^{-1} (ซึ่งโดยทั่วไปนิยมเขียนเป็น 2^{-1}) เป็นการแสดงว่าในกรณีที่ $\overline{U}$ เป็นเลขบวก ซึ่งบิทเครื่องหมายจะเป็นเลขศูนย์ โดยหลังจากเลื่อนข้อมูลไปทางขวา 1 บิทแล้ว บิทที่มาแทนบิทเครื่องหมายก็ยังคงเป็นเลขศูนย์ ส่วนในกรณีที่ $\overline{U}$ เป็นเลขลบ หลังจากเลื่อนข้อมูลไปทางขวา 1 บิทแล้ว บิทที่มาแทนบิทเครื่องหมายจะเป็นเลขหนึ่ง (จาก $1+2^{-1}\overline{U}$) ซึ่งในการสร้างเพื่อใช้งานจริงจำเป็นจะต้องมีวงจรที่ใช้ในการตรวจสอบบิทเครื่องหมาย (Sign digit) ทุกครั้งที่มีการเลื่อนข้อมูล

จากนั้นพิจารณาสมการที่ (2.14) และสมการที่ (2.15) จะได้ว่า

$$\begin{aligned}\overline{Y} &= -X_0\overline{m} + 2^{-1}X_1\overline{m} + 2^{-2}X_2\overline{m} + 2^{-3}X_3\overline{m} + \ldots + 2^{-L}X_L\overline{m} \\ &= -X_0\overline{m} + 2^{-1}(X_1\overline{m} + \ldots + 2^{-1}(X_{L-1}\overline{m} + 2^{-1}(X_L\overline{m})))\end{aligned} \qquad (2.17)$$

ซึ่งจากสมการที่ (2.17) จะเห็นได้ว่าผลคูณจากสมการที่ (2.5) สามารถหาได้โดยการใช้หลักการเลื่อนและบวก (Shift and add) โดยผลลัพธ์ที่ได้จากการคูณแบบเลขส่วนเติมเต็มสอง สามารถหาได้ตามขั้นตอนดังนี้

1. เคลียร์ค่าข้อมูลในแอคคิวมูเลเตอร์รีจิสเตอร์
2. บวก $X_L\overline{m}$ กับค่าที่อยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์
3. เลื่อนค่าที่อยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์ไปทางขวา 1 บิท
4. ทำซ้ำข้อ 2 และ 3 สำหรับค่า $X_{L-1},\ldots,X_1$
5. ลบค่า $X_0\overline{m}$ ออกจากค่าที่อยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์ (ลบแบบเลขส่วนเติมเต็มสอง)

ตัวอย่างการทำงานตามอัลกอริธึมนี้

$Y = Xm = 0.8125(-0.390625)$ โดยสมมุติให้ใช้แอคคิวมูเลเตอร์รีจิสเตอร์ขนาด 12 บิท

$m = -0.390625$

$\overline{m} = 2 - |m|$ m เป็นเลขลบ

$= 2 - 0.390625$

$= 1.609375$

$\therefore \overline{m} = 1.100111$

$X = 0.8125 = \overline{X}$ $\because X$ เป็นเลขบวก

$\therefore \overline{X} = 0.1101 = X_0.X_1X_2X_3X_4$

โดยมีขั้นตอนการทำงาน ดังตารางต่อไปนี้

ตารางที่ 2.2 ขั้นตอนการคูณเลขส่วนเติมเต็มสอง

การดำเนินการ	ข้อมูลในแอคคิวมูเลเตอร์รีจิสเตอร์
เคลียร์ ACC	0.000 0000 0000
ACC+ $X_4\overline{m}$	1.100 1110 0000
เลื่อนข้อมูลใน ACC ไปทางขวา 1 บิท	1.110 0111 0000
ACC+ $X_3\overline{m}$	1.110 0111 0000
เลื่อนข้อมูลใน ACC ไปทางขวา 1 บิท	1.111 0011 1000
ACC+ $X_2\overline{m}$	1.100 0001 1000
เลื่อนข้อมูลใน ACC ไปทางขวา 1 บิท	1.110 0000 1100
ACC+ $X_1\overline{m}$	1.010 1110 1100
เลื่อนข้อมูลใน ACC ไปทางขวา 1 บิท	1.101 0111 0110
ACC- $X_0\overline{m}$	1.101 0111 0110

$\therefore \overline{Y} = 1.101\ \ 0111\ \ 0110 = Y_0.Y_1Y_2...Y_{11}$

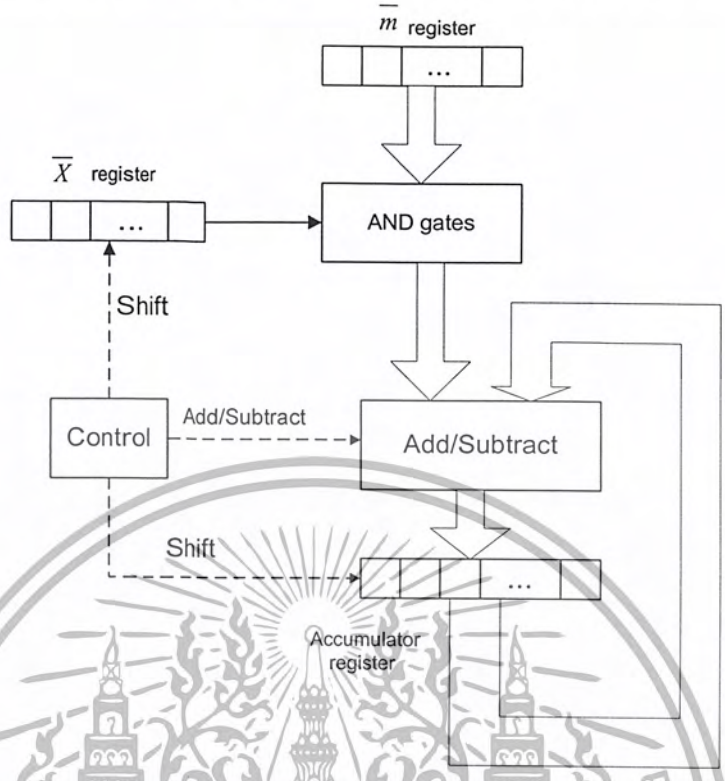
จะได้

$$Y = -Y_0 + \sum_{i=1}^{11} Y_i\, 2^{-i}$$

$$= -1 + \left(2^{-1} + 2^{-3} + 2^{-5} + 2^{-6} + 2^{-7} + 2^{-9} + 2^{-10}\right)$$

$$= -0.3173828125$$

จากอัลกอริธึมดังกล่าวสามารถออกแบบการทำงานและสร้างวงจรแสดงได้ดังรูปที่ 2.6



รูปที่ 2.6 การคูณแบบเลขส่วนเติมเต็มสองโดยใช้เลขคณิตกระจาย

ที่ผ่านมาเป็นหลักการคูณแบบเลขส่วนเติมเต็มสอง ส่วนทฤษฎีเลขคณิตกระจายก็อาศัยหลักการดังกล่าวมาใช้ โดยทำการกระจายสมการที่อยู่ในรูปผลบวกของผลคูณให้แตกออกมาอยู่ในระดับบิท (Bit level)

พิจารณาผลบวกของผลคูณต่อไปนี้

$$Y = \sum_{i=0}^{N} m_i X_i \tag{2.18}$$

โดย m_i เป็นค่าสัมประสิทธิ์ซึ่งทีค่าคงที่

X_i เป็นข้อมูลอินพุท

ถ้า X_i แต่ละค่าเป็นเลขส่วนเติมเต็มสอง โดย $|X_i|<1$ สามารถแสดง X_i แต่ละค่าได้ดังนี้

$$X_i = -X_{i0} + \sum_{j=1}^{L} X_{ij} 2^{-j} \tag{2.19}$$

โดย X_{ij} = บิทต่างๆของข้อมูล X_i มีค่าเป็น 0 หรือ 1

X_{i0} = บิทแสดงเครื่องหมาย

X_{iL} = บิทที่มีนัยสำคัญต่ำสุด (LSB)

$L+1$ = จำนวนบิทที่แทนข้อมูลอินพุท

แทนค่า X_i ในสมการที่ (2.19) ลงในสมการที่ (2.18) จะได้

$$Y = \sum_{i=0}^{N} m_i \left[-X_{i0} + \sum_{j=1}^{L} X_{ij} 2^{-j} \right] \tag{2.20}$$

จัดเทอมของผลบวกใหม่จะได้

$$\begin{aligned} Y &= -X_{i0} \sum_{i=0}^{N} m_i + \sum_{j=1}^{L} X_{ij} 2^{-j} \sum_{i=0}^{N} m_i \\ &= -\sum_{i=0}^{N} X_{i0} m_i + \sum_{i=0}^{N} \sum_{j=1}^{L} X_{ij} m_i 2^{-j} \end{aligned} \tag{2.21}$$

จากนั้นทำการกระจายออกให้เป็นระดับบิท ได้ดังนี้

$$\begin{aligned} Y = &-(X_{00} m_0 + X_{10} m_1 + X_{20} m_2 + \ldots + X_{N0} m_N) \\ &+ 2^{-1}(X_{01} m_0 + X_{11} m_1 + X_{21} m_2 + \ldots + X_{N1} m_N) \\ &+ 2^{-2}(X_{02} m_0 + X_{12} m_1 + X_{22} m_2 + \ldots + X_{N2} m_N) \\ &+ \ldots + 2^{-L}(X_{0L} m_0 + X_{1L} m_1 + X_{2L} m_2 + \ldots + X_{NL} m_N) \end{aligned} \tag{2.22}$$

สมการที่ (2.22) นี้ถูกกระจายออกให้อยู่ในรูปผลบวกของผลคูณระหว่างสัมประสิทธิ์กับข้อมูลอินพุทในระดับบิท ซึ่งเป็นนิยามของการคำนวณแบบเลขคณิตกระจาย และเมื่อเปรียบเทียบสมการที่ (2.22) กับสมการที่ (2.17) จะเห็นว่าการคำนวณหาค่า Y ก็ใช้เลขคณิตกระจายนั่นเอง เพียงแต่นำค่าผลคูณย่อย (Partial product) ที่คำนวณไว้ล่วงหน้าแล้วสำหรับแต่ละค่าที่สอดคล้องกับแต่ละบิทของข้อมูลอินพุทไปเก็บไว้ในตารางเปิดดู ซึ่งเป็นหน่วยความจำ EPROM และใช้ข้อมูลอินพุทเป็นแอดเดรสของหน่วยความจำ เพื่อนำค่าในตารางเปิดดูมาผ่านขั้นตอนการคำนวณตามบูทอัลกอริธึม ซึ่งค่าในตารางเปิดดูสามารถแสดงได้ดังนี้

ตารางที่ 2.3 ค่าผลคูณย่อยที่เก็บไว้ในตารางเปิดดูที่กำหนดโดยข้อมูลอินพุท

Bit pattern ของข้อมูลอินพุท X_{Nj} ..... X_{2j} X_{1j} X_{0j}	ผลคูณย่อยที่เก็บไว้ในตารางเปิดดู
0 ..... 0 0 0	0
0 ..... 0 0 1	m_0
0 ..... 0 1 0	m_1
0 ..... 0 1 1	$m_1 + m_0$
0 ..... 1 0 0	m_2
0 ..... 1 0 1	$m_2 + m_0$
0 ..... 1 1 0	$m_2 + m_1$
0 ..... 1 1 1	$m_2 + m_1 + m_0$
⋮	⋮
1 ..... 1 1 1	$m_N + m_{N-1} + ... + m_2 + m_1 + m_0$

2.2.3 การนำโครงสร้างเลขคณิตกระจายมาใช้กับวงจรกรองสัญญาณเชิงเลข

ในการนำโครงสร้างเลขคณิตกระจายมาใช้ในการออกแบบสำหรับวงจรกรองสัญญาณเชิงเลขนั้น เพื่อความสะดวกจะใช้สมการผลต่างสืบเนื่องอันดับที่สอง (Second order difference equation) มาพิจารณาเพื่อเป็นพื้นฐานในการสร้างวงจรกรองที่มีอันดับที่สูงขึ้นต่อไป

พิจารณาสมการผลต่างสืบเนื่องอันดับสองดังนี้

$$Y(n) = a_0X(n) + a_1X(n-1) + a_2X(n-2) - b_1Y(n-2) - b_2Y(n-2) \qquad (2.23)$$

แทนลำดับสัญญาณอินพุท X(n) และลำดับสัญญาณเอาต์พุท Y(n) ด้วยเลขส่วนเติมเต็มสองได้ดังนี้

$$\overline{X}(n) = X_0(n).X_1(n).X_2(n)...X_L(n)$$

$$\overline{Y}(n) = Y_0(n).Y_1(n).Y_2(n)...Y_L(n)$$

และให้ $\overline{a}_i$ และ $\overline{b}_i$ เป็นเลขส่วนเติมเต็มสองของ a_i และ b_i ตามลำดับ

X(n) และ Y(n) สามารถแสดงได้โดย

$$X(n) = -X_0(n) + \sum_{i=1}^{L} X_i(n)2^{-i}$$

$$Y(n) = -Y_0(n) + \sum_{i=1}^{L} Y_i(n)2^{-i}$$

นำค่า $X(n)$ และ $Y(n)$ แทนลงในสมการที่ (2.23) ได้

$$Y(n) = \sum_{i=1}^{L} 2^{-1}[a_0X_i(n) + a_1X_i(n-1) + a_2X_i(n-2) - b_1Y_i(n-1) - b_2Y_i(n-2)] - [a_0X_0(n) + a_1X_0(n-1) + a_2X_0(n-2) - b_1Y_0(n-1) - b_2Y_0(n-2)] \quad (2.24)$$

คูณ 2^{-1} ทั้ง 2 ข้างได้

$$2^{-1}Y(n) = \sum_{i=1}^{L} 2^{-1}[2^{-1}a_0X_i(n) + 2^{-1}a_1X_i(n-1) + 2^{-1}a_2X_i(n-2) - 2^{-1}b_1Y_i(n-1) - 2^{-1}b_2Y_i(n-2)] - [2^{-1}a_0X_0(n) + a_1X_0(n-1) + 2^{-1}a_2X_0(n-2) - 2^{-1}b_1Y_0(n-1) - b_2Y_0(n-2)] \quad (2.25)$$

พิจารณาสมการที่ (2.25) และสมการที่ (2.16) จะได้

$$2^{-1}\overline{Y}(n) = \sum_{i=1}^{L} 2^{-1}[2^{-1}\overline{a}_0X_i(n) + 2^{-1}\overline{a}_1X_i(n-1) + 2^{-1}\overline{a}_2X_i(n-2) - 2^{-1}\overline{b}_1Y_i(n-1) - 2^{-1}\overline{b}_2Y_i(n-2)] - [2^{-1}\overline{a}_0X_0(n) + \overline{a}_1X_0(n-1) + 2^{-1}\overline{a}_2X_0(n-2) - 2^{-1}\overline{b}_1Y_0(n-1) - \overline{b}_2Y_0(n-2)] \quad 2.26)$$

เพราะฉะนั้น

$$\overline{Y}(n) = \sum_{i=1}^{L} 2_2^{-i}F_i - F_o \quad (2.27)$$

โดย

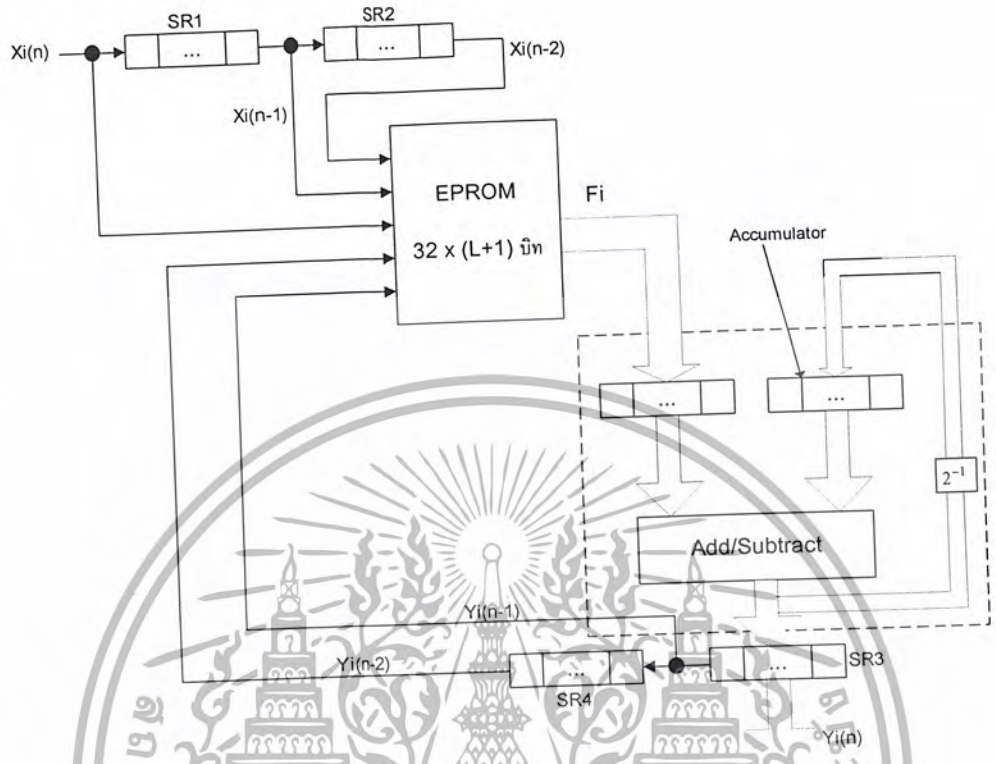
$$F_i = \overline{a}_0X_i(n) + \overline{a}_1X_i(n-1) + \overline{a}_2X_i(n-2) - \overline{b}_1Y_i(n-1) - \overline{b}_2Y_i(n-2) \quad (2.28)$$

$$F_o = \overline{a}_0X_0(n) + \overline{a}_1X_0(n-1) + \overline{a}_2X_0(n-2) - \overline{b}_1Y_0(n-1) - \overline{b}_2Y_o(n-2) \quad (2.29)$$

ส่วนเติมเต็มสองของ $Y(n)$ สามารถหาได้โดยใช้อัลกอริธึมดังนี้

1. เคลียร์ค่าของข้อมูลในแอคคิวมูเลเตอร์รีจิสเตอร์
2. คำนวณค่า F_i สำหรับ $i = L$
3. บวกค่า F_i กับค่าที่บรรจุอยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์
4. เลื่อนค่าที่บรรจุอยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์ไปทางขวา1บิต (เลื่อนข้อมูลแบบเลขส่วนเติมเต็มสอง)
5. ทำซ้ำข้อ2 ถึง 4 สำหรับ $i = L-1, L-2,\ldots,1$
6. คำนวณค่า F_o
7. ลบค่า F_o ออกจากค่าที่บรรจุอยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์ (ลบแบบเลขส่วนเติมเต็มสอง)

โดยอัลกอริธึมที่กล่าวมาสามารถออกแบบการทำงานและสร้างวงจรดังแสดงได้ดังรูปที่ 2.7



รูปที่ 2.7 โครงสร้างวงจรกรองสัญญาณป้อนกลับเชิงเลขอันดับที่สอง

จากรูปที่ 2.7 ส่วนที่อยู่ในเส้นประนิยมเรียกกันว่าวงจรสเกลลิ่งแอคคิวมูเลเตอร์ (Scaling Accumulator) โดยค่าที่อยู่ในแอคคิวมูเลเตอร์ ก่อนที่จะส่งไปบวกกับค่าผลลัพธ์จาก EPROM (หรือ partial sum ตัวต่อไป) จะต้องทำการเลื่อนข้อมูลไปทางขวา 1 บิทก่อน ดังที่กล่าวมาแล้ว ซึ่งการเลื่อนข้อมูลไปทางขวา 1 บิทนี้ เขียนแทนด้วยการคูณด้วย 2^{-1} และผลจากสมการที่ (2.28) นำมาสร้างเป็นตารางเปิดดูบรรจุไว้ใน EPROM ค่าในตารางเปิดดูเป็นค่าของ F_i ซึ่งเกิดจากตัวแปรที่เป็นลำดับสัญญาณอินพุท 5 ตัว ดังนั้นค่าของ F_i จะมีค่า $2^5 = 32$ ค่า โดยขนาดของ EPROM จะมีขนาด $32 \times (L+1)$ บิท

2.2.4 โครงสร้างของวงจรกรอง 2 มิติโดยวิธีการกระจายทางคณิตศาสตร์

ตัวกรอง 2 มิติเชิงเลขมีประโยชน์ประยุกต์ใช้ โดยเฉพาะเป็นตัวเพิ่มคุณภาพหรือลดสัญญาณรบกวนของสัญญาณในลักษณะ 2 มิติของระบบการประมวลสัญญาณเชิงเลข เช่น สัญญาณภาพรังสีเอ็กซ์ สัญญาณภาพจากดาวเทียม เป็นต้น สัญญาณเหล่านี้จะต้องพิจารณาเป็นลักษณะช่วงๆสัญญาณ ซึ่งส่งมาเป็นลำดับสามารถเขียนให้อยู่ในรูปของฟังก์ชั่นได้โดยมีตัวแปรจำนวนเต็ม 2 ตัว เช่น เป็นตัวแสดงลำดับของสัญญาณซึ่งถูกกำหนดให้ค่าทุกค่าเป็นจำนวนเต็ม การประยุกต์ DA กับวงจรกรองสัญญาณสองมิติ

สมการความแตกต่างเชิงเส้น (Linear Differential Equation) ของวงจรกรองสัญญาณสองมิติ (Two-dimension Second Order Digital Filter) โดยทั่วไปอยู่ในรูปของสมการที่ 2.30

$$Y_{m,n} = \sum_{k=0}^{2}\sum_{l=0}^{2} a_{k,l} X_{m-k,n-l} \tag{2.30}$$

$X_{m,n}$ และ $Y_{m,n}$ เป็นสัญญาณเข้าและสัญญาณออกตามลำดับ $a_{k,l}$ สัมประสิทธิ์ของการกรอง กำหนดให้สัญญาณทุกค่าอยู่ในช่วง +128,-128 ของเลขคอมพลีเมนท์ของ 2 (2's Complement) ความละเอียดขนาด 7 บิท รวมบิทเครื่องหมายอีก 1 บิท กระจายเป็นเลขฐานสองโดยสมการที่ (2.31)

$$X_{m-k,n-l} = \sum_{s=1}^{7} X^{s}_{m-k,n-l} 2^{-s} - X^{0}_{m-k,n-l} \tag{2.31}$$

แทนค่าสมการที่ (2.31) ลงในสมการที่ (2.30) จะได้สมการที่ (2.32)

$$Y_{m,n} = \sum_{s=1}^{7}\left(\sum_{k=0}^{2}\sum_{l=0}^{2}(a_{k,l} X^{s}_{m-k,n-l})2^{-s}\right) - \left(\sum_{k=0}^{2}\sum_{l=0}^{2}(a_{k,l} X^{s}_{m-k,n-l})\right) \tag{2.32}$$

การประมวลผลข้อมูลภาพเป็นการทำให้ข้อมูลภาพ หรือภาพใหม่ออกมาตามวัตถุประสงค์ต่างๆ ไม่ว่าจะเป็นการตกแต่งภาพให้ชัดขึ้น การเน้นขอบของภาพให้คมหรืออาจจะเป็นการตรวจหาขอบภาพ เป็นต้น ซึ่งวิธีการต่างๆเหล่านี้มีวิธีการแบบเดียวกันทั้งสิ้น ขึ้นอยู่กับว่าจะกำหนดอิมพัลส์เรสพอนส์ (Impulse response) ให้มีค่าเป็นเท่าใด ขนาดเท่าไร โดยใช้แผ่นข้อมูล (Mask) วางทาบไปบนข้อมูลภาพแล้วเคลื่อนที่ไปในตำแหน่งต่างๆของภาพทีละจุด ในแต่ละครั้งข้อมูลทั้งหมดที่อยู่ในบริเวณนี้ที่ตำแหน่งเดียวกันจะถูกนำมาคูณกันผลลัพธ์ที่ได้จะนำมารวมกันเพื่อหาขนาดแล้วเก็บเป็นข้อมูลใหม่ แผ่นข้อมูลจะมีค่าไม่เหมือนกันแล้วแต่ความต้องการ

1.) การกระจายทางคณิตศาสตร์กับการกรองความถี่ต่ำผ่าน

ปริมาณสัญญาณภาพส่วนใหญ่จะรวมส่วนที่เป็นสัญญาณความถี่ต่ำไว้มากกว่าส่วนที่เป็นความถี่สูง ส่วนที่เป็นความถี่ต่ำที่เหมาะสมคือ จุดแต่ละจุดจะต้องมีความสัมพันธ์กันกับจุดที่อยู่ข้างเคียง ถ้าไม่สัมพันธ์กันภาพจะไม่คมชัดโดยเหตุที่ว่าสัญญาณรบกวนที่สุ่มขึ้นมา (Random noise) มีช่วงความกว้างของความถี่ที่แผ่ออกไปจากกลุ่มของความถี่ปกติ ตัวกรองความถี่ต่ำผ่านจะทำการลดส่วนที่เป็นความถี่สูงขณะที่จะรักษาส่วนที่เป็นความถี่ต่ำเอาไว้ และจะลดสัญญาณรบกวนที่มีปริมาณมากให้น้อยลง

