การขจัดความถี่รบกวนจากสายส่งกำลัง สำหรับการวัดคลื่นไฟฟ้าหัวใจ

POWER-LINE FREQUENCY CANCELLATION TECHNIQUES FOR ELECTROCARDIOGRAPH MEASUREMENT

สุริยา วิทยาประดิษฐ์

SURIYA WITTHAYAPRADIT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าคุณทหารลาดกระบัง

พ.ศ. 2550

การขจัดความถี่รบกวนจากสายส่งกำลัง สำหรับการวัดคลื่นไฟฟ้าหัวใจ

POWER-LINE FREQUENCY CANCELLATION TECHNIQUES FOR ELECTROCARDIOGRAPH MEASUREMENT

สุริยา วิทยาประดิษฐ์

SURIYA WITTHAYAPRADIT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2550

POWER-LINE FREQUENCY CANCELLATION TECHNIQUES FOR ELECTROCARDIOGRAPH MEASUREMENT

SURIYA WITTHAYAPRADIT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN INSTRUMENTATION ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2007

หัวข้อวิทยานิพนธ์	การขจัดความถี่รบกวนจากสายส่งกำลัง สำหรับการวัดคลื่นไฟฟ้าหัวใจ
นักศึกษา	นายสุริยา วิทยาประดิษฐ์
รหัสนักศึกษา	47060654
ปริญญา	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมการวัดคุม
พ.ศ.	2550
อาจารย์ที่ปรึกษาวิทยานิพนธ์	รองศาสตราจารย์ สักรียา ชิตวงศ์

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้นำเสนอ การขจัดความถี่รบกวนจากสายส่งกำลังที่เหนี่ยวนำเข้ามากับสัญญาณคลื่นไฟฟ้าหัวใจ โดยใช้วงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอลชนิดอิมพัลส์ความยาวไม่จำกัด เปรียบเทียบกับวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอกชนิดติดตาม โดยทำการวัดเทียบกับสัญญาณคลื่นไฟฟ้าหัวใจชนิดสองขั้วในหลีด I ซึ่งเป็นรูปสัญญาณไฟฟ้าหัวใจมาตรฐาน พร้อมกับการเปรียบเทียบประสิทธิภาพของวงจรกรองความถี่ทั้งสองแบบ

วงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอลชนิดอิมพัลส์ความยาวไม่จำกัด ใช้ผลตอบสนองชนิดบัตเตอร์เวิอร์ธ อันดับสอง มีค่าคุณภาพเท่ากับ 8 และมีอัตราการลดทอนความถี่รบกวนของความถี่กลางเท่ากับ -35 ดีบี แล้วใช้ความถี่ของการชักตัวอย่าง 488 เฮิรตซ์ มีการแปลงสัญญาณแบบแอนะลอกกับแบบดิจิตอลขนาด 8 บิต ส่วนการประมวลผลแบบดิจิตอล .ใช้ชิพวงจรรวม XC2S100TQ144-5C ใช้ทรัพยากรของวงจรรวมทั้งสิ้น 27 เปอร์เซ็นต์

วงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอกชนิดติดตาม มีโครงสร้างแบบลูฟเปิดโดยใช้โครงข่ายตัวเก็บประจุ มีค่าคุณภาพของวงจรเท่ากับ 8 และมีอัตราการลดทอนความถี่รบกวนของความถี่กลางเท่ากับ -35 ดีบี ขอบเขตความถี่ของการติดตามตั้งแต่ 40 เฮิรตซ์ ถึง 70 เฮิรตซ์ จากการทดสอบการขจัดความถี่รบกวนที่ 50 เฮิรตซ์ , 55 เฮิรตซ์ และ 60 เฮิรตซ์ กับการทดสอบกับการวัดจากร่างกายจริง

Thesis Title	Power-Line Frequency Cancellation Techniques for Electrocardiograph Measurement
Student	Mr.Suriya Witthayapradit
Student ID.	47060654
Degree	Master of Engineering
Program	Instrumentation Engineering
Year	2007
Thesis Advisor	Associate Professor Sakreeya Chitwong

ABSTRACT

This thesis presents the method to cancel noises, induced from power line, in ECG signals. The stop-band IIR (Infinite Impulse Response) filter is used and compared with the adaptive (tracking) filter. For the performance comparison, the standard ECG signal (bipolar type in lead I) is used as a reference signal to compare with output signals from both types of filters.

The second-order Butterworth IIR filter, with 488Hz sampling rate and 8-bit precision , is designed to achieve the quality factor of 8 and -35dB noise attenuation rate. This filter is implemented in XC2S100TQ144-5C and consumes only 27% chip resource.

The adaptive or tracking filter, with open-loop structure and Switched capacitor network , is also designed to achieve the quality factor of 8 and -35dB noise attenuation rate. Tracking frequency can be ranged is from 40Hz to 70 Hz. From the experiments, the noises at 50Hz, 55Hz and 60Hz can be removed from both generated signals and real signals from human body.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ ด้วยความกรุณาจากอาจารย์ที่ปรึกษา รศ. สักรียา ชิตวงศ์ ที่ให้ความช่วยเหลือ ให้คำปรึกษาในหลายๆด้าน และข้อมูลต่างๆในการศึกษาและวิจัย อีกทั้งจัดหาเครื่องมือสำหรับการทดลองในห้องปฏิบัติการ ตลอดจนให้ความรู้และประสบการณ์ที่ดีแก่ข้าพเจ้า

ขอขอบพระคุณ รศ.ดร.ฟูศักดิ์ ชีวสุวิทย์ และ รศ.วิทยา ทิพย์สุวรรณพร กรรมการสอบหัวข้อและโครงร่างวิทยานิพนธ์ ที่ได้กรุณาให้คำแนะนำที่ดี จนกระทั่งวิทยานิพนธ์ฉบับนี้สำเร็จลงได้

ขอขอบพระคุณ ดร.สาธิต อินทจักร ที่ให้กำลังใจในระหว่างการศึกษา และแนะนำแนวทางของงานวิจัยและแหล่งค้นคว้าข้อมูล และตำราในระหว่างการศึกษา

ขอขอบคุณบริษัท MAXIM ที่ได้ให้การสนับสนุนอุปกรณ์ในการทดลอง และขอขอบคุณห้องปฏิบัติการอิเล็กทรอนิกส์ขั้นสูง มหาวิทยาลัยเทคโนโลยีมหานคร ที่ได้ให้การสนับสนุนทางด้านเครื่องมือ

สุดท้ายขอขอบคุณภรรยาข้าพเจ้า คุณดวงกมล วิทยาประดิษฐ์ ที่เป็นคู่คิดและให้กำลังใจที่ดีตลอดมา

สำหรับคุณงามความดีอันใดที่เกิดจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดา มารดา ซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนครู อาจารย์ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีแก่ข้าพเจ้า

