การออกแบบตัวกรองเชิงเลขรีเคอร์ซีฟ
โดยใช้คณิตศาสตร์แบบการกระจายและสร้างบน FPGA

AN FPGA IMPLEMENTATION OF RECURSIVE DIGITAL FILTER WITH DISTRIBUTED ARITHMETIC

มนูญ สันถวะคุปต์
MANOON SANTAWAKOOP

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2547

ISBN 974-9709-33-0

การออกแบบตัวกรองเชิงเลขรีเคอร์ซีฟ
โดยใช้คณิตศาสตร์แบบการกระจายและสร้างบน FPGA

AN FPGA IMPLEMENTATION OF RECURSIVE DIGITAL FILTER
WITH DISTRIBUTED ARITHMETIC

มนูญ สันถวะคุปต์
MANOON SANTAWAKOOP

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์มหาบัณฑิต
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2547

ISBN 974 - 9709 – 33 - 0

AN FPGA IMPLEMENTATION OF RECURSIVE DIGITAL FILTER WITH DISTRIBUTED ARITHMETIC

MANOON SANTAWAKOOP

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF ENGINEERING IN

ELECTRONICS ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2004

ISBN 974 - 9709 – 33 - 0

COPYRIGHT 2004

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

หัวข้อวิทยานิพนธ์	การออกแบบตัวกรองเชิงเลขรีเคอร์ซีฟโดยใช้คณิตศาสตร์แบบการกระจายและสร้างบน FPGA
นักศึกษา	นายมนูญ สันถวะคุปต์
รหัสนักศึกษา	44061319
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมอิเล็กทรอนิกส์
พ.ศ.	2547
อาจารย์ผู้ควบคุมวิทยานิพนธ์	ศ.ดร.วัลลภ สุระกำพลธร

บทคัดย่อ

วิทยานิพนธ์นี้นำเสนอการสร้างวงจรกรองเชิงเลขรีเคอร์ซีฟ โดยใช้ทฤษฎีคณิตศาสตร์แบบการกระจาย ทำการสร้างวงจรกรองโดยใช้อุปกรณ์ FPGA วิทยานิพนธ์นี้แสดงให้เห็นถึงประโยชน์ของการใช้ทฤษฎีคณิตศาสตร์แบบการกระจาย เพื่อเพิ่มความเร็วในการประมวลผลของวงจรกรองสัญญาณเชิงเลข นอกจากนี้ยังได้เสนอวิธีการแก้ไขความผิดพลาดที่เกิดขึ้นจากการคำนวณด้วยวิธีการจัดสเปกตรัมความผิดพลาดของสัญญาณ หรือเรียกอีกอย่างหนึ่งคือการป้อนกลับความผิดพลาด ด้วยโครงสร้างคณิตศาสตร์แบบการกระจายกับการป้อนกลับความผิดพลาด จะทำให้สามารถสร้างวงจรกรองเชิงเลขได้โดยไม่ต้องเปลี่ยนสถาปัตยกรรมภายในของคณิตศาสตร์แบบการกระจาย เป็นผลให้ไม่จำเป็นต้องเพิ่มอุปกรณ์ส่วนอื่นแต่อย่างใดในการออกแบบ นอกจากนี้ยังสามารถทำการจำลองการทำงานเบื้องต้นได้บนคอมพิวเตอร์ และให้ประสิทธิภาพด้านความเร็วและขนาดของวงจรด้วย FPGA ผลการทดสอบ, บทสรุปงานวิจัยและข้อเสนอแนะการทำงานของวงจรกรองเชิงเลขที่ออกแบบได้แสดงไว้ในส่วนท้ายของวิทยานิพนธ์

Thesis Title	An FPGA Implementation of Recursive Digital Filter with Distributed Arithmetic
Student	Mr. Manoon Santawakoop
Student ID.	44061319
Degree	Master of Engineering
Programme	Electronics Engineering
Year	2004
Thesis Advisor	Prof. Dr. Wanlop Surakampontorn

ABSTRACT

This thesis proposes the invention of recursive digital filter using distributed mathematic theorem implement on FPGA. As a result, the function of distributed mathematics leads to the increase in processing speed. Moreover, the arrangement of error spectrum of signal or error feedback is employed to correct the error. With the distributed mathematics, there is not any requirement to modify the system when the error feedback part is included. The simulation results show the efficiency of the system in the aspect of speed and size, and the results are also substantiated by real implementation on the FPGA.

กิตติกรรมประกาศ

ขอกราบขอบพระคุณ ศาสตราจารย์ ดร.วัลลภ สุระกำพลธร อาจารย์ที่ปรึกษาที่เป็นผู้ที่ได้ชี้นำแนวทางในการดำเนินงานวิจัยด้วยดีเสมอมา อีกทั้งยังได้ให้ความเมตตากรุณาในการอบรมสั่งสอนไม่เพียงแต่ความรู้ทางด้านวิชาการเท่านั้น แต่ยังรวมถึงประสบการณ์ที่ได้ถ่ายทอดอันพึงจะเป็นประโยชน์อย่างยิ่งต่อการดำเนินชีวิตในภายหน้า รวมทั้งยังได้แนะนำแนวทางในการแก้ไขปัญหาต่างๆ ที่เกิดขึ้นในระหว่างการดำเนินงานวิจัย จึงทำให้การจัดทำวิทยานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงสมบูรณ์ขึ้นมาได้

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.เกียรติศักดิ์ คมวัชระ ที่ได้กรุณาให้คำแนะนำในสิ่งที่เป็นประโยชน์ต่อการดำเนินงานวิจัย และขอขอบพระคุณ คุณเฉลิมภัณฑ์ ฟองสมุทร ผู้ซึ่งคอยดูแลแนะนำแนวทางวิจัยและถ่ายทอดประสบการณ์ต่างๆอย่างใกล้ชิด ตลอดจนแนวคิดทางวิชาการมากมายหลายแขนงที่ได้ถ่ายทอดอย่างดีเสมอมา ขอขอบคุณ คุณอดุลย์ ขันติชนะกุล และคุณวรินทร์ สุดคนึง ที่ได้เป็นแนวทางแบบอย่างที่ดีในการทำงาน และยังชี้แนะปัญหาต่างๆในการพัฒนางานวิจัยให้สามารถก้าวหน้าได้ ขอขอบคุณพี่ๆและผู้ร่วมงานทุกท่านในห้องปฏิบัติการ Mixed Signal Processing ตลอดจนเพื่อนสนิท-มิตรแท้ ที่คอยให้กำลังใจตลอดเวลาที่มีปัญหาและให้ความช่วยเหลือในทุกๆ ด้านแก่ข้าพเจ้า ในการดำเนินงานวิจัย ผลักดันและการปรับปรุงวิทยานิพนธ์ฉบับนี้ให้เสร็จสมบูรณ์

ขอขอบพระคุณ องค์การร่วมมือระหว่างประเทศของญี่ปุ่น (JICA) (ภายใต้โครงการสำนักวิจัยการสื่อสารและเทคโนโลยีสารสนเทศ) และกองทุนสนับสนุนการวิจัย (ภายใต้โครงการที่ RTA/04/2543) ที่ให้การสนับสนุนทางด้านทุนทรัพย์ และเครื่องมือในการดำเนินการวิจัยตลอดมา

สุดท้ายนี้ขอกราบขอบพระคุณ คุณพ่อชัยวัฒน์ คุณแม่รวมพร สันถวะคุปต์ ผู้ซึ่งรักและเป็นห่วงคอยเคียงข้างให้กำลังใจในยามท้อแท้และมีอุปสรรค์เสมอมา และขอขอบคุณผู้มีพระคุณทุกท่านที่ไม่ได้เอ่ยนามมา ณ. ที่นี้ด้วย ที่ได้มอบความรัก ความห่วงใย และกำลังใจ รวมถึงได้ให้การสนับสนุนในการศึกษาแก่ข้าพเจ้ามาโดยตลอด คุณค่าและประโยชน์อันพึงเกิดขึ้นจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ บิดา มารดา ตลอดจนผู้มีพระคุณทุกท่าน ส่วนข้อผิดพลาดประการใดในวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอน้อมรับไว้

มนูญ สันถวะคุปต์

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

หากกล่าวถึงกระบวนการประมวลผลสัญญาณเชิงเลขแล้ว การกรองสัญญาณเชิงเลขมีบทบาทอย่างมากในการออกแบบวงจรระบบที่มีการทำงานแบบเชิงเลข โดยพื้นฐานทางคณิตศาสตร์ของการประมวลผลสัญญาณใดๆ ไม่ว่าจะเป็นสัญญาณเชิงอุปมาน (Analog Signal) หรือ สัญญาณเชิงเลข (Digital Signal) ก็ตาม จะเป็นกระบวนการกระทำกับสัญญาณนั้นๆในลักษณะเชิงพีชคณิต คือ การบวกกัน (Addition) และ การคูณกัน (Multiplication) เป็นส่วนใหญ่ ซึ่งเมื่อกระบวนการทางคณิตศาสตร์ถูกทำการสร้างขึ้นเป็นระบบ (Implementation) หรือวงจรไฟฟ้าแล้วจะพบว่าระบบดำเนินการที่เกี่ยวกับการคูณสัญญาณจะมีขนาดของระบบหรือวงจรที่มีขนาดใหญ่ซึ่ง หากพิจารณาโดยเฉพาะถึงการคูณสัญญาณในรูปแบบเชิงเลขแล้ว ถ้าต้องการให้การทำงานเชิงเลขมีความแม่นยำที่มากขึ้น จำเป็นต้องใช้ขนาดบิตข้อมูลที่มากขึ้น เป็นผลให้ขนาดของวงจรคูณจะมีขนาดที่ใหญ่มากยิ่งขึ้นตามระดับบิตข้อมูลทางดิจิตอล

ดังนั้นจึงได้มีผู้คิดค้นกระบวนการทางคณิตศาสตร์หลากหลายวิธีเพื่อแก้ปัญหาในการสร้างระบบประมวลผลดิจิตอล รวมทั้งวิธีการในการออกแบบหน่วยคำนวณทางคณิตศาสตร์ให้มีขนาดของวงจรที่เล็กลงหรือเพิ่มประสิทธิภาพบางด้าน หนึ่งในกระบวนการทางคณิตศาสตร์ที่ถูกคิดค้นขึ้นเพื่อการสร้างรูปแบบการคูณกันของสมการพีชคณิตคือ “คณิตศาสตร์แบบการกระจาย (Distributed Arithmetic)” โดยมักจะถูกนำมาใช้สร้างระบบวงจรเชิงเลขที่มีการทำงานแบบพีชคณิตของการบวกและการคูณเป็นอย่างมาก อีกทั้งยังให้ประสิทธิภาพด้านขนาดวงจรที่เล็กลงและการทำงานที่รวดเร็ว เป็นผลให้รูปแบบคณิตศาสตร์แบบการกระจายมีความเหมาะสมกับระบบการประมวลผลสัญญาณเชิงเลข

แต่ด้วยข้อจำกัดทางขนาดความยาวของข้อมูลแบบดิจิตอล (Finite Word-length Effect) ทำให้การดำเนินการทางคณิตศาสตร์มีความคลาดเคลื่อนไปจากค่าจริง กระบวนการแก้ไขความผิดพลาดโดย การป้อนกลับความผิดพลาด (Error Feedback) จึงได้ถูกนำมาใช้ประกอบกับวิทยานิพนธ์นี้ด้วย

ในปัจจุบันและอนาคต แนวโน้มการรวมรูปแบบการออกแบบวงจรผสมสัญญาณ (Mixed-Signals) ขึ้นเป็นวงจรรวมที่ใช้เฉพาะงานหรือ เอสิค (ASIC: Application Specific Integrated Circuit) ต้องอาศัยเวลาและค่าใช้จ่ายในการออกแบบสูงซึ่งไม่เหมาะกับงานวิจัยและพัฒนาในขั้นต้น จึงหันมาให้ความสนใจในอุปกรณ์ เอฟพีจีเอ (FPGAs: Filed Programmable Gate Arrays)

ซึ่งเป็นอุปกรณ์ที่สามารถโปรแกรมให้ทำงานเป็นวงจรต่างๆได้ตามต้องการด้วยวิธีการออกแบบในลักษณะผังวงจร (Schematic Edition) และการออกแบบด้วยภาษาอธิบายฮาร์ดแวร์(Hardware Description Language: HDL) จึงทำให้เวลาในการออกแบบและค่าใช้จ่ายที่ต้องใช้ลดลงไปมาก และเมื่อผ่านการโปรแกรมลงในเอฟพีจีเอแล้ว การทดสอบการทำงานของวงจรจริงด้วยการวัดสัญญาณจริงก็ทำได้อย่างสะดวก และยังสามารถนำผลที่ผ่านการทดสอบไปใช้ในการออกแบบเป็นวงจรรวมที่ใช้เฉพาะงานได้ในอนาคต อย่างไรก็ดีอุปกรณ์ชนิดนี้จะมีจำนวนเกตที่จำกัดขึ้นอยู่กับราคาของอุปกรณ์ ถ้าต้องการอุปกรณ์ที่มีความจุเกตสูงๆ จำเป็นต้องใช้ค่าใช้จ่ายสูงตามไปด้วย วิทยานิพนธ์นี้จึงได้ทำการศึกษาและออกแบบวงจรกรองเชิงเลขชนิดรีเคอร์ซีฟโดยใช้คณิตศาสตร์แบบการกระจายด้วยเทคนิคบิตอนุกรมเวิร์ดขนาน เพื่อให้สอดคล้องกับข้อจำกัดทางด้านความจุและความเร็วของวงจรกรอง โดยใช้เอฟพีจีเอพื้นฐานที่มีราคาไม่สูงนัก

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

ในการดำเนินการค้นคว้าและวิจัยเพื่อให้ได้มาซึ่งวิทยานิพนธ์ฉบับนี้มีความมุ่งหมายที่จะออกแบบวงจรกรองสัญญาณเชิงเลขความถี่ต่ำผ่านด้วยคณิตศาสตร์แบบการกระจาย โดยรูปแบบการประมวลผลเป็นแบบ การกระจายแบบอนุกรม (Serial-Distributed Arithmetic) ลงในอุปกรณ์เอฟพีจีเอขนาดกลาง พร้อมทั้งผ่านการทดสอบวงจรกับสัญญาณจริง จึงได้กำหนดวัตถุประสงค์ในการจัดทำวิทยานิพนธ์เรียงตามลำดับขั้นตอนการดำเนินการไว้ดังนี้

1. ออกแบบและวิเคราะห์การจำลองผลการทำงาน ของวงจรกรองเชิงเลขความถี่ต่ำผ่านโดยคณิตศาสตร์แบบการกระจาย ในระดับฟังก์ชันถ่ายโอนด้วยโปรแกรม MATLAB (FDA)
2. ออกแบบวงจรกรองเชิงเลขความถี่ต่ำผ่านโดยคณิตศาสตร์แบบการกระจาย อันดับ 2 โดยใช้ตัวประมวลผลคณิตศาสตร์แบบการกระจาย ด้วยวิธีการที่เหมาะสมต่อการโปรแกรมลงในอุปกรณ์ เอฟพีจีเอ ที่มีความจุเกตทั้งหมดไม่เกิน 50,000 เกต
3. วิเคราะห์และเปรียบเทียบผลการออกแบบวงจร ในด้านขนาดของวงจรและผลตอบสนองทางความถี่

1.3 สมมติฐานของการศึกษา

การดำเนินการวิจัยในช่วงแรกจะทำการใช้โปรแกรม Matlab (FDA) เพื่อใช้คำนวณค่าการตอบสนองต่างๆของตัวกรองที่ต้องการออกแบบ รวมทั้งค่าสัมประสิทธิ์ของวงจรกรองจากข้อกำหนดทางความถี่ที่ต้องการ และใช้ฟังก์ชันภายในโปรแกรมคำนวณหาค่าสัมประสิทธิ์แบบควอนไตซ์ความยาวบิตสูง จากนั้นทำการเขียนโปรแกรมในรูปแบบ M-files สำหรับทำการตัดบิต

ควอนไตซ์ให้เหลือขนาด 8 บิตแล้วแปลงค่าในรูปแบบส่วนเติมเต็มสอง แล้วใช้หลักการคณิตศาสตร์แบบการกระจาย ทำการสร้างค่าตารางเฉพาะของตัวกรองเพื่อใช้ในการออกแบบวงจรต่อไป ซึ่งชนิดของตัวกรองไม่ว่าจะเป็นตัวกรองบัทเทอร์เวิร์ธ (Butterworth filters) ตัวกรองเชบีเชฟ (Chebychev filters) ตัวกรองอิลลิปติค (Elliptic filters) หรือตัวกรองชนิดอื่น สามารถคำนวณค่าบน Matlab ได้อย่างสะดวก เมื่อได้รูปแบบตารางฟังก์ชันของสัมประสิทธิ์ต่างๆ (Look-up table: LUT) แล้วจึงนำมาใช้เป็นข้อมูลในการดำเนินการในขั้นตอนสุดท้าย คือ การจัดสร้างเป็นวงจรจริง ในขั้นตอนนี้จะเลือกใช้โครงสร้างการประมวลผลคณิตศาสตร์แบบการกระจาย ข้อมูลที่ใช้อยู่ในลักษณะแบบส่วนเติมเต็มสอง (Two's Complement Representation) จากอัลกอริธึมแบบต่างๆมาสู่ขั้นตอนการออกแบบวงจรจะใช้วิธีการออกแบบด้วยภาษาอธิบายฮาร์ดแวร์ (Hardware description language: HDL)ให้สอดคล้องกับอัลกอริธึมที่ได้วางไว้ โดยมีเป้าหมายเพื่อสังเคราะห์และโปรแกรมลงสู่อุปกรณ์ เอฟพีจีเอ ที่มีราคาและความจุไม่สูงนักของบริษัท Xilinx จากนั้นจึงทำการเปรียบเทียบผลการสังเคราะห์วงจร และนำอุปกรณ์ไปทดสอบกับสัญญาณจริงต่อไป

1.4 ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย

การดำเนินการวิจัย เริ่มต้นด้วยการคำนวณฟังก์ชันถ่ายโอนของวงจรกรองแบบบัทเทอร์เวิร์ธ โดยฟังก์ชันถ่ายโอนของวงจรกรองจะถูกส่งสู่โดเมนดิจิตอลโดนผ่านการแปลงไบลิเนียร์ (Bilinear Transformation) จากนั้นจัดรูปแบบวงจรกรองเชิงเลขโครงสร้างไบควอดราติก (Biquadratic Structure) ซึ่งเป็นตัวกรองป้อนกลับอันดับสอง แล้วใช้หลักการของคณิตศาสตร์แบบการกระจายทำการสร้างค่าตาราง LUT และในการลดผลกระทบของความยาวคำจำกัดสามารถทำได้โดยใช้การจัดสเปกตรัมความผิดพลาด (Error Spectrum Shaping: ESS) ได้ ในส่วนของฮาร์ดแวร์ได้ดำเนินการออกแบบภายใต้รูปแบบโครงสร้างคณิตศาสตร์แบบการกระจาย และแทนข้อมูลในรูปแบบส่วนเติมเต็ม 2 แล้วจึงทำการเชื่อมโยงมาสู่วงจรจริง โดยการใช้ภาษา VHDL ในการกำหนดพฤติกรรมของดิจิตอลฮาร์ดแวร์ โดยเน้นการลดขนาดและเพิ่มความเร็วของอุปกรณ์เป็นสำคัญ

1.5 ขอบเขตการวิจัย

วิทยานิพนธ์นี้เป็นการวิจัยเพื่อออกแบบวงจรกรองคณิตศาสตร์แบบการกระจายด้วยอุปกรณ์ เอฟพีจีเอ ดำเนินการ ณ ห้องปฏิบัติการ Mixed Signal Processing คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง เริ่มต้นดำเนินการในภาคเรียนที่ 1 ปีการศึกษา 2544 สิ้นสุดในภาคเรียนที่ 2 ปีการศึกษา 2546 รวมระยะเวลาดำเนินการ 36 เดือน

1.6 ขั้นตอนของการศึกษา

การดำเนินการวิจัยแบ่งออกเป็น 4 ขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 1 การวางแผนการจัดทำวิทยานิพนธ์และรวบรวมทฤษฎีที่เกี่ยวข้องกับงานวิจัย ซึ่งประกอบไปด้วย

บทที่ 1 บทนำ ประกอบไปด้วย ความเป็นมาและความสำคัญของปัญหา ความมุ่งหมายและวัตถุประสงค์ของการศึกษา สมมติฐานของการศึกษา ทฤษฎีหรือแนวความคิดที่ใช้ในการวิจัย ขอบเขตการวิจัย และขั้นตอนการศึกษา

บทที่ 2 สถาปัตยกรรมและการออกแบบบน FPGA เป็นการกล่าวถึงลักษณะโครงสร้างภายใน FPGA ที่ใช้ในงานวิจัยนี้ เปรียบเทียบคุณสมบัติเด่นที่ได้จากการใช้งาน ตลอดจนรูปแบบและขั้นตอนในการออกแบบและประยุกต์ใช้งานในระดับต่างๆตามความเหมาะสม

บทที่ 3 การประมวลผลสัญญาณ สำหรับบทนี้จะเป็นการกล่าวถึงพื้นฐานทางทฤษฎีการประมวลผลสัญญาณ โครงสร้างของตัวประมวลผลสัญญาณเชิงเลข ตัวแปรที่ใช้ทำการแปลงระบบอนาลอกไปเป็นระบบเชิงเลข เงื่อนไขสำคัญต่างๆ ที่ใช้ในการพิจารณาคุณสมบัติของระบบเชิงเลข และหลักการที่ใช้สร้างวงจรกรองเชิงเลขสำหรับงานวิจัยนี้

บทที่ 4 ทฤษฎีของตัวดำเนินการทางคณิตศาสตร์ จะเป็นการนำเสนอรูปแบบวิธีการทางคณิตศาสตร์ที่สามารถนำมาประยุกต์ใช้งานในการออกแบบหน่วยคำนวณฮาร์ดแวร์ โดยได้ทำการนำเสนอรูปแบบวิธีการและที่มา โดยเฉพาะหลักการคณิตศาสตร์แบบการกระจายซึ่งถูกใช้ในงานวิจัยนี้อย่างสัมพันธ์กัน รวมทั้งรูปแบบหลักการชนิดอื่นๆที่อาจนำมาประยุกต์ใช้ในการพัฒนาต่อไป

บทที่ 5 ผลกระทบของความยาวคำจำกัดในวงจรกรองเชิงเลข เนื้อหาส่วนนี้จะเป็นการกล่าวถึงปัญหาที่เกิดขึ้นในระบบประมวลผลเชิงเลข ซึ่งได้นิยามถึงรูปแบบจำนวนชนิดต่างๆที่มักปรากฏในระบบเชิงเลข หลักการต่างๆที่ได้นำเสนอล้วนมีส่วนในการสร้างผลกระทบต่อความผิดพลาดที่อาจเกิดขึ้นได้ในระบบเชิงเลข จึงได้เสนอวิธีการแก้ไขความผิดพลาดจากผลกระทบนี้ทำการประยุกต์ใช้กับระบบที่ได้ออกแบบสำหรับงานวิจัยนี้

ขั้นตอนที่ 2 ทำการออกแบบวงจรกรองดิจิตอลความถี่ต่ำผ่านคณิตศาสตร์แบบการกระจาย และจำลองผลการทำงานในด้านผลตอบสนองทางความถี่และผลทางด้านขนาดของวงจร โดยใช้โปรแกรม MATLAB, XilinxISE และ ModelSIM ซึ่งได้ทำการสรุปเนื้อหาไว้ใน

บทที่ 6 การออกแบบวงจรกรองเชิงเลขความถี่ต่ำโดยใช้วิธีคณิตศาสตร์แบบการกระจาย บทนี้กล่าวถึง ขั้นตอนในการออกแบบวงจรกรองที่นำเสนอโดยละเอียด ตั้งแต่การจำลองลักษณะของวงจรกรองที่ต้องการด้วยโปรแกรม MATLAB การคำนวณค่าสัมประสิทธิ์ในรูปแบบควอนไตซ์ การปัดเศษ การตัดปลาย การคำนวณหาค่าตาราง LUT การแก้ไขความผิดพลาดที่เกิดโดยใช้หลักการจัดสเปกตรัมความผิดพลาด ทำการคำนวณหาค่า LUT ที่แก้ไขความผิดพลาด จึงทำการออกแบบแต่ละส่วนประกอบด้วยโปรแกรมภาษา VHDL ซึ่งตรวจสอบผลการสังเคราะห์และจำลองการทำงาน ได้จากโปรแกรม XilinxISE และ ModelSIM ตามลำดับ

ขั้นตอนที่ 3 ทำการทดสอบวงจรกรองที่ได้ออกแบบกับสัญญาณจริง ซึ่งได้สรุปเนื้อหาไว้ในบทที่ 7

บทที่ 7 ผลการทดสอบวงจร เป็นการรวบรวมผลการทดลองเพื่อหาผลตอบสนองทางความถี่โดยการผ่านสัญญาณจริงเข้าสู่วงจรกรองที่ได้ออกแบบไว้เพื่อตรวจสอบผลที่ได้ว่าสอดคล้องกับข้อกำหนดต่างๆ ที่ได้วางไว้อย่างไร

ขั้นตอนที่ 4 การสรุปผลการดำเนินงานทั้งหมด เนื้อหาประกอบไปด้วย

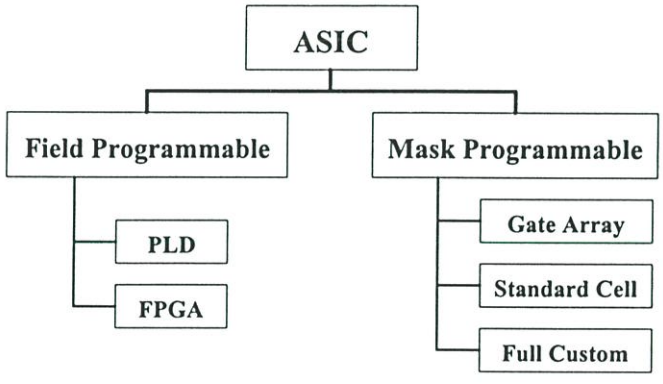
บทที่ 8 สรุปผลการวิจัยและข้อเสนอแนะ เป็นการสรุปผลการดำเนินการวิจัยทั้งหมด รวมถึงได้เสนอสิ่งที่ควรปรับปรุงในงานวิจัยนี้ เพื่อเป็นแนวทางพัฒนางานวิจัยต่อไปในอนาคต

บทที่ 2

สถาปัตยกรรมและการออกแบบบน FPGA

2.1 บทนำ

ปัจจุบันอุตสาหกรรมทางด้านอิเล็กทรอนิกส์มีความรุดหน้าไปอย่างรวดเร็วมาก และอีกทั้งประสิทธิภาพที่เพิ่มขึ้นของอุปกรณ์อิเล็กทรอนิกส์ เพื่อให้รองรับกับความต้องการของการนำมาใช้กับงานที่มีความซับซ้อนมากขึ้นเรื่อยๆนั้น สิ่งสำคัญที่สุดย่อมขึ้นอยู่กับประสิทธิภาพของชิ้นส่วนแต่ละชิ้นที่อยู่ภายในอุปกรณ์อิเล็กทรอนิกส์โดยจะเป็นส่วนประกอบที่สำคัญอย่างมากซึ่งก็คือ วงจรรวม หรือที่เรารู้จักกันดีในนามว่า ไอซี (IC, integrated circuit) หรือ ไมโครชิป สำหรับตัวอย่างของการพัฒนาเทคโนโลยีไอซีที่พบเห็นได้ชัดเจนนั้น เช่น เทคโนโลยีไมโครโพรเซสเซอร์ (MPU), ไมโครคอนโทรลเลอร์ (MCU) และหน่วยความจำ (memory) ซึ่งในท้องตลาดปัจจุบันนี้มีไอซีหลายชนิดให้เลือกนำมาใช้งานกัน ตามความเหมาะสมทั้งด้านราคาและประสิทธิภาพ โดยเฉพาะอย่างยิ่งไอซีที่มีประสิทธิภาพสูงๆ ที่ใช้ในงานเฉพาะด้านหรือที่เรียกว่า ASIC (Application Specific Integrated Circuit, ASIC) ซึ่งจะสามารถจัดแบ่งตามโครงสร้างออกได้เป็น 2 กลุ่มใหญ่ๆ คือ Field programmable และ Mask programmable ดังแสดงในรูปที่ 2.1



รูปที่ 2.1 การแบ่งกลุ่มของวงจรรวม ASIC

ถ้าจะพิจารณาถึงข้อดี ในเชิงการออกแบบและการผลิต ไอซีตระกูล mask programmable นั้นอาจจะทำให้ได้ชิป ASIC ที่มีประสิทธิภาพสูงกว่า แต่ต้องมีการส่งวงจรไปทำหน้ากาก (mask set) และเจือสารยังต่างประเภท จึงทำให้มีค่าใช้จ่ายเริ่มต้น (fixed cost) ที่สูงมาก ซึ่งเหมาะแก่การใช้งานที่ต้องการปริมาณ (volume) สูงเพื่อเฉลี่ยค่าใช้จ่ายเริ่มต้นเท่านั้น สำหรับการพัฒนาชิป ASIC ในปริมาณน้อยถึงปานกลางนั้น ควรใช้ชิป ASIC ชนิดโปรแกรมได้ (field programmable) ชนิดใดชนิดหนึ่งซึ่งสามารถหาซื้อได้ในราคาย่อมเยากว่า โดยผู้ใช้งานสามารถออกแบบและสร้างวงจรที่

ต้องการใช้ลงในตัวอุปกรณ์ได้เอง โดยไม่ต้องไปผลิตที่โรงงาน จึงเป็นทางเลือกที่น่าสนใจ โดยเฉพาะอย่างยิ่ง เทคโนโลยีที่ก้าวหน้าไปมาก ทำให้ประสิทธิภาพของไอซีแบบโปรแกรมได้ สามารถตอบสนองต่อความต้องการของการใช้งานที่มีความซับซ้อนสูงได้

2.2 เอฟพีจีเอ (FPGA: Field Programmable Gate Array)

FPGA เป็นอุปกรณ์ที่ถูกพัฒนาต่อจากอุปกรณ์แอลซีเอ ของบริษัทไซลิงซ์ (XILINX Inc.) โดยมีจุดเด่นที่ประสิทธิภาพการทำงานและปริมาณความหนาแน่นของจำนวนเกตสูง การใช้งานที่สะดวก สามารถกำหนดฟังก์ชั่นการทำงานได้ตามความต้องการของผู้ใช้งาน โดยผ่าน software เพื่อจะทำการโปรแกรมข้อมูลลงในชิป FPGA ให้ทำงานตามที่ออกแบบไว้โดยอัตโนมัติ ดังนั้นการออกแบบวงจรโดยใช้เอฟพีจีเอ สามารถออกแบบและทดสอบได้ภายในเวลาไม่นานเพียง 2-3 วัน เท่านั้น ซึ่งตรงกันข้ามกับการออกแบบโดยใช้เกตอาเรย์ ซึ่งจะใช้เวลานานหลายสัปดาห์ การเปลี่ยนแปลงแก้ไขต้นแบบก็เป็นเช่นเดียวกัน จากประโยชน์ของเอฟพีจีเอ ดังที่กล่าวมานั้น ทำให้เกิดการประหยัดค่าใช้จ่ายเป็นอย่างมาก เพราะได้ลดความเสี่ยงในการที่จะต้องแก้ไขตัววงจรต้นแบบ การเลื่อนเวลาการออกผลิตภัณฑ์ และลดค่าเอ็นอาร์อี (NRE: Nonrecurring Engineering Cost) ลงไปด้วย

2.2.1 เปรียบเทียบกับไมโครคอนโทรลเลอร์ (Microcontroller)

ไมโครคอนโทรลเลอร์ เป็นชิปประเภทใช้ประโยชน์ทั่วไป (general purpose devices) หมายความว่าสถาปัตยกรรม รวมถึงชุดคำสั่งทั้งหมดของชิปนั้นๆ ได้ถูกออกแบบและกำหนดมาเรียบร้อยแล้ว ผู้ใช้มีหน้าที่เพียงเรียบเรียงชุดคำสั่ง เพื่อให้ชิปปฏิบัติหน้าที่ตามที่ต้องการ ข้อจำกัดทางด้านสถาปัตยกรรมและชุดคำสั่งทำให้ ไมโครคอนโทรลเลอร์ไม่ใช่คำตอบของงานหลายงาน ตัวอย่างเช่น ไมโครคอนโทรลเลอร์ที่ทำงานที่ความเร็ว 20MHz ย่อมไม่สามารถใช้กับงานที่ต้องการความเร็วมากกว่านั้น ซึ่งอาจเป็นคุณสมบัติจำเป็นสำหรับงานในหลายๆด้าน เช่นโทรคมนาคม หรืองานด้านการประมวลภาพ (Image processing) ทั้งหมดนี้ต่างกับการใช้ FPGA ที่ผู้ใช้เป็นคนออกแบบสถาปัตยกรรมทั้งหมดเอง โดยที่พื้นฐานเทคโนโลยี FPGA ในปัจจุบันสามารถทำงานที่สัญญาณนาฬิกามากกว่า 100 MHz เพราะฉะนั้นประสิทธิภาพจริงๆของชิป จะขึ้นกับสถาปัตยกรรมที่ผู้ออกแบบคิดค้นและใช้เป็นหลัก นอกจากนั้นผู้ใช้ยังสามารถออกแบบชุดคำสั่งใหม่ๆขึ้นเองเพื่อให้เหมาะกับการใช้งานนั้นๆได้ รวมถึงความสามารถในการรวบรวมวงจรดิจิตอลอื่นๆ นอกชิปเข้ามาอยู่รวมกันใน FPGA เดียวได้ FPGA จึงเป็นมากกว่าทางเลือกหนึ่งของ ไมโครคอนโทรลเลอร์ แต่เป็นคำตอบที่ดีกว่าในงานออกแบบดิจิตอล

2.2.2 จุดเด่นของ FPGA

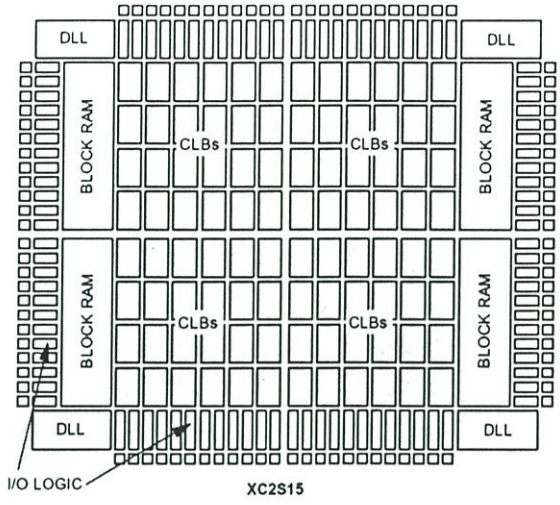
นอกจากระยะเวลาที่ใช้ในการออกแบบและพัฒนาจะเร็วใกล้เคียงกับการใช้ ชิปประเภทใช้ประโยชน์ทั่วไป ขณะที่ประสิทธิภาพใกล้เคียงกับชิป ASIC แบบ full-custom ซึ่งหมายถึง ชิปที่ถูกออกแบบและพัฒนาใหม่ทั้งหมดเฉพาะงานนั้นๆ ซึ่งมีราคาแพงกว่ามากแล้ว ผู้ใช้ชิป FPGA ยังสามารถนำวงจรที่พร้อมใช้งานที่มีผู้ออกแบบไว้แล้ว ในรูปของทรัพย์สินทางปัญญาหรือ (IP-cores: intellectual property cores) มาใช้รวมกับวงจรที่เราออกแบบเองได้ทันที ซึ่งเป็นการเพิ่มประสิทธิภาพ และย่นเวลาการพัฒนาชิป FPGA ของเราได้มาก

2.2.3 FPGA ในท้องตลาดและการเลือกใช้

ปัจจุบันมีผู้ผลิตชิป FPGA อยู่หลายบริษัท เช่น Xilinx Corp., Altera หรือ Actel โดยบริษัทที่มีขนาดใหญ่ที่สุด และมีส่วนแบ่งทางการตลาดสูงที่สุดคือ Xilinx (www.xilinx.com) ปกติ FPGA ของแต่ละบริษัท จะมีหลายตระกูลขึ้นอยู่กับราคาและขนาดความจุของชิบ FPGA ซึ่งมักถูกวัดด้วยค่าจำนวนเกทเทียบเท่า (equivalent gate) โดยปัจจุบัน FPGA จะมีขนาดตั้งแต่ น้อยกว่า 10,000 เกท จนถึงมากกว่า 5 ล้านเกทขึ้นอยู่กับตระกูล เช่น Xilinx ตระกูล 9,500 ซึ่งใช้เทคโนโลยี EEPROM มีขนาดความจุสูงสุดถึงประมาณ 7,000 เกท ขณะที่ Xilinx ตระกูล Spatan2 มีขนาดความจุสูงสุดถึงประมาณ 2 แสนเกท และสำหรับปัจจุบัน Xilinx ตระกูล Virtex ซึ่งใช้เทคโนโลยี SRAM จะมีราคาแพงกว่า แต่มีขนาดความจุเพิ่มขึ้นถึงกว่า 3 ล้านเกท เพื่อให้เห็นภาพเปรียบเทียบไมโครคอนโทรล-เลอร์ ตระกูล 8051 มีขนาดเกทประมาณ 15,000 เกท ขนาดความจุของเกท ที่มากกว่าหมายถึงศักยภาพในการใช้งาน FPGA ตัวนั้นสำหรับงานที่มีความซับซ้อนมากกว่า

2.3 รายละเอียดสถาปัตยกรรมของ FPGA ตระกูล Spartan II

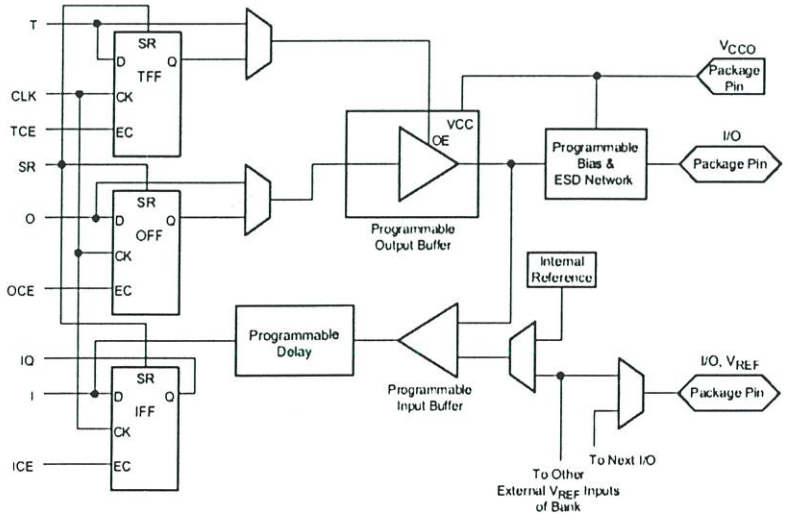
รูปที่ 2.2 แสดงบล็อกไดอะแกรมของโครงสร้างชิป FPGA ตระกูล Spartan II ประกอบด้วย Configurable Logic Block (CLBs) ซึ่งเป็นบล็อกลอจิกวางโครงแบบได้ ที่มีลักษณะสม่ำเสมอยืดหยุ่นได้และโปรแกรมได้ บล็อก CLB ทั้งหลายจะถูกล้อมรอบด้วย I/O Blocks (IOBs) ที่โปรแกรมได้ บล็อกต่างๆเหล่านี้เชื่อมต่อระหว่างกันด้วยแหล่งการเชื่อมโยง (Routing Resources) อันทรงประสิทธิภาพ สถาปัตยกรรมนี้มีความสามารถรองรับฟังก์ชันขั้นสูง อย่างบล็อก RAM และบล็อกสัญญาณนาฬิกาควบคุม (clock control blocks)



รูปที่ 2.2 บล็อกไดอะแกรมของโครงสร้าง FPGA ตระกูล Spartan II

2.3.1 บล็อกอินพุท/เอาท์พุท (IOB)

รูปที่ 2.3 แสดงบล็อกอินพุท/เอาท์พุต (IOB) ของโครงสร้าง FPGA ตระกูล Spartan II ซึ่งรองรับมาตรฐานการให้สัญญาณ I/O ความเร็วสูงได้ถึง 16 มาตรฐาน เช่น LVCMOS, HTSL, SSTL, และ GTL I/O ความเร็วสูงนี้สามารถรองรับหน่วยความจำ และการเชื่อต่อของบัส (bus interface) ที่ทันสมัยได้หลากหลายแบบ รีจีสเตอร์ IOB ทั้งสามตัว ทำหน้าที่เป็น edge-triggered D flip-flop (ฟลิปฟล็อปแบบประวิงชนิดที่ใช้ขอบสัญญาณนาฬิกาจุดชนวนการทำงาน) หรือไม่ก็เป็น level-sensitive latch (แล็ตช์แบบไวต่อระดับสัญญาณนาฬิกา)

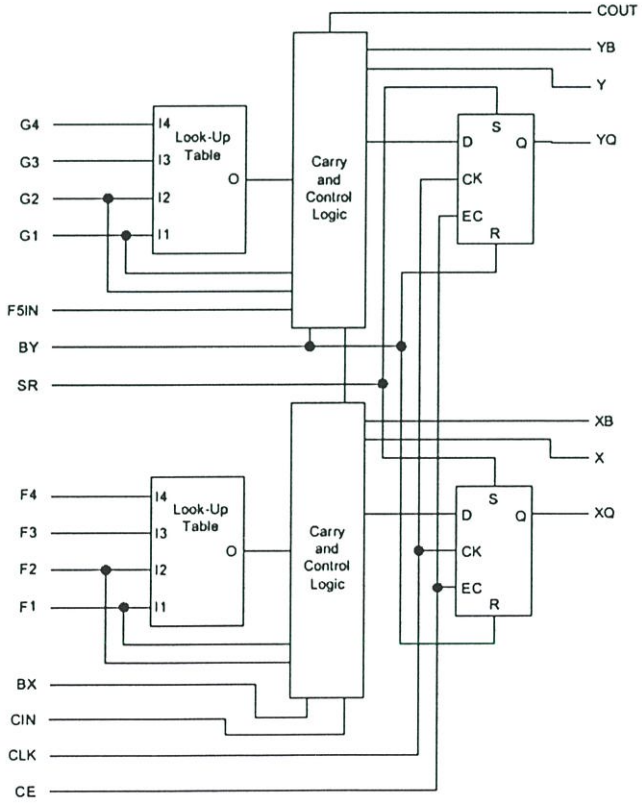


รูปที่ 2.3 บล็อกอินพุท/เอาท์พุท (IOB) ของโครงสร้าง FPGA ตระกูล Spartan II

2.3.2 ลอจิกเซลล์ (Logic Cells)

ลอจิกเซลล์ (LC) เป็นส่วนประกอบพื้นฐานสำหรับ CLB ใน Spartan II ดังแสดงในรูปที่ 2.4 ในเซลล์ลอจิกเซลล์หนึ่งมีตัวกำเนิดฟังก์ชันแบบ 4 input จำนวน 1 ตัว พร้อมลอจิกทด (carry logic) และส่วนสำรองข้อมูล (storage element) เอาท์พุทจากตัวกำเนิดฟังก์ชันในแต่ละ LC เป็นสัญญาณขับทั้ง CLB output และ D input ของฟลิปฟล็อป ใน CLB แต่ละบล็อกของ Spartan II จะประกอบด้วย LC จำนวน 4 เซลล์ จัดเรียงใน slice 2 slice นอกจาก LC พื้นฐานทั้งสี่แล้ว Spartan II CLB ยังมีลอจิกที่รวมเอาตัวกำเนิดฟังก์ชันที่ให้ฟังก์ชัน 5-6 input ไว้ด้วย ยังผลให้ CLB มีจำนวน LC ที่นับได้จริงเท่ากับ 4.5 เซลล์

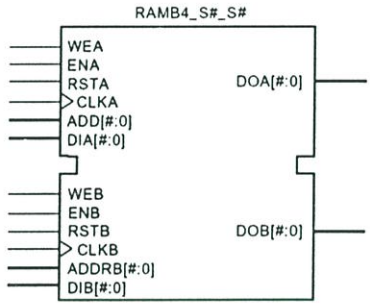
ตัวกำเนิดฟังก์ชันของ Spartan II มีลักษณะเป็นตารางค้นหา (Look-Up Table หรือ LUTs) แบบ 4 input นอกจากเป็นตัวกำเนิดฟังก์ชันแล้ว LUT เหล่านี้สามารถทำหน้าที่เป็น 16 x 1-bit synchronous RAM หรือเป็น 16-bit shift register ที่เหมาะสำหรับบันทึกข้อมูลความเร็วสูงหรือข้อมูลที่ถูกส่งเป็นชุดอย่างรวดเร็ว (burst-mode data) ซึ่งพบบ่อยในงานจำพวก Digital Signal Processing (DSP) ส่วนสำรองข้อมูลใน Spartan II slice อาจวางให้เป็นแบบ edge-triggered D flip-flops หรือเป็น level-sensitive latches ก็ได้



รูปที่ 2.4 บล็อก CLB Slice ของ FPGA ตระกูล Spartan II (สมมาตรในแต่ละ CLB)

2.3.3 บล็อกแรม (Block RAM)

บล็อกแรมพอร์ตคู่ของ Spartan II FPGA แสดงดังรูปที่ 2.5 จะรวมเอาบล็อก RAM ที่เลือกกับหน่วยความจำขนาด 4Kbits สิ่งเหล่านี้ช่วยเสริมเติมเต็มการกระจาย RAM ที่ถูกเลือก กับ แหล่งที่ให้ทำให้โครงสร้าง RAM บางลง ดังในรูป CLB แต่ละบล็อกสามารถวางคอนฟิกูเรชันที่อัตราส่วนระหว่าง 4Kx1 และ 256x16



รูปที่ 2.5 บล็อกแรมพอร์ตคู่

2.3.4 ดีเลย์ล็อคลูป (Delay-Locked Loop)

ดีเลย์ล็อคลูป (DLL) จะทำงานร่วมกับทุกสัญญาณนาฬิกาครอบคลุมถึงบัฟเฟอร์ (buffer) ในการกำจัดการหมุนวน (skew) ระหว่างขาสัญญาณนาฬิกาเข้า (clock input pad) และขาสัญญาณนาฬิกาเข้าภายใน (internal clock input pins) ทั้งอุปกรณ์ DLL แต่ละตัวสามารถขับสัญญาณนาฬิกาได้คลอบคลุมทั้งเครือข่าย (global clock networks) ได้ถึง 2 เครือข่าย DLL จะเฝ้าสังเกตสัญญาณนาฬิกาเข้า และสัญญาณนาฬิกาที่ถูกกกระจาย แล้วทำการปรับอุปกรณ์หน่วงสัญญาณนาฬิกา (clock delay element) โดยอัตโนมัติ เมื่อ สัญญาณนาฬิกาเดินทางถึงอินพุทแล้ว DLL จะทำให้เกิดการหน่วงเพิ่มเติมเพื่อให้ขอบสัญญาณนาฬิกา (clock edge) เดินทางถึงฟลิปฟล็อปภายในวงจร ในเวลา 1 คาบ ทำให้มั่นใจว่าขอบสัญญาณภายในฟลิปฟล็อป สามารถประสานเวลากับขอบสัญญาณที่อินพุทได้เป็นอย่างดี

2.3.5 คุณสมบัติใหม่ที่เพิ่ม ใน Spartan-II จาก Spartan Series

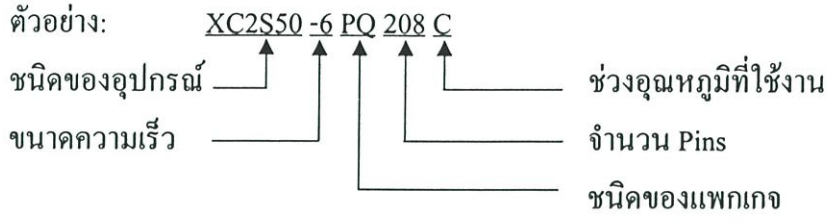
ตารางที่ 2.1 ตารางเปรียบเทียบคุณสมบัติ Spartan-II และ Spartan Series

	Spartan-II	**Spartan-XL**	**Spartan**
ความหนาแน่น (เกท)	15K – 200K	5K – 40K	5K – 40K
สมรรถนะของ I/O	200 MHz	100 MHz	80 MHz
สถาปัตยกรรม	Virtex derivative	XC4000 derivative	XC4000 derivative
Block RAM	มี	ไม่มี	ไม่มี
Distributed RAM	มี	มี	มี
DLL	มี	ไม่มี	ไม่มี
มาตรฐาน I/O	16	4	4
Core Voltage	2.5 V	3.3 V	5 V
5 V Tolerance	มี	มี	มี
Process (micron)	0.18/0.22	0.25/0.35	0.35/0.5
Configuration mode	Serial, Parallel, JTAG	Serial, Express, JTAG	Serial, JTAG
Packages	VQ100, TQ/CS144, PQ208, FG256/456	PC84, VQ100, TQ/CS144, PQ208/240, BG256, CS280	PC84, VQ100,TQ144, PQ208/240, BG256

ตารางที่2.2 ตารางเปรียบเทียบคุณสมบัติ FPGA ตระกูล Spartan II series

Feature	**XC2S15**	**XC2S30**	**XC2S50**	**XC2s100**	**XC2S150**	**XC2S200**
System Gates	15K	30K	50K	100K	150K	200K
Logic Cells	432	972	1728	2700	3888	5292
CLBs	96	216	384	600	864	1176
Block RAM (bits)	16K	24K	32K	40K	48K	56K
MAX I/O	86	132	176	196	260	284
Packages	VQ100 TQ144 CS144	VQ100 TQ144 CS144 PQ208	TQ144 PQ208 FG256	TQ144 PQ208 FG256 FG456	PQ208 FG256 FG456	PQ208 FG256 FG456

2.3.6 ข้อมูลเกี่ยวกับหมายเลข FPGA



ชนิดและหมายเลข FPGA ตระกูล Spartan II series

ตารางที่2.3 ตารางแสดงรหัสหมายเลข FPGA ตระกูล Spartan II series

อุปกรณ์
XC2S15
XC2S30
XC2S50
XC2S100
XC2S150
XC2S200

ขนาดความเร็ว	
-5	มาตรฐาน
-6	สูง

จำนวน pin / ชนิดแพกเกจ	
VQ100	100-pin Plastic Very Thin QFP
CS144	144-ball Chip-Scale BGA
TQ144	144-pin Plastic Thin QFP
PQ208	208-pin Plastic QFP
FG256	256-ball Fine Pitch BGA
FG456	456-ball Fine Pitch BGA

ช่วงอุณหภูมิใช้งาน (Tj)	
C = commercial	$0^{\circ}C$ ถึง $85^{\circ}C$
I = Industrial	$-40^{\circ}C$ ถึง $100^{\circ}C$

2.4 การออกแบบวงจรดิจิตอลในปัจจุบัน

รูปแบบการออกแบบวงจรดิจิตอลสามารถแบ่งออกได้เป็นหลายรูปแบบดังแสดงต่อไปนี้

2.4.1 การออกแบบโดยใช้การวาดผังวงจร (Schematic design)

ในอดีตการออกแบบวงจรดิจิตอลที่เรารู้จักกัน จะเป็นการออกแบบวงจรในระดับลอจิกเกต (Logic level) โดยอาศัยโปรแกรมช่วยในการวาดผังวงจร ซึ่งการออกแบบวงจรลักษณะนี้ ค่อนข้างใช้เวลามากในการออกแบบ เนื่องจากการออกแบบทุกๆอย่างผู้ออกแบบจะต้องออกแบบเองทั้งหมด ซึ่งถ้าวงจรที่ต้องการออกแบบมีความซับซ้อนสูง ก็จะทำให้การออกแบบทำได้ยาก ซึ่งเป็นข้อจำกัดให้ผู้ออกแบบไม่สามารถออกแบบวงจรดิจิตอลที่มีความซับซ้อนสูงได้ เช่น การออกแบบซีพียู และไมโครคอนโทรลเลอร์ เป็นต้น

2.4.2 การออกแบบโดยใช้ภาษาระดับสูง (HDL)

ในปัจจุบันเทคโนโลยีการออกแบบวงจรดิจิตอล ได้พัฒนาสูงขึ้นและได้มีการพัฒนาภาษาที่ใช้สำหรับออกแบบวงจรดิจิตอล พร้อมกระบวนการออกแบบแนวใหม่ ที่จะทำให้ผู้ออกแบบไม่ถูก

จำกัดในการออกแบบวงจรอีกต่อไป ซึ่งการออกแบบวงจรดิจิตอลแนวใหม่นี้ จะใช้การเขียนภาษาบรรยายพฤติกรรมฮาร์ดแวร์ (HDL: Hardware Description Language) มาใช้ในการออกแบบวงจร โดยภาษาจะบรรยายพฤติกรรมฮาร์ดแวร์ที่เป็นมาตรฐานซึ่งมีอยู่ 2 ภาษาที่นิยมใช้ด้วยกันคือ ภาษา Verilog และ VHDL โดยถ้านำมาเปรียบเทียบกับในโลกของการออกแบบซอฟต์แวร์ ก็เปรียบได้กับภาษา C และภาษา Pascal นั่นเอง โดยที่ภาษา Verilog จะมีโครงสร้างคล้ายกับภาษา C ส่วนภาษา VHDL จะมีโครงสร้างคล้ายภาษา Pascal

การออกแบบวงจรโดยใช้ภาษา HDL จะเป็นกระบวนการออกแบบที่เราเรียกกันว่า Top-Down design โดยกระบวนการดังกล่าว จะอาศัยซอฟต์แวร์ช่วยในการออกแบบ ซึ่งเราเรียกซอฟต์แวร์จำพวกนี้ว่า EDA: Electronics Design Automation ที่จะทำหน้าที่ช่วยในการจำลองการทำงานจากโค้ดที่เขียนขึ้น (HDL Simulation) และช่วยในการนำโค้ดที่ผ่านการจำลองการทำงาน ไปสังเคราะห์ (Synthesis) ให้เป็นผังวงจรระดับลอจิกเกต (Logic level) ในรูปแบบของเทคโนโลยี ที่ผู้ออกแบบต้องการนำไปใช้งาน ซึ่งในปัจจุบัน มีอุปกรณ์หรือชิปที่สามารถรองรับการออกแบบวงจรแบบใช้ภาษา HDL ในการออกแบบ เช่น ชิป CPLD (Complex Programmable Logic Devices) หรือชิป FPGA (Field Programmable Gate Arrays) ที่มีความจุของเกตภายในสูง สามารถออกแบบวงจรที่มีความซับซ้อนสูงได้มากกว่า 1 ล้านเกต หรือจะนำไปออกแบบในระดับของเอสิก (ASIC: Application Specific Integrated Circuits)

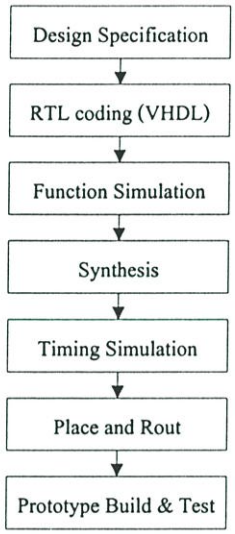
จะเห็นได้ว่าการออกแบบโดยใช้ภาษา HDL จะทำให้ผู้ออกแบบสามารถออกแบบวงจรที่ซับซ้อนได้อย่างสะดวกและรวดเร็ว และที่สำคัญการออกแบบจะไม่ยึดติดกับเทคโนโลยีใดๆ สามารถนำกลับมาใช้ใหม่ หรือกล่าวได้ว่า ถ้าผู้ออกแบบต้องการเปลี่ยนเทคโนโลยีของวงจรที่ออกแบบ ก็เพียงสังเคราะห์วงจรใหม่ ก็จะได้วงจรในเทคโนโลยีใหม่ทันที โดยไม่ต้องเสียเวลาในการวาดวงจรใหม่ เหมือนกับการออกแบบในลักษณะเดิมอีกต่อไป

2.5 การเขียนภาษา HDL

ดังที่ได้กล่าวไปแล้ว ในปัจจุบันแนวโน้มการออกแบบวงจรดิจิตอลขนาดใหญ่ จะหันไปสู่การใช้ภาษาอธิบายฮาร์ดแวร์ ซึ่งมีบทบาทมากขึ้นตามลำดับ ด้วยเหตุผลหลายประการ อาทิเช่น

- ความสามารถในการอธิบายการทำงานของวงจร ที่มีขนาดใหญ่มาก
- ความสามารถในการนำกลับไปใช้งานได้อีกของการออกแบบที่ถูกสร้างขึ้น
- ความสามารถในการนำไปใช้งานได้กับหลายๆเทคโนโลยี

โดยโค้ดที่ถูกเขียนจะไม่ขึ้นอยู่กับเทคโนโลยีเหล่านั้น หรือเป็นเพียงส่วนน้อยเท่านั้น นอกจากนั้นยังมีซอฟต์แวร์ที่มีประสิทธิภาพอีกมากมายสำหรับใช้ในการสังเคราะห์วงจร ในปัจจุบันภาษาที่ใช้อธิบายฮาร์ดแวร์จะเป็นที่นิยมใช้ และเป็นมาตรฐานในทางอุตสาหกรรมด้วยคือ ภาษา VHDL และ Verilog HDL โดยที่ภาษาทั้งสองก็มีจุดเด่นจุดด้อย และประสิทธิภาพในการใช้งานรวมถึงซอฟแวร์ที่ใช้สนับสนุนที่ไม่แตกต่างกัน โดยจะขอแนะนำวิธีการเขียนภาษา HDL ทั้ง VHDL และ Verilog แต่เพียงคร่าวๆ เพื่อเป็นพื้นฐานในการออกแบบ



รูปที่ 2.6 ขั้นตอนการออกแบบวงจรด้วยภาษา HDL

2.5.1 ลักษณะเฉพาะในการออกแบบ (Design specifications)

รูปที่ 2.6 เป็นการแสดงการออกแบบวงจรโดยใช้ภาษา HDL จะเริ่มต้นที่การเขียนโค้ดหลังจากที่ได้มีการออกแบบสถาปัตยกรรมของวงจรอย่างดีแล้ว เช่น อาจจะเริ่มจากการแบ่งวงจรรวมออกเป็นวงจรย่อย เพื่อให้ง่ายต่อการออกแบบตามหลักการ Top-down Design Methodology ถ้าวงจรหรือบล็อกย่อยไม่ซับซ้อนจนเกินไป และง่ายต่อการสังเคราะห์วงจรในภายหลัง ก็สามารถเขียนอธิบายพฤติกรรมการทำงานของวงจรเหล่านั้น (Behavioral Description) เป็นโมเดลในภาษา HDL และเมื่อวงจรย่อยเหล่านั้นเสร็จแล้ว จึงนำมาประกอบเข้าด้วยกัน โดยการเรียกใช้ส่วนประกอบ และมีสายสัญญาณเชื่อมต่ออย่างเหมาะสม ดังนั้นการอธิบายวงจรในระดับนี้ จึงเป็นการอธิบายในเชิงโครงสร้าง (Structural Description) การเขียนโค้ดภาษา HDL เช่นเดียวกับภาษาสูงอื่นๆ สามารถใช้โปรแกรมจำพวก Text Editor ทั่วๆไปได้ แต่อย่างไรก็ตาม ยังมีซอฟต์แวร์หลายตัวที่สนับสนุนการเขียนโค้ดภาษา HDL ซึ่งทำให้ง่ายขึ้นเวลาเขียนโค้ดในภาษา HDL รวมทั้งการคอมไพล์โค้ดเพื่อตรวจเช็คความถูกต้องตามหลักไวยากรณ์ด้วย ตัวอย่างของคอมไพล์เลอร์ภาษา HDL สำหรับเครื่องพีซี ก็เช่น ซอฟต์แวร์ชื่อ ModelSim ของ Model Technology, Inc. เป็นต้น

2.5.2 การจำลองการทำงานฟังก์ชัน (Functional simulation)