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

รูปที่ 2.8 ผลตอบสนองทางอิมพัลส์ของตัวกรองความถี่ต่ำผ่าน

2.) การกระจายทางคณิตศาสตร์กับ Sobel Edge Detector

Sobel edge detector เป็นกระบวนการทำให้ข้อมูลภาพในส่วนที่เป็นเส้นขอบรูป (Contour) ของวัตถุในภาพมีลักษณะเด่นชัดขึ้นโดยจะไม่คำนึงถึงข้อมูลภาพในส่วนอื่นมากนัก วิธีการทำได้โดยใช้แผ่นข้อมูล (mask) 2 ชุด ที่มีขนาด 3×3 จุด ซึ่งมีข้อมูลดังรูปที่ 2.10 ทำหน้าที่ Horizontal Edge Detector และข้อมูลรูปที่ 2.11 ทำหน้าที่ Vertical Edge Detector

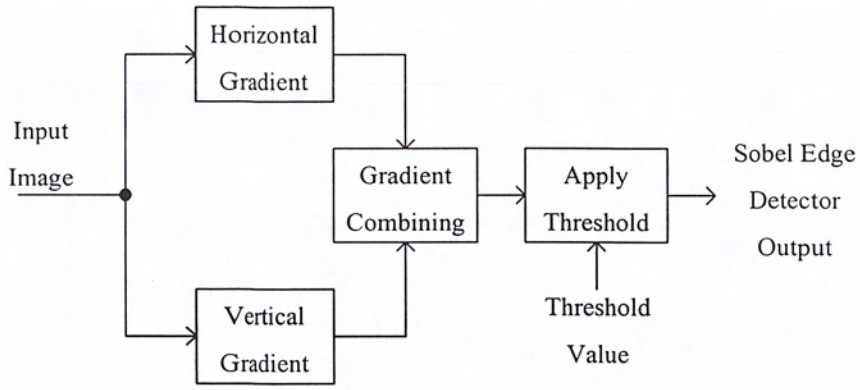
1	0	-1
2	0	-2
1	0	-1

รูปที่ 2.9 ผลตอบสนองทางอิมพัลส์ของ Vertical Edge Detector

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

รูปที่ 2.10 ผลตอบสนองทางอิมพัลส์ของ Horizontal Edge Detector

แผ่นข้อมูลทั้ง 2 ชุดนี้จะวางทาบไปบนข้อมูลภาพแล้วเคลื่อนที่ไปในตำแหน่งต่างๆของภาพทีละจุด ในแต่ละครั้งข้อมูลทั้งหมดที่อยู่ในบริเวณนี้ที่ตำแหน่งเดียวกันจะถูกนำมาคูณกันผลลัพธ์ที่ได้ทั้ง 9 ค่า จะนำมารวมกันเพื่อหาขนาด (Absolute Value) แล้วเก็บเป็นข้อมูลใหม่ แสดงการทำงานของ Sobel Edge Detector ดังรูปที่ 2.12



รูปที่ 2.11 การทำงานของ Sobel edge detector

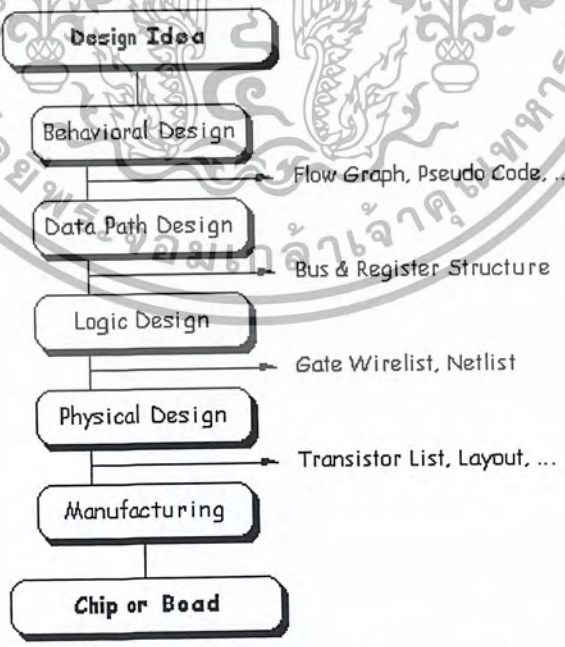
บทที่ 3
การคำนวณและการสร้าง

3.1 ภาษา VHDL

ความซับซ้อนและขนาดของระบบดิจิตอลในปัจจุบันได้เพิ่มมากขึ้นทุกขณะ ส่งผลให้มีการนำคอมพิวเตอร์เพื่อช่วยใน การออกแบบหรือ CAD มาใช้ในขบวนการออกแบบฮาร์ดแวร์เพิ่มขึ้นเช่นกัน อีกทั้งอุปกรณ์และวิธีการ ออกแบบใหม่ๆ ก็ถูกพัฒนาขึ้นมาเพื่อช่วยอำนวยความสะดวกให้กับนักออกแบบมากขึ้นด้วย สำหรับภาษาบรรยายอุปกรณ์ฮาร์ดแวร์ (HDL : Hardware Description Language) ก็เป็นเครื่องมืออย่างหนึ่งที่ได้รับการพัฒนามาอย่างต่อเนื่อง เพื่อช่วยให้การปรับปรุงขบวนการออกแบบระบบดิจิตอลเป็นไปอย่างมีประสิทธิภาพ

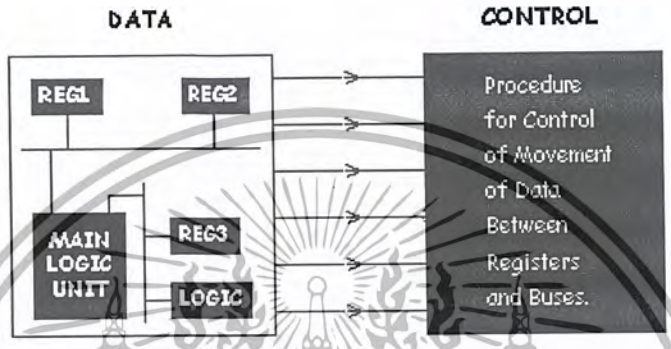
3.1.1 การออกแบบระบบดิจิตอล

ในการออกแบบระบบดิจิตอล เริ่มตั้งแต่การกำหนดแนวความคิดเบื้องต้นจนกระทั่งได้ออกมาเป็นอุปกรณ์ฮาร์ดแวร์ ที่ใช้งานได้จะต้องผ่านขั้นตอนต่างๆ มากมาย และในแต่ละขั้นตอนผู้ออกแบบจะต้องตรวจสอบผลลัพธ์ในแต่ละขั้น ก่อนเข้าสู่กระบวนการออกแบบในขั้นต่อไป รูปที่ 3.1 แสดงขั้นตอนปกติที่ใช้ในการออกแบบระบบดิจิตอลทั่วไป ขั้น แรกผู้ออกแบบจะกำหนดแนวความคิดในการออกแบบแล้วทำการพัฒนาให้สามารถนำมาใช้ได้อย่างสมบูรณ์ ซึ่งภาย ในขั้นตอนนี้ผู้ออกแบบจำเป็นต้องสร้างรูปแบบระบบในเชิงพฤติกรรมขึ้นมาตรวจสอบซึ่งอาจจะเป็นผังงานแสดงแบบหรือ รหัสคำสั่งเทียม (Pseudo code) ก็ได้



รูปที่ 3.1 แสดงขั้นตอนการออกแบบระบบดิจิตอล

ขั้นตอนต่อไปเป็นการออกแบบระบบเส้นทางของข้อมูล ผู้ออกแบบจะกำหนดส่วนประกอบของรีจิสเตอร์และวงจรลอจิก ที่จำเป็นทั้งหมดเพื่อนำมาประกอบเป็นระบบที่สมบูรณ์ โดยแต่ละองค์ประกอบสามารถเชื่อมต่อกันด้วยบัสหนึ่งหรือสอง ทิศทาง (Unidirectional or Bidirectional Bus) ส่วนกระบวนการในการควบคุมการเคลื่อนย้ายข้อมูลระหว่าง รีจิสเตอร์และวงจรลอจิกจะขึ้นอยู่กับพฤติกรรมของระบบที่กำหนดไว้ดังรูปที่ 3.2



รูปที่ 3.2 การออกแบบระบบเส้นทางของข้อมูล

ขั้นตอนถัดมาเป็นการออกแบบวงจรลอจิก ซึ่งจะเกี่ยวข้องกับการนำเกทดิจิตอลพื้นฐานและฟลิปฟลอป (flip-flop) มาประกอบเป็นอุปกรณ์ย่อยต่างๆ เช่น รีจิสเตอร์เก็บข้อมูล บัสวงจรลอจิก และส่วนควบคุมฮาร์ดแวร์ ซึ่งผลลัพธ์ ที่ได้ในขั้นตอนนี้จะเป็นเครือข่ายของการโยงใยระหว่างเกทและฟลิปฟลอปนั่นเองการออกแบบในขั้นตอนถัดไป เป็นการเปลี่ยนเครือข่ายการโยงใยที่ได้จากขั้นตอนที่แล้วให้เป็นลำดับของทรานซิสเตอร์ (Transistor List) และ Layout ซึ่งขั้นตอนนี้จะเกี่ยวข้องโดยตรงกับการจัดวางทรานซิสเตอร์หรือไลบรารีเซลล์เพื่อ แทนเกทและฟลิปฟลอปต่างๆและในขั้นตอนสุดท้ายจะเป็นการส่งระบบที่ออกแบบไว้ไปทำการเจือสารที่โรงงานเพื่อผลิตออกมาเป็น วงจรรวมในที่สุด

3.1.2 ประวัติความเป็นมาของภาษา VHDL

VHDL ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC : Very High Speed Integrated Circuit) เป็นภาษาโปรแกรมระดับสูง (High Level Language) ที่ใช้สำหรับการออกแบบฮาร์ด แวร์ในระบบดิจิตอล ตัวของภาษาสามารถบรรยายพฤติกรรมการทำงานในรูปของลำดับชั้น (Hierarchy) และสามารถเขียนได้หลายรูปแบบ ด้วยเหตุผลนี้จึงทำให้ภาษา VHDL เป็นเครื่องมือที่ใช้ออกแบบตั้งแต่ขั้นตอนบนสุด คือ แนวความคิดที่จะแก้ปัญหา ลงไปทีละขั้นจนถึงขั้นตอนของการสร้างวงจรจริง และตัวภาษาก็เปิดโอกาสให้วิศวกร ได้พัฒนาและจำลองการทำงานของรูปแบบฟังก์ชันการทำงานของวงจรอย่างสังเขป โดยยังไม่ต้องคำนึงถึงรายละเอียดเกี่ยวกับโครงสร้างวงจรจริง นอกจากนั้น VHDL ยังเป็นภาษาที่สนับสนุนลักษณะ

ต่างๆ ของระบบดิจิตอลที่มี ความซับซ้อนได้ทั้งหมด ดังนั้น VHDL จึงเป็นภาษาที่น่าสนใจในการศึกษาและนำไปใช้งานเป็นอย่างยิ่ง วิวัฒนาการของภาษา VHDL เริ่มต้นประมาณปี ค.ศ. 1981 เมื่อกระทรวงกลาโหมสหรัฐอเมริกา หรือ DoD (Department of Defense) ได้พยายามปรับปรุงอุปกรณ์อิเล็กทรอนิกส์และคอมพิวเตอร์ที่ใช้ในกิจการทางทหาร ให้มีความทันสมัยมากขึ้น ประกอบกับเทคโนโลยีทางด้านไมโครอิเล็กทรอนิกส์มีการพัฒนาไปอย่างรวดเร็วดังจะเห็นได้ จากการนำวงจรดิจิตอลหลายๆ วงจรมาทำการผลิตอยู่บนแผ่นซิลิกอนที่มีพื้นที่เพียง 1 - 2 ตารางเซนติเมตรเท่านั้น ซึ่ง เป็นผลให้ประสิทธิภาพในการทำงานของวงจรสูงขึ้นตลอดจนความน่าเชื่อถือ ในการทำงานและความคงทนต่อสภาพ แวดล้อมสูง แต่เนื่องจากในขณะนั้นขั้นตอนของการออกแบบ การผลิต และการตรวจสอบวงจรต้นแบบ เป็นขบวนการที่ ต้องใช้วิศวกรและเวลาในดำเนินการมาก ฉะนั้นทาง DoD จึงจัดตั้งโครงการขึ้นมาเพื่อศึกษาวิธีการที่ช่วยในการพัฒนา วงจรอิเล็กทรอนิกส์ โดยเฉพาะอย่างยิ่งวงจรระบบดิจิตอล ให้สามารถนำไปผลิตได้เร็วขึ้น ซึ่งโครงการดังกล่าวมีชื่อว่า "Very High Speed Integrated Circuits" หรือ VHSIC โดยในระยะแรกนั้นโครงการนี้ถือเป็นความลับทางด้านความมั่งคงของประเทศ และอยู่ภายใต้ความควบคุมดูแลของ United States International Traffic and Arms Regulations (ITAR) สำหรับมาตรฐานของภาษาที่ใช้บรรยาย พฤติกรรมวงจรหรือฮาร์ดแวร์ของระบบสำหรับโครงการ VHSIC ที่ DoDได้ให้ไว้สามารถสรุปได้ดังนี้

- ต้องเป็นภาษาที่นำไปเขียนรูปแบบระบบดิจิตอล และมีคุณสมบัติที่สามารถเข้าใจได้ทั้งมนุษย์และเครื่อง คอมพิวเตอร์โดยไม่ต้องมีการแปลหรือเปลี่ยนแปลงอีก
- สามารถนำไปใช้เป็นเอกสารประกอบโครงการได้
- ต้องเป็นภาษาที่เขียนขึ้นสำหรับใช้จำลองการทำงานของวงจร

ฉะนั้นภาษาดังกล่าวนี้จึงจัดเป็นภาษาโปรแกรมระดับสูง เช่นเดียวกับภาษาปาสคาล หรือภาษาซี ซึ่งในทางวิศวกรรม ภาษาที่ใช้ในการออกแบบฮาร์ดแวร์นี้เรียกว่า "Hardware Description Language" หรือ HDL ในตอนเริ่มแรกนั้น DoD ได้มอบหมายให้บริษัทไอบีเอ็ม เท็กซัสอินสตูเมนท์ และอินเตอร์เมทริกซ์ เป็นผู้ศึกษาและพัฒนา โครงการ ซึ่งการดำเนินงานเป็นไปอย่างต่อเนื่อง จนกระทั่งในปี ค.ศ.1985 ทาง ITAR ได้ยกเลิกข้อจำกัดในการถ่ายทอด เทคโนโลยีทางทหารออกจากโครงการนี้ ดังนั้นภาษา VHDL จึงเริ่มเป็นที่รู้จักกันโดยทั่วไป และประมาณปี ค.ศ. 1987 IEEE ได้ทำการกำหนดมาตรฐานของภาษานี้เป็น IEEE 1076-1987 และมีชื่อเรียกว่า VHDL ซึ่งมาตรฐานนี้ได้รับ การปรับปรุงจนเป็นมาตรฐาน IEEE 1076-1993 หรือ VHDL 1993 เนื่องจากในขณะนั้น DoD เป็นลูกค้ารายใหญ่ ของอุตสาหกรรมอิเล็กทรอนิกส์และคอมพิวเตอร์ ดังนั้นจึงมีผู้รับโครงการต่างๆ จาก DoD ไปดำเนินการวิจัยและพัฒนา เป็นจำนวนมาก และเพื่อให้ทุกโครงการอยู่ในมาตรฐานเดียวกันหมด ดังนั้นทาง DoD จึงได้กำหนดว่า ทุกๆ โครงการต้อง เขียนอยู่ในรูปของภาษา VHDL เท่านั้น ซึ่งทำให้ DoD สามารถนำโครงการเหล่านี้ไปจำลองกับเครื่องคอมพิวเตอร์ได้ หลายๆระบบ

3.1.3 ข้อกำหนด

DoD ได้ตั้งข้อกำหนดสำหรับภาษา VHDL ในเดือนมกราคมปี ค.ศ.1983 ไว้ดังนี้

1.) ลักษณะทั่วไป

DoD ได้กำหนดให้ VHDL เป็นภาษาสำหรับการออกแบบและบรรยายของฮาร์ดแวร์ ซึ่งหมายถึงความสามารถ ในการอธิบายและออกแบบในระดับสูง การจำลอง (Simulation) การสังเคราะห์ (Synthesis) และการทดสอบ (Testing) นอกจากนั้น VHDL ยังถูกกำหนดไว้สำหรับการบรรยายฮาร์ดแวร์ตั้งแต่ระดับบนซึ่งก็คือระบบจนถึง ระดับเกทอีกด้วย เนื่องจากในการทำงานของระบบดิจิตอลนั้น ทุกๆ องค์ประกอบภายในระบบไม่ว่าเล็กหรือใหญ่ จะทำงานไปพร้อมๆ กัน ซึ่งในเรื่องของความพร้อมเพรียงในการทำงานนี้ก็ถือเป็นข้อกำหนดที่สำคัญอย่างหนึ่งของ VHDL ด้วยเช่นกัน (สำหรับในภาษาที่ใช้ในการบรรยายฮาร์ดแวร์นั้นความพร้อมเพรียงจะหมายถึงทุกๆ คำสั่ง องค์ประกอบ เกทหรือวงจรลอจิกจะถูกนำมาปฏิบัติทั้งหมด ดังนั้นในที่สุดแล้วก็จะดูเหมือนว่าได้มีการปฏิบัติไปพร้อมๆ กัน)

2.) สนับสนุนการออกแบบแบบลำดับขั้น

การออกแบบแบบลำดับขั้นเป็นลักษณะที่สำคัญอย่างหนึ่งสำหรับการออกแบบระบบที่มีหลายๆ ระดับ โดยในการ ออกแบบจะประกอบด้วยส่วนการบรรยายการเชื่อมต่อ และส่วนการบรรยายหน้าที่การทำงาน ซึ่งหน้าที่การทำงาน ของระบบสามารถกำหนดได้ด้วยตัวเอง หรืออาจถูกกำหนดโดยโครงสร้างที่ประกอบด้วยองค์ประกอบย่อยๆ ลง ไปได้เช่นกัน แต่ที่ระดับล่างสุด องค์ประกอบต้องถูกบรรยายหน้าที่การทำงานด้วยตัวมันเอง และไม่สามารถกำหนด การทำงาน โดยลักษณะแบบโครงสร้างได้

3.) ไลบรารี

VHDL ได้สนับสนุนการมีไลบรารีเพื่อระบบการจัดการที่ดี ผู้ออกแบบสามารถกำหนดลักษณะและการทำงานของ อุปกรณ์พื้นฐานไว้ในระบบไลบรารี หรือจะใช้ไลบรารีที่ระบบได้จัดเตรียมไว้แล้วก็ได้ โมเดลและการบรรยายที่ถูก ต้องควรจัดเก็บไว้ในไลบรารีหลังจากที่ได้ผ่านการคอมไพล์เรียบร้อยแล้วเพื่อให้ผู้ออกแบบคนอื่นๆ สามารถนำไป ใช้ได้ด้วย

4.) ลำดับคำสั่ง

แม้ว่าการปฏิบัติคำสั่งหรือกระบวนการ โดยพร้อมเพรียงกันจะเป็นคุณสมบัติที่สำคัญของ VHDL ก็ตาม ตัวภาษา เองก็ยังมีการจัดเตรียมลักษณะการควบคุมแบบลำดับคำสั่งไว้ให้ด้วย เมื่อผู้ออกแบบได้กำหนดหน้าที่และองค์ประกอบ ที่ทำงานพร้อมกันของระบบไว้เรียบร้อยแล้ว ผู้ออกแบบยังสามารถบรรยายหน้าที่การทำงานซึ่งเป็นรายละเอียดภายใน ของแต่ละองค์ประกอบได้ในลักษณะเดียวกับการเขียนโปรแกรมที่ประกอบด้วยโครงสร้างแบบ case, if - then - else และ loop ทั่วๆ ไปได้ การบรรยายแบบลำดับคำสั่งทำให้การออกแบบหน้าที่การทำงานของอุปกรณ์กระทำได้ สะดวกและง่ายขึ้น อย่างไรก็ตามโครงสร้างทั้งหมดของ VHDL ก็ยังคงเป็นการทำงานแบบพร้อมเพรียงกันเช่นเดิม

5.) การกำหนดคุณสมบัติ

นอกจากการกำหนดอินพุทและเอาท์พุทแล้ว เงื่อนไขอื่นๆ ก็มีผลต่อการปฏิบัติหน้าที่ของอุปกรณ์ฮาร์ดแวร์ด้วยเช่นกัน โดยสิ่งนี้รวมถึงสภาพแวดล้อมและลักษณะทางกายภาพของอุปกรณ์นั้นๆ ด้วย ซึ่งภาษาสำหรับการออกแบบที่ดีควร ให้ผู้ออกแบบกำหนดคุณสมบัติของอุปกรณ์ที่ใช้ได้ด้วย เช่น สามารถกำหนดขนาด ลักษณะทางกายภาพเวลา โหลด และเงื่อนไขทางสภาพแวดล้อมอื่นๆ ซึ่งความสามารถในการกำหนดคุณสมบัตินี้ก็เป็นส่วนหนึ่งที่มีอยู่ในภาษา VHDL ด้วยเช่นกัน

6.) ชนิดของข้อมูล

VHDL สามารถกำหนดชนิดของข้อมูลไม่เพียงแต่ชนิด BIT และ BOOLEAN เท่านั้น แต่ยังสามารถกำหนดชนิด ของข้อมูลเป็นจำนวนเต็ม จำนวนจริง จุดทศนิยม และชนิดลำดับการนับ (Enumerate Type) หรือแม้แต่ชนิดของ ข้อมูลที่ผู้ออกแบบกำหนดขึ้นมาเองก็ได้

7.) โปรแกรมย่อย

ความสามารถในการใช้ฟังก์ชันและโพรซีเจอร์ (Procedure) ก็เป็นข้อกำหนดอีกอย่างหนึ่งใน VHDL ซึ่งผู้ออกแบบ สามารถนำโปรแกรมย่อยมาใช้ในการเปลี่ยนแปลงชนิดของข้อมูล การกำหนดหน่วยของลอจิก การกำหนดตัวกระทำต่างๆ หรือหน้าที่อื่นๆ ตามที่ต้องการได้เช่นเดียวกับการเขียน โปรแกรมทั่วไป

8.) การควบคุมเวลา

VHDL อนุญาตให้ผู้ออกแบบสามารถกำหนดเวลาในการส่งผ่านข้อมูลหรือสัญญาณได้ตามต้องการ การตรวจสอบ การออกแบบเทสหรือการหน่วงเวลาก็สามารถกระทำได้โดยการกำหนดช่วงเวลาที่แน่นอนหรือกำหนดให้มีการรอคอย เหตุการณ์ (Event) นอกจากนี้ก็ยังสามารถกำหนดรูปแบบของสัญญาณนาฬิกาได้อีกด้วย

9.) การกำหนดแบบโครงสร้าง

การกำหนดโครงสร้างขององค์ประกอบต่างๆ สามารถกระทำได้ในทุกระดับของการออกแบบ โดยการกำหนดโครง สร้างขององค์ประกอบร่วมที่เกิดจากองค์ประกอบย่อยซึ่งแตกต่างกันหรือ เหมือนกันก็เป็นข้อกำหนดอย่างหนึ่งของ VHDL เช่นกัน

3.1.4 องค์ประกอบพื้นฐานของ VHDL

รูปแบบพื้นฐานที่ใช้ในการบรรยายถึงองค์ประกอบของ VHDL จะประกอบไปด้วยส่วนกำหนดการเชื่อมต่อ (Interface) และส่วนกำหนดลักษณะเชิงสถาปัตยกรรม (Architecture) ดังแสดงในรูปที่ 5.3 โดยในการบรรยายการเชื่อมต่อจะขึ้น ต้นด้วยคำว่า ENTITY แล้วตามด้วยชื่อขององค์ประกอบจากนั้นตามด้วยคำว่า IS และถัดมาจะเป็นการบรรยายถึงพอร์ต การติดต่อ อินพุท - เอาท์พุท ขององค์ประกอบ ส่วนลักษณะภายนอกอื่น ๆ เช่น เวลา อุณหภูมิก็สามารถรวมเข้าไปในส่วนนี้ ได้เช่นกัน ในส่วนของการกำหนดลักษณะเชิงสถาปัตยกรรมจะขึ้นต้นด้วยคำว่า ARCHITECTURE ซึ่งเป็นส่วนที่ใช้ บรรยายหน้าที่การทำงานของ

องค์ประกอบ โดยหน้าที่การทำงานนี้จะขึ้นอยู่กับสัญญาณอินพุท เอาท์พุทและพารามิเตอร์ อื่นๆ ที่ได้กำหนดไว้ในส่วนของการเชื่อมต่อดังรูปที่ 3.3 และสำหรับการบรรยายหน้าที่ขององค์ประกอบจะเริ่มต้นหลังจาก คำว่า BEGIN เป็นต้นไป