นายสุริยา วิทยาประดิษฐ์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

คลื่นไฟฟ้าหัวใจโดยทั่วไปมีขนาดแรงดันเพียง 0.5 – 4 มิลลิโวลต์ และมีองค์ประกอบของความถี่ตั้งแต่ 0.01 – 250 เฮิรตซ์[1] ดังนั้นในกระบวนการวัดสัญญาณคลื่นไฟฟ้าหัวใจ จึงต้องการวงจรอิเล็กทรอนิกส์ที่มีอัตราการขยายสูง อีกทั้งต้องคำนึงถึงมาตรฐานความปลอดภัยของสมาคมโรคหัวใจแห่งประเทศสหรัฐอเมริกา (American Heart Association ; AHA1982) ที่ได้กำหนดมาตรฐานไว้ นอกจากนี้ยังมีสัญญาณรบกวนที่เกิดจากสายส่งกำลัง(Power line) ซึ่งมีค่าความถี่ 50 เฮิรตซ์ หรือ 60 เฮิรตซ์ เหนี่ยวนำ(interference) เข้ามาทางสายวัดสัญญาณกับทางร่างกายของผู้ป่วย[2] รวมกับสัญญาณคลื่นไฟฟ้าหัวใจ

ปัจจุบันนี้ นอกจากเครื่องวัดสัญญาณคลื่นไฟฟ้าหัวใจแล้ว ยังมีเครื่องมือทางการแพทย์หลายชนิด ประสพปัญหาเกี่ยวกับสัญญาณรบกวนที่เกิดจากสายส่งกำลัง เหนี่ยวนำเข้ามาทางสายวัดสัญญาณกับทางร่างกายของผู้ป่วย เมื่อทำการออกแบบวงจรกรองความถี่เพื่อขจัดสัญญาณรบกวนข้างต้น จะต้องออกแบบวงจรกรองแถบความถี่หยุดผ่าน(Band Stop Filter ; BSF) ตามพื้นที่ที่นำไปใช้งานอย่างใดอย่างหนึ่ง อาจทำให้ไม่สะดวกต่อการนำไปใช้งาน และในพื้นที่บางพื้นที่ถ้ามีการใช้ไฟฟ้ากันมาก ความถี่อาจเบี่ยงเบนจากความถี่ข้างต้นออกไป การออกแบบวงจรกรองความถี่หยุดผ่านแบบจำกัดค่าของความถี่อาจมิได้แก้ไขปัญหาดังกล่าวได้ เพื่อแก้ไขปัญหาสัญญาณรบกวนที่มีความถี่เบี่ยงเบน จึงจำเป็นต้องออกแบบวงจรกรองความถี่ ที่สามารถติดตามขจัดความถี่รบกวนเหล่านี้ได้เองโดยอัตโนมัติ

จากงานวิจัยที่ผ่านๆมา ได้มีการวิจัยเกี่ยวกับวงจรกรองแถบความถี่หยุดผ่าน สำหรับการวัดคลื่นไฟฟ้าหัวใจ มีทั้งแบบวงจรกรองความถี่ดิจิตอล(digital filter) หรือวงจรกรองความถี่แอนะลอก (analog filter) พบว่าวงจรกรองความถี่ดิจิตอล ส่วนมากออกแบบโดยใช้วงจรรวมเอฟพีจีเอ(Field Programmable Gate Array ; FPGA) มีทั้งโครงสร้างแบบอิมพัลส์ความยาวจำกัด(Finite Impulse Response ; FIR)[4] กับโครงสร้างแบบอิมพัลส์ความยาวไม่จำกัด(Infinite Impulse Response ; IIR)[5] ซึ่งวงจรกรองความถี่ดิจิตอล มีข้อได้เปรียบที่มีความแม่นยำในการขจัดความถี่รบกวนได้สูงกว่าแบบแอนะลอก แต่ในทางกลับกันก็มีข้อเสียหลายประการ เช่น มีขนาดวงจรที่ใหญ่ ซับซ้อน และ ราคาแพง เป็นต้น

ถึงแม้ว่าในปัจจุบันมีการพัฒนางานวิจัย จนมีการออกแบบให้มีการปรับตัว(Adaptive) ตามค่าของความถี่ตัด[6,7] ตามสัญญาณรบกวน แต่ในความเป็นจริง การออกแบบวงจรกรองดิจิตอลสำหรับการวัดสัญญาณคลื่นไฟฟ้าหัวใจที่ใช้โครงสร้างแบบอิมพัลส์ความยาวจำกัดหรือเอฟไอาร์นั้น ไม่

เหมาะสมต่อการใช้งานจริง เนื่องจากผลตอบสนองทางเฟส(Phase Response) ในช่วงความถี่แถบผ่านเปลี่ยนแปลงเชิงเส้น ดังนั้นการออกแบบวงจรกรองความถี่หยุดผ่านสำหรับการวัดคลื่นไฟฟ้าหัวใจ ควรใช้โครงสร้างแบบอิมพัลส์ความยาวไม่จำกัดหรือไอไออาร์ จึงมีความเหมาะสมมากกว่า เพราะการตอบสนองทางเฟส ในช่วงความถี่แถบผ่านมีค่าคงที่มากกว่า

ในปัจจุบันสำหรับการออกแบบวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก ได้นำเสนอวงจรกรองความถี่แบบติดตาม(tracking filter) หรือวงจรกรองความถี่แบบปรับตัว(adaptive filter) ที่สามารถปรับค่าความถี่ตัดตามสัญญาณรบกวนที่เกิดขึ้น ซึ่งมีการยอมรับคุณสมบัติของวงจรแบบสวิตช์ตัวเก็บประจุ (Switched Capacitor) คือใช้สัญญาณนาฬิกาในการเปิด-ปิดตัวเก็บประจุ โดยส่วนใหญ่เลือกใช้วงจรกรองความถี่ที่มีโครงสร้างวงจรเป็นแบบลูฟปิด (Closed Loop) [10-11] ใช้หลักการแบบป้อนกลับภายในเพื่อเปรียบเทียบกับสัญญาณรบกวน

วงจรกรองแถบความถี่หยุดผ่านชนิดปรับตัวได้ โดยการเปลี่ยนแปลงค่าอัตราการชักตัวอย่าง (Adaptive Sampling Rate Filter ; ASRF)[10] ใช้การนำสัญญาณเอาต์พุตของวงจรกรองความไว (sensitivity filter) คูณกับสัญญาณทางอินพุต ผ่านไปยังวงจรอินทิเกรเตอร์(Integrator) เพื่อปรับเปลี่ยนเป็นค่าแรงดันไฟฟ้ากระแสตรงที่แปรค่าตามสัญญาณรบกวน แล้วผ่านค่าแรงดันไฟฟ้ากระแสตรงดังกล่าวไปยังวงจรกำเนิดความถี่ควบคุมด้วยแรงดัน(Voltage Controlled Oscillator ; VCO) เพื่อใช้ในการควบคุมสวิตช์ตัวเก็บประจุของวงจรกรองแถบความถี่ผ่านกับวงจรกรองความไว ส่วนเอาต์พุตของวงจรกรองแถบความถี่หยุดผ่านได้จากวงจรรวมสัญญาณ(summing) ของสัญญาณอินพุตกับสัญญาณเอาต์พุตของวงจรกรองแถบความถี่ผ่าน