ขั้นตอนอีกขั้นตอนหนึ่งที่สำคัญในการออกแบบวงจร คือ การตรวจสอบว่าแบบจำลอง (Model)ในภาษา HDL ที่เราได้สร้างขึ้นนั้นสามารถทำหน้าที่ตามที่ได้กำหนดไว้หรือไม่ ซึ่งสามารถทำได้โดย การทำการจำลองการทำงาน เพื่อดูพฤติกรรมการทำงานของต้นแบบเหล่านั้น (Behavioral Simulation) และด้วยจุดประสงค์นี้เอง จะต้องมีการเขียนต้นแบบพิเศษสำหรับการทำการจำลองการทำงานขึ้นอีก ที่เรียกว่า HDL Testbench ซึ่งภายในจะมีการเรียกใช้หน่วยย่อยหรือส่วนประกอบที่ต้องการจะตรวจสอบพฤติกรรม การทำงาน และมีการกำหนดรูปแบบของสัญญาณขาเข้า (Input Signal) อันเป็นการกระตุ้น (Stimulus) และเมื่อทำการจำลองการทำงาน สัญญาณขาออก (Output Signal) ของหน่วยย่อยนั้นก็จะปรากฏให้เห็นบนจอภาพ และสามารถนำไปเปรียบเทียบกับสัญญาณขาเข้าเพื่อตรวจดูว่า การทำงานของหน่วยย่อย หรือวงจรที่ออกแบบแล้วถูกตรวจสอบ (Design Under Test หรือ DUT) เป็นไปตามที่คาดหวังไว้หรือไม่

2.5.3 การสังเคราะห์วงจร (Synthesis)

เมื่อได้ทำการตรวจสอบเป็นที่แน่ใจแล้วว่า ต้นแบบต่างๆที่ได้สร้างขึ้นสามารถทำงานได้อย่างถูกต้อง ขั้นตอนต่อไปก็คือ การผ่านโค้ดของต้นแบบเหล่านั้นไปทำการสังเคราะห์วงจร เพื่อให้ได้เนตลิสต์ในระดับเกตออกมา เครื่องมือที่ใช้ในการสังเคราะห์โค้ดภาษา HDL (HDL Synthesis Tool) ตัวอย่างเช่น ซอฟต์แวร์ของบริษัทเมนเตอร์กราฟฟิกส์ (Mentor Graphics) ที่มีชื่อว่า "ลีโอนาโด สเปกตรัม" (Leonardo Spectrum) ซึ่งสามารถใช้ในการสังเคราะห์วงจรให้เป็นเอสิก หรือเอฟพีจีเอก็ได้เพราะมีไลบารี่ ให้เลือกใช้ได้จากหลายบริษัทผู้ผลิต เนื่องจากเราจะสังเคราะห์วงจรสำหรับใช้งานกับเอฟพีจีเอของ Xilinx ดังนั้นไลบารี่ที่ใช้ก็จะเป็นของ Xilinx เช่น ไลบารี่ของตระกูล Sparntan II เป็นต้น

2.5.4 Timing simulation

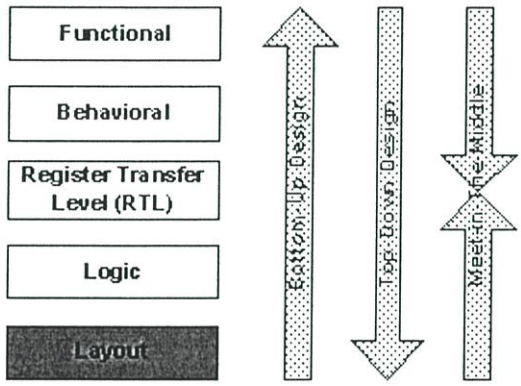
การทำการจำลองการทำงานในระดับพฤติกรรม หรือหลังจากการสังเคราะห์วงจร แต่ยังมิได้ผ่านขั้นตอนการวางและเชื่อมเส้นทางนั้น จะใช้ในการตรวจสอบความถูกต้องทางลอจิกของวงจรเท่านั้น แต่ยังไม่มีรายละเอียดที่แน่นอนเกี่ยวกับเรื่องของเวลา (Timing) มากนัก เช่น ความล่าช้าของสัญญาณไฟฟ้าตามเส้นทางต่างๆภายในวงจร ระยะคาบของสัญญาณเวลาน้อยที่สุดเท่าที่จะสามารถใช้กับวงจรได้ เป็นต้น เพราะจะต้องให้ผ่านขั้นตอนการวางและเชื่อมเส้นทางก่อน ดังนั้นการทำการจำลองการทำงานเพื่อตรวจสอบการทำงานของวงจร ในอุปกรณ์เอฟพีจีเอโดยคำนึงถึงเรื่องไทม์มิ่ง จะกระทำเป็นขั้นตอนสุดท้าย

2.5.5 Place & Route

ผลที่ได้จากการสังเคราะห์นั้นจะเป็นเนตลิสต์ในระดับเกต โดยสามารถนำไปผ่านขั้นตอนการแปลงให้เป็นลอจิกบล็อก และรวมถึงการวางและเชื่อมเส้นทาง (Place & Route) ภายในอุปกรณ์เอฟพีจีเอ โดยใช้ซอฟต์แวร์ Xilinx ISE ของ Xilinx ก็ได้

2.6 การออกแบบจากบนลงล่างและการออกแบบจากล่างขึ้นบน

ในการออกแบบวงจรด้วยเทคโนโลยี ASIC หรือ FPGA มีวิธีการและกระบวนการออกแบบอยู่ 2 วิธีคือ การออกแบบจากบนลงล่าง (Top-Down Design) และการออกแบบจากล่างขึ้นบน (Bottom-Up Design) ดังรูปที่ 2.7 ซึ่งทั้งสองกระบวนการมีข้อดีและข้อเสียแตกต่างกัน



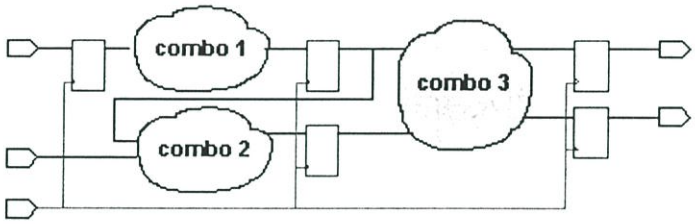
รูปที่ 2.7 กระบวนการออกแบบ (Design Methodology)

การออกแบบจากบนลงล่าง (Top-Down Design) จะมองการออกแบบจากระดับบนสุดของวงจร หรือระบบที่ต้องการออกแบบ นั้นคือ ระดับของฟังก์ชัน (Functional level) การทำงานของวงจรที่ต้องการออกแบบ และลงลึกในรายละเอียดระดับล่างของฟังก์ชัน ในระดับของพฤติกรรม (Behavioral level) การทำงานของแต่ละฟังก์ชัน และออกแบบรายละเอียดของแต่ละฟังก์ชัน ในระดับรีจิสเตอร์ (RTL) ของแต่ละฟังก์ชัน และค่อยวิเคราะห์ลึกลงไป ในรายละเอียดระดับลอจิก (Logic level) และระดับสุดท้าย ซึ่งเป็นระดับล่างที่สุดของการออกแบบ คือระดับผังภูมิวงจร (Layout) ซึ่งผู้ออกแบบจะไม่ทราบรายละเอียดของวงจรที่ออกแบบ จนกว่าจะถึงระดับสุดท้ายของการออกแบบ ซึ่งจะแตกต่างจากการออกแบบจากล่างขึ้นบน (Bottom-Up Design) ที่จะเริ่มออกแบบจากระดับล่างสุด และนำแต่ละส่วนประกอบกัน ขึ้นเป็นบล็อกในระดับบน เพื่อให้วงจรสามารถทำหน้าที่ตามฟังก์ชันนั้นๆได้ กระบวนการออกแบบลักษณะที่ผู้ออกแบบจะทราบรายละเอียดของวงจรทั้งหมด แต่ข้อเสียของกระบวนการนี้ คือผู้ออกแบบจะต้องเลือกเทคโนโลยีใดๆ ก่อนนำไปออกแบบวงจร ซึ่งถ้ามีการเปลี่ยนเทคโนโลยี จะต้องทำการออกแบบวงจรใหม่ แต่

ในทางกลับกันผู้ออกแบบที่ใช้กระบวนการออกแบบจากบนลงล่าง จะไม่ยึดติดกับเทคโนโลยี ในการออกแบบวงจร เพราะการออกแบบลักษณะนี้วงจรจะอยู่ในรูปแบบของโมเดลภาษา HDL ซึ่งถ้าต้องการใช้เทคโนโลยีอะไร ก็เพียงนำไลบรารีของเทคโนโลยีมาสังเคราะห์ เป็นโมเดลของวงจรในระดับล่างต่อไปเท่านั้น นี้คือความแตกต่างระหว่างกระบวนการออกแบบ จากบนลงล่างกับกระบวนการออกแบบจากล่างขึ้นบน

2.7 RTL (Register Transfer Level)

RTL คือการเขียนบรรยายพฤติกรรมของวงจรในระดับของรีจีสเตอร์และวงจรคอมบิเนชัน การออกแบบในระดับนี้ นักออกแบบต้องนึกถึงรูปแบบการประมวลผล, ไทม์มิ่งและการเชื่อมต่อของสัญญาณระหว่าง กลุ่มวงจรคอมบิเนชันและรีจีสเตอร์ โดยที่ไม่จำเป็นพิจารณาพฤติกรรมวงจรระดับลอจิก หรือระดับทรานซิสเตอร์ รูปที่ 2.8 แสดงบล็อกไดอะแกรมของโครงสร้างของ RTL



รูปที่ 2.8 ตัวอย่าง Register Transfer Level บล็อกไดอะแกรม

ในการออกแบบวงจรดิจิตอลด้วย HDL ส่วนใหญ่ จะทำการเขียนบรรยายพฤติกรรมวงจรในระดับ RTL เนื่องจากว่ามีความสะดวกในการออกแบบ และให้วงจรที่มีประสิทธิภาพมากกว่าการเขียนบรรยายในระดับลอจิก ซึ่งใช้เวลาออกแบบนาน และหาข้อผิดพลาดลำบาก หรือในระดับพฤติกรรม ออกแบบง่ายแต่วงจรไม่ให้ผลดีที่สุด (Optimize) ดังนั้นการออกแบบในระดับ RTL จะมีความสะดวกในขั้นตอนของการสังเคราะห์วงจร, การวิเคราะห์เวลา (Timing Analysis) ซึ่งเป็นการวิเคราะห์แก้ปัญหาของค่าเวลาภายในวงจร รวมทั้งการแบ่งวงจรแยกออกจากกันเป็นบล็อกย่อย (Partitioning) สำหรับการสังเคราะห์ (synthesis) เพื่อให้ผลลัพธ์ที่ดีที่สุด และยังสามารถทำความเข้าใจได้ไม่ยากนักสำหรับนักออกแบบ

2.8 บทสรุป

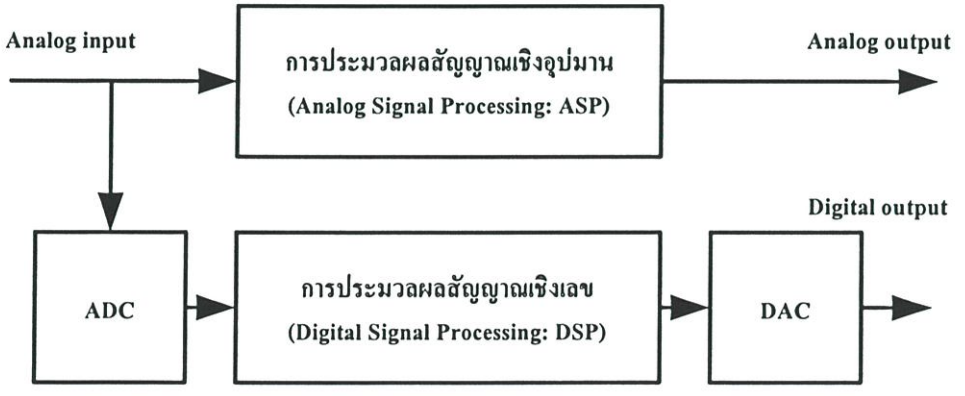
ในบทนี้ได้บรรยายถึงประวัติความเป็นมา, สถาปัตยกรรม, โครงสร้างและองค์ประกอบต่างๆของ FPGA และรูปแบบชนิดต่างๆที่ใช้ในการออกแบบ, ลำดับขั้นในการออกแบบ, ภาษาอธิบายพฤติกรรมทางฮาร์ดแวร์ โดยที่กล่าวมาทั้งหมดนั้นจะพื้นฐานความเข้าใจในการประยุกต์ใช้งาน FPGA ให้ได้ประสิทธิภาพสูงสุด แต่ที่สำคัญคือประสบการณ์ในการแก้ปัญหาทางเทคนิคต่างๆ

บทที่ 3

การประมวลผลสัญญาณ (Signal Processing)

3.1 บทนำ

เราอาจกล่าวได้ว่า การประมวลผลสัญญาณ คือการกระทำบางอย่างกับสัญญาณที่เป็นผลให้เกิดการปรับปรุง หรือเปลี่ยนรูปไป เพื่อประโยชน์ในการสื่อความหมาย หรือการตีความ หรือการตัดสินใจและการควบคุม การประมวลผลสัญญาณนั้นอาจจะกระทำได้โดยตรงต่อสัญญาณ เช่น การขยาย (Amplification) การลดทอน (Attenuation) การกรอง (Filtering) หรืออาจมีการเปลี่ยนรูปแบบของสัญญาณไปก่อน เช่นการประมวลผลในแบบเชิงเลข หรือการประมวลผลดิจิตอล (Digital Signal Processing: DSP) และการประมวลผลที่โดเมนที่แตกต่างกันออกไป โดยพึงสังเกตว่า แม้จะมีการเปลี่ยนแปลงการแทนสัญญาณไป (เช่นเปลี่ยนเป็นสัญญาณดิจิตอล) หรือเปลี่ยนรูปแบบในการประมวลผลต่อไป (เช่นการกระทำในโดเมนอื่น) สัญญาณที่ถูกทำการประมวลผลจะยังคงเป็นสัญญาณเดิม หรือมีความหมายเดิมในตัวมันเองอยู่อย่างครบถ้วน โดยที่การประมวลผลสัญญาณใดๆ เราสามารถแบ่งรูปแบบการประมวลผลได้เป็น 2 ชนิดหลักคือ การประมวลผลสัญญาณเชิงอุปมานหรืออนาลอก (Analog Signal Processing :ASP) และการประมวลผลสัญญาณเชิงเลข (Digital Signal Processing: DSP) แสดงได้ดังรูปที่ 3.1



รูปที่ 3.1 การประมวลผลเชิงอุปมาน และการประมวลผลเชิงเลข

3.2 ข้อดีและข้อเสียของการประมวลผลสัญญาณแบบเชิงเลข

จากรูปแบบการประมวลผลสัญญาณทั้งสองชนิด ต่างก็มีคุณสมบัติเฉพาะตัวที่แตกต่างกัน เราสามารถเลือกพิจารณาการใช้งานระบบใดๆ ได้ตามความเหมาะสมซึ่งพอที่จะแบ่งข้อดี-ข้อเสียของระบบการประมวลผลเชิงเลขได้ดังนี้

3.2.1 ข้อดีของการประมวลผลเชิงเลข

1. รูปแบบการประมวลผลเชิงเลขเหมาะสมกับอุปกรณ์ที่ข้อมูลเป็นรูปแบบสัญญาณเชิงเลขเหมือนกัน ทำให้สะดวกต่อการประมวลผลต่อเนื่องไป เช่น การประมวลผลภาพดิจิตอลด้วยคอมพิวเตอร์ การกรองสัญญาณดิจิตอล
2. อุปกรณ์ทางด้านดิจิตอลมีแนวโน้มราคาต่ำลง, ขนาดเล็กลง, ประสิทธิภาพสูงขึ้น ความเที่ยงตรงเพิ่มมากขึ้น ความเร็วในการทำงานก็สูงขึ้น โดยเฉพาะเรื่องความเที่ยงตรงแม่นยำในการประมวลผล จะเป็นจุดเด่นในการประมวลผลแบบเชิงเลข
3. การรับส่งข้อมูลทำได้แน่นอนกว่าสัญญาณอนาลอก เนื่องจากสัญญาณเชิงเลขมีค่าระดับเพียงศูนย์ (0) หรือ หนึ่ง (1) เท่านั้น การแก้ไขข้อมูลที่ผิดพลาดก็สามารถทำได้โดยง่าย
4. การประมวลผลเชิงเลขสามารถทำได้โดยง่ายเพราะอัลกอริทึม (Algorithm) ในการประมวลผลมีเพียง การบวก, การลบ, การคูณ, และการเลื่อนตัวเลข
5. การประมวลผลเชิงเลขสามารถทำได้พร้อมๆกันหลายๆช่องสัญญาณในลักษณะ การแบ่งช่วงเวลาทำงาน (Time Sharing)
6. มีความคล่องตัวสูง เนื่องจากสามารถทำการมัลติเพล็กข้อมูลได้ อีกทั้งยังสามารถส่งสัญญาณข้อมูลในอัตราบิตข้อมูล ที่ต่างกันได้ (Bits Rate, Baud Rate)
7. ความผิดพลาดของอุปกรณ์ที่นำมาสร้างเป็นตัวประมวลผลไม่จำเป็นต้องมีความถูกต้องแม่นยำสูงมากนัก
8. มีเสถียรภาพที่ดี ไม่ว่าจะในระยะสั้น (เช่นผลต่อการเปลี่ยนแปลง อุณหภูมิ) หรือในระยะยาว (เช่น การเปลี่ยนแปลงของบางพารามิเตอร์ของอุปกรณ์)
9. ความแม่นยำของการควบคุมสามารถกำหนดได้ตามจำนวนบิตข้อมูลที่ใช้ หรือความยาวของคำ (Word Length)
10. ตัวประมวลผลเชิงเลขสามารถทำงานซ้ำหน้าที่ (Function) เดิมได้โดยไม่จำกัดครั้ง

3.2.2 ข้อเสียของการประมวลผลเชิงเลข

1. ระบบประมวลผลเชิงเลขต้องมีสัญญาณสำหรับ การซิงโครไนซ์ (Synchronize), การจัดเวลา (Timing) และการกำหนดกรอบ (Framing) ของข้อมูลด้วยถ้าสัญญาณเหล่านี้สูญหาย หรือผิดพลาดไป การทำงานของระบบก็จะผิดพลาดไปด้วย
2. ปัญหาเรื่องการเชื่อมต่อ (Interfacing) กับระบบการประมวลผลสัญญาณอนาลอก ทำให้วงจรหรือระบบมีความซับซ้อนมากขึ้น
3. สัญญาณเชิงเลขไม่ใช่สัญญาณตามธรรมชาติ แต่ถูกสร้างขึ้นมา ดังนั้นการส่งสัญญาณนี้ไปในตัวกลางธรรมชาติจะมีแถบปฏิบัติงาน (Band Width) ที่จำกัด และยังมี

ผลตอบสนองเฟสไม่เป็นเชิงเส้น (Non-linear Phase Response) จึงทำให้เกิดความผิดเพี้ยนได้

4. การออกแบบระบบประมวลผลเชิงเลข จะมีความซับซ้อนมากกว่า โดยเฉพาะกับระบบประมวลผลเชิงเลขขนาดใหญ่
5. แถบปฏิบัติงาน (Band Width) ของระบบประมวลผลเชิงเลขต่ำกว่าระบบประมวลผลเชิงอุปมานมาก เป็นข้อจำกัดของอุปกรณ์ที่ใช้ เช่น วงจรเกต, รีจีสเตอร์, ADC, DAC เป็นต้น
6. ระบบประมวลผลเชิงเลขมักจะต้องมีไฟเลี้ยงอยู่เสมอ ในขณะที่ระบบประมวลผลเชิงอุปมานอย่างง่ายๆไม่จำเป็นต้องใช้ไฟเลี้ยง
7. สัญญาณเชิงอุปมานเมื่อถูกเปลี่ยนเป็นเชิงเลขแล้ว จะทำให้ความถูกต้องของสัญญาณบางส่วนขาดหายไป (เนื่องจากการตัดทิ้ง : Truncation, และการปัดเศษ : Round Off) โดยไม่สามารถจะทำคืนให้ถูกต้องเหมือนเดิมได้

จากตารางที่ 3.1 แสดงให้เห็นว่าความคลาดเคลื่อนที่เกิดขึ้นเนื่องจากการปัดเศษ (Round-Off Error) และการตัดเศษทิ้ง (Truncation Error) โดย Round-Off Error มีค่าเป็น $E_{Rnd} = x(n) - Rnd[x(n)]$ ส่วน Truncation Error มีค่าเป็น $E_{Trc} = x(n) - Trc[x(n)]$ เมื่อ $x(n) = (0.7)^n$ ถ้าให้การปัดทศนิยมและการตัดทศนิยมนั้นเพียง 2 ตำแหน่ง จะเห็นว่าการปัดทศนิยมให้ค่าความคลาดเคลื่อนได้ทั้งค่าบวกและลบ แต่จะน้อยกว่าความคลาดเคลื่อนจากการตัดทิ้ง

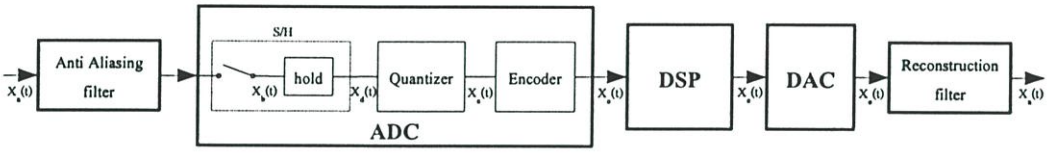
ตารางที่ 3.1 แสดงความคลาดเคลื่อนที่เกิดขึ้นเนื่องจากความจำกัดในการเก็บค่า

n	$x(n) = (0.7)^n$	Rnd{x(n),2}	Trc{x(n),2}	E_{Rnd}	E_{Trc}
1	0.7	0.70	0.70	0.00	0.00
2	0.49	0.49	0.49	0.00	0.00
3	0.343	0.34	0.34	+0.003	0.003
4	0.2401	0.24	0.24	+0.0001	0.0001
5	0.16807	0.17	0.16	-0.00193	0.00807
6	0.117649	0.12	0.11	-0.002351	0.007649
7	0.0823543	0.08	0.08	+0.0023543	0.0023543
8	0.05764801	0.06	0.05	-0.00235199	0.00764801
9	0.040353607	0.04	0.04	+0.000353607	0.000353607
10	0.028247525	0.03	0.02	-0.001752475	0.008247525

3.3 โครงสร้างของระบบการประมวลผลสัญญาณเชิงเลข

(Structure of Digital Signal Processing)

เราสามารถพิจารณาการประมวลผลของสัญญาณเชิงเลข แสดงตามรูปที่3.2 ได้สองลักษณะ คือ การพิจารณาในลักษณะโครงสร้างของระบบ DSP และการพิจารณาลักษณะของสัญญาณภายในระบบ DSP



รูปที่ 3.2 บล็อกไดอะแกรมโครงสร้างระบบ DSP ที่กระทำกับสัญญาณอนาลอก

3.3.1 ลักษณะของโครงสร้างระบบ DSP

การพิจารณาในส่วนนี้จะเป็นไปในลักษณะขององค์ประกอบทางฮาร์ดแวร์ (Hardware) เป็นหลัก ซึ่งจะแยกส่วนประกอบต่างๆภายในระบบ ได้ดังต่อไปนี้

3.3.1.1 ตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล (Analog to Digital Converter : ADC)

จะเป็นวงจรเชื่อมต่อที่ทำการแปลงสัญญาณอนาลอกไปเป็นสัญญาณดิจิตอล ดังนั้นเพื่อให้สัญญาณอนาลอกที่จะถูกแปลงโดย ADC จะต้องสุ่มวางค่าขนาดแอมปลิจูด (Amplitude) ค้างไว้ในช่วงเวลาสั้นๆช่วงหนึ่ง จนกว่าการแปลงจะเสร็จสมบูรณ์ จึงจะทำให้ความผิดพลาดมีค่าน้อยที่สุด สามารถทำได้โดยวงจร Sample & Hold (S/H) ซึ่งจะ จากนั้นจะถูกจัดระดับสัญญาณ (Quantized) ที่ตัวจัดระดับสัญญาณ (Quantizer) ให้เป็นระดับอ้างอิงที่ใกล้เคียงกันที่สุด โดยระดับอ้างอิงนี้จะแบ่งเป็น $2^n - 1$ ระดับ เมื่อ n เป็นจำนวนบิตของการเปลี่ยนแปลง หลังจากนั้นจึงถูกเข้ารหัสเป็นสัญญาณข้อมูลเลขฐานสองขนาด n บิต แล้วจึงทำการสุ่มสัญญาณที่เข้ามาใหม่อีกครั้ง ทำเช่นนี้ซ้ำๆ เรื่อยไป

3.3.1.2 ตัวประมวลผลสัญญาณเชิงเลข (Digital Signal Processor)

ในส่วนนี้จะทำการประมวลผลสัญญาณเชิงเลขโดยมีรูปแบบอัลกอริทึมเฉพาะหน้าที่ ซึ่งขึ้นอยู่กับเงื่อนไขของสัญญาณทางออก ว่าต้องการให้มีลักษณะเป็นเช่นไร เช่น การทำการกรองสัญญาณความถี่สูง, การกรองสัญญาณรบกวน หรือ การมอดูเลตสัญญาณ เป็นต้น โดยรูปแบบการประมวลผลเชิงเลขจะมี ตัวดำเนินการ (Operator) คือ ตัวบวก (Adder) และตัวคูณ (Multiplier) ที่มักจะถูกใช้ดำเนินการกับสัญญาณเชิงเลขอยู่บ่อยๆ อย่างไรก็ตามวิทยานิพนธ์ฉบับนี้ จะทำการ

นำเสนอการประมวลผลสัญญาณเชิงเลขโดยใช้อัลกอริทึมดำเนินการต่างออกไป ซึ่งจะอธิบายละเอียดในภายหลัง

3.3.1.3 ตัวแปลงสัญญาณดิจิตอลเป็นอนาลอก (Digital to Analog Converter: DAC)

จะทำหน้าที่ตรงกันข้ามกันกับ ADC คือแปลงสัญญาณดิจิตอลไปเป็นสัญญาณอนาลอกในกรณีที่เอาท์พุทของ DAC เป็นลักษณะเหมือนขั้นบันได มันอาจจำเป็นต้องทำการปรับค่าจาก DAC ให้สัญญาณอนาลอกมีความเรียบขึ้น สามารถทำได้โดยใช้วงจร Reconstruction (Smoothing) Filter

แต่อย่างไรก็ตาม การประยุกต์ใช้งานส่วนมากแล้วสัญญาณอนาลอกที่จะถูกประมวลผลมักจะมี Band Width ที่กว้างกว่าตัวประมวลผลสัญญาณดิจิตอล เพื่อป้องกันปรากฏการณ์นี้ซึ่งเรียกว่า (Aliasing Effect) จะใช้ตัวกรองอนาลอก (Anti-aliasing Filter) ต่อไว้ก่อนภาคของวงจร S/H

3.3.2 ลักษณะของสัญญาณภายในระบบ DSP

คำจำกัดความของสัญญาณ 5 ชนิด ที่มักปรากฏในระบบการประมวลผลสัญญาณเชิงเลขแปรเปลี่ยนต่อเวลาดังนี้

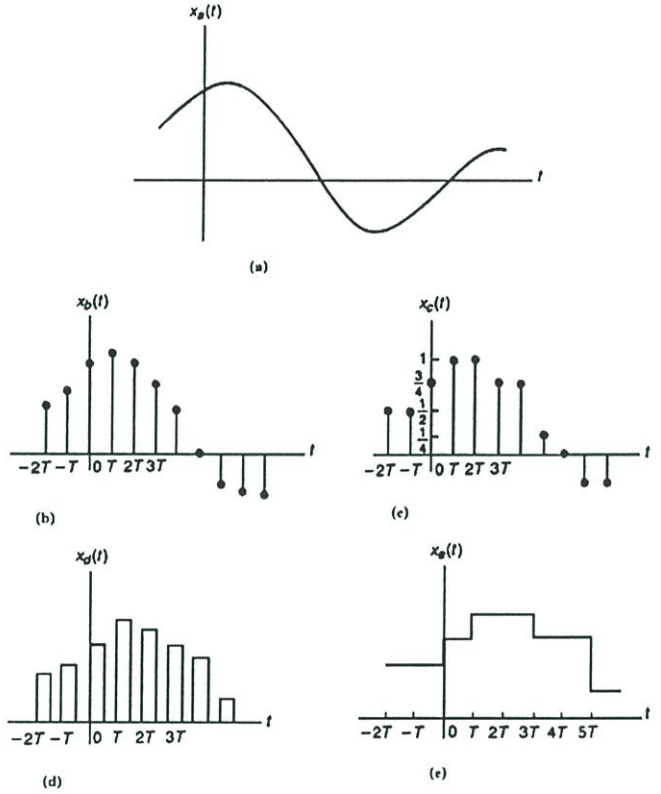
3.3.2.1 เวลาต่อเนื่องและแอมปลิจูดต่อเนื่อง ปกติมักเรียกว่าสัญญาณอนาลอกโดยเป็นสัญญาณซึ่งเรามักจะพบเจอในปรากฏการณ์ทางกายภาพ ซึ่งเราสมมุติว่าตลอดช่วงทั้งเวลาและแอมปลิจูดมีความต่อเนื่อง ดังรูปที่3.3 (a) สัญญาณอนาลอกปรากฏในหลายแขนงวิชาเช่น วิศวกรรม, วิทยาศาสตร์, การแพทย์, เศรษฐศาสตร์และธรณีวิทยา เป็นต้น

3.3.2.2 เวลาไม่ต่อเนื่องและแอมปลิจูดต่อเนื่อง ปกติมักเรียกว่า ตัวอย่างของสัญญาณ (Sample) ขั้นของเวลาจะทิ้งช่วงสม่ำเสมอ แต่สัญญาณมีได้หลายระดับดังรูปที่3.3 (b) แสดงให้เห็นช่วงของการสุ่มตัวอย่าง T วินาที ที่ใช้แสดงกระบวนการสุ่มสัญญาณ จะได้กล่าวถึงในหัวข้อต่อไป

3.3.2.3 เวลาไม่ต่อเนื่องและแอมปลิจูดไม่ต่อเนื่อง เช่น การควอนไตซ์ของแอมปลิจูด ด้วย Step เวลาที่คงที่ ดังนั้นเราอาจกล่าวได้ว่าทั้งแอมปลิจูดและเวลานั้นถูกควอนไตซ์ โดย ADC จะเป็นตัวสร้างสัญญาณชนิดนี้ ซึ่งรู้จักในนามของ สัญญาณดิจิตอลลักษณะดังรูปที่3.3 (c)

3.3.2.4 เวลาต่อเนื่องและแอมปลิจูดต่อเนื่องด้วยขั้นของเวลาที่คงที่ รู้จักในนาม สัญญาณสุ่มอนาลอก หรือ สัญญาณข้อมูลที่ถูกสุ่มและมีลักษณะของเอาท์พุท (o/p) เป็นอุปกรณ์การสุ่มและค้างค่าสัญญาณไว้ (S/H) สัญญาณสามารถมีตลอดช่วงต่อเนื่องและแอมปลิจูดเป็นผลลัพธ์จากการสุ่มสัญญาณอนาลอกทุกๆเวลา T วินาที ดังรูปที่3.3 (d) ค่าที่ถูกสุ่มจะถูกค้างคงที่ตลอดช่วงเวลาเมื่อสัญญาณเป็นศูนย์ ในรูปที่3.3 (d), อุปกรณ์ S/H จะจับค่าอนาลอกอินพุท (i/p) ไว้ในการเปรียบเทียบกับการสุ่มถัดไป

3.3.2.5 เวลาต่อเนื่องและแอมปลิจูดไม่ต่อเนื่องด้วยขั้นของเวลาคงที่ คล้ายกับสัญญาณดิจิตอลที่ได้อธิบายในข้อ (c) แต่สัญญาณจะค้างค่าระหว่างเวลาที่สุ่ม สัญญาณนี้แสดงดังรูปที่3.3 (e) เป็นลักษณะเฉพาะของ o/p จาก DAC

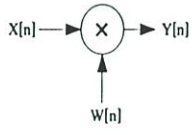


รูป 3.3 ชนิดของสัญญาณที่พบในระบบประมวลผลดิจิตอล

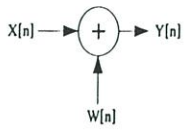
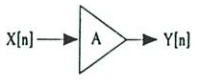
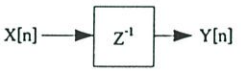
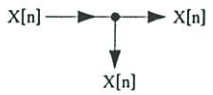
3.4 โอเปอเรชันของลำดับสัญญาณ (Operation on Sequence)

ในระบบการประมวลผลสัญญาณเชิงเลขหนึ่งๆ จะมีโอเปอเรชันของสัญญาณ หรือการกระทำต่อลำดับสัญญาณ ซึ่งโอเปอเรชันที่มักจะได้พบในระบบการประมวลผลเชิงเลขแสดงดังตารางที่ 3.2

ตารางที่ 3.2 แสดงการดำเนินการกับสัญญาณเชิงเลขพื้นฐานที่ใช้ในระบบ DSP

โอเปอเรชัน	สัญลักษณ์	หมายเหตุ
การคูณกันของสัญญาณ (Modulation Operation)	 $Y[n] = X[n] \times W[n]$	- จะเรียกตัวดำเนินการคูณกันของสัญญาณว่า มอดูเลเตอร์ (Modulator) - ใช้งานในการกำหนดรูปแบบของลำดับให้มีความยาวที่จำกัด (Finite-length Sequence) จากลำดับอนันต์ เช่น การทำวินโดว์ (Windowing)

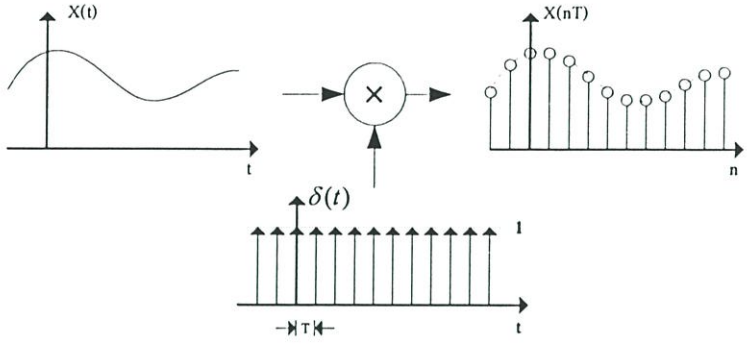
ตารางที่ 3.2 (ต่อ)

การบวกกันของสัญญาณ (Addition Operation)	 $Y[n] = X[n] + W[n]$	- จะเรียกตัวดำเนินการบวกกันของสัญญาณว่า แอดเดอร์ (Adder) - ใช้ในการรวมสัญญาณ (บวก) อย่างพีชคณิต
การคูณสัญญาณด้วยค่าคงที่ (Multiplication Operation)	 $Y[n] = A \cdot X[n]$	- จะเรียกตัวดำเนินการคูณค่าคงที่กับสัญญาณว่า มัลติพลายเออร์ (Multiplier) - ใช้ในการสเกลขนาดของสัญญาณ (Scaling) เมื่อ A เป็นเลขจำนวนใดๆ
การหน่วงเวลา (Delaying Operation)	 $Y[n] = X[n-1]$	- จะเรียกตัวดำเนินการหน่วงเวลากับสัญญาณว่า ตัวหน่วงเวลาหนึ่งหน่วย (Unit Delay) - ใช้ในการหน่วงค่าสัญญาณไปหนึ่งหน่วยเวลา มักอยู่ในวงจรกรองเชิงเลข
การแตกกิ่ง (Branching Operation)		- ใช้ในการคัดลอกสัญญาณออกเป็นหลายๆชุด

จากโอเปอเรชันต่างๆ ที่ได้กล่าวไปแล้วข้างต้น ทำให้เราสามารถทำการสร้างขึ้นเป็นระบบ DSP ได้ โดยการเชื่อมต่อกันของโอเปอเรเตอร์ (Operator) โดยจะได้อธิบายในเรื่องตัวกรองเชิงเลขต่อไป

3.5 ทฤษฎีการสุ่มตัวอย่าง

จากที่ธรรมชาติของสัญญาณต่างๆมักจะเป็นสัญญาณที่ต่อเนื่องทางเวลา ดังนั้นในการประมวลผลสัญญาณเชิงเลข สัญญาณต่อเนื่องดังกล่าวจะถูกเปลี่ยนรูปให้เป็นสัญญาณเชิงเลขให้สอดคล้องกัน วิธีการก็คือการแทนสัญญาณต่อเนื่องเป็นช่วงๆ ห่างกันที่เวลา Ts คงที่ (คือการสุ่มตัวอย่าง) ถ้าหาก Ts มีค่าที่เหมาะสมแล้วการแทนดังกล่าวจะยังคงความถูกต้องเอาไว้ ดังนั้นผลการสุ่มก็คือการคูณสัญญาณต่อเนื่องด้วยอิมพัลส์ที่เลื่อนไปเรื่อยๆ แต่ละครั้งก็ห่างกันเป็นเวลา $T = Ts = 1/f_s$ กระบวนการสุ่มสัญญาณสามารถแสดงได้ดังรูปที่ 3.4



รูปที่ 3.4 การสุ่มสัญญาณ (Sampling Signal)

ค่าความถี่ของการสุ่มสัญญาณ (f_s) ซึ่งไม่ทำให้สัญญาณสูญเสียข้อมูลที่สำคัญไป ทฤษฎีการสุ่มตัวอย่าง (Sampling Theory) ของ แชนนอน (Shannon) กล่าวไว้ว่า *"ถ้าหากสัญญาณต่อเนื่อง x(t) ที่มีความถี่ไม่เกิน* $\omega_{\max} = 2\pi f_{a\max}$ *ข้อมูลของสัญญาณต่อเนื่องนั้นสามารถจะอธิบายได้ด้วย x(nT) ก็ต่อเมื่อความถี่ในการสุ่มตัวอย่าง* f_s *มีค่ามากกว่าหรือเท่ากับ สองเท่าของความถี่ปฏิบัติงาน* ($f_{a\max}$) *หรือ* $f_s \geq 2f_{a\max}$*"* โดยทั่วไปเราอาจสุ่มตัวอย่างด้วยค่าความถี่ $f_{sN} = 2f_{a\max}$ พอดี ค่าความถี่นี้มีชื่อรียกว่า ความถี่ ไนควิสต์ (Nyquist frequency) และคาบเวลา $T_N = 1/(2f_{a\max})$ นี้เรียกว่า ช่วงเวลาสุ่มตัวอย่างไนควิสต์ (Nyquist interval)

3.6 พื้นฐานวงจรกรองอนาลอกต้นแบบ

เนื่องจากในการออกแบบวงจรกรองดิจิตอลอาศัยพื้นฐานของวงจรกรองอนาลอกเป็นต้นแบบ ดังนั้นในการเริ่มต้นออกแบบวงจรควรทราบการคำนวณฟังก์ชันถ่ายโอนของวงจรอนาลอกให้ได้ตามข้อกำหนดที่ต้องการก่อน หลังจากนั้นจึงเขียนให้อยู่ในรูปของการเชื่อมต่อของอุปกรณ์อนาลอกชนิดต่างๆ หรืออาจใช้วิธีการแปลงสู่โดเมนดิจิตอลจากฟังก์ชันถ่ายโอนโดยตรงเพื่อสร้างเป็นวงจรกรองดิจิตอลชนิดต่างๆขึ้นมา

3.6.1 ฟังก์ชันถ่ายโอนของวงจรกรองอนาลอก

การสร้างฟังก์ชันถ่ายโอนต้นแบบอาศัยการประมาณพหุนามตามวิธีการแบบ บัทเทอร์เวิร์ธ เชบีเชฟ และอิลลิปติค ซึ่งอาจใช้วิธีการเปิดตาราง หรือใช้โปรแกรมช่วยในการคำนวณจากข้อกำหนดที่ต้องการ รูปแบบของฟังก์ชันถ่ายโอนต้นแบบในโดเมน s เป็นดังสมการที่ (3.1)-(3.3)

$$H(s) = \frac{k(s-z_1)(s-z_2)\ldots}{(s-p_0)(s-p_1)(s-p_1^*)\ldots} \tag{3.1}$$

$$H(s) = \frac{N(s)}{(s+p_1)\prod\prod(s+p_i)(s+p_i^*)} = \frac{N(s)}{D(s)} \tag{3.2}$$

$$H(s) = \frac{\sum_{i=0}^{m} b_i s^i}{\sum_{i=0}^{n} a_i s_i} \tag{3.3}$$

เมื่อ z_1 , z_2 ,...,z_m เป็นซีโร่ของวงจรกรอง $p_0, p_1, p_1^*, ..., p_n$ เป็นโพลของวงจรกรอง k เป็นอัตราขยาย และ a_i , b_i เป็นสัมประสิทธิ์ของพหุนาม s

3.6.2 วงจรกรองอนาลอกต้นแบบชนิดต่างๆ

ในที่นี้จะได้นำเสนอการคำนวณหาข้อมูลที่จำเป็นต้องใช้สำหรับการสร้างพหุนามต้นแบบที่มีรูปแบบใดรูปแบบหนึ่ง ตามสมการที่ (3.1)-(3.3) ข้อมูลที่จำเป็นต้องทราบคือ อันดับ (Order) และความถี่ตัด (Cut-off Frequency) ของวงจรกรอง ซึ่งก็คำนวณมาจากข้อกำหนดทางขนาดและความถี่ของวงจรกรองตามการประมาณชนิดต่างๆ นั่นเอง เมื่อได้ข้อมูลข้างต้นมาแล้ว ก็สามารถนำมาสร้างเป็นฟังก์ชันถ่ายโอนได้ด้วยวิธีการใน [10]

3.6.2.1 วงจรกรองอนาลอกต้นแบบ บัทเทอร์เวิร์ธ

ฟังก์ชันถ่ายโอนของวงจรกรองอนาลอกต้นแบบ บัทเทอร์เวิร์ธ สร้างจาก อันดับ และความถี่ตัดของวงจรกรอง แสดงดังสมการ

$$N = \frac{\log\left\{\left(10^{Rp/10}-1\right)\left(10^{As/10}-1\right)\right\}}{2\log\left(\Omega_p/\Omega_s\right)} \tag{3.4}$$

และ

$$\Omega_c = \frac{\Omega_p}{\sqrt[2N]{10^{Rp/10}-1}} \tag{3.5}$$

เมื่อ N คือ อันดับของวงจรกรอง

Rp คือ การกระเพื่อมในแถบผ่าน [dB]

As คือ การลดทอนในแถบหยุด [dB]

Ω_p คือ ความถี่เชิงมุมอนาลอกที่ขอบแถบผ่าน [rad/s]

Ω_s คือ ความถี่เชิงมุมอนาลอกที่ขอบแถบหยุด [rad/s]

Ω_c คือ ความถี่ตัดเชิงมุมอนาลอก [rad/s]

3.6.2.2 วงจรกรองอนาลอกต้นแบบ เชบีเชฟ

ฟังก์ชันถ่ายโอนของวงจรกรองอนาลอกต้นแบบ เชบีเชฟ สร้างจาก อันดับ และความถี่ตัดของวงจรกรอง แสดงดังสมการ

$$N = \frac{\log\left(g + \sqrt{g^2 - 1}\right)}{\log\left(\Omega_r + \sqrt{\Omega_r^2 - 1}\right)} \tag{3.6}$$

เมื่อ $$g = \sqrt{\frac{10^{0.1As} - 1}{10^{0.1Rp} - 1}} \text{ และ } \Omega_r = \frac{\Omega_s}{\Omega_p}$$

$$\Omega_c = \Omega_p \tag{3.7}$$

เมื่อ N คือ อันดับของวงจรกรอง

Rp คือ การกระเพื่อมในแถบผ่าน [dB]

As คือ การลดทอนในแถบหยุด [dB]

Ω_p คือ ความถี่เชิงมุมอนาลอกที่ขอบแถบผ่าน [rad/s]

Ω_s คือ ความถี่เชิงมุมอนาลอกที่ขอบแถบหยุด [rad/s]

Ω_c คือ ความถี่ตัดเชิงมุมอนาลอก [rad/s]

3.6.2.3 วงจรกรองอนาลอกต้นแบบ อิลลิปติค

ฟังก์ชันถ่ายโอนของวงจรกรองอนาลอกต้นแบบ อิลลิปติค สร้างจาก อันดับ และความถี่ตัดของวงจรกรอง แสดงดังสมการ

$$N = \frac{K(k)K\left(\sqrt{1-k_1^2}\right)}{K(k_1)K\left(\sqrt{1-k^2}\right)} \tag{3.8}$$

เมื่อ $k = \frac{\Omega_p}{\Omega_s}$, $k_1 = \sqrt{\frac{10^{0.1Rp} - 1}{10^{0.1As} - 1}}$ และ $K(x) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - x^2 \sin^2\theta}}$

และมีความถี่ตัดเช่นเดียวกับของวงจรกรอง เชบีเชฟ คือ

$$\Omega_c = \Omega_p \tag{3.9}$$

เมื่อ N คือ อันดับของวงจรกรอง

Rp คือ การกระเพื่อมในแถบผ่าน [dB]

As คือ การลดทอนในแถบหยุด [dB]

Ω_p คือ ความถี่เชิงมุมอนาลอกที่ขอบแถบผ่าน [rad/s]

Ω_s คือ ความถี่เชิงมุมอนาลอกที่ขอบแถบหยุด [rad/s]

Ω_c คือ ความถี่ตัดเชิงมุมอนาลอก [rad/s]

3.7 การออกแบบตัวกรองดิจิตอลโดยใช้การแปลงตัวแปร

วิธีการออกแบบตัวกรองดิจิตอลที่นิยมมากวิธีหนึ่งก็คือ การออกแบบโดยอิงตัวกรองอนาลอกต้นแบบ ซึ่งได้แก่ตัวกรองแบบ บัทเทอร์เวิร์ธ, เชบีเชฟ, อิลลิปติค และอื่นๆ โดยเราสามารถใช้ฟังก์ชันหรือการเปลี่ยนแปลงอย่างใดอย่างหนึ่งที่สามารถแปลงฟังก์ชันถ่ายโอนของระบบอนาลอก (ซึ่งอยู่ใน S โดเมน) มาเป็นระบบดิจิตอล (Z โดเมน) ได้ ซึ่งการแปลงที่ใช้มีอยู่หลายแบบ เช่น การแปลงอิมพัลส์ไม่แปรเปลี่ยน (Impulse-Invariant Transform), การแปลงไบลิเนียร์ (Bilinear Transform) การแปลงย้อนกลับ และการแปลงไปข้างหน้า โดยหัวข้อนี้จะกล่าวเพียงการแปลงไบลิเนียร์ซึ่งเป็นที่นิยมใช้มากกว่า

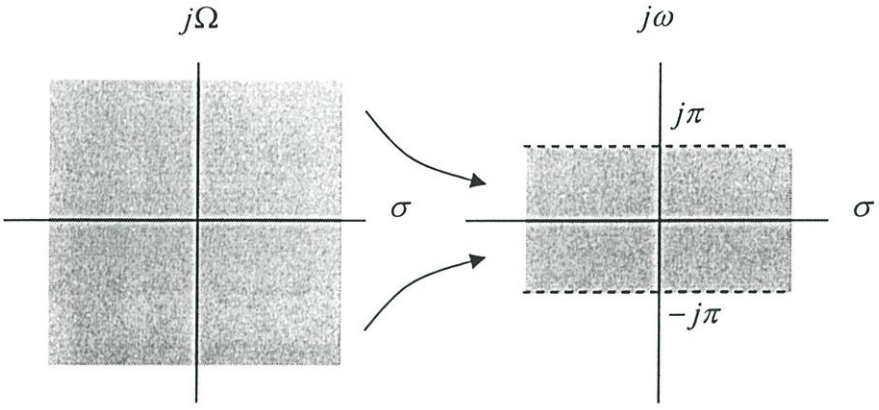
3.7.1 การแปลงไบลิเนียร์

เพื่อหลีกเลี่ยงการเกิด aliasing ของการการตอบสนองความถี่ซึ่งอาจเกิดขึ้นได้ในวิธีการแปลงแบบอิมพัลส์ไม่แปรเปลี่ยน เราจึงต้องทำการวาด (Mapping) แบบหนึ่งต่อหนึ่ง (One-to-one) จากระนาบแกน s ไปสู่ระนาบแกน z ปัญหาที่เกิดจากการแปลง $z = e^{sT}$ คือเป็นการวาดแบบมากมายไปหนึ่ง (Many-to-one) อย่างไรก็ตาม ถ้าเราเริ่มใช้การแปลงแบบหนึ่งต่อหนึ่งจาก s ไปสู่ s' ก่อน ซึ่งจะทำการบีบอัดทั้งระนาบ s เข้าไปในบริเวณช่วง $-\pi/T \le \mathrm{Im}(s') \le \pi/T$ ดังนั้น s' สามารถถูกแปลงไปสู่แกน z ด้วย $z = e^{sT}$ อย่างไม่มีผลกระทบต่อการเกิด aliasing การวาดที่ต้องการจากระนาบ s ไปยัง s' แสดงดังรูปที่ 3.4 ซึ่งการแปลงแบบหนึ่งต่อหนึ่งจาก s ไปยัง s' หาได้จากสมการ(3.10)

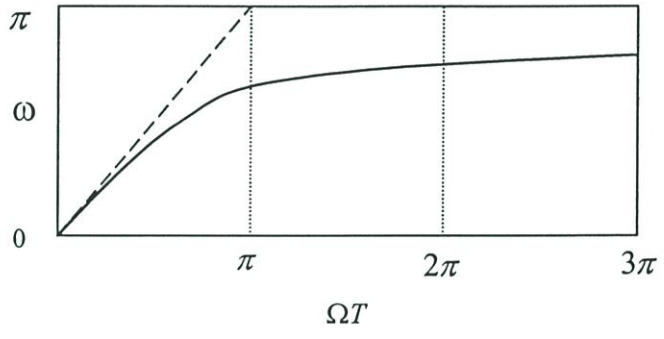
$$s' = \frac{2}{T}\tanh^{-1}\left(\frac{sT}{2}\right) \tag{3.10}$$

ธรรมชาติของการแปลงนี้จะเห็นได้ว่าง่าย จากผลกระทบบนแกน $j\Omega$ การแทน $s = j\Omega$ และ $s' = j\Omega'$ ลงในสมการ(3.10) เราจะได้ว่า

$$\Omega' = \frac{2}{T}\tan^{-1}\left(\frac{\Omega T}{2}\right) \tag{3.11}$$



รูปที่ 3.5 การวาด (Mapping) จากระนาบ s ไปยังระนาบ s' เพื่อเลี่ยงการเกิด aliasing



รูปที่ 3.6 การวาด (Mapping) จาก Ω ไปยัง ω ด้วยการแปลงไบลิเนียร์

หรือถ้าความถี่นอมัลไลซ์ ω ที่สอดคล้องกับ $\Omega'T$ จะได้ว่า

$$\omega = 2\tan^{-1}\left(\frac{\Omega T}{2}\right) \tag{3.12}$$

สมการ (3.12) สามารถวาดกราฟความสัมพันธ์ได้ดังรูปที่ 3.6 ดังนั้น ตลอดทั้งแกน Ω จะถูกบีบอัดลงสู่ช่วง $(-\pi,\pi)$ สำหรับ ω ในลักษณะหนึ่งต่อหนึ่งตามต้องการ จะสังเกตได้ว่าความสัมพันธ์ระหว่าง ω และ Ω จะไม่เป็นเชิงเส้น จะประมาณว่าเป็นเชิงเส้นสำหรับค่าน้อยๆที่ $\omega \approx \Omega T$

การแปลงที่เราต้องการจาก s ไป z ตอนนี้จะได้โดยการแปลงกลับ สมการ (3.10) ได้เป็น

$$s = \frac{2}{T}\tanh\left(\frac{s'T}{2}\right) \tag{3.13}$$

แทนค่า $z = e^{s'T}$ หรือ $s' = (1/T)\ln z$ ซึ่งผลที่ได้คือ

$$s = \frac{2}{T}\tanh\left(\frac{\ln z}{2}\right) \tag{3.14}$$

ซึ่ง

$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3.15}$$

แทนค่า (3.15) ใน (3.14) จะได้ว่า

$$s = \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) \tag{3.16}$$

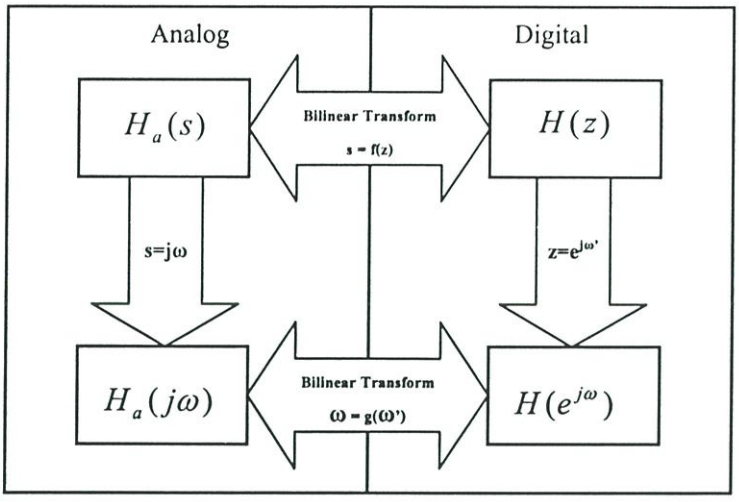
ดังนั้นตัวกรองไม่ต่อเนื่องเชิงเวลาหรือตัวกรองดิจิตอลที่ออกแบบได้จากการออกแบบเชิงเวลาต่อเนื่อง ด้วยการแปลงไบลิเนียร์ คือ

$$H(z) = H_a(s)\Big|_{s=\frac{2}{T}(1-z^{-1})/(1+z^{-1})} \tag{3.17}$$

จะได้ การแปลงกลับของสมการ (3.16) เป็น

$$z = \frac{1 + \frac{T}{2}s}{1 - \frac{T}{2}s} \tag{3.18}$$

ในรูปที่ 3.7 แสดงความสัมพันธ์กันของระบบทั้งสอง จะเห็นได้ว่า จากฟังก์ชันถ่ายโอนของระบบต่อเนื่อง ($H_a(s)$ ซึ่งเป็นฟังก์ชันของ s) เราสามารถหาผลตอบสนองเชิงความถี่ได้ โดยแทน $s = j\omega$ และสำหรับฟังก์ชันถ่ายโอนของระบบไม่ต่อเนื่อง (H(z) ซึ่งเป็นฟังก์ชันของ z) ก็สามารถหาผลตอบสนองความถี่ได้โดยแทน $z = e^{j\omega'}$

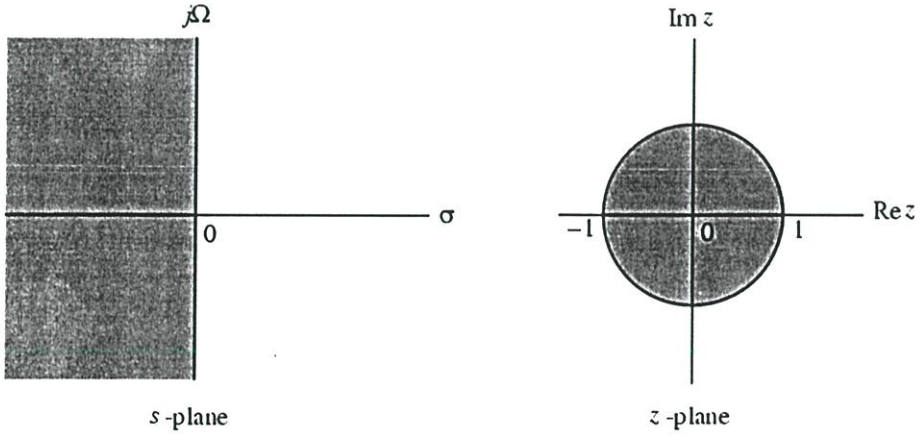


รูปที่ 3.7 ความสัมพันธ์กันของระบบแบบต่อเนื่อง และไม่ต่อเนื่อง

3.7.2 ผลที่เกิดจากการแปลงไบลิเนียร์สามารถมองได้เป็น 2 จุดใหญ่ ๆ คือ

1) เกิดการแปลงโพล และซีโร่ บนระนาบ s ของระบบต่อเนื่องไปเป็นโพล และซีโร่บนระนาบ z ของระบบไม่ต่อเนื่อง

ซึ่งจุดที่เราให้ความสนใจเป็นพิเศษ คือ เกิดการดึงพื้นที่ในซีกซ้ายของระนาบ s ไปยังพื้นที่ภายใต้วงกลมขนาด 1 หน่วยของระนาบ z ถ้าระบบแบบอนาลอกมีโพลอยู่ในซีกซ้ายของระนาบ s เมื่อแปลงเป็นระบบดิจิตอล โพลนั้นก็จะอยู่ภายใต้วงกลมขนาด 1 หน่วยของระนาบ z ดังในรูปที่ 3.8



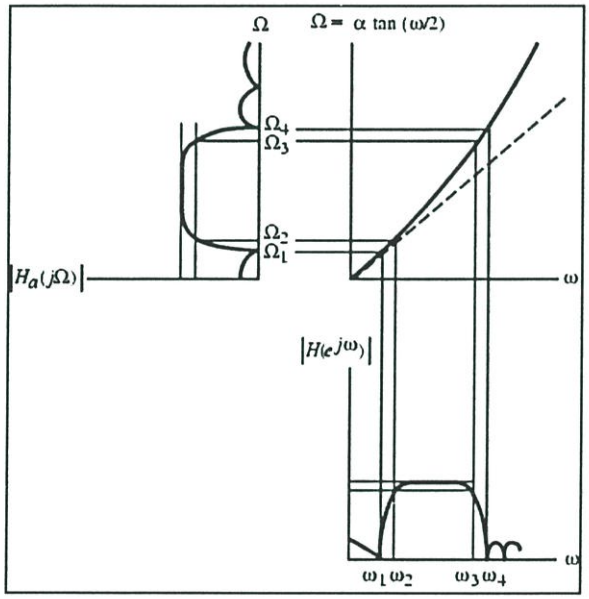
รูปที่ 3.8 การแปลงระหว่างโพลในระนาบ s ไปยังโพลในระนาบ z

2) เกิดการแปลงระหว่างความถี่อนาลอกไปเป็นความถี่ดิจิตอล

ความสัมพันธ์แบบไม่เป็นเชิงเส้นระหว่าง ω และ Ω ในสมการ (3.12) รู้จักกันในนาม "ความถี่การโค้งงอ (Frequency Warping)" โดยสอดคล้องกับผลกระทบของ $H(e^{j\omega})$ ที่สัมพันธ์กับ $H_a(j\Omega)$ ที่ได้จากสมการ (3.17) คือ

$$H(e^{j\omega}) = H_a(j\Omega)\Big|_{\Omega=(2/T)\tan(\omega/2)} \tag{3.19}$$

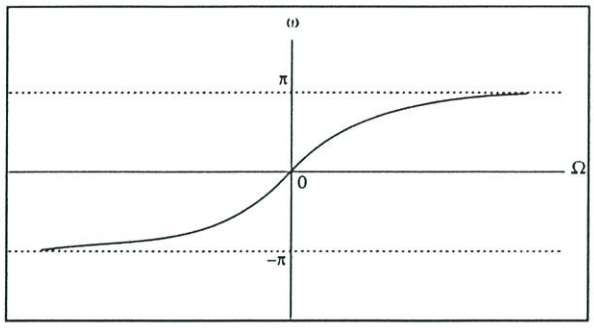
$$\Omega = \frac{2}{T}\tan\left(\frac{\omega}{2}\right) \tag{3.20}$$



รูปที่ 3.9 การแปลงไบลิเนียร์ของ $|H_a(j\Omega)|$ ไปเป็น $|H(e^{j\omega})|$

รูปที่ 3.9 แสดงการถูกบีบอัดในความถี่เนื่องจากการแปลง แต่อีกนัยหนึ่งลักษณะเฉพาะของ $H_a(j\Omega)$ ยังคงรักษาไว้อยู่ใน $H(e^{j\omega})$ โดยเฉพาะ ถ้า $|H_a(j\Omega)|$ มีการกระเพื่อมในแถบผ่านหรือแถบหยุดด้วยลักษณะเหมือนกัน $|H(e^{j\omega})|$ คุณสมบัตินี้เป็นลักษณะสำคัญที่สุดของการแปลงแบบลิเนียร์ และทำให้เป็นการนิยมใช้อย่างแพร่หลายสำหรับการออกแบบ