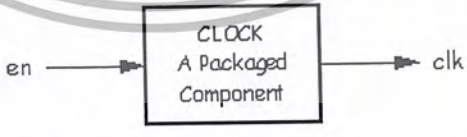
```
ENTITY component_name IS
    Input and output ports
    Physical and other parameters
END [component_name] ;

ARCHITECTURE identifier OF component_name IS
    [declartion]
BEGIN
    specification of the functionality of the component
    in terms of its input lines and as influenced
    by physical and other parameters
END [identifier];
```

รูปที่ 3.3 การกำหนดการเชื่อมต่อและสถาปัตยกรรม

1.) การกำหนดการเชื่อมต่อ

การกำหนดการเชื่อมต่อเป็นระดับบนสุดของการออกแบบ โดยในระดับนี้ต้องกำหนดพอร์ตสำหรับการติดต่อกับองค์ประกอบ ภายนอกอื่นๆ ดังตัวอย่างในรูปที่ 3.4 ซึ่งเป็นบล็อคไดอะแกรม และการบรรยายการเชื่อมต่อขององค์ประกอบสำหรับตัวจ่าย สัญญาณนาฬิกา ในบรรทัดแรกของการบรรยายการเชื่อมต่อเป็นการกำหนดชื่อขององค์ประกอบซึ่งกำหนดเป็น clock_ component ตามด้วยคำว่า PORT และชื่อของพอร์ตอยู่ภายในวงเล็บ ส่วน IN และ OUT เป็นการกำหนด โหนดของสัญญาณให้เป็นอินพุทหรือเอาท์พุท และ BIT เป็นการแสดงชนิดของข้อมูล




```
ENTITY clock_component IS
    PORT (en : IN BIT;ck : OUT BIT)
END clock_name;
```

รูปที่ 3.4 บล็อคไดอะแกรมและการบรรยายการเชื่อมต่อของ clock_ component

2.) การกำหนดรูปแบบการบรรยาย

หน้าที่การทำงานขององค์ประกอบจะถูกบรรยายภายในส่วนนี้ ซึ่งในการบรรยายสามารถกำหนดค่าของสัญญาณ เอาท์พุทในเทอมของอินพุทหรือในรูปขององค์ประกอบอื่นๆ หรือทั้งสองอย่างรวมกันก็ได้ ดังตัวอย่างการบรรยายของ clock_component ในรูปที่ 3.5 ซึ่งเป็นการบรรยายในเชิงพฤติกรรมโดยมี en เป็นอินพุทและ ck เป็นเอาท์พุท PROCESS เป็นคำที่ใช้ในการเริ่มต้นสำหรับการบรรยายในเชิงพฤติกรรม และภายในโปรเซสกำหนดให้ periodic เป็นตัวแปรที่มีค่าเริ่มต้นเป็น "0" ถ้าสัญญาณ en มีค่าเป็น "1"' จะทำให้ตัวแปร periodic ถูกคอมพลีเมนท์ (complement) และส่งค่าให้กับ clk ซึ่งเป็นสัญญาณเอาท์พุท และสำหรับคำสั่ง WAIT จะเป็นการกำหนดให้สัญญาณมีคาบเวลาเท่ากับ 1 ไมโครวินาที

```
ARCHITECTURE behavioral OF clock_component IS

BEGIN
   PEOCESS
      VARIABLE periodic : BIT := '0';
   BEGIN
      IF en='1' THEN
       periodic := Not periodic;
      END IF;
       ck <= periodic;
       WAIT FOR 1 US;
   END PROCESS;
END behavioral;
```

รูปที่ 3.5 การบรรยายเชิงพฤติกรรมของ clock _ component

3.) หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อย ที่เป็นประโยชน์ต่อการเขียนรูปแบบการบรรยายระบบดิจิตอลสามารถเก็บไว้ใน ส่วนของแพ็กเก็ต ซึ่งหน่วยการออกแบบต่างๆ เช่น หน่วยการออกแบบ Entity หน่วยการออกแบบสถาปัตยกรรมหรือ หน่วยการออกแบบแพ็กเกจอื่นๆ สามารถเรียกข้อมูลเหล่านี้ไปใช้ได้ นอกจากนั้นสิ่งที่นิยมทำกันมากคือการนำรูปแบบ มาตรฐานต่างๆ เช่น อุปกรณ์มาตรฐาน (เช่น ไอซีตระกูล 74XX เป็นต้น) มาเก็บไว้ในรูปของแพ็กเกจ ที่ทุกคนสามารถ เข้าถึงได้ ตามปกติแล้วแพ็กเกจจะแบ่งออกเป็น 2 ส่วนคือ การประกาศแพ็กเกจ (Package declaration)และ ส่วนของบอดี้แพ็กเกจ (Package body) เนื่องจาก แพ็กเกจถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่ กำลังเขียนอยู่ ฉะนั้นการที่นำแพ็กเกจไปใช้นั้นจะต้องมีการเชื่อมโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษา VHDL สามารถ กระทำได้ด้วยชุดคำสั่ง USE

3.1) PACKAGE DECLARATION

ส่วนที่มีความสำคัญที่สุดของแพ็กเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ส่วนการประกาศแพ็กเกจ เนื่องจากเป็นส่วนที่ใช้กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็กเกจ สำหรับนำไปใช้ภายนอกตัวของแพ็กเกจเอง ถ้ามี การประกาศสิ่งใดๆ ในส่วนของส่วนบอดี้แพ็กเกจ แต่ไม่ถูกประกาศในส่วนการประกาศแพ็กเกจจะทำให้ค่าและพฤติ กรรมไม่สามารถนำไปใช้งานในส่วนนอกได้ซึ่งเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของการประกาศ Entity คือ จุดเชื่อมต่อ หรือ พอร์ต ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเกจสามารถสร้างขึ้นได้โดยไม่จำเป็น ต้องมีส่วนบอดี้ และยังสามารถนำไปใช้งานจากรูปแบบภายนอกได้เช่น ใช้สำหรับประกาศ ชนิด (Type) หรือสัญญาณ เช่นเดียวกับ ส่วนบอดี้แพ็กเกจที่ไม่จำเป็นต้องมี ส่วนของการประกาศแพ็กเกจ แต่แพ็กเกจนั้นจะไม่สามารถนำไปใช้จาก รูปแบบอื่นได้

```
PACKAGE package_name IS
    Package_declarative_part
END package_name;
```

รูปที่ 3.6 โครงสร้างทั่วไปของส่วนการประกาศแพ็กเกจ

3.2) PACKAGE BODY

โครงสร้างซึ่งประกอบด้วยลำดับคำสั่งที่ใช้บรรยายฟังก์ชันการทำงานของโปรแกรมย่อยทั้งหลาย ซึ่งชื่อของโปรแกรมย่อยนั้นๆ ได้ถูกประกาศไปแล้วในส่วนของการประกาศแพ็กเกจ จะถูกเก็บไว้ในส่วนของบอดี้แพ็กเกจ ทั้งนี้รวมถึง การกำหนดค่าคงที่ต่างๆ อันได้แก่ค่าคงที่ที่ถูกประกาศชื่อไว้ก่อนในส่วนของการประกาศแพ็กเกจ และถูกกำหนดค่าใน ส่วนของบอดี้แพ็กเกจ ฉะนั้นในส่วนของบอดี้แพ็กเกจจึงไม่จำเป็นต้องมี ถ้าในส่วนของการประกาศแพ็กเกจไม่มีการ ประกาศชื่อที่เป็นโปรแกรมย่อย หรือค่าคงที่ การเขียนบอดี้แพ็กเกจนั้นจะเป็นไปตามกฎเกณฑ์ดังแสดงในรูปที่ 3.7

```
PACKAGE BODY package_name IS
    declarative part
END package_name;
```

รูปที่ 3.7 โครงสร้างของบอดี้แพ็กเกจ

4.) หน่วยการออกแบบ Configuration

ดังที่ทราบกันแล้วว่าระบบดิจิตอลรูปแบบหนึ่งไม่ว่าจะเป็นอะไรก็ตาม จะสามารถมีหน่วยการออกแบบ Entity ได้ เพียงหนึ่งเดียวเท่านั้น ซึ่งในหน่วยการออกแบบ Entity หนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลาย หน่วย ดังนั้นจะต้องมีหน่วยการออกแบบ Configuration มาเพื่อกำหนดการใช้ Configuration ของการประกอบ Entity กับหน่วยการออกแบบสถาปัตยกรรมหน่วยใดๆ เข้าด้วยกัน

```
COMFIGURTRION identifier OF entiry_name IS
    Configuration_declarative_part
END ;
```

รูปที่ 3.8 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

5.) โปรแกรมย่อย

การใช้ฟังก์ชันและโพรซีเจอร์ใน VHDL เปรียบได้กับการใช้โปรแกรมย่อยในการเขียนโปรแกรมภาษาชั้นสูงทั่วๆ ไปค่าที่ถูกส่งกลับหรือถูกเปลี่ยนแปลงโดยโปรแกรมย่อยอาจจะมีหรือไม่มีผลต่อฮาร์ดแวร์โดยตรงก็ได้ เช่นถ้าใช้ฟังก์ชันแทนการกระทำในสมการบูลีนก็จะมีผลต่อวงจรลอจิกจริงๆ ในขณะที่ถ้าใช้โปรแกรมย่อยในการเปลี่ยนชนิดของข้อมูล หรือในการคำนวณค่าการหน่วงเวลาแล้วก็จะไม่มีผลต่อโครงสร้างของฮาร์ดแวร์ รูปที่ 3.9 แสดงการใช้โพรซีเจอร์ เพื่อเปลี่ยนข้อมูลชนิด 8 บิตเป็นค่าจำนวนเต็ม และรูปที่ 3.10 แสดงการใช้ฟังก์ชันโดยกำหนดให้ X เป็นตัวแปรชนิด บิตแทนการกระทำในสมการบูลีน

```
TYPE byte IS ARRAY (7 DOWNTO 0) OF BIT;
 ...
PROCEDURE byte_to_integer (ib : IN byte; oi : OUT INTEGER) IS
        VARIABLE result: INTEGER := 0;
BEGIN
        FOR i IN 0 TO 7 LOOP
                IF ib(i) = '1' THEN
                result  := result + 2**i;
                END IF;
        END LOOP;
        oi := result;
END byte_to_integer
```

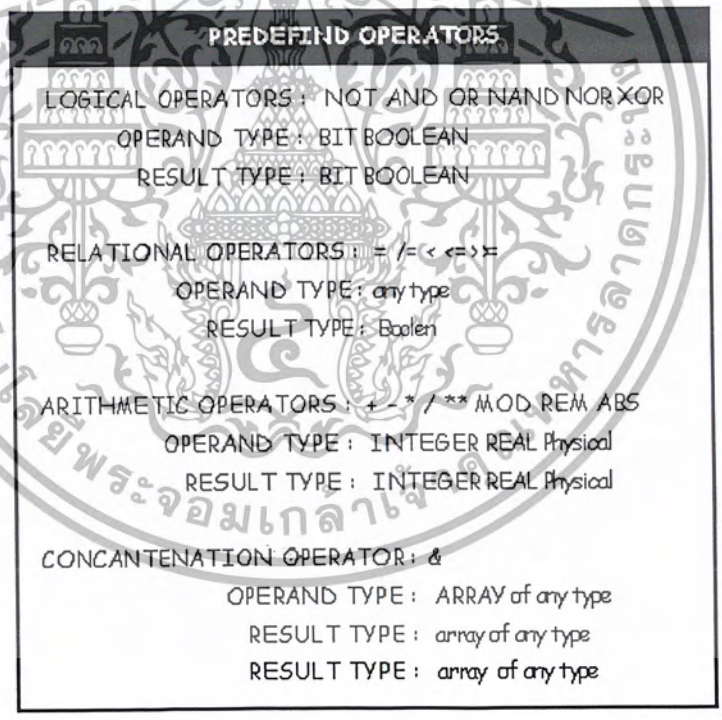
รูปที่ 3.9 การใช้โพรซีเจอร์

```
FUNCTION f (a, b, c:BIT) RETURN BIT  IS
        VARIABLE x:BIT;
BEGIN
        x := ((NOT a) AND (NOT b) AND c);
        RETURN x;
END f;
```

รูปที่ 3.10 การใช้ฟังก์ชัน

6.) โอเปอร์เรเตอร์

การบรรยายเชิงพฤติกรรมในภาษา VHDL มีตัวดำเนินการหรือโอเปอร์เรเตอร์ทางลอจิกและคณิตศาสตร์เช่นเดียว กับภาษาซอฟต์แวร์ทั่วไปดังรูปที่ 3.11



รูปที่ 3.11 ตัวดำเนินการใน VHDL

7.) เวลาและความพร้อมเพรียง

ในวงจรอิเล็กทรอนิกส์อุปกรณ์ทุกๆ ตัวจะอยู่ในสภาพเตรียมพร้อมเสมอ (Always Active) และจะมีเรื่องของเวลา เข้ามาเกี่ยวข้องในทุกๆเหตุการณ์ที่เกิดขึ้นเสมอ VHDL เป็นภาษาที่ได้รับการออกแบบมาเพื่อให้สามารถบรรยายรูป แบบและการพ้องกันของเวลาสำหรับการทำงานของอุปกรณ์ได้อย่างถูกต้อง การบรรยายการทำงานที่อยู่ภายในส่วน ของการบรรยายสถาปัตยกรรม จะมีการทำงานที่พร้อมเพรียงกันเสมอ หรือแม้แต่โปรเซสซึ่งมีการทำงานภายในเป็น แบบลำดับคำสั่งก็ตาม ซึ่งหากมีหลายๆ โปรเซสอยู่ภายในโครงสร้างเดียวกัน ทุกๆ โปรเซสก็จะทำงานไปพร้อมๆ กัน ด้วย

8.) สัญญาณและตัวแปร

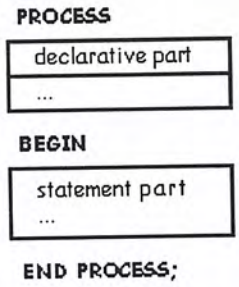
สัญญาณมีลักษณะเป็นเสมือนตัวกลางฮาร์ดแวร์ที่ใช้ในการส่งผ่านข้อมูลและมีเรื่องของเวลาเข้ามาเกี่ยวข้องด้วยการ กำหนดค่าให้กับสัญญาณจะใช้สัญลักษณ์ <= ในการส่งค่าและสามารถใช้คำสั่ง AFTER เพื่อกำหนดช่วงเวลาในการ ส่งผ่านค่าของสัญญาณ เช่น w <= a AFTER 12 NS หมายถึงการกำหนดค่าสัญญาณ a ให้กับ w หลังจากเวลา ผ่านไป 12 นาโนวินาที ในทางตรงข้ามตัวแปรมีลักษณะเป็นเสมือนตัวกลางที่ใช้ในการส่งผ่านข้อมูลและ ไม่มีเรื่องของ เวลาเข้ามาเกี่ยวข้องด้วย ซึ่งตัวแปรจะถูกใช้ในส่วนที่มีการทำงานเป็นแบบลำดับคำสั่งเช่นใน ฟังก์ชัน โพรซีเจอร์ และ โปรเซส สำหรับการกำหนดค่าให้กับตัวแปรจะใช้สัญลักษณ์ :=

3.1.5 การบรรยายเชิงพฤติกรรม

การบรรยายลักษณะการทำงานของอุปกรณ์ฮาร์ดแวร์ในเชิงพฤติกรรม เป็นการบรรยายลักษณะการเปลี่ยนแปลงของ ข้อมูลในรูปแบบของอัลกอริธึมสำหรับการคำนวณผลลัพธ์ที่เกิดขึ้นซึ่งสืบเนื่องมาจากการเปลี่ยนแปลงสภาวะของข้อมูล ที่เข้ามาโดยไม่คำนึงถึงลักษณะ โครงสร้างหรือความสัมพันธ์ของอุปกรณ์ที่อยู่ภายในว่าจะเป็นอย่างไร ในหัวข้อนี้จะแสดงถึงการบรรยายเชิงพฤติกรรม แทนการใช้โมดูลฮาร์ดแวร์รวมถึงข้อกำหนดต่างๆ ที่ควรรู้

3.1.6 โปรเซส

โปรเซสเป็นรูปแบบพื้นฐานอย่างหนึ่งที่ใช้ในการกำหนดให้กับสัญญาณ โปรเซสจะอยู่ในสถานะที่เตรียมพร้อมอยู่เสมอ และจะปฏิบัติคำสั่งพร้อมๆ กันกับโปรเซสอื่นๆ ที่อยู่ในสถาปัตยกรรมบรรยายเดียวกัน โดยโปรเซสจะปฏิบัติงานตามคำ สั่งทันทีที่มีเหตุการณ์เกิดขึ้นกับสัญญาณที่อยู่ทางด้านขวามือของสัญลักษณ์กำหนดค่าให้กับสัญญาณ (<=) การบรรยาย โปรเซสจะเริ่มต้นด้วยคำสั่ง PROCESS และจบด้วยคำสั่ง END PROCESS ในรูปที่ 3.12 เป็นการแสดงส่วน ประกอบของการบรรยายแบบโปรเซส ซึ่งประกอบด้วยส่วนของการประกาศตัวแปรที่ต้องใช้และส่วนของการปฏิบัติ คำสั่งเพื่อให้ได้ผลลัพธ์ที่ต้องการ



รูปที่ 3.12 รูปแบบของการบรรยายแบบโปรเซส

3.1.7 การกำหนดตัวดำเนินการภายในโปรเซส

ตัวดำเนินการภายในโปรเซสมี 3 ชนิดคือ ตัวแปร (Variable) ไฟล์ (File) และตัวคงที่ (Constant) ซึ่งตัวดำเนิน การทั้งสามชนิดนี้หากมีการประกาศไว้ในโปรเซสใดก็จะใช้ได้เฉพาะภายในโปรเซสนั้นเท่านั้น สำหรับการติดต่อกับภาย นอกหรือระหว่างโปรเซสสามารถทำได้โดยใช้สัญญาณ (Signal) หรือตัวคงที่ที่ได้ประกาศไว้ในส่วนของ ARCHITECTURE ในรูปที่ 3.13 แสดงตัวอย่างการประกาศตัวกระทำภายในโปรเซส ซึ่งจะอยู่ระหว่างคำสั่ง PROCESS และ BEGIN และค่าเริ่มต้นที่ถูกกำหนดให้กับตัวดำเนินการภายในโปรเซสจะถูกนำมาใช้ในตอนเริ่มต้น ของการปฏิบัติเพียงครั้งเดียวเท่านั้น ต่างกับค่าเริ่มต้นที่อยู่ภายในโปรแกรมย่อยจะถูกนำมาใช้ทุกครั้งที่มีการเรียกใช้ โปรแกรมย่อยนั้น ๆ

```
PROCESS
  FILE flush : TEXT IS IN "filename.dat";
  VARIABLE var : BIT;
  CONSTANT n : INTEGER := 0;
BEGIN
  ...
END PROCESS;
```

รูปที่ 3.13 ตัวอย่างการประกาศตัวดำเนินการภายในโปรเซส

3.1.8 การกำหนดการกระทำภายในโปรเซส

การกระทำใดๆ ภายในโปรเซสจะเป็นการปฏิบัติแบบลำดับ (Sequential) เสมอ ซึ่งภายในโปรเซสสามารถใช้ประโยค เงื่อนไขหรือการทำซ้ำได้เช่น IF-THEN - ELSE,CASE - WHEN, FOR LOOP และ WHILE LOOP ดังตัวอย่างในรูปที่ 3.14 และ 3.15

```
ARCHITECTURE demo OF partial_process IS
 ....
BEGIN
  PROCESS
  ....
   BEGIN
          ....
          x <= '1';
          IF x = '1' THEN
            perform action_1
          ELSE
            perform action_2
          END IF;
          ...
  END PROCESS;
END demo;
```

รูปที่ 3.14 เงื่อนไขการกระทำในโปรเซส

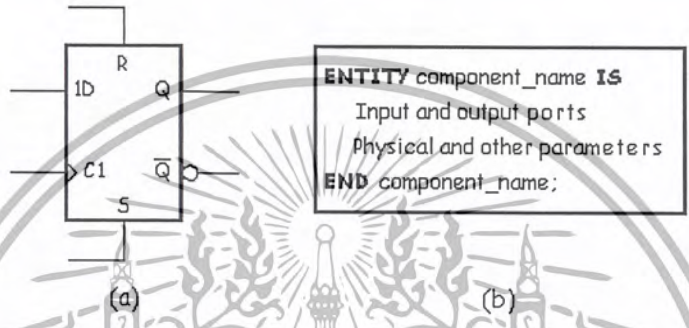
```
ARCHITECTURE demo OF partial_process IS
 ...
BEGIN
  PROCESS
   BEGIN
          ...
          x <= a AFTER 10 NS;
          y <= b AFTER 6 NS;
          ...
  END PROCESS;
END demo;
```

รูปที่ 3.15 แสดงการกระทำในโปรเซส

3.1.9 การกระตุ้นและยับยั้งการกระทำของโปรเซส

การกระทำภายในโปรเซสจะอยู่ในสภาพเตรียมพร้อม และมีการปฏิบัติงานอยู่ตลอดเวลาที่มีการเปลี่ยนแปลงของเหตุการณ์ เกิดขึ้น อย่างไรก็ดีเราสามารถกระตุ้นหรือยับยั้งการกระทำภายในโปรเซสได้โดยการกำหนดรายการของสัญญาณที่ต้อง การให้โปรเซสปฏิบัติงานเมื่อมีเหตุการณ์เกิดขึ้นกับสัญญาณที่กำหนด

ไว้เท่านั้น ส่วนเหตุการณ์ใดๆ ที่เกิดขึ้นกับสัญญาณ ที่ไม่ได้กำหนดไว้ในรายการก็จะไม่ส่งผลให้มีการกระทำภายในโปรเซส ซึ่งรายการของสัญญาณนี้เรียกว่า Sensitivity List และจะกำหนดไว้ภายในวงเล็บหลังคำสั่ง PROCESS รูปที่ 3.16 (a) แสดงตัวอย่างโมเดล และรูปที่ 3.16 (b) เป็นตัวอย่างการบรรยายการเชื่อมต่อของ D-Flip Flop ส่วนรูปที่ 3.17 แสดงถึงการบรรยายเชิงพฤติกรรมของ D-Flip Flop โดยในรูปที่ 3.17 (a) เป็นการใช้ตัวกระทำภายนอกโปรเซส และรูปที่ 3.17 (b) เป็นการใช้ตัวกระทำภายในโปรเซส โดยมีรายการของสัญญาณ (rst, set, clk) เป็นตัวกระตุ้นการปฏิบัติงานภายในโปรเซส



รูปที่ 3.16 (a) ตัวอย่างโมเดล D-Flip Flop (b) การบรรยายการเชื่อมต่อของ D-Flip Flop

```
ARCHITECTURE behavioral OF d_sr_flipflop IS
  SIGNAL state : BIT := '0';
BEGIN
  dff : PEOCESS (rst, set, clk)
        BEGIN
      IF set= '1' THEN
          state <= '1' AFTER sq_delay;
       ELSIF rst = '1' THEN
          state <= '0' AFTER rq_delay;
       ELSIF clk = '1' AND clk ' EVENT THEN
          state <= d AFTER cq_delay;
      END IF;
  END PROCESS dff;
  q <= state;
  qb <= NOT state;
END behavioral;
```

(a)

```
ARCHITECTURE average_dalay_ behavioral OF d_sr_flipflop IS
BEGIN
  dff : PEOCESS (rst, set, clk )
          VARIABLE state : BIT := '0';
        BEGIN
      IF set= '1' THEN
          state <= '1';
        ELSIF rst = '1' THEN
          state <= '0';
        ELSIF clk = '1' AND clk ' EVENT THEN
          state <= d;
      END IF;
        q <= state AFTER (sq_delay + rq_delay + cq_delay)/3;
        qb <= NOT state AFTER(sq_delay + rq_delay + cq_delay)/3;
  END PROCESS dff;
END behavioral;
```

(b)

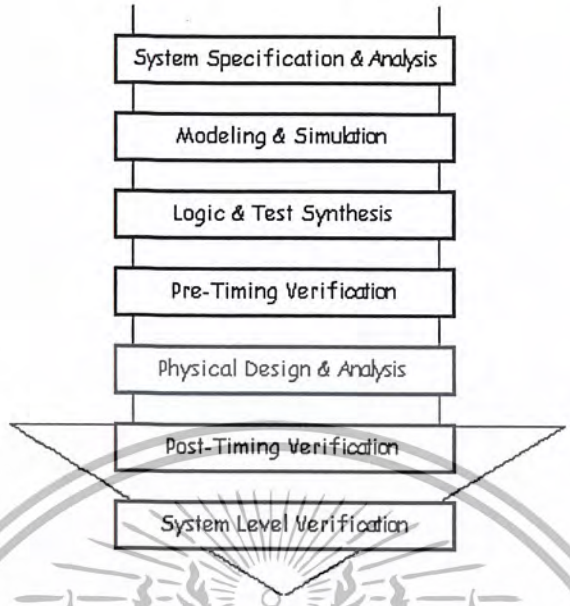
รูปที่ 3.17 การบรรยายเชิงพฤติกรรมของ D-FlipFlop