สำหรับวงจรกรองแถบความถี่หยุดผ่านแบบติดตาม[11] ใช้วิธีการตรวจจับลำดับเฟสของความถี่(Sequential-Phase Frequency Detector ; S-PFD) โดยทำการเปรียบเทียบเฟสของสัญญาณอินพุตเทียบกับสัญญาณเอาต์พุตของวงจรกรองแถบความถี่ผ่าน แล้วนำผลลัพธ์ที่ได้คูณกับสัญญาณเอาต์พุตยกกำลังสอง ผ่านไปยังวงจรอินทิเกรเตอร์ เพื่อปรับเปลี่ยนเป็นค่าแรงดันไฟฟ้ากระแสตรงที่แปรค่าตามสัญญาณรบกวน ผ่านไปยังวงจรกำเนิดความถี่ควบคุมด้วยแรงดัน เพื่อใช้ในการควบคุมสวิตช์ตัวเก็บประจุของวงจรกรองแถบความถี่ผ่าน ส่วนเอาต์พุตของวงจรกรองแถบความถี่หยุดผ่านได้จากวงจรรวมสัญญาณ ของสัญญาณอินพุตกับสัญญาณเอาต์พุตของวงจรกรองแถบความถี่ผ่าน

ทั้งสองวิธีการที่ได้กล่าวถึง เป็นการนำเสนอวงจรกรองแถบความถี่หยุดผ่านแบบติดตามหรือปรับตัวตามความถี่รบกวนโดยใช้วิธีลูฟปิด จะเห็นว่าวงจรมีขนาดใหญ่ และมีความซับซ้อนมาก อีกทั้งผลการทดลองของทั้งสองวิธีการนั้น มีประสิทธิภาพในการขจัดสัญญาณรบกวนได้น้อยมาก สังเกตได้จากการเปรียบเทียบผลการทดลองที่ได้นำเสนอในบทที่ 6

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

วิทยานิพนธ์ฉบับนี้มุ่งหมายเพื่อศึกษาการออกแบบวงจรกรองแถบความถี่หยุดผ่าน ซึ่งทำหน้าที่ขจัดความถี่รบกวนจากสายส่งกำลัง(Power-line Frequency) ที่ใช้ในการวัดสัญญาณคลื่นไฟฟ้าหัวใจ โดยคุณสมบัติของวงจรกรองความถี่ต้องมีแถบขจัดที่แคบมาก และมีอัตราการลดทอนความถี่รบกวนได้สูง ในช่วงแถบความถี่ผ่านต้องมีผลตอบสนองที่ราบเรียบ และมีเฟสที่มีค่าคงที่ตลอดช่วงแถบความถี่ผ่าน โดยวิทยานิพนธ์ฉบับนี้ได้นำเสนอวงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอล ชนิดผลตอบสนองอิมพัลส์ความยาวไม่จำกัด เพราะมีความแม่นยำของการขจัดสัญญาณรบกวนและมีผลตอบสนองทางเฟสในช่วงแถบความถี่ผ่านที่คงที่ ส่วนวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก วิทยานิพนธ์ฉบับนี้นำเสนอการใช้โครงข่ายสวิตช์ตัวเก็บประจุ เพราะมีความแม่นยำทางด้านค่าความถี่ตัด อีกทั้งสามารถประยุกต์ให้สามารถติดตามขจัดความถี่รบกวนได้ เพื่อใช้เป็นแนวทางในการพัฒนาวงจรรวมแบบชิพเดี่ยว(single chip) เพื่อทำหน้าที่ตรวจวัดสัญญาณคลื่นไฟฟ้าหัวใจกับการขจัดความถี่รบกวนจากสายส่งกำลัง ออกไปจากสัญญาณคลื่นไฟฟ้าหัวใจ

1.3 สมมติฐานของการศึกษา

การขจัดความถี่รบกวนจากสายส่งกำลังออกไปจากสัญญาณคลื่นไฟฟ้าหัวใจ ที่ใช้วงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอล ส่วนมากนิยมใช้โครงสร้างชนิดผลตอบสนองอิมพัสซ์ความยาวจำกัด เพราะมีเฟสในแถบความถี่ผ่านเปลี่ยนค่าแบบเชิงเส้น แต่ในสัญญาณคลื่นไฟฟ้าหัวใจที่มีองค์ประกอบทางความถี่ตั้งแต่ 0.01 เฮิรตซ์ ถึง 250 เฮิรตซ์ ดังนั้นการใช้โครงสร้างชนิดผลตอบสนองอิมพัสซ์ความยาวจำกัด จึงไม่เหมาะสมกับการขจัดสัญญาณรบกวนในคลื่นไฟฟ้าหัวใจ ส่วนโครงสร้างชนิดผลตอบสนองอิมพัลซ์ความยาวไม่จำกัดที่มีฟังก์ชันการถ่ายโอนเหมือนกันกับวงจรกรองความถี่แบบแอนะลอก จะให้ผลตอบสนองทางเฟสของโครงสร้างชนิดผลตอบสนองอิมพัลซ์ความยาวไม่จำกัด ของวงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอล จะไม่ส่งผลกระทบต่อเฟสของสัญญาณคลื่นไฟฟ้าหัวใจ สำหรับฟังก์ชันการถ่ายโอนแบบบัตเตอร์เวิอร์ธ(Butterworth) ที่มีผลตอบสนองทางขนาดช่วงความถี่แถบผ่านที่ราบเรียบ จึงมีความเหมาะสมอย่างยิ่งในการขจัดสัญญาณรบกวนออกจากสัญญาญคลื่นไฟฟ้าหัวใจ

ส่วนวงจรกรองความถี่แบบแอนะลอก จะขาดความแม่นยำทางด้านความถี่ตัด แต่การประยุกต์ใช้วงจรรวมโครงข่ายสวิตช์ตัวเก็บประจุเป็นวงจรกรองความถี่ จะมีความแม่นยำทางด้านความถี่ตัดไม่ด้อยไปกว่าวงจรกรองความถี่แบบดิจิตอล อีกทั้งการปรับเปลี่ยนค่าความถี่ของสัญญาณนาฬิกา ทำให้ค่าความถี่ตัดของวงจรกรองความถี่สามารถปรับตัวตามไปได้ด้วย ดังนั้นการนำเสนอวงจร

กรองแถบความถี่หยุดผ่านแบบแอนะลอกของวิทยานิพนธ์ฉบับนี้ สามารถติดตามความถี่ที่ต้องการขจัดออกไปจากสัญญาณคลื่นไฟฟ้าหัวใจได้

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

วงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอลชนิดผลตอบสนองอิมพัลส์ความยาวไม่จำกัด ส่วน มากนำเสนอด้วยการจำลองจากโปรแกรม MATLAB เป็นหลัก การนำเสนอของวิทยานิพนธ์ฉบับนี้ใช้การคำนวณหาค่าสัมประสิทธิ์(Coefficient) และการออกแบบวงจรกรองด้วยภาษาวีเอชดีแอล จากนั้นทำการวัดประสิทธิภาพของวงจรกรองความถี่ การออกแบบวิธีการนี้สามารถเอื้อประโยชน์ในการออกแบบวงจรกรองความถี่ดิจิตอลแบบติดตามความถี่ตัดโดยวิธีเปิดตาราง(Look-Up Table) ต่อไปได้

สำหรับวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก การใช้โครงข่ายสวิตช์ออกแบบวงจรกรองความถี่แบบลูฟปิด จะได้โครงสร้างของวงจรอิเล็กทรอนิกส์ที่มีขนาดใหญ่มาก อีกทั้งอัตราการขจัดสัญญาณรบกวนทำได้ไม่ดีพอ การนำเสนอวงจรกรองความถี่แบบลูฟเปิด จะได้วงจรอิเล็กทรอนิกส์ที่มีขนาดเล็กลง และมีอัตราการขจัดสัญญาณรบกวนได้ดี