จะสังเกตได้ว่า การเปลี่ยนโดเมนความถี่อนาลอก Ω ไปเป็นโดเมนความถี่เชิงเลข ω ทางคณิตศาสตร์คือแปลงจากโดเมน s ไปเป็น z ตามสมการที่ (3.19 - 3.20) โดยฟังก์ชันแทนเจนต์ ซึ่งมีลักษณะไม่เป็นเชิงเส้น ทำให้ช่วงความถี่สูงที่ได้จากการแปลงหดแคบลงไป ความถี่สูงมากขึ้นการหดแคบก็มากขึ้น เราเรียกผลนี้ว่าเป็น ปรากฏการณ์หดแคบ (Wrapping effect) ถ้าให้ T=2 ในสมการ (3.20) จะสามารถวาดกราฟความสัมพันธ์ของความถี่อนาลอกและความถี่ดิจิตอลได้ดังรูปที่ 3.10



รูปที่ 3.10 กราฟความสัมพันธ์ของความถี่ดิจิตอลกับความถี่อนาลอกจากการแปลงไบลิเนียร์

3.8 โครงสร้างตัวกรองป้อนกลับเชิงเลขชนิดไบควอดราติก

ตัวกรองป้อนกลับเชิงเลขสามารถแสดงได้หลายแบบ อาจอยู่ในรูปโครงสร้างโดยตรงชนิดที่ 1 หรือ 2, โครงสร้างแบบต่ออนุกรมหรือต่อแบบขนาน, โครงสร้างแบบเศษส่วนต่อเนื่อง เป็นต้น แต่โครงสร้างอีกชนิดหนึ่งซึ่งวิทยานิพนธ์นี้จะใช้อ้างอิงคือ โครงสร้างแบบไบควอดราติก โดยไม่ว่าจะมีรูปแบบโครงสร้างตัวกรองเป็นชนิดไหนก็ตาม ล้วนมาจากพื้นฐานของสมการฟังก์ชันถ่ายโอนที่เป็นสมการผลต่างสืบเนื่อง สมการ(3.21) ทั้งสิ้น

$$y(n) = \sum_{i=0}^{M} a_i\, x(n-i) + \sum_{i=1}^{N} b_i\, y(n-i) \tag{3.21}$$

โดยที่

$x(n)\,;\; y(n)$ เป็นลำดับสัญญาณเข้าและสัญญาณออกตามลำดับ

$a_i\,;\; b_i$ เป็นสัมประสิทธิ์ของลำดับสัญญาณเข้าและออก

เมื่อทำการแปลงแซด ทั้งสองข้างในสมการ(3.21) จะได้ฟังก์ชันถ่ายโอนดังนี้

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{M} a_i z^{-i}}{\sum_{i=0}^{N} b_i z^{-i}} \tag{3.22}$$

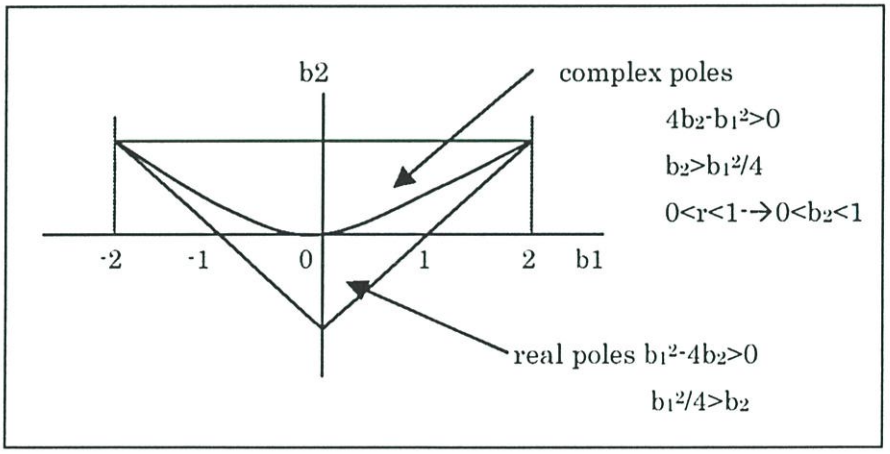
โครงสร้างตัวกรองป้อนกลับเชิงเลขต่างๆที่กล่าวไว้ข้างต้นสามารถสร้างขึ้นได้โดยการจัดรูปสมการ (3.22) ซึ่งในที่นี้จะขอกล่าวเฉพาะเพียงโครงสร้างไบควอดราติก กล่าวคือ ในทางปฏิบัติแล้วการสร้างตัวกรองเชิงเลข เพื่อให้ได้อันดับสูงๆ หรืออันดับที่ N นั้น เรานิยมสร้างตัวกรองอันดับต่ำๆ มาต่อแบบอนุกรมหรือแบบขนานกัน ทั้งนี้จากผลการควบคุมปรากฏการณ์ไม่เป็นเชิงเส้นให้น้อยทำได้ง่ายกว่า ถ้าหากว่าเรามีโครงสร้างของตัวกรองเชิงเลขภาคย่อยที่มีปรากฏการณ์ที่ไม่เป็นเชิงเส้นต่ำ

แล้ว การนำวงจรเหล่านี้มาต่อกันเพื่อให้ได้อันดับที่ N แล้วค่อนข้างแน่ใจได้ว่าจะมีปรากฏการณ์ไม่เป็นเชิงเส้นที่ต่ำด้วย ด้วยเหตุผลนี้จึงได้มีผู้คิดค้นโครงสร้างของตัวกรองเชิงเลขอันดับที่ 2 ที่มีปรากฏการณ์ไม่เป็นเชิงเส้นต่ำขึ้นมาหลายรูปแบบ แต่สำหรับฟังก์ชันไบควอดราติก (Biquadratic Function) จะนิยามได้ดังนี้

$$H(z)=\frac{a_0+a_1z^{-1}+a_2z^{-2}}{1+b_1z^{-1}+b_2z^{-2}} \tag{3.23}$$

ซึ่งเงื่อนไขที่ทำให้ฟังก์ชันนี้มีเสถียรภาพ สามารถพิจารณาได้จากรูปที่ 3.11 และจากสมการจะทราบได้ว่ามีโพล 2 จุดซึ่งจะแทนด้วย p_1 และ p_2 โดยอยู่ตำแหน่งที่

$$p_1=\frac{-b_1+\sqrt{b_1^2-4b_2}}{2} \text{ และ } p_2=\frac{-b_1-\sqrt{b_1^2-4b_2}}{2}$$



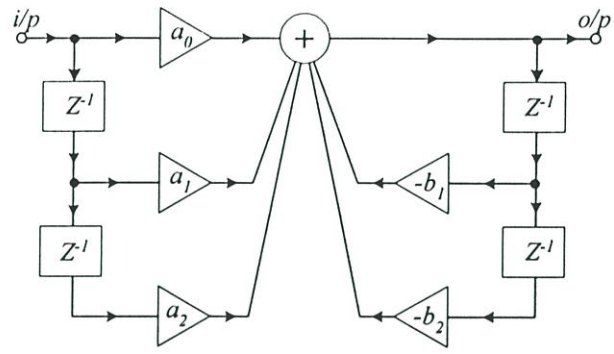
รูปที่ 3.11 แสดงบริเวณตำแหน่งโพลของระบบที่มีเสถียรภาพ

จากรูปที่ 3.11 เราจะได้บริเวณที่ตัวกรองมีเสถียรภาพ (บริเวณ Complex poles) อยู่ในส่วนของสามเหลี่ยมซึ่งเรียกว่า "สามเหลี่ยมเสถียรภาพ" (Stability Triangle) โดยที่มีเงื่อนไขคือ

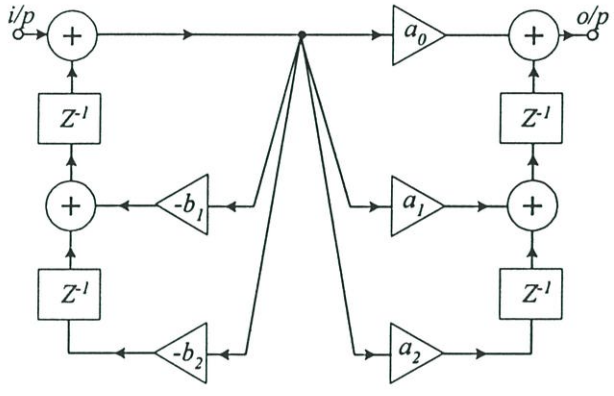
$$0\le|b_2|<1$$

$$|b_1|\le 1+b_2 \tag{3.24}$$

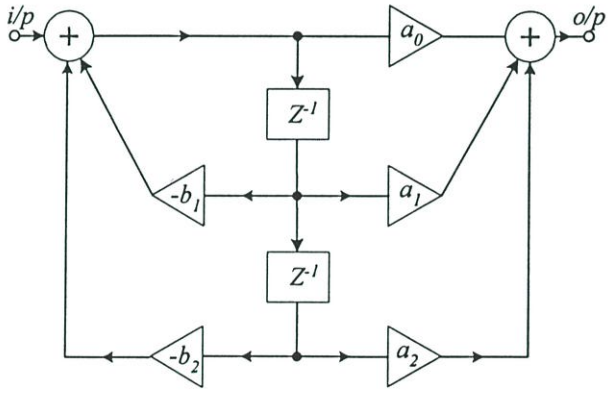
โครงสร้างตัวกรองป้อนกลับแบบไบควอดราติกสามารถแสดงดังรูปที่ 3.12



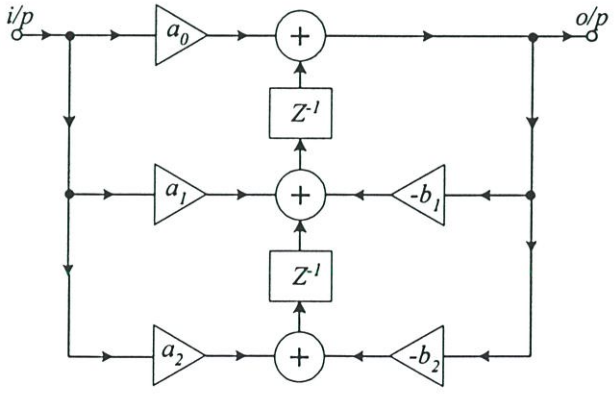
(a)



(b)



(c)



(d)

รูปที่ 3.12 โครงสร้างตัวกรองป้อนกลับเชิงเลขอันดับที่สอง

(a) แบบโดยตรง 1, (b) แบบโดยตรง 1 ทรานสโพส, (c) แบบโดยตรง 2 และ (d) แบบโดยตรง 2 ทรานสโพส

3.9 บทสรุป

เนื้อหาในบทนี้ได้กล่าวถึงความหมายของการประมวลผลสัญญาณทั้งอนาลอกและดิจิตอล ประโยชน์ใช้สอยที่ได้จากการประมวลผลสัญญาณทั้งสองแบบ ซึ่งแสดงไว้ในรูปแบบโครงสร้างที่แตกต่างกันได้หลายแบบ โดยมีขั้นตอนในการดำเนินการกับสัญญาณในการเปลี่ยนแปลงให้สอดคล้องกับระบบนั้นๆ รวมทั้งเงื่อนไขที่ต้องพิจารณา โดยเฉพาะปัญหาเสถียรภาพของระบบที่ทำการออกแบบ โดยได้แสดงไว้ในรูปแบบทฤษฎีต่างๆ รวมถึงความหมายในเชิงเปรียบเทียบของระบบอนาลอกกับระบบดิจิตอล ซึ่งทั้งสองระบบมีความสัมพันธ์สอดคล้องตามนิยามที่ได้แสดงไป เนื้อหาทั้งหมดที่กล่าวมาเป็นเพียงประเด็นสำคัญๆ ที่ควรทราบและใช้พิจารณาในการออกแบบวิจัยระบบตัวกรองเชิงเลข แต่นอกเหนือจากทฤษฎีในการประมวลผลสัญญาณแล้ว รูปแบบโครงสร้างและระเบียบวิธีทางคณิตศาสตร์เชิงฮาร์ดแวร์ ก็มีส่วนสำคัญไม่ยิ่งหย่อนไปกว่ากัน ซึ่งจะได้กล่าวถึงในบทต่อไป

บทที่ 4

ทฤษฎีของตัวดำเนินการทางคณิตศาสตร์

(Theory of arithmetical operation)

4.1 บทนำ

คณิตศาสตร์แบบการกระจาย ได้เริ่มมีกำเนิดขึ้นมากว่า 3 ทศวรรษแล้ว จวบจนถึงปัจจุบันนี้ ก็ยังเป็นเทคนิคที่ยังคงได้รับความนิยม ในการนำไปประยุกต์ใช้งานด้านการออกแบบวงจรรวมดิจิตอล โดยเฉพาะอย่างยิ่งทางด้านการประมวลผลสัญญาณดิจิตอล อะไรจึงเป็นเหตุผลที่ทำให้ทฤษฎีนี้ยังคงแพร่หลายอยู่นั้น ในบทนี้ของวิทยานิพนธ์จะได้มีการกล่าวถึง ซึ่งจะอธิบายเท้าความตั้งแต่รูปแบบชนิดของระบบจำนวนเลขฐานสอง, ความเป็นมาของการดำเนิน การทางคณิตศาสตร์ซึ่งอยู่ในรูปแบบผลรวมของผลคูณเลขจำนวนฐานสอง (Sum of Products: SOP), กระบวนการคูณและสะสม (Multiply and Accumulate: MAC), การเปลี่ยนรูปแบบ MAC ไปสู่รูปแบบคณิตศาสตร์แบบการกระจาย (Distributed Arithmetic: DA) จนกระทั่งการแปลงสมการทางคณิตศาสตร์ไปเป็นวงจรดิจิตอลในระดับลอจิก ซึ่งวิวัฒนาการต่างๆจะได้อธิบายตามลำดับดังรายละเอียดต่อจากนี้

4.2 ระบบจำนวนเลขฐานสอง

ถ้าหากกล่าวถึงการแบ่งระบบจำนวนเลขฐานสอง ที่เรามักได้พบเห็นกันจะอยู่ในระบบดิจิตอลคอมพิวเตอร์แล้ว ส ามารถที่จะแบ่งระบบเลขฐานสองได้มากมายหลายชนิดด้วยกัน ได้สรุปไว้ในตารางที่ 4.1 โดยแต่ละชนิดต่างก็มีข้อดี-ข้อเสีย ที่แตกต่างกันออกไป แต่ด้วยคุณสมบัติพิเศษต่างๆ ที่เราพบในระบบเลขฐานสองของจำนวนทศนิยมก็คือจะคิดเครื่องหมายแบบส่วนเติมเต็มสอง (Signed Two's Complement) ซึ่งจะพบว่าสะดวกและเหมาะสม ต่อการคำนวณและการออกแบบวงจรในระบบดิจิตอล อีกทั้งในระบบการประมวลผลสัญญาณดิจิตอลที่ต้องการแทนรูปแบบสัญญาณดิจิตอลที่คิดขนาดเครื่องหมายของสัญญาณ หรือการคำนวณทางคณิตศาสตร์ของเลขฐานสองแบบคิดเครื่องหมาย ก็ได้มีการสร้างอุปกรณ์สำหรับการแปลงจากสัญญาณอนาลอกไปเป็นดิจิตอลที่สนับสนุนการคำนวณในรูปแบบนี้อยู่เป็นจำนวนมาก จึงทำให้ระบบจำนวนชนิดนี้เหมาะกับการนำมาประยุกต์ใช้งานทางการประมวลผลสัญญาณดิจิตอล และการออกแบบกับคณิตศาสตร์แบบการกระจายได้

ตารางที่ 4.1 สรุปข้อดีข้อเสียของระบบจำนวนเลขฐานสองชนิดต่างๆ

ระบบจำนวน	ช่วงของจำนวน	ข้อดี	ข้อเสีย
จำนวนเต็มแบบไม่กำหนดเครื่องหมาย (Unsigned Integer)	0 ถึง 2^N-1	เป็นระบบจำนวนสากล ง่ายต่อการกระทำทางเลขคณิต เช่น การบวก การลบ	ไม่สามารถเก็บค่าจำนวนลบได้
ส่วนเติมเต็มสอง (Two's complement)	$-2^{(N-1)}$ ถึง $2^{(N-1)}$ -1	เก็บค่าจำนวนทั้งบวกและลบ ง่ายต่อการกระทำทางเลขคณิตด้วยการบวกแบบปกติ	ต้องใช้บิตพิเศษในการเก็บค่าจำนวนบวก
ทศนิยมแบบไม่กำหนดเครื่องหมาย (Unsigned Fraction)	0 ถึง $2^N - 2^M$	เก็บค่าจำนวนบวกที่มีค่ามากกว่า 1 และน้อยกว่า1 การกระทำทางเลขคณิตคล้ายกันกับ ระบบจำนวนเต็มแบบไม่กำหนดเครื่องหมาย	ไม่สามารถเก็บค่าจำนวนลบได้
ทศนิยมแบบส่วนเติมเต็มสอง (Signed Two's complement Fraction)	$-2^{(N-1)}$ ถึง $2^{(N-1)}$ -2^{-M} ใน 2^{-M} ขั้น	เก็บค่าจำนวนบวกและลบทั้งมากกว่าและน้อยกว่า 1 การกระทำทางเลขคณิตคล้ายกันกับ2's complement	-
รหัสเกรย์ (Gray Code)	0 ถึง $2^{(N-1)}$	มีเพียง 1 บิต ที่เปลี่ยนแปลงค่าจำนวน ซึ่งสะดวกต่อการเชื่อมต่อกับระบบกายภาพ	ยากต่อการกระทำทางเลขคณิต
ขนาดที่กำหนดเครื่องหมาย (Signed Magnitude)	$-2^{(N-1)} - 1$ ถึง $2^{(N-1)}$ -1	ประโยชน์ต่อการใช้งานที่ต้องการขนาดที่มีเครื่องหมายต่างกัน	ยากต่อการกระทำทางเลขคณิต (แม้ว่าจะง่ายกว่าแบบรหัสเกรย์)
ชดเชยส่วนเติมเต็มสอง (Offset Two's Complement)	$-2^{(N-1)}$ ถึง $2^{(N-1)}$ -1	ถูกใช้ใน A/D และ D/A ง่ายต่อการกระทำทางเลขคณิต	-
ส่วนเติมเต็มหนึ่ง (One's Complement)	$-2^{(N-1)} - 1$ ถึง $2^{(N-1)}$ -1	ง่ายต่อการทำการลบเลข	ลำบากต่อการกระทำทางเลขคณิตที่ไม่ใช่การลบเลข
โฟลทติ้งพอยท์ (Floating Point)		ช่วงการทำงานพลวัตกว้างมาก	ต้องการอุปกรณ์จำนวนมาก
บล็อกโฟลทติ้งพอยท์ (Block Floating Point)		ช่วงการทำงานพลวัตกว้างและต้องการจำนวนอุปกรณ์น้อย	ทุกจำนวนจะมีจำนวนเลขยกกำลังเท่ากันทุกค่าเวลา

4.2.1 การแปลงรหัส (Code Conversion)

การแปลงรหัสอาจมีความจำเป็นต้องกระทำหลังจากการแปลงที่ได้จากตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล (ADC) หรือก่อนการแปลงจากตัวแปลงสัญญาณดิจิตอลเป็นอนาลอก (DAC) ในกรณีที่ต้องการทำการแปลงอาจจะใช้ตัวแปลงเฉพาะ (Converter) ซึ่งให้ผลลัพธ์ที่ดี แต่เพื่อความประหยัดในการทำการนี้สามารถอาศัยตารางความสัมพันธ์ที่ใช้ทำการแปลงรหัสดังตารางที่ 4.2 ซึ่งเป็นการอธิบายถึงความสัมพันธ์และวิธีการแปลงรูปแบบรหัส โดยหลักการสำคัญจะใช้การคอมพลีเมนที่บิตนัยสำคัญสูงสุด (MSB)

ตารางที่ 4.2 ความสัมพันธ์ในการแปลงรหัสแบบไบโพลาร์

การแปลงรหัส จาก⇩ เป็น ⇨	**ขนาดที่กำหนดเครื่องหมาย (Signed Magnitude)**	**ส่วนเติมเต็มสอง (2's Complement)**	**ชดเชยเลขฐาน 2 (Offset Binary)**	**ส่วนเติมเต็มหนึ่ง (1's Complement)**
ขนาดที่กำหนดเครื่องหมาย (Signed Magnitude)	ไม่เปลี่ยนแปลง	ถ้า MSB = 1 คอมพลีเมนบิตอื่นๆ แล้วบวกด้วย 00...01	คอมพลีเมน MSB ถ้า MSB ใหม่ =1 คอมพลีเมนบิตอื่นๆแล้วบวกด้วย 00...01	ถ้า MSB = 1 คอมพลีเมนบิตอื่นๆ
ส่วนเติมเต็มสอง (2's Complement)	ถ้า MSB = 1 คอมพลีเมนบิตอื่นๆ แล้วบวกด้วย 00...01	ไม่เปลี่ยนแปลง	คอมพลีเมน MSB	ถ้า MSB = 1 บวกด้วย 00...01
ชดเชยเลขฐาน 2 (Offset Binary)	คอมพลีเมน MSB ถ้า MSB ใหม่ =0 คอมพลีเมนบิตอื่นๆแล้วบวกด้วย 00...01	คอมพลีเมน MSB	ไม่เปลี่ยนแปลง	คอมพลีเมน MSB ถ้า MSB ใหม่ =0 บวกด้วย 00...01
ส่วนเติมเต็มหนึ่ง (1's Complement)	ถ้า MSB = 1 คอมพลีเมนบิตอื่นๆ	ถ้า MSB = 1 บวกด้วย 11...11	คอมพลีเมน MSB ถ้า MSB ใหม่ =1 บวกด้วย 11...11	ไม่เปลี่ยนแปลง

4.2.2 ระบบเลขส่วนเติมเต็มสอง (Two's Complement)

ในระบบเลขส่วนเติมเต็มสอง จะสามารถนิยามได้ดังสมการที่ (4.1)

$$\bar{x} = \begin{cases} x & x \geq 0 \\ 2 - |x| & x < 0 \end{cases} \tag{4.1}$$

โดยที่ $\bar{x}$ คือ ส่วนเติมเต็มของเลขจำนวน x

x คือ จำนวนเลขที่เป็นเศษส่วน (Fractional Number)

$$x = \pm 0.\, b_{-1}\, b_{-2}\, b_{-3} \ldots b_{-m} \tag{4.2}$$

ซึ่งในระบบเลขส่วนเติมเต็มสอง จะใช้บิตที่มีนัยสำคัญสูงสุดเป็นบิตเครื่องหมายถ้าเป็นบวกแทนด้วย "0" ส่วนลบแทนด้วย "1" ถ้าให้ x แทนด้วยเลขฐานสองจำนวน **B** บิต ดังนั้น รูปแบบของเลขส่วนเติมเต็มสอง อาจเขียนแทนด้วยรูปแบบทางคณิตศาสตร์ดังนี้

$$\bar{x} = x_0 . x_1\, x_2\, x_3 \ldots x_B \tag{4.3}$$

ค่าของ $\bar{x}$ หาได้จากสมการต่อไปนี้

$$x = -x_0 + \sum_{i=1}^{B} x_i 2^{-i} \tag{4.4}$$

โดยที่ $i = 1, 2, 3, \ldots, B$

<u>ตัวอย่าง</u> การแปลงค่าจำนวน -0.375 ไปเป็นเลขส่วนเติมเต็มสอง

จากนิยาม $-0.375 < 0$ ดังนั้นพิจารณาใช้ $2 - |x|$

ในที่นี้ $x = -0.375$ ฉะนั้น $|x| = 0.375$

$2 - 0.375 = 1.625$ จะได้ค่าไบนารี่คือ 1.101

ตรวจสอบค่าที่แปลงได้โดยการแทนค่าไบนารี่กลับเข้าในสูตร จะได้

$$-1 + \left(1 \times 2^{-1}\right) + \left(0 \times 2^{-2}\right) + \left(1 \times 2^{-3}\right) = -1 + 0.5 + 0.125 = -0.375$$

จากสมการการแปลงเลขส่วนเติมเต็มสอง จะได้จำนวนที่มีค่ามากที่สุด คือ $1 - 2^{-B+1}$ (0111...1) และจำนวนที่มีค่าน้อยที่สุด คือ -1 (1000...0) นอกจากนั้นค่า 0 แทนด้วยค่าเดียวคือ (000...0) และมีค่าเป็นจำนวนลบมากกว่าค่าที่เป็นจำนวนบวกอยู่หนึ่งจำนวน คือ (-1)

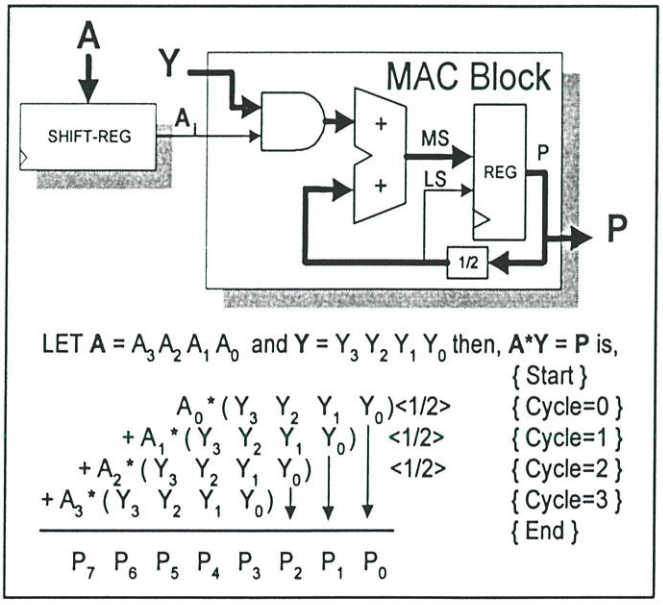
4.3 รายละเอียดการทำงานของคณิตศาสตร์แบบการกระจาย

คณิตศาสตร์แบบการกระจาย หรือ (Distributed Arithmetic: DA) ได้ถูกนำเสนอขึ้นเป็นครั้งแรกในปี ค.ศ. 1971 โดย Croisier และคณะร่วมงาน ซึ่งภายหลังได้ถูกประยุกต์ใช้งานในการสร้างตัวกรองสัญญาณดิจิตอลโดย Abraham Peled และ Bede Liu [1]โดยรูปแบบการคำนวณของ DA เป็นรูปแบบที่สอดคล้องกับการคำนวณทางด้าน DSP อีกทั้งยังสามารถแปลงไปสู่รูปแบบวงจรดิจิตอลลอจิกได้ จึงทำให้มีความแพร่หลายอย่างมากจนถึงปัจจุบัน โดยพื้นฐานของคณิตศาสตร์แบบการกระจายแล้ว จะเริ่มต้นที่การพิจารณาจากสมการผลบวกของผลคูณ (Sum of Product: SOP) ซึ่งเป็นการนำผลคูณย่อย [2] (Partial Products) ตั้งแต่สองผลคูณเป็นต้นไปมาทำการบวกกัน ถ้ากำหนดให้ Y เป็นข้อมูลเวกเตอร์ขนาดความยาว N ซึ่งทำการคูณอยู่กับค่าสัมประสิทธิ์เวกเตอร์ A ขนาดความยาว N เช่นเดียวกัน จะได้ผลลัพธ์ P มีค่าดังนี้

$$P = \sum_{i=1}^{N} A_i Y_i \tag{4.5}$$

4.3.1 การคูณค่าและการบวกสะสมค่าผลคูณ

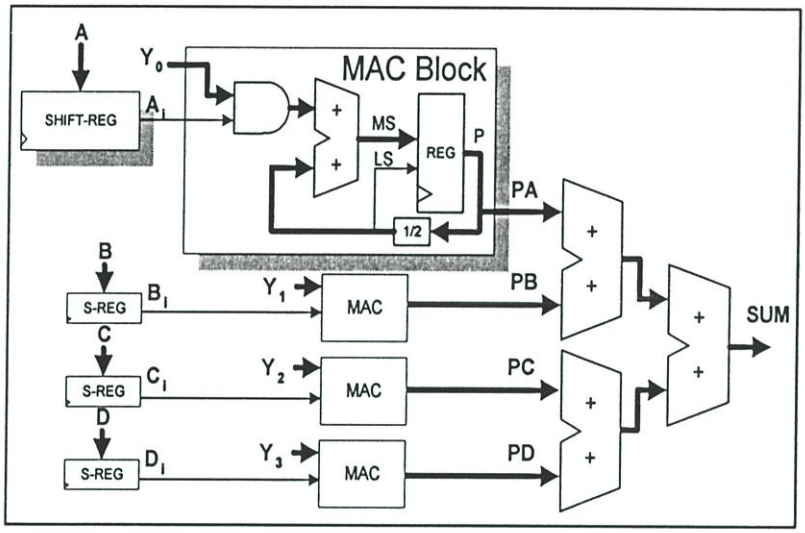
ถ้าพิจารณาสมการที่ (4.5) เฉพาะค่าผลคูณย่อยของ $A_i Y_i$ โดยแทน A_i และ Y_i เป็นข้อมูลเลขฐานสองขนาด 4 บิต แสดงตามรูปที่ 4.1 ผลคูณ $A_i Y_i$ แบบเลขฐานสองจะเป็นการคูณค่าแล้วทำการบวกสะสมค่าผลคูณ ที่ได้ทั้ง 4 ค่า (Multiply and Accumulate: MAC) ซึ่งจะถูกเรียกเป็นบล็อก MAC [5]



รูปที่ 4.1 MAC ขนาด 4 บิต คูณค่าโดยการใช้เทคนิค การเลื่อนและบวก (shift-and-add)

MAC จะทำการหาผลคูณย่อยโดยการคูณครั้งละบิตของค่าสัมประสิทธิ์ Y ด้วยข้อมูล A ด้วยการ and กัน เทคนิคนี้จะทำการบวกค่าผลคูณย่อยเหล่านี้เข้าด้วยกัน โดยแอกคิวมูเลเตอร์ซึ่งจะทำการเลื่อนบิตไปทางขวา 1 บิต เป็นการทำการหารด้วย 2 (แทนด้วยสัญลักษณ์ 1/2) ในแต่ละรอบของสัญญาณนาฬิกาซึ่งเป็นการชดเชยสำหรับการถ่วงน้ำหนักบิตของการเข้ามาของผลคูณย่อย

ถ้าให้สมการ SOP ของผลคูณย่อย 4 ค่า มีค่าผลลัพธ์ $\mathrm{SUM} = AY_0 + BY_1 + CY_2 + DY_3$ แสดงได้ดังรูปที่ 4.2

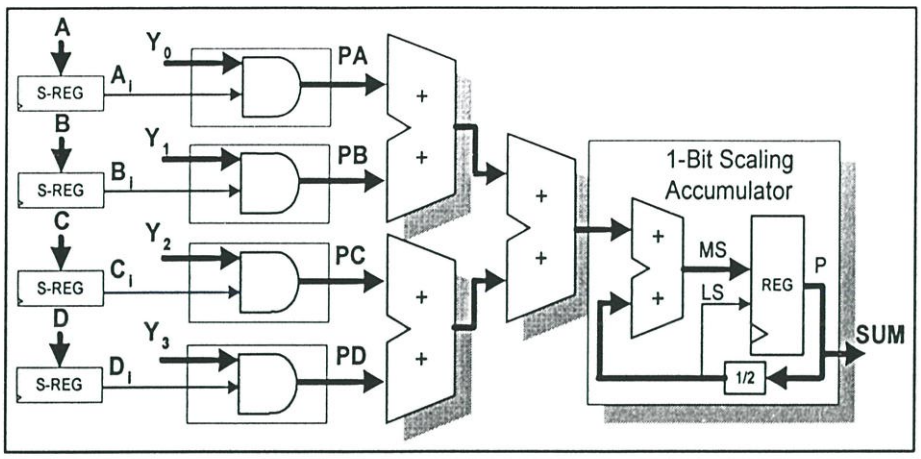


รูปที่ 4.2 ผลลัพธ์จากการรวมผลคูณย่อยจาก MAC ทั้ง 4 ชุด

ตัวคูณ 4 ตัวจะถูกทำตามลำดับ และผลลัพธ์จะถูกรวมเข้าด้วยกัน เมื่อแต่ละตัวคูณเสร็จ การทำงานนี้จะใช้จำนวนรอบสัญญาณนาฬิกา n รอบ สำหรับข้อมูลขนาด n บิต ดังนั้นอัตราของสัญญาณนาฬิกาจึงมีค่าเท่ากับอัตราของข้อมูลหารด้วยจำนวนบิต โดยที่ระหว่างแต่ละรอบสัญญาณนาฬิกาของข้อมูลตัวคูณ 4 ตัวจะให้เทอมของผลลัพธ์ออกมา (ในรูป PA, PB, PC และ PD) ทั้ง 4 ตัวจะถูกรวมเข้าด้วยกันเป็นผลลัพธ์เอาท์พุท จะเห็นได้ว่า ถ้าจำนวนเทอมของผลคูณย่อยที่มีปริมาณมากขึ้น จะทำให้ต้องเพิ่มจำนวนของแอกคิวมูเลเตอร์ขึ้นด้วย ทำให้ขนาดวงจรมีขนาดใหญ่ขึ้น

4.3.2 แนวทางลดขนาดอุปกรณ์ของ MAC ด้วย DA [5]

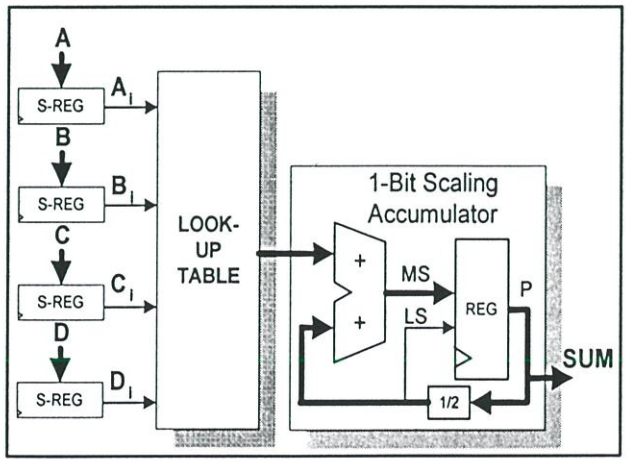
สำหรับคณิตศาสตร์แบบการกระจาย หรือ DA นั้นต่างไปจากวิธีการทำงานของ MAC โดยจะทำการบวกค่าของผลคูณย่อยก่อน หลังจากนั้นก็จะทำการชดเชยถ่วงน้ำหนักบิตของการบวกสะสม สามารถแสดงโครงสร้างการใช้ DA ได้ดังรูปที่ 4.3



รูปที่ 4.3 ผลลัพธ์จากการรวมผลคูณย่อยจาก Serial Distributed Arithmetic แทน MAC ทั้ง 4 ชุด

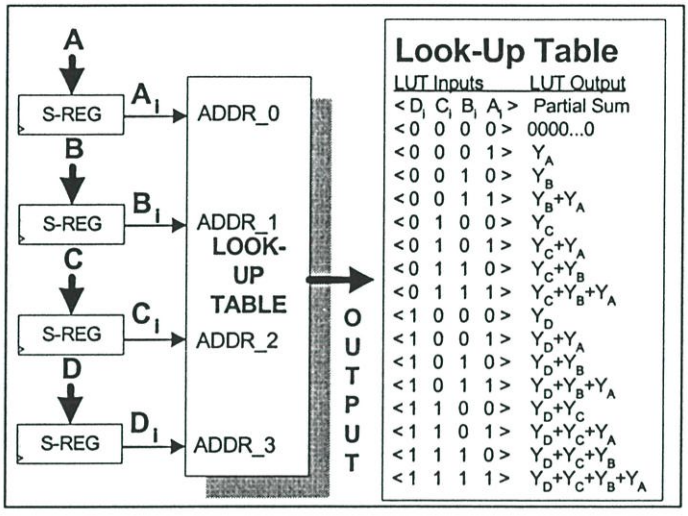
การทำงานจะเป็นการเรียงลำดับใหม่ เทคนิคนี้จะช่วยลดจำนวนวงจรเลื่อนบิตและบวก (Shift & Add) เป็นเพียงชุดเดียว แต่จำนวนของตัวบวกยังคงเท่าเดิม

ถ้าเปรียบเทียบเทอมสัมประสิทธิ์ของสมการ SOP เป็นสัมประสิทธิ์คงที่ในตัวกรองเชิงเลข แล้ว จะได้เอาท์พุทของฟังก์ชัน and และตัวบวกทั้ง 3 ตัว จะขึ้นอยู่กับเพียงบิตอินพุททั้ง 4 จากชีฟรีจีสเตอร์ ซึ่งเราสามารถแทนฟังก์ชัน and และตัวบวกทั้ง 3 ตัวเป็นตารางดูค่า (look up table: LUT) ขนาด 4 bit ง่ายๆ ที่ให้ผลลัพธ์ที่มีขนาดเล็กลง เรียกรูปแบบนี้ว่า คณิตศาสตร์แบบการกระจายอย่างอนุกรม (bit-Serial Distributed Arithmetic MAC หรือ SDA-MAC) แสดงดังรูปที่ 4.4



รูปที่ 4.4 โครงสร้าง SDA-MAC แบบ LUT

ข้อมูลของ LUT ที่อ้างตามรูปที่ 4.5 จะประกอบด้วยผลคูณย่อยของสัมประสิทธิ์ ($Y_0Y_1Y_2$และ Y_3) LSB (output จากแต่ละ serial shift register) ของข้อมูลทั้ง 4 จะอ้างอิงตำแหน่งใน LUT ถ้าทั้ง 4 บิต เป็น "1" output จาก LUT ก็จะสอดคล้องกับสัมประสิทธิ์ทั้ง 4 ค่า ถ้าบิตใดบิตหนึ่งเป็น "0" บิตที่เป็น "0" ก็จะไม่มี เพราะว่า address ของ LUT จะประกอบด้วยค่าที่เป็นได้เพียง 0 กับ 1 ภายใต้เงื่อนไขนี้ LUT จะมีผลรวมที่เป็นไปได้ 16 ค่า ของค่าสัมประสิทธิ์



รูปที่ 4.5 ค่าของ LUT ที่สอดคล้องกับสัมประสิทธิ์ทั้ง 16 ค่า

จากตัวอย่างของ 4 MAC ขนาดความกว้างของ LUT โดยทั่วไปจะใหญ่กว่า 2 บิต ของความกว้างสัมประสิทธิ์ ในการบวกเลข 2 บิต อาจทำให้เวิร์ดข้อมูลขยายขึ้นจากการบวกทดค่าของสัมประสิทธิ์ทั้ง 4 ได้ วงจรจึงต้องการอย่างน้อย 2 บิต เพราะวงจรจะรวมค่าสัมประสิทธิ์ 4 ตัว จำนวนบิตที่น้อยกว่าอาจเพียงพอในกรณีซึ่งการรวมของผลลัพธ์สัมประสิทธิ์น้อยกว่าขนาดเวิร์ดข้อมูลที่ขยายขึ้น ถ้าสัมประสิทธิ์ มากกว่า 4 ตัวก็จะต้องใช้ขนาดเวิร์ดข้อมูลที่กว้างกว่า โดยปกติแล้วจำนวนตัวบวกควรจะมีค่าอย่างน้อยเป็น $\log_2$(ของจำนวนของสัมประสิทธิ์) เราอาจใช้จำนวนบิตให้เพียงพอกับสัมประสิทธิ์เฉพาะค่าที่รู้และ LUT สามารถคำนวณออกมาได้

จากที่จำนวนของสัมประสิทธิ์ เพิ่มขึ้นขนาดจำนวนแอดเดรสของ LUT ก็จะโตขึ้น อย่างฟังก์ชันเอ็กโพเนนเชียล (มีค่าเป็น 2^n; เมื่อ n เป็นจำนวนสัมประสิทธิ์) ขนาด LUT ที่ใหญ่ขึ้นสามารถเลี่ยงได้โดยการแบ่งวงจรเป็นกลุ่มย่อยเล็กๆ และรวมเอาท์พุทของ LUT โดยตัวบวก เพราะการใช้ตัวบวก จะประหยัดขนาดทางฮาร์ดแวร์กว่าการใช้ LUT ที่มีขนาดแอดเดรสมากๆ เทคนิคนี้จะเรียกเป็น รูปแบบคณิตศาสตร์แบบกระจายอย่างขนาน (Parallel Distributed Arithmetic: PDA) ซึ่งในที่นี้จะไม่ขอกล่าวถึง

4.4 การประยุกต์ใช้ DA กับตัวกรองสัญญาณเชิงเลข [7]

จากที่กล่าวไว้ในตอนต้นว่า ระบบสมการการประมวลผลสัญญาณดิจิตอลจะมีรูปแบบทางพีชคณิตที่สองคล้องกับคณิตศาสตร์แบบการกระจาย โดยเนื้อหาส่วนนี้จะแยกกล่าวแต่เฉพาะระบบสมการของตัวกรองสัญญาณดิจิตอลซึ่งแบ่งออกเป็นสองชนิดหลักคือ ตัวกรองไม่ป้อนกลับเชิงเลข และตัวกรองป้อนกลับเชิงเลข ซึ่งภายในงานวิจัยนั้นจะเน้นหนักไปที่การออกแบบตัวกรองป้อนกลับเชิงเลข อันดับที่ 2 ซึ่งจะได้กล่าวต่อไปในหัวข้อของการออกแบบตัวกรอง

4.4.1 ตัวกรองไม่ป้อนกลับเชิงเลข (FIR Filter)

แทนสมการความสัมพันธ์ ของตัวกรองไม่ป้อนกลับเชิงเลข คือ

$$Y = \sum_{k=1}^{K} A_k X_k \tag{4.6}$$

สมการที่ (4.6) ให้ A_k เป็นค่าสัมประสิทธิ์, X_k เป็นข้อมูลอินพุทซึ่ง X_k เป็นตัวเลขส่วนเติมเต็มสอง (Two's complement) และให้ค่าสัมบูรณ์ของ $X_k < 1$ ดังนั้นแสดงค่าของ X_k ในเทอมของบิตได้

$$X_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} \cdot 2^{-n} \tag{4.7}$$

เมื่อ b_{kn} คือ บิตต่างๆภายในเวิร์ด X_k จะมีค่าเป็น 0 หรือ 1

b_{k0} คือ บิตเครื่องหมาย

b_{kN-1} คือ บิตนัยสำคัญต่ำสุด (LSB)

N คือ จำนวนบิตข้อมูลอินพุท

เมื่อแทนค่า X_k จากสมการที่ (4.7) ลงในสมการที่ (4.6) ได้เป็นสมการที่ (4.8) แสดงค่า Y ในเทอมของบิตของ X_k ดังนี้

$$Y = \sum_{k=1}^{K} A_k \left(-b_{k0} + \sum_{n=1}^{N-1} b_{kn} \cdot 2^{-n} \right) \tag{4.8}$$

สมการที่ (4.8) จัดรูปใหม่จะได้

$$Y = \sum_{n=1}^{N-1}\left(\sum_{k=1}^{K} A_k b_{kn}\right)2^{-n} + \sum_{k=1}^{K} A_k(-b_{k0}) \tag{4.9}$$

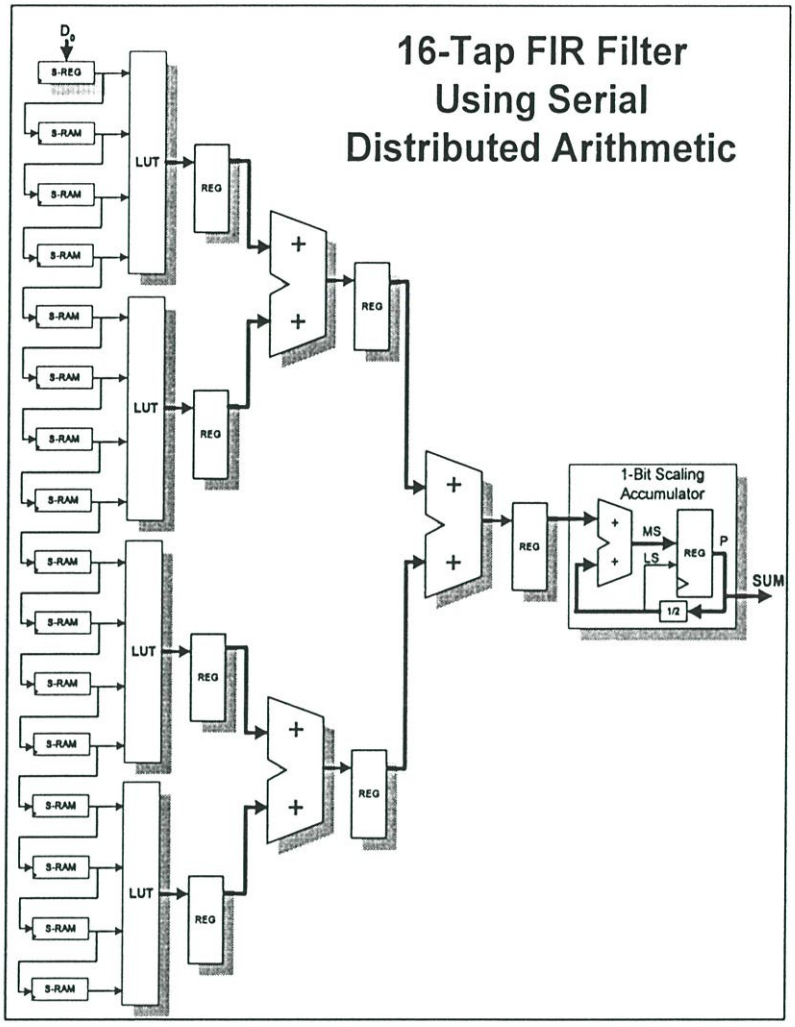
$$\begin{aligned}
Y = {} & -2^{0}(b_{10}A_1 + b_{20}A_2 + b_{30}A_3 + \ldots + b_{k0}A_k) \\
& +2^{-1}(b_{11}A_1 + b_{21}A_2 + b_{31}A_3 + \ldots + b_{k1}A_k) \\
& +2^{-2}(b_{12}A_1 + b_{22}A_2 + b_{32}A_3 + \ldots + b_{k2}A_k) \\
& +2^{-3}(b_{13}A_1 + b_{23}A_2 + b_{33}A_3 + \ldots + b_{k3}A_k) \\
& +2^{-4}(b_{14}A_1 + b_{24}A_2 + b_{34}A_3 + \ldots + b_{k4}A_k) \\
& \vdots \\
& +2^{-(N-1)}(b_{1,N-1}A_1 + b_{2,N-1}A_2 + b_{3,N-1}A_3 + \ldots + b_{k,N-1}A_k)
\end{aligned} \tag{4.10}$$

สมการที่ (4.10) เป็นการกระจายทางคณิตศาสตร์ของค่าสัมประสิทธิ์กับบิตต่างๆ ของข้อมูลอินพุท ด้วยเหตุนี้จึงได้ชื่อเรียกว่า **Distributed Arithmetic (DA)** พิจารณาวงเล็บใหญ่ของสมการที่ (4.9) คือ

$$Y' = \sum_{k=1}^{K} A_k b_{kn} \tag{4.11}$$

เพราะว่า b_{kn} จะมีค่าเป็น 0 หรือ 1 เท่านั้น ดังนั้นผลลัพธ์สมการ (4.11) จะมีค่าที่เป็นไปได้ทั้งหมด 2^K ค่า ซึ่งค่าต่างๆเหล่านี้ถูกคำนวณแล้วเก็บไว้ในหน่วยความจำแบบอ่านได้อย่างเดียวหรือรอม (ROM) ข้อมูลบิตของอินพุทถูกใช้เป็นแอดเดรส (Address) ของรอมได้โดยตรง ผลลัพธ์ถูกส่งไปเก็บไว้ในตัวรวมข้อมูลหรือแอกคิวมูเลเตอร์ (Accumulator) หลังจากที่มีการส่งข้อมูลอินพุทที่ทำเป็นแอดเดรสจำนวน n รอบ ก็จะได้ผลลัพธ์ Y

ตัวอย่างของตัวกรองสัญญาณดิจิตอลแบบ FIR อันดับที่ 16 โดยใช้ LUT-SDA สามารถสร้างขึ้นได้โดยการต่อส่วนของรีจีสเตอร์, Look-up Table, Adder เพิ่มเข้ามาก็จะทำให้สร้างตัวกรองเลียนแบบสมการ FIR อันดับที่ 16 ได้ แต่มีข้อสังเกตว่า ถ้ายิ่งอันดับตัวกรอง FIR สูงมากๆ จะทำให้ระบบใช้จำนวนอุปกรณ์ดังกล่าวสิ้นเปลือง อีกทั้งการคำนวณค่าใน LUT ก็จะมีหลายชุด ทำให้ยากต่อการออกแบบระบบยิ่งขึ้น



รูปที่ 4.6 ตัวกรอง FIR อันดับที่ 16 โดยใช้ LUT-SDA

4.4.2 ตัวกรองป้อนกลับเชิงเลข (IIR Filter)

สมการความสัมพันธ์ของตัวกรองป้อนกลับเชิงเลขสามารถแสดงได้ดังนี้

$$Y = \sum_{k=0}^{M} a_k X_k - \sum_{k=1}^{N} b_k Y_k \tag{4.12}$$

กำหนดให้ค่าอินพุทและเอาท์พุทของข้อมูล ด้วยเลขแบบส่วนเติมเต็มสอง แสดงได้ดังนี้

$$X_k = -x_{k0} + \sum_{n=1}^{N-1} x_{kn} \cdot 2^{-n} \qquad \text{และ} \qquad Y_k = -y_{k0} + \sum_{n=1}^{N-1} y_{kn} \cdot 2^{-n} \tag{4.13}$$

แทนค่าสมการที่ (4.13) ลงในสมการที่ (4.12) จะได้ดังนี้

$$Y=\sum_{k=0}^{M}a_k\left(-x_{k0}+\sum_{n=1}^{N-1}x_{kn}2^{-n}\right)-\sum_{k=1}^{N}b_k\left(-y_{k0}+\sum_{n=1}^{N-1}y_{kn}2^{-n}\right) \tag{4.14}$$

$$=\left[\sum_{k=1}^{N}b_k y_{k0}-\sum_{k=0}^{M}a_k x_{k0}\right]+\left[\sum_{n=1}^{N-1}\left(\sum_{k=0}^{M}a_k x_{kn}\right)2^{-n}-\sum_{n=1}^{N-1}\left(\sum_{k=0}^{M}b_k y_{kn}\right)2^{-n}\right] \tag{4.15}$$

$$\begin{aligned}Y=&-2^0(a_0X_{00}+a_1X_{10}+a_2X_{20}+\ldots+a_MX_{M0}-b_1Y_{10}-b_2Y_{20}-b_3Y_{30}-\ldots-b_NY_{N0})\\&+2^1(a_0X_{01}+a_1X_{11}+a_2X_{21}+\ldots+a_MX_{M1}-b_1Y_{11}-b_2Y_{21}-b_3Y_{31}-\ldots-b_NY_{N1})\\&+2^2(a_0X_{02}+a_1X_{12}+a_2X_{22}+\ldots+a_MX_{M2}-b_1Y_{12}-b_2Y_{22}-b_3Y_{32}-\ldots-b_NY_{N2})\\&+2^3(a_0X_{03}+a_1X_{13}+a_2X_{23}+\ldots+a_MX_{M3}-b_1Y_{13}-b_2Y_{23}-b_3Y_{33}-\ldots-b_NY_{N3})\\&\vdots\\&+2^{-(N-1)}(a_0X_{(N-1),0}+a_1X_{(N-1),1}+a_2X_{(N-1),2}+\ldots+a_MX_{(N-1),M}\\&-b_1Y_{(N-1),1}-b_2Y_{(N-1),2}-b_3Y_{(N-1),3}-\ldots-b_NY_{(N-1),N})\end{aligned} \tag{4.16}$$

จากสมการจะเห็นว่า ได้โครงสร้างที่คล้ายกันกับกรณีของตัวกรองไม่ป้อนกลับ แต่จะใช้สัญญาณเอาท์พุท ที่ทำการเลื่อนค่าเพื่อการอ้างอิงตำแหน่งหน่วยความจำได้เช่นกัน โดยใช้สัญญาณอ้างอิงตำแหน่งข้อมูลที่ ROM จำนวน N+M+1 เส้น และขนาดหน่วยความจำเท่ากับ 2^{N+M+1} words

4.5 โครงสร้างของวงจรกรองป้อนกลับเชิงเลขชนิดไบควอดโดยวิธีคณิตศาสตร์แบบการกระจาย [2]

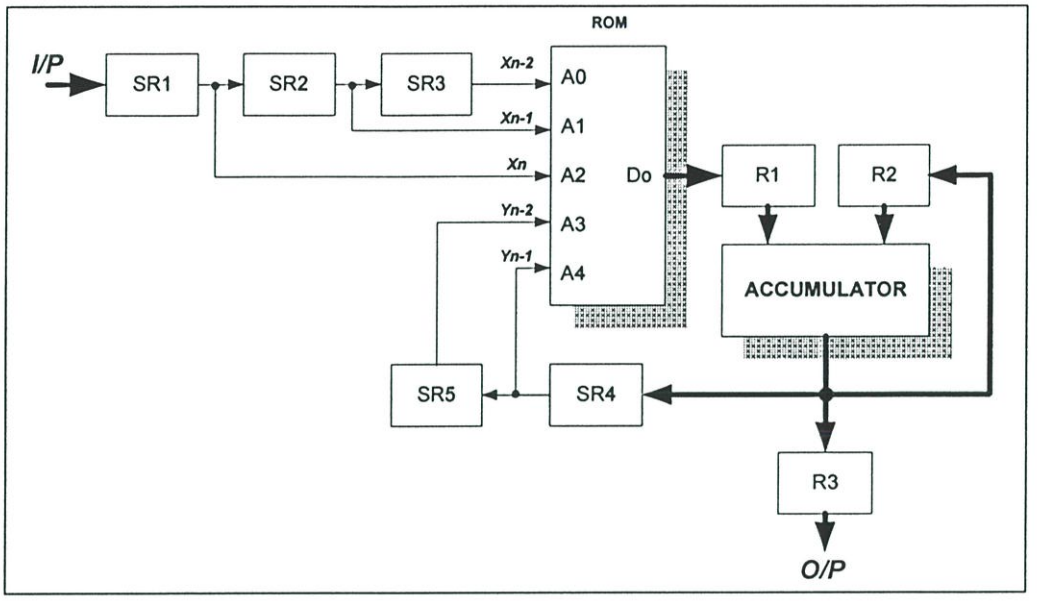
สำหรับงานวิจัยในวิทยานิพนธ์นี้ จะเป็นการสร้างวงจรกรองความถี่ดิจิตอลแบบไบควอดราติก (Biquadratic) ซึ่งมีทรานเฟอร์ฟังก์ชั่นดังนี้

$$H(z)=\frac{\sum_{k=0}^{2}A_kZ^{-k}}{1+\sum_{k=1}^{2}B_kZ^{-k}} \tag{4.17}$$

ทำการกระจายเทอมผลรวมของผลคูณ ในสมการที่ (4.17) ได้เป็นสมการ (4.18)

$$\frac{Y(z)}{X(z)}=\frac{A_0+A_1Z^{-1}+A_2Z^{-2}}{1+B_1Z^{-1}+B_2Z^{-2}} \tag{4.18}$$

ค่า โพล (Pole) จะถูกกำหนดโดยค่าสัมประสิทธิ์ B_1 และ B_2 ส่วนซีโร่ (Zero) คือค่า A_0 , A_1 , A_2 เป็นตัวกำหนด แสดงในรูปของตัวแปรเวลาได้ว่า $Y_n = A_0X_n + A_1X_{n-1} + A_2X_{n-2} - B_1Y_{n-1} - B_2Y_{n-2}$ ใช้หลักการ DA เปลี่ยนเป็นโครงสร้างทางฮาร์ดแวร์ของวงจรการกระจายทางคณิตศาสตร์ข้อมูลเข้าแบบอนุกรมครั้งละหนึ่งบิต (bit at a time: BAAT) ได้ดังรูปที่ 4.7



รูปที่ 4.7 โครงสร้างวงจรกรองสัญญาณเชิงเลขอันดับสองแบบอนุกรม (1BAAT)

ถ้าพิจารณาการแทน X_n และ Y_n ด้วยจำนวนเลขส่วนเติมเต็มสองขนาด (B+1) บิต ดังนี้

$$X_n = -x_n^0 + \sum_{j=1}^{B} x_n^j \cdot 2^{-j} \quad \text{and} \quad Y_n = -y_n^0 + \sum_{j=1}^{B} y_n^j \cdot 2^{-j} \tag{4.19}$$

แทนค่าสมการที่ (4.19) ลงในสมการที่ (4.12) แล้วทำการจัดรูปสมการใหม่ให้สั้น ฟังก์ชัน Y_n จะสามารถถูกแสดงเป็นฟังก์ชันที่ขึ้นอยู่กับค่าของสัมประสิทธิ์ของวงจรกรอง ที่ถูกอ้างอิงด้วยค่าอินพุท X_n, X_{n-1}, X_{n-2} และเอาท์พุท Y_{n-1}, Y_{n-2} แทนฟังก์ชันสำหรับการอ้างอิงตำแหน่งหน่วยความจำ LUT ด้วย F(.) ดังสมการที่ (4.20)

$$Y_n = \sum_{j=1}^{B} 2^{-j} F(X_n^j, X_{n-1}^j, X_{n-2}^j, Y_{n-1}^j, Y_{n-2}^j) - F(X_n^0, X_{n-1}^0, X_{n-2}^0, Y_{n-1}^0, Y_{n-2}^0) \tag{4.20}$$

เมื่อ F (.) คือ ฟังก์ชันที่แสดงตามสมการที่ (4.21)

$$F(X_n^j, X_{n-1}^j, X_{n-2}^j, Y_{n-1}^j, Y_{n-2}^j) = a_0X_n^j + a_1X_{n-1}^j + a_2X_{n-2}^j - b_1Y_{n-1}^j - b_2Y_{n-2}^j \tag{4.21}$$

โดยปกติแล้วค่าของ F (.) ที่เป็นไปได้ที่เกิดจากสมการที่ (4.21) มีค่าเป็นไปได้ 2^5 ค่า ถูกปัดเศษแล้วเก็บไว้ในหน่วยตารางความจำขนาด B บิต สัญญาณ X_n, X_{n-1}, X_{n-2}, Y_{n-1}, Y_{n-2} จะถูกแท็ปจากชีฟรีจีสเตอร์ เพื่อการอ้างอิงค่าตำแหน่งของหน่วยความจำ เอาท์พุท Y_n จะได้จากการบวกกันติดต่อกัน B+1 ครั้งที่แอกคิวมูเลเตอร์ โดยจะถูกเลื่อนออกมาจากเอาท์พุทของหน่วยความจำดังในรูปที่ 4.5

รายละเอียดของขั้นการสร้างวงจรกรองดิจิตอลแบบ DA จะได้กล่าวโดยละเอียดในหัวข้อการสร้างตัวกรองต่อไป ซึ่งจะได้ทำการพิจารณาถึงผลกระทบของความยาวคำจำกัด (Finite word-length effect) และการลดความผิดพลาดดังกล่าวโดยประยุกต์ร่วมกับการใช้โครงสร้าง DA เพื่อให้วงจรกรองที่สร้างขึ้นไม่ต้องเพิ่มอุปกรณ์สำหรับการลดความผิดพลาด

4.6 พีชคณิตของตัวคูณแบบอื่นๆ

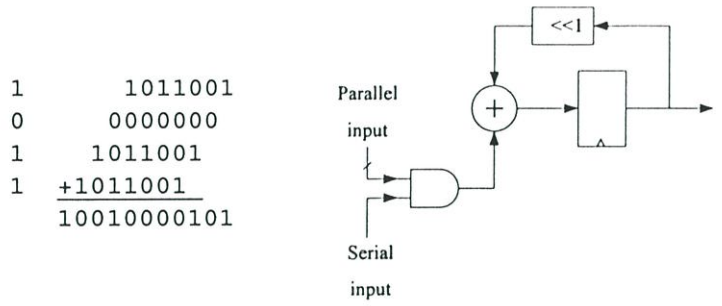
การคูณโดยพื้นฐานแล้วเป็นกระบวนการที่ใช้หลักการเลื่อนค่าและการบวก แต่อย่างไรก็ตามพีชคณิตการคูณยังสามารถทำได้อีกหลากหลายวิธีการ ซึ่งสามารถสร้างลงในอุปกรณ์ FPGA ได้ ซึ่งในหัวข้อนี้จะขอกล่าวตัวคูณรูปแบบอื่นๆแต่พอสังเขปดังต่อไปนี้

4.6.1 ตัวคูณแบบสเกลแอกคิวมูเลเตอร์ (Scaling Accumulator Multipliers)