(a) การใช้ตัวกระทำภายนอกโปรเซส

(b) การใช้ตัวกระทำภายในโปรเซส

3.1.10 การออกแบบจากบนลงล่าง

ในการพัฒนาวงจรรวมดิจิตอลขนาดใหญ่ที่มีความซับซ้อน วิศวกรหรือผู้ออกแบบมักจะมองการออกแบบให้อยู่ในรูปของ บล็อกไดอะแกรมก่อนที่ทำวิเคราะห์ให้ลึกถึงรายละเอียดต่อไป ซึ่งภาษา VHDL นั้นอนุญาตให้อธิบายและวิเคราะห์การ ทำงานของแต่ละบล็อก รวมถึงการปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามต้องการ นอกจากนี้ยัง สามารถเพิ่มเติมในรายละเอียดในแต่ละขั้นตอนได้ ซึ่งหลักการนี้สอดคล้องกับหลักการออกแบบจากบนลงล่าง (Top - Down Design) นั่นเอง ถ้าทดลองเปรียบเทียบกับการออกแบบจากล่างขึ้นบน (Bottom - Up Design) จะเห็นได้ว่า การออกแบบจากล่างขึ้นบนจะใช้เวลาการออกแบบมากกว่า 90% เนื่องจากเป็นการวาดวงจรด้วยอุปกรณ์ต่างๆ (Schematic capture) ที่ประกอบกันเข้าเป็นวงจรที่ต้องการออกแบบ ก่อนแล้วจึงทำการจำลองการทำงาน และตรวจ สอบความถูกต้อง ดังนั้นการใช้ภาษา VHDL กับหลักการออกแบบจากบนลงล่างจึงเป็นทางออกให้กับวิศวกรให้สามารถ ออกแบบและพัฒนาวงจรที่มีความซับซ้อนได้มากขึ้น ทั้งยังช่วยลดเวลาและค่าใช้จ่ายในการออกแบบด้วย



รูปที่ 3.18 ขั้นตอนการออกแบบจากบนลงล่าง

จากรูปที่ 3.18 แสดงถึงขั้นตอนของการออกแบบจากบนลงล่าง ทั้งนี้ในทางปฏิบัติอาจมีข้อแตกต่างไปจากนี้บ้าง เล็กน้อยเนื่องจากขั้นตอนของการผลิต (Implementation) สามารถกระทำได้หลายเทคโนโลยี สำหรับรายละเอียดของขั้นตอน การออกแบบจากบนลงล่างในแต่ละขั้นตอนมีดังนี้

1. สร้างข้อกำหนดของความต้องการ และวิเคราะห์ระบบ เพื่อหาแนวความคิดและหลักการ (Idea and Concept) ใน การแก้ปัญหา

2. เขียนรูปแบบของระบบที่ต้องการออกแบบ โดยใช้ภาษา VHDL หรือ ภาษา HDL อื่น ๆ สำหรับบรรยายพฤติกรรมการ ทำงาน พร้อมทั้งจำลองการทำงาน เพื่อเปรียบเทียบและตรวจสอบความถูกต้องกับข้อกำหนด

3. หลังจากที่ได้หลักการขั้นต้นพร้อมแนวความคิดที่ผ่านการตรวจสอบแล้วหลักการนี้จะถูกเพิ่มเติมในรายละเอียดลงมา เป็นลำดับขั้นที่สอง จนกระทั่งอยู่ในระดับที่จะนำไปผลิตวงจริง หรือสังเคราะห์ในขั้นตอนนี้เองเทคโนโลยีที่จะมารองรับ วงจรออกแบบจะถูกกำหนดขึ้น และระบบช่วยการออกแบบจะสังเคราะห์วงจรที่ได้จากรูปแบบที่เขียนขึ้นให้อยู่ในรูปของ วงจรที่ประกอบด้วยอุปกรณ์อิเล็กทรอนิกส์ หรือวงจรในระดับเกท และการเชื่อมต่อระหว่างกันของอุปกรณ์เหล่านั้นหรือ ไม่ก็อยู่ในรูปของ Netlist ที่สามารถนำไปผลิตในอุปกรณ์อื่นได้

4. หลังจากการสังเคราะห์วงจรให้อยู่ในระดับเกทหรือ Netlist แล้ว ข้อมูลนี้จะถูกใช้สำหรับจำลองการทำงานในเรื่อง ความถูกต้องของฟังก์ชัน พร้อมกับนำข้อมูลที่เกี่ยวข้องกับเวลาเข้ามาประกอบการพิจารณาด้วย ซึ่งตามปกติแล้วอุปกรณ์ ทางอิเล็กทรอนิกส์ทุกชิ้นจะมีเวลาหน่วงของการแพร่กระจาย (Propagation

Delay Time) เสมอ ถึงแม้ว่าจะเป็น เวลาที่น้อยมากในระดับนาโนวินาทีก็ตาม แต่ถ้าภายในวงจรหนึ่งประกอบด้วยเกทของฟังก์ชันต่างๆ จำนวน 10,000 เกท ขึ้นไป เวลาดังกล่าวนี้จะสะสมกันมากขึ้น จนอาจทำให้การทำงานของวงจรรวมทั้งหมดผิดพลาดไป หรือไม่สามารถทำ งานในย่านความถี่สัญญาณนาฬิกาที่สูงได้

5. ผลิตเป็นวงจรจริง (Technology and device mapping) โดยนำข้อมูลที่ได้จากการสังเคราะห์มาผลิต ซึ่งอาจ จะอยู่ในรูปของแผงวงจรไฟฟ้า ที่ประกอบด้วยอุปกรณ์หลายๆ ชิ้นหรืออยู่ในรูปของวงจรรวม ASIC

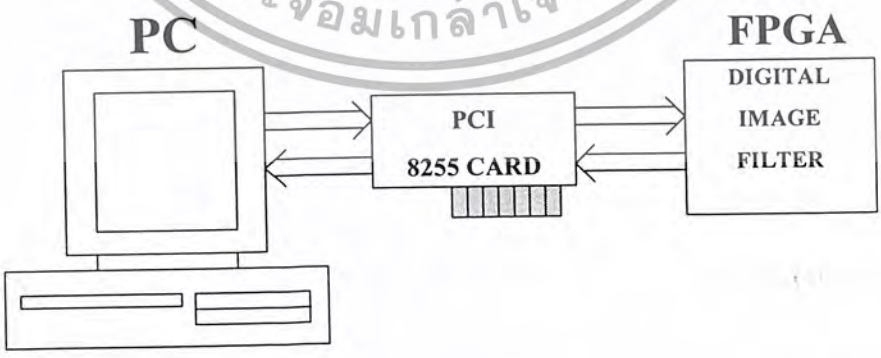
6. ทำการตรวจสอบการทำงานและตัวแปรทางด้านเวลาทั้งหมด เพื่อความถูกต้องของวงจรเป็นครั้งสุดท้ายก่อนนำไปรวมเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบดิจิตอล เนื่องจากในขั้นตอนนี้ วงจรที่ออกแบบ จะประกอบด้วยจุดต่อทางอินพุทและเอาท์พุท ซึ่งเป็นจุดต่อสำหรับการรับและส่งสัญญาณกับภายนอก

7. นำวงจรที่ออกแบบไว้ประกอบเข้ากับอุปกรณ์อื่นๆ ให้เป็นระบบที่สมบูรณ์ แล้วทำการทดสอบการทำงานทั้งระบบร่วมกับอุปกรณ์อื่นๆ อีกครั้งเพื่อควบคุมคุณภาพของผลิตภัณฑ์

3.2 การติดต่อระหว่างวงจรกรองสัญญาณภาพเชิงเลขกับคอมพิวเตอร์

ในการสร้างการกระจายทางคณิตศาสตร์ใช้คอมพิวเตอร์ส่วนบุคคล (Personal Computer) เป็นตัวจำลองการทำงานของตัวรับข้อมูลภาพ เช่น กล้องวีดีโอ สแกนเนอร์ เป็นต้น โดยคอมพิวเตอร์จะทำหน้าที่ส่งข้อมูลภาพขนาด 256×256 ไปยังวงจรประมวลผลภาพ โดยผ่านอุปกรณ์เชื่อมต่อ (Interface Card) ที่ทำหน้าที่ส่งสัญญาณข้อมูลขนาด 8 บิท และสัญญาณ Clock

จากรูปที่ 3.19 แสดงการติดต่อระหว่างวงจรกรองสัญญาณภาพเชิงเลขกับคอมพิวเตอร์ โดยคอมพิวเตอร์จะส่งลำดับข้อมูลภาพ 256 ระดับสีเทาขนาด 256×256 จุด ผ่าน Card IO แบบ PCI Bus โดยกำหนดให้ ขา OUT 8 ทำหน้าที่ในการส่ง Clock ไปให้กับ FPGA ขา OUT 0 (LSB)–OUT 7 (MSB) เป็นเอาท์พุทขนาด 8 บิท ทำหน้าที่ส่งข้อมูลภาพต้นแบบจากคอมพิวเตอร์ไปยังวงจรกรองสัญญาณภาพเชิงเลข ขา In 8 (LSB) -In 15 (MSB) เป็นอินพุทขนาด 8 บิท ทำหน้าที่รับข้อมูลภาพใหม่ที่ผ่านการประมวลผลแล้วกลับมายังคอมพิวเตอร์เพื่อทำการแสดงผลต่อไป



รูปที่ 3.19 แสดงการติดต่อระหว่างวงจรกรองสัญญาณภาพเชิงเลขกับคอมพิวเตอร์

3.3 การออกแบบและการสร้างวงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจาย

สามารถแบ่งได้ออกเป็นส่วนใหญ่ๆ ดังนี้

3.3.1 การออกแบบวงจรจัดลำดับข้อมูลภาพ

3.3.2 การออกแบบหน่วยความจำผลคูณสัมประสิทธิ์กับข้อมูลภาพ

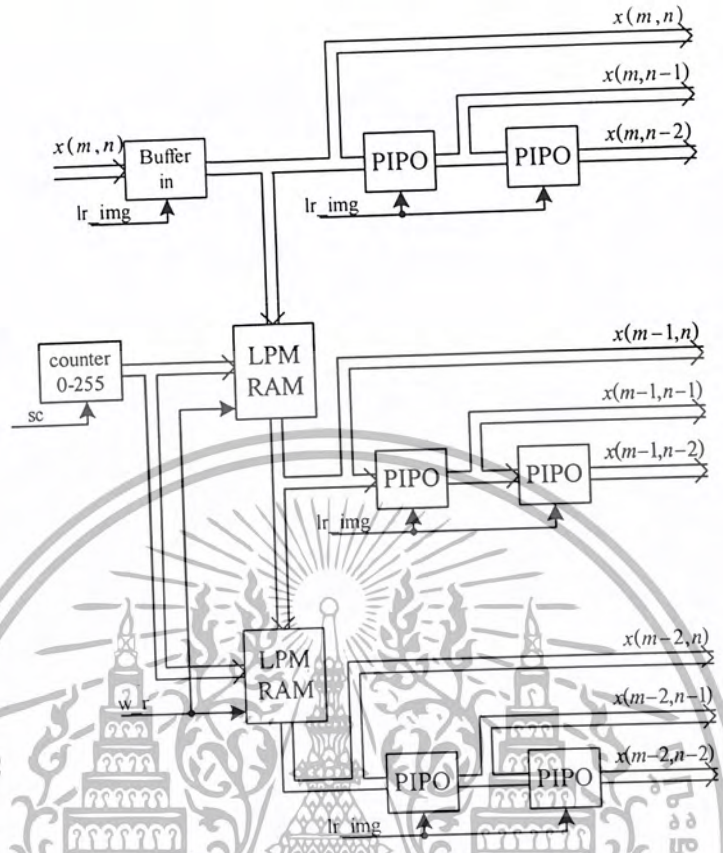
3.3.3 การออกแบบวงจรการกระจายทางคณิตศาสตร์

3.3.4 การออกแบบสัญญาณควบคุมการทำงานของจงจรและรอมภาพ

3.3.1 การออกแบบวงจรจัดลำดับข้อมูลภาพ

ลำดับข้อมูลภาพที่ส่งเข้ามานี้ จำเป็นจะต้องจัดลำดับสัญญาณข้อมูลเข้าให้ถูกต้องก่อนนำไปประมวลผลนั่นคือต้องสร้างลำดับข้อมูล X(m,n), X(m,n-1), X(m,n-2), X(m-1,n), X(m-1,n-1), X(m-1,n-2), X(m-2,n), X(m-2,n-1), X(m-2,n-2) ส่งไปคูณกับแผ่นข้อมูล(Mask) ขนาด 3×3 ที่มีคุณสมบัติเป็น Low-pass Filtering และ Sobel Edge Detector โดยกำหนดให้ m เป็นค่าชี้ข้อมูลแถว(row) และ n เป็นค่าชี้ข้อมูลหลัก (column) โดยลำดับข้อมูลภาพมีขนาด 256×256 pixel ตัวหน่วงข้อมูลต้องมีค่าเท่ากับ 256×2 ดังนั้นสามารถใช้ตัวเลื่อนข้อมูล (Shift Register) ต่ออนุกรมกันจำนวน 512 ตัว ซึ่งจะเห็นว่ามีการสิ้นเปลืองจำนวนเกทมาก การแก้ทางหนึ่งของการลดขนาดของวงจรก็คือต้องลดขนาดของงจรหน่วงข้อมูลที่ทำหน้าที่จัดลำดับข้อมูลแนวตั้งที่มีขนาด 256 หน่วยจำนวน 2 ชุด โดยการใช้หน่วยความจำชนิดอ่านและเขียนได้ (RAM) เป็นตัวเลื่อนข้อมูล ที่ถูกควบคุมแอดเดรส (address) โดยวงจรนับ (counter) ขนาด 8 บิท ในที่นี้เลือกใช้ LPM(RAM) ในตัว FPGA

จากการสร้างหน่วยความจำและวงจรนับเข้ามาแทนตัวหน่วงนี้ ทำให้วงจรจัดลำดับข้อมูลมีขนาดเล็กลงมาก แต่ว่าจำเป็นต้องสร้างสัญญาณควบคุมหน่วยความจำและวงจรนับให้มีการทำงานได้เหมือนตัวหน่วงก็จะทำให้การทำงานของวงจรทั้งหมดช้าลงไปด้วยเพราะต้องมีการควบคุมเขียนอ่านหน่วยความจำและวงจรนับด้วย แต่ในขณะที่ใช้ตัวเลื่อนต่ออนุกรมกัน 512 ตัวจะทำงานเร็วกว่ามากเพราะใช้สัญญาณนาฬิกา 1 ลูกต่อลำดับสัญญาณข้อมูล 1 ชุด นอกจากนี้การทำงานของวงจรนับไม่จำเป็นต้องเริ่มจาก 0 ก็ได้ เพราะว่าเมื่อเริ่มต้นที่ตำแหน่งใดของหน่วยความจำจะถือว่าตำแหน่งนั้นเป็นตำแหน่งเริ่มต้นและวงจรนับจะนับเพิ่มไป 256 ครั้งก็จะกลับมาที่ตำแหน่งเดิม ดังนั้นจึงไม่จำเป็นต้องมีสัญญาณไปรีเซ็ท (reset) วงจรนับในช่วงเริ่มต้นของการนับก็ได้ ส่วนลำดับสัญญาณออกทั้ง 9 จะถูกส่งไปวงจรกระจายทางคณิตศาสตร์ต่อไป



รูปที่ 3.20 โครงสร้างการจัดลำดับข้อมูลภาพ

3.3.2 การออกแบบหน่วยความจำผลคูณสัมประสิทธิ์กับข้อมูลภาพ

การคูณด้วยทฤษฎีการกระจายทางคณิตศาสตร์จะเก็บผลคูณหน้ากากภาพกับข้อมูลภาพไว้ในหน่วยความจำแบบรอม (Rom) กรณีการประมวลผลสัญญาณสองมิติมีลำดับข้อมูลเข้า 9 ชุดดังนั้นต้องใช้รอมขนาด 512 ไบท์ (byte) หรือ 512×8 บิท จากนั้นนำค่าสัมประสิทธิ์จากหน้ากากภาพจากรูปที่ 3.21 สมมุติให้มีค่า A,B,C,D,E,F,G,H,I เป็นค่าที่อยู่ในหน้ากากภาพขนาด 3×3 นำค่าทั้ง 9 มาบวกกันโดยใช้ระบบเลขฐานสองเป็นตัวกำหนดการบวกได้ผลลัพธ์ Y คือค่าที่ต้องบรรจุลงในรอม ถ้าค่า Y เป็นค่าลบต้องทำเป็นเลขคอมพลีเมนท์ของสอง (2's complement) ก่อน แสดงการสร้างข้อมูลในตารางที่ 3.1

A	B	C
D	E	F
G	H	I

รูปที่ 3.21 แสดงค่าสมมุติที่เก็บในหน้ากาก

ตารางที่ 3.1 การสร้างข้อมูลที่เก็บไว้ในรอม

Binary Control Sumation									Output Data for ROM
R	S	T	U	V	W	X	Y	Z	Q=RA + SB + TC + UD + VE + WF + XG +YH + ZI
0	0	0	0	0	0	0	0	0	Q=0
0	0	0	0	0	0	0	0	1	Q=I
0	0	0	0	0	0	0	1	0	Q=H
0	0	0	0	0	0	0	1	1	Q=H+I
0	0	0	0	0	0	1	0	0	Q=G
:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
0	0	1	0	1	0	1	0	1	Q=C+E+G+I
0	0	1	0	1	0	1	1	0	Q=C+E+G+H
:	:	:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:	:	:
1	1	1	1	1	1	1	1	0	Q=A+B+C+D+E+F+G+H
1	1	1	1	1	1	1	1	1	Q=A+B+C+D+E+F+G+H+I

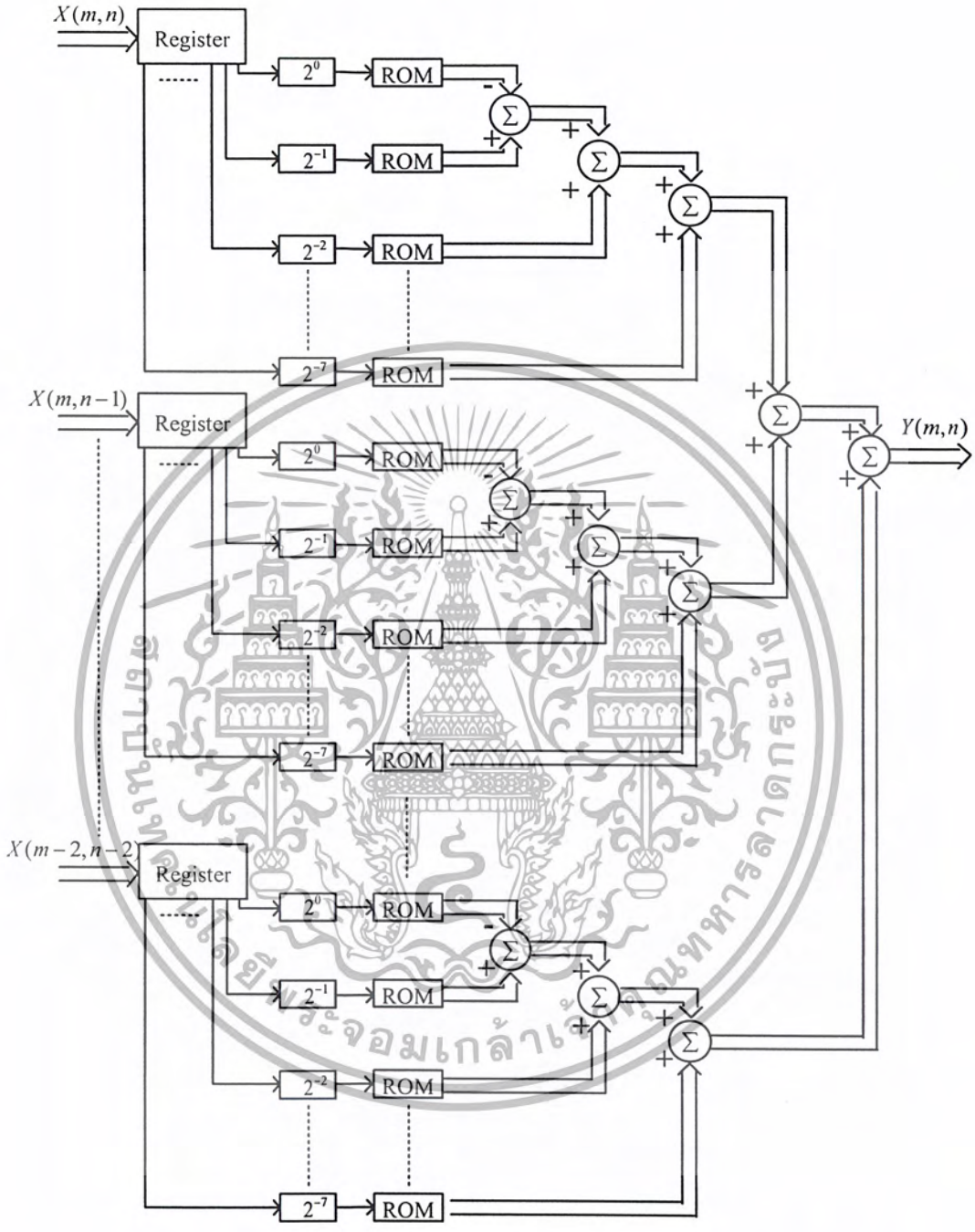
หมายเหตุ R=X(m,n), S=X(m,n-1), T=X(m,n-2), U=X(m-1,n), V=X(m-1,n-1), W=X(m-1,n-2), X=X(m-2,n), Y=X(m-2,n-1), Z=X(m-2,n-2)

3.2.3 การออกแบบวงจรการกระจายทางคณิตศาสตร์

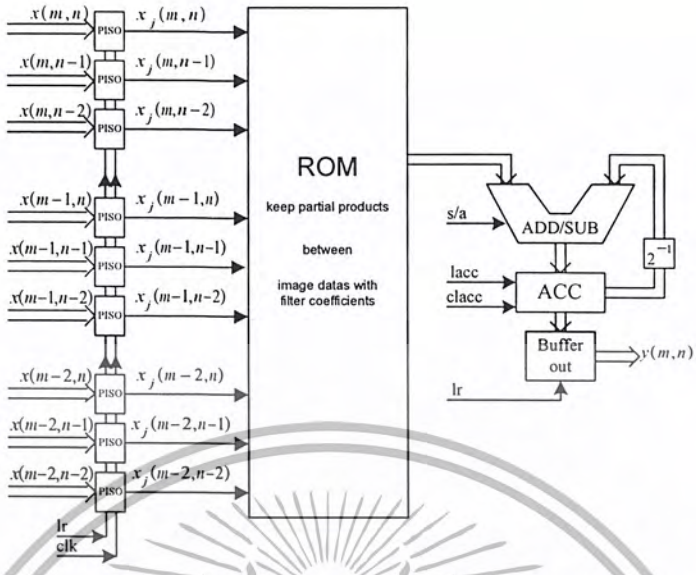
การประมวลผลข้อมูลภาพโดยวิธีการกระจายทางคณิตศาสตร์ใช้วิธีนำดำดับสัญญาณ X(m,n) มาจัดลำดับใหม่ดังที่กล่าวมาแล้ว ลำดับสัญญาณทั้ง 9 ค่าส่งมาประมวลผลโดยการเปิดตารางข้อมูล (look-up table) ซึ่งเก็บแผ่นข้อมูลภาพแบบต่างๆ ไม่ว่าจะเป็น Low-pass Filtering และ Sobel Edge Detector การส่งลำดับ X(m,n) จะมีอยู่ 2 แบบ คือ แบบขนานและแบบ 1 BAAT (Bit At A Time)

การส่งลำดับ X(m,n) แบบขนานทั้งนี้ก็เพื่อต้องการให้ได้ความเร็วในการประมวลผลสูง จากรูปที่ 3.22 จะแยกบิทแต่ละบิทของสัญญาณเข้าทั้ง 9 แล้วส่งเข้าหน่วยความจำแต่ละตัวข้อมูลที่ออกจากหน่วยความจำจะถูกส่งไปรวมกันทั้งหมดแล้วได้ลำดับข้อมูล Y(m,n) ออกมา

การส่งลำดับ X(m,n) แบบ 1 BAAT (Bit At A Time) ลำดับข้อมูลที่ออกมาจากหน่วยความจำจะถูกส่งไปรวมข้อมูลเพื่อให้ได้ข้อมูลออก Y(m,n) ต่อไป จากรูปที่ 3.23 แสดงถึงวิธีการส่งลำดับ X(m,n) แบบอนุกรม เมื่อส่งครบ 8 บิทแล้วถึงจะได้ Y(m,n) 1 ไบท์