1.5 การเปรียบเทียบระหว่างวิธีการที่นำเสนอกับวิธีการแบบพื้นฐาน

วงจรกรองความถี่แบบดิจิตอลชนิดผลตอบสนองอิมพัลส์ความยาวไม่จำกัด นำเสนอวิธีทางตรงแบบที่ 1 (direct form 1) หรือวิธีทางตรงแบบที่ 2 (direct form 2) ใช้โปรแกรม MATLAB จำลองการทำงานแต่ไม่ได้สร้างวงจรจริง ซึ่งบางวิธีการอาจกระทำได้ยุ่งยาก สำหรับวิทยานิพนธ์ฉบับนี้ใช้วิธีปรับคุณสมบัติจากค่าสัมประสิทธิ์ ทำให้การปรับเปลี่ยนค่าความถี่ตัด สามารถกำหนดจากค่าสัมประสิทธิ์ได้โดยตรง ไม่ต้องปรับค่าความถี่ของสัญญาณนาฬิกา

ทางด้านวงจรกรองความถี่แบบแอนะลอกที่ใช้โครงข่ายสวิตช์ตัวเก็บประจุ มีข้อได้เปรียบกว่าวงจรกรองความถี่ที่สร้างจากอุปกรณ์พาสซิฟ(passive device) เพราะมีความแม่นยำทางด้านความถี่ตัดมาก อีกทั้งมีการพัฒนาให้สามารถติดตามขจัดความถี่ที่ไม่ต้องการได้ดี แต่มีข้อเสียที่การออกแบบกระทำได้ยุ่งยากมากกว่า

1.6 ขอบเขตการวิจัย

ขอบเขตการวิจัยของวิทยานิพนธ์ฉบับนี้ นำเสนอวงจรกรองแถบความถี่หยุดผ่านสำหรับขจัดความถี่รบกวนจากสายส่งกำลัง ที่สอดแทรกในสัญญาณคลื่นไฟฟ้าหัวใจ โดยใช้วงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอลชนิดผลตอบสนองอิมพัลส์ความยาวไม่จำกัด ใช้ฟังก์ชันการถ่ายโอนแบบบัตเตอร์เวิร์ธ อันดับสอง โดยใช้โปรแกรม MATLAB จำลองและออกแบบวงจรกรองความถี่ พร้อมทั้งคำนวณหาสัมประสิทธิ์ ลำดับต่อมาออกแบบภาษาวีเอชดีแอลที่มีการประมวลผลสัญญาณขนาด 8 บิต เพื่อสังเคราะห์วงจรกรองความถี่ที่ผ่านการออกแบบลงในวงจรรวมเอฟพีจีเอ

ส่วนวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอกใช้การติดตามความถี่ ที่ต้องการขจัดแบบลูฟเปิด มีโครงข่ายสวิตช์ตัวเก็บประจุทำหน้าที่เป็นวงจรกรอง เพราะมีความแม่นยำทางด้านความถี่ตัดแล้วใช้วงจรสังเคราะห์ความถี่แบบเฟสล็อกลูฟ ทำหน้าที่สร้างสัญญาณนาฬิกาให้กับวงจรรวมโครงข่ายสวิตช์ตัวเก็บประจุ

ลำดับสุดท้ายทำการทดสอบประสิทธิภาพของวงจรกรองความถี่ทั้งสองแบบ จากนั้นทำการทดสอบการวัดสัญญาณคลื่นไฟฟ้าหัวใจจากร่างกายจริง พร้อมแสดงค่าผิดพลาดแบบค่าเฉลี่ยกำลังสอง

1.7 ขั้นตอนของการศึกษา

วิทยานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาของการวิจัย ออกเป็น 6 บทด้วยกันโดยแต่ละบทประกอบด้วยองค์ประกอบดังต่อไปนี้

บทที่ 1 กล่าวถึงความเป็นมาและความสำคัญของปัญหาของวิทยานิพนธ์ , ความมุ่งหมายและวัตถุประสงค์ของการศึกษา , สมมติฐานของการศึกษา , ทฤษฎีหรือแนวคิดในการวิจัย , การเปรียบเทียบระหว่างวิธีการที่นำเสนอกับวิธีการแบบพื้นฐาน , ขอบเขตของการวิจัย และขั้นตอนการศึกษา

บทที่ 2 กล่าวถึงวงจรขยายสัญญาณที่ใช้ในการวัดสัญญาณคลื่นไฟฟ้าหัวใจ โดยวิทยานิพนธ์ฉบับนี้ได้ทำการทดลองวัดกับร่างกายจริง อีกทั้งเป็นการนำเสนอแนวทางในการออกแบบวงจรรวมแบบชิพเดี่ยว ดังนั้นวงจรขยายสัญญาณ จำเป็นต้องใช้ตามมาตรฐานของสมาคมโรคหัวใจแห่งประเทศสหรัฐอเมริกา

บทที่ 3 กล่าวถึงวงจรกรองแถบความถี่หยุดผ่านชนิดอิมพัลส์ความยาวไม่จำกัด การจำลองและออกแบบโดยใช้โปรแกรม MATLAB , การออกแบบโดยใช้ภาษาวีเอชดีแอล และการออกแบบโดยใช้วงจรรวมเอฟพีจีเอ

บทที่ 4 กล่าวถึงการออกแบบวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอกที่ใช้การติดตามความถี่ตัด ซึ่งประกอบด้วยวงจรสวิตช์ตัวเก็บประจุแบบขั้วโปรแกรม วงจรเฟสล็อกลูฟ วงจรหารความถี่ กับวงจรตรวจจับความถี่จากสายส่งกำลัง

บทที่ 5 กล่าวถึงผลการจำลองกับผลการทดลองของวงจรกรองแถบความถี่หยุดผ่านทั้งแบบดิจิตอล และวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก รวมถึงการเปรียบเทียบประสิทธิภาพของวงจรกรองความถี่ทั้งสอง

บทที่ 6 กล่าวถึงบทสรุปผลการวิจัย ข้อเสนอแนะ และแนวทางการพัฒนา เพื่อก่อให้เกิดประสิทธิภาพสูงสุด