<u>คุณสมบัติเด่น</u>

- อัลกอริทึมขนานโดยการอนุกรม
- ทำการบวกและเลื่อนวนซ้ำ
- ใช้สัญญาณนาฬิกา N รอบจึงเสร็จสมบูรณ์
- ขนาดการออกแบบกะทัดรัด
- ข้อมูลอินพุทสามารถเป็น MSB หรือ LSB ก่อนขึ้นอยู่กับทิศทางการเลื่อนของแอกคิวมูเลเตอร์
- อินพุทแบบขนาน

รูปที่ 4.8 แสดงตัวคูณแบบสเกลแอกคิวมูเลเตอร์จะทำการคูณด้วยการเลื่อนและบวกวนซ้ำ อินพุทหนึ่งๆจะถูกแทนด้วยรูปแบบบิตขนาน ขณะที่รูปแบบอื่นเป็นแบบบิตอนุกรมแต่ละบิตในอินพุทอนุกรมจะคูณด้วย 0 หรือ 1 โดยที่อินพุทขนานจะเก็บค่าคงที่ ขณะที่แต่ละบิตอนุกรมแสดงค่า จะสังเกตได้ว่าจะเป็นการคูณ 1 บิตผ่านอินพุทขนานที่ไม่เปลี่ยนแปลงหรือแทนด้วยศูนย์ ผลลัพธ์แต่ละบิตจะถูกบวกในแอกคิวมูเลเตอร์ การบวกจะถูกชิฟหนึ่งบิตก่อนที่ผลลัพธ์ของการคูณถัดไปจะถูดบวกเพิ่มเข้ามา



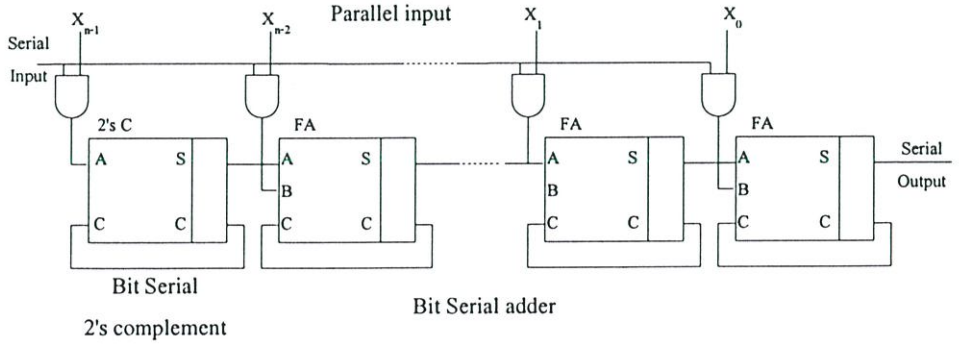
รูปที่ 4.8 ตัวคูณแบบสเกลแอกคิวมูเลเตอร์

4.6.2 ตัวคูณแบบอนุกรมโดยบูทขนาน (Serial by Parallel Booth Multipliers)

<u>คุณสมบัติเด่น</u>

- ต้องการกำจัดด้วยการบวกแบบบิตอนุกรมสำหรับแคริเชน
- เหมาะสำหรับ FPGA ที่ไม่มีแคริ่ลอจิกแบบเร็ว
- อินพุทอนุกรมต้องเริ่มต้นด้วย LSB
- เอาท์พุทแบบอนุกรม
- การเชื่องโยงภายในต้องใกล้ที่สุดยกเว้นอินพุทอนุกรมที่ต้องการกระจาย
- หนึ่งช่วงเวลาแฝงภายใน

เป็นการบวกทั่วไปโดยอาศัยตัวคูณแบบบูท โดยเฉพาะอย่างยิ่งเหมาะสมกับตัวประมวลผลแบบอนุกรมที่สร้างบน FPGAs ซึ่งไม่มีห่วงโซ่แคริ่ เพราะการเชื่อมโยงภายในจะอยู่ใกล้ชิดมาก ยกเว้นเฉพาะแต่อินพุทซึ่งเป็นแบบอนุกรม และต้องเป็นแบบขยายบิตเครื่องหมายให้มีความยาวบิตเท่ากับขนาดของผลรวมของอินพุทอนุกรม และหลีกเลี่ยงการเกิดโอเวอร์โฟว์ของอินพุทขนาน ซึ่งหมายความว่าตัวคูณจะใช้จำนวนสัญญาณนาฬิกามากกว่าแบบการปรับขนาดแอกคิวมูเลเตอร์ โครงสร้างแบบ TTL ของตัวคูณอนุกรมแบบบูทขนานแสดงดังรูปที่ 4.9



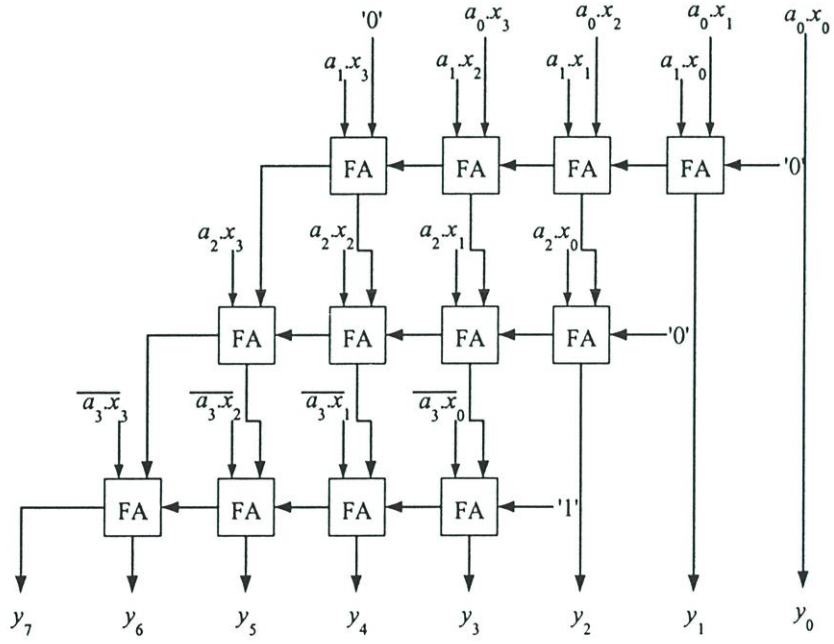
รูปที่ 4.9 ตัวคูณแบบอนุกรมโดยบูทขนาน

4.6.3 ตัวคูณแบบริปเปิ้ลแคร์รีอาเรย์ (Ripple Carry Array Multipliers)

<u>คุณสมบัติเด่น</u>

- รูปแบบโรว์ริปเปิ้ล (Row Ripple)
- อัลกอริทึมการเลื่อนและบวกแบบคลื่ออก
- ความล่าช้าขึ้นกับจำนวน N

รูปที่ 4.10 แสดงตัวคูณแบบริปเปิ้ลแคร์รีอาเรย์ (หรือเรียก รูปแบบ Row Ripple) เป็นการคลื่ออก แสดงรูปลักษณ์การทำการเลื่อนและบวกแบบอัลกอริทึมดั้งเดิม ตัวอย่างตัวคูณขนาด 4x4 แสดงโครงสร้างตัวบวกที่ต่อร่วมกันทุกบิตผลคูณในตัวคูณ บิตผลคูณจะเป็นค่าลอจิกที่ได้ของแต่ละอินพุทความ และความล่าช้าเกิดขึ้นภายในเส้นทางอินพุท LSB จนถึง MSB ของผลคูณ มีขนาดโดยประมาณเท่ากับ 2*n



รูปที่ 4.10 ตัวคูณแบบริปเปิ้ลแคร์รีอาเรย์ขนาด 4x4

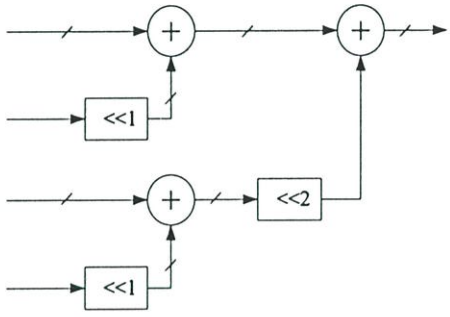
โครงสร้างพื้นฐานแบบนี้ง่ายต่อการสร้างลงบน FPGA แต่เปลืองจำนวนลอจิกเซลบน FPGA มาก และยังมีขนาดใหญ่กว่าอีกทั้งยังทำงานช้ากว่าตัวคูณแบบอื่น

4.6.4 ตัวคูณแบบโรว์แอดเดอร์ทรี (Row Adder Tree Multipliers)

<u>คุณสมบัติเด่น</u>

- รูปแบบโรว์ริปเปิ้ลที่มีประโยชน์สูง
- โดยพื้นฐานแล้วใช้จำนวนเกทเท่ากันกับแบบโรว์ริปเปิ้ล
- ตัวบวกแบบแถวจัดรูปเป็นแบบต้นไม่เพื่อลดความล่าช้า
- การเชื่อโยงยุ่งยากซับซ้อน แต่สามารถทำงานกับ FPGA ได้ดี
- สัดส่วนความล่าช้าเท่ากับ log2(N)

รูปที่ 4.11 แสดงตัวคูณแบบโรว์แอดเดอร์ทรี ที่ทำการจัดรูปตัวบวกใหม่ของตัวคูณแบบโรว์ริปเปิ้ล เพื่อทำให้จำนวนของตัวบวกที่ได้ในแต่ละผลคูณย่อยจะต้องผ่านค่าเท่ากัน ทำให้ใช้จำนวนตัวบวกเท่ากัน แต่ทางผ่านที่แย่สุดจะเป็นเพียง log2(n) ของจำนวนตัวบวกแทนที่จะเป็น n ตัวบวก แต่ที่สำคัญคือจะช่วยลดความล่าช้าได้ สำหรับตัวคูณแบบไปป์ไลน์ จะสามารถช่วยลดจำนวนสัญญาณนาฬิกาได้ โครงสร้างการเชื่อมต่อแบบต้นไม้ทำให้บางเส้นเชื่อมต่อมีความยาวมากกว่าแบบ Row Ripple เป็นผลให้ ตัวคูณ pipelined row ripple multiplier สามารถให้ปริมาณเอาท์พุทที่สูงกว่าใน FPGA (รอบสัญญาณนาฬิกาที่สั้นกว่า) แม้ว่าจะให้จำนวนรอบเพิ่มขึ้นก็ตาม

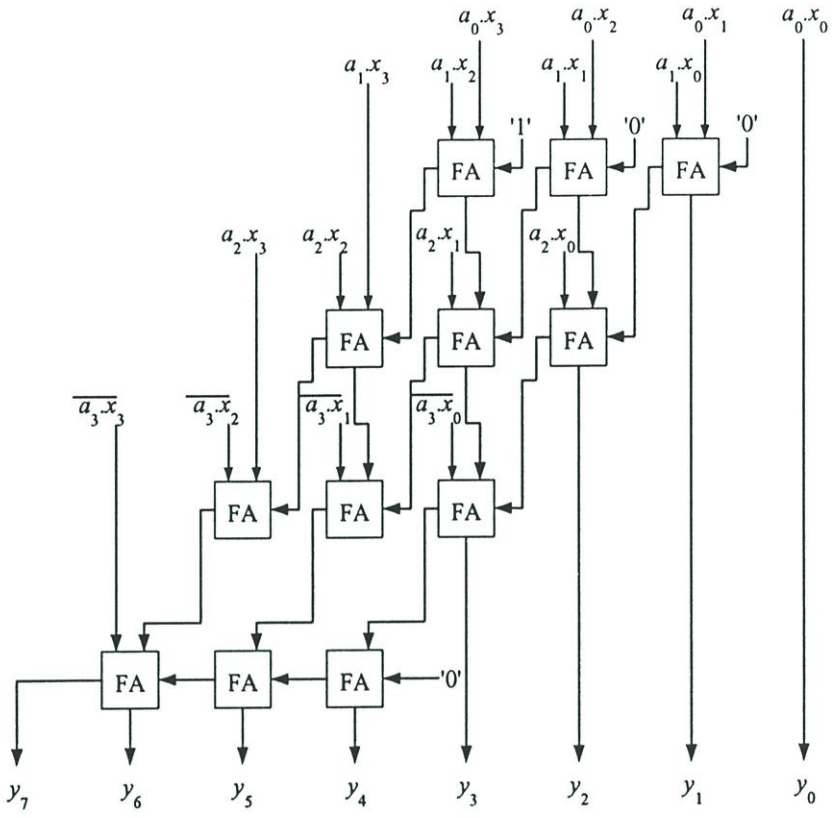


รูปที่ 4.11 ตัวคูณแบบโรว์แอดเดอร์ทรี

4.6.5 ตัวคูณแบบแครีเซฟอาเรย์ (Carry Save Array Multipliers)

<u>คุณสมบัติเด่น</u>

- รูปแบบคอลัมน์ริปเปิ้ล
- โดยพื้นฐานจะมีดีเลย์และจำนวนเกทที่เท่ากับแบบโรว์ริปเปิ้ล
- ระดับความเร็วเกทเพิ่มขึ้นได้สำหรับ ASICs
- ตัวบวกริปเปิ้ลสามารถแทนที่ด้วย ตัวบวกแครี่ทรี
- รูปแบบการเชื่อมโยงปกติ



รูปที่ 4.12 ตัวคูณแบบแครีเซฟอาเรย์

โดยเมื่อพิจารณาตามรูปที่ 4.12 จะพบว่าวงจรคูณที่ประกอบไปด้วยวงจรบวกชนิดแครี่เซฟเป็นองค์ประกอบหลัก จะมีเวลาหน่วงสูงสุด ลดลงไปกว่าเมื่อเทียบกับการใช้วงจรบวกชนิดแครี่ริปเปิ้ล เวลาหน่วงของวงจรนี้มีค่าเท่ากับเวลาหน่วงของตัวบวกเต็มเพียง 6 ตัวเรียงกัน ถ้าต้องการลดเวลาหน่วงสูงสุดของวงจรคูณลงอีก เราสามารถนำวงจรบวกชนิดการทดมองล่วงหน้ามาแทนในแถวสุดท้ายได้

4.6.6 ตัวคูณแบบลุคอัพเทเบิ้ล (Look up Table Multipliers)

<u>คุณสมบัติเด่น</u>

- เป็นการรวมตารางเวลาสมบูรณ์ของทุกค่าอินพุท
- หนึ่งบิตแอดเดรสสำหรับแต่ละบิตในแต่ละอินพุท
- ขนาดตารางโตขึ้นอย่างเอ็กโพเนนเชียล
- ใช้ได้จำกัดค่ามาก
- รวดเร็ว ผลลัพธ์ได้ทันทีเพียงแอกเซสหน่วยความจำ

ตารางที่ 4.3 แสดงตัวอย่างของตัวคูณแบบตาราง LUT เป็นบล็อกหน่วยความจำง่ายๆ ที่เก็บค่าผลคูณทั้งหมดทุกค่าที่เป็นไปได้ไว้ ขนาดของตารางจะใหญ่ขึ้นตามจำนวนอินพุทและสามารถสร้างบน FPGA ได้ในทางปฏิบัติ ตารางต่อไปนี้เป็นค่าของผลคูณ 3x3 บิต หรือ 6 บิตอินพุท

ตารางที่ 4.3 ผลคูณแบบลุคอัพเทเบิ้ล 3x3 บิต

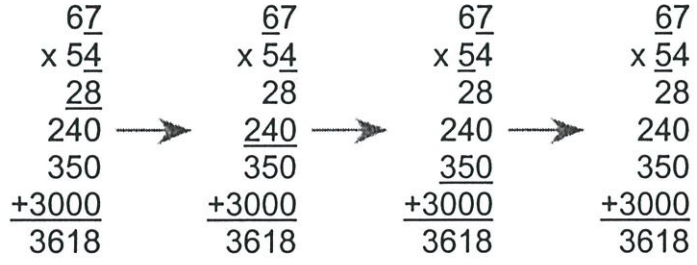
	000	001	010	011	100	101	110	111
000	000000	000000	000000	000000	000000	000000	000000	000000
001	000000	000001	000010	000011	000100	000101	000110	000111
010	000000	000010	000100	000110	001000	001010	001100	001110
011	000000	000011	000110	001001	001100	001111	010010	010101
100	000000	000100	001000	001100	010000	010100	011000	011100
101	000000	000101	001010	001111	010100	011001	011110	100011
110	000000	000110	001100	010010	011000	011110	100100	101010
111	000000	000111	001110	010101	011100	100011	101010	110001

4.6.7 ตัวคูณแบบลุคอัพเทเบิ้ลผลคูณย่อย (Partial Product LUT Multipliers)

<u>คุณสมบัติเด่น</u>

- การทำงานเหมือนกับการตั้งคูณยาวด้วยมือ
- LUT ถูกใช้เพื่อรับค่าผลคูณของข้อมูล
- ผลคูณย่อยจะถูกรวมด้วยตัวบวกแบบต้นไม้

พิจารณารูปที่ 4.13 เป็นการแสดงการคูณด้วยตัวคูณแบบลุคอัพเทเบิ้ลผลคูณย่อย ซึ่งจะใช้เทคนิคเดียวกับการตั้งคูณยาวด้วยมือและใช้ LUT เก็บค่าผลคูณไว้เพื่อดึงค่ามาใช้สำหรับแต่ละผลคูณย่อย



รูปที่ 4.13 ลำดับการคูณแบบลุคอัพเทเบิ้ลผลคูณย่อย

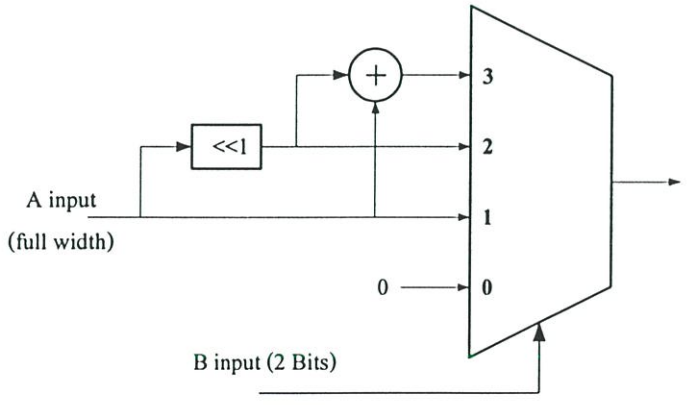
ด้วยการทำการคูณในแต่ละหลักที่เวลาหนึ่งและทำการเลื่อนและบวกค่าผลคูณย่อย ขนาดของหน่วยความจำตารางจะมีขนาดเล็กลงไปมาก ในกรณีที่แต่ละผลคูณย่อยได้รับหรือถูกบวกไม่มีความสำคัญ ปกติแล้วการถ่วงน้ำหนักทำโดยการเลื่อนจะต้องทำการเก็บค่าไว้ก่อน

4.6.8 ตัวคูณแบบคำนวณผลคูณย่อย (Computed Partial Product Multipliers)

<u>คุณสมบัติเด่น</u>

- ผลคูณย่อยที่ออปติไมซ์สำหรับ FPGAs ที่มี LUTs ขนาดเล็ก
- ผลคูณย่อยลดลงน้อยกว่าความลึกของตัวบวกแบบต้นไม้
- ผลคูณย่อยขนาด 2 x n ถูกสร้างขึ้นโดยลอจิกแทนที่จะเป็น LUT
- เล็กกว่าและเร็วกว่าตัวคูณแบบ ผลคูณย่อย 4 LUT

ผลบวกย่อยจากตัวคูณจะถูกสร้างขึ้นจาก 4 LUT ซึ่งสามารถพบใน FPGA แต่ไม่เหมาะสมเพราะยังให้ขนาดผลคูณย่อยที่มีจำนวนมากที่จะต้องถูกบวกรวมกัน และยังต้องการใช้ LUT จำนวนมากอีกด้วย เราสามารถสร้างตัวคูณที่มีประสิทธิภาพขึ้นได้โดยใช้บิตอินพุท 2 บิต จะให้ผลคูณ 4 ค่า และสามารถดึงผลคูณ 4 ค่านี้ออกมาใช้ได้ง่าย โดยในแต่ละลำดับการทำงานจะใช้ ตัวบวกและตัวเลื่อน และตัวมัลติเพล็กที่ถูกควบคุมด้วย 2 บิต เลือกตัวคูณที่เหมาะสมต่อค่าผลลัพธ์ออกมาแสดง ซึ่งไม่เหมือนผลลัพธ์แบบ LUT จึงไม่จำกัดขนาดความกว้างของผลคูณย่อยอินพุท A โครงสร้างแบบนี้จะช่วยลดจำนวนผลคูณย่อยและความลึกในตัวบวกต้นไม้ไปได้มาก รูปที่ 4.14 เป็นตัวอย่างโครงสร้างแสดงตัวคูณแบบคำนวณผลคูณย่อยขนาด 2x n บิต โดยใช้ 2 บิตทำการมัลติเพล็กเลือกค่าผลคูณย่อย



รูปที่ 4.14 ผลคูณย่อยขนาด 2 x n bit ที่ถูกสร้างขึ้นจากตัวบวกและตัวมัลติเพล็ก

4.6.9 ตัวคูณแบบเคซีเอ็ม (KCM Multipliers)

คุณสมบัติเด่น

- คูณอินพุทด้วยค่าคงที่
- LUT จะเก็บค่าจำนวนตารางเวลา
- ขนาดกว้างของค่าคงที่ไม่มีผลต่อความลึกของตัวบวกต้นไม้
- ทุก LUT อินพุทสำหรับเป็นตัวคูณ
- ประสิทธิภาพดีกว่าการคูณแบบเต็ม
- ขนาดคงที่เพราะเป็นเพียงค่าคงที่

ตัวคูณแบบเต็มที่มีช่วงความกว้างของอินพุทได้เต็มค่าของตัวคูณ ซึ่งถ้าตัวคูณหนึ่งเป็นค่าคงที่แล้ว มันค่อนข้างจะด้อยประสิทธิภาพในการสร้างเป็นตารางเวลา ซึ่งจะมีเพียงค่าคอลัมน์ที่สัมพันธ์เฉพาะค่าเท่านั้น ซึ่งเป็นที่รู้กันว่าค่าคงที่ (K) เป็นสัมประสิทธิ์คงที่ของตัวคูณ หรือ KCM พิจารณาตัวอย่างในตารางที่ 4.4 ซึ่งเป็นการคูณกันของค่าอินพุท 5 บิต (ค่า 0 ถึง 31) ด้วยค่าคงที่ 67 จะสังเกตได้ว่า ด้วยค่าคงที่ของตัวคูณทั้งหมดใน LUT ที่เป็นไปตามอินพุทตัวคูณที่แปรเปลี่ยน ซึ่งทำให้ KCM มีประสิทธิภาพกว่าการคูณแบบเต็มๆ (มีจำนวนผลคูณย่อยน้อยกว่าสำหรับจำนวนที่ให้มา)

ตารางที่ 4.4 ผลคูณอินพุท 5 บิตกับ "67" แบบเคซีเอ็ม

	5 bit input * 67			
input	**00**	**01**	**10**	**11**
000	0	536	1072	1608
001	67	603	1139	1675
010	134	670	1206	1742
011	201	737	1273	1809
100	268	804	1340	1876
101	335	871	1407	1943
110	402	938	1474	2010
111	469	1005	1541	2077

เมื่อ LUT ไม่สามารถให้ค่าได้ตามอินพุทตามต้องการได้อย่างเหมาะสมกับขนาดช่วงกว้างที่ต้องการ ลักษณะ LUT ที่สมมูลสามารถถูกรวมโดยการใช้เทคนิคของผลคูณย่อยจากที่ได้กล่าวไปแล้ว ในกรณีนี้ตัวคูณค่าคงที่ที่มีความยาวเต็มผลคูณย่อยจะได้เป็นขนาด m x n เมื่อ m เป็นจำนวนอินพุทของ LUT และ n เป็นความกว้างของค่าคงที่

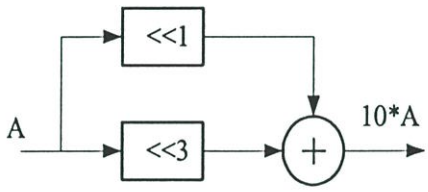
4.6.10 ตัวคูณแบบคูณค่าคงที่จากตัวบวก (Constant Multipliers from Adders)

<u>คุณสมบัติเด่น</u>

- บวกสำหรับแต่ละบิต '1' ในค่าคงที่
- ลบแทนค่าใหม่ด้วย '1' โดยใช้บูทรีโค้ดดิ้ง
- มีประสิทธิภาพขนาดขึ้นอยู่กับค่าคงที่
- ตัวคูณ KCM โดยทั่วไปจะมีประสิทธิภาพสำหรับค่าคงที่ไม่เจาะจง

อัลกอริทึมของการคูณแบบบวกและเลื่อนจะให้ผลลัพธ์ m ขนาด 1xn ผลคูณย่อย และจะถูกรวมกันด้วยการเลื่อนค่าก่อนอย่างเหมาะสม โดยที่ผลคูณย่อยจะมีความสอดคล้องกับค่าบิต '0' ในอินพุทแต่ละหนึ่งบิตที่เป็นศูนย์และยังไม่ต้องทำการรวมเข้ากับผลบวก พิจารณาตัวอย่างตามรูปที่ 4.15 (เป็นการคูณกันของ 1010 กับ 1011001) แสดงให้เห็นว่าถ้าจำนวนค่าบิต '1' ในสัมประสิทธ์คงที่ของตัวคูณจะมีขนาดเล็ก ซึ่งค่าคงที่ของตัวคูณอาจจะสร้างด้วยการเชื่อมต่อตัวเลื่อนชิฟและตัวบวกไม่กี่ตัว ตัวอย่างแสดงดังรูปต่อไปนี้

```
0       0000000
1      1011001
0     0000000
1   +1011001
     1101111010
```

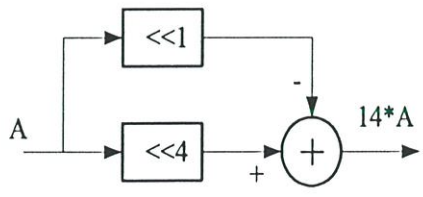


รูปที่ 4.15 ขั้นตอนการคูณแบบคูณค่าคงที่จากตัวบวก

พิจารณาตัวอย่างในรูปที่ 4.16 เป็นการหาผลคูณของ ($14=1110_2$ กับ $89=1011001_2$ การตั้งคูณด้านซ้าย เทียบกับการคูณตรงกลาง) ในกรณีที่จำนวนบิตที่เป็น '1' ในค่าคงที่ เราสามารถตัดตัวบวกออกโดยการใช้วิธีการบูทรีโค้ดดิ้งกับตัวลบได้ ตัวอย่างเช่น 14 = 8+4+2 สามารถแสดงเป็น 14=16-2 ซึ่งจะช่วยลดจำนวนของผลคูณย่อยได้ขณะที่ให้ผลลัพธ์ที่เท่ากัน

```
0       0000000
1      1011001
1     1011001
1   +1011001
     10011011110
```

```
0       0000000
-1    1110100111
0      0000000
0     0000000
1   +1011001
     10011011110
```



รูปที่ 4.16 ขั้นตอนการคูณแบบคูณค่าคงที่จากตัวบวก (2)

เป็นที่แน่ชัดว่าความซับซ้อนของตัวคูณคงที่ที่ถูกสร้างขึ้นจากตัวบวก จะอิสระต่อค่าคงที่ สำ+หรับค่าคงที่ไม่เจาะจง ตัวคูณแบบ KCM ที่ได้กล่าวไปแล้วตอนต้นจึงเป็นตัวเลือกหนึ่งที่ดีสำหรับแอพพลิเคชั่นที่ต้องการความรวดเร็วและสเกลได้

4.6.11 ตัวคูณแบบจำกัดเซท LUT (Limited Set LUT Multipliers)

<u>คุณสมบัติเด่น</u>

- คูณอินพุทด้วยเซทค่าคงที่หนึ่ง
- คล้ายตัวคูณแบบ KCM
- LUT อินพุทบิตใช้เลือกใช้ค่าคงที่อันไหน
- มีประโยชน์ในตัวมอดูเลท หรือ การประยุกต์ใช้ประมวลสัญญาณอื่นๆ

ในการประมวลผลสัญญาณ บ่อยครั้งที่ขณะคูณค่าตัวคูณหนึ่งจะได้ค่าคงที่น้อยๆเซทหนึ่ง ในกรณีนี้ตัวคูณ KCM สามารถทำการขยายทำให้ LUT เก็บค่าตารางเวลาสำหรับแต่ละค่าคงที่ได้ อินพุทหนึ่งหรือมากกว่าของ LUT จะทำการเลือกค่าคงที่ที่จะถูกใช้ ในขณะที่อินพุทที่เหลือจะเป็นตัวคูณ ตัวอย่างของ 6 บิต LUT ที่เก็บค่าตารางสำหรับผลคูณค่าคงที่ 67 และ 85 แสดงดังตารางที่ 4.5

ตารางที่ 4.5 ผลคูณ 5 บิตกับ “67 และ 85” ด้วยตัวคูณแบบจำกัดเซท LUT

	5 bit input * 67				5 bit input * 85			
	000	**001**	**010**	**011**	**100**	**101**	**110**	**111**
000	0	536	1072	1608	0	680	1360	2040
001	67	603	1139	1675	85	765	1445	2125
010	134	670	1206	1742	170	850	1530	2210
011	201	737	1273	1809	255	935	1615	2295
100	268	804	1340	1876	340	1020	1700	2380
101	335	871	1407	1943	425	1105	1785	2465
110	402	938	1474	2010	510	1190	1870	2550
111	469	1005	1541	2077	595	1275	1955	2635

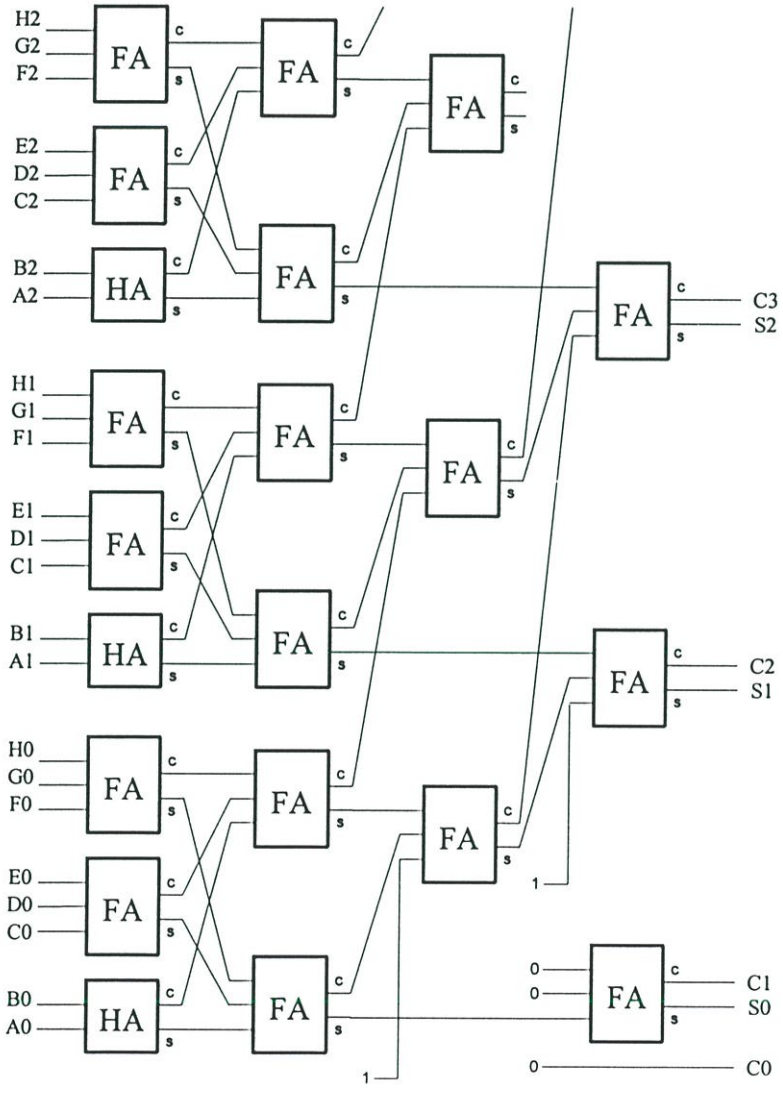
4.6.12 ตัวคูณแบบวอลเลซทรี (Wallace Trees)

<u>คุณสมบัติเด่น</u>

- ออปติไมซ์คอลัมน์แอดเดอร์ทรี
- รวมทุกผลคูณย่อยในสองเวกเตอร์ (แคร่ีและผลรวม)
- แคร่ีและผลรวมเอาท์พุทถูกรวมโดยการใช้ตัวบวกธรรมดา

- ดีเลย์มีค่าเป็น log(n)
- ตัวคูณวอลเลซทรีจะใช้วอลเลซทรีทำการรวมผลคูณย่อย 1 x n
- การเชื่อมต่อไม่ปกติเรียบง่าย
- ไม่เหมาะกับ FPGA

จากรูปที่ 4.17 เป็นตัวอย่างแสดงตัวคูณแบบวอลเลซทรี ขนาด 8 อินพุท ซึ่งจะทำการรวม 8 อินพุท ของผลคูณย่อยกับเอาท์พุทเวกเตอร์ที่สัมพันธ์กับค่าผลรวมและแคร่ ตัวบวกธรรมดาถูกใช้ทำการรวมเอาท์พุทเหล่านี้เพื่อได้เป็นผลลัพธ์สมบูรณ์



รูปที่ 4.17 ส่วนหนึ่งของวอลเลซทรี ขนาด 8 อินพุท

4.6.13 ตัวคูณแบบบูทรีโคดดิ้ง (Booth Recoding)

บูทรีโคดดิ้ง เป็นวิธีการในการลดจำนวนผลคูณย่อยที่จะเป็นผลรวม โดยใช้หลักการของบูททำการสังเกตว่าเมื่อไรที่ตัวเลข 1 จะเกิดขึ้นในการคูณ จำนวนของผลคูณย่อยสามารถที่จะลดลงโดยใช้การลบออก ตัวอย่างเช่นการคูณเลข 89 ด้วย 15 แสดงได้ดังต่อไปนี้ คือมีผลคูณย่อย 4 ชุดขนาด 1xn ที่จะต้องถูกบวกรวมกัน จะเสมือนการหักค่าออกดังแสดงด้านขวามือ

```
1        1011001          1       -1011001
1       1011001           1        0000000
1      1011001            1       0000000
1     1011001             1      0000000
0   +0000000              0    +1011001
     10100110111               10100110111
```

รูปที่ 4.18 การคูณแบบบูทรีโค้ดดิ้ง

4.7 บทสรุป

ในบทนี้ได้กล่าวถึงทฤษฎีที่เกี่ยวข้องกับการดำเนินการทางพีชคณิตต่างๆ โดยให้ความสำคัญกับคณิตศาสตร์แบบการกระจาย (DA) เป็นหลัก ที่มาต่างๆอันเป็นการนำไปสู่การพัฒนาประสิทธิภาพในการคำนวณให้ดียิ่งขึ้น รวมถึงโครงสร้างของส่วนประกอบในการสร้างขึ้นเป็นระบบดิจิตอล ตลอดไปจนอัลกอริทึมการคูณแบบอื่นๆ ที่อาจประยุกต์ใช้งานได้ เมื่อเราได้เข้าใจถึงอัลกอริทึมในการคำนวณอย่างดีแล้วจะช่วยให้ขั้นตอนการออกแบบเป็นระบบของวงจรกรองดิจิตอลนั้นง่ายยิ่งขึ้น ซึ่งเป็นรูปแบบที่มีความพร้อมในการเชื่อมโยงไปสู่การออกแบบเป็นวงจรในระดับเกตต่อไป ดังจะได้กล่าวถึงใน บทที่ 6 แต่ด้วยข้อจำกัดของความยาวคำ ซึ่งมีผลต่อความถูกต้องแม่นยำ เราจึงต้องวิเคราะห์ปัญหานี้และแนวทางการแก้ไขปัญหา โดยจะนำเสนอต่อจากนี้

บทที่ 5
ผลกระทบของความยาวคำจำกัดในวงจรกรองเชิงเลข

5.1 บทนำ

ในทางอุดมคติแล้วตัวแปรของระบบพร้อมด้วยกับตัวแปรของสัญญาณ จะมีความเที่ยงตรงตลอดทั้งช่วงที่ไม่จำกัด แต่ในทางปฏิบัติแล้วเราสามารถแทนค่าได้เพียง ค่าไม่ต่อเนื่องในช่วงที่จำกัดเฉพาะ ซึ่งในทางดิจิตอลแล้วเป็นการใช้รีจีสเตอร์ในการเก็บค่าขนาดความยาวที่จำกัดเท่านั้น ผลของการประมวลแบบไม่ต่อเนื่อง (discrete process) จะให้ผลลัพธ์เป็นสมการความต่างแบบไม่เป็นเชิงเส้น ซึ่งเป็นลักษณะเฉพาะของระบบเวลาไม่ต่อเนื่อง สมการแบบไม่เป็นเชิงเส้นเหล่านี้ โดยหลักการแล้วไม่สามารถทำการวิเคราะห์และปฏิบัติได้ตรงกับความจริงโดยแท้ แต่อย่างไรก็ตามถ้าขนาดปริมาณการควอนไตซ์เพียงเล็กน้อยเมื่อเทียบกับขนาดของสัญญาณที่แปรเปลี่ยนและตัวแปรของวงจรกรองแล้วเราสามารถประมาณแบบง่ายๆด้วยแบบจำลองได้ การใช้แบบจำลองทางสถิติสามารถศึกษาผลกระทบของความไม่ต่อเนื่องและปรับปรุงผลลัพธ์ซึ่งสามารถตรวจสอบการทดลองได้ เราอาจจำแนกกระบวนการควอนไตซ์เลขจำนวนซึ่งทำให้เกิดความผิดพลาดขึ้นได้ 3 ชนิด คือ ความผิดพลาดที่เกิดจากการควอนไตซ์ค่าสัมประสิทธิ์, ความผิดพลาดที่เกิดจากการควอนไตซ์ค่าผลการคูณ และความผิดพลาดที่เกิดจากการควอนไตซ์ค่าอินพุท นอกจากนั้นแล้ว ในการจัดสร้างวงจรกรองที่มีการป้อนกลับ (Recursive Filters) ผลกระทบที่เกิดจากความยาวคำที่จำกัดนี้ยังอาจขึ้นอยู่กับสาเหตุอื่นๆอีกได้ เช่น โครงสร้างของวงจรกรอง (Filter Structures) การวางตำแหน่งของโพลและซีโร่ (Pole-Zero Configuration) การแทนจำนวนลบ (Representation of Negative Number) การปัดเศษหรือการตัดปลาย (Rounding or Truncation) การล้น (Overflow) และลักษณะของสัญญาณขาเข้า เป็นต้น ผลกระทบนี้ก่อให้เกิดข้อผิดพลาดในรูปแบบต่างๆ ซึ่งเกิดจากความไม่เป็นเชิงเส้น (Nonlinearities) ประเภทต่างๆ ที่สามารถคาดเดาได้ยาก

5.2 การควอนไตซ์เลขจำนวน

จากที่ขนาดความยาวบิตของรีจีสเตอร์มีค่าจำกัด ถ้าให้มีความยาวบิตเป็น L บิต (ยกเว้นบิตเครื่องหมาย) ค่าของจำนวนน้อยที่สุดที่แปรเปลี่ยน (หมายความรวมถึงสัญญาณ) สามารถแทนได้ด้วยการเปลี่ยนค่าบิตเป็น 1 ที่ตำแหน่งบิตนัยสำคัญต่ำสุด ซึ่งมีความสัมพันธ์กับ 2^{-L} ดังนั้นจำนวนที่ประกอบกันเป็นขนาด B บิต (ยกเว้นบิตเครื่องหมาย) เมื่อ B > L จะถูกควอนไตซ์ ซึ่งสามารถทำได้ 2 วิธีคือ (1) โดยการตัดปลาย (Truncating) คือตัดทุกบิตที่ไม่สามารถแสดงในรีจีสเตอร์ได้ทิ้งไป หรือ (2) โดยการปัดเศษ (Rounding) ทำการปัดเศษของจำนวนให้มีค่าใกล้เคียงที่สุด

ถ้าจำนวน x ถูกทำการควอนไตซ์ค่าแล้ว ความผิดพลาด ε สามารถอธิบายได้ดังสมการที่ (5.1)

$$\varepsilon = x - Q[x] \tag{5.1}$$

เมื่อ $Q[x]$ เป็นค่าของ x ที่ถูกควอนไตซ์ สัมพันธ์กับปริมาณ 2^{-L}

ε เป็นฟังก์ชันที่มีค่าไม่แน่นอน (Random) ชนิดหนึ่ง

โดยขนาดช่วงของ ε จะขึ้นอยู่กับการแทนจำนวนและยังขึ้นกับชนิดของการควอนไตซ์อีกด้วย ลองพิจารณาตรวจสอบค่าต่างๆที่เป็นไปได้เริ่มตั้งแต่การตัดปลาย จะเห็นได้ว่าในตารางที่ 5.1 เป็นการแทนจำนวนบวกจะสมมาตรในการแทนแบบกำหนดจุด (Fixed point) ทั้งสามชนิด ซึ่งมีการตัดปลายเพียงชนิดเดียวที่ลดจำนวนที่เป็นบวกไป ε เป็นจำนวนบวก และจะมีค่าสูงสุดเกิดขึ้นเมื่อทุกบิตมีค่าเป็น "1" ในแต่ละกรณีแสดงได้ว่า

$$0 \le \varepsilon_T \le 2^{-L} - 2^{-B} \qquad \text{เมื่อ} \quad x \ge 0 \tag{5.2}$$

ตารางที่ 5.1 ค่าสมมูลกับเลขฐานสิบของจำนวน 0.000 ถึง 1.111

Binary number	ค่าสมมูลกับเลขฐานสิบ		
	Signed Magnitude	One's complement	Two's complement
0.000	0	0	0
0.001	1	1	1
0.010	2	2	2
0.011	3	3	3
0.100	4	4	4
0.101	5	5	5
0.110	6	6	6
0.111	7	7	7
1.000	-0	-7	-8
1.001	-1	-6	-7
1.010	-2	-5	-6
1.011	-3	-4	-5
1.100	-4	-3	-4
1.101	-5	-2	-3
1.110	-6	-1	-2
1.111	-7	-0	-1

5.2.1 การตัดปลายของเลขจำนวนลบ

การแทนค่าจำนวนทั้งสามแบบในตาราง จะต้องทำการพิจารณาแยกกัน คือ

5.2.1.1 การแทนจำนวนแบบ Signed-magnitude

การตัดปลายจะลดค่าขนาดของจำนวนหรือเพิ่มค่าเครื่องหมาย ซึ่ง $Q[x] > x$ หรือ

$$-\left(2^{-L} - 2^{-B}\right) \le \varepsilon_T \le 0 \qquad \text{สำหรับ } x < 0 \tag{5.3}$$

5.2.1.2 การแทนด้วยเลขส่วนเติมเต็มหนึ่ง

แทนสัญลักษณ์ของเลขส่วนเติมเต็มหนึ่งด้วย x_1 และนิยามเลขส่วนเติมเต็มหนึ่งตามสมการที่ (5.4)

$$x = -\sum_{i=1}^{B} b_{-i} 2^{-i} \tag{5.4}$$

เมื่อ $b_{-i} = 0$ หรือ 1 ซึ่งเมื่อแทนแบบเลขส่วนเติมเต็มหนึ่งจะได้ว่า

$$x_1 = 2 - 2^{-L} - \sum_{i=1}^{B} b_{-i} 2^{-i} \tag{5.5}$$

ถ้ากำหนดให้การควอนไตซ์ของเลขส่วนเติมเต็มหนึ่งเป็น $Q[x_1]$ และไม่สนใจว่าทุกบิตมีค่าเป็น "0" จะได้ $\varepsilon = 0$ ที่ขีดสูงสุดถ้าทุกบิตไม่สนใจว่าเป็น "1" เราจะได้ว่า

$$Q[x_1] = 2 - 2^{-L} - \sum_{i=1}^{B} b_{-i} 2^{-i} - \left(2^{-L} - 2^{-B}\right) \tag{5.6}$$

ดังนั้น จากสมการที่ (5.6) ค่าเลขฐานสิบที่สมมูลกับ $Q[x_1]$ คือ

$$Q[x] = -\left[\sum_{i=1}^{B} b_{-i} 2^{-i} + \left(2^{-L} - 2^{-B}\right)\right] \tag{5.7}$$

ดังนั้นจากสมการที่ 5.1 ถึง 5.7 จะได้ว่า

$$0 \le \varepsilon_T \le 2^{-L} - 2^{-B} \qquad \text{สำหรับ} \qquad x < 0 \tag{5.8}$$

5.2.1.3 การแทนด้วยเลขส่วนเติมเต็มสอง

ความไม่สมมูลกันสำหรับจำนวนส่วนเติมเต็มสองก็สามารถแสดงสรุปอย่างง่ายได้เช่นเดียวกันกับ จำนวนแบบคิดขนาดเครื่องหมายได้ว่า

$$-q < \varepsilon_T < q \tag{5.9}$$

เมื่อ $q = 2^{-L}$ เป็นระดับขั้นของการควอนไตซ์ สำหรับเมื่อเป็นจำนวนส่วนเติมเต็มหนึ่งหรือสอง

$$0 \le \varepsilon_T < q \tag{5.10}$$

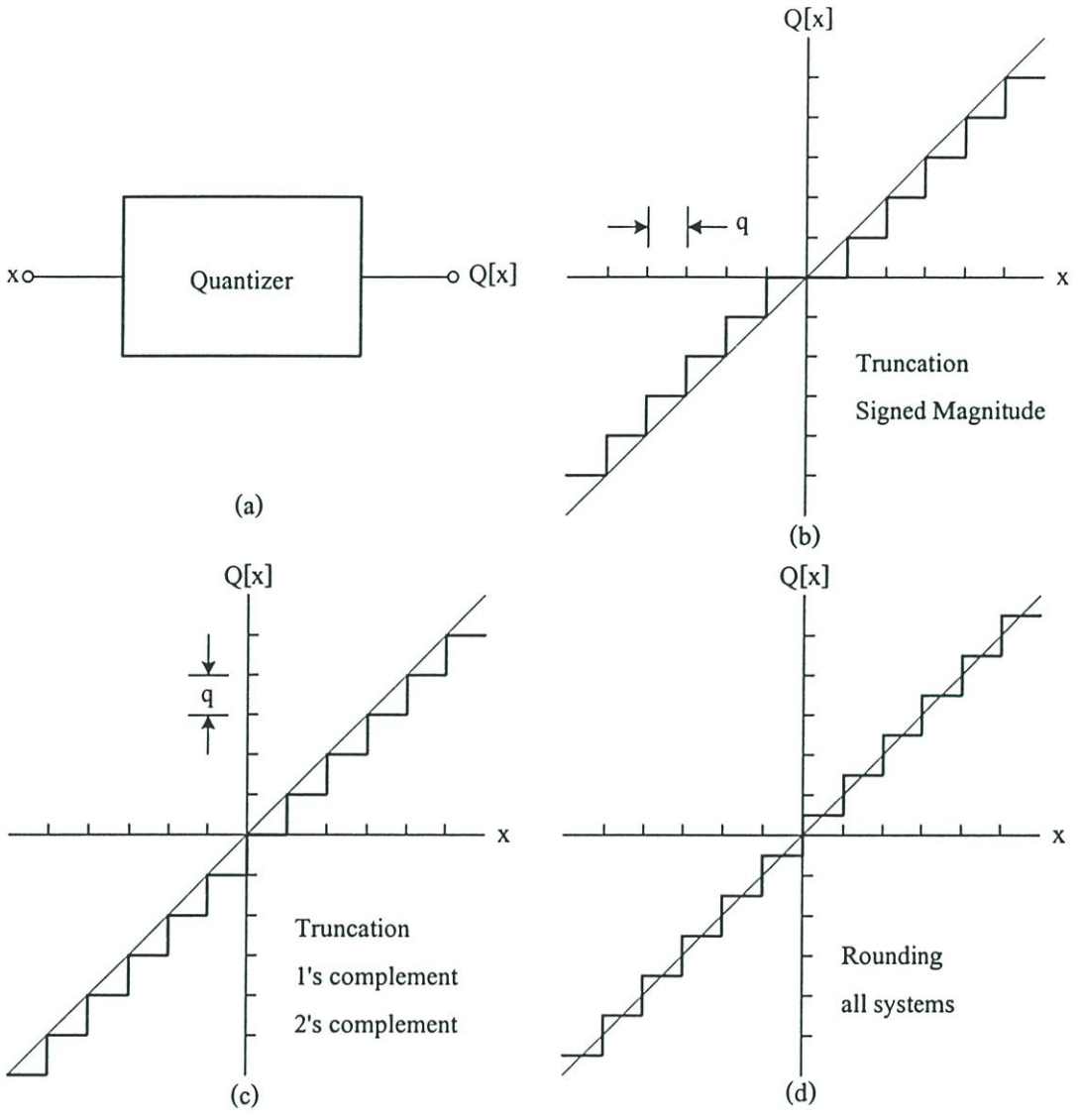
จะเห็นได้ว่าความผิดพลาดที่เกิดจากการควอนไตซ์ สามารถทำให้มีค่าน้อยลงได้ตามต้องการด้วยการเลือกค่า L ให้มีค่าเพียงพอเหมาะสม

5.2.2 การปัดเศษ (Rounding)

ค่าความผิดพลาดสามารถเป็นได้ทั้งบวกและลบตามนิยามและค่าสูงสุดของ $q/2$ แต่ถ้าจำนวนอยู่ระหว่างกลางระดับการควอนไตซ์ก็จะถูกปัดขึ้น เราจะได้ว่า

$$-\frac{q}{2} \le \varepsilon_R \le \frac{q}{2} \tag{5.11}$$

การปัดเศษมีผลในทางปฏิบัติโดยการบวกหนึ่งที่ตำแหน่ง L+1 จึงทำการตัดค่าจำนวนทิ้งเป็น L บิต พิจารณารูปที่ 5.1 แสดงชนิดของการควอนไตซ์ที่ต่างกัน



รูปที่ 5.1 การควอนไตซ์จำนวน (a) ตัวควอนไตซ์, (b), (c) และ (d) แสดง Q[x] เทียบกับ x

5.3 การควอนไตซ์สัมประสิทธิ์

ความผิดพลาดของการควอนไตซ์สัมประสิทธิ์ จะทำให้ค่าซีโร่และโพลของทรานสเฟอร์ฟังก์ชันสับสนคลาดเคลื่อน ซึ่งเป็นที่แน่นอนการตอบสนองความถี่ย่อมผิดพลาดไป ผลผลิตที่ได้จากการควอนไตซ์ผิดพลาด นอกจากจะเป็นแหล่งกำเนิดสัญญาณรบกวนแล้วยังทำให้ที่เอาท์พุทมีสัญญาณรบกวนจากการปัดเศษเพิ่มมากขึ้นด้วย ซึ่งต้นเหตุความผิดพลาดสองชนิดสำคัญที่สามารถเปลี่ยนแปลงไปได้มากจากการใช้งานรูปแบบหนึ่งไปยังอีกแบบหนึ่ง นั้นก็คือขนาดความยาวคำของสัมประสิทธิ์และค่าสัญญาณ ความยาวคำของสัมประสิทธิ์จะเป็นตัวเลือกที่กำหนดลักษณะการ

ตอบสนองความถี่ ในขณะที่ความยาวคำของสัญญาณจะเป็นตัวกำหนดลักษณะของอัตราส่วนสัญญาณต่อการรบกวน (SNR) พิจารณาลักษณะของวงจรกรองดิจิตอล $H(z)$ และให้ค่าต่างๆ เหล่านี้แสดงไว้ในรูปที่ 5.2

$M(\omega)=\left|H(e^{j\omega t})\right|$ คือ การตอบสนองทางขนาดที่ปราศจากการควอนไตซ์

$M_Q(\omega)$ คือ การตอบสนองทางขนาดที่ทำการควอนไตซ์

$M_I(\omega)$ คือ การตอบสนองทางขนาดทางอุดมคติ

$\delta_p(\delta_a)$ คือ ค่าที่ยอมในย่านแถบความถี่ผ่านของการตอบสนองทางขนาด

ผลกระทบจากสัมประสิทธิ์ควอนไตซ์จะอธิบายในรูปของความผิดพลาด ΔM ใน $M(\omega)$ ซึ่งสามารถหาได้จาก

$$\Delta M = M(\omega) - M_Q(\omega) \tag{5.12}$$

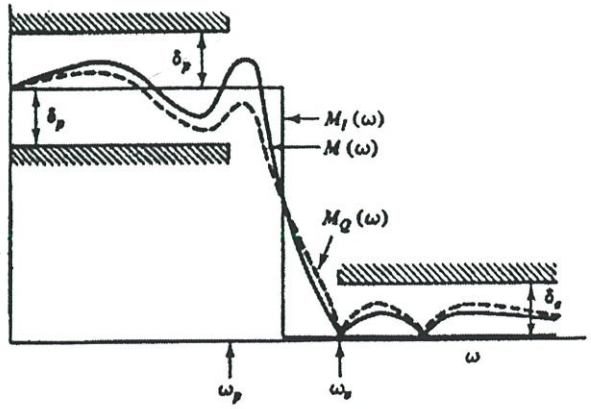
ค่าสูงสุดที่เป็นไปได้ของ $|\Delta M|$ สามารถที่จะแทนด้วย $\Delta M_{\max}(\omega)$ จากรูปที่ 5.2 สามารถพิจารณาได้ว่า

$$\Delta M_{\max}(\omega)=\begin{cases}\delta_p-\left|M(\omega)-M_I(\omega)\right| & for \quad \omega\le\omega_p\\ \delta_a-\left|M(\omega)-M_I(\omega)\right| & for \quad \omega\ge\omega_a\end{cases} \tag{5.13}$$

และถ้า

$$\left|\Delta M\right|\le\Delta M_{\max}(\omega) \tag{5.14}$$

สำหรับ $0\le\omega\le\omega_p$ และ $\omega_a\le\omega\le\omega_s/2$ ก็จะได้ค่าลักษณะเฉพาะที่ต้องการ ค่าความยาวคำที่ดีที่สุดสามารถประเมินจาก $|\Delta M|$ คือฟังก์ชันของความถี่สำหรับค่ามากที่สุดที่ใช้ได้ตามสมการ (5.14)



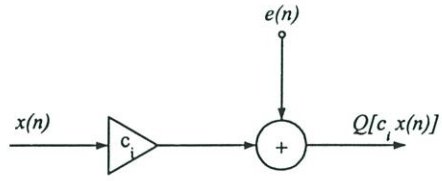
รูปที่ 5.2 การตอบสนองความถี่จากควอนไตซ์เซชันของสัมประสิทธิ์

5.4 การควอนไตซ์ของผลคูณ

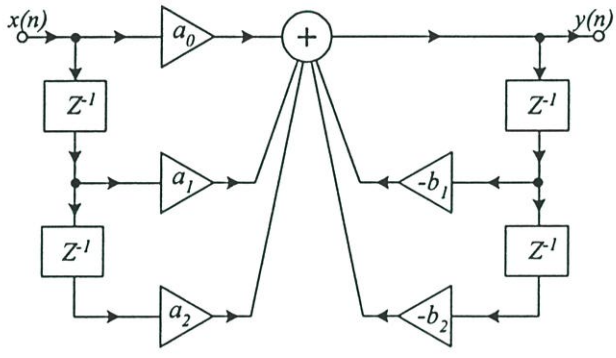
เราสามารถแสดงค่าเอาท์พุทที่ได้จากตัวคูณที่มีความยาวคำจำกัดได้ว่า

$$Q[c_i x(n)] = c_i x(n) + e(n) \tag{5.15}$$

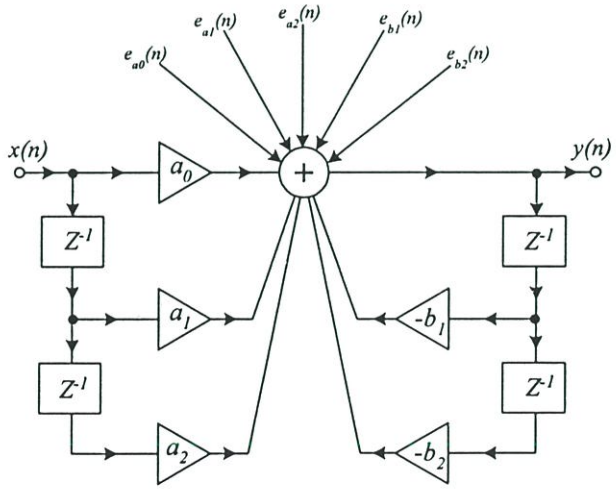
เมื่อ $c_i x(n)$ และ $e(n)$ เป็น ผลคูณแท้จริง และ ความผิดพลาดที่ได้ ตามลำดับ ซึ่งกลไกของตัวคูณสามารถแทนเป็นแบบจำลองได้ตามรูปที่ 5.3 (a) เมื่อ $e(n)$ เป็นแหล่งกำเนิดสัญญาณรบกวน



(a)



(b)



(c)

รูปที่ 5.3 การควอนไตซ์ของผลคูณ (a) แบบจำลองสัญญาณรบกวนของตัวคูณ, (b) โครงสร้างวงจรกรอง IIR อันดับสอง, (c) แบบจำลองสัญญาณรบกวนของวงจรกรอง IIR อันดับสอง

ถ้าพิจารณาโครงสร้างของวงจรกรองจากรูปที่ 5.3 (b) และสมมติให้เป็นพีชคณิตแบบกำหนดจุดแล้ว ที่แต่ละตัวคูณสามารถถูกแทนด้วยแบบจำลองรูป 5.3 (a) แสดงใหม่ได้ดังรูป 5.3 (c) ซึ่ง ควอนไตซ์ของผลคูณได้จากการปัดเศษ และสัญญาณรบกวน $e_i(n)$ แต่ละค่าสามารถพิจารณาโดยกระบวนการสุ่มด้วยการกระจายตัวของโอกาสรูปแบบสม่ำเสมอคือ

$$p(e_i;n)=\begin{cases} 1/q & for \quad -\frac{q}{2} \le e_i(n) \le \frac{q}{2} \\ 0 & otherwise \end{cases} \tag{5.16}$$

5.5 การสั่นปะปน (Parasitic Oscillations)

การสั่นปะปนเกิดจากความไม่เป็นเชิงเส้นของจำนวนที่แทนด้วยรูปแบบที่มีความยาวคำจำกัดมีชื่อเรียกอีกอย่างว่า ลิมิตไซเคิล (Limit Cycles) การสั่นปะปนมีสาเหตุที่สำคัญ 3 สาเหตุคือ เกิดจากการล้น เกิดจากการปัดเศษหรือการตัดปลาย และเกิดจากการที่ระบบมีสัญญาณขาเข้าที่เป็นคาบ รวมถึงยังมีสาเหตุมาจากความไม่เป็นเชิงเส้นหลายชนิดที่ให้ผลพร้อมๆ กัน รูปแบบของการสั่นปะปนสามารถจัดได้ดังนี้

5.5.1 การสั่นขณะที่สัญญาณขาเข้าเป็นศูนย์ (Zero-Input Oscillations)

ในระบบที่เป็นเชิงเส้น (Linear System) ถ้าระบบมีสัญญาณขาเข้าที่มีค่าเป็นศูนย์ ระบบจะให้สัญญาณขาออกที่มีค่าเข้าสู่ศูนย์ด้วย ความไม่เป็นเชิงเส้นของระบบที่มีความยาวคำจำกัดทำให้ระบบเกิดการสั่นชนิดหนึ่งออกมาแม้ว่าสัญญาณขาเข้าจะมีค่าเป็นศูนย์ เรียกการสั่นชนิดนี้ว่า ลิมิตไซเคิล ขณะที่สัญญาณขาเข้าเป็นศูนย์ (Zero-Input Limit Cycles) การสั่นชนิดนี้จะไปรบกวนระบบเป็นอย่างมาก โดยเฉพาะในงานประมวลผลสัญญาณเสียง เช่น งานประมวลผลคำสนทนาหรือดนตรี คือเมื่ออยู่ในช่วงระหว่างบทสนทนาหรือช่วงที่สัญญาณขาเข้าเงียบลง ความไม่เป็นเชิงเส้นจะให้สัญญาณขาออกที่มีการสั่นชนิดนี้ออกมา ลักษณะทางขนาดและสเปกตรัมของการสั่นขณะที่สัญญาณขาเข้าเป็นศูนย์นี้ขึ้นอยู่กับการคงค่าข้อมูลในองค์ประกอบการหน่วง (Delay Elements)ในขณะที่สัญญาณขาเข้ามีค่าลดลงสู่ศูนย์