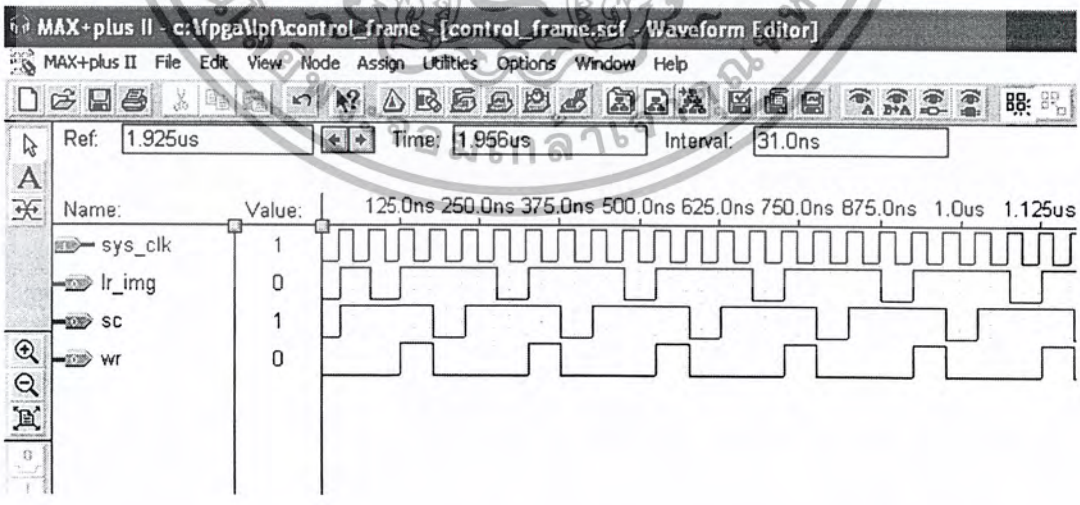
รูปที่ 3.22 โครงสร้างวงจรกรองสัญญาณภาพเชิงเลขสองมิติแบบขนาน (DA full parallel)



รูปที่ 3.23 โครงสร้างวงจรกรองสัญญาณเชิงเลขสองมิติแบบ I BAAT (Bit At A Time)

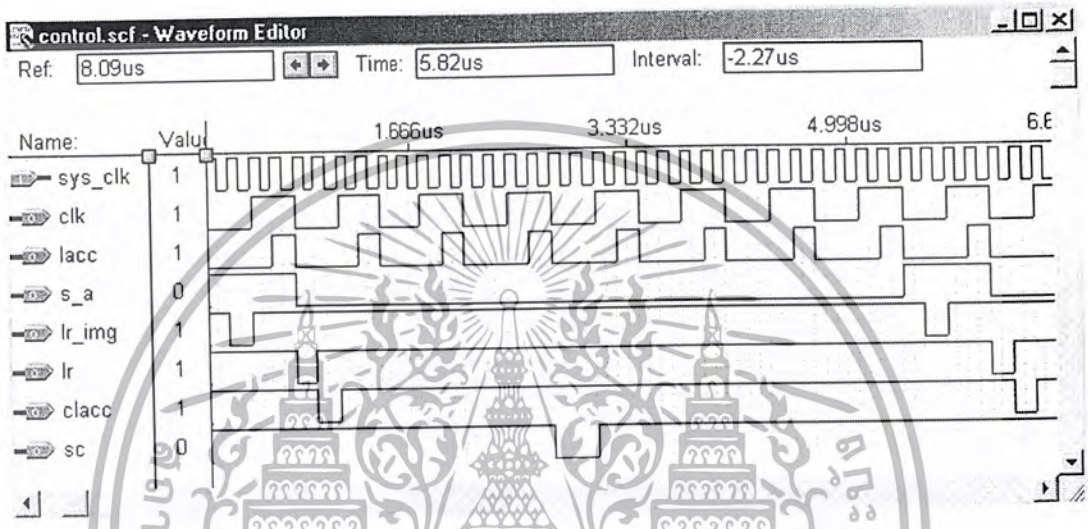
จะเห็นว่าการใช้วิธีสัญญาณเข้าแบบขนานจะประมวลผลได้เร็วกว่าแบบอนุกรมถึง 8 เท่า เมื่อใช้ความถี่ของสัญญาณเข้าแบบขนานนี้จะประมวลผลได้เร็วกว่าแบบอนุกรมถึง 8 เท่า เมื่อใช้ความถี่ของสัญญาณนาฬิกาเท่ากันการทำงานจะไม่ซับซ้อนเหมือนกับแบบอนุกรม

3.2.4 การออกแบบสัญญาณควบคุมการทำงานของวงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจายและรอมภาพ



รูปที่ 3.24 Time Diagram สัญญาณควบคุมจังหวะการทำงานแบบขนาน (DA full parallel)

จากรูปที่ 3.24 สัญญาณควบคุมการทำงานของจงจรการกระจายทางคณิตศาสตร์แบบขนาน สัญญาณ lr_img เป็นสัญญาณนำข้อมูลจากรอมภาพเข้าไปเก็บใน Buffer ของวงจรจัดลำดับข้อมูลภาพพร้อมทั้งเลื่อนข้อมูลภาพตามลำดับข้อมูลภาพ สัญญาณ sc ทำหน้าที่ให้ counter นับเพื่อให้วงจรหน่วงข้อมูลโดยใช้ LPM RAM หน่วงลำดับข้อมูลไป 256 ลำดับ และสัญญาณ wr เป็นสัญญาณที่ใช้ควบคุมการอ่านเขียนของ LPM RAM



รูปที่ 3.25 Time Diagram สัญญาณควบคุมจังหวะการทำงานแบบ 1 BAAT (Bit At A Time)

จากรูปที่ 3.25 สัญญาณควบคุมการทำงานของจงจรการกระจายทางคณิตศาสตร์แบบอนุกรม จังหวะแรกสัญญาณ lr_img เป็นสัญญาณนำข้อมูลจากรอมภาพเข้าไปเก็บใน Buffer ของวงจรจัดลำดับข้อมูลภาพพร้อมทั้งเลื่อนข้อมูลภาพตามลำดับข้อมูลภาพ จังหวะที่สองสัญญาณ lr ทำหน้าที่ให้ตัวเลื่อนข้อมูลภาพเข้าขนานออกอนุกรม (Parallel In Serial Out : PISO) นำข้อมูลขนาด 8 บิท ทั้ง 9 ลำดับข้อมูล (ที่ส่งมาจากวงจรลำดับข้อมูลภาพ) เข้าไปเก็บไว้ในตัวเลื่อนข้อมูลพร้อมทั้งให้ Buffer ของส่วน output นำค่าที่ได้จากการคำนวณเก็บไว้เป็น output ด้วย จังหวะที่สามสัญญาณ clacc ทำการเคลียร์(clear) ข้อมูลใน Accumulator จังหวะที่สี่สัญญาณ clk และ lacc ทำหน้าที่ให้จังหวะการเลื่อนข้อมูลของตัวเลื่อนข้อมูลภาพเข้าขนานออกอนุกรมและตัว Accumulator เพื่อทำการคำนวณผลลัพธ์ โดยทำการเลื่อนข้อมูลแปดทั้งต่อหนึ่งรอบการทำงาน จังหวะที่ห้าสัญญาณ sc ทำหน้าที่ให้ counter ของวงจรข้อมูลภาพทำการเลื่อนตำแหน่งข้อมูลภาพของรอมภาพ สัญญาณนี้จะเกิดขึ้นเมื่อมีการเลื่อนข้อมูลในตัวเลื่อนข้อมูลภาพเข้าขนานออกอนุกรมแล้วสามครั้งดั้งแสดงในรูปที่ 3.23 จังหวะที่หกสัญญาณ s_a ทำหน้าที่ให้ตัวบวกเปลี่ยนจากการบวกเป็นลบเมื่อถึงการเลื่อนข้อมูลในตัวเลื่อนข้อมูลภาพเข้าขนานออกอนุกรมแล้วเจ็ดครั้งเพื่อทำการลบในครั้งที่ 8

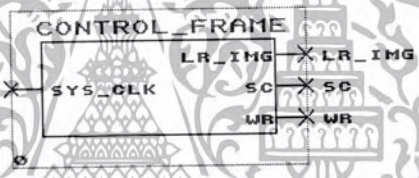
บทที่ 4
การทดลองและผลการทดลอง

การออกแบบส่วนต่างๆของวงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจาย ทำการเขียนโปรแกรมโดยให้แต่ละส่วนทำงานตามที่ได้ออกแบบ โดยใช้ภาษา VHDL ทำการ Compile แล้วทำการจำลองการทำงานของโปรแกรมแต่ละส่วนที่ได้เขียนขึ้นให้ได้ผลตามที่ออกแบบไว้

4.1 การจำลองการทำงานของส่วนต่างๆที่ออกแบบด้วยโปรแกรม Max+Plus II

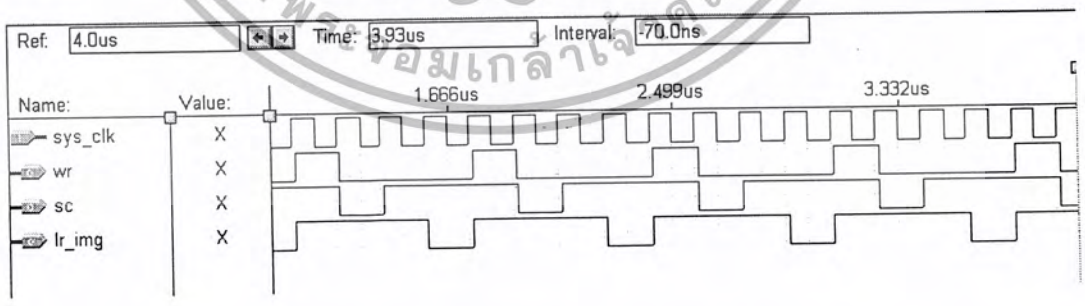
4.1.1 ส่วนของการสร้างชุดควบคุมการจัดลำดับสัญญาณภาพ

ส่วนของการสร้างชุดควบคุมการจัดลำดับสัญญาณภาพ ทำหน้าที่ในการส่งสัญญาณไปควบคุม การจัดลำดับสัญญาณของภาพที่เข้ามาให้กับ ทำการเขียนโปรแกรมด้วยภาษา VHDL ที่สามารถสังเคราะห์เป็นอุปกรณ์ที่มีสัญลักษณ์ (Symbol) ได้มีลักษณะดังรูปที่ 4.1



รูปที่ 4.1 สัญลักษณ์ของชุดควบคุมการจัดลำดับสัญญาณภาพ

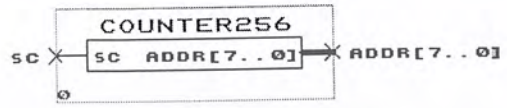
จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (Simulation) ได้ดังนี้



รูปที่ 4.2 ผลการจำลองการทำงานของชุดควบคุมการจัดลำดับสัญญาณภาพ

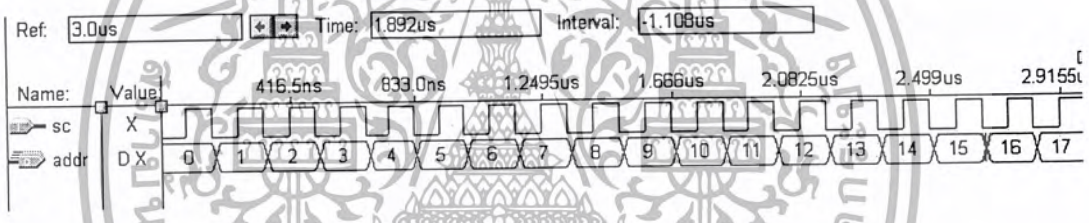
4.1.2 ส่วนของการสร้างวงจรนับ 256

ส่วนของการสร้างวงจรนับ 256 เป็นส่วนหนึ่งในส่วนของการจัดลำดับสัญญาณภาพต้นแบบ เพื่อใช้เป็น Address ให้กับ LPM RAM ซึ่งใช้ทำงานเป็นวงจรหน่วงลำดับสัญญาณภาพต้นแบบก่อนที่จะนำลำดับสัญญาณภาพเข้าไปประมวลผลวงจรจัดลำดับสัญญาณภาพ ทำการเขียนโปรแกรมด้วยภาษา VHDL ที่สามารถสังเคราะห์เป็นอุปกรณ์ที่มีสัญลักษณ์ (Symbol) ได้มีลักษณะดังรูปที่ 4.3



รูปที่ 4.3 สัญลักษณ์ของวงจรนับ 256

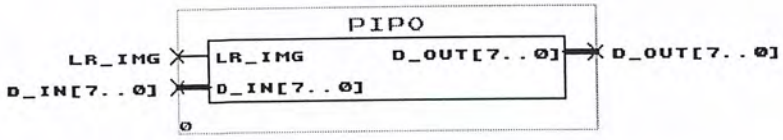
จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (Simulation) ได้ดังนี้



รูปที่ 4.4 ผลการจำลองการทำงานของวงจรนับ 256

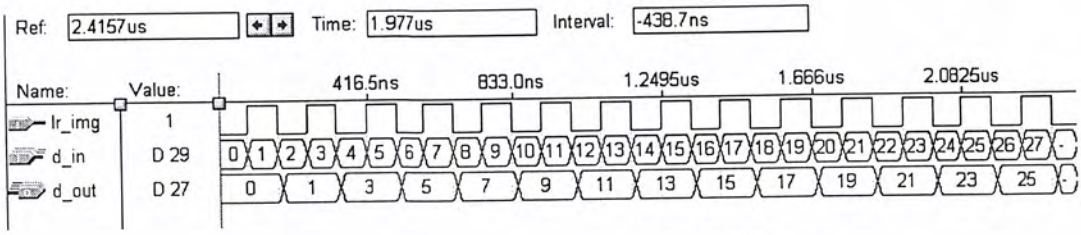
4.1.3 ส่วนของการสร้างวงจรขนานเข้าขนานออก (PIPO)

ส่วนของการสร้างวงจรขนานเข้าขนานออกเพื่อใช้เป็นส่วนจัดลำดับสัญญาณภาพต้นแบบก่อนที่จะนำลำดับสัญญาณภาพเข้าไปประมวลผลวงจรนับ 256 ทำการเขียนโปรแกรมด้วยภาษา VHDL ที่สามารถสังเคราะห์เป็นอุปกรณ์ที่มีสัญลักษณ์ (Symbol) ได้มีลักษณะดังรูปที่ 4.5



รูปที่ 4.5 สัญลักษณ์ของวงจรขนานเข้าขนานออก

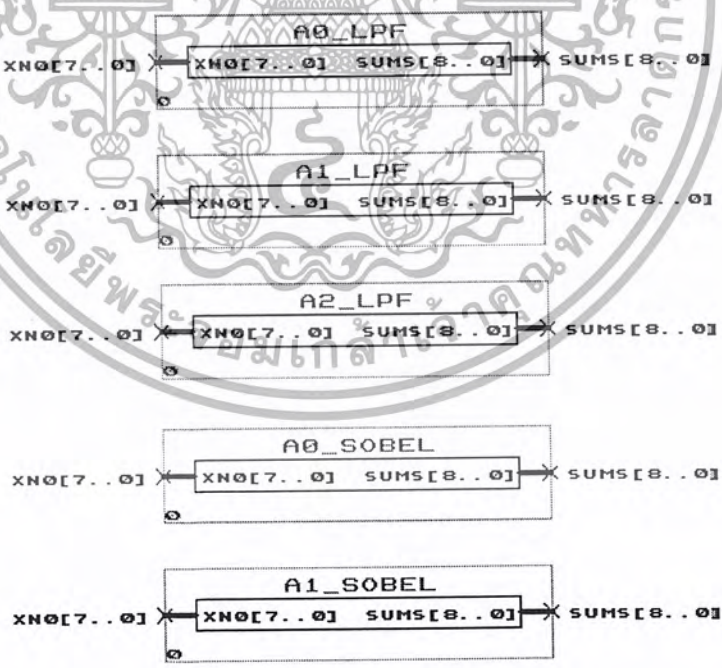
จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (Simulation) ได้ดังนี้



รูปที่ 4.6 ผลการจำลองการทำงานของวงจรขนานเข้าขนานออก

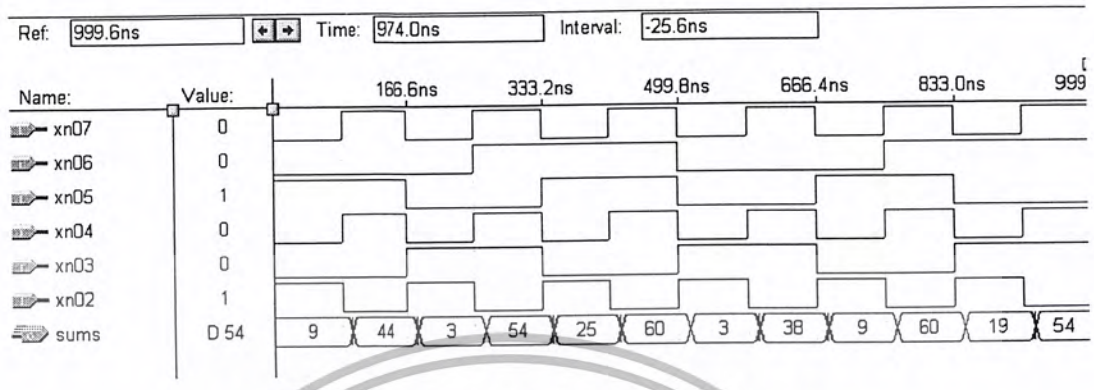
4.1.4 ส่วนของการสร้างวงจร Rom DA แบบต่างๆ

ส่วนของการสร้างวงจร Rom DA ซึ่งแต่ละแบบจะแตกต่างกันขึ้นอยู่กับค่าสัมประสิทธิ์ของหน้ากากที่จะนำมาใช้ในการกรองสัญญาณภาพ เป็นส่วนที่อยู่ในส่วนของการประมวลผลด้วยวิธีเลขคณิตกระจายแบบขนาน โดยนำลำดับสัญญาณภาพทั้ง 9 ค่า ที่ผ่านการจัดลำดับจาก วงจรจัดลำดับภาพมาทำการประมวลผลด้วยเลขคณิตกระจาย ทำการเขียนโปรแกรมด้วยภาษา VHDL ที่สามารถสังเคราะห์เป็นอุปกรณ์ที่มีสัญลักษณ์ (Symbol) ได้มีลักษณะดังรูปที่ 4.7



รูปที่ 4.7 สัญลักษณ์ของวงจร Rom DA แบบต่างๆ

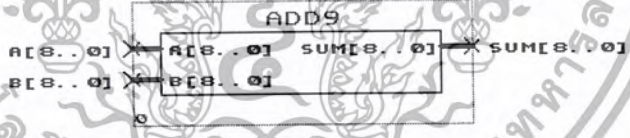
จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (Simulation) ได้ดังนี้



รูปที่ 4.8 ผลการจำลองการทำงานของวงจร Rom DA

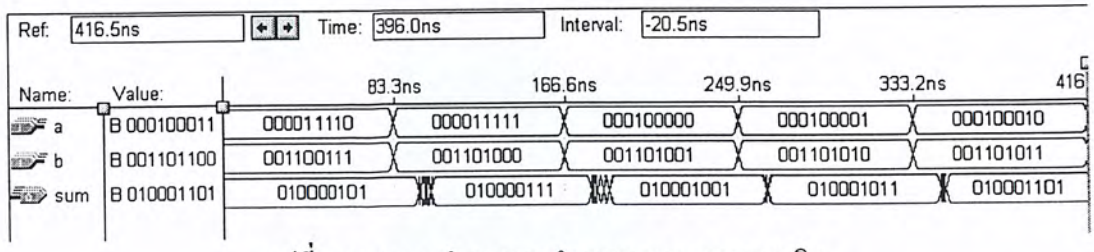
4.1.5 ส่วนของการสร้างวงจรบวก 9 บิท

ส่วนของการสร้างวงจรบวก 9 บิท ซึ่งเป็นส่วนหนึ่งในการประมวลผลด้วยเลขคณิตกระจายแบบขนาน โดยจะทำหน้าที่ในการบวกผลลัพธ์ที่ออกมาจากส่วนของ Rom DA ทั้ง 9 สัญญาณให้เป็นสัญญาณเดียวกัน ซึ่งสัญญาณทั้งหมดนั้นจะมีจำนวนทั้งหมด 9 บิท ทำการเขียนโปรแกรมด้วยภาษา VHDL ที่สามารถสังเคราะห์เป็นอุปกรณ์ที่มีสัญลักษณ์ (Symbol) ได้มีลักษณะดังรูปที่ 4.9



รูปที่ 4.9 สัญลักษณ์ของวงจรบวก 9 บิท

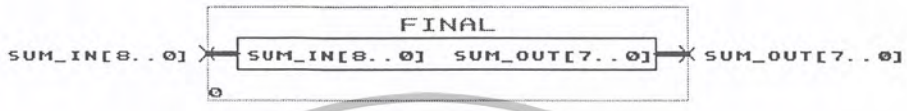
จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (Simulation) ได้ดังนี้



รูปที่ 4.10 ผลการจำลองการทำงานของวงจรบวก 9 บิท

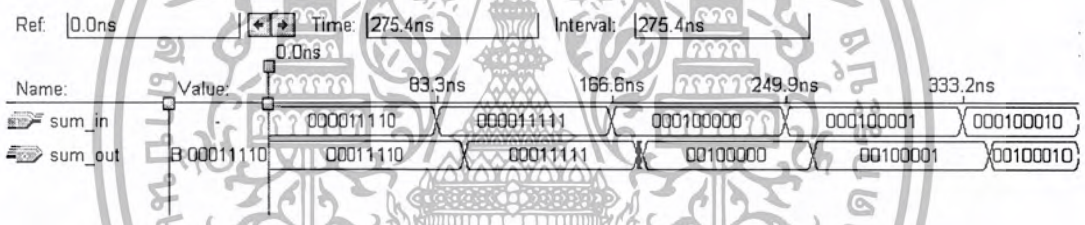
4.1.6 ส่วนของการสร้างวงจรสเกลค่ากลับเป็น 8 บิท

ส่วนของการสร้างวงจรสเกลค่ากลับเป็น 8 บิท ซึ่งเป็นวงจรที่ทำหน้าที่เปลี่ยนสัญญาณภาพที่ได้จากการประมวลผลที่มีขนาด 9 บิท กลับไปเป็นสัญญาณภาพขนาด 8 บิท เพื่อส่งกลับมายังคอมพิวเตอร์เพื่อแสดงผล ทำการเขียนโปรแกรมด้วยภาษา VHDL ที่สามารถสังเคราะห์เป็นอุปกรณ์ที่มีสัญลักษณ์ (Symbol) ได้มีลักษณะดังรูปที่ 4.11



รูปที่ 4.11 สัญลักษณ์ของวงจรสเกลค่ากลับเป็น 8 บิท

จากโปรแกรมที่เขียนขึ้นสามารถจำลองการทำงาน (Simulation) ได้ดังนี้



รูปที่ 4.12 ผลการจำลองการทำงานของวงจรสเกลค่ากลับเป็น 8 บิท

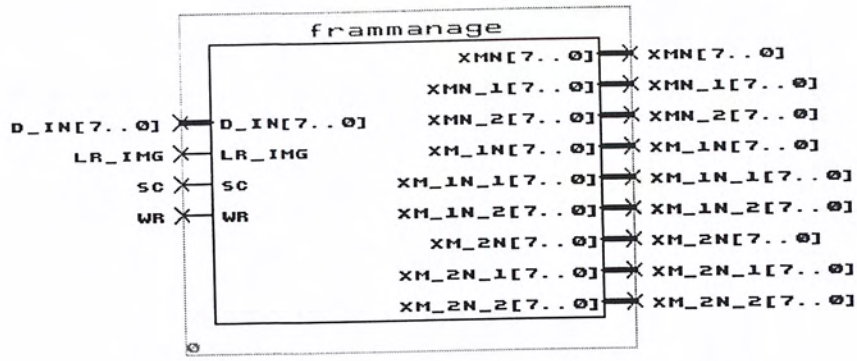
จากการออกแบบส่วนต่างๆ ที่นำมาใช้ในวงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจายเสร็จแล้ว ก็จะทำการนำส่วนต่างๆมาประกอบกันเพื่อเป็นวงจรกรองสัญญาณภาพเชิงเลขที่สมบูรณ์ ซึ่งจะประกอบไปด้วยส่วนหลักๆ 2 ส่วนด้วยกันคือ

1.) ส่วนของวงจรจัดลำดับสัญญาณภาพต้นแบบ (Frame Manager)

2.) ส่วนของการประมวลผลด้วยโครงสร้างเลขคณิตกระจาย

4.1.7 ส่วนของการสร้างวงจรจัดลำดับสัญญาณภาพต้นแบบ (Frame Manager)

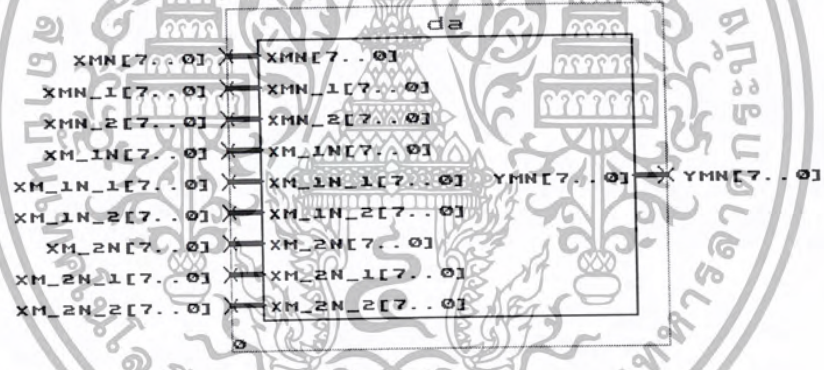
ประกอบไปด้วย วงจรนับ 256 , วงจรขนานเข้าขนานออก และ วงจร LPM RAM เมื่อนำมาประกอบกันแล้วทำการสังเคราะห์เป็นอุปกรณ์ตัวใหม่ที่มีสัญลักษณ์ (Symbol) ได้มีลักษณะดังรูปที่ 4.13