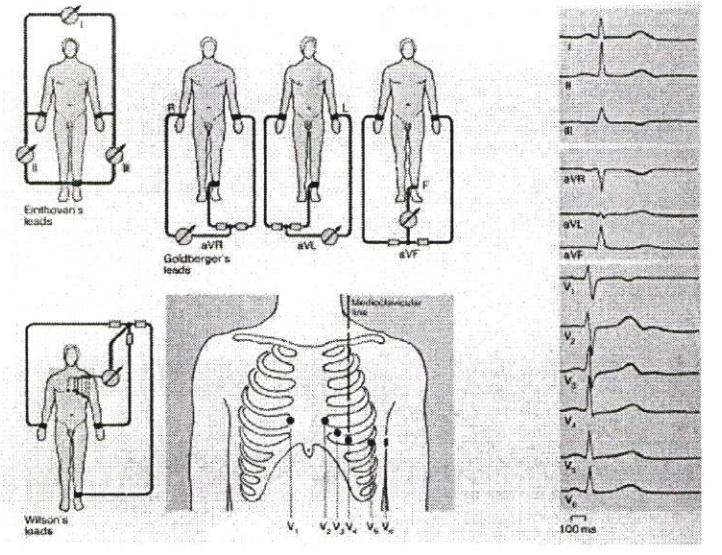
บทที่ 2
วงจรขยายสัญญาณคลื่นไฟฟ้าหัวใจ

ในบทนี้ขอกล่าวถึง ลักษณะของการวัดสัญญาณคลื่นไฟฟ้าหัวใจ กับวงจรขยายสัญญาณสำหรับการวัดสัญญาณคลื่นไฟฟ้าหัวใจ ตามมาตรฐาน AHA1982 ซึ่งวิทยานิพนธ์ฉบับนี้ได้นำเสนอวงจรขยายสัญญาณแบบแยกโดดที่ใช้วงจรรวมชิพเดี่ยว กับวงจรขยายสัญญาณอินสทรูเมนเตชั่นแบบเชื่อมโยงทางแสง ตามรายละเอียดดังต่อไปนี้

2.1 ลักษณะกับมาตรฐานของการวัดสัญญาณคลื่นไฟฟ้าหัวใจ

การวัดสัญญาณคลื่นไฟฟ้าหัวใจ แบ่งออกเป็นสองชนิดด้วยกันคือการวัดสัญญาณไฟฟ้าแบบสองขั้ว(Bipolar lead) กับการวัดสัญญาณแบบขั้วเดี่ยว(Unipolar lead) ในการวัดแบบสองขั้วแบ่งออกเป็นการวัดโดยอาศัยกฎสามเหลี่ยมของไอน์โธเฟน(Einthoven triangle) มีชื่อเรียกว่าไอน์โธเฟน-หลีด(Einthoven Lead) เป็นการวัดความต่างศักย์ของ Lead I , Lead II และ Lead III ตามรูปที่ 2.1โดยที่

Lead I ได้ความต่างศักย์ระหว่างแขนขวากับแขนซ้าย
Lead II ได้ความต่างศักย์ระหว่างแขนขวากับขาซ้าย
Lead III ได้ความต่างศักย์ระหว่างแขนซ้ายกับขาซ้าย



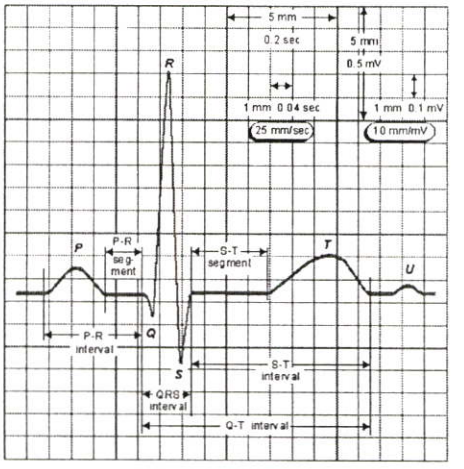
รูปที่ 2.1 ความต่างศักย์ของคลื่นไฟฟ้าหัวใจทั้ง 12 รูปแบบ

การวัดแบบสองขั้วอีกลักษณะ เป็นการประยุกต์จากกฎสามเหลี่ยมไอน์โธเฟน โดยมีความต้านทานเพิ่มขึ้นมาในแต่ละด้านเรียกว่า โครงข่ายของวิลสัน (Wilson Network) เรียกว่าโกลด์เบอร์เกอร์-หลีด(Goldbergers Lead) ทำให้เกิดความต่างศักย์ที่เรียกว่า aVR , aVL และ aVF ดังแสดงในรูปที่ 2.1

ส่วนการวัดแบบขั้วเดี่ยว เรียกว่าวิลสันหลีด(Wilson's Lead) ใช้การตรวจวัดตามตำแหน่งมาตรฐาน 6 จุด คือ $V_1 - V_6$ ดังนั้นการวัดคลื่นไฟฟ้าหัวใจเต็มรูปแบบสามารถวัดความต่างศักย์ได้ถึง 12 สัญญาณดังแสดงในรูปที่ 2.1 แต่ในวิทยานิพนธ์ฉบับนี้ เลือกใช้เพียงสัญญาณคลื่นไฟฟ้าหัวใจของ Lead I ซึ่งเป็นสัญญาณมาตรฐานที่ใช้วัดประสิทธิภาพของผลงานวิจัยทั่วๆไป

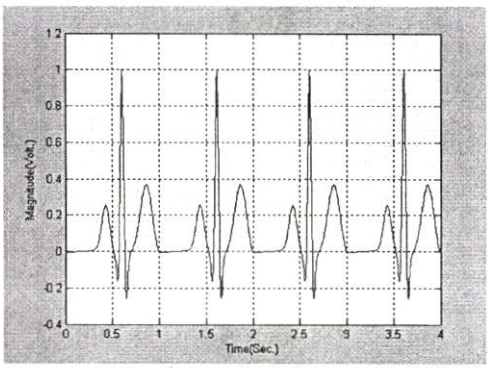
เพื่อให้เกิดความเข้าใจพื้นฐานเกี่ยวกับ ลักษณะและมาตรฐานของสัญญาณคลื่นไฟฟ้าหัวใจตามที่สมาคมโรคหัวใจแห่งประเทศสหรัฐอเมริกา (American Heart Association ; AHA1982) ได้กำหนดมาตรฐานไว้ มีขนาดแรงดันและช่วงเวลาของสัญญาณคลื่นไฟฟ้าหัวใจ ใน Lead I ตามรูปที่ 2.2 ดังนี้

- สัญญาณพี (P wave) เกิดจากการเหนี่ยวนำไฟฟ้าที่เกิดจากการบีบตัวของหัวใจห้องด้านบน มีขนาดแรงดันประมาณ 0.1 – 0.25 มิลลิโวลต์ มีช่วงเวลาประมาณ 80 – 120 มิลลิวินาที
- สัญญาณรวม คิวอาร์เอส (QRS Complex) เกิดจากการเหนี่ยวนำไฟฟ้าที่เกิดจากการบีบตัวของหัวใจห้องด้านล่าง มีขนาดแรงดันประมาณ 0.8 – 1.6 มิลลิโวลต์ มีช่วงเวลาประมาณ 80 – 100 มิลลิวินาที
- สัญญาณที (T wave) เกิดจากการเหนี่ยวนำไฟฟ้าที่เกิดจากการคลายตัวของหัวใจห้องด้านล่าง มีขนาดแรงดันประมาณ 0.2 – 0.5 มิลลิโวลต์ มีช่วงเวลาประมาณ 50 – 180 มิลลิวินาที