การลดปัญหาการเกิดลิมิตไซเคิลขณะที่สัญญาณขาเข้าเป็นศูนย์กระทำได้ด้วยวิธีการแทนข้อมูลให้มีความยาวคำมากขึ้น หรือใช้วิธีการตัดบิตที่มีนัยสำคัญต่ำที่สุดของข้อมูลออกไป

5.5.2 การสั่นจากการล้น (Overflow Oscillations)

ในขณะประมวลผล เมื่อสัญญาณมีขนาดเกินช่วงจำกัดที่จะสามารถแทนค่าได้ จะมีการล้นเกิดขึ้นการระบุค่าข้อมูลจึงผิดพลาดไป ผลที่เกิดจากข้อผิดพลาดนี้ทำให้ระบบผลิตการสั่นออกมาทั้งในระดับรุนแรงและในระดับต่ำ การสั่นอย่างรุนแรงมักจะเกิดในงานที่มีการแทนข้อมูลในรูปแบบส่วนเติมเต็ม 2 เพราะว่าช่วงระหว่างข้อมูลค่าสูงสุดและต่ำสุดมีความแตกต่างกันมาก เราสามารถแก้ไขผลจากการล้นได้โดยการใช้การระบุจำนวนที่มีสภาวะอิ่มตัว (Saturation Arithmetic) ซึ่งเป็นการจำกัดค่าของข้อมูลไม่ให้เกินช่วงสูงสุดและต่ำสุด ส่งผลให้ขนาดของการสั่นลดลงแต่ก็จะไม่สามารถขจัดออกไปได้อย่างสมบูรณ์ อีกวิธีการหนึ่งคือการขยายข้อมูลเพิ่มเติมให้มีบิตขยาย (Extension Bits) ซึ่งเป็นการขยายข้อมูลเพิ่มเติมในส่วนที่มีนัยสำคัญสูงสุดเพื่อรองรับการเกินช่วงของข้อมูลในระหว่างการคำนวณภายในวงจรกรอง เมื่อเสร็จสิ้นการคำนวณจึงตัดให้มีขนาดเท่าเดิม

5.5.3 การสั่นจากสัญญาณคาบขาเข้า (Periodic Input Oscillations)

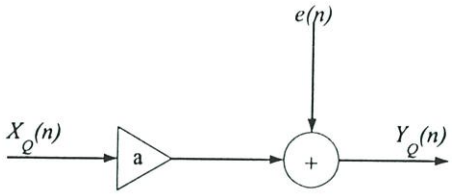
การสั่นชนิดนี้เกิดขึ้น เมื่อสัญญาณขาเข้ามีลักษณะเป็นคาบอย่างสมบูรณ์โดยระบบจะให้สัญญาณขาออกที่มีองค์ประกอบ ฮาร์โมนิค และฮาร์โมนิคย่อยแฝงอยู่ โดยองค์ประกอบเหล่านั้นจะมีความถี่เป็นสัดส่วนกับความถี่ของสัญญาณขาเข้า

5.6 ข้อผิดพลาดจากการปัดเศษ (Round-Off Errors)

ในการคำนวณข้อมูลที่ใช้การแทนเลขโดยตรง ปัญหาเกี่ยวกับการเกิดการล้นในการบวกและการคูณมักจะถูกจัดการได้โดยง่าย อย่างไรก็ดีผลที่ได้จากการคูณที่ใช้การแทนเลขโดยตรงนี้ต้องมีการถูกควอนไตซ์โดยการปัดเศษหรือตัดปลาย จึงมีข้อผิดพลาดปรากฏเป็นสัญญาณรบกวนปะปนมากับสัญญาณขาออก คุณสมบัติของสัญญาณรบกวนนี้ขึ้นอยู่กับปัจจัยหลายชนิดเช่น ประเภทของความไม่เป็นเชิงเส้น โครงสร้างของวงจรกรอง ลักษณะการแทนตัวเลข การแทนจำนวนลบ และคุณสมบัติของสัญญาณขาเข้า

แบบจำลองเชิงเส้นที่ใช้ในการคำนวณข้อผิดพลาดจากการปัดเศษสำหรับวงจรกรองดิจิตอลที่ใช้การแทนตัวเลขโดยตรง แสดงดังรูปที่ 5.4 โดยอาศัยหลักการแทนวงจรคูณขนาด n บิตด้วย วงจรคูณในอุดมคติ และแหล่งกำเนิดสัญญาณรบกวนที่เพิ่มเข้ามา ดังนั้นวงจรกรองที่มีความยาวคำจำกัดที่มีความไม่เป็นเชิงเส้นจากการควอนไตซ์จึงถูกจัดให้เป็นเชิงเส้นได้ ทำให้สามารถใช้หลักการซ้อนทับ (Superposition) ในการคำนวณผลอันเกิดจากสัญญาณรบกวน จากรูปที่ 5.4 สามารถแสดงได้ดังสมการต่อไปนี้

$$Y_Q(n) = a\,X_Q(n) + e(n) \tag{5.17}$$



รูปที่ 5.4 แบบจำลองสัญญาณรบกวนที่เป็นเชิงเส้น

โดยปกติ จะกำหนดให้ *e(n)* มีลักษณะเป็นสัญญาณรบกวนขาว (White Noise) ฟังก์ชันความหนาแน่น (Density Function) ของสัญญาณรบกวนมักถูกแทนด้วย ฟังก์ชันสี่เหลี่ยม (Rectangular Function) ที่เป็นฟังก์ชันไม่ต่อเนื่อง (Discrete) โดยจะมีค่าเป็นช่วงๆเฉพาะเวลาที่มีการตัดบิตบางบิต ในขั้นตอนการควอนไตซ์ ความแปรปรวน (Variance) ของแหล่งกำเนิดสัญญาณรบกวนเป็นดังสมการที่ (5.18)

$$\sigma_e^2 = k_e\left(1-Q_c^2\right)\sigma^2 \tag{5.18}$$

โดย

$$k_e = \begin{cases} 1 & for \quad \text{rounding or truncation} \\ 4-\dfrac{6}{\pi} & for \quad \text{magnitude truncation} \end{cases} \tag{5.19}$$

และ

$$\sigma^2 = \frac{Q^2}{12} \tag{5.20}$$

เมื่อ Q คือขนาดขั้นของการควอนไตซ์ข้อมูล

Q_c คือขนาดขั้นของการควอนไตซ์สัมประสิทธิ์

สำหรับวงจรกรองที่แทนตัวเลขโดยตรง แหล่งกำเนิดสัญญาณรบกวนตัวที่ i จะส่งผลให้เป็นสัญญาณรบกวนขาออกที่มีค่าความแปรปรวนเป็นดังนี้

$$\sigma_{yi}^{\ 2} = \sigma_i^{\ 2}\sum_{n=0}^{\infty} g_i(n)^2 \tag{5.21}$$

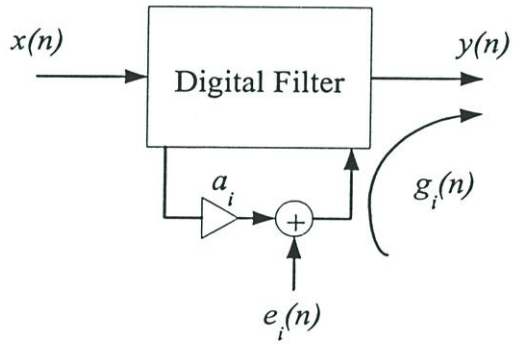
เมื่อ $g_i(n)$ คือ ผลตอบสนองอิมพัลส์ที่วัดจากแหล่งกำเนิดสัญญาณรบกวนตัวที่ i ไปยังช่องสัญญาณขาออกของวงจรกรอง โดยพิจารณาตามรูปที่ 5.5

เราสามารถทำการคำนวณค่าความแปรปรวนรวมของสัญญาณรบกวนขาออกทั้งหมดได้ โดยการนำสัญญาณ รบกวนขาออกที่เกิดจากแหล่งกำเนิดสัญญาณรบกวนแต่ละตัวมารวมกันจะได้ผลเป็นดังสมการที่ (5.22)

ตามรูปที่ 5.5 เป็นโครงสร้างแบบจำลองสำหรับการแทนแหล่งหรือจุดที่มีการควอนไตซ์โดยส่วนใหญ่จะเกิน ณ. ตำแหน่งค่าสัมประสิทธิ์ของตัวกรอง หากให้แหล่งกำเนิดควอนไตซ์มีจำนวน i ตำแหน่ง

$$\sigma_{ytot}^{\ 2} = \sum_{i=0}^{M} \sigma_{yi}^{\ 2} \tag{5.22}$$

เมื่อ M คือจำนวนจุดที่มีการควอนไตซ์ทั้งหมดในวงจรกรอง



รูปที่ 5.5 แบบจำลองสัญญาณรบกวนสำหรับวงจรกรองดิจิตอล

5.7 การประยุกต์ใช้งานของการจัดรูปสเปกตรัมความผิดพลาด

ทางเลือกหนึ่งของการลดระดับของสัญญาณรบกวนปัดเศษที่เอาท์พุท (Round-off noise) ก็คือ การจัดรูปสเปกตรัมความผิดพลาด (Error Spectrum Shaping: ESS) หรือ การป้อนกลับความผิดพลาด (Error Feedback: EF) เป็นเทคนิคที่เกี่ยวข้องกับการสร้างสัญญาณความผิดพลาดปัดเศษ (round-off error signal) และการประยุกต์ใช้การป้อนกลับสัญญาณเพื่อควบคุมและจัดการกับสัญญาณรบกวนที่เอาท์พุท จึงนำมาซึ่งการเพิ่มฮาร์ดแวร์ โดยจะเพิ่มขึ้นเป็นสัดส่วนกับจำนวนตัวบวกในโครงสร้างนั้นๆ อย่างไรก็ตามเพียงแค่โครงสร้างวงจรกรองที่เอาท์พุทของตัวคูณทุกๆตัว

เป็นอินพุทให้กับตัวบวกเพียงตัวเดียว ก็เหมาะกับการประยุกต์ใช้เทคนิค ESS นี้แล้ว เช่น โครงสร้างวงจรกรองโดยตรง (Direct form)

การประยุกต์ใช้ ESS กับโครงสร้างแบบโดยตรงสามารถแสดงดังรูปที่ 5.6 โดยสัญญาณและค่าสัมประสิทธิ์ถูกสมมุติให้เป็นรูปแบบกำหนดจุด (Fixed Point) ใช้จำนวน L บิตสำหรับขนาดและอีกหนึ่งบิตสำหรับแสดงเครื่องหมาย โดยที่แต่ละตัวบวก A_1 และ A_2 สามารถบวกผลคูณขนาด 2L บิตได้ผลรวมขนาด 2L บิต ตัวควอนไตซ์ Q_1 จะทำการปัดเศษเอาท์พุทของตัวบวก A_1 ให้มีขนาด L บิต และจะทำการสร้างค่าปรับขนาดควอนไตซ์ผิดพลาดขึ้นในเวลาเดียวกัน แล้วป้อนกลับมายังตัวบวก A_1 โดยผ่านเน็ตเวอร์คย่อย β ตัวควอนไตซ์ Q_2 อีกตัวหนึ่งจะทำการปรับขนาดค่าลงและปัดเศษเอาท์พุทของตัวบวก A_2 เป็น 2L บิต ค่าปรับขนาดแฟกเตอร์ที่เหมาะสมสำหรับเน็ตเวอร์คย่อย β มี 2^L ค่า ซึ่งบิตที่นำหน้า L บิต ของการควอนไตซ์ผิดพลาดมีค่าเป็นศูนย์ ค่าคงที่ λ ถูกใช้เพื่อทำการปรับขนาดค่าอินพุทของตัวควอนไตซ์ Q_1 โดยสมมุติให้สัญญาณ L_2 เป็นการปรับขนาด ดังนั้นจะได้ว่า

$$\lambda = \frac{1}{\|H\|_2^2} \tag{5.23}$$

เมื่อกำหนดให้

$H(z)$ คือ ทรานสเฟอร์ฟังก์ชันของโครงสร้างในรูป 5.6 (a)

ค่าคงที่ λ ในสมการที่ (5.23) เป็นการหาส่วนกลับของค่านอมัลไลซ์ทรานสเฟอร์ฟังก์ชันยกกำลังสอง

แบบจำลองของสัญญาณรบกวนแสดงในรูปที่ 5.6(b) เมื่อพิจารณาความผิดพลาดจากการควอนไตซ์แล้ว สามารถแสดงได้ดังรูปในรูปที่ 5.6(c) เมื่อ $-q_i/2 \le e_i(n) \le q_i/2$ และ $q_2 = 2^{-2L}$ ดังนั้นกำลังงานสเปกตรัมความหนาแน่น (Power Spectrum Density: PSD) ของสัญญาณ $e_1(n)$ และ $e_2(n)$ จะหาได้จาก

$$S_{e_i}(z) = \sigma_{e_i}^2 = \frac{q_i^2}{12} \tag{5.24}$$

ซึ่งกำลังงานสเปกตรัมความหนาแน่นของสัญญาณรบกวนเอาท์พุทสามารถหาได้จาก

$$S_n(z) = \sum_{i=1}^{2} \frac{q_i^2}{12} H_i(z) H_i(z^{-1}) \tag{5.25}$$

จากรูปที่ 5.6 (c) เราสามารถพิจารณาได้ว่า

$$H_1(z) = \frac{1}{\lambda}\left(\frac{z^2 + \beta_1 z + \beta_0}{z^2 + b_1 z + b_0}\right) \tag{5.26}$$

และ

$$H_2(z) = \frac{1}{\lambda\left(z^2 + b_1 z + b_0\right)} \tag{5.27}$$

โดยที่ $H_1(z)$ และ $H_2(z)$ เป็นทรานสเฟอร์ฟังก์ชันจากแหล่งสัญญาณรบกวน $e_1(n)$ และ $e_2(n)$ ต่อเอาท์พุท ตามลำดับ ค่าจำนวนของกำลังสัญญาณรบกวนเอาท์พุทจะเท่ากับ ออโตคอรีเลชันของเอาท์พุทประมาณได้เป็นดังสมการ

$$\begin{aligned} r_n(0) \quad = \sigma_n^2 \quad &= \frac{1}{2\pi j}\oint_{\Gamma} S_n(z) z^{-1} dz \\ &= \frac{1}{2\pi j}\oint_{\Gamma} \sum_{i=1}^{2} \frac{q_i^2}{12} H_i(z) H_i(z^{-1}) z^{-1} dz \\ &= \sum_{i=1}^{2} \frac{q_i^2}{12} \left\| H_i \right\|_2^2 \end{aligned} \tag{5.28}$$

สำหรับสัญญาณสุ่มอินพุทที่มีแอมปลิจูดเป็นรูปแบบในช่วง (-1, 1) เราจะได้ว่า $r_x(k) = \sigma_x^2 = \frac{1}{3}$ ดังนั้นกำลังที่เอาท์พุทจะได้ว่า

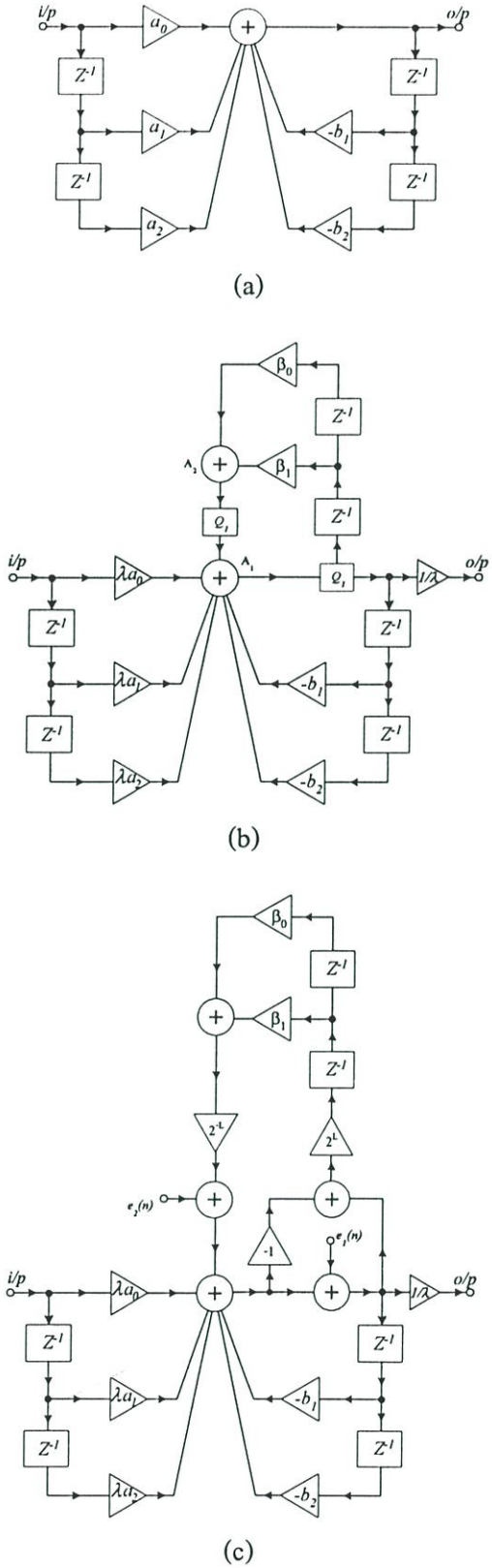
$$r_y(0) = \sigma_y^2 = \frac{1}{3}\left\| H \right\|_2^2 \tag{5.29}$$

ดังนั้นแล้วจากสมการที่ (5.23) และ (5.26) ถึง (5.29) อัตราส่วนสัญญาณต่อสัญญาณรบกวนสามารถหาได้จาก

$$\text{SNR} = \frac{\sigma_y^2}{\sigma_n^2} = \frac{4 \times 2^{2L}}{\left\| \frac{z^2 + \beta_1 z + \beta_0}{z^2 + b_1 z + b_0} \right\|_2^2 + 2^{-2L} \left\| \frac{1}{z^2 + b_1 z + b_0} \right\|_2^2} \tag{5.30}$$

ถ้าตัวแปร β_1 และ β_2 ถูกเลือกให้เท่ากับค่า b_1 และ b_2 ตามลำดับ จะทำให้ SNR มีค่าสูงสุด*

*นำเสนอโดย Higgins และ Munson 1982



รูปที่ 5.6 การพิจารณาแหล่งกำเนิดความผิดพลาดและการประยุกต์ใช้ ESS;
(a) วงจรกรองป้อนกลับอันดับสองแบบโดยตรง, (b) การประยุกต์ใช้ ESS และ
(c) แบบจำลองสัญญาณรบกวน

5.8 โครงสร้างของวงจรกรองเชิงเลข DA ชนิด IIR อันดับสองและ ESS [6, 8-9]

จากสมการที่ (4.20) และ (4.21) เราสามารถพิจารณาผลของความผิดพลาดจากความผิดพลาดจากการปัดเศษ และชดเชยความผิดพลาดด้วยการใช้ ESS ในการแก้ไขความผิดพลาดได้

$$Y_n = \sum_{j=1}^{B} 2^{-j} F(X_n^j, X_{n-1}^j, X_{n-2}^j, Y_{n-1}^j, Y_{n-2}^j) - F(X_n^0, X_{n-1}^0, X_{n-2}^0, Y_{n-1}^0, Y_{n-2}^0) \qquad (4.20)$$

เมื่อฟังก์ชัน F (.) นิยามตามที่กล่าวไปแล้วในบทที่ 4 คือ

$$F(X_n^j, X_{n-1}^j, X_{n-2}^j, Y_{n-1}^j, Y_{n-2}^j) = a_0 X_n^j + a_1 X_{n-1}^j + a_2 X_{n-2}^j - b_1 Y_{n-1}^j - b_2 Y_{n-2}^j \qquad (4.21)$$

โดยพิจารณาค่าของฟังก์ชัน F (.) ใหม่ดังสมการ

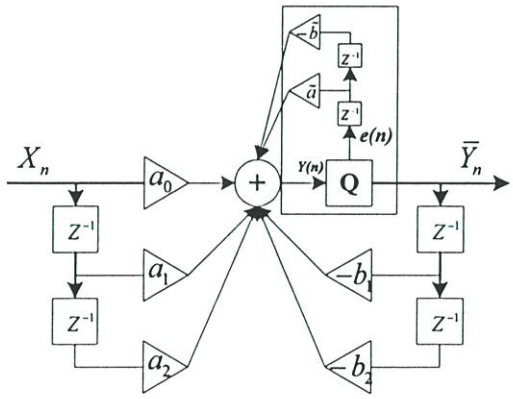
$$Y_n = \sum_{j=1}^{B} 2^{-j} \left\{ F_q(X_n^j, \ldots) + F_\varepsilon(X_n^j, \ldots) \right\} - \left\{ F_q(X_n^0, \ldots) + F_\varepsilon(X_n^0, \ldots) \right\} \qquad (5.31)$$

เมื่อ $F_q(.)$ คือ ค่าของฟังก์ชัน $F(.)$ ที่ถูกควอนไตซ์ และ $F_\varepsilon(.)$ เป็นค่า noise function (ที่ขึ้นอยู่กับการอ้างอิงตำแหน่ง) ถ้า $\bar{a}_0, \bar{a}_1, \bar{a}_2, \bar{b}_1$, และ $\bar{b}_2$ เป็นค่าสัมประสิทธิ์ที่ถูกควอนไตซ์ แล้วจะได้ว่า

$$F_q(.) = \bar{a}_0 X_n^j + \bar{a}_1 X_{n-1}^j + \bar{a}_2 X_{n-2}^j - \bar{b}_1 Y_{n-1}^j - \bar{b}_2 Y_{n-2}^j \qquad (5.32)$$

และ

$$F_\varepsilon(.) = (a_0 - \bar{a}_0) X_n^j + (a_1 - \bar{a}_1) X_{n-1}^j + (a_2 - \bar{a}_2) X_{n-2}^j - (b_1 - \bar{b}_1) Y_{n-1}^j - (b_2 - \bar{b}_2) Y_{n-2}^j \qquad (5.33)$$



รูปที่ 5.7 วงจรกรองเชิงเลข IIR อันดับที่ 2 กับโครงสร้างการป้อนกลับของตัวควอนไตซ์อันดับที่ 2

สำหรับการป้อนกลับของตัวควอนไตซ์อันดับที่ 2 ใน บล็อกเส้นประ (EF block) แสดงตามรูปที่ 5.7 ค่าความผิดพลาดที่เกิดจากการปัดเศษในตัวควอนไตซ์สามารถพิจารณาได้ดังนี้

$$\varepsilon_n = Y_n - \overline{Y}_n = e(n) - \tilde{a}e(n-1) + \tilde{b}e(n-2) \tag{5.34}$$

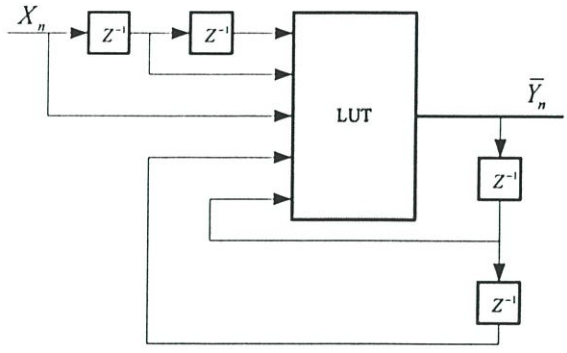
เมื่อความผิดพลาด *e(n-1)* และ *e(n-2)* ในสมการที่ (5.35) สามารถทำการคำนวณไว้ก่อนได้ และถูกประมาณค่าจากสมการที่ (5.33) ดังแสดงข้างล่าง

$$\begin{aligned} e(n-1) &= (a_0 - \overline{a}_0)X_{n-1} + (a_1 - \overline{a}_1)X_{n-2} - (b_1 - \overline{b}_1)Y_{n-2} \\ e(n-2) &= (a_0 - \overline{a}_0)X_{n-2} + (a_2 - \overline{a}_2)Y_{n-2} \end{aligned} \tag{5.35}$$

โดยปกติแล้วค่าของ $\tilde{a}$ และ $\tilde{b}$ ในสมการที่ (5.34) หาค่าได้ลำบาก ดังนั้น $\tilde{a}$ และ $\tilde{b}$ ในรูปแบบจำนวนเต็มสามารถถูกใช้ในการหาค่าถ่วงน้ำหนัก K_a, K_b, K_c ที่จะให้ค่าสัญญาณรบกวนต่ำๆ ซึ่งทำให้ฟังก์ชัน $F_\varepsilon(.)$ กลายเป็น

$$\begin{aligned} F_\varepsilon(.) = {} & K_a\left\{(a_0 - \overline{a}_0)X_{n-1}^j + (a_1 - \overline{a}_1)X_{n-2}^j - (b_1 - \overline{b}_1)Y_{n-2}^j\right\} \\ & + K_b\left\{(a_0 - \overline{a}_0)X_{n-2}^j + (a_2 - \overline{a}_2)Y_{n-2}^j\right\} + K_c(a_2 - \overline{a}_2)Y_{n-1}^j \end{aligned} \tag{5.36}$$

เมื่อ K_a, K_b และ K_c เป็นค่าแฟกเตอร์จำนวนเต็มถ่วงน้ำหนัก ซึ่งลักษณะของวงจรกรองที่ต่างชนิดกันจะให้ค่าที่ต่างกันด้วย ตอนนี้ค่าในหน่วยความจำของวงจรกรองเชิงเลขจะเป็นผลรวมของค่าที่เกิดจากการปัดเศษจากสมการที่ (4.20) และค่าความผิดพลาดที่ได้คำนวณไว้ก่อนจากสมการที่ (5.34) ตามลำดับ ซึ่งสามารถแสดง บล็อกไดอะแกรมของวงจรกรองเชิงเลขได้ ดังรูปที่ 5.8 ซึ่งจะมีรูปแบบที่เหมือนกันกับบทความของ Peled and Liu ได้นำเสนอไว้[6] จะมีความแตกต่างกันเพียงค่าที่เก็บในหน่วยความจำ ซึ่งโครงสร้างของวงจรกรองรวมทั้งส่วนของค่าในหน่วยความจำ LUT สามารถทำการเขียนโปรแกรมในรูปแบบภาษาVHDL จะทำให้วงจรกรองมีประสิทธิภาพในด้านความสามารถโปรแกรมค่าลักษณะเฉพาะของวงจรกรองได้ง่ายขึ้น



รูปที่ 5.8 บล็อกไดอะแกรมของวงจรกรอง DA-IIR อันดับที่ 2 รวมวิธี ESS

จะเห็นได้ว่าค่าของบิตที่เกิดการปัดเศษจะถูกทำการชดเชยแล้วเก็บในหน่วยความจำเรียบร้อยจึงทำให้ขนาดความยาวคำของ แอคคิวมูเลเตอร์คงค่าเดิมได้ แต่ถ้าหากต้องการความละเอียดของวงจรกรองมากยิ่งขึ้นก็สามารถทำการขยายบิตได้

5.9 บทสรุป

ในบทนี้ได้บรรยายถึงผลกระทบของความยาวคำจำกัด ซึ่งมีผลกระทบโดยตรงต่อระบบวงจรกรองเชิงเลข หรือระบบการประมวลผลสัญญาณเชิงเลขใดๆ โดยได้อธิบายรูปแบบการแทนข้อมูลในลักษณะความยาวคำที่จำกัด การควอนไตซ์ของสัญญาณหรือข้อมูล การตัดปลาย และการปัดเศษ ของข้อมูลความยาวคำจำกัด ไม่ว่าจะอยู่ในรูปแบบจำนวน Signed Magnitude, จำนวนเลขส่วนเติมเต็มหนึ่ง หรือเลขส่วนเติมเต็มสองก็ตาม จะมีรูปแบบเงื่อนไขบางประการที่แตกต่างกันบ้าง ระบบจำนวนเหล่านี้ล้วนส่งผลต่อความเป็นอุดมคติของระบบเชิงเลข ทำให้เกิดความคลาดเคลื่อนไปค่าหนึ่ง อาจเรียกเป็นความผิดพลาดที่เกิดขึ้น ซึ่งในบทนี้เองก็ได้กล่าวถึงแนวคิดในการแก้ไขความผิดพลาดที่เกิดขึ้นในระบบเชิงเลข โดยได้นำเสนอวิธีการแก้ความผิดพลาดด้วยการใช้หลักการจัดสเปกตรัมความผิดพลาด (ESS) นอกจากนี้ยังได้กล่าวถึงแหล่งที่ทำให้เกิดความผิดพลาดอื่นๆอีกไว้พอสังเขป เมื่อทราบพื้นฐานสำคัญต่างๆแล้ว บทต่อไปจะได้นำเสนอขั้นตอนในการออกแบบตัวกรองเชิงเลขในงานวิจัยนี้สำหรับวิทยานิพนธ์นี้โดยละเอียด

บทที่ 6

การออกแบบวงจรกรองเชิงเลขความถี่ต่ำ โดยใช้วิธีคณิตศาสตร์แบบการกระจาย

6.1 บทนำ

ในบทนี้จะได้นำเสนอรายละเอียดของแต่ละขั้นตอนในการออกแบบวงจรกรองเชิงเลขแบบรีเคอร์ซีฟอันดับที่ 2 โดยใช้วิธีคณิตศาสตร์แบบการกระจาย ซึ่งในขั้นต้นของการออกแบบวงจรกรองนี้ ต้องเริ่มจากการหาฟังก์ชันถ่ายโอนของวงจรกรองที่สอดคล้องกับข้อกำหนดทางความถี่ที่ต้องการก่อน อันได้แก่ อันดับของวงจรกรอง (Filter Order) ความถี่ตัด (Cut-off Frequency) ความถี่ของการสุ่ม (Sampling Frequency) ถ้าในรายละเอียด เช่น ความถี่ที่ขอบแถบผ่าน (Pass-band Edge Frequency) ความถี่ที่ขอบแถบหยุด (Stop-band Edge Frequency) การกระเพื่อมในแถบผ่าน (Pass-band Ripple) และการลดทอนในแถบหยุด (Stop-band Edge Frequency) เป็นต้น โดยใช้การประมาณค่าด้วยพหุนามต้นแบบชนิด บัทเทอร์เวิร์ธ เชบีเชฟ หรือ อิลลิปติค คำนวณหาค่าสัมประสิทธิ์ของฟังก์ชันถ่ายโอนออกมา เมื่อได้ฟังก์ชันถ่ายโอนของวงจร กรองอนาลอกต้นแบบแล้ว จะมีแนวทางสร้างเป็นวงจรกรองเชิงเลขได้ 2 แนวทางคือ ทำการสร้างวงจรอนาลอก LC ก่อนแล้วทำการแปลงอุปกรณ์อนาลอกต้นแบบให้เป็นอุปกรณ์ดิจิตอล หรืออีกแนวทางหนึ่งเป็นการแปลงจากฟังก์ชันถ่ายโอนโดยตรงสู่โดเมนดิจิตอล ซึ่งเป็นวิธีการที่ได้นำเสนอไว้ เพราะมีขั้นตอนที่เหมาะสมต่อการคำนวณด้วยคอมพิวเตอร์จึงสะดวกและประหยัดเวลาเป็นอย่างมาก

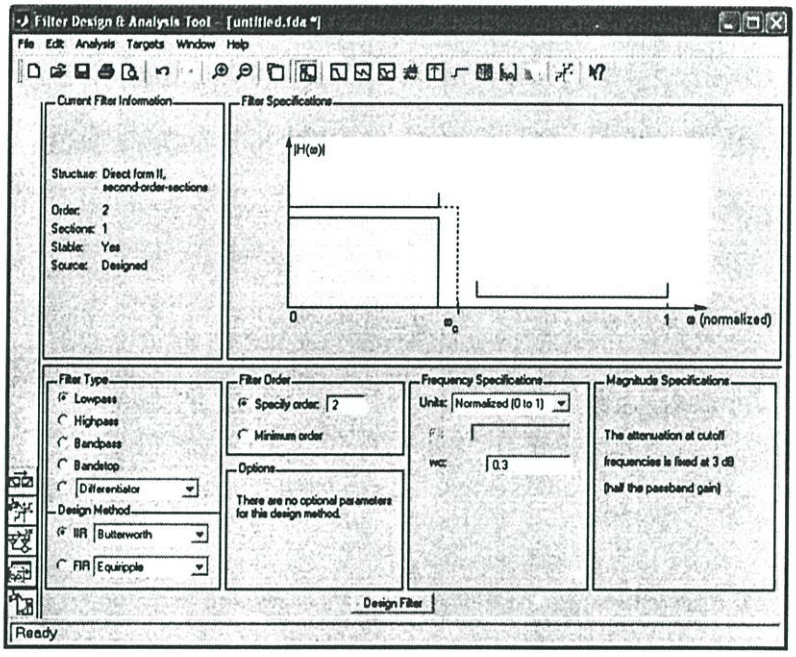
ค่าสัมประสิทธิ์ของวงจรกรองเชิงเลขที่คำนวณนั้นจะต้องทำการควอนไตซ์ (Quantization) เพื่อจัดรูปแบบของสัมประสิทธิ์ให้มีการแทนเลขโดยตรงได้ (Fixed-point) ถ้าจำนวนบิตของการควอนไตซ์มีมากจะทำให้วงจรมีขนาดใหญ่ แต่ถ้าทำการควอนไตซ์ด้วยจำนวนบิตที่น้อยเกินไปก็จะทำให้วงจรมีผลตอบสนองทางความถี่ผิดพลาดไปจากข้อกำหนด ดังนั้นในการคำนวณค่าต่างๆ ของวงจรกรองเชิงเลข เพื่อความสะดวกแล้วจะใช้โปรแกรม MATLAB6.x ทำการคำนวณรวมทั้งทำการวิเคราะห์ผลของความยาวคำจำกัดต่อวงจรกรองเชิงเลข นอกจากนี้การคำนวณค่าฟังก์ชัน LUT ของสัมประสิทธิ์ของวงจรกรองดังที่ได้กล่าวมานี้ จะใช้การเขียนโปรแกรม m-file ในการคำนวณบนโปรแกรม MATLAB จะได้ค่าฟังก์ชันของสัมประสิทธิ์สำหรับโครงสร้างวงจรกรองคณิตศาสตร์แบบการกระจาย จากนั้นจึงทำการออกแบบตามลักษณะฮาร์ดแวร์ของวงจรกรองคณิตศาสตร์แบบการกระจาย ด้วยการใช้เอฟพีจีเอในการสร้างเป็นวงจรจริง โดยทำการออกแบบด้วยโปรแกรม Xilinx ISE ขั้นตอนการออกแบบวงจรกรองเป็นดังรูปที่ 6.1

รูปที่ 6.1 ขั้นตอนการออกแบบวงจรกรองเชิงเลขโดยใช้วิธีคณิตศาสตร์แบบการกระจาย

6.2 การออกแบบวงจรกรองเชิงเลขชนิดคณิตศาสตร์แบบการกระจายด้วย MATLAB

ในส่วนนี้เป็นการแสดงวิธีการออกแบบวงจรกรองเชิงเลขความถี่ต่ำโดยใช้โปรแกรม FDA ใน MATLAB เพื่อคำนวณหาค่าสัมประสิทธิ์ของวงจรกรองเชิงเลข เมื่อทราบค่าอันดับของวงจรกรอง (N) และความถี่ตัด (f_c) หรือเมื่อทราบข้อกำหนดอย่างละเอียดคือ ความถี่ที่ขอบแถบผ่าน (f_p)

ความถี่ที่ขอบแถบหยุด (f_s) การกระเพื่อมในแถบผ่าน(Rp) การลดทอนในแถบหยุด (As) และความถี่ของการสุ่ม (f_t) โปรแกรมนี้มีรูปแบบแสดงตามรูปที่ 6.2 โดยมีตัวแปรต้นเป็นข้อกำหนดต่างๆ มีผลลัพธ์เป็นสัมประสิทธิ์ของวงจรกรอง



รูปที่ 6.2 โปรแกรม Filter Design & Analysis Tool บน MATLAB

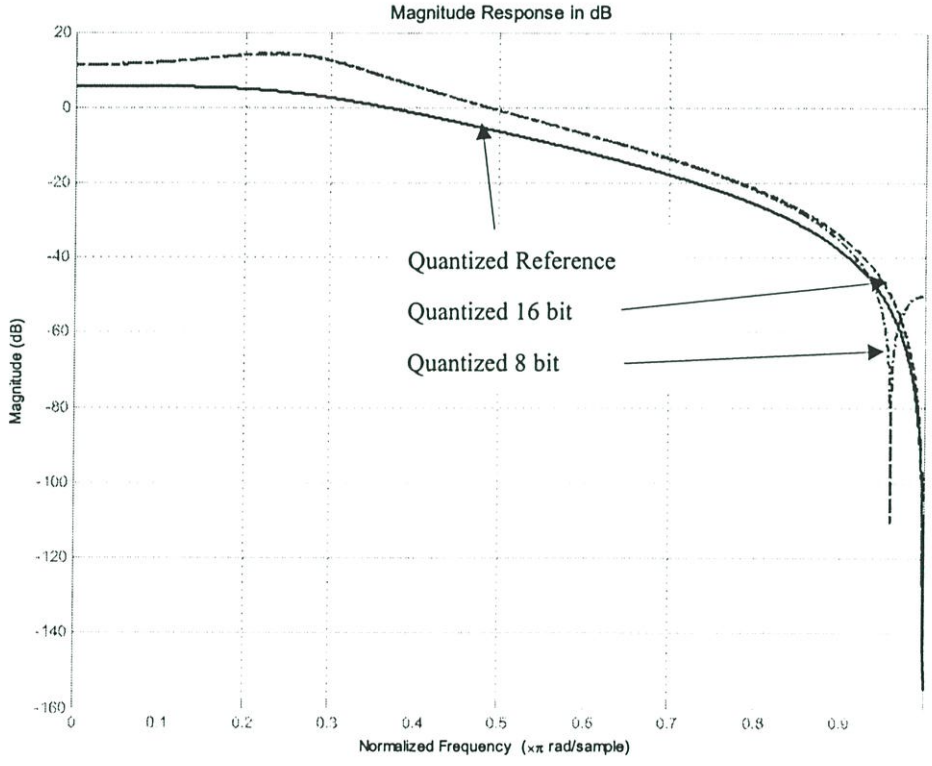
6.2.1 การเลือกกำหนดลักษณะเฉพาะของวงจรกรอง

ในการสร้างวงจรกรองเชิงเลขใดๆก็ตาม แต่ละวงจรกรองจะมีค่าความถี่ตัด (Cut off Frequency) เฉพาะค่าประจำวงจรกรองนั้นๆ ดังนั้นในการออกแบบทุกครั้งจึงจำเป็นต้องกำหนดคุณสมบัติของวงจรกรองก่อน โดยในที่นี้จะขอกล่าวแต่เฉพาะการออกแบบวงจรกรองโดย การใช้ค่าความถี่นอมัลไลซ์ (Normalized Frequency) เพื่อความยืดหยุ่นในการสร้างเป็นระบบวงจรกรองเชิงเลข เพราะปัจจัยของขีดจำกัดทางฮาร์ดแวร์ เช่น ความเร็วในการแปลงสัญญาณที่อุปกรณ์ ADC, DAC, ขนาดจำนวนบิตข้อมูล หรือแม้แต่อัลกอริทึมที่ใช้ในการคำนวณต่างๆ ล้วนมีผลต่อการสร้างเป็นวงจรทั้งสิ้น

วงจรกรองที่จะได้ทำการออกแบบสำหรับวิทยานิพนธ์นี้ กำหนดไว้ดังนี้

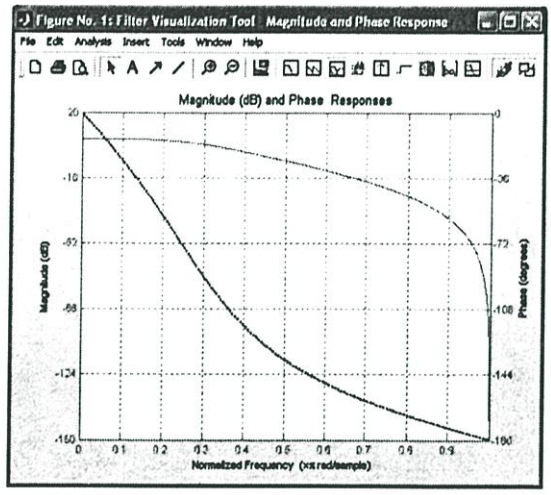
1) วงจรกรองเชิงเลขชนิด *IIR*
2) อันดับของวงจรกรอง *2*
3) ความถี่นอมัลไลซ์ $w_c = 0.3$
4) ประเภทการทำงาน *Low pass Filter*
5) วิธีการออกแบบ *แบบบัทเทอร์เวิร์ธ*

ทำการกำหนดค่าต่างๆลงในโปรแกรมแสดงดังรูปที่ 6.2 จากนั้นสั่งให้โปรแกรมทำการออกแบบวงจรกรอง โดยการปดปุ่ม "Design Filter" โปรแกรมจะทำการคำนวณค่าผลการตอบสนองทางขนาด, เฟส, สเปกตรัมความผิดพลาด,ตำแหน่งโพล-ซีโร่ ดังรูปที่ 6.3 ค่าต่างๆเหล่านี้สามารถถูกเรียกดูได้โดยการกดปุ่มบนเมนูของโปรแกรมเพื่อเลือกแสดงค่า ซึ่งช่วยให้การวิเคราะห์วงจรกรองทำได้รวดเร็วมาก และค่าผลการคำนวณต่างๆที่ได้ยังสามารถนำมาวิเคราะห์ปัญหา และแก้ไขปัญหาต่างๆที่เกิดขึ้นในการออกแบบได้โดยค่าผลลัพธ์ที่คำนวณได้นี้จะให้ความสนใจไปที่ค่าสัมประสิทธิ์ของวงจรกรองเป็นหลัก เพราะเป็นค่าลักษณะเฉพาะประจำของวงจรกรอง และยังมีความสำคัญต่อการวิเคราะห์ผลกระทบของความยาวบิตข้อมูลจำกัดอีกด้วย และยังต้องแปลงให้อยู่ในรูปแบบ LUT เพื่อให้สอดคล้องกับกระบวนการคณิตศาสตร์แบบกระจาย ซึ่งจะได้ถูกแยกออกแบบแต่ละส่วนประกอบต่อไป

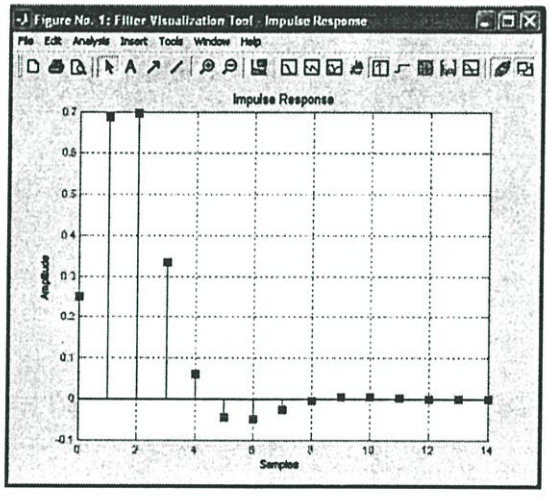


รูปที่ 6.3 การตอบสนองความถี่ของวงจรกรองเชิงเลขเมื่อใช้สัมประสิทธิ์ควอนไตซ์ 8 และ 16 บิต เทียบกับสัมประสิทธิ์อ้างอิง

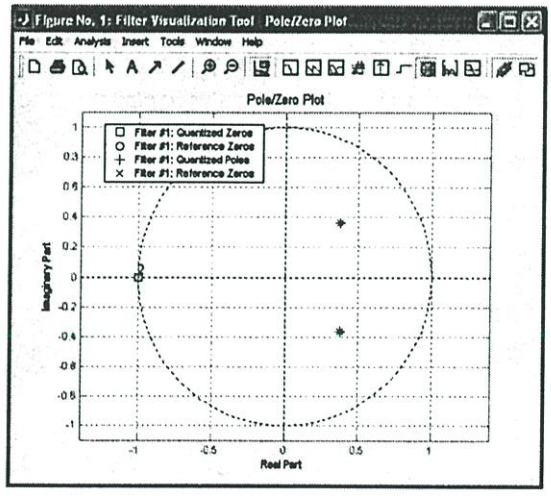
จากรูปที่ 6.3 จะเห็นได้ชัดเจนว่าผลของการควอนไตซ์สัมประสิทธิ์ที่จำนวนบิตน้อยกว่าจะให้ความผิดพลาดมากกว่าจำนวนบิตมากกว่า เมื่อเทียบกับค่าสัมประสิทธิ์อ้างอิง ซึ่งสอดคล้องตามทฤษฎีที่ได้กล่าวไว้แล้วในตอนต้น รูปที่ 6.4 – 6.8 เป็นการแสดงผลการจำลองการทำงานที่คำนวณได้จาก FDA ของสัมประสิทธิ์วงจรกรองอ้างอิง



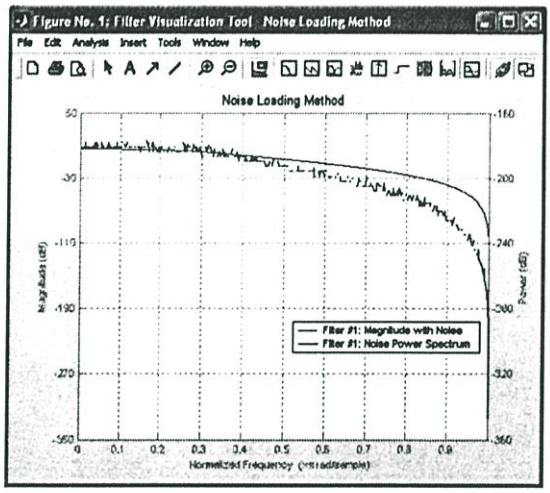
รูปที่ 6.4 การตอบสนองทางขนาดเทียบกับการตอบสนองทางเฟส



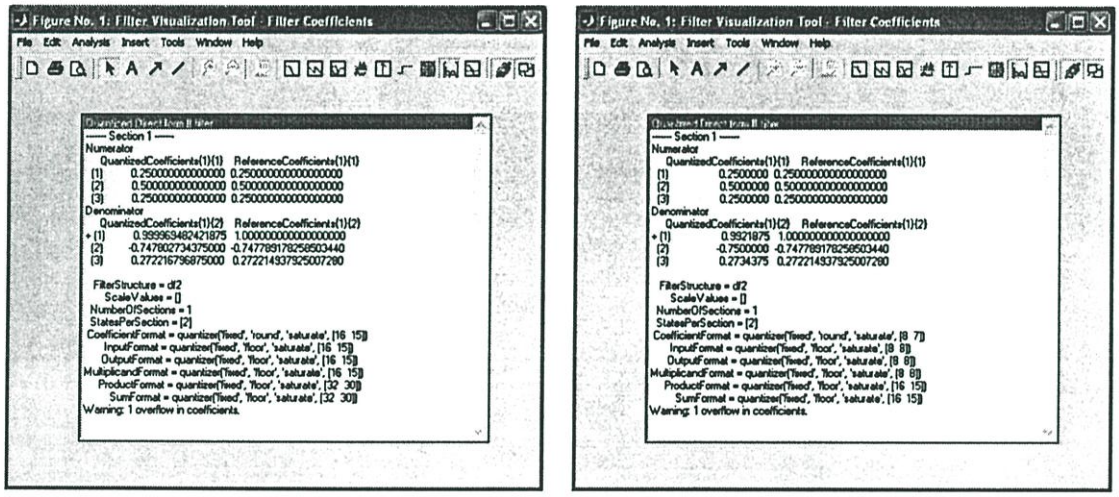
รูปที่ 6.5 การตอบสนองอิมพัลส์ของวงจรกรองเชิงเลขอ้างอิง



รูปที่ 6.6 ตำแหน่งโพลและซีโร่ของวงจรกรองเชิงเลข ส.ป.ส. อ้างอิง กับ ส.ป.ส. ควอนไตซ์



รูปที่ 6.7 สเปกตรัมกำลังของสัญญาณรบกวนเทียบกับการตอบสนองทางขนาด



(a) (b)

รูปที่ 6.8 การเปรียบเทียบค่าสัมประสิทธิ์อ้างอิงกับค่าสัมประสิทธิ์ควอนไตซ์ที่ขนาด 8 และ 16 บิต

(a) ค่าสัมประสิทธิ์อ้างอิงกับค่าสัมประสิทธิ์ควอนไตซ์ 16 บิต

(b) ค่าสัมประสิทธิ์อ้างอิงกับค่าสัมประสิทธิ์ควอนไตซ์ 8 บิต

6.2.2 การพิจารณารูปแบบของค่าสัมประสิทธิ์จำนวนบิตจำกัด

จากค่าสัมประสิทธิ์อ้างอิงและค่าสัมประสิทธิ์ ควอนไตซ์ที่คำนวณได้จากโปรแกรม FDA จะเป็นสัมประสิทธิ์ที่สอดคล้องกับรูปแบบวงจรกรองไบควอดราติก เราจะนำค่าสัมประสิทธิ์ที่ได้มาทำการพิจารณา ค่าสัมประสิทธิ์ที่ได้จากการปัดเศษขนาด 8 บิต (Rounding) และค่าสัมประสิทธิ์ที่ได้การตัดส่วน 8 บิต (Truncation) โดยสามารถแสดงได้ดังตารางที่ 6.1 ดังนั้นสมการฟังก์ชันถ่ายโอนอ้างอิงสามารถแสดงได้ดังนี้

$$H_{\text{Ref}}(z) = \frac{0.25000000000000000 + 0.50000000000000000z^{-1} + 0.25000000000000000z^{-2}}{1.00000000000000000 - 0.74778917825850300z^{-1} + 0.27221493792500700z^{-2}}$$

ตารางที่ 6.1 แสดงค่าสัมประสิทธิ์ของวงจรกรองเชิงเลขที่ทำการควอนไตซ์, ตัดส่วน และปัดเศษ

	ส.ป.ส.	ส.ป.ส. ควอนไตซ์ (16บิต)	ส.ป.ส. ควอนไตซ์ (8บิต)	ส.ป.ส. ตัดส่วน (8บิต)	ส.ป.ส. ปัดเศษ (8บิต)
ตัวแปรเศษ	a_0	0.250000000000000	0.2500000	0.2500000	0.2500000
	a_1	0.500000000000000	0.5000000	0.5000000	0.5000000
	a_2	0.250000000000000	0.2500000	0.2500000	0.2500000
ตัวแปรส่วน	b_0	0.999969482421875	0.9921875	0.9999694	0.9999695
	b_1	-0.747802734375000	-0.7500000	-0.7478027	-0.7478027
	b_2	0.272216796875000	0.2734375	0.2722167	0.2722168

จากตารางที่ 6.1 พิจารณาเฉพาะค่าสัมประสิทธิ์ขนาด 8 บิตที่ได้ ถึงแม้ว่าจะมีค่าทศนิยมที่ใกล้เคียงกัน แต่หากถูกสร้างขึ้นเป็นวงจรกรองเชิงเลขแล้ว จะให้ผลการตอบสนองความถี่ที่ไม่เหมือนกันตลอดช่วงการทำงาน แต่อาจมีลักษณะที่คล้ายกันได้ ค่าสัมประสิทธิ์ที่ได้มานี้จะถูกนำไปทำการแปลงให้อยู่ในรูปแบบเลขไบนารี่ส่วนเติมเต็มสอง (Signed 2's complement) ตามเงื่อนไขคณิตศาสตร์แบบการกระจาย ดังนั้นเพื่อความสะดวกในการสร้างเป็นตาราง LUT เราจะทำการเขียนโปรแกรมสำหรับรับค่าสัมประสิทธิ์เหล่านี้ แล้วทำการแปลงเลขทศนิยมเหล่านี้ให้อยู่ใน หลักการในการแปลงเลขฐานสองและการสร้างตาราง LUT สามารถสร้างขึ้นได้ตามลำดับ แสดงดังโฟลว์ชาร์ตรูปที่ 6.9 ซึ่งสามารถเขียนโปรแกรมในรูปแบบ m-file เพื่อให้สามารถทำการคำนวณค่าได้บนโปรแกรม MATLAB โปรแกรมเดียว แต่ก่อนที่จะศึกษาขั้นตอนตามรูปที่ 6.9 นั้นควรทราบถึงคำจำกัดความของการหาค่าสัมประสิทธิ์ควอนไตซ์และการแปลงเลขทศนิยมเป็นเลขส่วนเติมเต็มสองก่อน ซึ่งสามารถนิยามได้ว่า

$$C_Q = \text{round}\,(C_{\text{Ref}} \times 2^N)\,/2^N \qquad (6.1)$$

เมื่อ C_{Ref} คือ ส.ป.ส. อ้างอิง

C_Q คือ สัมประสิทธิ์ควอนไตซ์

N คือ จำนวนความยาวบิตข้อมูล

และ

$$\overline{A} = \text{dec2bin}\,(\,\text{round}\,(\,A*2^{N-1}), \text{N}) \qquad (6.2)$$

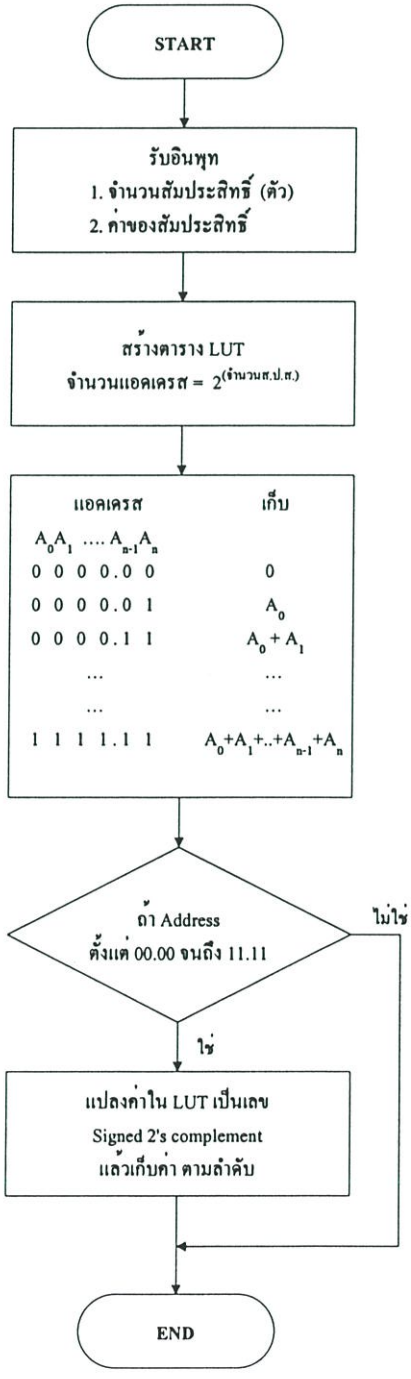
เมื่อให้

$\overline{A}$ คือ เลขส่วนเติมเต็มสองของทศนิยมที่กำหนด

A คือ เลขทศนิยมที่กำหนดมา

N คือ จำนวนบิตข้อมูล

เราทราบหลักการหาค่าสัมประสิทธิ์ควอนไตซ์และเลขส่วนเติมเต็มสองของทศนิยม แล้วจะช่วยทำให้การแปลงค่าไปสู่รูปแบบเลขส่วนเติมเต็มสองใน LUT สามารถคำนวณออกมาได้ง่ายขึ้น เพราะลักษณะการคำนวณตาราง LUT จะเป็นค่าของฟังก์ชันที่ขึ้นกับสัมประสิทธิ์ทั้งหมดโดยตรง โดยทำการคำนวณวนรอบ (Loop) ตามลำดับค่าแอดเดรสของ LUT พิจารณาตามรูปที่ 6.9



รูปที่ 6.9 โฟล์วชาร์ทแสดงการสร้างตาราง LUT และการแปลงฟังก์ชันเป็นเลขส่วนเติมเต็มสอง

6.2.3 ตาราง LUT สำหรับการแปลงสัมประสิทธิ์ของวงจรกรองเชิงเลข

ก่อนที่เราจะได้เริ่มทำการแปลงสัมประสิทธิ์ของวงจรกรองเชิงเลข เราจะศึกษาอย่างรอบคอบสำหรับค่าฟังก์ชันที่ได้ก่อน โดยฟังก์ชันนี้จะขึ้นกับสัมประสิทธิ์ของวงจรกรองเชิงเลขแทนด้วยสัญลักษณ์ F(.) ค่านี้จะถูกทำการเก็บลงไว้ในหน่วยความจำโดยมีสัญญาณแอดเดรส 5 สัญญาณเพื่อทำการอ้างอิงตำแหน่งข้อมูล คือ สัญญาณ x(n), x(n-1), x(n-2), y(n-1) และ y(n-2) ดังนั้นฟังก์ชัน F(.) จะมีค่าที่เป็นได้ ไม่ซ้ำกัน 2^5 =32 แอดเดรส และเราแทนข้อมูลด้วยขนาด 8 บิต ความจุของหน่วยความจำจึงมีขนาดเท่ากับ 32 x 8 = 256 บิต มีแอดเดรสเริ่มต้นคือ 00000 จนถึงแอดเดรส 11111 จากสมการที่ (4.21) คือ

$$F(x_n, x_{n-1}, x_{n-2}, y_{n-1}, y_{n-2}) = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} - b_1 y_{n-1} - b_2 y_{n-2}$$

ถ้าแทนสัมประสิทธิ์อ้างอิง a_0, a_1, a_2, b_1, b_2 ด้วยสัมประสิทธิ์แบบเลขส่วนเติมเต็มสอง แทนด้วย $\overline{a}_0, \overline{a}_1, \overline{a}_2, \overline{b}_1, \overline{b}_2$ จะดังสมการ (6.3)

$$F(x_n, x_{n-1}, x_{n-2}, y_{n-1}, y_{n-2}) = \overline{a}_0 x_n + \overline{a}_1 x_{n-1} + \overline{a}_2 x_{n-2} - \overline{b}_1 y_{n-1} - \overline{b}_2 y_{n-2} \qquad (6.3)$$

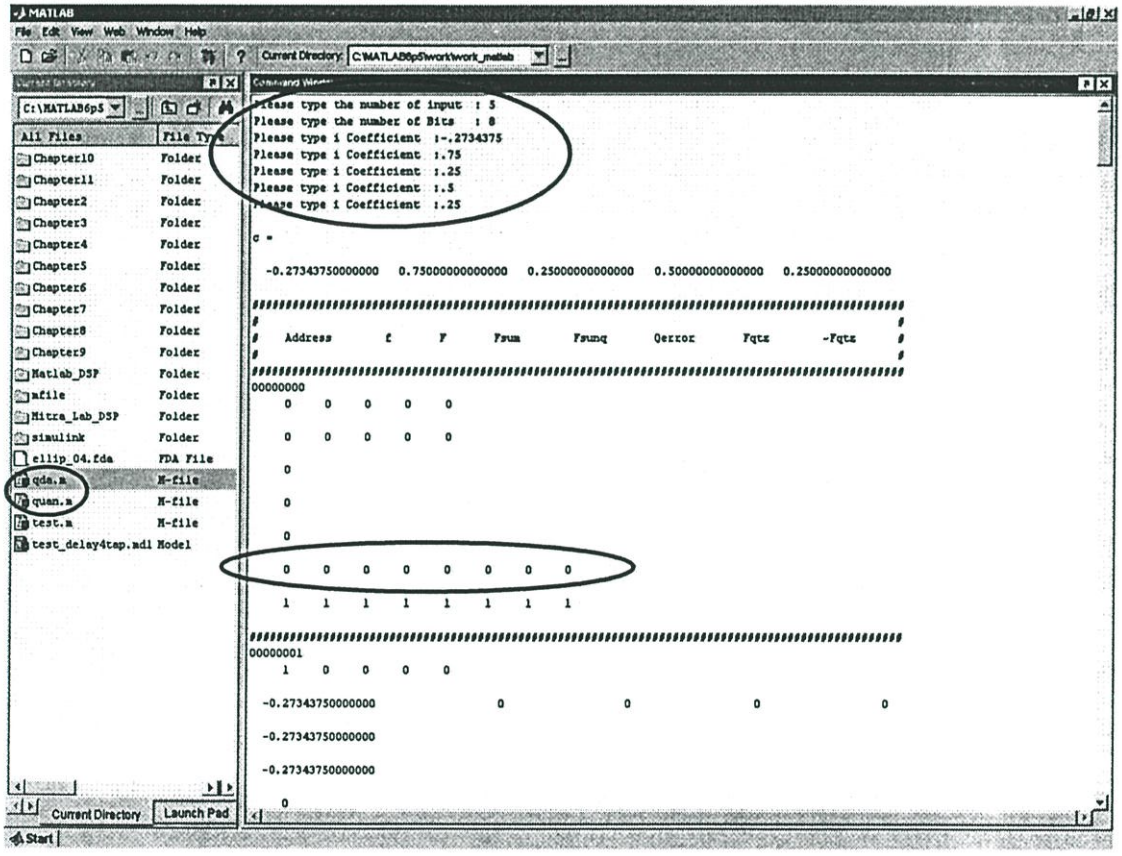
ค่า F(.) ที่ขึ้นกับสัมประสิทธิ์ $\overline{a}_0, \overline{a}_1, \overline{a}_2, \overline{b}_1, \overline{b}_2$ สามารถแสดงได้ดังตารางที่ 6.2