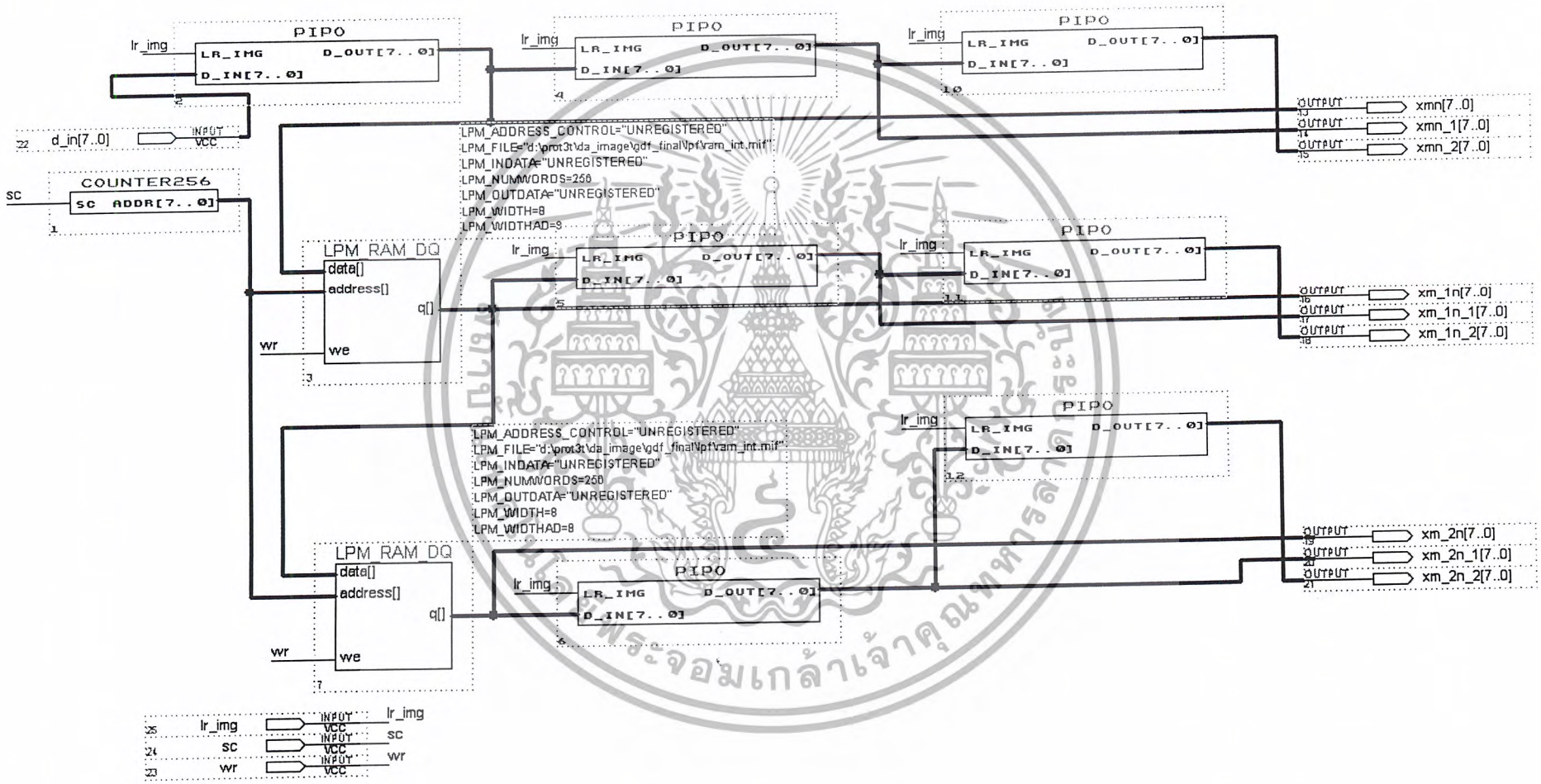
รูปที่ 4.13 สัญลักษณ์ของวงจรจัดลำดับสัญญาณภาพต้นแบบ

4.1.8 ส่วนของการสร้างวงจรประมวลผลสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจาย

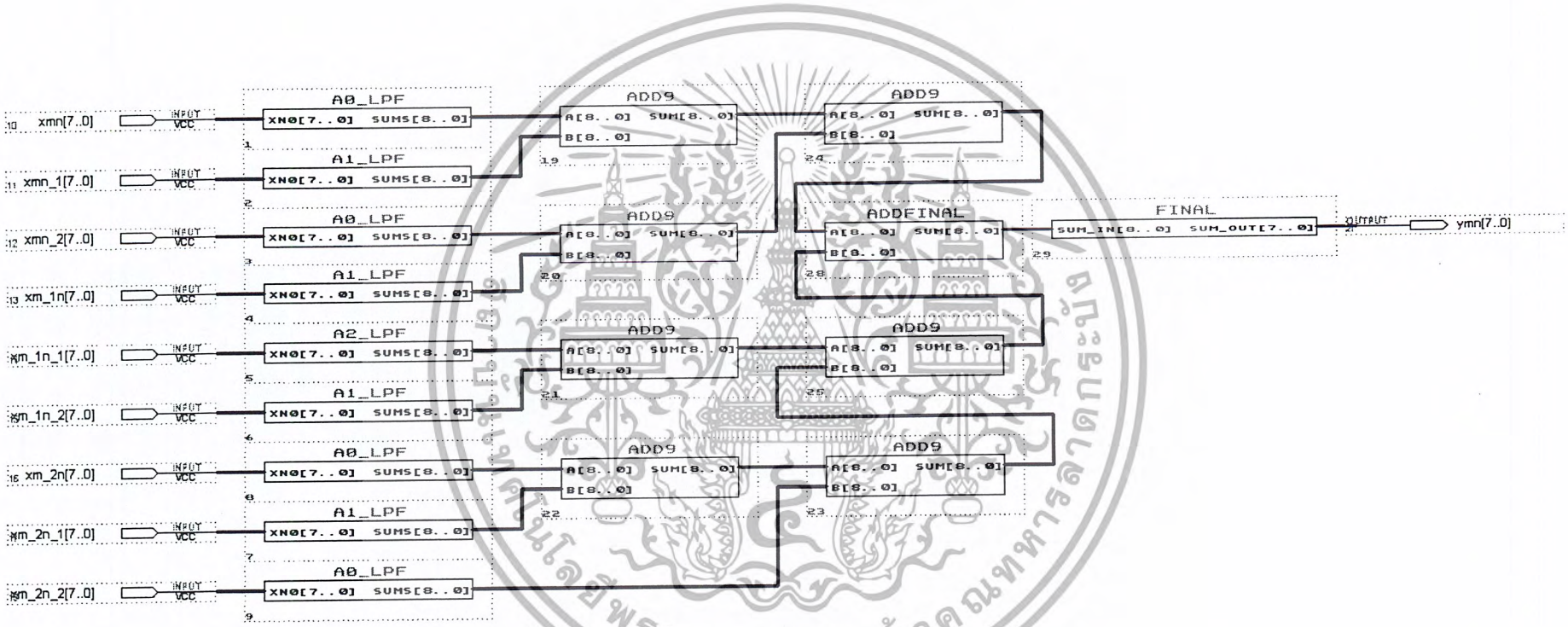
ประกอบไปด้วย วงจร Rom DA แบบต่างๆ, วงจรบวก 9 บิท และวงจรสเกลค่ากลับเป็น 8 บิท เมื่อนำมาประกอบกันแล้วทำการสังเคราะห์เป็นอุปกรณ์ตัวใหม่ที่มีสัญลักษณ์ (Symbol) ได้มีลักษณะดังรูปที่ 4.14



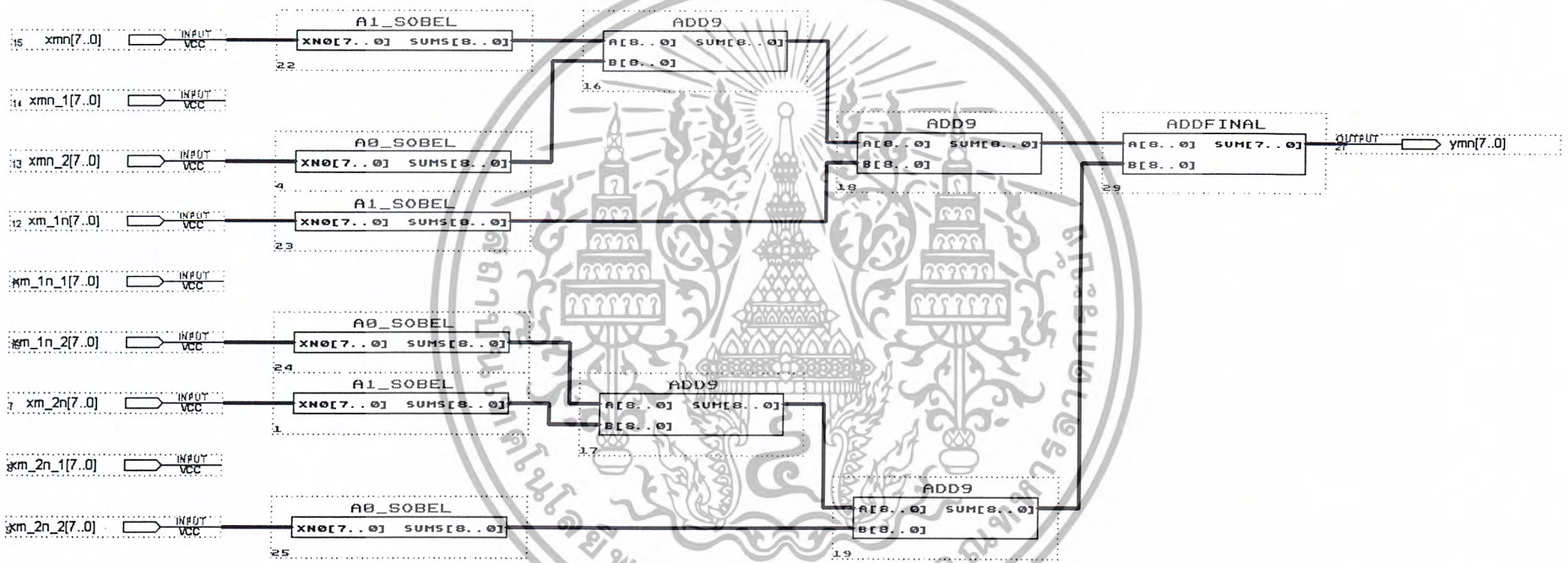
รูปที่ 4.14 สัญลักษณ์ของวงจรประมวลผลสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจาย



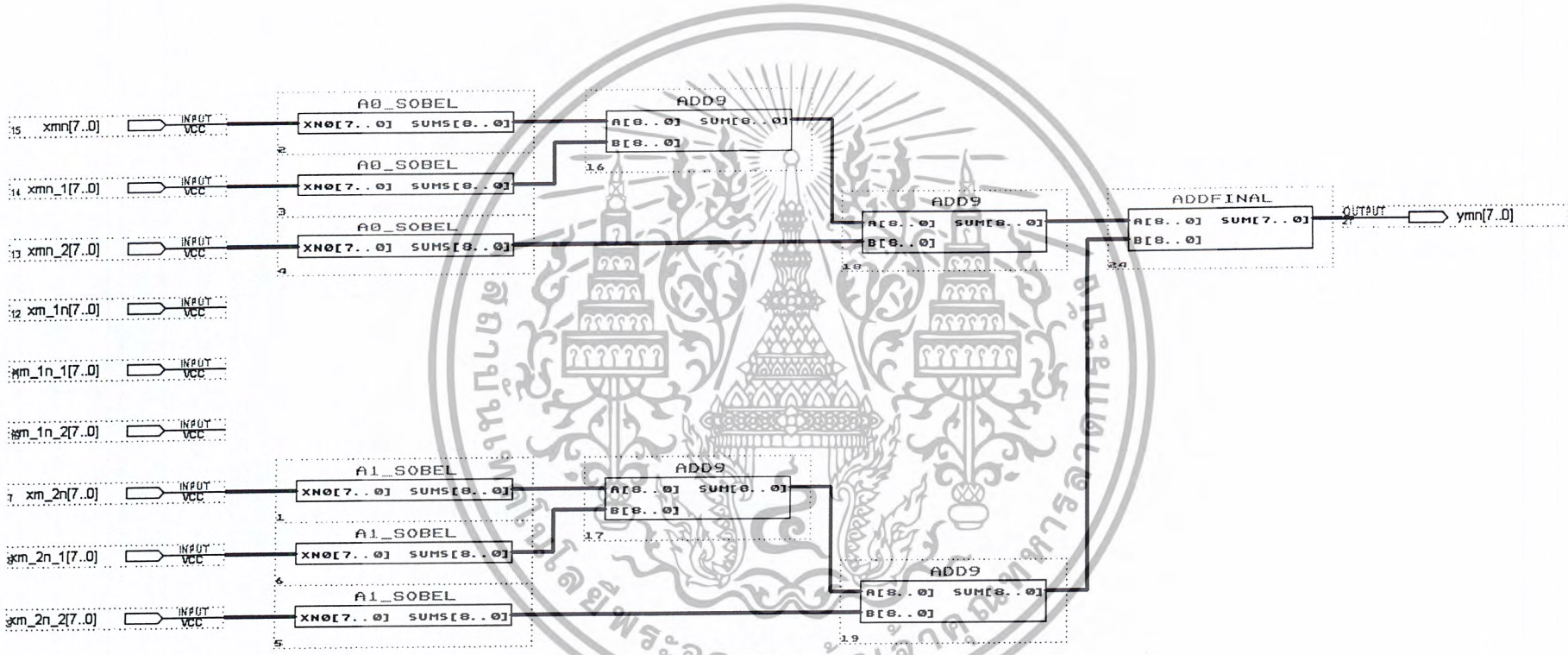
รูปที่ 4.15 ส่วนประกอบภายในและการเชื่อมต่อของวงจรจัดลำดับสัญญาณภาพต้นแบบ



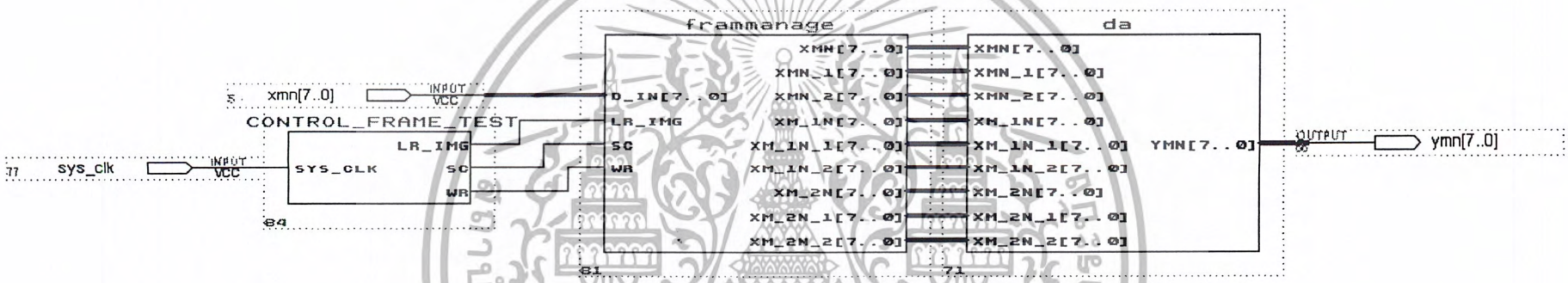
รูปที่ 4.16 ส่วนประกอบภายในและการเชื่อมต่อของวงจรประมวลผลสัญญาณภาพเชิงเลข
ที่ใช้โครงสร้างเลขคณิตกระจายแบบกรองความถี่ต่ำผ่าน



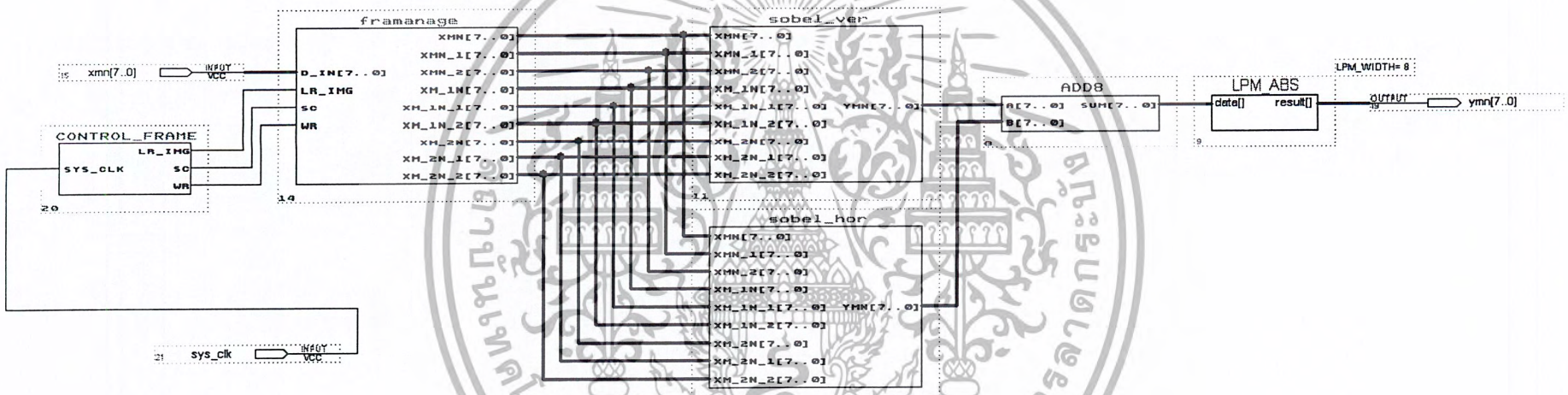
รูปที่ 4.17 ส่วนประกอบภายในและการเชื่อมต่อของวงจรประมวลผลสัญญาณภาพเชิงเลข
ที่ใช้โครงสร้างเลขคณิตกระจายแบบ Vertical Sobel Edge Detector



รูปที่ 4.18 ส่วนประกอบภายในและการเชื่อมต่อของวงจรประมวลผลสัญญาณภาพเชิงเลข
ที่ใช้โครงสร้างเลขคณิตกระจายแบบ Horizontal Sobel Edge Detector



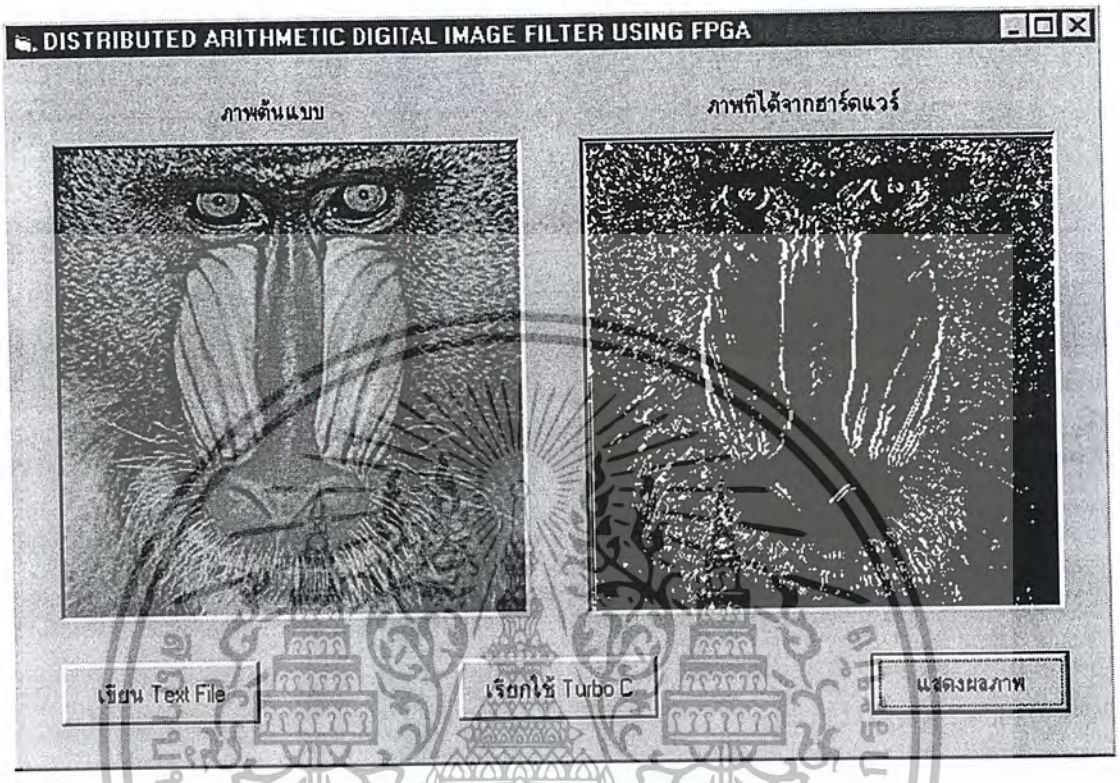
รูปที่ 4.19 ส่วนประกอบภายในและการเชื่อมต่อของวงจรกรองความถี่ต่ำผ่าน
ที่ใช้โครงสร้างเลขคณิตกระจาย



รูปที่ 4.20 ส่วนประกอบภายในและการเชื่อมต่อของวงจรกรองแบบ Sobel Edge Detector ที่ใช้โครงสร้างเลขคณิตกระจาย

4.2 โปรแกรมแสดงผลภาพต้นแบบและภาพที่ได้จากฮาร์ดแวร์

เป็นโปรแกรมที่ใช้ในการทดสอบการทำงานของวงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจาย ซึ่งเขียนขึ้นจากโปรแกรม Microsoft Visual Basic 6.0



รูปที่ 4.21 โปรแกรมที่ใช้ทดสอบการทำงานของฮาร์ดแวร์

การทำงานของโปรแกรมนั้นเริ่มต้นที่ ปุ่ม "เขียน Text File" ที่อยู่ด้านซ้ายมือ ซึ่งจะมีไว้เพื่อเขียนข้อมูลภาพต้นแบบที่อยู่ทางซ้ายเป็น Text File เพื่อเอาไว้ติดต่อกับโปรแกรม Turbo C ปุ่มถัดมาคือปุ่ม "เรียกใช้ Turbo C" มีไว้เพื่อเรียกใช้งานโปรแกรมภาษาซี ซึ่งโปรแกรมภาษาซีนั้นจะทำหน้าที่ติดต่อกับฮาร์ดแวร์โดยตรง โดยเมื่อรันโปรแกรมภาษาซีแล้วจะได้ Text File อีก 1 ไฟล์ที่เป็นข้อมูลภาพที่ผ่านการประมวลผลแล้ว แล้วจึงกดปุ่ม "แสดงผลภาพ" โปรแกรม Microsoft Visual Basic 6.0 ก็จะทำการอ่านข้อมูลภาพแล้วแสดงผลภาพที่ได้จากฮาร์ดแวร์เป็นภาพทางด้านขวา

4.3 การทดลองของวงจรกรองความถี่ต่ำผ่าน

4.3.1 การทดลองวงจรกรองความถี่ต่ำผ่านด้วยภาพ “cameraman.jpg’

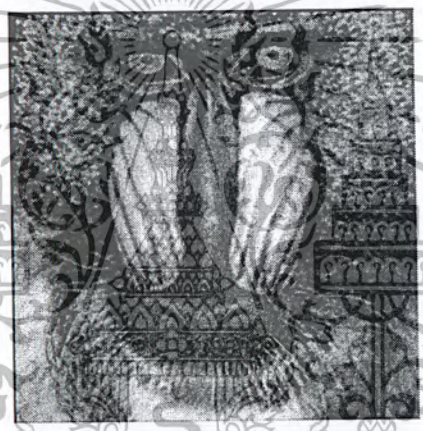
รูปที่ 4.22 ภาพต้นแบบ

รูปที่ 4.23 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.24 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.3.2 การทดลองวงจรกรองความถี่ต่ำผ่านด้วยภาพ “baboon.jpg’

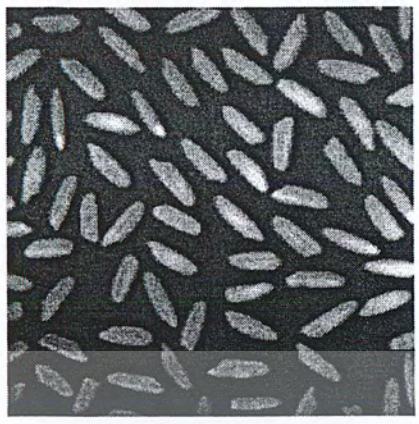
รูปที่ 4.25 ภาพต้นแบบ



รูปที่ 4.26 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.27 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.3.3 การทดลองวงจรกรองความถี่ต่ำผ่านด้วยภาพ “rice.jpg’