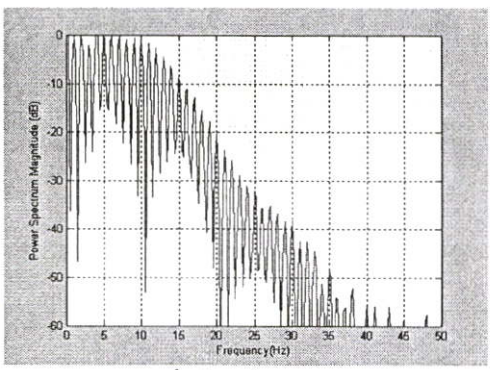


รูปที่ 2.2 มาตรฐานของสัญญาณไฟฟ้าใน Lead I

สำหรับสัญญาณคลื่นไฟฟ้าหัวใจที่ใช้ในการทดสอบประสิทธิภาพของวงจรขยายสัญญาณกับวงจรกรองแถบความถี่หยุดผ่านได้มาจากสมการทางคณิตศาสตร์[3] แสดงในโดเมนเวลากับโดเมนความถี่ ดังรูปที่ 2.3(ก) กับรูปที่ 2.3(ข) ตามลำดับ โดยวิทยานิพนธ์ใช้คลื่นไฟฟ้าหัวใจต้นแบบนี้รวมเข้ากับความถี่รบกวน แล้วทำการออกแบบวงจรกรองความถี่เพื่อขจัดความถี่รบกวนที่สอดแทรกมานี้ ให้ออกไปจากสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบ พร้อมกับแสดงถึงค่าผิดพลาดแบบกำลังสองเฉลี่ยและอัตราส่วนของสัญญาณต่อสัญญาณรบกวน เมื่อเปรียบเทียบสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบกับสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการกรองความถี่แล้ว ต้องการให้ได้ลักษณะของสัญญาณคลื่นไฟฟ้าหัวใจที่คล้ายคลึงกับต้นแบบมากที่สุดเช่นเดียวกันกับงานวิจัยอื่นๆที่เกี่ยวข้อง โดยวิทยานิพนธ์นี้ไม่ขอกล่าวถึงสัญญาณคลื่นไฟฟ้าหัวใจทางคลินิก(Clinic) หรือสรีรทางไฟฟ้า(Electrophysiology)



(ก) คลื่นไฟฟ้าหัวใจในโดเมนเวลา



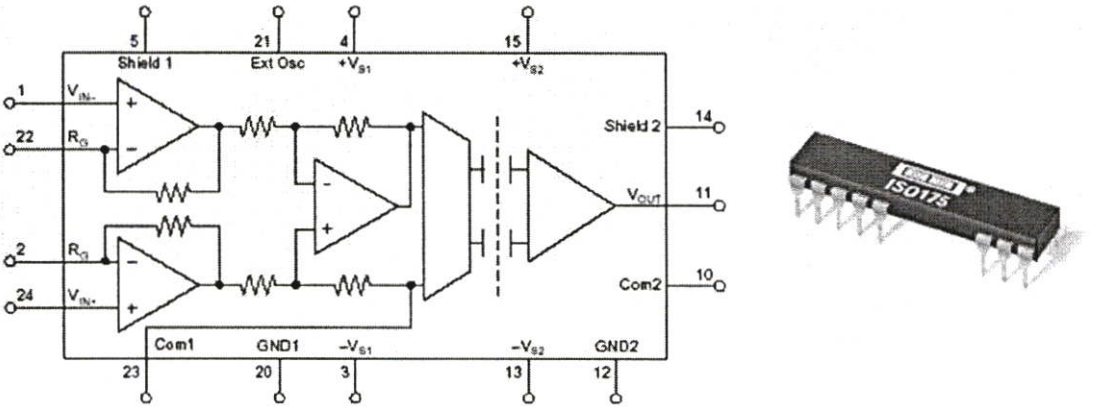
(ข) คลื่นไฟฟ้าหัวใจในโดเมนเวลา

รูปที่ 2.3 โดเมนเวลากับความถี่ของสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบ[3]

จากโดเมนความถี่ของสัญญาณคลื่นไฟฟ้าหัวใจในรูปที่ 2.3(ข) จะเห็นว่ามีองค์ประกอบของความถี่หลายๆความถี่ เพราะฉนั้นสัญญาณอินพุตที่ใช้ในการทดสอบประสิทธิภาพของวงจรกรองแถบความถี่หยุดผ่านสำหรับการวัดคลื่นไฟฟ้าหัวใจนั้น ควรใช้สัญญาณคลื่นไฟฟ้าหัวใจมาตรฐานเท่านั้น การใช้สัญญาณไฟฟ้าอื่นๆอาจแสดงสมรรถนะได้ไม่สมบูรณ์

2.2 วงจรขยายสัญญาณอินสทรูเมนเทชันแบบแยกโดด

การวัดสัญญาณคลื่นไฟฟ้าหัวใจแบบสองขั้ว นิยมใช้วงจรขยายสัญญาณอินสทรูเมนเตชันมากกว่าวงจรขยายผลต่างทั่วๆไป เพราะมีอัตราขจัดแบบโหมดร่วมสูงกว่า กับค่าความต้านทานทางเอาต์พุตมีค่าสูงมาก และสามารถปรับค่าอัตราขยายสัญญาณได้ง่ายกว่า อย่างไรก็ตามการใช้วงจรขยายสัญญาณกับการวัดคลื่นไฟฟ้าหัวใจ ต้องคำนึงถึงมาตรฐานตามสมาคมโรคหัวใจแห่งประเทศสหรัฐอเมริกา ที่ต้องแยกโดด(Isolated) สัญญาณระหว่างภาคอินพุตกับภาคเอาต์พุต ในอดีตจะมีวงจรรวมแบบชิพเดี่ยวที่มีโครงสร้างแบบแยกโดด เช่น ISO175 ของบริษัท BURR-BROWN ตามแผนผัง (Block diagram) และกายภาพดังรูปที่ 2.4 ที่มีภาคอินพุตเป็นวงจรขยายอินทรูเมนเตชั่น



รูปที่ 2.4 แผนผังและกายภาพของวงจรขยายสัญญาณแบบแยกโดดของวงจรรวมแบบชิพเดี่ยว

แหล่งจ่ายกำลังไฟฟ้ากระแสตรงของวงจรขยายสัญญาณแบบแยกโดด จะต้องใช้แหล่งจ่ายกำลังแบบทวิสองวงจรด้วยกัน สำหรับแหล่งจ่ายชุดแรกมีขนาดแรงดัน ± 5 โวลต์ เทียบกับขั้ว GND_0 โดยที่แรงดันด้านบวกขนาด +5 โวลต์ เชื่อมต่อที่ขั้ว $+V_{S1}$ ส่วนแรงดันด้านลบขนาด -5 โวลต์ เชื่อมต่อที่ขั้ว $-V_{S1}$ สำหรับแหล่งจ่ายชุดที่สองมีขนาดแรงดัน ±9 โวลต์ เทียบกับขั้ว GND_1 โดยที่แรงดันด้านบวกขนาด +9 โวลต์ เชื่อมต่อที่ขั้ว $+V_{S2}$ ส่วนแรงดันด้านลบขนาด -9 โวลต์ เชื่อมต่อที่ขั้ว $-V_{S2}$ ดังแสดงในรูปที่ 2.5

ส่วนอัตราขยายแรงดันของวงจร วิทยานิพนธ์ฉบับนี้กำหนดอัตราขยายแรงดันมีค่า 1000 เท่า โดยการกำหนดค่าอัตราขยาย ได้จากความต้านทาน R_G ซึ่งมีค่าอัตราขยายแรงดันเท่ากับ