ตารางที่ 6.2 การแทนค่าแอดเดรสที่ได้จากฟังก์ชัน F(.)

x(n)	x(n-1)	x(n-2)	y(n-1)	y(n-2)	F(.)
0	0	0	0	0	0
0	0	0	0	1	$-\overline{b}_2$
0	0	0	1	0	$-\overline{b}_1$
0	0	0	1	1	$-\overline{b}_1 - \overline{b}_2$
0	0	1	0	0	$\overline{a}_2$
		.			.
		.			.
		.			.
1	1	1	0	0	$\overline{a}_0 + \overline{a}_1 + \overline{a}_2$
1	1	1	0	1	$\overline{a}_0 + \overline{a}_1 + \overline{a}_2 - \overline{b}_2$
1	1	1	1	0	$\overline{a}_0 + \overline{a}_1 + \overline{a}_2 - \overline{b}_1$
1	1	1	1	1	$\overline{a}_0 + \overline{a}_1 + \overline{a}_2 - \overline{b}_1 - \overline{b}_2$

ดังนั้นจากรูปที่ 6.9 เราสามารถเขียนโปรแกรม m-file ทำการคำนวณบนโปรแกรม MATLAB ได้โดยซอสโค้ด (Source Codes) ได้อ้างอิงอยู่ในส่วนภาคผนวก ก. ชื่อไฟล์ quan.m, qda.m และไฟล์ f2com.mโดยไฟล์แรกจะเป็นฟังก์ชันสำหรับทำการควอนไตซ์ ไฟล์ถัดมาจะทำการสร้างตาราง LUT และทำการแปลงค่าเป็นฟังก์ชัน F(.) ของสัมประสิทธิ์ที่คำนวณมาในตอนแรก แนวทางพัฒนาสำหรับโปรแกรมเหล่านี้อาจสามารถสร้างให้ได้เป็นไฟล์ VHDL ได้เพื่อความสะดวกในขั้นตอนการสร้างด้วย FPGA

เมื่อทำการใช้โปรแกรมทั้งสองคำนวณค่าสัมประสิทธิ์สามารถแสดงได้ดังรูปที่ 6.10



รูปที่ 6.10 แสดงการคำนวณฟังก์ชัน F(.) และการแปลงเลข จากข้อมูลในตารางที่ 6.1 บนโปรแกรม MATLAB (Command Window)

<u>หมายเหตุ</u>

ลำดับการป้อนค่าสัมประสิทธิ์ให้ป้อน $-\overline{b}_2, -\overline{b}_1, \overline{a}_2, \overline{a}_1, \overline{a}_0$ ค่าที่คำนวณได้จากโปรแกรมดังกล่าวจะได้ผลลัพธ์ขนาด 8 บิตให้ทำการบวก "1" ที่บิตต่ำสุดก่อนเพื่อให้ได้ข้อมูลแบบส่วนเติมเต็มสองที่สมบูรณ์ แล้วจึงนำค่าเหล่านี้ โปรแกรมสู่ LUT แบบภาษา VHDL ต่อไป

6.2.4 สำหรับค่าสัมประสิทธิ์ควอนไตซ์ 8 บิต

เมื่อ $$H(z)=\frac{a_0+a_1z^{-1}+a_2z^{-2}}{1+b_1z^{-1}+b_2z^{-2}}$$

$a_0 = 0.2500000$ $\quad b_1 = -0.7500000$

$a_1 = 0.5000000$ $\quad b_2 = 0.2734375$

$a_2 = 0.2500000$

Scaling Factor n = 2

ตารางที่ 6.3 แสดงค่าสัมประสิทธิ์ควอนไตซ์ 8 บิตของวงจรกรองเชิงเลขที่ทำการแปลงเป็นเลขส่วนเติมเต็มสอง ด้วยโปรแกรม qda.m , quan.m และ f2com.m

	x(n)	x(n-1)	x(n-2)	y(n-1)	y(n-2)	F	F/n	Memory Map F	HEX
1.	0	0	0	0	0	0.0000000	0.0000000	0000 0000	00
2.	0	0	0	0	1	-0.2734375	-0.1367188	1110 1110	EE
3.	0	0	0	1	0	0.7500000	0.3750000	0011 0000	30
4.	0	0	0	1	1	0.4765625	0.2382813	0001 1111	1F
5.	0	0	1	0	0	0.2500000	0.1250000	0001 0000	10
6.	0	0	1	0	1	-0.0234375	-0.0117188	1111 1110	FE
7.	0	0	1	1	0	1.0000000	0.5000000	0100 0000	40
8.	0	0	1	1	1	0.7265625	0.3632813	0010 1111	2F
9.	0	1	0	0	0	0.5000000	0.2500000	0010 0000	20
10.	0	1	0	0	1	0.2265625	0.1132813	0000 1111	0F
11.	0	1	0	1	0	1.2500000	0.6250000	0101 0000	50
12.	0	1	0	1	1	0.9765625	0.4882813	0011 1111	3F
13.	0	1	1	0	0	0.7500000	0.3750000	0011 0000	30
14.	0	1	1	0	1	0.4765625	0.2382813	0001 1111	1F
15.	0	1	1	1	0	0.5000000	0.2500000	0010 0000	20
16.	0	1	1	1	1	1.2265625	0.6132813	0100 1111	4F
17.	1	0	0	0	0	0.2500000	0.1250000	0001 0000	10
18.	1	0	0	0	1	-0.0234375	-0.0117188	1111 1110	FE
19.	1	0	0	1	0	1.0000000	0.5000000	0100 0000	40
20.	1	0	0	1	1	0.7265625	0.3632813	0010 1111	2F
21.	1	0	1	0	0	0.5000000	0.2500000	0010 0000	20
22.	1	0	1	0	1	0.2265625	0.1132813	0000 1111	0F
23.	1	0	1	1	0	1.2500000	0.6250000	0101 0000	50
24.	1	0	1	1	1	0.9765625	0.4882813	0011 1111	3F
25.	1	1	0	0	0	0.7500000	0.3750000	0011 0000	30
26.	1	1	0	0	1	0.4765625	0.2382813	0001 1111	1F
27.	1	1	0	1	0	1.5000000	0.7500000	0110 0000	60
28.	1	1	0	1	1	1.2265625	0.6132813	0100 1111	4F
29.	1	1	1	0	0	1.0000000	0.5000000	0100 0000	40
30.	1	1	1	0	1	0.7265625	0.3632813	0010 1111	2F
31.	1	1	1	1	0	1.7500000	0.8750000	0111 0000	70
32.	1	1	1	1	1	1.4765625	0.7382813	0101 1110	5E

6.2.5 สำหรับค่าสัมประสิทธิ์ตัดส่วน 8 บิต (Truncate)

เมื่อ $$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}}$$

$a_0 = 0.2500000$ $b_1 = -0.7478027$

$a_1 = 0.5000000$ $b_2 = 0.2722167$

$a_2 = 0.2500000$

Scaling Factor n = 2

ตารางที่ 6.4 แสดงค่าสัมประสิทธิ์ตัดส่วน 8 บิตของวงจรกรองเชิงเลขที่ทำการแปลงเป็นเลขส่วนเติมเต็มสอง ด้วยโปรแกรม qda.m , quan.m และ f2com.m

	x(n)	x(n-1)	x(n-2)	y(n-1)	y(n-2)	F	F/n	Memory Map F	HEX
1.	0	0	0	0	0	0.0000000	0.0000000	0000 0000	00
2.	0	0	0	0	1	-0.2722167	-0.1361084	1110 1111	EF*
3.	0	0	0	1	0	0.7478027	0.3739014	0011 0000	30
4.	0	0	0	1	1	0.4755860	0.2377930	0001 1110	1E*
5.	0	0	1	0	0	0.2500000	0.1250000	0001 0000	10
6.	0	0	1	0	1	-0.0222167	-0.0111084	1111 1111	FF*
7.	0	0	1	1	0	0.9978027	0.4989014	0100 0000	40
8.	0	0	1	1	1	0.7255860	0.3627930	0010 1110	2E*
9.	0	1	0	0	0	0.5000000	0.2500000	0010 0000	20
10.	0	1	0	0	1	0.2277833	0.1138917	0000 1111	0F
11.	0	1	0	1	0	1.2478027	0.6239014	0101 0000	50
12.	0	1	0	1	1	0.9755860	0.4877930	0011 1110	3E*
13.	0	1	1	0	0	0.7500000	0.3750000	0011 0000	30
14.	0	1	1	0	1	0.4777833	0.2388917	0001 1111	1F
15.	0	1	1	1	0	1.4978027	0.7489014	0011 0000	30*
16.	0	1	1	1	1	1.2255860	0.6127930	0100 1110	4E*
17.	1	0	0	0	0	0.2500000	0.1250000	0001 0000	10
18.	1	0	0	0	1	-0.0222167	-0.0111084	1111 1111	FF
19.	1	0	0	1	0	0.9978027	0.4989014	0100 0000	40
20.	1	0	0	1	1	0.7255860	0.3627930	0010 1110	2E*
21.	1	0	1	0	0	0.5000000	0.2500000	0010 0000	20
22.	1	0	1	0	1	0.2277833	0.1138917	0000 1111	0F
23.	1	0	1	1	0	1.2478027	0.6239014	0101 0000	50
24.	1	0	1	1	1	0.9755860	0.4877930	0011 1110	3E*
25.	1	1	0	0	0	0.7500000	0.3750000	0011 0000	30
26.	1	1	0	0	1	0.4777833	0.2388917	0001 1111	1F
27.	1	1	0	1	0	1.4978027	0.7489014	0110 0000	60
28.	1	1	0	1	1	1.2255860	0.6127930	0100 1110	4E*
29.	1	1	1	0	0	1.0000000	0.5000000	0100 0000	40
30.	1	1	1	0	1	0.7277833	0.3638917	0010 1111	2F
31.	1	1	1	1	0	1.7478027	0.8739014	0111 0000	70
32.	1	1	1	1	1	1.4755860	0.7377930	0101 1110	5E

* ค่าF (.) ที่เปลี่ยนไปเมื่อเทียบกับตารางที่ 6.3

6.2.6 สำหรับค่าสัมประสิทธิ์ปัดเศษ 8 บิต (Rounding)

เมื่อ $$H(z)=\frac{a_0+a_1z^{-1}+a_2z^{-2}}{1+b_1z^{-1}+b_2z^{-2}}$$

$a_0 = 0.2500000$ $\quad b_1 = -0.7478027$

$a_1 = 0.5000000$ $\quad b_2 = \ 0.2722168$

$a_2 = 0.2500000$

Scaling Factor n = 2

ตารางที่ 6.5 แสดงค่าสัมประสิทธิ์ของวงจรกรองเชิงเลขที่ทำการแปลงเป็นเลขส่วนเติมเต็มสอง ด้วยโปรแกรม qda.m , quan.m และ f2com.m

	x(n)	x(n-1)	x(n-2)	y(n-1)	y(n-2)	F	F/n	Memory Map F	HEX
1.	0	0	0	0	0	0.0000000	0.0000000	0000 0000	00
2.	0	0	0	0	1	-0.2722168	-0.1361084	1110 1111	EF
3.	0	0	0	1	0	0.7478027	0.3739014	0011 0000	30
4.	0	0	0	1	1	0.4755859	0.2377930	0001 1110	1E
5.	0	0	1	0	0	0.2500000	0.1250000	0001 0000	10
6.	0	0	1	0	1	-0.0222168	-0.0111084	1111 1111	FF
7.	0	0	1	1	0	0.9978027	0.4989014	0100 0000	40
8.	0	0	1	1	1	0.7255859	0.3627930	0010 1110	2E
9.	0	1	0	0	0	0.5000000	0.2500000	0010 0000	20
10.	0	1	0	0	1	0.2277832	0.1138916	0000 1111	0F
11.	0	1	0	1	0	1.2478027	0.6239014	0101 0000	50
12.	0	1	0	1	1	0.9755859	0.4877930	0011 1110	3E
13.	0	1	1	0	0	0.7500000	0.3750000	0011 0000	30
14.	0	1	1	0	1	0.4777832	0.2388916	0001 1111	1F
15.	0	1	1	1	0	1.4978027	0.7489014	0011 0000	30
16.	0	1	1	1	1	1.2255859	0.6127930	0100 1110	4E
17.	1	0	0	0	0	0.2500000	0.1250000	0001 0000	10
18.	1	0	0	0	1	-0.2722168	-0.1361084	1110 1111	EF*
19.	1	0	0	1	0	0.9978027	0.4989014	0100 0000	40
20.	1	0	0	1	1	0.7255859	0.3627930	0010 1110	2E
21.	1	0	1	0	0	0.5000000	0.2500000	0010 0000	20
22.	1	0	1	0	1	0.2277832	0.1138916	0000 1111	0F
23.	1	0	1	1	0	1.2478027	0.6239014	0101 0000	50
24.	1	0	1	1	1	0.9755859	0.4877930	0011 1110	3E
25.	1	1	0	0	0	0.7500000	0.3750000	0011 0000	30
26.	1	1	0	0	1	0.4777832	0.2388916	0001 1111	1F
27.	1	1	0	1	0	1.4978027	0.7489014	0110 0000	60
28.	1	1	0	1	1	1.2255859	0.6127930	0100 1110	4E
29.	1	1	1	0	0	1.0000000	0.5000000	0100 0000	40
30.	1	1	1	0	1	0.7277832	0.3638916	0010 1111	2F
31.	1	1	1	1	0	1.7478027	0.8739014	0111 0000	70
32.	1	1	1	1	1	1.4755859	0.7377930	0101 1110	5E

* ค่าF (.) ที่เปลี่ยนไปเมื่อเทียบกับตารางที่ 6.4

จากตารางที่ 6.3, 6.4 และ 6.5 จะพบว่า ค่า Memory Map F ของทั้งสามตารางให้ค่าที่แตกต่างกัน โดยค่าในตารางที่ 6.4 และ 6.5 จะให้ค่า Memory Map F ที่เท่ากันทุกค่ายกเว้นเฉพาะที่แอดเดรส "10001" ที่มีค่าต่างกัน แสดงให้เห็นว่า การปัดเศษ (Rounding) หรือการตัดส่วน (Truncate) แทบจะไม่มีผลต่อค่า Memory Map F ขนาด 8 บิต ของวงจรกรองเลย หรือมีผลน้อยมาก ซึ่งถ้า LUT เก็บค่าความยาวบิตที่สูงกว่านี้ จะให้ความแตกต่างของค่า F(.) ใน LUT ชัดเจนหลายค่า เพราะความแม่นยำจะสูงขึ้นตามจำนวนบิตข้อมูล สำหรับตารางที่ 6.3 เมื่อเทียบกับ ตารางที่ 6.4 และ 6.5 จะเห็นได้ชัดว่ามีค่า Memory Map F หลายค่าที่แตกต่างกัน ทำให้เราสามารถประมาณการคร่าวๆได้ว่าผลการตอบสนองความถี่ของ LUT แบบปัดเศษและตัดปลาย จะให้ค่าไกล้เคียงกันมาก แต่จะต่างจาก LUT แบบควอนไตซ์ 8 บิต ธรรมดา

6.3 การหาค่าฟังก์ชัน F(.) สำหรับ LUT แบบชดเชยความผิดพลาดด้วย ESS

เมื่อทำการคำนวณตาราง LUT สำหรับวงจรกรองเชิงเลขเรียบร้อยแล้ว หากนำค่าไปทำการสร้างวงจรกรองทันที จะทำให้ค่าการทำงานของวงจรกรองมีความคลาดเคลื่อน ดังนั้นเพื่อลดความผิดพลาดดังกล่าว เราสามารถทำการชดเชยความผิดพลาดโดยวิธีการ ESS ซึ่งได้กล่าวไว้แล้วในบทที่ 5 ทำการคำนวณค่า LUT ที่ทำการชดเชยความผิดพลาดได้ ซึ่งจากสมการที่ (5.36) คือ

$$\begin{aligned} F_{\varepsilon}(.) = {} & K_a\left\{(a_0-\overline{a}_0)X^j_{n-1}+(a_1-\overline{a}_1)X^j_{n-2}-(b_1-\overline{b}_1)Y^j_{n-2}\right\} \\ & +K_b\left\{(a_0-\overline{a}_0)X^j_{n-2}+(a_2-\overline{a}_2)Y^j_{n-2}\right\}+K_c(a_2-\overline{a}_2)Y^j_{n-1} \end{aligned}$$

ค่าถ่วงน้ำหนัก K_a, K_b และ K_c เป็นค่าเฉพาะตัวสำหรับวงจรกรองเชิงเลขใดๆ ซึ่งมีค่าเหมือนหรือไม่เหมือนกัน แต่โดยปกติจะให้ค่าแตกต่างกันเสมอ

เราจะทำการประมาณหาค่าถ่วงน้ำหนักสำหรับวงจรกรอง ที่จะชดเชยความผิดพลาดได้มากที่สุด ซึ่งใช้หลักการทางสถิติ คำนวณหาค่ากำลังสองความผิดพลาดเฉลี่ย (Mean Square Error: MSE) โดยนิยามของ MSE ได้ดังสมการที่ (6.4)

$$\mathrm{MSE}(t) = \frac{1}{n}\sum_{i=1}^{k} f_i(x_i-t)^2 = \sum_{i=1}^{k} p_i(x_i-t)^2 \qquad (6.4)$$

สามารถคำนวณ MSE ได้ดังตารางที่ 6.6 ซึ่งค่า MSE ที่ได้เราจะนำมาวิเคราะว่าค่า MSE ที่น้อยที่สุดจะเป็นค่าที่เหมาะสมในการนำมาใช้ชดเชยความผิดพลาด โดยกำหนดใช้ค่าถ่วงน้ำหนักของการประมาณค่า จากสมการที่ (5.36) มีค่าเป็นจำนวนเต็มระหว่าง -2 ถึง 2

ตารางที่ 6.6 แสดงการหาค่า MSE สำหรับการเลือกค่าถ่วงน้ำหนักที่เหมาะสม

K_a	K_b	K_c	T_1 = -0.0078725	T_2 = 0	T_3=0	MSE
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	0	-1	0	0	0	0
0	0	2	0	0	0	0
0	0	-2	0	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	0	0
0	1	-1	0	0	0	0
0	1	2	0	0	0	0
0	1	-2	0	0	0	0
0	-1	0	0	0	0	0
0	-1	1	0	0	0	0
0	-1	-1	0	0	0	0
0	-1	2	0	0	0	0
0	-1	-2	0	0	0	0
0	2	0	0	0	0	0
0	2	1	0	0	0	0
0	2	-1	0	0	0	0
0	2	2	0	0	0	0
0	2	-2	0	0	0	0
0	-2	0	0	0	0	0
0	-2	1	0	0	0	0
0	-2	-1	0	0	0	0
0	-2	2	0	0	0	0
0	-2	-2	0	0	0	0
1	0	0	6.19763E-05	0	0	2.06588E-05
1	0	1	6.19763E-05	0	0	2.06588E-05
1	0	-1	6.19763E-05	0	0	2.06588E-05
1	0	2	6.19763E-05	0	0	2.06588E-05
1	0	-2	6.19763E-05	0	0	2.06588E-05
1	1	0	6.19763E-05	0	0	2.06588E-05
1	1	1	6.19763E-05	0	0	2.06588E-05
1	1	-1	6.19763E-05	0	0	2.06588E-05
1	1	2	6.19763E-05	0	0	2.06588E-05
1	1	-2	6.19763E-05	0	0	2.06588E-05
1	-1	0	6.19763E-05	0	0	2.06588E-05
1	-1	1	6.19763E-05	0	0	2.06588E-05
1	-1	-1	6.19763E-05	0	0	2.06588E-05
1	-1	2	6.19763E-05	0	0	2.06588E-05
1	-1	-2	6.19763E-05	0	0	2.06588E-05

ตารางที่ 6.6 (ต่อ)

1	2	0	6.19763E-05	0	0	2.06588E-05
1	2	1	6.19763E-05	0	0	2.06588E-05
1	2	-1	6.19763E-05	0	0	2.06588E-05
1	2	2	6.19763E-05	0	0	2.06588E-05
1	2	-2	6.19763E-05	0	0	2.06588E-05
1	-2	0	6.19763E-05	0	0	2.06588E-05
1	-2	1	6.19763E-05	0	0	2.06588E-05
1	-2	-1	6.19763E-05	0	0	2.06588E-05
1	-2	2	6.19763E-05	0	0	2.06588E-05
1	-2	-2	6.19763E-05	0	0	2.06588E-05
-1	0	0	-6.19763E-05	0	0	-2.06588E-05
-1	0	1	-6.19763E-05	0	0	-2.06588E-05
-1	0	-1	-6.19763E-05	0	0	-2.06588E-05
-1	0	2	-6.19763E-05	0	0	-2.06588E-05
-1	0	-2	-6.19763E-05	0	0	-2.06588E-05
-1	1	0	-6.19763E-05	0	0	-2.06588E-05
-1	1	1	-6.19763E-05	0	0	-2.06588E-05
-1	1	-1	-6.19763E-05	0	0	-2.06588E-05
-1	1	2	-6.19763E-05	0	0	-2.06588E-05
-1	1	-2	-6.19763E-05	0	0	-2.06588E-05
-1	-1	0	-6.19763E-05	0	0	-2.06588E-05
-1	-1	1	-6.19763E-05	0	0	-2.06588E-05
-1	-1	-1	-6.19763E-05	0	0	-2.06588E-05
-1	-1	2	-6.19763E-05	0	0	-2.06588E-05
-1	-1	-2	-6.19763E-05	0	0	-2.06588E-05
-1	2	0	-6.19763E-05	0	0	-2.06588E-05
-1	2	1	-6.19763E-05	0	0	-2.06588E-05
-1	2	-1	-6.19763E-05	0	0	-2.06588E-05
-1	2	2	-6.19763E-05	0	0	-2.06588E-05
-1	2	-2	-6.19763E-05	0	0	-2.06588E-05
-1	-2	0	-6.19763E-05	0	0	-2.06588E-05
-1	-2	1	-6.19763E-05	0	0	-2.06588E-05
-1	-2	-1	-6.19763E-05	0	0	-2.06588E-05
-1	-2	2	-6.19763E-05	0	0	-2.06588E-05
-1	-2	-2	-6.19763E-05	0	0	-2.06588E-05
2	0	0	0.000123953	0	0	4.13175E-05
2	0	1	0.000123953	0	0	4.13175E-05
2	0	-1	0.000123953	0	0	4.13175E-05
2	0	2	0.000123953	0	0	4.13175E-05
2	0	-2	0.000123953	0	0	4.13175E-05
2	1	0	0.000123953	0	0	4.13175E-05
2	1	1	0.000123953	0	0	4.13175E-05

ตารางที่ 6.6 (ต่อ)

2	1	-1	0.000123953	0	0	4.13175E-05
2	1	2	0.000123953	0	0	4.13175E-05
2	1	-2	0.000123953	0	0	4.13175E-05
2	-1	0	0.000123953	0	0	4.13175E-05
2	-1	1	0.000123953	0	0	4.13175E-05
2	-1	-1	0.000123953	0	0	4.13175E-05
2	-1	2	0.000123953	0	0	4.13175E-05
2	-1	-2	0.000123953	0	0	4.13175E-05
2	2	0	0.000123953	0	0	4.13175E-05
2	2	1	0.000123953	0	0	4.13175E-05
2	2	-1	0.000123953	0	0	4.13175E-05
2	2	2	0.000123953	0	0	4.13175E-05
2	2	-2	0.000123953	0	0	4.13175E-05
2	-2	0	0.000123953	0	0	4.13175E-05
2	-2	1	0.000123953	0	0	4.13175E-05
2	-2	-1	0.000123953	0	0	4.13175E-05
2	-2	2	0.000123953	0	0	4.13175E-05
2	-2	-2	0.000123953	0	0	4.13175E-05
-2	0	0	-0.000123953	0	0	-4.13175E-05
-2	0	1	-0.000123953	0	0	-4.13175E-05
-2	0	-1	-0.000123953	0	0	-4.13175E-05
-2	0	2	-0.000123953	0	0	-4.13175E-05
-2	0	-2	-0.000123953	0	0	-4.13175E-05
-2	1	0	-0.000123953	0	0	-4.13175E-05
-2	1	1	-0.000123953	0	0	-4.13175E-05
-2	1	-1	-0.000123953	0	0	-4.13175E-05
-2	1	2	-0.000123953	0	0	-4.13175E-05
-2	1	-2	-0.000123953	0	0	-4.13175E-05
-2	-1	0	-0.000123953	0	0	-4.13175E-05
-2	-1	1	-0.000123953	0	0	-4.13175E-05
-2	-1	-1	-0.000123953	0	0	-4.13175E-05
-2	-1	2	-0.000123953	0	0	-4.13175E-05
-2	-1	-2	-0.000123953	0	0	-4.13175E-05
-2	2	0	-0.000123953	0	0	-4.13175E-05
-2	2	1	-0.000123953	0	0	-4.13175E-05
-2	2	-1	-0.000123953	0	0	-4.13175E-05
-2	2	2	-0.000123953	0	0	-4.13175E-05
-2	2	-2	-0.000123953	0	0	-4.13175E-05
-2	-2	0	-0.000123953	0	0	-4.13175E-05
-2	-2	1	-0.000123953	0	0	-4.13175E-05
-2	-2	-1	-0.000123953	0	0	-4.13175E-05
-2	-2	2	-0.000123953	0	0	-4.13175E-05
-2	-2	-2	-0.000123953	0	0	-4.13175E-05

จากตารางที่ 6.6 เราจะพบว่า

K_a = 0 จะทำให้ได้ค่า MSE = 0 ไม่สามารถเลือกค่า K_a = 0 เพราะไม่เป็นจริง

K_a = 1 หรือ -1 จะทำให้ได้ค่า MSE = 2.06588E-05 และ -2.06588E-05 ตามลำดับ

K_a = 2 หรือ -2 จะทำให้ได้ค่า MSE = 4.13175E-05 และ -4.13175E-05 ตามลำดับ

สำหรับกรณีวงจรกรองนี้ จะเห็นได้ว่าขึ้นอยู่กับเพียงค่า K_a เพียงค่าเดียว ส่วนค่า K_b และ K_c ที่เปลี่ยนแปลงไม่มีผลต่อค่าที่คำนวณได้ เพราะถูกคูณด้วยค่าศูนย์ตามสมการที่ (5.36)

เนื่องจากค่า $(a_0 - \overline{a}_0)$, $(a_1 - \overline{a}_1)$, $(a_2 - \overline{a}_2)$ ที่ได้จากโปรแกรม MATLAB มีค่าเป็นศูนย์ เพราะการควอนไตซ์, การปัดเศษ และการตัดปลาย ลงตัวเท่ากันพอดี จึงทำให้สมการที่ (5.36) ลดรูปเหลือดังแสดงในสมการที่ (6.5)

$$F_\varepsilon(.) = -(b_1 - \overline{b}_1) K_a Y_{n-2}^j \qquad (6.5)$$

แทนค่า $-(b_1 - \overline{b}_1)$ = -0.0078725

เนื่องจาก Y_{n-2}^j เป็นค่าบิตข้อมูลมีค่า เป็นได้เพียง “0” หรือ “1”

ดังนั้นแล้วจะพิจารณาได้เป็น 2 กรณีคือ

1) กรณีไม่ชดเชยความผิดพลาด เมื่อ Y_{n-2}^j = “0” (LUT ไม่เปลี่ยนแปลง)
2) กรณีชดเชยความผิดพลาด เมื่อ Y_{n-2}^j = “1”

ตารางที่ 6.7 แสดงการแก้ไขความผิดพลาดของ LUT ด้วยการถ่วงน้ำหนัก

K_a	บวกค่า F ในตารางที่ 6.3 ด้วย		MSE
	จำนวน	จำนวนส่วนเติมเต็มสอง	
0	0	0000 0000	0
1	-0.0078725	1111 1111	2.06588E-05
-1	0.0078725	0000 0001	-2.06588E-05
2	-0.0157450	1111 1110	4.13175E-05
-2	0.0157450	0000 0010	-4.13175E-05

เพื่อความสะดวกในการแก้ไข ถ้าพิจารณาเฉพาะ K_g มีค่าเป็น -1 และ -2 จะได้ตารางที่แก้ไขความผิดพลาดใหม่ด้วย วิธีการ ESS แสดงการเปรียบเทียบต่างๆ ดังตารางที่ 6.8

ตารางที่ 6.8 ตารางการเปรียบเทียบค่า LUT แก้ไขความผิดพลาดกับ LUT ที่คำนวณได้

F_q	F_q+F_e (K_g= -1)	F_q+F_e (K_g= -2)	F_{Trc}	F_{Rnd}
0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
1110 1110	1110 1111	1111 0000	1110 1111	1110 1111
0011 0000	0011 0000	0011 0000	0011 0000	0011 0000
0001 1111	0010 0000	0010 0001	0001 1110	0001 1110
0001 0000	0001 0000	0001 0000	0001 0000	0001 0000
1111 1110	1111 1111	0000 0000	1111 1111	1111 1111
0100 0000	0100 0000	0100 0000	0100 0000	0100 0000
0010 1111	0011 0000	0011 0001	0010 1110	0010 1110
0010 0000	0010 0000	0010 0000	0010 0000	0010 0000
0000 1111	0001 0000	0001 0001	0000 1111	0000 1111
0101 0000	0101 0000	0101 0000	0101 0000	0101 0000
0011 1111	0100 0000	0100 0001	0011 1110	0011 1110
0011 0000	0011 0000	0011 0000	0011 0000	0011 0000
0001 1111	0010 0000	0010 0001	0001 1111	0001 1111
0010 0000	0010 0000	0010 0000	0011 0000	0011 0000
0100 1111	0101 0000	0101 0001	0100 1110	0100 1110
0001 0000	0001 0000	0001 0000	0001 0000	0001 0000
1111 1110	1111 1111	0000 0000	1111 1111	1110 1111
0100 0000	0100 0000	0100 0000	0100 0000	0100 0000
0010 1111	0011 0000	0011 0001	0010 1110	0010 1110
0010 0000	0010 0000	0010 0000	0010 0000	0010 0000
0000 1111	0001 0000	0001 0001	0000 1111	0000 1111
0101 0000	0101 0000	0101 0000	0101 0000	0101 0000
0011 1111	0100 0000	0100 0001	0011 1110	0011 1110
0011 0000	0011 0000	0011 0000	0011 0000	0011 0000
0001 1111	0010 0000	0010 0001	0001 1111	0001 1111
0110 0000	0110 0000	0110 0000	0110 0000	0110 0000
0100 1111	0101 0000	0101 0001	0100 1110	0100 1110
0100 0000	0100 0000	0100 0000	0100 0000	0100 0000
0010 1111	0011 0000	0011 0001	0010 1111	0010 1111
0111 0000	0111 0000	0111 0000	0111 0000	0111 0000
0101 1110	0101 1111	0110 0000	0101 1110	0101 1110

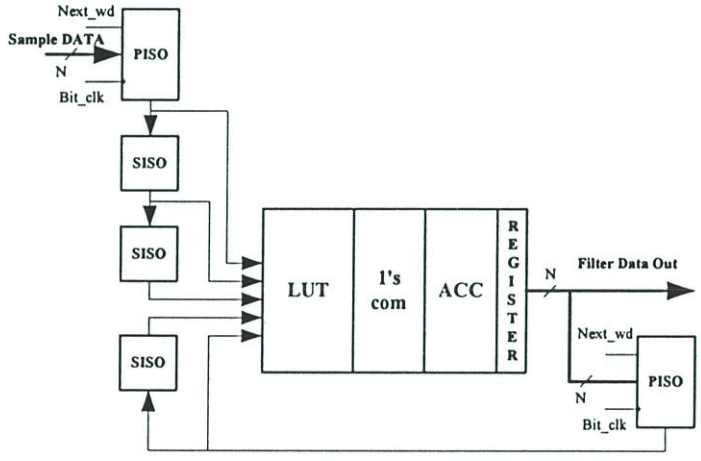
เมื่อทำการคำนวณค่าเปิดตารางแบบแก้ไขความผิดพลาดด้วยวิธี ESS เป็นที่เรียบร้อยแล้ว จากนี้ไปเราจะทำการออกแบบโครงสร้างของวงจรกรองในระดับฮาร์ดแวร์ โดยใช้ภาษา VHDL กำหนดพฤติกรรมการทำงานของแต่ละองค์ประกอบ

6.4 การสร้างวงจรกรองจากสถาปัตยกรรมคณิตศาสตร์แบบการกระจาย

หลังจากที่ค่าฟังก์ชันของสัมประสิทธิ์เฉพาะค่าใดๆ ได้ทำการคำนวณเป็นที่เรียบร้อยแล้ว ก็สามารถที่จะทำการออกแบบรายละเอียดของโครงสร้างสถาปัตยกรรมของวงจรกรองได้โดยใช้ภาษา VHDL ทำการกำหนดการทำงานของแต่ละส่วนประกอบของวงจรกรองเชิงเลขคณิตศาสตร์แบบกระจายได้ โดยจะทำการแยกออกแบบเป็น 3 ส่วนหลักๆ คือ

1) โครงสร้างตารางเปิดดูค่า (LUT)
2) โครงสร้างกระบวนการทางคณิตศาสตร์แบบการกระจาย
3) โครงสร้างกำเนิดสัญญาณควบคุมการทำงาน

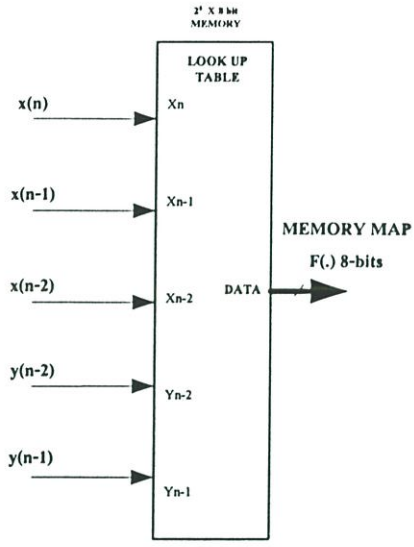
รูปที่ 6.11 แสดงบล็อกไดอะแกรมสถาปัตยกรรมของวงจรกรองเชิงเลขคณิตศาสตร์แบบการกระจายที่ออกแบบซึ่งจะประกอบไปด้วยส่วนทั้งสาม



รูปที่ 6.11 บล็อกไดอะแกรมของโครงสร้างวงจรกรองที่ออกแบบ

6.4.1 โครงสร้างตารางเปิดดูค่า (LUT)

จะทำหน้าที่ในการเก็บค่าคงที่สำคัญของวงจรกรองเชิงเลขป้อนกลับอันดับ 2 ที่คำนวณไว้ก่อนล่วงหน้า (Memory Map F) แล้วเก็บผลลัพธ์ไว้ในหน่วยความจำรอม (ROM) แต่เมื่อเราทำการสร้างบน FPGA แล้วจะทำให้วงจรกรองเชิงเลขที่ได้สามารถโปรแกรมการตอบสนองได้ ทำให้มีความยืดหยุ่นในการใช้งานมากขึ้น สะดวกต่อการเปลี่ยนแปลงชนิดของวงจรกรองได้ รูปที่ 6.12 แสดงบล็อกไดอะแกรมของอินพุทและเอาท์พุทของ LUT



รูปที่ 6.12 บล็อกไดอะแกรมของอินพุทและเอาท์พุทของ LUT

ข้อมูลจากตารางที่ 6.8 ที่เราได้ทำการคำนวณค่า LUT ทั้ง 5 แบบคือ แบบควอนไตซ์, แบบการตัดปลาย, แบบการปัดเศษ, แบบแก้ไขความผิดพลาด K_a = -1 และ แบบแก้ไขความผิดพลาด Ka = -2 จะถูกทำการเขียนค่าลงในรูปแบบโปรแกรมภาษา VHDL ที่อธิบายในลักษณะแบบหน่วยความจำถาวร แต่ให้ความหมายที่สามารถโปรแกรมค่าการทำงานของวงจรกรองได้ เนื่องจากเป็นการโปรแกรมการทำงานทุกครั้งลงสู่ FPGA โดยรูปแบบการแทนไฟล์โปรแกรมเป็นดังนี้

LUT แบบควอนไตซ์	แทนด้วยไฟล์ :	Rom_Fq.vhd
LUT แบบตัดคำ	แทนด้วยไฟล์ :	Rom_Trc.vhd
LUT แบบปัดเศษ	แทนด้วยไฟล์ :	Rom_Rnd.vhd
LUT แบบควอไตซ์แก้ไขความผิดพลาด K_a = -1	แทนด้วยไฟล์ :	Rom_FqKa1.vhd
LUT แบบควอไตซ์แก้ไขความผิดพลาด K_a = -2	แทนด้วยไฟล์ :	Rom_FqKa2.vhd

6.4.2 โครงสร้างกระบวนการทางคณิตศาสตร์

เป็นส่วนของฮาร์ดแวร์ที่ทำการจัดการข้อมูลดิจิตอลที่รับมาจาก ADC ทำการดำเนินการตามกระบวนการกระจายแบบคณิตศาสตร์ โดยทำการหน่วงสัญญาณขาเข้าในลักษณะบิตอนุกรมเวิร์ดขนาน (Bit-Serial Word-Parallel) และกระบวนการบวกสะสม ดังนั้นเราสามารถทำการออกแบบอุปกรณ์ (Component) ในโครงสร้างของส่วนนี้ได้โดยจะประกอบไปด้วย

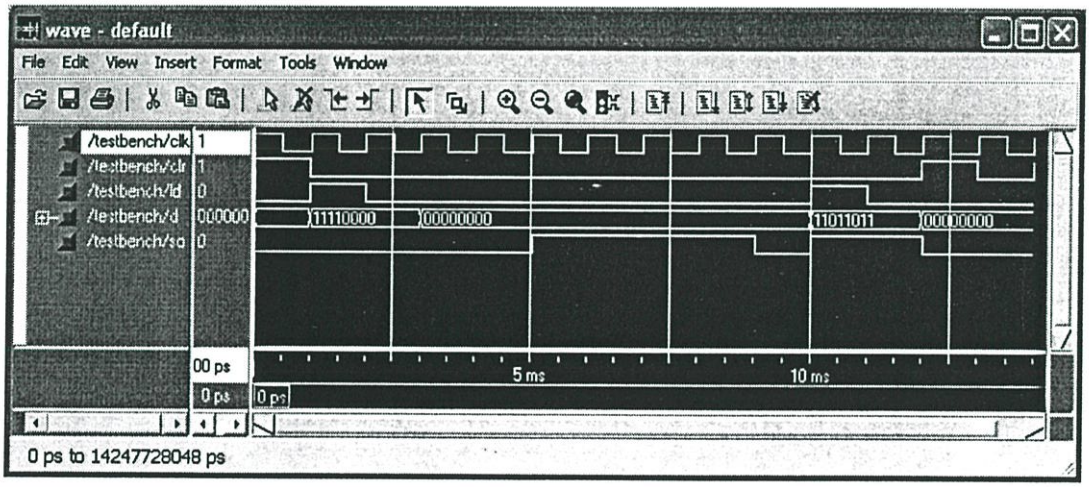
1. ชิพรีจีสเตอร์แบบขนานเข้าอนุกรมออกขนาด 8 บิต
2. ชิพรีจีสเตอร์แบบอนุกรมเข้าอนุกรมออกขนาด 8 บิต
3. แอคคิวมูเลเตอร์ เข้าขนาด 8 บิต
4. ตัวแปลงค่าคอมพลีเมนต์ขนาด 8 บิต

โปรแกรมภาษา VHDL ของอุปกรณ์ประกอบด้วยไฟล์ piso8.vhd, siso8.vhd, Acc9_sc_me.vhd และ one_com.vhd ตามลำดับ

ขั้นตอนในการออกแบบของแต่ละคอมโพเนนต์ จะต้องทราบถึงการทำงานของสัญญาณต่างๆ ที่เข้ามาแล้วทำการกำหนดฟังก์ชันการกระทำให้กับสัญญาณที่เข้ามา ว่าต้องการให้สัญญาณของเอาท์พุทคอมโพเนนต์นั้นๆ เป็นอย่างไร การทำงานและผลการจำลองการทำงานของแต่ละคอมโพเนนต์สามารถออกแบบและทำการจำลองการทำงานของสัญญาณอินพุทด้วยโปรแกรม Xilinx ISE และ ModelSim ได้ดังต่อไปนี้

6.4.2.1 ชิฟรีจีสเตอร์แบบขนานเข้าอนุกรมออกขนาด 8 บิต

รูปที่ 6.13 เป็นการทดสอบการทำงานเมื่อป้อนสัญญาณอินพุทตามที่กำหนดให้กับรีจีสเตอร์ piso8.vhd เพื่อดูความถูกต้องของเอาท์พุทที่ได้

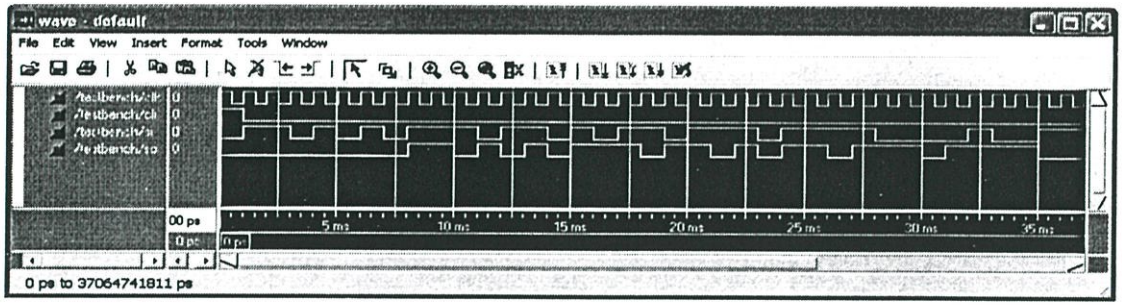


รูปที่ 6.13 การจำลองการทำงานของสัญญาณอินพุท piso8.vhd ด้วยโปรแกรม ModelSim

จากรูปที่ 6.13 จะสังเกตได้ว่าเมื่อเราทำการโหลดข้อมูล 8บิต ด้วยสัญญาณ "ld = 1" รีจีสเตอร์จะทำการเก็บค่าข้อมูล 8 บิตและจะทำการเลื่อนข้อมูล "so" ออกตามสัญญาณนาฬิกา 8 รอบสัญญาณนาฬิกา จึงจะเลื่อนข้อมูลออกทางเอาท์พุทของรีจีสเตอร์ครบทุกบิต

6.4.2.2 ชิพรีจีสเตอร์แบบอนุกรมเข้าอนุกรมออกขนาด 8 บิต

รูปที่ 6.14 เป็นการทดสอบการทำงานเมื่อป้อนสัญญาณอินพุทตามที่กำหนดให้กับรีจีสเตอร์ siso8.vhd เพื่อดูความถูกต้องของเอาท์พุทที่ได้

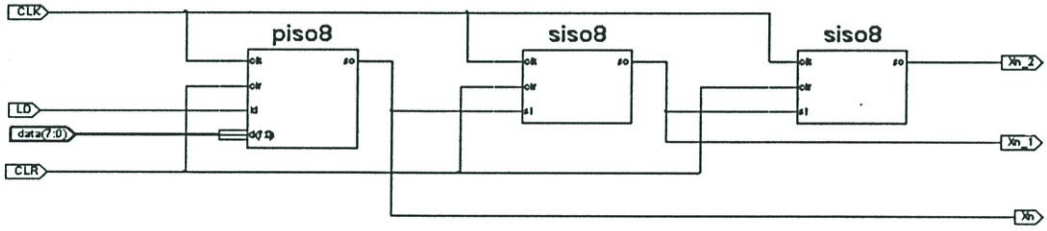


รูปที่ 6.14 การจำลองการทำงานของสัญญาณอินพุท siso8.vhd ด้วยโปรแกรม ModelSim

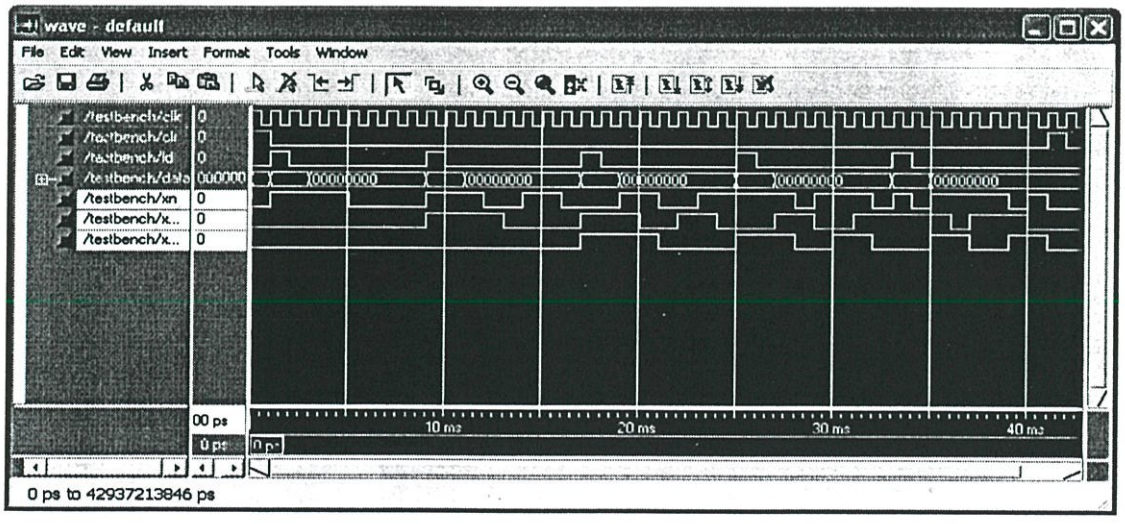
จากรูปที่ 6.14 เมื่อข้อมูล 1บิตอินพุทที่ต่อป้อนให้กับรีจีสเตอร์ถูกต่อจะทำการเก็บค่าที่บิต MSB ของรีจีสเตอร์ทันทีเมื่อสัญญาณนาฬิกาเป็น "1" จะทำการเลื่อนค่าที่เก็บไว้ไปทางขวาทั้ง 8 บิต เมื่อสัญญาณนาฬิกาถูกผ่านเข้ามาครบเพียง 7รอบสัญญาณนาฬิกาแรกก็จะทำการเลื่อนข้อมูลครบ 8 บิต สาเหตุที่ออกแบบในลักษณะนี้เพื่อต้องการให้ข้อมูลจากภาคแรกส่งผ่านต่อเนื่องตลอดทุก 8รอบสัญญาณนาฬิกา

6.4.2.3 ตัวหน่วงสัญญาณเข้าขนาด 8 บิต

รูปที่ 6.16 เป็นการทดสอบการทำงานของตัวหน่วงเวลา 2 ช่วง เพื่อดูความถูกต้องของเอาท์พุทที่ได้ โดยมีลักษณะการเชื่อมต่อดังรูปที่ 6.15



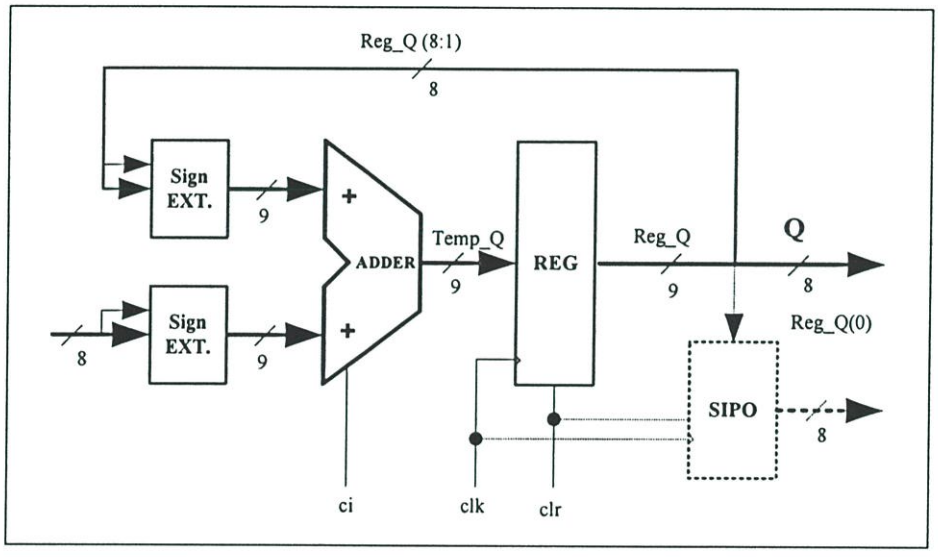
รูปที่ 6.15 การเชื่อมต่อของตัวหน่วงเวลาจากรีจีสเตอร์



รูปที่ 6.16 การจำลองการทำงานของสัญญาณอินพุทของตัวหน่วงเวลา

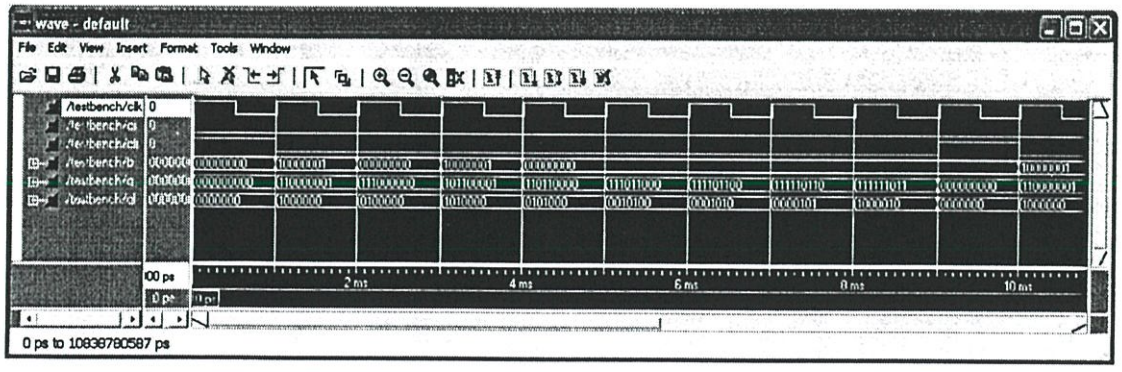
6.4.2.4 แอคคิวมูเลเตอร์ ขนาด 8 บิต

ส่วนประกอบภายในแอคคิวมูเลเตอร์ที่ทำการออกแบบแสดงดังรูปที่ 6.17 จะเห็นว่าประกอบไปด้วย การทำการขยายบิตเครื่องหมายหนึ่งบิตเพื่อป้องกันการเกิดโอเวอร์โฟล์ว แล้วจึงนำค่าอินพุทมาทำการบวกสะสม โดยค่าที่ทำการบวกสะสมแล้วจะถูกพักค่าเก็บไว้ที่รีจีสเตอร์ก่อนที่จะถูกส่งออกไปยังเอาท์พุท สำหรับส่วนของเส้นประที่แสดงในรูปนั้นเป็นส่วนที่อาจเพิ่มเข้ามาได้ ถ้าหากต้องการความแม่นยำของข้อมูลผลรวม ซึ่งจะเป็นการขยายบิตเอาท์พุทให้มีจำนวนมากขึ้นเป็น 16 บิตได้



รูปที่ 6.17 โครงสร้างของแอคคิวมูเลเตอร์ที่ออกแบบ

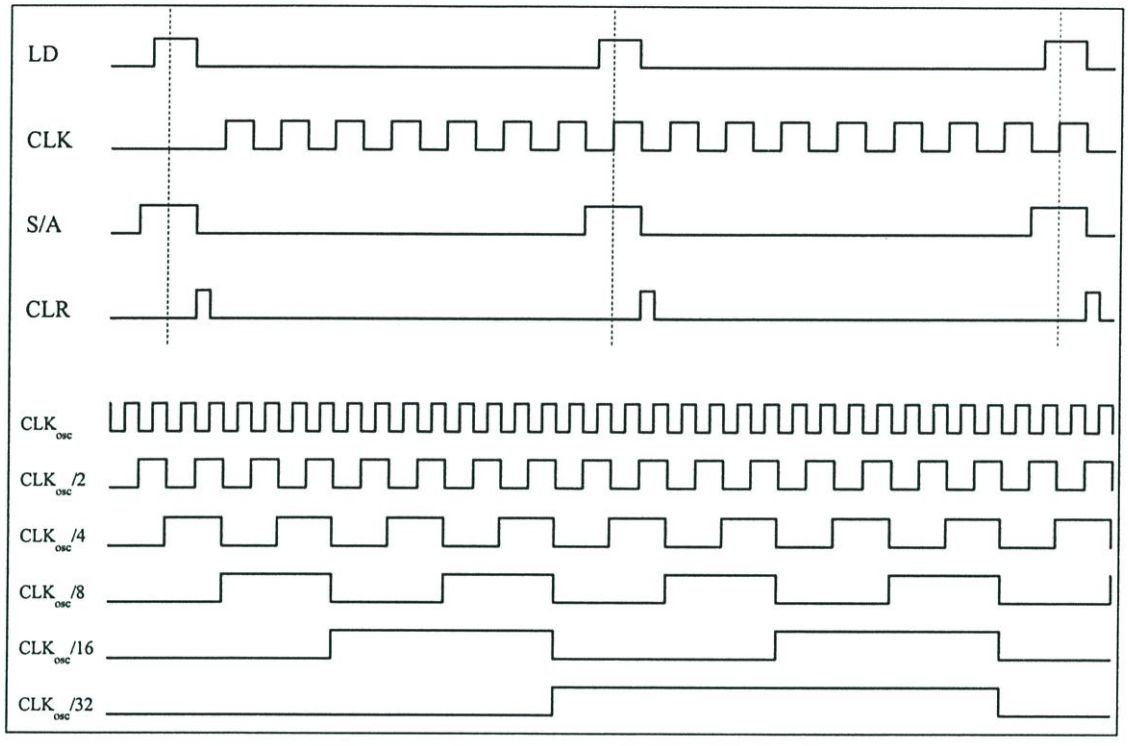
การทำงานของการบวกสะสมค่าในแอคคิวมูเลเตอร์ สามารถทดสอบการทำงานได้ดังรูปที่ 6.18 จะเห็นได้ว่าข้อมูลจะถูกทำการบวกสะสมทุกๆช่วงขอบขาขึ้นของสัญญาณนาฬิกา และเมื่อต้องการเคลียค่าเอาท์พุทสามารถทำได้โดยการให้ขาสัญญาณ clr เป็น “1”



รูปที่ 6.18 การจำลองการทำงานของสัญญาณอินพุทแอคคิวมูเลเตอร์ที่ทำการสะสมค่าเอาท์พุท

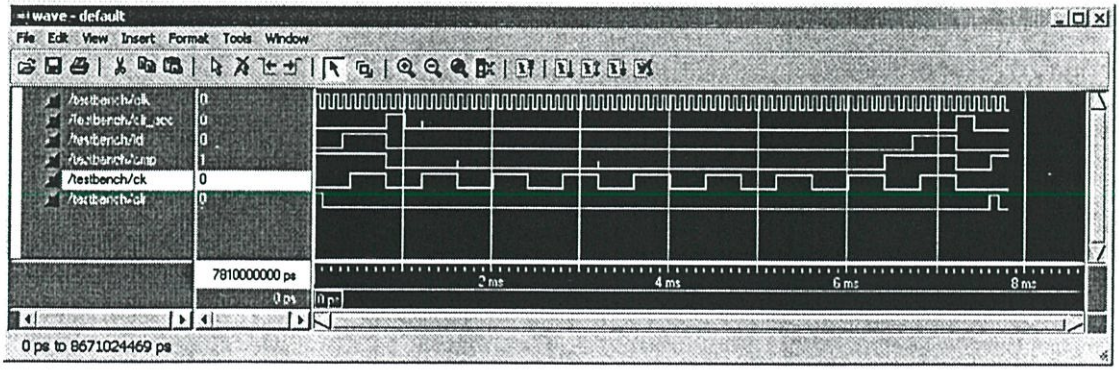
6.4.3 โครงสร้างกำเนิดสัญญาณควบคุมการทำงาน

เป็นวงจรส่วนที่สำคัญที่สุดของวงจรกรอง จะทำการกำหนดลำดับการทำงานของส่วนต่างๆ ให้สอดคล้องกันโดยจะใช้สัญญาณนาฬิกาอ้างอิง ทำการสร้างเป็นสัญญาณควบคุมอื่นๆ ซึ่งสามารถสรุปเป็นแผนภาพทางเวลาดังรูปที่ 6.19



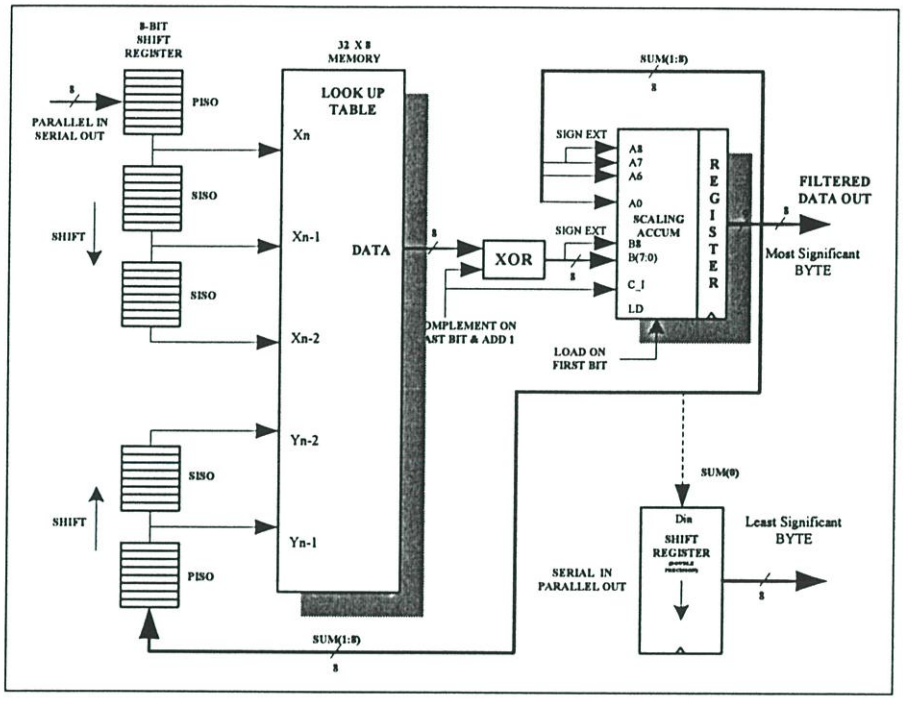
รูปที่ 6.19 แผนภาพเวลาสำหรับควบคุมการทำงานของวงจรกรองเชิงเลข

เมื่อทำการทำลองการทำงานโปรแกรม control.vhd ที่บรรยายพฤติกรรมของส่วนควบคุมผ่านโปรแกรม ModelSim แล้วจะได้การสร้างสัญญาณดังรูปที่ 20