รูปที่ 4.28 ภาพต้นแบบ

รูปที่ 4.29 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.30 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.3.4 การทดลองวงจรกรองความถี่ต่ำผ่านด้วยภาพ “sassy.jpg’

รูปที่ 4.31 ภาพต้นแบบ

รูปที่ 4.32 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.33 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.4 การทดลองของวงจรกรองแบบ Sobel edge Detector

4.4.1 การทดลองวงจรกรองแบบ Sobel edge Detector ด้วยภาพ "cameraman.jpg'

1.) การทดลองวงจรกรองแบบ Vertical Gradient

รูปที่ 4.34 ภาพต้นแบบ

รูปที่ 4.35 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.36 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

2.) การทดลองวงจรกรองแบบ Horizontal Gradient

รูปที่ 4.37 ภาพต้นแบบ

รูปที่ 4.38 ภาพที่ได้จากโปรแกรม Matlab

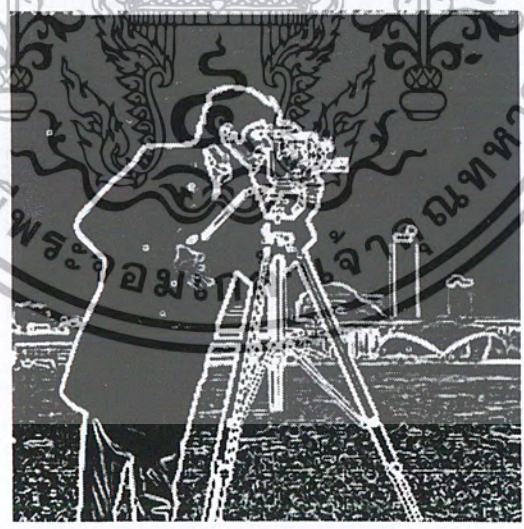
รูปที่ 4.39 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

3.) การทดลอง Gradient Combining

เป็นการนำผลที่ได้จากการทดลอง Vertical Gradient และ Horizontal Gradient มาทำการรวมกัน ตามสมการที่ 4.1

$$Y = \sqrt{Ver^2 + Hor^2} \qquad (4.1)$$

รูปที่ 4.40 ภาพที่ได้จากโปรแกรม Matlab



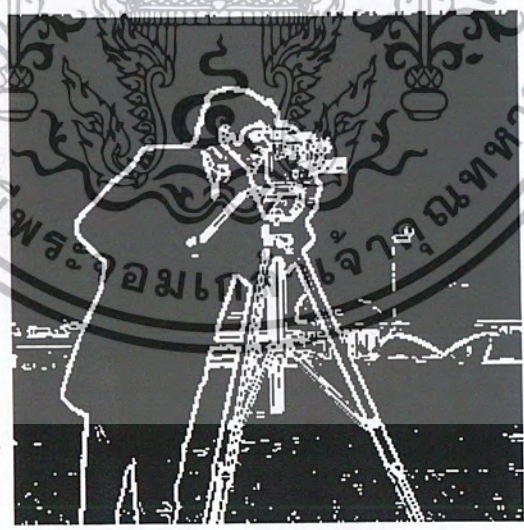
รูปที่ 4.41 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.) การทดลอง Apply Threshold

เป็นการนำผลที่ได้จากการทดลอง Gradient Combining มาทำการ Apply Threshold เพื่อปรับให้ภาพมีความเข้ม 2 ระดับ คือ ดำ และ ขาว

4.1.) กำหนดให้ Threshold = 50

รูปที่ 4.42 ภาพที่ได้จากโปรแกรม Matlab



รูปที่ 4.43 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.2.) กำหนดให้ Threshold = 80

รูปที่ 4.44 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.45 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.4.2 การทดลองวงจรกรองแบบ Sobel edge Detector ด้วยภาพ "baboon.jpg'

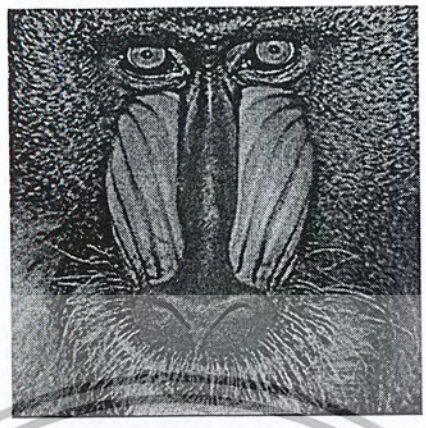
1.) การทดลองวงจรกรองแบบ Vertical Gradient

รูปที่ 4.46 ภาพต้นแบบ

รูปที่ 4.47 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.48 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

2.) การทดลองวงจรกรองแบบ Horizontal Gradient



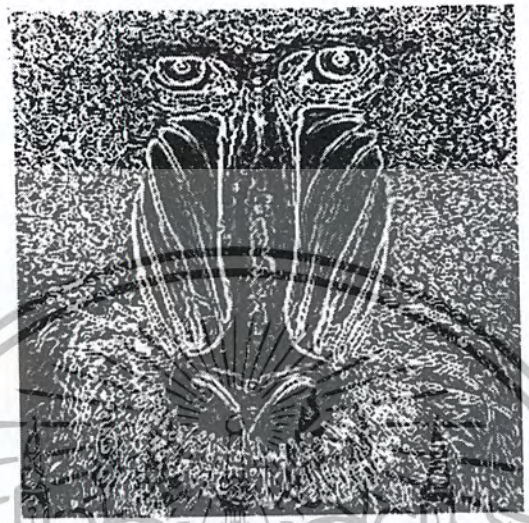
รูปที่ 4.49 ภาพต้นแบบ

รูปที่ 4.50 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.51 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

3.) การทดลอง Gradient Combining

เป็นการนำผลที่ได้จากการทดลอง Vertical Gradient และ Horizontal Gradient มาทำการรวมกัน ตามสมการที่ 4.1



รูปที่ 4.52 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.53 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.) การทดลอง Apply Threshold

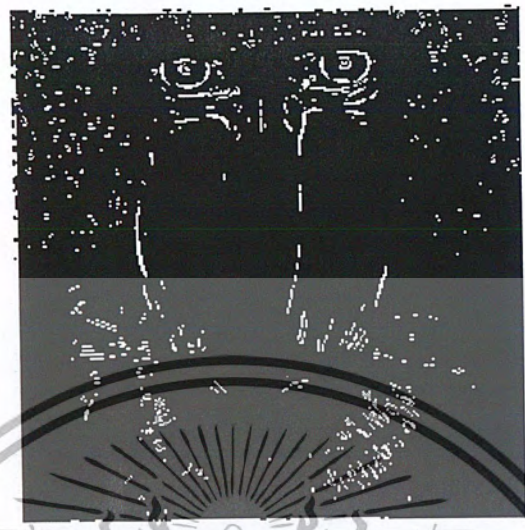
เป็นการนำผลที่ได้จากการทดลอง Gradient Combining มาทำการ Apply Threshold เพื่อปรับให้ภาพมีความเข้ม 2 ระดับ คือ ดำ และ ขาว

4.1.) กำหนดให้ Threshold = 50

รูปที่ 4.54 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.55 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.2.) กำหนดให้ Threshold = 80



รูปที่ 4.56 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.57 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.4.3 การทดลองวงจรกรองแบบ Sobel edge Detector ด้วยภาพ "sassy.jpg'

1.) การทดลองวงจรกรองแบบ Vertical Gradient

รูปที่ 4.58 ภาพต้นแบบ

รูปที่ 4.59 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.60 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

2.) การทดลองวงจรกรองแบบ Horizontal Gradient

รูปที่ 4.61 ภาพต้นแบบ

รูปที่ 4.62 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.63 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

3.) การทดลอง Gradient Combining

เป็นการนำผลที่ได้จากการทดลอง Vertical Gradient และ Horizontal Gradient มาทำการรวมกัน ตามสมการที่ 4.1

รูปที่ 4.64 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.65 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.) การทดลอง Apply Threshold

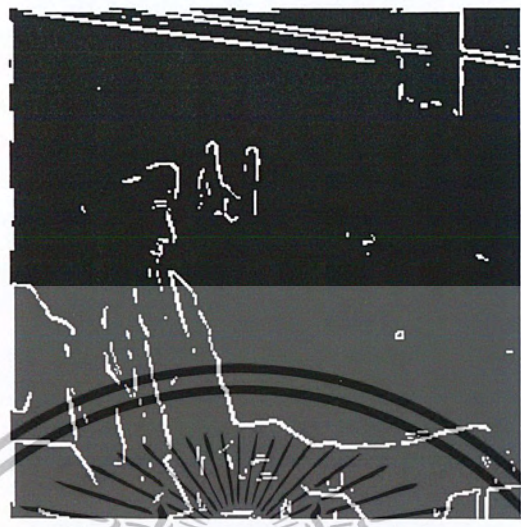
เป็นการนำผลที่ได้จากการทดลอง Gradient Combining มาทำการ Apply Threshold เพื่อปรับให้ภาพมีความเข้ม 2 ระดับ คือ ดำ และ ขาว

4.1.) กำหนดให้ Threshold = 50

รูปที่ 4.66 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.67 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.2.) กำหนดให้ Threshold = 80

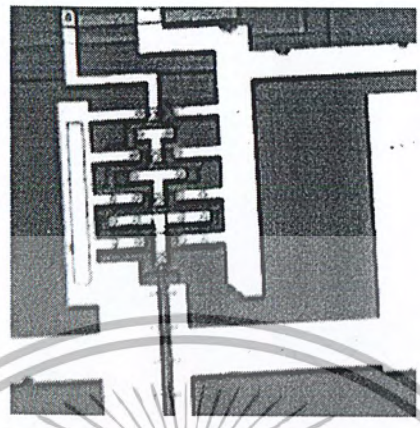


รูปที่ 4.68 ภาพที่ได้จากโปรแกรม Matlab

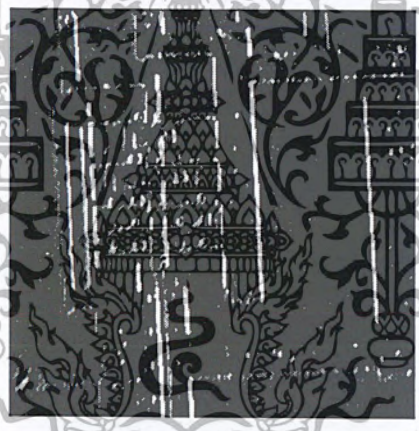
รูปที่ 4.69 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.4.4 การทดลองวงจรกรองแบบ Sobel edge Detector ด้วยภาพ "ic.jpg'

1.) การทดลองวงจรกรองแบบ Vertical Gradient



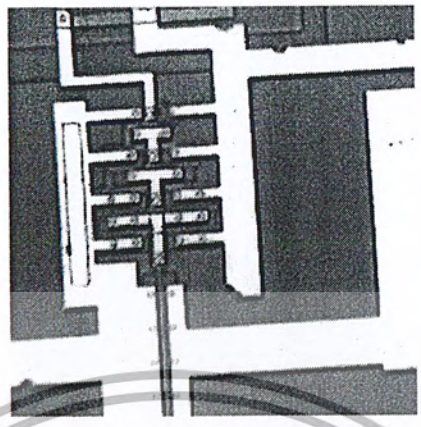
รูปที่ 4.70 ภาพต้นแบบ



รูปที่ 4.71 ภาพที่ได้จากโปรแกรม Matlab

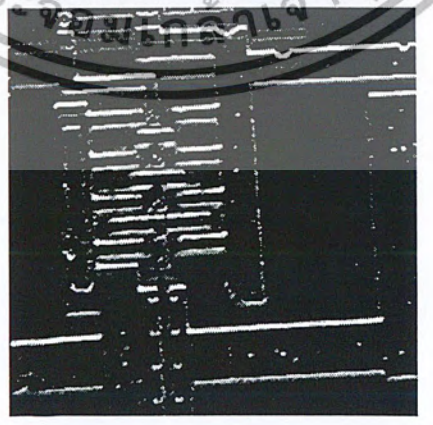
รูปที่ 4.72 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

2.) การทดลองวงจรกรองแบบ Horizontal Gradient



รูปที่ 4.73 ภาพต้นแบบ

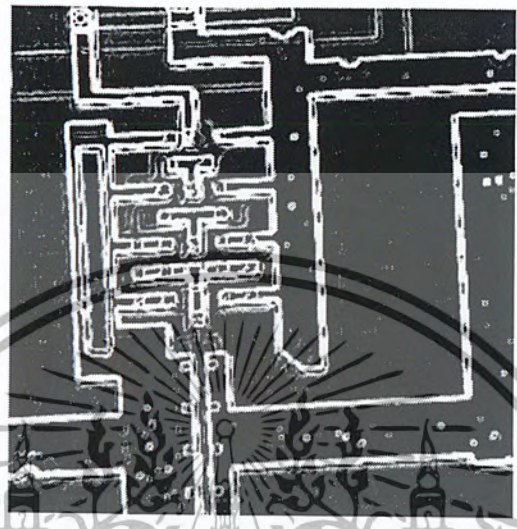
รูปที่ 4.74 ภาพที่ได้จากโปรแกรม Matlab



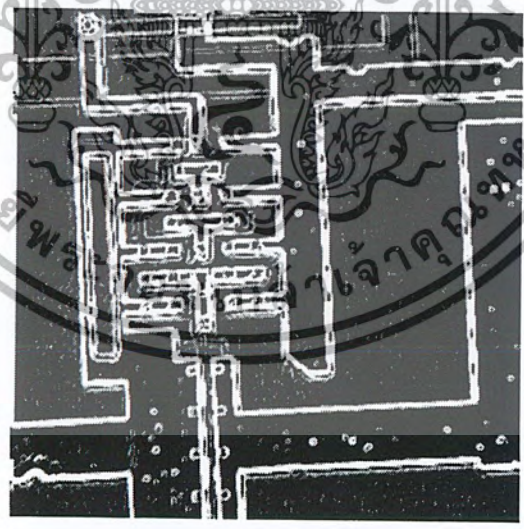
รูปที่ 4.75 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

3.) การทดลอง Gradient Combining

เป็นการนำผลที่ได้จากการทดลอง Vertical Gradient และ Horizontal Gradient มาทำการรวมกัน ตามสมการที่ 4.1



รูปที่ 4.76 ภาพที่ได้จากโปรแกรม Matlab

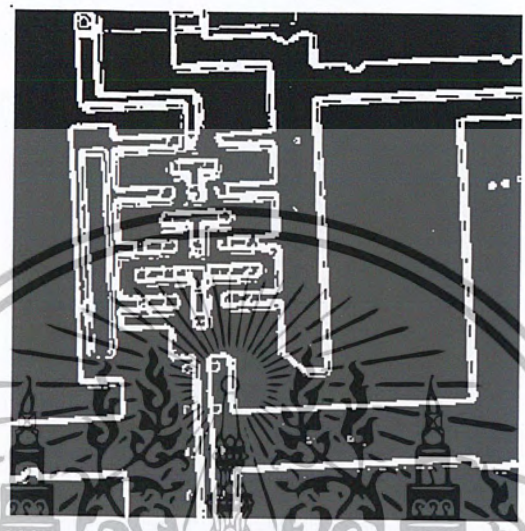


รูปที่ 4.77 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

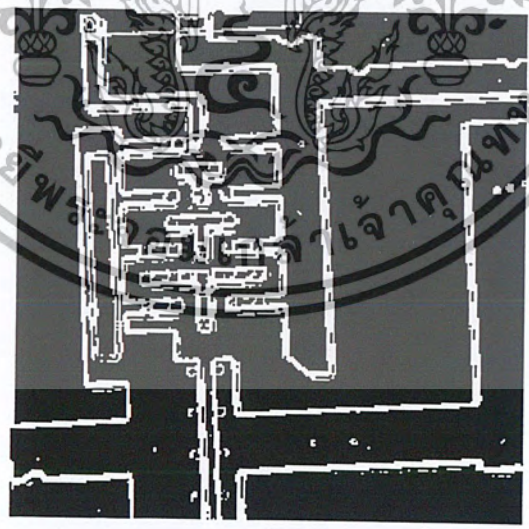
4.) การทดลอง Apply Threshold

เป็นการนำผลที่ได้จากการทดลอง Gradient Combining มาทำการ Apply Threshold เพื่อปรับให้ภาพมีความเข้ม 2 ระดับ คือ ดำ และ ขาว

4.1.) กำหนดให้ Threshold = 50

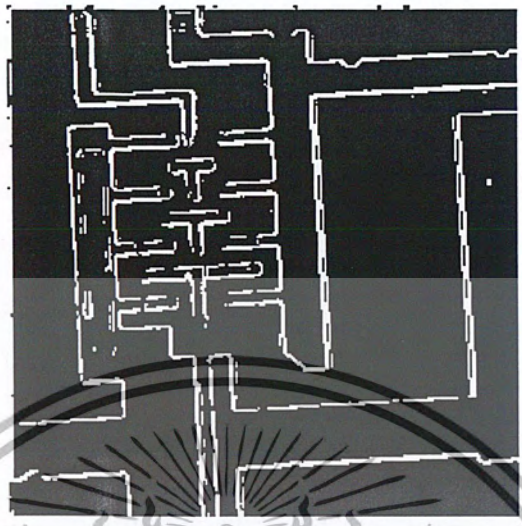


รูปที่ 4.78 ภาพที่ได้จากโปรแกรม Matlab

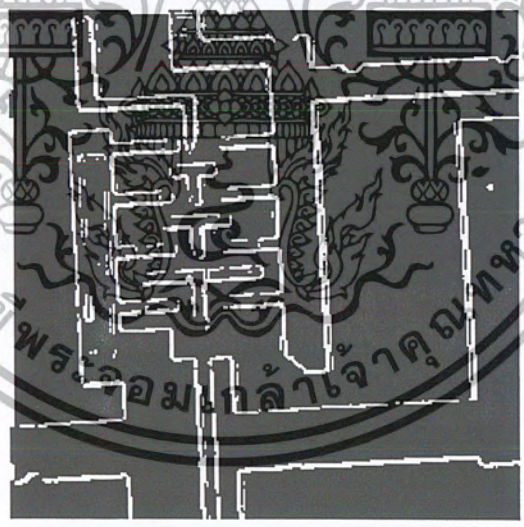


รูปที่ 4.79 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.2.) กำหนดให้ Threshold = 80



รูปที่ 4.80 ภาพที่ได้จากโปรแกรม Matlab



รูปที่ 4.81 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.4.5 การทดลองวงจรกรองแบบ Sobel edge Detector ด้วยภาพ "map.jpg'

1.) การทดลองวงจรกรองแบบ Vertical Gradient

รูปที่ 4.82 ภาพต้นแบบ

รูปที่ 4.83 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.84 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

2.) การทดลองวงจรกรองแบบ Horizontal Gradient

รูปที่ 4.85 ภาพต้นแบบ

รูปที่ 4.86 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.87 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

3.) การทดลอง Gradient Combining

เป็นการนำผลที่ได้จากการทดลอง Vertical Gradient และ Horizontal Gradient มาทำการรวมกัน ตามสมการที่ 4.1

รูปที่ 4.88 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.89 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.) การทดลอง Apply Threshold

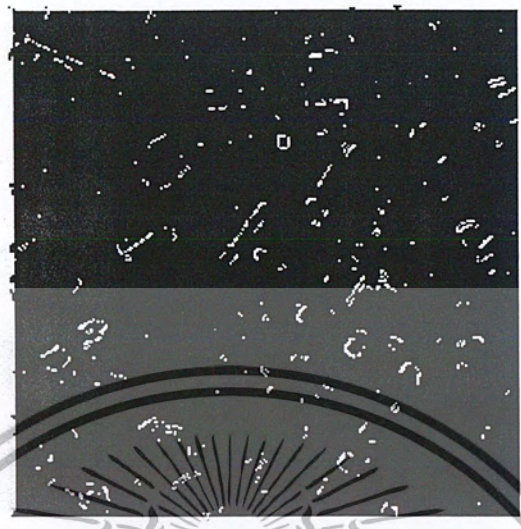
เป็นการนำผลที่ได้จากการทดลอง Gradient Combining มาทำการ Apply Threshold เพื่อปรับให้ภาพมีความเข้ม 2 ระดับ คือ ดำ และ ขาว

4.1.) กำหนดให้ Threshold = 50

รูปที่ 4.90 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.91 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

4.2.) กำหนดให้ Threshold = 80



รูปที่ 4.92 ภาพที่ได้จากโปรแกรม Matlab

รูปที่ 4.93 ภาพที่ได้จากการทำงานของฮาร์ดแวร์

บทที่ 5
บทวิจารณ์และบทสรุป

จากการศึกษาและทำการสร้างวงจรกรองสัญญาณภาพเชิงเลขที่ใช้โครงสร้างเลขคณิตกระจาย โดยทำการสร้างวงจรจรบน FPGA พบว่าโครงสร้างเลขคณิตกระจายสามารถนำมาประยุกต์ใช้งานกับการประมวลผลภาพได้ โดยให้ผลภาพที่ได้ออกมามีลักษณะคล้ายกับภาพที่ทำการจำลองการทำงานจากโปรแกรม Matlab ซึ่งภาพที่ได้ทั้งจากโปรแกรม Matlab และจากฮาร์ดแวร์ จะเกิดการ error นิดหน่อยตรงขอบภาพด้านบนและด้านซ้าย เนื่องมาจากช่วงขอบภาพนั้นไม่มีข้อมูลของ X(m,n-1), X(m,n-2), X(m-1,n), X(m-2,n)

สำหรับปัญหาที่เกิดขึ้น เนื่องมาจากข้อมูลภาพที่นำมาประมวลผลนั้นมีจำนวนของข้อมูลค่อนข้างมาก ทำให้ยุ่งยากและใช้เวลานานในการจัดการข้อมูล ทั้งข้อมูลภาพที่จะนำมาเก็บเข้า Rom ภาพ และข้อมูลภาพที่ได้หลังจากทำการประมวลผลเสร็จสิ้นแล้ว

จากการทดลองจะเห็นว่าสามารถสร้างวงจรกรองสัญญาณภาพโดยใช้ FPGA ได้ ซึ่งในส่วนของ FPGA ในการออกแบบนั้นจะใช้ภาษา VHDL ในการอธิบายลักษณะพฤติกรรมของวงจรที่ต้องการ จากนั้นจึงดาวน์โหลดข้อมูลทางลอจิกลงใน FPGA แล้วนำไปทดสอบการทำงาน จะเห็นว่า FPGA เหมาะสำหรับการออกแบบระบบต้นแบบเพราะ FPGA สามารถกำหนดฟังก์ชันการทำงานตามความต้องการของผู้ออกแบบได้ อีกทั้งการแก้ไขปรับปรุงยังทำได้ง่ายสะดวกและรวดเร็ว ทำให้การออกแบบระบบต้นแบบของวงจรทางดิจิตอลทำได้รวดเร็วยิ่งขึ้น

ภาคผนวก

ภาคผนวก ก

โปรแกรม Matlab

1. โปรแกรม Low-pass Filter

```
clc;  %clear screen
clear variables;
clear all;
I = imread('cameraman.tif');     %read image
a=256;      %    กำหนดขนาดภาพเป็น    a*a
s=imresize(I,[a a]);
b=double(s);
b1=b/256;
bb=round(b1*(2^7))/(2^7);
for m=1:a
    for n=1:a
      x(m+2,n+2)=bb(m,n);
  end
end
h1 = [1 2 1;2 4 2;1 2 1]/16;     % Mask ของ LPF
h=round(h1*(2^7))/(2^7);
for m=3:a+2
   for n=3:a+2
y(m-2,n-2)=(x(m,n)*h(1,1))+(x(m-1,n)*h(2,1))+(x(m-2,n)*h(3,1))...
        +(x(m,n-1)*h(1,2))+(x(m-1,n-1)*h(2,2))+(x(m-2,n-1)*h(3,2))...
        +(x(m,n-2)*h(1,3))+(x(m-1,n-2)*h(2,3))+(x(m-2,n-2)*h(3,3));
  end
end
y1=round(y*(2^7))/(2^7);
imshow(bb)      %โชว์ภาพต้นแบบ
figure
imshow(y1)      %โชว์ภาพที่ผ่านวงจรกรองแล้ว
```