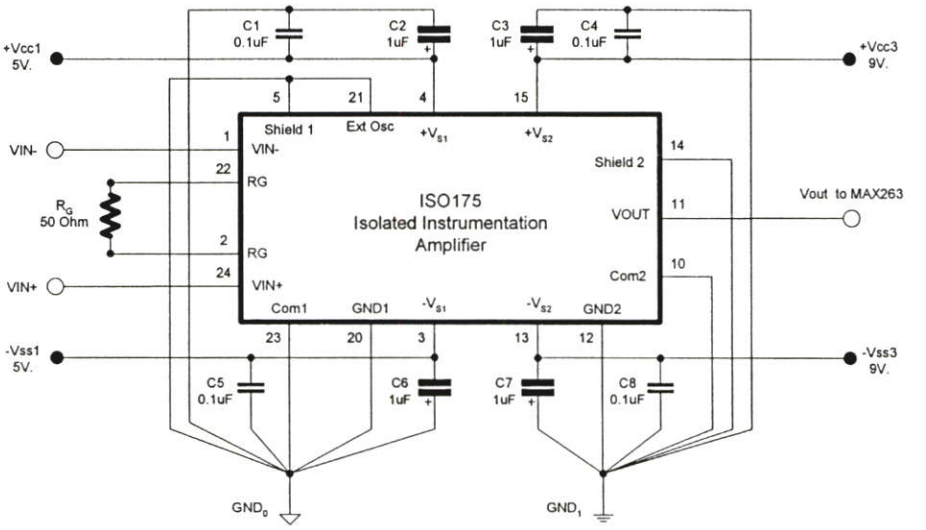
$$A_V = \frac{R_{INT}}{R_G} \qquad (2.1)$$

โดยที่

A_V หมายถึง อัตราขยายแรงดัน

R_{INT} หมายถึง ความต้านทานภายในของวงจรขยายสัญญาณ มีค่าเท่ากับ 50 กิโลโอห์ม

R_G หมายถึง ความต้านทานค่าคงที่ ที่ใช้กำหนดค่าอัตราขยายแรงดัน



รูปที่ 2.5 การเชื่อมต่อแรงดันแบบทวิให้กับวงจรขยายสัญญาณแบบแยกโดด

2.3 วงจรขยายสัญญาณอินสทรูเมนเทชันแบบเชื่อมโยงทางแสง

วงจรขยายสัญญาณชนิดแยกโดดแบบชิพเดี่ยว ในปัจจุบันมีการผลิตออกสู่ตลาดน้อยมาก ดังนั้นวิทยานิพนธ์ฉบับนี้ จึงได้นำเสนอวงจรขยายสัญญาณชนิดแยกโดดแบบเชื่อมโยงทางแสง(Opto Isolation) โดยใช้โครงสร้างวงจรขยายสัญญาณแบบอินสทรูเมนเทชัน ที่มีวงจรขยายส่วนหน้าแบบสมดุลย์(Balance Amplifier) แยกส่วนกับวงจรขยายแบบผลต่าง(Differential Amplifier) ผ่านทางวงจรทรานซิสเตอร์แสง(Opto-Transistor) ตามรายละเอียดดังนี้

2.3.1 วงจรขยายส่วนหน้าแบบสมดุลย์

วงจรขยายส่วนหน้าแบบสมดุลย์ มีลักษณะวงจรดังรูปที่ 2.6 จะมีขั้วอินพุตสองขั้วคือ V_{IN1} กับ V_{IN2} ส่วนทางเอาต์พุตจะมีสองขั้วเช่นกันคือ V_{O1} กับ V_{O2} เทียบกับกราวด์ของวงจรขยายสัญญาณ จากวงจรถ้ากำหนดให้ A คืออออปแอมป์ในอุดมคติ ดังนั้นอัตราขยายแรงดันของวงจรขยายส่วนหน้าแบบสมดุลย์ จะมีค่าเท่ากับ

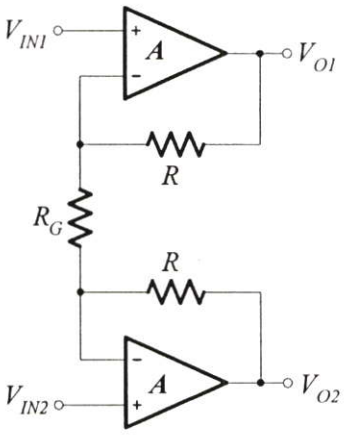
$$A_{V1} = 1+\frac{2R}{R_G} \tag{2.2}$$

โดยที่

A_{V1} หมายถึง อัตราขยายแรงดันของวงจรขยายส่วนหน้าแบบสมดุลย์

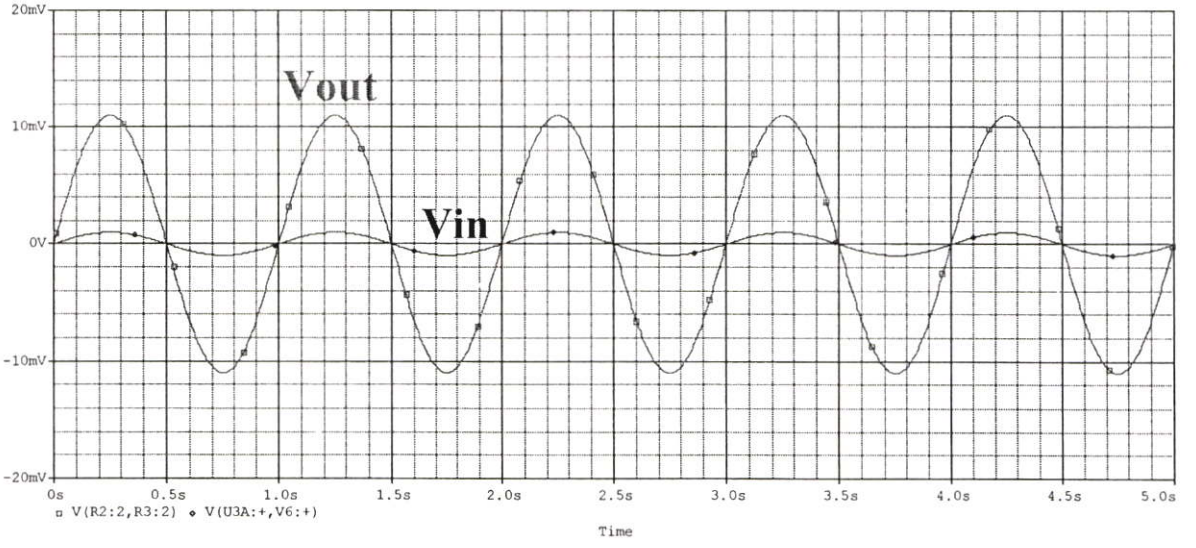
R หมายถึง ความต้านทานค่าคงที่มีหน่วยเป็นโอห์ม