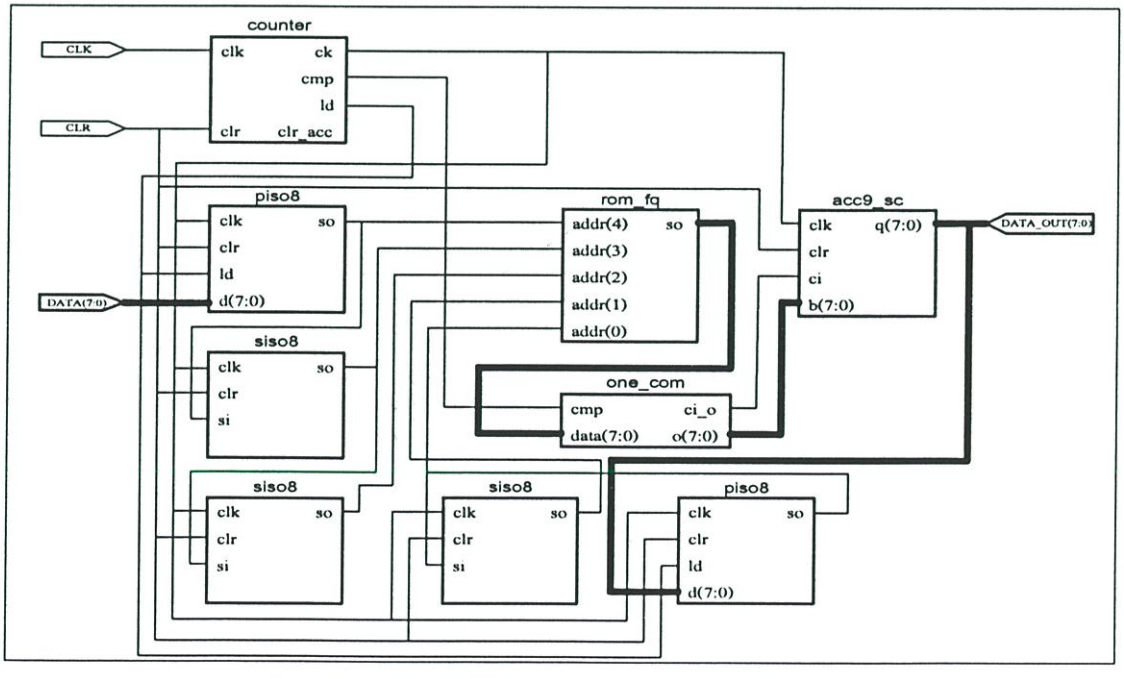


รูปที่ 6.20 การจำลองการทำงานของสัญญาณควบคุมการทำงานของวงจรกรองเชิงเลข

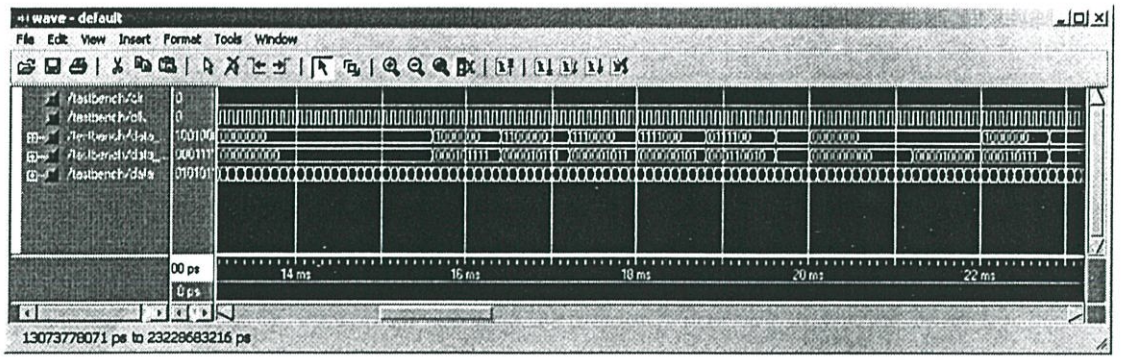
ดังนั้นแล้วจากโครงสร้างสถาปัตยกรรมคณิตศาสตร์แบบการกระจายขององค์ประกอบต่างๆ ดังรูปที่ 6.21 ที่ได้ทำการออกแบบไว้ในรูปแบบภาษา VHDL จะทำการเชื่อมต่อสัญญาณภายในด้วยการออกแบบในลักษณะ Schematic Design แสดงดังรูปที่ 6.22



รูปที่ 6.21 โครงสร้างของวงจรกรองเชิงเลขคณิตศาสตร์แบบการกระจายที่ทำการออกแบบ



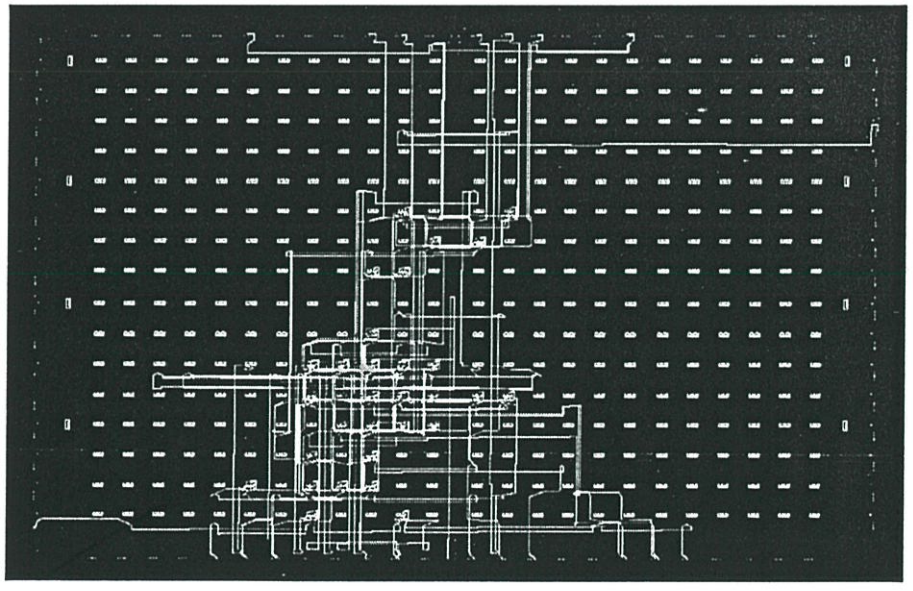
รูปที่ 6.22 การเชื่อมต่อสัญญาณภายในแต่ละองค์ประกอบของวงจรกรองเชิงเลข



รูปที่ 6.23 สัญญาณการจำลองการทำงานของวงจรกรองเชิงเลขที่ได้จากการออกแบบ

6.5 การสังเคราะห์วงจรกรอง

หลังจากที่ได้ทำการออกแบบโครงสร้างในแต่ละอุปกรณ์เป็นที่เรียบร้อย และทำการจำลองการทำงานได้ผลถูกต้อง ขั้นตอนสุดท้ายของการสร้างก็คือการสังเคราะห์วงจรที่ได้ออกแบบ เพื่อทำการโปรแกรมลงสู่ FPGA โดยผลการสังเคราะจะได้ลักษณะการจัดวางใน FPGA แสดงดังรูปที่ 6.24 ซึ่งทำการสร้างโดยใช้ FPGA หมายเลข xc2s50-5tq144 ในการโปรแกรมวงจรกรองทำงาน



รูปที่ 6.24 แผนผังการวางอุปกรณ์ต่างๆในโครงสร้างของ FPGA เบอร์ xc2s50-5tq144

จากโปรแกรม Xilinx ISE ที่ทำการสังเคราะห์วงจรกรองจะรายงานผลการสังเคราะห์ รวมทั้งขีดความสามารถต่างๆที่ วงจรทำงานได้ รวมถึงปริมาณทรัพยากรณ์ภายใน FPGA ที่ใช้ไปสามารถแสดงได้ดังนี้

ทำการสังเคราะห์โครงสร้างของวงจรกรองได้ผลการรายงานจากโปรแกรม XilinxISE ดังรูปที่ 6.25

```
HDL Synthesis Report

Macro Statistics
# ROMs                              : 1
  32x8-bit ROM                      : 1
# Registers                         : 34
  1-bit register                    : 31
  8-bit register                    : 2
  9-bit register                    : 1
# Counters                          : 1
  6-bit up counter                  : 1
# Multiplexers                      : 2
  2-to-1 multiplexer                : 2
# Adders/Subtractors                : 1
  9-bit adder carry in              : 1
# Xors                              : 8
  1-bit xor2                        : 8
```

(a) รายการอุปกรณ์ที่ทำการสังเคราะห์

```
Device utilization summary:
---------------------------

Selected Device : 2s50tq144-5

 Number of Slices:                      38  out of    768     4%
 Number of Slice Flip Flops:            63  out of   1536     4%
 Number of 4 input LUTs:                44  out of   1536     2%
 Number of bonded IOBs:                 25  out of     96    26%
 Number of GCLKs:                        1  out of      4    25%
```

(b) รายการปริมาณการใช้ทรัพยากรภายใน FPGA

```
Timing Summary:
---------------
Speed Grade: -5

   Minimum period: 6.982ns (Maximum Frequency: 143.225MHz)
   Minimum input arrival time before clock: 6.307ns
   Maximum output required time after clock: 8.449ns
   Maximum combinational path delay: No path found
```

(c) รายงานสรุปการทำงานเชิงเวลาและความถี่

```
Design Summary:
Number of errors:      0
Number of warnings:    0
Logic Utilization:
  Number of Slice Flip Flops:        62 out of  1,536    4%
  Number of 4 input LUTs:            39 out of  1,536    2%
Logic Distribution:
    Number of occupied Slices:                          40 out of    768    5%
    Number of Slices containing only related logic:     40 out of     40  100%
    Number of Slices containing unrelated logic:         0 out of     40    0%
       *See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs:           42 out of  1,536    2%
      Number used as logic:                      39
      Number used as a route-thru:                3
   Number of bonded IOBs:            25 out of     92   27%
      IOB Flip Flops:                             1
   Number of GCLKs:                   1 out of      4   25%
   Number of GCLKIOBs:                1 out of      4   25%

Total equivalent gate count for design:  825
Additional JTAG gate count for IOBs:  1,248
Peak Memory Usage:  48 MB
```

(d)รายการปริมาณการใช้ทรัพยากรภายใน FPGA จำแนกรายละเอียด

รูปที่ 6.25 รูปแสดงการรายงานที่ได้จากการสังเคราะห์วงจรกรองเชิงเลขด้วยโปรแกรม XilinxISE

6.6 บทสรุป

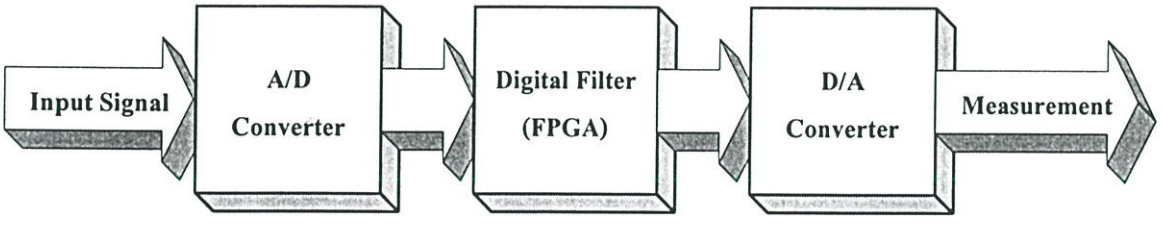
ในบทนี้ได้บรรยายถึงขั้นตอนสำหรับการออกแบบวงจรกรองเชิงเลขโดยใช้วิธีคณิตศาสตร์แบบการกระจายอย่างละเอียด โดยประยุกต์ใช้การจัดรูปสเปกตรัมความผิดพลาดเข้าร่วมในโครงสร้างเดียวกันของวงจรกรอง ทำให้วงจรกรองไม่จำเป็นต้องเปลี่ยนแปลงโครงสร้าง DA ไป ขณะที่จะให้ผลการทำงานที่สามารถแก้ไขความผิดพลาดที่เกิดขึ้นเนื่องจากรูปแบบการแทนข้อมูลที่มีความยาวคำจำกัด การวิเคราะห์การออกแบบเชิงคุณสมบัติของวงจรกรอง ไม่ว่าจะเป็นการตอบสนองทางความถี่และเฟส, ผลกระทบความผิดพลาดที่เกิดขึ้น และการคำนวณการแก้ไขความผิดพลาดด้วยโปรแกรม MATLAB, การสร้างตาราง LUT ซึ่งเป็นหัวใจสำคัญในการกำหนดลักษณะของวงจรกรองให้ได้คุณสมบัติตามต้องการ ต้องอาศัยหลักการและทฤษฎีต่างๆที่ได้กล่าวถึงในบทก่อนหน้านี้เป็นพื้นฐานสำคัญในการออกแบบและคำนวณค่า และขั้นตอนการสร้างฮาร์ดแวร์ แล้วจึงทำการสังเคราะห์วงจรที่แทนด้วยภาษาที่อธิบายการทำงานฮาร์ดแวร์ (VHDL) ของวงจร ซึ่งใช้โปรแกรม XilinxISE และ ModelSim ในการออกแบบและตรวจสอบการทำงานในเบื้องต้นก่อนทำการโปรแกรมวงจรลงสู่ FPGA ในบทถัดไปจะเป็นการทดสอบการทำงานของวงจรกรองที่ได้ทำการออกแบบโดยจะทำการตรวจสอบค่าสัญญาณต่างๆ และทำการเปรียบเทียบผลการตอบสนองทางความถี่ของค่า LUT ในแต่ละชนิดต่อไป

บทที่ 7

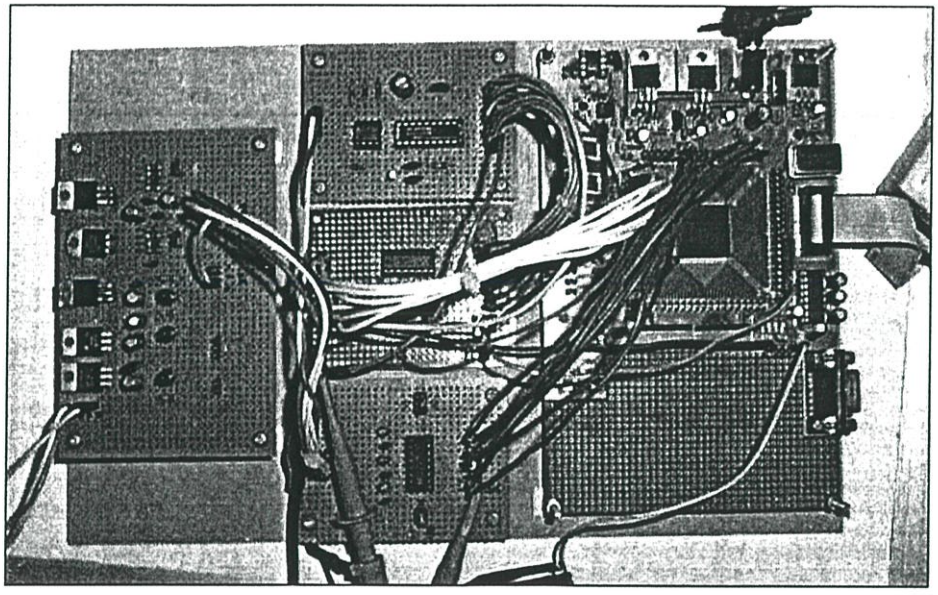
ผลการทดสอบวงจร

สำหรับบทนี้จะเป็นการแสดงผลการทดสอบการตอบสนองทางความถี่ของวงจรกรองเชิงเลขที่ได้ทำการออกแบบด้วยหลักการคณิตศาสตร์แบบการกระจาย และได้ทำการโปรแกรมลงสู่อุปกรณ์เอฟพีจีเอ โดยการทดลองในส่วนนี้จะทำการวัดผลเปรียบเทียบการทำงานของวงจรกรองในลักษณะแตกต่างกัน 5 แบบ คือ การสร้างวงจรกรองเชิงเลขบัทเทอร์เวิร์ธค่าสัมประสิทธิ์ควอนไตซ์ 8 บิต, ค่าสัมประสิทธิ์ตัดส่วน 8 บิต, ค่าสัมประสิทธิ์ปัดเศษ 8 บิต, ค่าสัมประสิทธิ์ควอนไตซ์แบบแก้ไขความผิดพลาด 8 บิต เมื่อ $K_a = -1$ และ -2 ตามลำดับ โดยจะใช้สัญญาณขาเข้ารูปคลื่นไซน์ที่มีขนาดแรงดันยอดถึงยอดขนาด 4-5 โวลต์ (V_{p-p}) ผ่านตัวแปลงสัญญาณอนาลอกเป็นดิจิตอลเข้าสู่วงจรกรองเชิงเลขที่จะทำการทดสอบ ส่วนสัญญาณดิจิตอลที่ออกจากวงจรกรองเชิงเลข จะนำมาผ่านตัวแปลงสัญญาณดิจิตอลเป็นอนาลอกเพื่อพิจารณาดูรูปคลื่นของสัญญาณขาออกที่ผ่านวงจรกรองเชิงเลขแล้ว ดังผังที่แสดงในรูปที่ 7.1 วงจรกรองเชิงเลขที่ทำการทดสอบจริงแสดงดังรูปที่ 7.2 โดยเลือกใช้ ไอซี (IC) ตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล และตัวแปลงสัญญาณดิจิตอลเป็นอนาลอก ขนาด 8 บิต ของบริษัท MAXIM และ HARRIS เบอร์ MAX165 และ CA3338 ตามลำดับ สำหรับวงจรทดสอบและคุณสมบัติของตัวแปลงสัญญาณดังกล่าวได้แสดงไว้ในภาคผนวก ข.

จากข้อจำกัดของอุปกรณ์ทดสอบจึงได้จัดให้วงจรกรองมีความถี่ของการสุ่ม 20 kHz การทดสอบกระทำโดยป้อนสัญญาณรูปคลื่นไซน์ที่ความถี่ต่างๆ ลงในวงจรกรองเชิงเลขที่ได้ออกแบบให้มีความถี่ตัดนอมัลไลซ์เป็น 0.3 หรือ 6 kHz โดยบันทึกขนาดของสัญญาณที่ผ่านวงจรกรองออกมาแล้วนำมาพล็อตกับความถี่ของสัญญาณ เป็นรูปกราฟแสดงผลตอบสนองทางความถี่ของ วงจรกรองเชิงเลขทั้ง 5 แบบที่กล่าวตอนต้นเปรียบเทียบกันแล้วทำการวิเคราะห์ผลการทำงาน



รูปที่ 7.1 ผังการทดสอบวงจรกรอง



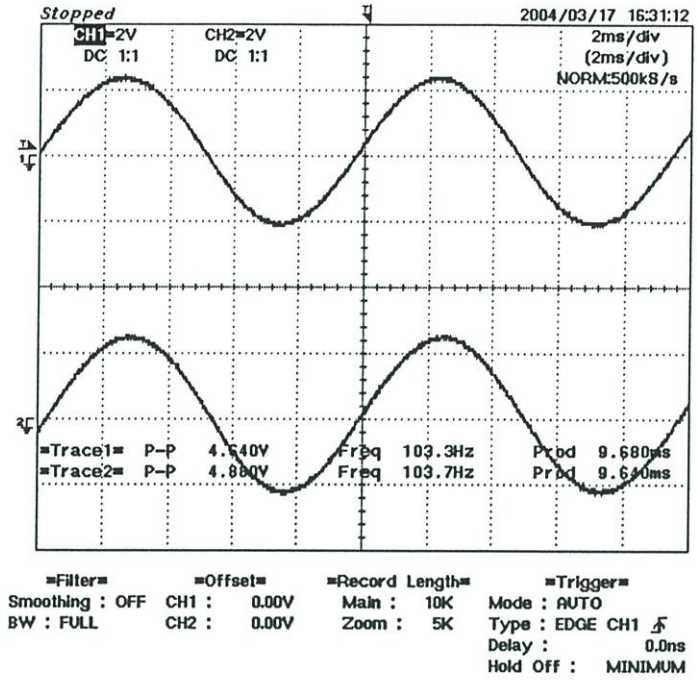
รูปที่ 7.2 แสดงการทดสอบวงจรกรองเชิงเลขด้วยการเชื่อมต่อกับวงจรแปลงสัญญาณ

7.1 ผลการทดสอบวงจรกรอง บัทเทอร์เวิร์ธ ที่มีความถี่ตัดเป็น 0.3 f_t= 6 kHz

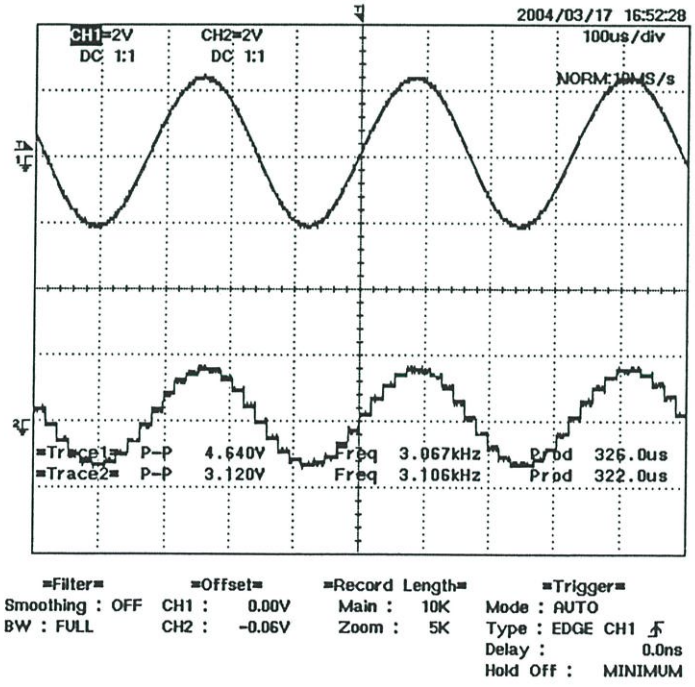
จากที่ได้กล่าวไปแล้วในตอนต้นสำหรับการทดสอบการทำงาน การตอบสนองความถี่ของวงจรกรองเชิงเลข 5 แบบ ดังนั้นในส่วนนี้จะได้แสดงผลการทดสอบผลตอบสนองทางความถี่ของวงจรกรองเชิงเลข โดยจะแบ่งเป็น 5 หัวข้อย่อยซึ่งผลการออกแบบวงจรกรองในระดับฟังก์ชันถ่ายโอนแต่ละชนิดที่ได้อธิบายไว้ตั้งแต่บทที่ 3 ถึง 5 จะทำการป้อนสัญญาณรูปไซน์ที่ความถี่ต่างๆกัน ที่ขนาดสัญญาณมีแรงดันยอดถึงยอดขนาด 4-5 โวลต์ (V_{p-p}) แล้ววัดรูปคลื่นสัญญาณที่ได้ออกมาจากวงจรกรองเชิงเลข

7.1.1 ผลการทดสอบวงจรกรอง บัทเทอร์เวิร์ธค่าสัมประสิทธิ์ควอนไตซ์ 8 บิต

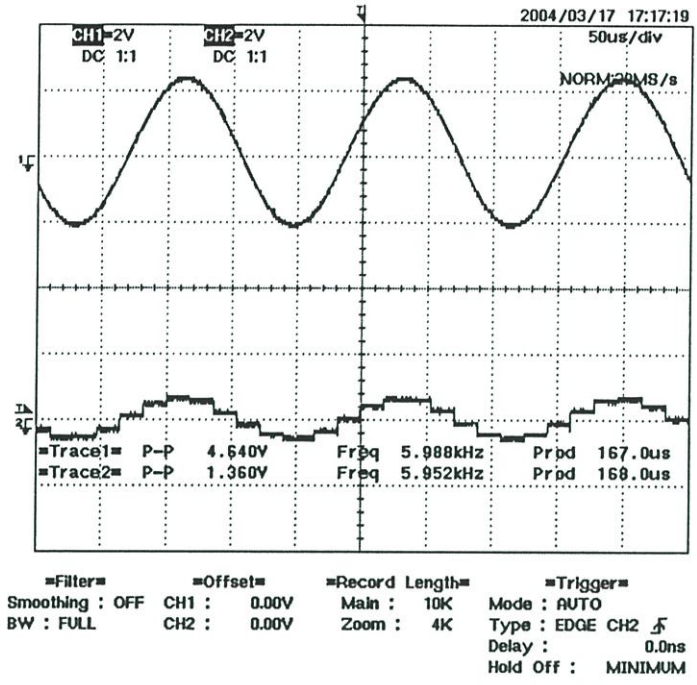
ผลการทดสอบผลตอบสนองทางความถี่ของวงจรกรองที่มีผลการออกแบบวงจรกรองในระดับฟังก์ชันถ่ายโอนที่แทนค่าฟังก์ชันใน LUT แบบควอนไตซ์ เมื่อใส่สัญญาณรูปไซน์ที่ความถี่ต่างๆ แล้ววัดรูปคลื่นที่ออกจากวงจรกรองได้ผลเป็น ดังรูปที่ 7.3


Stopped
2004/03/17 16:31:12
CH1=2V
DC 1:1
CH2=2V
DC 1:1
2ms/div
(2ms/div)
NORM:500kS/s
=Trace1= P-P 4.640V Freq 103.3Hz Prod 9.680ms
=Trace2= P-P 4.800V Freq 103.7Hz Prod 9.640ms
=Filter=
Smoothing : OFF
BW : FULL
=Offset=
CH1 : 0.00V
CH2 : 0.00V
=Record Length=
Main : 10K
Zoom : 5K
=Trigger=
Mode : AUTO
Type : EDGE CH1
Delay : 0.0ns
Hold Off : MINIMUM

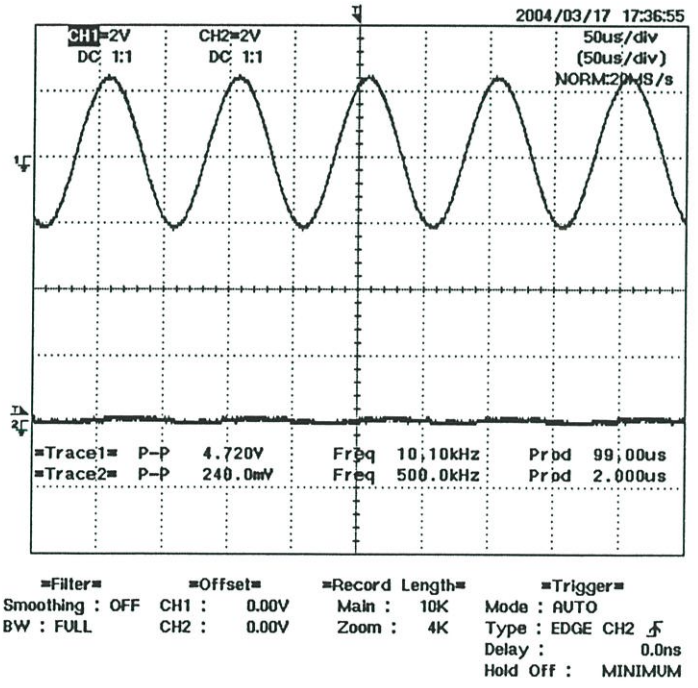
(a)


2004/03/17 16:52:28
CH1=2V
DC 1:1
CH2=2V
DC 1:1
100us/div
NORM:10MS/s
=Trace1= P-P 4.640V Freq 3.067kHz Prod 326.0us
=Trace2= P-P 3.120V Freq 3.106kHz Prod 322.0us
=Filter=
Smoothing : OFF
BW : FULL
=Offset=
CH1 : 0.00V
CH2 : -0.06V
=Record Length=
Main : 10K
Zoom : 5K
=Trigger=
Mode : AUTO
Type : EDGE CH1
Delay : 0.0ns
Hold Off : MINIMUM

(b)



(c)



(d)

รูปที่ 7.3 สัญญาณเข้าและออกที่ทำการวัดจากวงจรกรองเชิงเลขที่มีความถี่ตัด 6 kHz. (ควอนไตซ์ 8บิต)

(a) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 100 Hz

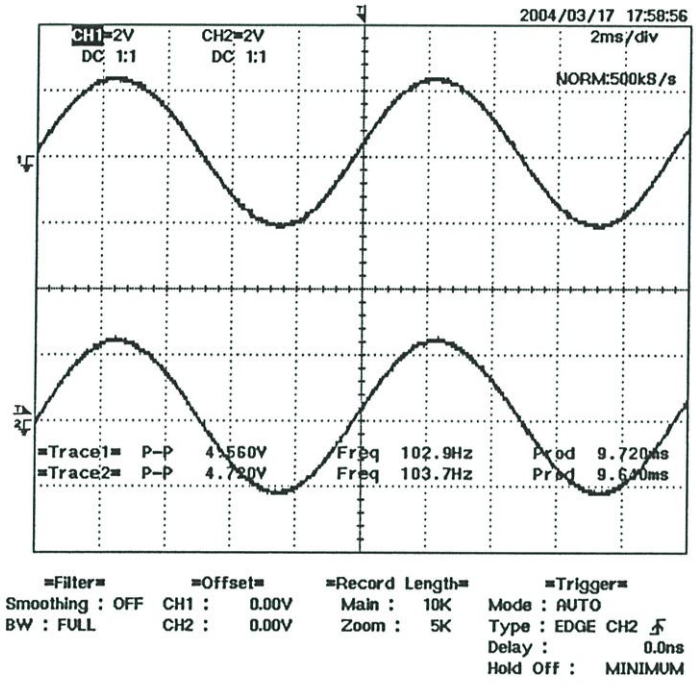
(b) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 3 kHz

(c) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 6 kHz

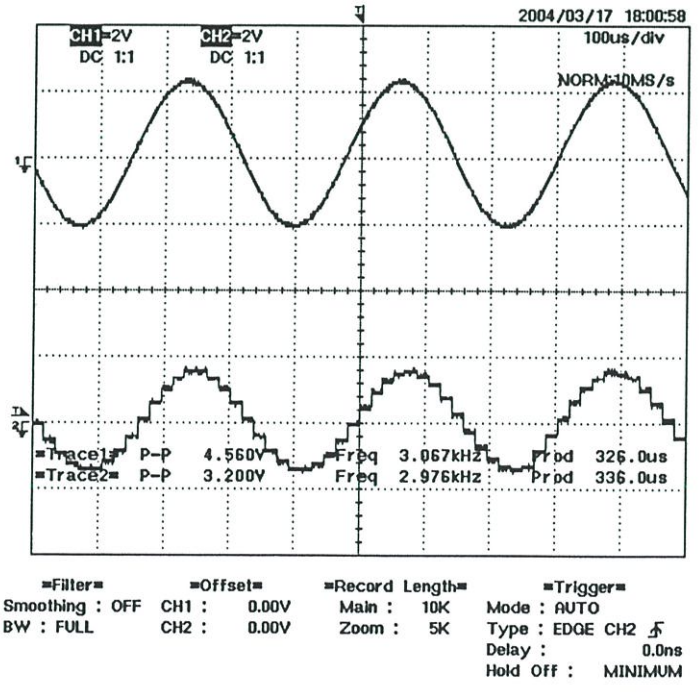
(d) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 10 kHz

7.1.2 ผลการทดสอบวงจรกรอง บัทเทอร์เวิร์ธค่าสัมประสิทธิ์ตัดส่วน 8 บิต

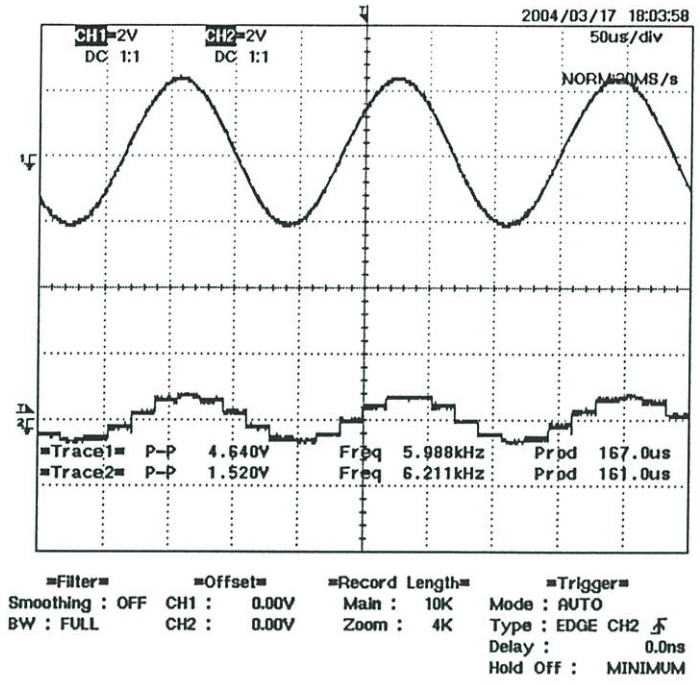
ผลการทดสอบผลตอบสนองทางความถี่ของวงจรกรองที่มีผลการออกแบบวงจรกรองในระดับฟังก์ชันถ่ายโอนที่แทนค่าฟังก์ชันใน LUT แบบตัดส่วน เมื่อใส่สัญญาณรูปไซน์ที่ความถี่ต่างๆ แล้ววัดรูปคลื่นที่ออกจากวงจรกรองได้ผลเป็น ดังรูปที่ 7.4



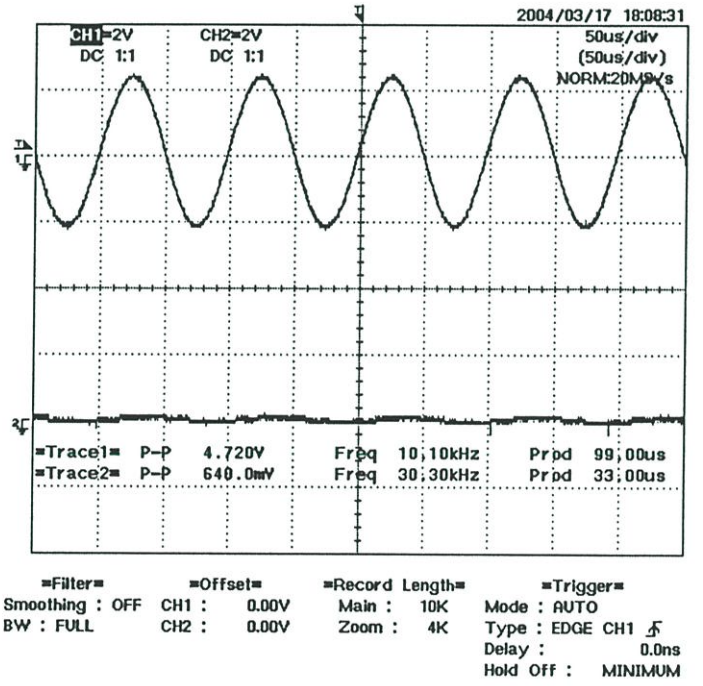
(a)



(b)



(c)



(d)

รูปที่ 7.4 สัญญาณเข้าและออกที่ทำการวัดจากวงจรกรองเชิงเลขที่มีความถี่ตัด 6 kHz. (ตัดส่วน 8 บิต)

(a) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 100 Hz

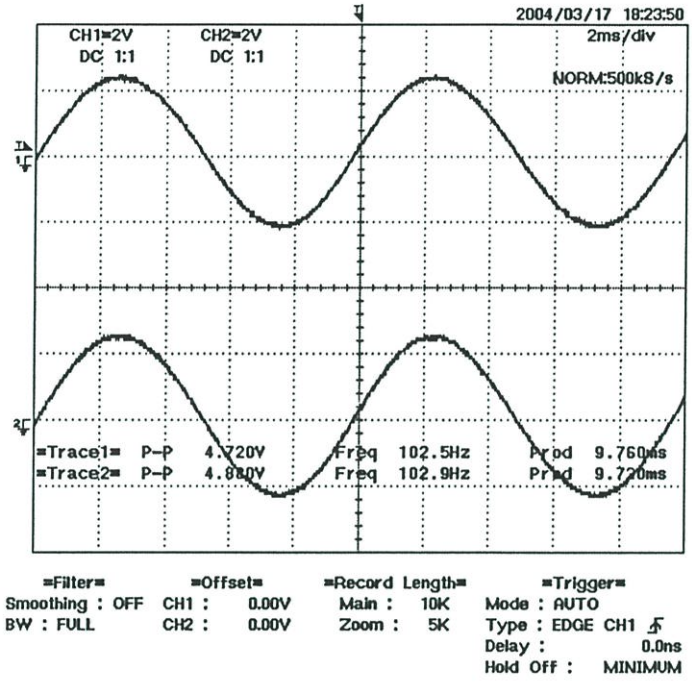
(b) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 3 kHz

(c) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 6 kHz

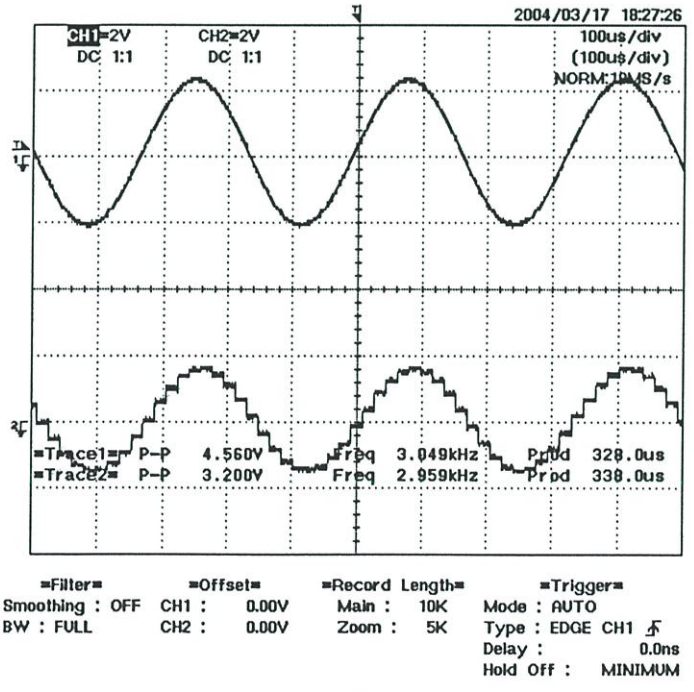
(d) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 10 kHz

7.1.3 ผลการทดสอบวงจรกรอง บัทเทอร์เวิร์ธค่าสัมประสิทธิ์ปัดเศษ 8 บิต

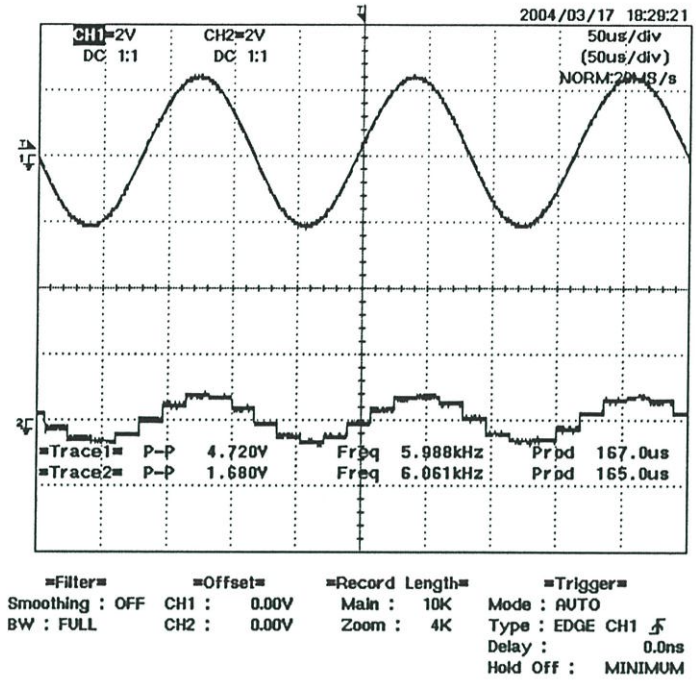
ผลการทดสอบผลตอบสนองทางความถี่ของวงจรกรองที่มีผลการออกแบบวงจรกรองในระดับฟังก์ชันถ่ายโอนที่แทนค่าฟังก์ชันใน LUT แบบปัดเศษ 8 บิต เมื่อใส่สัญญาณรูปไซน์ที่ความถี่ต่างๆ แล้ววัดรูปคลื่นที่ออกจากวงจรกรองได้ผลเป็น ดังรูปที่ 7.5



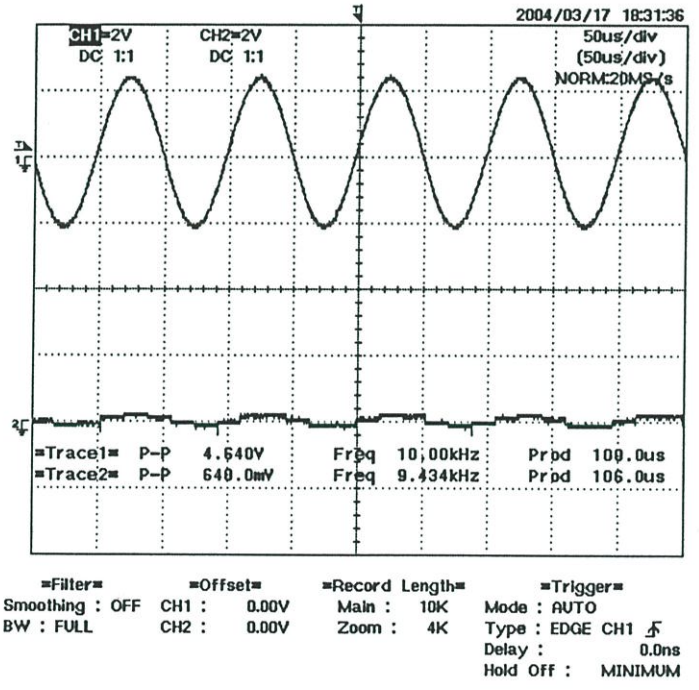
(a)



(b)



(c)



(d)

รูปที่ 7.5 สัญญาณเข้าและออกที่ทำการวัดจากวงจรกรองเชิงเลขที่มีความถี่ตัด 6 kHz. (ปัดเศษ 8 บิต)

(a) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 100 Hz

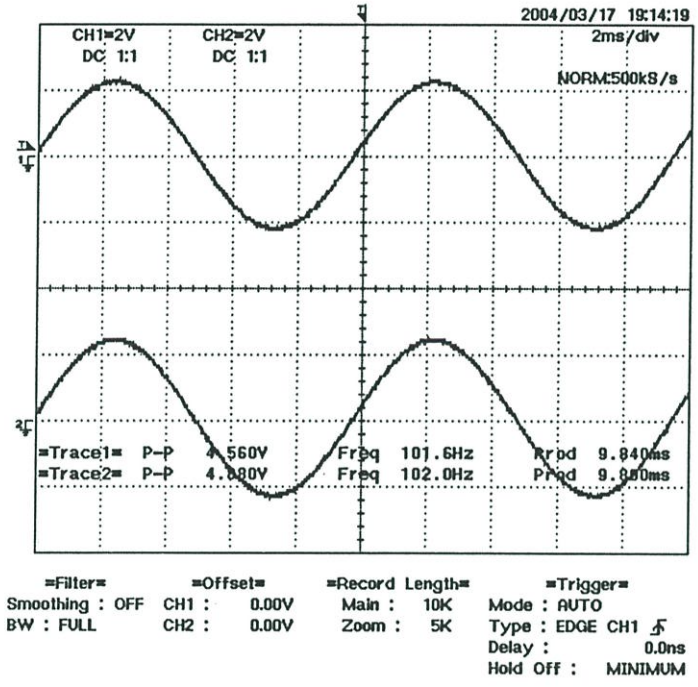
(b) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 3 kHz

(c) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 6 kHz

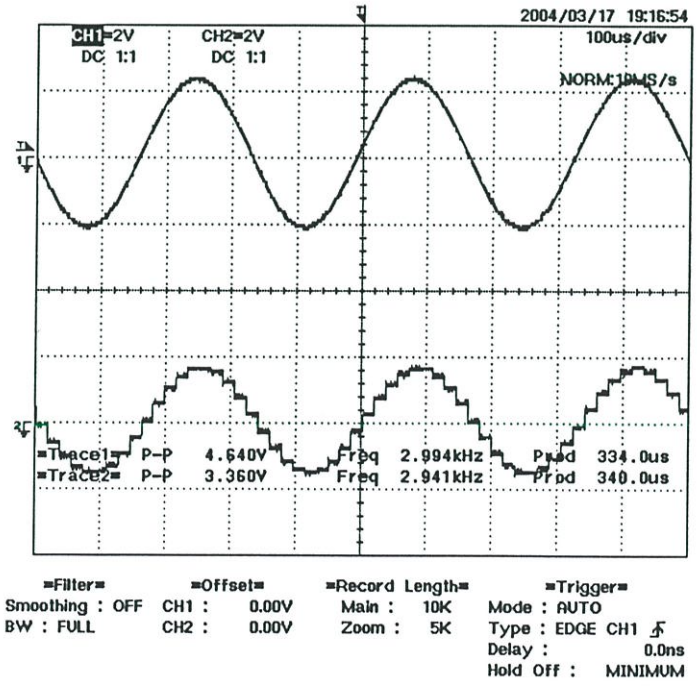
(d) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 10 kHz

7.1.4 ผลการทดสอบวงจรกรอง บัทเทอร์เวิร์ธค่าสัมประสิทธิ์ควอนไตซ์แบบแก้ไขความผิดพลาด 8 บิต เมื่อ $K_a = -1$

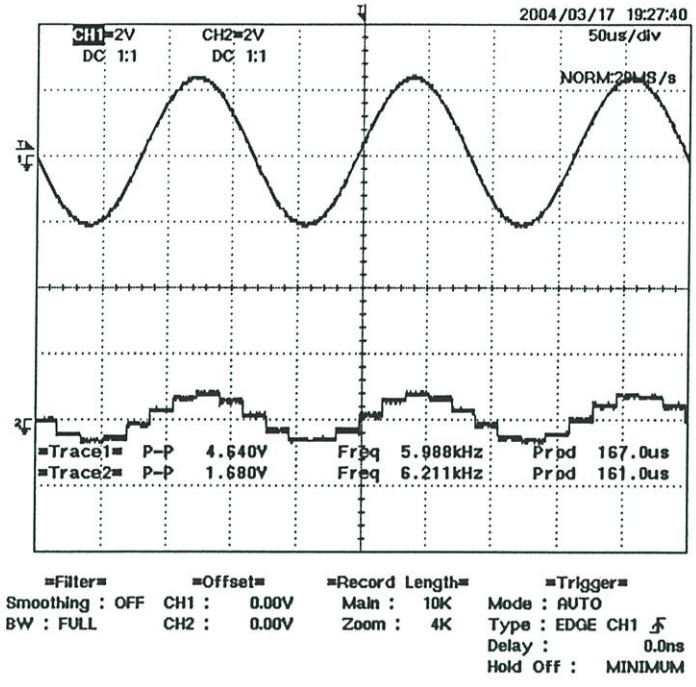
ผลการทดสอบผลตอบสนองทางความถี่ของวงจรกรองที่มีผลการออกแบบวงจรกรองในระดับฟังก์ชันถ่ายโอนที่แทนค่าฟังก์ชันใน LUT แบบแก้ไขความผิดพลาด เมื่อใส่สัญญาณรูปไซน์ที่ความถี่ต่างๆ แล้ววัดรูปคลื่นที่ออกจากวงจรกรองได้ผลเป็น ดังรูปที่ 7.6



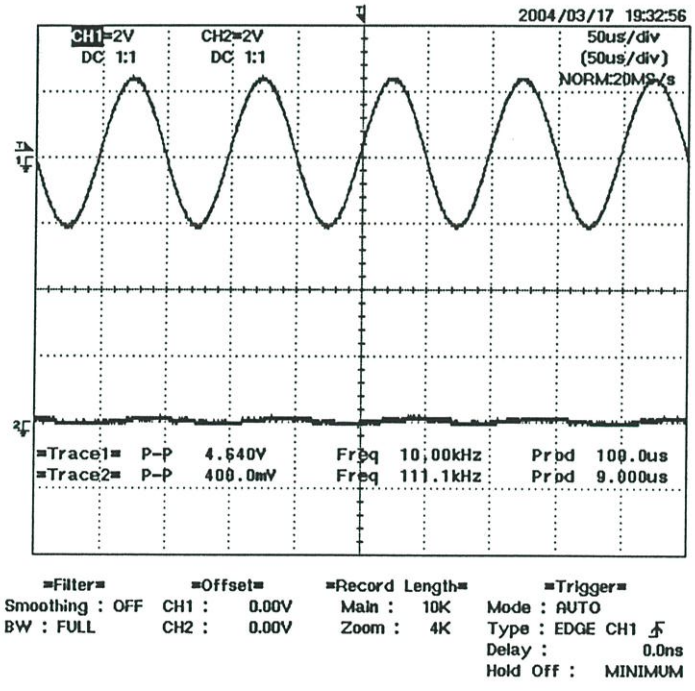
(a)



(b)



(c)



(d)

รูปที่ 7.6 สัญญาณเข้าและออกที่ทำการวัดจากวงจรกรองเชิงเลขที่มีความถี่ตัด 6 kHz. (ESS: $K_a = -1$)

(a) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 100 Hz

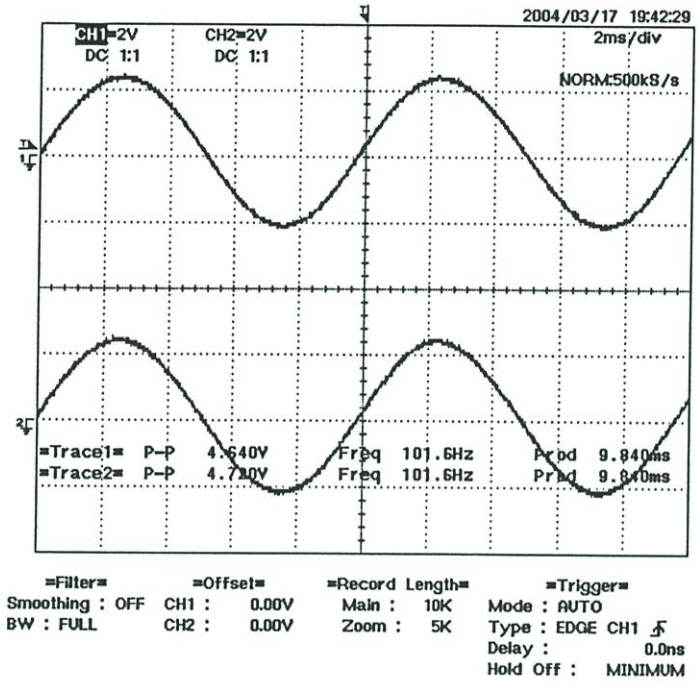
(b) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 3 kHz

(c) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 6 kHz

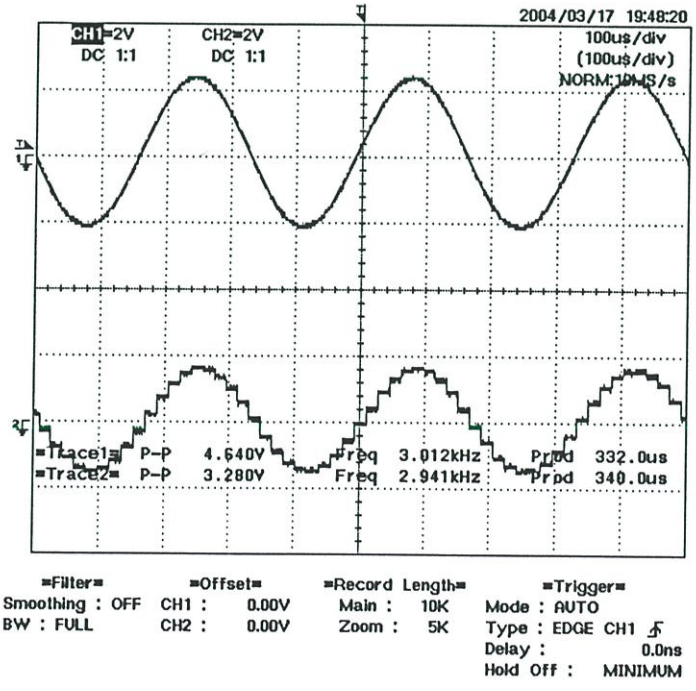
(d) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 10 kHz

7.1.5 ผลการทดสอบวงจรกรอง บัทเทอร์เวิร์ธค่าสัมประสิทธิ์ควอนไตซ์แบบแก้ไขความผิดพลาด 8 บิต เมื่อ $K_a = -2$

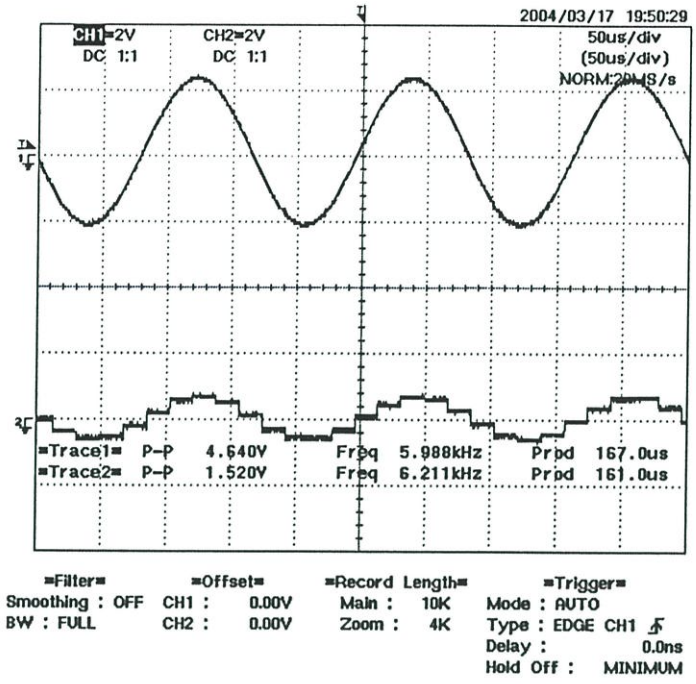
ผลการทดสอบผลตอบสนองทางความถี่ของวงจรกรองที่มีผลการออกแบบวงจรกรองในระดับฟังก์ชันถ่ายโอนที่แทนค่าฟังก์ชันใน LUT แบบแก้ไขความผิดพลาด เมื่อใส่สัญญาณรูปไซน์ที่ความถี่ต่างๆ แล้ววัดรูปคลื่นที่ออกจากวงจรกรองได้ผลเป็น ดังรูปที่ 7.7



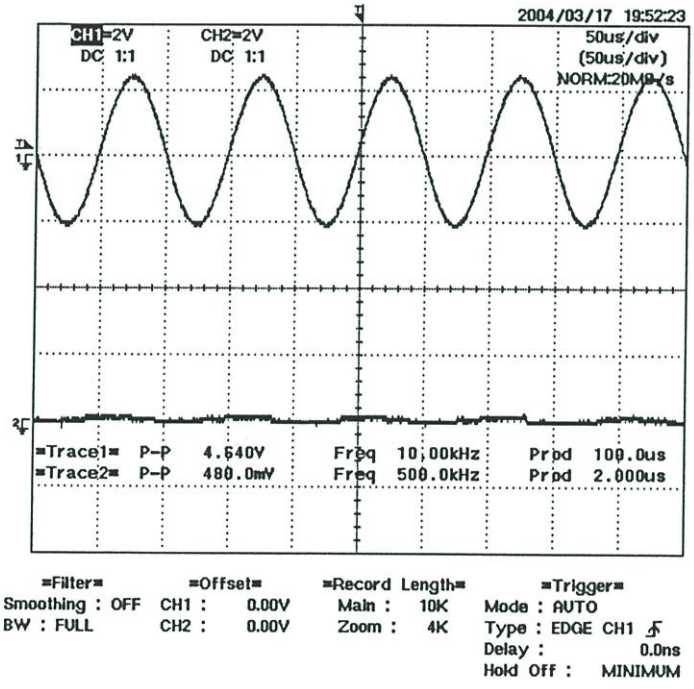
(a)



(b)



(c)



(d)

รูปที่ 7.7 สัญญาณเข้าและออกที่ทำการวัดจากวงจรกรองเชิงเลขที่มีความถี่ตัด 6 kHz. (ESS: $K_a = -2$)

(a) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 100 Hz

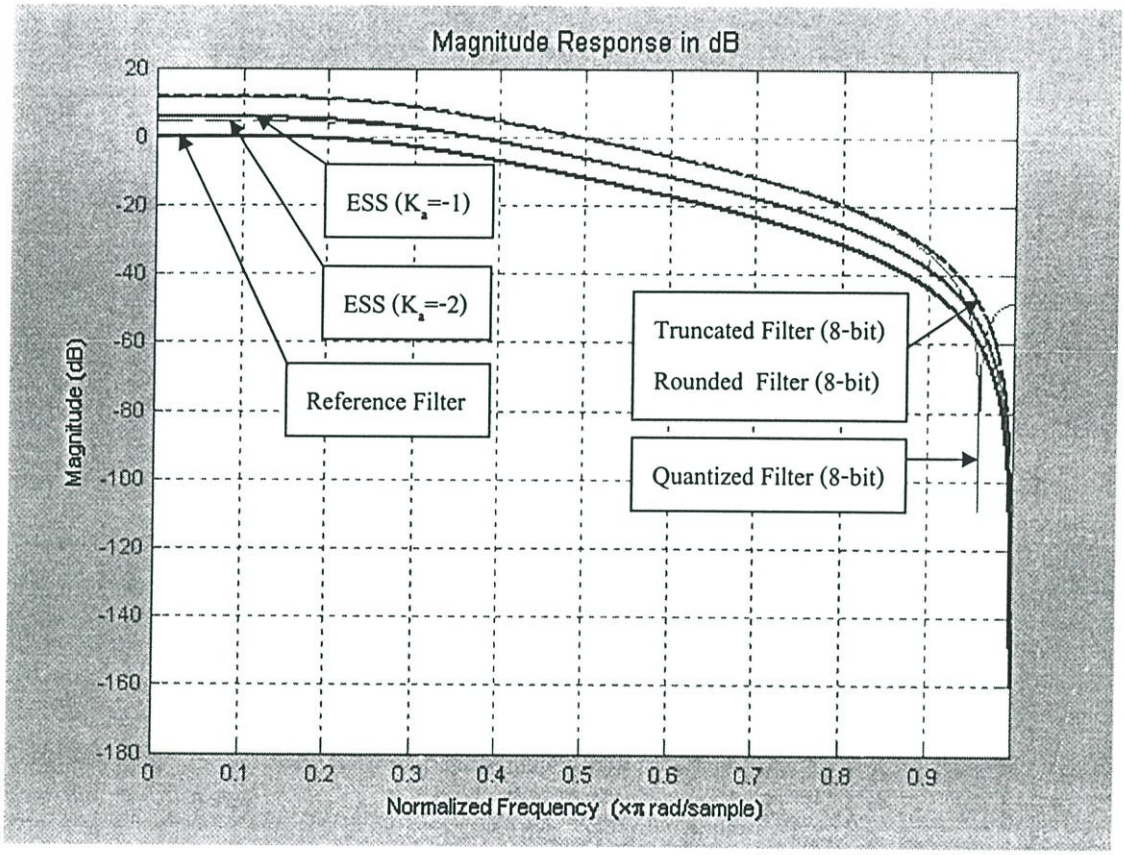
(b) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 3 kHz

(c) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 6 kHz

(d) สัญญาณเข้ารูปไซน์ที่ความถี่ประมาณ 10 kHz

7.2 ผลการตอบสนองความถี่ของวงจรกรองเชิงเลขที่ทดสอบ

ทำการพล็อตค่าผลการตอบสนองทางขนาดเทียบกับความถี่ที่เปลี่ยนแปลงไป จากการวัดขนาดของสัญญาณที่ออกสามารถแสดงผลดังรูปที่ 7.8



รูปที่ 7.8 ผลการตอบสนองความถี่ของวงจรกรองเชิงเลขที่ออกแบบ

เมื่อสังเกตผลที่ได้จากการพล็อตกราฟการตอบสนองความถี่ของวงจรกรองทั้งห้าแบบ ที่ได้ทำการโปรแกรมลงสู่เอฟพีจีเอแล้วจะพบว่าการทำงานของวงจรกรองจะให้ผลการตอบสนองขนาดมีค่ามากกว่าค่าวงจรกรองอ้างอิง (Reference Filter) ซึ่งได้จากการคำนวณบนโปรแกรม MATLAB FDA สาเหตุที่เกิดก็เนื่องจากตัวกรองที่ทำการคำนวณ เมื่อถูกทำการสร้างขึ้นเป็นตัวกรองเชิงเลขด้วยหลักการ DA นั้นถูกทำการปรับขนาดค่าของสัมประสิทธิ์ลง ด้วยการหารสองทำให้ค่าที่ได้ทางเอาท์พุทจะต้องมีค่าต่ำกว่าค่าอ้างอิง แต่เนื่องจากวงจรในส่วนของการแปลงสัญญาณสามารถทำการปรับขนาดแรงดันอ้างอิงได้ทำให้สามารถเพิ่มระดับขนาดสัญญาณนั้นด้วยการปรับแรงดันอ้างอิงแทน เมื่อทำการคำนวณค่าขนาดที่ตอบสนองตามความถี่แล้วจึงให้ค่าที่มากกว่าวงจรกรองอ้างอิง ส่วนกรณีของวงจรกรองที่ทำการแก้ไขความผิดพลาดก็อธิบายได้ในลักษณะเดียวกัน เพียงแต่มีการชดเชยความผิดพลาดที่เกิดขึ้นให้ลดน้อยลงการตอบสนองที่ได้จึงมีค่าน้อยกว่าวงจรกรองควอนไตซ์,ตัดคำและปัดเศษขนาด 8 บิต