2. โปรแกรม Sobel edge Detector

```
clc;  %clear screen
clear variables;
clear all;
I = imread('cameraman.tif');     %read image
a=256;      %    กำหนดขนาดภาพเป็น    a*a
s=imresize(I,[a a]);
b=double(s);
b1=b/256;
bb=round(b1*(2^7))/(2^7);
for m=1:a
    for n=1:a
      x(m+2,n+2)=bb(m,n);
  end
end
h = [-1 -2 -1;0 0 0;1 2 1]; %  Mask of Sobel
d=h';

for m=3:a+2          %vertical
   for n=3:a+2
ver(m-2,n-2)=(x(m,n)*h(1,1))+(x(m-1,n)*h(2,1))+(x(m-2,n)*h(3,1))...
        +(x(m,n-1)*h(1,2))+(x(m-1,n-1)*h(2,2))+(x(m-2,n-1)*h(3,2))...
        +(x(m,n-2)*h(1,3))+(x(m-1,n-2)*h(2,3))+(x(m-2,n-2)*h(3,3));
  end
end

for m=3:a+2          %horizontal
   for n=3:a+2
hor(m-2,n-2)=(x(m,n)*d(1,1))+(x(m-1,n)*d(2,1))+(x(m-2,n)*d(3,1))...
        +(x(m,n-1)*d(1,2))+(x(m-1,n-1)*d(2,2))+(x(m-2,n-1)*d(3,2))...
        +(x(m,n-2)*d(1,3))+(x(m-1,n-2)*d(2,3))+(x(m-2,n-2)*d(3,3));
  end
end
figure(1)
imshow(bb)
figure(2)
imshow(ver)
figure(3)
imshow(hor)
ss = (sqrt(ver.^2 +hor.^2));
figure(4)
imshow(ss)
cutoff=1.5;
for r = 1:256
    for c = 1:256
        if ss(r,c)>cutoff
           k(r,c)=1;
        else k(r,c)=0;
        end
   end
end
figure(5)
imshow(k)
```

3. โปรแกรมแปลงค่าภาพที่รับเข้ามาเป็นเลขไบนารี่เพื่อเก็บค่าในรอมภาพ

```
clc;  %clear screen
clear variables;
clear all;
I = imread('cameraman.tif');     %read image
a=256;      %    กำหนดขนาดภาพเป็น    a*a
s=imresize(I,[a a]);
b=double(s);
bb=b/256;
c=bb*(2^7);
for m=1:a
    for n=1:a
      x(m+2,n+2)=bb(m,n);
  end
end
pa=1;
pb=8;     %   แสดงทีละ   pb แถว
uj = 11;
while uj > 1;
clc;
    for ia = pa:1:pb
        for ib = 1:1:256          %256=ความกว้างของภาพ
            ss = c(ia,ib);
            bn = dec2bin(ss,8);
            fprintf('  "%s", \n',bn);

   end;
end;
fprintf(' pa =   %.0f      ,     pb =   %.0f   \n',pa,pb );
uj=input(' Enter 1 for exit  2 for loop  :');
pa=pa+8;      %   เพิ่มค่าแถวต่อไป
pb=pb+8;      %   เพิ่มค่าแถวต่อไป
end;
```

4. โปรแกรมหาค่าสัมประสิทธิ์ที่จะนำไปเก็บไว้ในรอม DA ของ Low-pass Filter

```
hg  = [1 2 1 2 4 2 1 2 1]/16;
pg=-1;
g=0;
xa = [ 0 1 ];
for p1a = 1:1:2
for p2a = 1:1:2
for p3a = 1:1:2
for p4a = 1:1:2
for p5a = 1:1:2
for p6a = 1:1:2
for p7a = 1:1:2
for p8a = 1:1:2
for p9a = 1:1:2
pg = pg+1;
mg =(hg(1)*xa(p1a))+(hg(2)*xa(p2a))...
    +(hg(3)*xa(p3a))+(hg(4)*xa(p4a))...
    +(hg(5)*xa(p5a))+(hg(6)*xa(p6a))...
    +(hg(7)*xa(p7a))+(hg(8)*xa(p8a))...
    +(hg(9)*xa(p9a));
f=4;      %บิทของทศนิยม
n=8;      %จำนวนบิททั้งหมด
          if mg<0
             a=round(mg*(2^f));
             aa=(2^n)+a;
          else
             aa=round(mg*(2^f));
          end
%pgg = dec2bin(pg,9);
mp = aa*(2^(-4));
mgg = fix(mp);
mmg = mp-mgg;
mgm = mmg*(2^(4));
mgm = dec2bin(aa,8);
fprintf('   "%s", \n',mgm);
end;
end;
end;
end;
end;
end;
end;
end;
end;
```

5. โปรแกรมหาค่าสัมประสิทธิ์ที่จะนำไปเก็บไว้ในรอม DA ของ Sobel edge Detector (Ver)

```
hg  = [1 0 -1 2 0 -2 1 0 -1];
pg=-1;
g=0;
xa = [ 0 1 ];
for p1a = 1:1:2
for p2a = 1:1:2
for p3a = 1:1:2
for p4a = 1:1:2
for p5a = 1:1:2
for p6a = 1:1:2
for p7a = 1:1:2
for p8a = 1:1:2
for p9a = 1:1:2
pg = pg+1;
mg =(hg(1)*xa(p1a))+(hg(2)*xa(p2a))...
    +(hg(3)*xa(p3a))+(hg(4)*xa(p4a))...
    +(hg(5)*xa(p5a))+(hg(6)*xa(p6a))...
    +(hg(7)*xa(p7a))+(hg(8)*xa(p8a))...
    +(hg(9)*xa(p9a));
f=4;     %บิทของทศนิยม
n=8;     %จำนวนบิททั้งหมด
         if mg<0
             a=round(mg*(2^f));
             aa=(2^n)+a;
         else
             aa=round(mg*(2^f));
         end
%pgg = dec2bin(pg,9);
mp = aa*(2^(-4));
mgg = fix(mp);
mmg = mp-mgg;
mgm = mmg*(2^(4));
mgm = dec2bin(aa,8);
fprintf('   "%s", \n',mgm);
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
```

6. โปรแกรมหาค่าสัมประสิทธิ์ที่จะนำไปเก็บไว้ในรอม DA ของ Sobel edge Detector (Hor)

```
h  = [1 2 1 0 0 0 -1 -2 -1];
hg';
pg=-1;
g=0;
xa = [ 0 1 ];
for p1a = 1:1:2
for p2a = 1:1:2
for p3a = 1:1:2
for p4a = 1:1:2
for p5a = 1:1:2
for p6a = 1:1:2
for p7a = 1:1:2
for p8a = 1:1:2
for p9a = 1:1:2
pg = pg+1;
mg =(hg(1)*xa(p1a))+(hg(2)*xa(p2a))...
    +(hg(3)*xa(p3a))+(hg(4)*xa(p4a))...
    +(hg(5)*xa(p5a))+(hg(6)*xa(p6a))...
    +(hg(7)*xa(p7a))+(hg(8)*xa(p8a))...
    +(hg(9)*xa(p9a));
f=4;      %บิทของทศนิยม
n=8;      %จำนวนบิททั้งหมด
          if mg<0
            a=round(mg*(2^f));
            aa=(2^n)+a;
          else
            aa=round(mg*(2^f));
          end
%pgg = dec2bin(pg,9);
mp = aa*(2^(-4));
mgg = fix(mp);
mmg = mp-mgg;
mgm = mmg*(2^(4));
mgm = dec2bin(aa,8);
fprintf('   "%s", \n',mgm);
end;
end;
end;
end;
end;
end;
end;
end;
end;
```

7. โปรแกรมเลือกค่าข้อมูล output ที่ถูกต้องจากโปรแกรม Model Sim

```
clear all;
clc;%clear screen
out_put_data=importdata('out_put.dat')
load_signal=importdata('load.dat')

count1 = 0;
for i = 1:1:length(out_put_data)
   if load_signal(i) == 0
      count1                = count1 + 1;
      position1(count1)     = i;
      out_put_real(count1) = out_put_data(i);
   %   fprintf(' %.0f \n',out_put_real(count1) );
end;

end;
%save out_putfrom_cut.dat dataout;

count2     = 0 ;
res1       = 1 ;
poch       = [1:1:length(out_put_real)];
poch(1)    = -1 ;
for i  = 1:1:length(out_put_real+1)
    res1 = res1 + 1 ;
    poch(res1) = position1(i);
    comp       = position1(i) - poch(i);
    if   comp == 1
         out_image(count2) = out_put_real(i);
    else count2 = count2 + 1;
         out_image(count2) = out_put_real(i);
    %fprintf(' %.0f \n',out_image(count2));
    j=j+1
    end;
end;
save out_put_hpf.dat out_image;
```

8. โปรแกรมจัดเรียงข้อมูลที่ได้จาก M ให้ออกมาเป็นภาพ

```
clc;
clear all;
a=importdata('out_put_hpf.dat')

n=256;   % dimension n*n
k=1;
    for i=1:n
          for j=1:n
         a_row(i,j)=a(k);
           if k<length(a)          % k= จำนวนข้อมูลทั้งหมด
                k=k+1;

end
end
end
s=a_row/64;
imshow(s)
```

ภาคผนวก ข

โปรแกรมภาษา VHDL

1. โปรแกรมจัดเรียงข้อมูลภาพ

```
library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_Unsigned.ALL;
Entity frame_manage is
Port( image_data      : in  std_logic_vector(7 downto 0);
      lr_img          : in  std_logic;
      xmn,xmn_1,xmn_2 : out std_logic_vector(7 downto 0)
      xm_1n,xm_1n_1   : out std_logic_vector(7 downto 0)
      xm_1n_2,xm_2n   : out std_logic_vector(7 downto 0)
      xm_2n_1,xm_2n_2 : out std_logic_vector(7 downto 0));
End;

Architecture rtl  of frame_manage  is
type de is array (0 to 255) of std_logic_vector(7 downto 0);
signal kly : de;
signal k2y : de;
signal bu,p1i,p2i,p3i,p4i,p5i,p6i : std_logic_vector(7 downto 0);
Begin
Process(lr_img)
        begin
          if lr_img='0' then
          p2i<=p1i;
          p4i<=p3i;
          p6i<=p5i;
  kly(255)<=kly(254);
  kly(254)<=kly(253);
  kly(253)<=kly(252);
  kly(252)<=kly(251);
  kly(251)<=kly(250);
             .
             .0
             .
  kly(4)<=kly(3);
  kly(3)<=kly(2);
  kly(2)<=kly(1);
  kly(1)<=kly(0);
-------------------------
  k2y(255)<=k2y(254);
  k2y(254)<=k2y(253);
  k2y(253)<=k2y(252);
  k2y(252)<=k2y(251);
  k2y(251)<=k2y(250);
             .
             .
             .

  k2y(4)<=k2y(3);
  k2y(3)<=k2y(2);
  k2y(2)<=k2y(1);
  k2y(1)<=k2y(0);
-----------------------
  p5i<=k2y(255);
```

```
 p3i<=kly(255);
 k2y(0)<=kly(255);
 kly(0)<=bu;
 p1i<=bu;
 bu<=image_data;
end if;
end process;
xmn<=bu;
xmn_1<=p1i;
xmn_2<=p2i;
xm_1n<=kly(255);
xm_1n_1<=p3i;
xm_1n_2<=p4i;
xm_2n<=k2y(255);
xm_2n_1<=p5i;
xm_2n_2<=p6i;
end;
```

2. โปรแกรมเก็บค่าใน Rom DA

```
library ieee;
use ieee.std_logic_1164.all;
package rom_hpf is
        constant  rom_hpf_width  : integer := 8;
        constant  rom_hpf_length : integer := 512;
        subtype   rom_hpf_word  is std_logic_vector (rom_hpf_width-1
downto 0);
        type      rom_hpf_table is array (0 to rom_hpf_length - 1) of
rom_hpf_word;
        constant  rom_hpf       : rom_hpf_table :=
rom_hpf_table'("00000000",
  "00000000",
  "11110000",
  "11110000",
  "00000000",
  "01000000",
  "00110000",
  "00110000",
  "11110000",
  "11110000",
  "11100000",
  "11100000",
      .
      .
      .
  "11110000",
  "11110000",
  "11100000",
  "11100000",
  "11100000",
  "11100000",
  "11010000",
  "11010000",
  "11100000",
  "11100000",
  "00100000",
  "00010000",
  "00010000");
  end;
```

3. โปรแกรมการทำงานของ DA

```
library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_Unsigned.ALL;
Entity add_acc is
Port( add_acc_in          : in  std_logic_vector(7 downto 0);
      s_a,clacc,lacc,lr   : in  std_logic;
      add_acc_out         : out std_logic_vector(7 downto 0));
End;

Architecture add_acc_a of add_acc is
Signal add_out,acc_out : std_logic_vector(7 downto 0);
Begin

addp : Process (s_a,add_acc_in,acc_out)
         Variable s_sub,sum : std_logic_vector(7 downto 0);
         Variable s         : std_logic_vector(8 downto 0);
          Begin
            if   s_a = '0' then
                 s:=("0"&add_acc_in)+acc_out;
                 sum(7):=s(8) xor add_acc_in(7) xor acc_out(7) ;
                 sum(6 downto 0):=s(7 downto 1);
            else s_sub:=(add_acc_in xor "11111111");
                 s_sub:=s_sub+1;
                 sum(7 downto 0):=acc_out(7 downto 0)+s_sub(7 downto
0);
            end if;
            add_out<=sum;
          End Process addp ;

accp : Process (clacc,lacc)
         Begin
         if  clacc='0' then
             acc_out<=(others=>'0');
         elsif lacc'event and lacc='1' then
             acc_out<=add_out;
         end if;
         End Process accp ;

Buff : Process (lr)
       begin
         if lr='0' then
            add_acc_out<=acc_out;
         end if;
       End process Buff ;
End;
```

4. โปรแกรมการทำงานของ PISO รวมกัน 9 ชุด สำหรับชี้ Address ให้กับ Rom DA

```
library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_Unsigned.ALL;
Entity shift is
  port( lr,clk     : in  std_logic;
      xmn_s,xmn1_s,xmn2_s      : in  std_logic_vector(7 downto 0);
      xm1n_s,xm1n1_s,xm1n2_s   : in  std_logic_vector(7 downto 0);
      xm2n_s,xm2n1_s,xm2n2_s   : in  std_logic_vector(7 downto 0);
      data_shift_out           : out std_logic_vector(8 downto 0));
  End;
Architecture shifta of shift is
Signal s1h,s2h,s3h,s4h,s5h,s6h,s7h,s8h,s9h : std_logic_vector(7
downto 0);
begin
process (clk)
  begin
    if lr='0' then
      s1h<=xmn_s;   s2h<=xmn1_s;   s3h<=xmn2_s;
      s4h<=xm1n_s;  s5h<=xm1n1_s;  s6h<=xm1n2_s;
      s7h<=xm2n_s;  s8h<=xm2n1_s;  s9h<=xm2n2_s;
      elsif clk'event and clk = '1' then
      s1h(6 downto 0)<=s1h(7 downto 1);
      s2h(6 downto 0)<=s2h(7 downto 1);
      s3h(6 downto 0)<=s3h(7 downto 1);
      s4h(6 downto 0)<=s4h(7 downto 1);
      s5h(6 downto 0)<=s5h(7 downto 1);
      s6h(6 downto 0)<=s6h(7 downto 1);
      s7h(6 downto 0)<=s7h(7 downto 1);
      s8h(6 downto 0)<=s8h(7 downto 1);
      s9h(6 downto 0)<=s9h(7 downto 1);
      data_shift_out(8)<=s1h(0);
      data_shift_out(7)<=s2h(0);
      data_shift_out(6)<=s3h(0);
      data_shift_out(5)<=s4h(0);
      data_shift_out(4)<=s5h(0);
      data_shift_out(3)<=s6h(0);
      data_shift_out(2)<=s7h(0);
      data_shift_out(1)<=s8h(0);
      data_shift_out(0)<=s9h(0);
      end if;
 end process;
 end;
```

5. โปรแกรมการทำงานของวงจรกรองภาพด้วยเลขคณิตกระจาย

```
library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_Unsigned.ALL;
Use work.rom_hpf.all;
Use work.romimage256.all;

Entity hpf is
Port( sys_clk           : in  std_logic;
      start_new_frame   : out std_logic;
      ymn               : out std_logic_vector(7 downto 0));
End;

Architecture hpf_a of hpf is
-- Component declaration

component counter
port(sc               :in std_logic;
     addr_img         :out std_logic_vector(15 downto 0);
     start_new_frame :out std_logic);
end component;

component control
Port (   sys_clk                          : in  std_logic;
       clk,s_a,lacc,lr_img,lr,clacc,sc  : out std_logic);
end component;

component frame_manage
Port( image_data                  : in  std_logic_vector(7 downto 0);
      lr_img                      : in  std_logic;
      xmn,xmn_1,xmn_2xm_1n        : out std_logic_vector(7 downto 0));
      xm_1n_1,xm_1n_2             : out std_logic_vector(7 downto 0));
      xm_2n,xm_2n_1,xm_2n_2       : out std_logic_vector(7 downto 0));

end component;

component shift
port( lr,clk                      : in  std_logic;
      xmn_s,xmn1_s,xmn2_s         : in std_logic_vector(7 downto 0);
      xm1n_s,xm1n1_s,xm1n2_s      : in std_logic_vector(7 downto 0);
      xm2n_s,xm2n1_s,xm2n2_s      : in std_logic_vector(7 downto 0);
      data_shift_out              : out std_logic_vector(8 downto 0));
end component;

component add_acc
Port( add_acc_in                  : in  std_logic_vector(7 downto 0);
      s_a,clacc,lacc,lr           : in  std_logic;
      add_acc_out                 : out std_logic_vector(7 downto 0));
end component;

component   rom_hpf_out
    port (romhpf_in               : in std_logic_vector (8 downto 0);
          romhpf_out              : out rom_hpf_word);
end component;

component romimage256_out
     port (romim_in               : in std_logic_vector (15 downto 0);
           romim_out              : out romimage256_word);
end component;
```

```
   signal clka,s_aa,lacca,lr_imga,lra,clacca,sca  :  std_logic;
   signal add_imga              : std_logic_vector(15 downto 0);
   signal image_dataa           : std_logic_vector(7 downto 0);
   signal shift_outa            : std_logic_vector(8 downto 0);
   signal xmna,xmn_1a,xmn_2a          : std_logic_vector(7 downto 0);
   signal xm_1na,xm_1n_1a,xm_1n_2a    : std_logic_vector(7 downto 0);
   signal xm_2na,xm_2n_1a,xm_2n_2a    : std_logic_vector(7 downto 0);
   signal signal rom_hpf_outa         : std_logic_vector(7 downto 0);
Begin
-- Component instantiation

controlU : control
portmap(sys_clk,clka,s_aa,lacca,lr_imga,lra,clacca,sca);
counterU : counter  port map(sca,add_imga,start_new_frame);
romimage256_outU : romimage256_out port map(add_imga,image_dataa);
frame_manageU : frame_manage
portmap(image_dataa,lr_imga,xmna,xmn_1a,xmn_2a,xm_1na,xm_1n_1a,xm_1n_
2a,xm_2na,xm_2n_1a,xm_2n_2a);
shiftU : shift
portmap(lra,clka,xmna,xmn_1a,xmn_2a,xm_1na,xm_1n_1a,xm_1n_2a,xm_2na,x
m_2n_1a,xm_2n_2a,shift_outa);
rom_hpf_outU : rom_hpf_out port map(shift_outa,rom_hpf_outa);
add_accU : add_acc port map(rom_hpf_outa,s_aa,clacca,lacca,lra,ymn);
end;
```

ภาคผนวก ค

โปรแกรมภาษาซี

โปรแกรมที่ใช้ติดต่อระหว่างคอมพิวเตอร์กับ FPGA โดยผ่านการ์ดอินเตอร์เฟส 8255 แบบ PCI

```
#include <string.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <dos.h>
int main()
{
clrscr();
FILE *stream;
char enter[1]="\n";
char bitp[10];
int DATA[65][257],DATA[65][257];
int row,col;
long int n;
long int pixel=65536;
char buffer[6];
char Tmp[3];
char temp[10];
if ((stream = fopen("C:\\image2\\input.TXT", "r"))== NULL)
   {
      fprintf(stderr, "Cannot open output file.\n");
      return 1;
   }

  for(n=1;n<=pixel;n++)
   {
      row=(n/256)+1;
      col=n%256;
      if (col==0)
      {
       row--;
       col=256;
      }
      fread(buffer,6, 1, stream);
      Tmp[0]=buffer[1];
      Tmp[1]=buffer[2];
      Tmp[2]=buffer[3];
      DATA[row][col]=atoi(Tmp);
      outport(0xf300,0);
      outportb(0xf300,DATA1[row][col]);     //1
      outportb(0xf301,2);                   //2
      outportb(0xf301,0);                   //3
      outportb(0xf301,1);                   //4
      outportb(0xf301,0);                   //5
      outportb(0xf301,1);                   //6
      outportb(0xf301,0);                   //7
      outportb(0xf301,1);                   //8
      outportb(0xf301,0);                   //9
      outportb(0xf301,1);                   //10
      outportb(0xf301,0);                   //11
      outportb(0xf301,4);                   //12
      DATA[row][col]=inportb(0xf301);       //13
```

```
      outport(0xf301,0);                    //14
   }
if ((stream = fopen("C:\\image2\\output.txt", "w+"))== NULL)
   {
      fprintf(stderr, "Cannot open output file.\n");
      return 1;
   }
for(n=1;n<=pixel;n++)
 {
     row=(n/256)+1;
     col=n%256;
     if (col==0)
     {
      row1-;
      col=256;
     }
     ltoa(DATA[row][col],temp,10);
     fprintf(stream, "%s\n",temp)
 }
   fclose(stream);
  return 0;
}
```

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สามารถสำเร็จลุล่วงลงได้ ด้วยความช่วยเหลือและชี้แนะแนวทางจากหลายท่าน ผู้จัดทำขอขอบพระคุณอาจารย์ที่ปรึกษา รศ.ดร. กอบชัย เดชหาญ และ อาจารย์ศรวัฒน์ ชิวปรีชา ที่ให้คำปรึกษาด้านการใช้งาน FPGA ความช่วยเหลือด้านข้อมูลและอุปกรณ์ในการทำปริญญานิพนธ์มาโดยตลอด ขอขอบคุณ คุณปิยะพันธ์ คงเศรษฐ ที่ให้ความช่วยเหลือทางด้านโปรแกรมภาษาซีและการ์ดอินเตอร์เฟส 8255 ตลอดจนเพื่อนๆห้อง T ทุกคนสำหรับความมีน้ำใจที่คอยให้ความช่วยเหลือและให้กำลังใจในหลายๆด้าน ซึ่งทำให้ปริญญานิพนธ์สำเร็จลุล่วงไปได้ด้วยดี ผู้จัดทำขอขอบพระคุณทุกท่านไว้เป็นอย่างสูงไว้ ณ ที่นี้ด้วย

สุดท้ายนี้ขอขอบพระคุณ บิดา มารดา ที่สนับสนุนลูกๆ ทั้งทางด้านการศึกษา การเงิน ความรักและความเข้าใจ ที่ช่วยให้ลูกๆ เติบโตและมีโอกาสได้ศึกษาในสถาบันอันมีชื่อเสียงแห่งนี้

นายอภิรักษ์ ปานชื่น
นายอมรเทพ สนินัด

หนังสืออ้างอิง

[1] สมจินต์ ทองปลิว, "การสร้างวงจรกรองสัญญาณเชิงเลข 2 มิติโดย Distributed Arithmetics",คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, พ.ศ.2538

[2] S.Sjoholm and L.Lindh, "VHDL for Designers", PRENTICE HALL, 1997.

[3] S.Brown and Z.Vranesic, "Fundametals of digital logic with VHDL design", McGraw-HIL'L, 2000

[4] Rafael C.Gonzalez and Richard E.Woods, 'Digital Image Processing", Prentice Hall, 2001

[5] A.Peled and B.Liu, "A new hardware relization of digital filter', IEEE Trans. On A.S.S.P., Vol. ASSP-22, pp. 456-462, December 1974

[6] A.Peled and B.Liu, "Digital Signal Processing", John Wiley & Sons, pp. 212-226, 1976.

[7] S.A.White, "Application of Distributed Arithmetic", IEEE ASSP magazine, Vol. 6, No.3, pp. 4-9, July 1989.

[8] G.K.Ma, F.J.Taylor, "Multiplier Policies for Digital Signal Processing", IEEE ASSP magazine, Vol. 7, No.1, pp. 6-20, January 1990.

[9] C.F.N.Cowan, Stewart G.Smith and J.H.Elliott, "A Digital Adaptive Filter Using a Memory Accumulator Architecture", IEEE Trans. On A.S.S.P., Vol. ASSP-31, June 1983.

[10] C.S.Burrus, "Digital Filter Sturcture Described by Distributed Arithmetic", IEEE on circuit and systems, Vol. CAS-24, No.12, December 1977.

[11] S.Zohar, "A VLSI Implementation of a Digital Filter Based on Distributed Arithmetic", IEEE ASSP, Vol. 37, No.1, January 1989.

[12] H.Jaggernauth and A.N.Venetsanopoulos, "Real-Time Image Processing Through Distributed Arithmetic", IEEE Trans. On Circuit and systems, Vol.1, pp. 394-397, May 1983.

[13] P.M.Embree and B.Kimble, "C Language Algorithm for Digital Signal Processing", Prentice Hall, pp. 393-400, 1991.

[14] J.S.Lim, "Two-Dimensional Signal and Image Processing", Prentice Hall, 1990.

[15] D.F.Elliott, "Handbook of Digital Signal Processing", Acadimic press, pp. 964-772, 1987.

[16] K.S.Lin, "Texas Instruments Digital Signal Processing", Prentice Hall, pp. (F)2-(F)9, 1988.

[17] C.E.Sporck, "National Semiconductor F100K ECL Logic Databook", pp.(3)199-(3)205, 1990.