7.3 วิเคราะห์ผลการทดลอง

หลังจากที่ทำการโปรแกรมวงจรกรองที่ได้ออกแบบทั้ง 5 ชนิดลงใน เอฟพีจีเอ แล้วทำการป้อนสัญญาณอินพุทไซน์ที่ขนาดไม่เกิน 5 โวลต์ทำการเปลี่ยนค่าความถี่ต่างๆเพื่อตรวจสอบผลการตอบสนองขนาดสัญญาณเมื่อความถี่แปรเปลี่ยนไป แสดงกราฟผลการตอบสนองความถี่ด้วยการใช้โปรแกรม MATLAB พล็อตค่าได้ดังรูปที่ 7.8 ซึ่งจะพบว่า

ผลตอบสนองทางความถี่ของวงจรกรองจากการวัดจริง มีแนวโน้มการทำงานใกล้เคียงกับผลตอบสนองทางความถี่ที่ได้จากโปรแกรม MATLAB คำนวณ โดยเห็นได้ว่าวงจรกรองที่สร้างชนิดการควอนไตซ์ 8 บิต, การตัดปลาย 8 บิตและการปัดเศษ 8 บิต จะให้ผลการตอบสนองความถี่ที่แตกต่างกันน้อยมากจนอาจประมาณได้ว่าให้ผลตอบสนองความถี่ที่เหมือนกัน

เนื่องจากโครงสร้างของวงจรกรองที่ได้ออกแบบมีขนาดข้อมูลอินพุทและเอาท์พุทเป็น 8 บิต ซึ่งถือว่าค่อนข้างต่ำดังนั้นผลกระทบของความยาวคำจำกัดจึงทำให้เกิดความผิดพลาดขึ้น จากการประยุกต์ใช้หลักการ ESS เข้ากับโครงสร้าง DA แล้วเราจะสามารถสร้างวงจรกรองที่แก้ไขความผิดพลาดโดยไม่ต้องทำการเปลี่ยนแปลงโครงสร้างทางฮาร์ดแวร์แต่อย่างใด ดังนั้นแล้วการประยุกต์ใช้หลักการ ESS จึงเหมาะสมกับโครงสร้างของวงจรกรองที่มีขนาดบิตข้อมูลต่ำ จะทำให้ได้วงจรที่มีประสิทธิภาพขึ้น และเนื่องจากยังไม่เคยมีการศึกษาการสร้างวงจรกรอง DA ด้วยหลักการ ESS โดยอุปกรณ์ FPGA มาก่อนดังนั้นประสิทธิภาพในการทำงานของวงจรที่ได้นำเสนอมาสามารถพัฒนางานวิจัยได้อย่างสะดวกและประหยัดอีกด้วย

จากหลักการชดเชยความผิดพลาดด้วยการถ่วงน้ำหนัก K_a, K_b และ K_c ซึ่งอ้างอิงในเอกสาร [6],[8] และ [9] เป็นเพียงหลักการที่ได้นำเสนอไว้ วิทยานิพนธ์นี้ได้แสดงให้เห็นว่า K_a, K_b และ K_c เปรียบเสมือนการสร้างวงจรกรองที่ทำการกำจัดสัญญาณรบกวน หรืออาจเรียกเป็นความผิดพลาด โดยที่ค่าคงที่ K_a, K_b และ K_c ที่เท่ากับ -1 และ -2 ที่ใช้เป็นค่าที่แทนตัวกรองสัญญาณรบกวนออก(Noise Filter) ซึ่งวงจรกรองเชิงเลขชนิดค่า LUT ที่ทำการแก้ไขความผิดพลาดทั้งค่า $K_a = -1$ และ $K_a = -2$ จะให้ผลการตอบสนองความถี่ที่ใกล้เคียงกับผลตอบสนองความถี่ที่คำนวณได้จากโปรแกรม MATLAB แต่จากการทดลองพบว่าที่ $K_a = -2$ จะให้การตอบสนองความถี่ที่เข้าใกล้ค่าจริงของวงจรกรองอ้างอิง กรณีถ้ากำหนดค่าคงที่ K_a, K_b และ K_c มีค่ามากกว่านี้ อาจทำให้วงจรกรองที่ออกแบบเกิดการสั่น (Oscillation)[6] ได้ เนื่องจากเมื่อค่ามากขึ้นจะทำให้ผลคูณกับค่าความผิดพลาดสัมประสิทธิ์มีค่ามากขึ้นจนทำให้การชดเชยมากเกินไป ซึ่งจะทำให้ค่าสัมประสิทธิ์จริงของวงจรกรองเปลี่ยนไป

เมื่อพิจารณาถึงค่าที่จะทำการชดเชยความผิดพลาดจากค่า K_a ค่าที่จะทำการชดเชยจะทำการถูกบวกเพิ่มค่าจากเดิม ณ ตำแหน่ง 2 บิตล่างในตาราง LUT

ถ้าทำการวิเคราะห์ถึงระดับสัญญาณที่วัดได้ จะเห็นว่าที่ความถี่ช่วงเดียวกัน ค่าคงที่ $K_a = -2$ จะให้การชดเชยผลจากความผิดพลาดที่เกิดขึ้นในตัวกรองเชิงเลขได้ดีกว่าค่าคงที่ $K_a = -1$ ซึ่ง

สอดคล้องกับค่า MSE ที่คำนวณได้ในตารางที่ 6.6 ข้อสังเกตที่ได้จากการเลือกใช้ค่าคงที่ชดเชยความผิดพลาด ถ้าพิจารณาแต่เฉพาะค่าสัมบูรณ์ของ MSE ที่คำนวณได้ ขนาดที่เท่ากันจะให้ผลการชดเชยที่ประมาณได้ว่ามีค่าเท่ากัน ผู้ออกแบบจึงต้องพิจารณาค่าคงที่ใด เมื่อแทนในรูปแบบเลขฐานสองแล้วสะดวกก็ให้เลือกทำการพิจารณาใช้ค่านั้นในการทำการชดเชยความผิดพลาด

จะพบว่ารูปแบบในการทำงานของวงจรกรองที่ได้สร้างขึ้นจะมีโครงสร้างที่สอดคล้องกับกระบวนการทางพีชคณิต ซึ่งจุดนี้เองจะเป็นข้อด้อยในการออกแบบวงจรให้มีขนาดที่กะทัดรัด ดังนั้นสิ่งสำคัญในการสร้าง ควรพิจารณาถึงความเหมาะสมของอัลกอริทึมทางคณิตศาสตร์ที่จะใช้สร้าง หรือลักษณะในการกำหนดการทำงานให้ถูกต้องที่สุด

อย่างไรก็ดียังมีวิธีการหนึ่งที่สามารถช่วยเพิ่มความแม่นยำในการแทนข้อมูลให้ถูกต้องมากขึ้น คือการขยายบิตข้อมูลที่ทางออกจากวงจรกรองเชิงเลข แต่จะมีผลต่อวงจรแปลงสัญญาณดิจิตอลไปเป็นสัญญาณอนาลอกที่ต้องทำการเปลี่ยนเป็นขนาดบิตข้อมูลที่สูงขึ้นตาม

ในการทดสอบวงจรกรอง ได้กำหนดให้ใช้ความถี่สุ่มที่ 20 kHz. ป้อนให้กับตัวแปลงสัญญาณอนาลอกเป็นดิจิตอล ที่ใช้ประกอบการทดสอบกับ เอฟพีจีเอ ที่ได้ออกแบบ จะสังเกตได้ว่ารูปคลื่นของสัญญาณขาออกที่ทำการทดสอบที่ความถี่สูงกว่า 10 kHz. ซึ่งอยู่ในย่านใกล้กับค่าความถี่ครึ่งเท่าของความถี่สุ่ม จะมีลักษณะเป็นขั้น ไม่ราบเรียบเหมือนกับรูปคลื่นของสัญญาณขาออกที่ทำการทดสอบในย่านความถี่ที่ต่ำกว่าครึ่งเท่าของความถี่สุ่มมากๆ ซึ่งก็เป็นเรื่องปกติของกระบวนการสุ่มตัวอย่างจากสัญญาณ อนาลอกให้เป็นสัญญาณดิจิตอล ที่คุณภาพของข้อมูลทางดิจิตอล นอกจากจะมีผลมาจากจำนวนบิตที่ใช้ในการควอนไตซ์แล้ว ยังมีผลมาจากค่าความถี่ของการสุ่มอีกด้วย กล่าวคือ ถ้าทำการสุ่มตัวอย่างสัญญาณอนาลอกด้วยจำนวนบิตและด้วยความถี่สุ่มที่ต่ำ สัญญาณดิจิตอลที่ได้ก็จะมีคุณภาพไม่ดีตามไปด้วย หรือกล่าวอีกนัยหนึ่งถ้าสัญญาณอนาลอกมีความถี่สูงขึ้น คุณภาพของสัญญาณดิจิตอลที่ได้จะยิ่งต่ำลง เมื่อพิจารณาให้ระบบมีค่าความถี่สุ่มคงที่

บทที่ 8
สรุปผลการวิจัยและข้อเสนอแนะ

8.1 สรุปผลการดำเนินการวิจัย

การดำเนินงานวิจัยเพื่อจัดทำขึ้นเป็นวิทยานิพนธ์ชุดนี้ เป็นการนำเสนอแนวทางเพื่อการออกแบบวงจรกรองสัญญาณเชิงเลขรีเคอร์ซีฟอันดับที่สอง ด้วยหลักการคณิตศาสตร์แบบการกระจายและการแก้ไขความผิดพลาดด้วยหลักการจัดสเปกตรัมสัญญาณความผิดพลาด ทำการสร้างวงจรบน FPGA โดยเริ่มจากการกำหนดลักษณะเฉพาะของวงจรกรองเชิงเลขที่ต้องการ, การวิเคราะห์ผลการคำนวณจากการควอนไตซ์สัมประสิทธิ์ของตัวกรองเชิงเลข, การคำนวณหาค่าคงที่ที่เหมาะสมเพื่อการชดเชยความผิดพลาดที่เกิดขึ้นกับวงจรกรอง จนถึงการสร้างเป็นวงจรจริง ในส่วนของเครื่องมือที่ใช้ในการออกแบบเลือกใช้ โปรแกรม MATLAB, XilinxISE และ ModelSIM เพื่อทำการออกแบบและจำลองผลการทำงานของวงจรกรองในระดับฟังก์ชันถ่ายโอน และในระดับวงจรตามลำดับ

ในการออกแบบวงจรกรองในระดับฟังก์ชันถ่ายโอนได้ทำการจัดทำโปรแกรมที่ใช้สำหรับคำนวณค่าตารางมองค่า (LUT) ตามรูปแบบการคำนวณฟังก์ชันของสัมประสิทธิ์ของวงจรกรองคณิตศาสตร์แบบการกระจาย และโปรแกรมสำหรับทำการแปลงค่าตาราง LUT ให้อยู่ในรูปแบบเลขฐานสองชนิดส่วนเติมเต็มสองเพื่อทำการสร้างเป็นส่วนประกอบของวงจรในรูปแบบภาษา VHDL ต่อไป รูปแบบวงจรกรองเชิงเลข ไม่ว่าจะเป็นวงจรกรองบัทเทอร์เวิร์ธ วงจรกรองเชบีเชฟ และวงจรกรองอิลลิปติค หรือชนิดอื่นๆ สามารถทำการสร้างวงจรขึ้นได้ด้วยหลักการคณิตศาสตร์แบบการกระจาย โดยอาศัยเพียงค่าสัมประสิทธิ์ของฟังก์ชันถ่ายโอนวงจรกรองที่อยู่ในรูปแบบไบควอดราติกมาทำการสร้างค่าตาราง LUT หากพิจารณากรณีการชดเชยความผิดพลาดก็ทำได้ด้วยการแก้ไขตาราง LUT ทำให้การออกแบบและการแก้ไขวงจรทำได้อย่างง่ายดาย

แต่อย่างไรก็ตามความยุ่งยากในการคำนวณค่าตาราง LUT มีอยู่บ้างเนื่องจากเป็นการคำนวณจำนวนคงที่มากมายความถูกต้องของตำแหน่งทศนิยมจึงมีผลสำคัญต่อค่าตาราง LUT ที่จะทำการสร้างสำหรับวงจรกรองแต่ละแบบ เมื่อในส่วนของการคำนวณค่าคงที่ต่างๆถูกทำการคำนวณเป็นที่เรียบร้อยแล้ว จึงทำการออกแบบโครงสร้างสถาปัตยกรรมคณิตศาสตร์แบบการกระจาย โดยแยกทำการออกแบบในแต่ละส่วนด้วยการบรรยายพฤติกรรมการทำงานของอุปกรณ์นั้นๆ ในรูปแบบภาษา VHDL โดยใช้โปรแกรม XilinxISE และ ModelSIM ทำการออกแบบและจำลองผลการทำงานตามลำดับแล้วจึงทำการโปรแกรมวงจรกรองเชิงเลขที่ได้ออกแบบลงสู่ FPGA ความยืดหยุ่นในการออกแบบและการแก้ไขวงจรสามารถทำได้ด้วยการแก้ไขเพียงโค้ดโปรแกรมที่เขียนให้เหมาะ

สมตามที่ต้องการโดยไม่ต้องทำการเพิ่มเติมหรือแก้ไขอุปกรณ์ฮาร์ดแวร์แต่อย่างใด ทำให้การออกแบบวงจรบน FPGA เหมาะสมอย่างยิ่งในงานพัฒนาการวิจัย และหลังจากที่ได้ทำการโปรแกรมสร้างวงจรกรองเป็นที่เรียบร้อยแล้ว จึงทำการตรวจสอบการทำงานของสัญญาณจริงเพื่อพิจารณาเปรียบเทียบการทำงานของวงจรกรองที่ออกแบบกับผลที่ได้จากการคำนวณโปรแกรม MATLAB จะพบว่าการทำงานของวงจรกรองเชิงเลขในรูปแบบที่แก้ไขความผิดพลาดด้วยค่าขนาด MSE ที่สูงจะให้การแก้ไขความผิพลาดได้ดีขึ้น โดยผลจากการแก้ไขความผิดพลาดสามารถชดเชยไปได้อยู่ในช่วงประมาณน้อยกว่า 10 เดซิเบล ส่วนกรณีวงจรกรองชนิดควอนไตซ์, ตัดปลายและการปัดเศษขนาด 8 บิต จะให้ผลการตอบสนองความถี่ที่ใกล้เคียงกันมาก จนอาจประมาณได้ว่าเหมือนกันเมื่อถูกสร้างขึ้นเป็นวงจรกรองเชิงเลขจริง แต่ให้ค่าความผิดพลาดไปจากผลการคำนวณไปมากกว่า 10 เดซิเบล

ในด้านขนาดของวงจรกรองเชิงเลขที่ออกแบบ เมื่อพิจารณาผลการสังเคราะห์วงจร จะใช้ปริมาณทรัพยากรณ์บน FPGA ไปประมาณ 25 เปอร์เซ็นต์ซึ่งถือว่าน้อยมากเมื่อเทียบกับการสร้างวงจรกรองชนิดเดียวกันด้วยวิธีการอื่นที่รายงานอื่นได้นำเสนอมา ในด้านประสิทธิภาพความเร็วในการทำงานของวงจรกรองที่ออกแบบสามารถตอบสนองความถี่ทำงานสูงสุดได้ถึง 143.225 MHz จำนวนเกทที่ใช้ไปมีค่า 825 เกท แต่ด้วยขีดจำกัดของอุปกรณ์ฮาร์ดแวร์ด้านอินพุทและเอาท์พุทที่ถูกใช้เชื่อมต่อการแปลงสัญญาณที่ไม่สามารถตอบรับสนองการทำงานได้เท่ากับ FPGA ทำให้เกิดข้อจำกัดในการกำหนดลักษณะเฉพาะของวงจรขึ้น จึงทำให้ต้องพิจารณาผลของตัวแปรค่าหลายค่าไม่ว่าจะเป็น ช่วงความถี่ตัดที่ออกแบบ, ความถี่ของสัญญาณสุ่ม, ความถี่ของสัญญาณความคุมระบบการทำงาน เป็นต้น ทุกตัวแปรดังกล่าวมีผลต่อการตอบสนองขีดความสามารถที่จะทำได้ถึง จึงจำเป็นต้องทำความเข้าใจในส่วนนี้เป็นอย่างดี เพื่อให้การทำงานสามารถทำได้ในช่วงที่ออกแบบ

8.2 แนวทางในการพัฒนา

จากที่ได้นำเสนอมาทั้งหมดข้างต้น รูปแบบในการคำนวณที่ได้นำเสนอไว้เป็นรูปแบบคณิตศาสตร์แบบการกระจาย ชนิดการทำงานแบบ บิตอนุกรมเวิร์ดขนาน ซึ่งโดยแท้จริงแล้วหลักการคณิตศาสตร์แบบการกระจายยังมีชนิดย่อยอื่นอีก เช่น การทำงานแบบขนานของคณิตศาสตร์การกระจาย ซึ่งจะให้ประสิทธิ์ภาพในการคำนวณที่รวดเร็วที่สุดแต่จำเป็นต้องใช้ทรัพยากรบน FPGA มากกว่า ซึ่งอาจเป็นอีกแนวทางในการพัฒนาให้การทำงานของวงจรกรองทำได้ที่ความถี่สูงๆ แต่อย่างไรก็ตามวงจรที่ใช้ทำการแปลงสัญญาณระหว่าง ดิจิตอลและอนาลอกก็ต้องมีประสิทธิ์ภาพในการแปลงที่รวดเร็วสอดคล้องกับระบบ FPGA ด้วย

ในด้านความแม่นยำของการคำนวณและการแทนค่าข้อมูลที่เอาท์พุทสามารถขยายเพิ่มความแม่นยำขึ้นด้วยการเพิ่มจำนวนบิตผลลัพธ์ให้มากขึ้น แทนการตัดบิตทิ้งเช่นแสดงในงานวิจัยนี้ จะทำให้ได้ระดับของสัญญาณที่ละเอียดขึ้น

สำหรับกรณีที่การใช้งานวงจรกรองรีเคอร์ซีฟในทางปฏิบัติแล้วอันดับของตัวกรองเพียงแค่อันดับสองไม่สามารถให้ผลการตอบสนองความถี่ ตลอดจนความคมของช่วงความถี่ตัดได้ดีพอ จึงจำเป็นต้องใช้อันดับของตัวกรองที่สูงขึ้นมากๆ เราก็สามารถให้วงจรกรองอันดับที่สองที่นำเสนอในวิทยานิพนธ์นี้ทำการต่อพวง (cascade) กันจะทำให้อันดับของวงจรกรองเพิ่มขึ้นเป็นอันดับที่ 2 N ถ้า N เป็นจำนวนชุดที่ทำการต่อพ่วง แต่อย่างไรก็ดีต้องทำการแก้ไขในส่วนของวงจรควบคุมการทำงานด้วยเพื่อให้การทำงานทั้งระบบรวมเป็นไปอย่างถูกต้อง

ทางด้านสถาปัตยกรรมของหน่วยคำนวณสะสมค่า สามารถแก้ไข หรือ เปลี่ยนรูปแบบเป็นชนิดอื่นได้ เช่นอาจใช้ตัวบวกแบบลุคอัพเทเบิ้ลเพื่อลดความหน่วงเวลาที่เกิดจากการคำนวณได้ รูปแบบสถาปัตยกรรมใดที่นำมาเลือกใช้ชนิดต่างๆ ล้วนมีข้อดีข้อเสียคู่กัน กล่าวคือ ถ้าการทำงานที่ได้รวดเร็วขึ้น จะมีผลกลับกันในด้านขนาดวงจรคือมีขนาดวงจรที่ใหญ่ขึ้น จึงจำเป็นต้องเลือกตามความเหมาะสมของระบบ

เนื่องจากการคำนวณในส่วนต่างๆที่ได้นำเสนอมามีจำนวนที่มาก การเขียนโปรแกรมเพื่อช่วยในการคำนวณจะช่วยในการอำนวยความสะดวก รวดเร็ว ถูกต้องยิ่งขึ้น ดังนั้นหากการออกแบบโครงสร้างที่ซับซ้อนจำเป็นต้องวิเคราะห์ให้กระจ่างเพื่อตัดปัญหา ความผิดพลาดเพียงเล็กน้อยที่เกิดอาจทำให้ระบบไม่สามารถทำงานได้อย่างถูกต้อง

ขั้นตอนในการออกแบบวงจรกรอง จะต้องใช้เครื่องมือที่เป็นซอฟท์แวร์หลายชนิด ซึ่งนับว่ายังไม่สะดวกแก่ผู้ที่เริ่มออกแบบวงจร จึงมีการพัฒนาแนวคิดที่จะรวมเครื่องมือต่างๆ เข้าด้วยกันโดยใช้ซอฟท์แวร์เพียงตัวเดียว ทำหน้าที่ออกแบบวงจรกรองตั้งแต่ระดับฟังก์ชันถ่ายโอนจนถึงระดับวงจรได้ในคราวเดียว อีกประการหนึ่ง การออกแบบวงจรกรองเชิงเลขยังมีแนวทางการใช้ตัวประมวลได้อย่างหลากหลายประเภท ในแต่ละประเภทก็มีข้อดีข้อเสียที่แตกต่างกันไป การสร้างหรือเลือกใช้ตัวประมวลผลที่เหมาะสมสำหรับงานประยุกต์บางงานจึงเป็นเรื่องที่ต้องศึกษาและพัฒนากันต่อไป

การออกแบบโดยใช้เอฟพีจีเอ มีความเหมาะสมในการวิจัยและพัฒนาขั้นต้น เนื่องจากเป็นการประหยัดทั้งเวลา และค่าใช้จ่ายในการออกแบบและทดสอบ ซึ่งเมื่อเสร็จสิ้นการทดสอบในขั้นตอนสุดท้ายแล้ว อาจนำผลที่ได้จากการออกแบบด้วย เอฟพีจีเอ มาพัฒนาสู่การสร้างเป็นวงจรรวมที่ใช้เฉพาะงาน (ASIC: Application Specific Integrated Circuit) ในเชิงพาณิชย์ได้ในอนาคต

เอกสารอ้างอิง

[1] Abraham Peled, Bede Liu, "A New Hardware Realization of digital Filters" IEEE Trans. On A.S.S.P. December 1974, vol. 22, No.6,

[2] Stanley A. White, "Applications of Distributed Arithmetic to Digital Signal Processing: a tutorial review" IEEE Trans. On A.S.S.P. July 1989, pp 4-19

[3] Bernie New, "A Distributed Arithmetic approach to designing Scalable DSP Chips", EDN access for design by design, August 17, 1995.

[4] Radhika S.Grover, Weijia Shang and Qiang Li, "A Fast Distributed Arithmetic Architecture for FPGAs"

[5] Gregory Ray Goslin, "A Guide to Using Field Programmable Gate Arrays (FPGAs) for Application Specific Digital Signal Processing Performance" V.1.0, 1995 Xilinx,Inc.

[6] W.Surakampontorn, M.I.Sobhy, "Self-error feedback in recursive digital filters" IEE Procedings, Vol.130 Pt.G.No.5, October 1983.

[7] Newguard Bruce, "Seminar: Signal Processing with Xilinx FPGAs" Xilinx Publication, June 1996.

[8] A.Djebba Ri, J.M.Rouvaen, L.Camus, "Noise reduction in recursive digital filters using optimal and suboptimal error feedback" INT.J. Electronics 1997, vol.83, NO.6, 743-751

[9] Timo I. Laakso, Iiro O. Hartimo, "Noise Reduction in Recursive Digital Filters Using High-Order Error -Feedback" IEEE Trans., On Signal Processing Vol.40 No.5, May 1992.

[10] Digital Signal Processing: A Computer Based Approach, 2nd Edition, Sanjit K. Mitra, Mc Graw Hill.

[11] First Principles of Discrete Systems and Digital Signal Processing, Robert D. Strum, Donald E. Kirk, Addison-Wesley Publishing Company.

[12] Digital Filters: Analysis, Design, and Applications, 2nd Edition, AnDreas Antoniou, Mc Graw Hill.

[13] การประมวลผลสัญญาณเชิงเลข, ศ.ดร.วัลลภ สุระกำพลธร, สถาบันเทคโนโลยีพระจอมเกล้าฯ เจ้าคุณทหารลาดกระบัง, พ.ศ.๒๕๓๓

[14] การประมวลผลสัญญาณเชิงเลขเบื้องต้น, รศ.ดร.สมศักดิ์ ชุมช่วย, สถาบันเทคโนโลยี พระจอมเกล้าฯ เจ้าคุณทหารลาดกระบัง, พ.ศ.๒๕๔๕

ภาคผนวก

ภาคผนวก ก.

โปรแกรมแมทแล็บที่ใช้ออกแบบวงจรกรองดิจิตอลชนิดคณิตศาสตร์แบบการกระจาย

ก.1 โปรแกรม quan.m

```
function c=quan(coef,b)
    %  quant(coef,b) rounds the elements of the vector or matrix
    % 'coef' to (b) fractional bits past the binary point.
     c=round(2^b*coef)/2^b;
```

ก.2 โปรแกรม qda.m

```
clear;
clc;
%Use this program for computing the look-up table
%of Distributed arithmetic Digital Filter
%Idea:
%Get input number of coefficients :          "n = input('Please type the number of input  : ');"
%Get the number of wated Bits significant:  "bit = input('Please type the number of Bits   : ');"
%Bit size of Addresses = Bits quantized
%number of addresses = 2^n;
%Calculate the summing result in lines of all addresses
%Quantize the result then convert to binary number
%Convert to One's complement number
%And last display all result
n = input('Please type the number of input  : ');
bit = input('Please type the number of Bits   : ');
            %Bit size of Addresses = Bits quantized
for i=1:n;     %recieve n input value
  c(i) = input('Please type i Coefficient  :');
end
  c         %show all n input
```

```
address = 2^n;  %number of address
disp('##################################################################################')
disp('#                                                                                #')
disp('#   Address      f     F     Fsum      Fsunq      Qerror      Fqtz      ~Fqtz     #')
disp('#                                                                                #')
disp('##################################################################################')
for adr = 0:address-1;
   y = dec2bin(adr,bit);
   for m = 1:n;
   f(m) = bitget(bin2dec(y),m);
   end
   F = f(1:m).*c(1:n);
    Fsum = 0;
    for i=1:n;
    Fsum = F(i)+Fsum;
    end
  temp = Fsum;
  temp = quan(temp,bit) ; %quantized coefficient
     %  function c=quan(coef,b)
     %  quant(coef,b) rounds the elements of the vector or matrix
     % 'coef' to (b) fractional bits past the binary point.
     %  c=round(2^b*coef)/2^b;
 Fsumq = temp; %collect quantized value for calculate error in quantized effect
 Qerror= Fsum-Fsumq;
  for k = 1:bit;
     temp = temp*2;
     if temp >= 1
        b(k) = 1;
        temp = temp-1;
```

```
        else b(k) = 0;
        end
    end
    b;
    bcom = ~b;
%% Below is Display Table
    disp(y),disp(f),disp(F),disp(Fsum),disp(Fsumq),disp(Qerror),disp(b),disp(bcom)
disp('##################################################################################')
end
disp('##################################################################################')
```

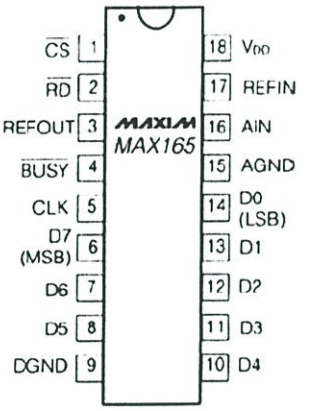
ก.3 โปรแกรม f2com.m

```
clear;
clc;
n = input('Please type the number of input  : ');
bit = input('Please type the number of Bits   : ');
for i=1:n;      %recieve n input value
   c(i) = input('Please type i Coefficient  :');
end
  c
   D = c*2^(bit-1);
   Af = round(D)
   for  i = 1:n;
      if Af(i) >= 0
         dec2bin (Af(i),8)
      else
         dec2bin ((Af(i) + 2^bit),8)
      end
   end
```

ภาคผนวก ข.

วงจรทดสอบ

ข.1 วงจรแปลงสัญญาณแอนะล็อกเป็นดิจิตอล

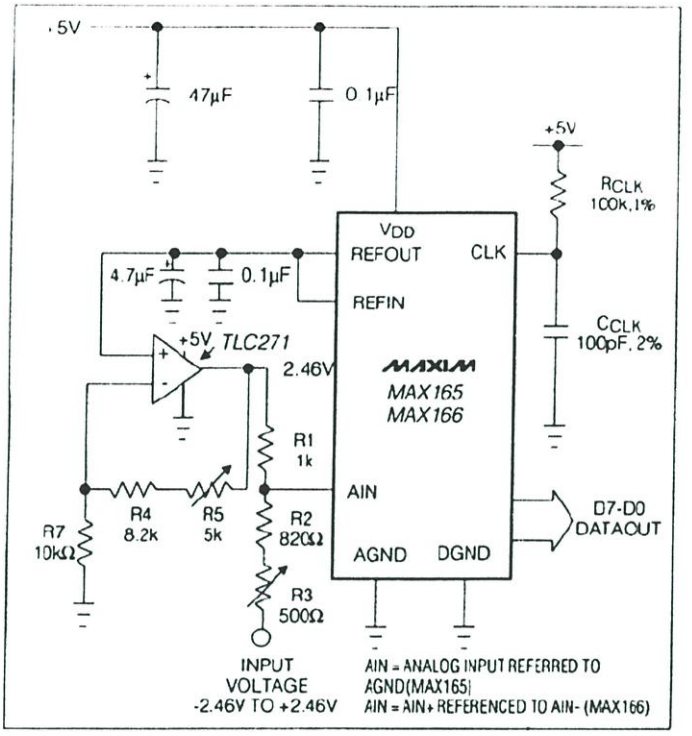


รูปที่ ข.1 การจัดขาของอุปกรณ์

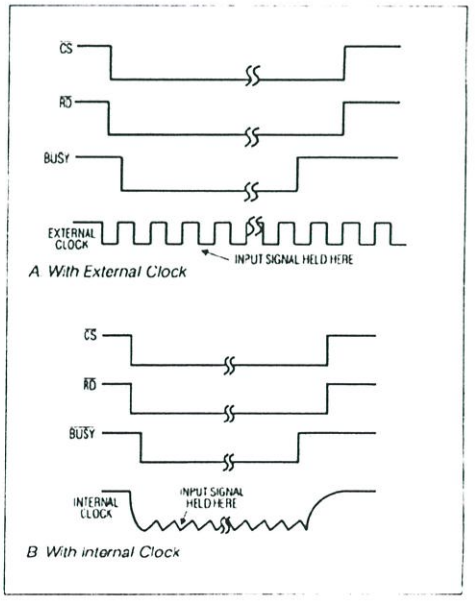
PIN		NAME	FUNCTION
MAX165	MAX166		
1	1	$\overline{CS}$	CHIP SELECT Input. $\overline{CS}$ must be low for the device to be selected, or to recognize the $\overline{RD}$ input.
2	2	$\overline{RD}$	READ Input. $\overline{RD}$ must be low to access data. $\overline{RD}$ is also used to start conversions. See the *Digital Interface* section
3	3	REFOUT	Output of the internal 1.23V bandgap reference
	4	MODE	MODE (MAX166). Mode = low puts the ADC into asynchronous-conversion mode. MODE has to be tied high for synchronous-conversion mode and ROM interface mode
4	5	$\overline{BUSY}$	$\overline{BUSY}$ Output $\overline{BUSY}$ going low indicates the start of a conversion. BUSY going high indicates the end of a conversion.
5	6	CLK	External Clock Input/Internal Oscillator Pin for frequency setting RC components.
6	7	D7 (MSB)	Three State Data Output, bit 7 (MSB)

PIN		NAME	FUNCTION
MAX165	MAX166		
7	8	D6	Three-State Data Output, bit 6
8	9	D5	Three-State Data Output, bit 5
9	10	DGND	Digital Ground
10	11	D4	Three-State Data Output, bit 4
11	12	D3	Three-State Data Output, bit 3
12	13	D2	Three-State Data Output, bit 2
13	14	D1	Three-State Data Output, bit 1
14	15	D0 (LSB)	Three-State, Data Output, bit 0 (LSB)
15	16	AGND	Analog Ground
16		AIN	Analog Input - (single-ended with respect to AGND) 0V to 2VREF input range
	17	AIN-	Negative Analog Input differential (MAX166)
	18	AIN+	Positive Analog Input differential (MAX166)
17	19	REFIN	Reference Input +1.23V nominal
18	20	V_{DD}	Power-Supply Voltage +5V nominal

ตารางที่ ข.1 ลักษณะการทำงาน

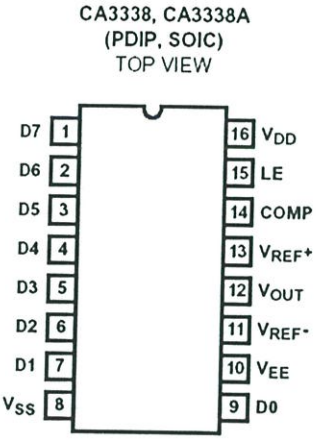


รูปที่ ข.2 การต่อวงจร



รูปที่ ข.3 แผนภาพการทำงาน

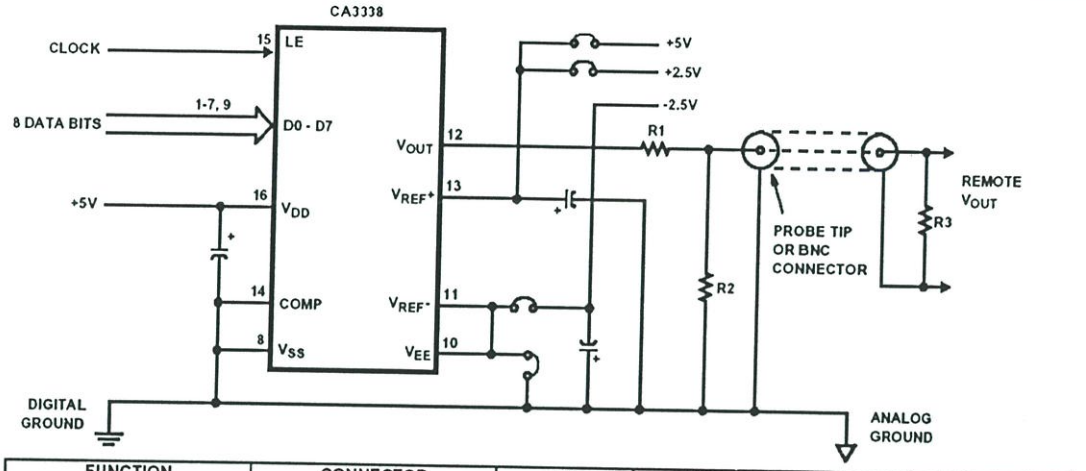
ข.2 วงจรแปลงสัญญาณดิจิตอลเป็นแอนะล็อก



รูปที่ ข.4 การจัดขาของอุปกรณ์

PIN	NAME	DESCRIPTION
1	D7	Most Significant Bit
2	D6	Input
3	D5	Data
4	D4	Bits
5	D3	(High = True)
6	D2	
7	D1	
8	V_{SS}	Digital Ground
9	D_0	Least Significant Bit. Input Data Bit
10	V_{EE}	Analog Ground
11	V_{REF}-	Reference Voltage Negative Input
12	V_{OUT}	Analog Output
13	V_{REF}+	Reference Voltage Positive Input
14	COMP	Data Complement Control input. Active High
15	LE	Latch Enable Input. Active Low
16	V_{DD}	Digital Power Supply, +5V

ตารางที่ ข.2 ลักษณะการทำงาน



FUNCTION	CONNECTOR	R1	R2	R3	V_{OUT} (P-P)
Oscilloscope Display	Probe Tip	82Ω	62Ω	N/C	1V
Match 93Ω Cable	BNC	75	160	93	1V
Match 75Ω Cable	BNC	18	130	75	1V
Match 50Ω Cable	BNC	Short	75	50	0 79V

NOTES:

3. $V_{OUT(P-P)}$ is approximate, and will vary as R_{OUT} of D/A varies.
4. All drawn capacitors are 0.1μF multilayer ceramic/4.7μF tantalum.
5. Dashed connections are for unipolar operation. Solid connection are for bipolar operation.

รูปที่ ข.5 การต่อวงจร

ภาคผนวก ค.

บทความที่ได้รับการตีพิมพ์

ISCIT2003

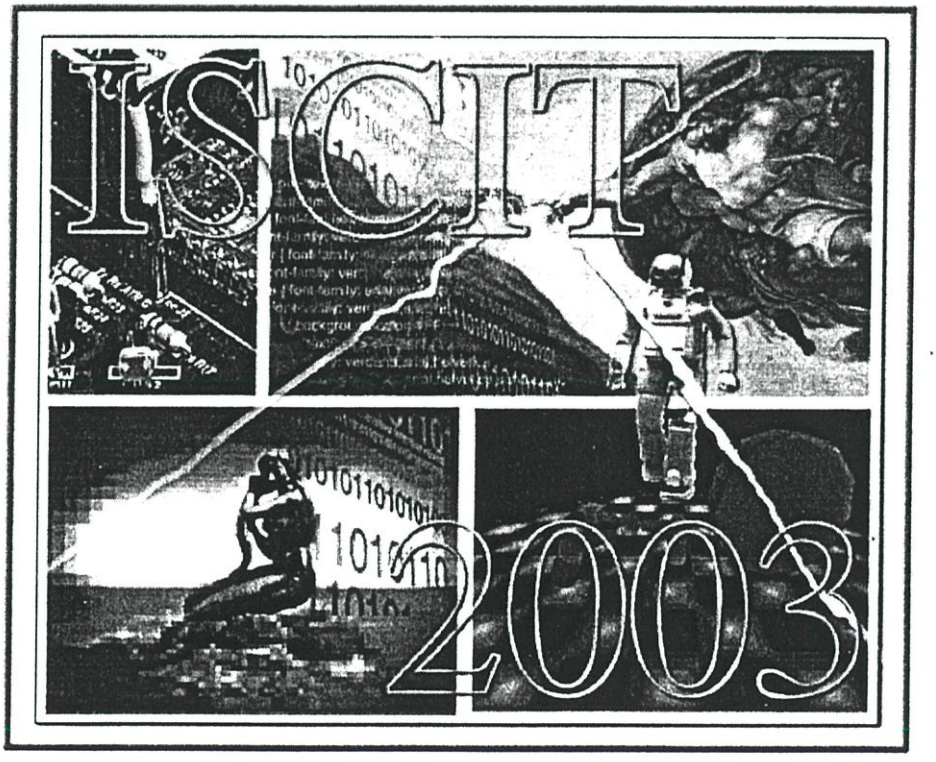
Volume II

Proceedings

The Third International Symposium on Communications and Information Technologies

September 3-5, 2003

BP Samila Beach Hotel and Resort, Songkhla, Thailand



ISBN 974-644-437-9

A Programmable FPGA-based Distributed Arithmetic Digital Filter

M. Santawakoop, C. Fongsamut, A. Kantichanakul and W. Surakampontorn*

Department of Electronics, Faculty of Engineering,
King Mongkut's Institute of Technology Ladkrabang,
Bangkok, Thailand 10520
** s4061319@kmitl.ac.th*

Abstract

A FPGA-based recursive digital filter using Distributed Arithematic is presented in this paper, where the characteristic of the filter can be programmed. The filter round-off error is reduced by employing an error spectrum shaping method. The whole system is designed by VHDL and implemented on Xilinx's FPGA (Spartan II-xc2s50). Furthermore, the hardware size is small and no addition other equipment (same structure). The experimental results are also included.

Keywords

Digital Filter, Distributed Arithmetic(DA), Error Feed-back (EF), Look Up Table(LUT), Scaling Accumulator

1. Introduction

FPGAs are an array of programmable logic cells interconnected by a matrix of programmable switches. The logic cells communicate with the outside world using programmable input/output blocks. The architecture of FPGAs makes them suitable for dedicated functions like Digital Signal Processing (DSP). The obtainable benefits can be achieved by mapping algorithms to FPGAs. It becomes a viable solution for making custom chips and programmable DSP devices. Most of the DSP algorithms require multiplication and addition function that called MAC (multiply accumulate). However, Distributed Arithmetic (DA) [1-3] provides an approach for multiplier-less implementation of DSP systems. It is an algorithm that can perform multiplication with lookup table (LUT) based schemes (also called DALUT) [4]. DA specifically targets the sum of products (also referred to as the vector dot product) computation that is found in several of the DSP filtering and frequency transforming functions. Combined with Xilinx FPGA lookup table architecture, the DA algorithm was shown to produce very efficient filter designs [5]. Nevertheless, round off error or round off noise can be appeared during the computation. It has been shown that an Error Feedback (EF) or Error spectrum shaping (ESS) can be applied to reduce the noise [6-9]. Using feedback, it is possible to introduce zeros in the pass band of the filter to shape the spectrum of the quantization error.[6] The designs are implemented on a Xilinx FPGA (spartan2 xc2s50-tq144-5) using Xilinx ISE tools.

In this paper, a brief overview of DA algorithm is presented. And the error feedback will be utilized in DA structure, where the lookup table content of memory will be modified to include noise.

2. Basic principle

2.1 Distributed Arithmetic (DA)

Usually, the recursive digital filter defines the relationship of the input signal and the output signal, by the difference equation as eqn.(1)

$$Y_n = \sum_{k=0}^{M} a_n X_{n-k} - \sum_{k=1}^{N} b_n Y_{n-k} \qquad (1)$$

where a_n and b_n are constant coefficients; X_n and Y_n are the input and output data respectively.

From eqn.(1), the transfer function of a 2nd order recursive digital filter (IIR), M=N=2 can be determined by

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \qquad (2)$$

The 2nd order IIR-digital filter in eqn.(2) can be demonstrated in a Direct-form structure as shown in Fig.1

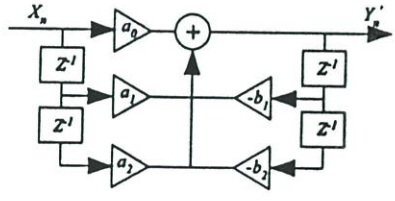


Fig.1 The 2nd order IIR digital filter structure

If X_n and Y_n are represented in (B+1) bits signed two's complement number as given in the eqn. (3)

$$X_n = -x_n^0 + \sum_{j=1}^{B} x_n^j \Box^{-j} \quad \text{and} \quad Y_n = -y_n^0 + \sum_{j=1}^{B} y_n^j \Box^{-j} \qquad (3)$$

Substituting eqn. (3) in (1), and reforming in compact form Y_n can be expressed as in eqn (4).

$$Y_n = \sum_{j=1}^{B} 2^{-j} F(X_n^j, X_{n-1}^j, X_{n-2}^j, Y_{n-1}^j, Y_{n-2}^j) - F(X_n^0, X_{n-1}^0, X_{n-2}^0, Y_{n-1}^0, Y_{n-2}^0) \quad (4)$$

where F(.) is

$$F(X_n^j, X_{n-1}^j, X_{n-2}^j, Y_{n-1}^j, Y_{n-2}^j) = a_0 X_n^j + a_1 X_{n-1}^j + a_2 X_{n-2}^j - b_1 Y_{n-1}^j - b_2 Y_{n-2}^j \quad (5)$$

Normally, 2^5 possible values of F (.) are generated from eqn. (5), rounded to B bits and stored in the memory. The signals $X_n^j, X_{n-1}^j, X_{n-2}^j, Y_{n-1}^j, Y_{n-2}^j$ are tapped from shift registers for addressing the data result contents in memory. The output Y_n is obtained by B+1 consecutive additions in the accumulator, which is shifted from the memory output [6].

2.2 Error Feedback and DA

Error Feedback (EF) is a general method that is useful in reducing the error inherent in any quantization operation. Thus, it can also help to reduce the quantization errors introduced in finite word length implementations of IIR filter [8]. To implement the system of eqn.(1), usually, the quantized value of the function F(.) is stored. However, if we apply the EF for improving the round off error, thus the eqn.(4) can be rewritten as eqn.(6) [6].

$$Y_n = \sum_{j=1}^{B} 2^{-j} \left\{ F_q(X_n^j, \ldots) + F_e(X_n^j, \ldots) \right\} - \left\{ F_q(X_n^0, \ldots) + F_e(X_n^0, \ldots) \right\} \quad (6)$$

where $F_q(.)$ is the quantized function of $F(.)$ and, $F_e(.)$ is the noise function (depended on addressing). If $\bar{a}_0, \bar{a}_1, \bar{a}_2, \bar{b}_1$, and $\bar{b}_2$ are quantized coefficients. then

$$F_q(.) = \bar{a}_0 X_n^j + \bar{a}_1 X_{n-1}^j + \bar{a}_2 X_{n-2}^j - \bar{b}_1 Y_{n-1}^j - \bar{b}_2 Y_{n-2}^j \quad (7)$$

and,

$$F_e(.) = (a_0 - \bar{a}_0) X_n^j + (a_1 - \bar{a}_1) X_{n-1}^j + (a_2 - \bar{a}_2) X_{n-2}^j - (b_1 - \bar{b}_1) Y_{n-1}^j - (b_2 - \bar{b}_2) Y_{n-2}^j \quad (8)$$

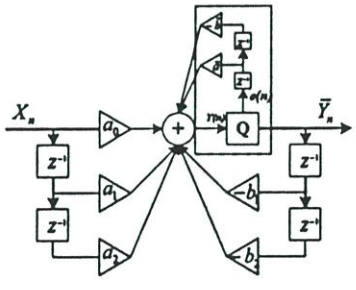


Fig.2 The 2nd order recursive digital filter with 2nd order quantizer error feedback structure

For the 2nd-order quantizer dash line block (error feedback block) shown in Fig. 2, the round off error occur in this quantizer that can be given by

$$\varepsilon_n = Y_n - \bar{Y}_n = e(n) - \tilde{a}e(n-1) + \tilde{b}e(n-2) \quad (9)$$

where the errors $e(n-1)$ and $e(n-2)$ in eqn.(9) can be pre-computed and approximated from eqn.(8), as follows

$$e(n-1) = (a_0 - \bar{a}_0) X_{n-1} + (a_1 - \bar{a}_1) X_{n-2} - (b_1 - \bar{b}_1) Y_{n-2} \quad (10)$$

$$e(n-2) = (a_0 - \bar{a}_0) X_{n-2} + (a_2 - \bar{a}_2) Y_{n-2} \quad 11)$$

Actually, $\tilde{a}$ and $\tilde{b}$ in eqn.(9) are difficult to find. Therefore, $\tilde{a}$ and $\tilde{b}$ in integer format can be used to search the grid point that will give the minimum noise. Then $F_e(.)$ becomes

$$F_e(.) = K_a \left\{ (a_0 - \bar{a}_0) X_{n-1}^j + (a_1 - \bar{a}_1) X_{n-2}^j - (b_1 - \bar{b}_1) Y_{n-2}^j \right\} + K_b \left\{ (a_0 - \bar{a}_0) X_{n-2}^j + (a_2 - \bar{a}_2) Y_{n-2}^j \right\} + K_c (a_2 - \bar{a}_2) Y_{n-1}^j \quad (12)$$

where K_a, K_b and K_c are integer weighting factors and accordingly and the difference filter will give the difference factors. Now the memory contents of the filter are the summing of the rounded values (4) and the pre-computed error of (9). Consequently, the filter can be represented as shown in Fig.3, which is same to that given by Peled and Liu [1], only the changing in memory contents are different. By programming LUT in VHDL formatted will give efficiently a programmable filter.

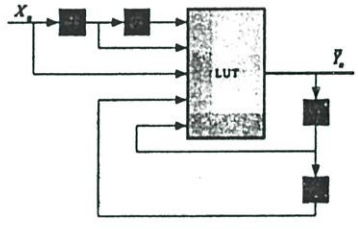


Fig.3 The 2nd-order IIR filter with error feedback

It should be noted that the round off bits are already stored in the memory. The accumulator word length can be kept to equal to the data word length, and/or extend it for the high precision filter.

3. Designing steps

The designing steps of this filter, can be separate in two main parts as shown in Fig.4.

1) Defining the Filter Specification: By using Matlab, the filter frequency response, filter coefficients, quantized coefficients are calculated. The quantized coefficients plus round off are converted to signed binary 2'complement number for programming the filter.

2) Designing through HDL language: From the DA filter structure as shown in Fig.3, we can design all components using VHDL language. Those component are comprised of parallel to serial shift registers, a look up table memory, a scaling accumulator, and controller unit. XilinxISE and Model SIM are used to design, synthesis, implement the VHDL components, and timing simulation.

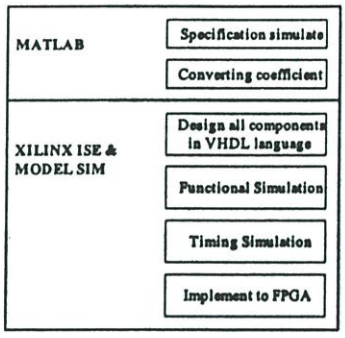


Fig.4 Steps in designing 2nd-order DA-IIR filter

Fig.5 shows the schematic diagram of the programmable 2nd-order DA IIR filter, which consists of a 8-bit PISO (Parallel in Serial out-shift register), four 8-bit SISOs (Serial in Serial out-shift registers), a 8-bit LUT (Look up table memory), and a parallel Add/Sub accumulator. The summation of the quantized coefficient, and noise error values were stored in LUT. Besides, the double precision block can be extended to the structure [7].

A brief description of the Fig.5 can be explain as following. The initial condition of the two taps (y_{n-2} and y_{n-1}) addressing from the SISOs is set to "0". During the 8-bits input data x_n from ADC is fed to a PISO, then, it is serially passed to the next registers. Each tap from the registers are connected to addressing the memory content. Every clock cycles, the data output from LUT and the filter output y_n are extended one-bit (at signed bit), after the addition in accumulator the output is valid.

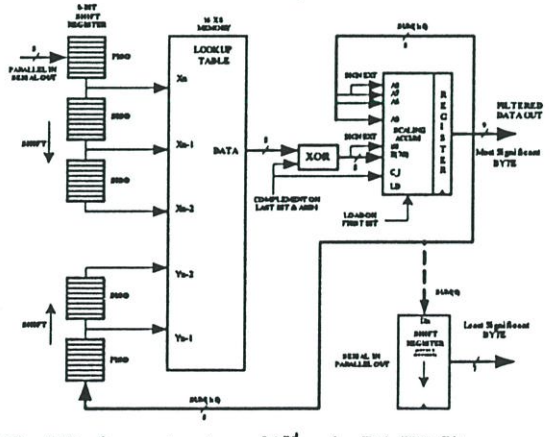


Fig.5 Hardware structure of 2nd-order DA IIR filter with error feedback

4. Simulation results and Filter Implementation

In this paper, two recursive digital filters are considered

(1) 2nd order Elliptic filter with ω_p= 0.4, A_p = 1dB, and A_s = 50dB

(2) 2nd order Butterworth filter with ω_p = 0.3

where ω_p is normalized passband frequency, A_p is the passband attenuation, and A_s is the stopband attenuation.

Using Matlab simulation, table 2 (a) and (b) are comparison of the reference filter coefficients and the quantized coefficients.

	Co	Quantized Coef.	Reference Coef.
Numerator	b_0	0.999969482421875	1.00000000000000000
	b_1	0.999969482421875	1.97587219341086810
	b_2	0.999969482421875	0.99999999999999922
Denominator	a_0	0.999969482421875	1.00000000000000000
	a_1	-0.350677490234375	-.35066371873525548
	a_2	0.330718994140625	.330717675431367320

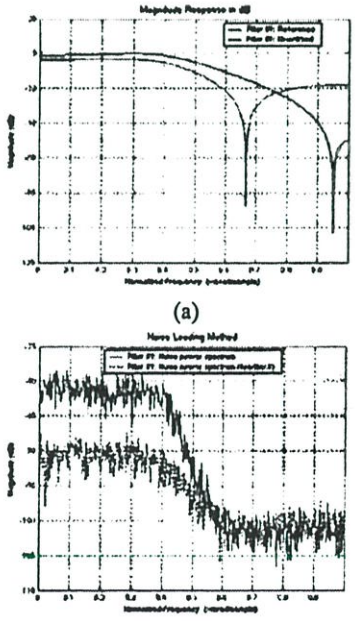
(a)

	Co	Quantized Coef.	Reference Coef.
Numerator	b_0	0.999969482421875	1.00000000000000000
	b_1	0.999969482421875	2.00000000000000000
	b_2	0.999969482421875	1.00000000000000000
Denominator	a_0	0.999969482421875	1.00000000000000000
	a_1	-0.747802734375000	-.74778917825850344
	a_2	0.272216796875000	0.27221493792500728

(b)

Tab.1 (a) The 2nd order Elliptic coefficients,(b) The 2nd order Butterworth coefficients

In Fig.6(a) and Fig.7(a) show the magnitude response of the two filters, which were effected by the quantization transfer function.

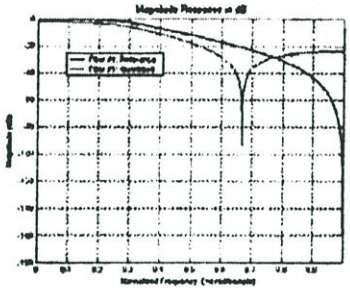


(a)

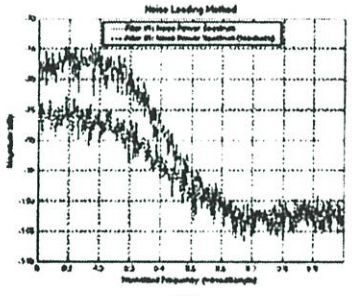
(b)

Fig.6 The 2nd order IIR Elliptic ω_p= 0.4, A_p = 1dB, A_s = 50dB (a) Magnitude response. (b) Noise Loading Method.

From the Fig.6 (b) and Fig.7 (b), the solid line is a noise power spectrum of quantized filter, and the dashed line is a noise power spectrum of quantized filter with error feedback. We can see that the noise power spectrum of the EF filter can be attenuated approximate -10dB and -7dB respectively, at the transition band.



(a)



(b)

Fig.7 The 2nd order IIR Butterworth ω_p= 0.3
(a) Magnitude response. (b) Noise Loading Method.

After, the specification of filter was defined and simulated. Then we start to design the hardware structures, which are realized on Xilinx FPGA (spartan2 xc2s50-tq144-5). The block components of DA-IIR filter structure from Fig.5 are written by VHDL code, after that, synthesis the VHDL code and implementation on FPGA. Fig.8 is shown the timming signals of the filter, that include the delay time on FPGA. The floor planner of realized structure can be shown in Fig.9.

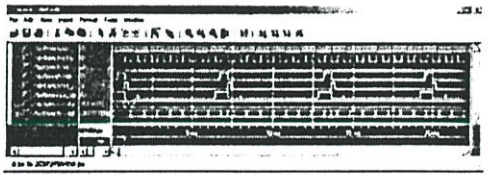


Fig.8 The timming simulation of a 2nd order DA-IIR filter.

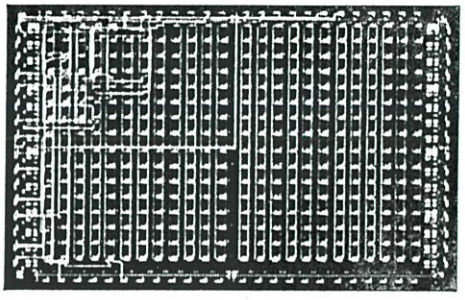


Fig.9 The floor planner of a 2nd order DA-IIR filter.

The results of implementation DA-IIR filter utilizes the environment of FPGA are as following.

Number of Slice Flip Flops:	47 out of 1,536	3%
Total Number 4 input LUTs:	30 out of 1,536	1%
Total equivalent gate count for design:	635	
Maximum Frequency:	131.631MH	

8-bits Analog to digital converter (ADC) MX7821 and Digital to analog converter (DAC) MX7224 were used in this research, and the connection of hardware circuits is shown below.

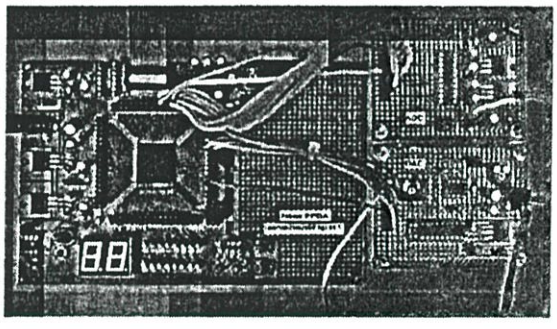


Fig.10 Interfacing analog input signal and digital output signal to FPGA board.

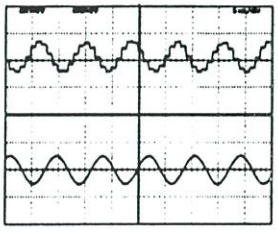


Fig.11 Analog input signal versus anolog output signal.

Cut off frequency 300KHz of 2nd order Butterworth filter is programed to the FPGA. The analog input signal with the amplitude of 2Vp-p at 120KHz is fed to ADC

MX7821, a sampling rate 1MHz, given the digital input data to FPGA. The digital output signal from FPGA is connected to DAC MX7224, conversion for analog signal as shown in Fig.11. In the transition band, the output signal without Error Feedback was gave noise magnitude more than Error Feedback output, which is agree to the simulation serult. Generally, most of digital filters are in higher order form; however, the order of filter can be increased from second order to N time of second order by cascading the group of 2^{nd} order filter as demonstated in Fig.12.

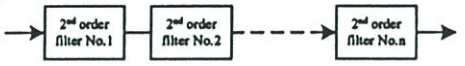


Fig.12 Block diagram of the cascade 2n-order filter

5. Conclusions

In this paper, the method has been introduced of programmable direct form digital filter realization. The lookup table contents of memory were modified for reducing coefficient round off noise, noise feedback was included in the memory. Designing steps and implementation have shown that reduction in noise output and low hardware size could be achieved by such a realization.

ACKNOWLEDGMENT

This work is funded by the Thailand Research Fund (TRF) through the Senior Research Scholar Program, grant number RTA/04/2543. The Support by The Japan International Coorperation Agency (JICA) is also acknowledge.

REFERENCES

[1] Abraham Peled, Bede Liu, "A New Hardware Realization of digital Filters" IEEE Trans. On ASSP, vol. 22, No.6, December 1974

[2] Stanley A. White, "Applications of Distributed Arithmetic to Digital Signal Processing: a tutorial review" IEEE Trans. On ASSP Magazine, July 1989 pp 4-19

[3] Bernie New,"A Distributed Arithmetic approach to designing Scalable DSP Chips" EDN access for design by design, August 17, 1995

[4] Radhika S.Grover, Weijia Shang and Qiang Li, "A Fast Distributed Arithmetic Architecture for FPGAs"

[5] Gregory Ray Goslin, "A Guide to Using Field Programmable Gate Arrays (FPGAs) for Application Specific Digital Signal Processing Performance" V.1.0, 1995 Xilinx,Inc.

[6] W.Surakampontorn, M.I.Sobhy, "Self-error feedback in recursive digital filters" IEE Procedings, Vol.130 Pt.G.No.5, October 1983

[7] Newguard Bruce, "Seminar: Signal Processing with Xilinx FPGAs" Xilinx Publication, June 1996.

[8] A.Djebba Ri, J.M.Rouvaen, L.Camus, "Noise reduction in recursive digital filters using optimal and suboptimal error feedback" INT.J. Electronics,1997, vol.83, NO.6, 743-751

[9] Timo I. Laakso, Iiro O. Hartimo, "Noise Reduction in Recursive Digital Filters Using High-Order Error -Feedback" IEEE Trans. On Signal Processing Vol.40 No.5, May 1992

ประวัติผู้เขียน

ชื่อผู้เขียน	นาย มนูญ สันถวะคุปต์
วันเดือนปี	วันที่ 28 พฤษภาคม 2522
สถานที่เกิด	จังหวัดกรุงเทพฯ
วุฒิการศึกษาระดับปริญญาตรี	วิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมอิเล็กทรอนิกส์
สถาบันที่สำเร็จการศึกษา	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีที่สำเร็จการศึกษา	ปีการศึกษา 2543
งานวิจัยที่มีความสนใจ	การประมวลผลสัญญาณเชิงเลข, วงจรกรองสัญญาณเชิงเลข, การออกแบบวงจรรวมดิจิตอลด้วย FPGA