R_G หมายถึง ความต้านทานค่าคงที่ ที่ใช้กำหนดค่าอัตราขยายแรงดัน



รูปที่ 2.6 วงจรขยายส่วนหน้าแบบสมดุลย์

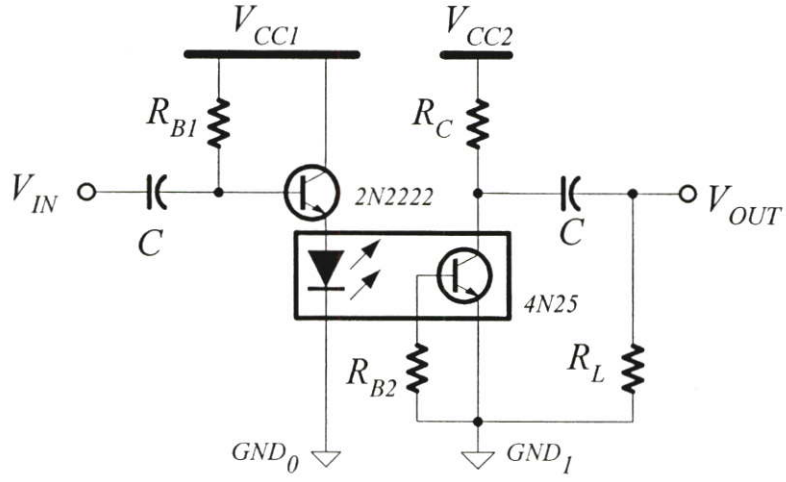
จากการออกแบบวงจรขยายส่วนหน้าแบบสมดุลย์ ของวิทยานิพนธ์ฉบับนี้กำหนดให้มีอัตราขยายแรงดันเท่ากับ 11 เท่าของแรงดันอินพุต เพราะต้องการให้สัญญาณที่ผ่านการขยายแล้วมีระดับสัญญาณอยู่ในช่วง สัญญาณระดับต่ำ (Small Signal) เมื่อกำหนดให้ความต้านทาน(R) มีค่าเท่า 5 กิโลโอห์ม และความต้านทานค่าคงที่ ที่ใช้กำหนดค่าอัตราขยายแรงดันมีค่าเท่ากับ 1 กิโลโอห์ม เมื่อกำหนดแรงดันอินพุต(Vin) ขนาด 2 โวลต์ให้กับขั้ว V_{IN1} เทียบกับ V_{IN2} แล้วทำการวัดแรงดันเอาต์พุต(Vout) ที่ขั้ว V_{O1} เทียบกับขั้ว V_{O2} แล้วทำการจำลอง(Simulation)การทำงานของวงจร โดยใช้โปรแกรม OrCAD จะได้ลักษณะของแรงดันทางขั้วอินพุตเท่ากับ 2 โวลต์ และมีแรงดันทางด้านขั้วเอาต์พุตเท่ากับ 22 โวลต์ ดังรูปที่ 2.7



รูปที่ 2.7 ผลการจำลองการทำงานของวงจรขยายส่วนหน้าแบบสมดุลย์

2.3.2 วงจรขยายสัญญาณโดยทรานซิสเตอร์แสง(Opto-Transistor Amplifier)

โดยปกติทรานซิสเตอร์แสง มักนิยมใช้ในการสวิตช์อิเล็กทรอนิกส์(Electronic Switch) แต่ก็มีบางวิธีการที่นำมาใช้ในการเชื่อมโยงสัญญาณ[18] หรือการขยายสัญญาณ[19] ที่สัญญาณอินพุตระดับต่ำ จากรูปที่ 2.8 ซึ่งเป็นวงจรขยายสัญญาณโดยทรานซิสเตอร์แสง สามารถกำหนดอัตราขยายสัญญาณได้จาก



รูปที่ 2.8 วงจรขยายสัญญาณโดยทรานซิสเตอร์แสง

$$A_{V2} = -\frac{R_C // R_L}{r_E} \tag{2.3}$$

$$r_E = \frac{V_T}{I_E} \tag{2.4}$$

โดยที่

A_{V2} หมายถึง อัตราขยายแรงดันของวงจรขยายโดยทรานซิสเตอร์แสง

R_C หมายถึง ความต้านทานที่ขั้วคอลเลกเตอร์กับแหล่งจ่ายกำลัง

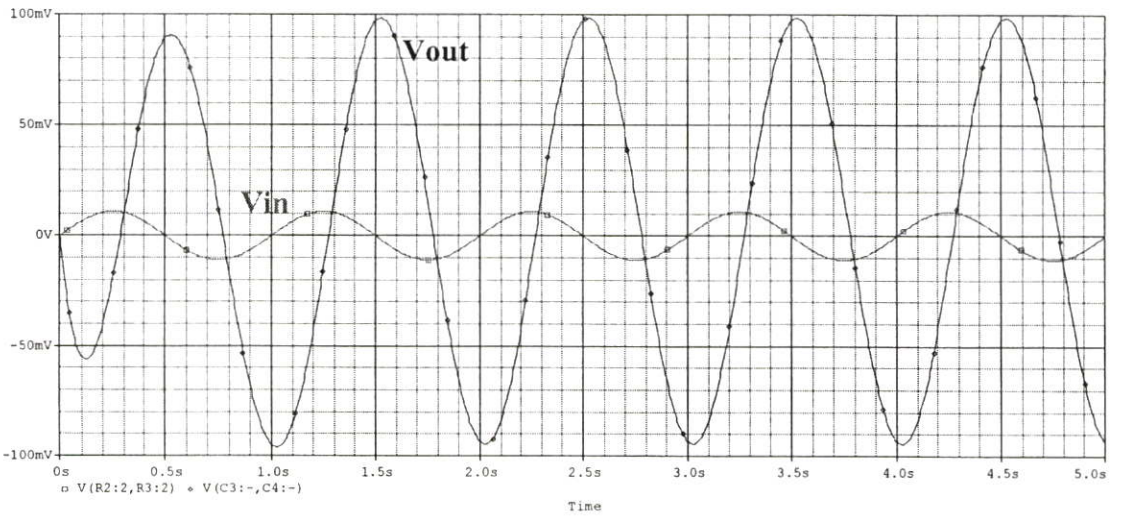
R_L หมายถึง ความต้านทานขนานทางขั้วเอาต์พุต

r_E หมายถึง ความต้านทานแฝงของขั้วอิมิตเตอร์

V_T หมายถึง แรงดันแฝงทางอุณหภูมิ มีค่า 26 มิลิโวล์ต ณ อุณหภูมิ 25 องศาเซนเซียล

I_E หมายถึง กระแสอิมิตเตอร์ มีหน่วยเป็นมิลลิแอมป์

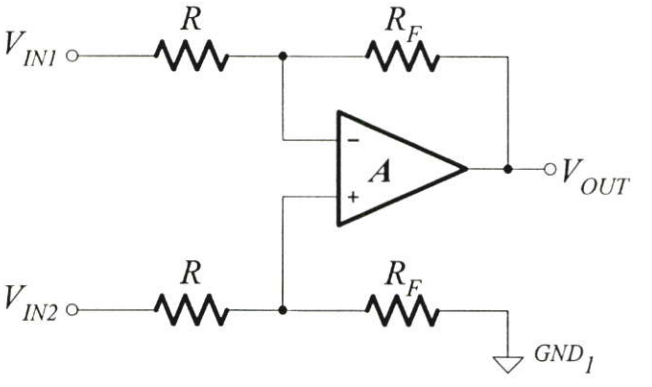
จากการออกแบบวงจรขยายสัญญาณโดยทรานซิสเตอร์แสง ของวิทยานิพนธ์ฉบับนี้ กำหนดให้มีอัตราขยายแรงดันเท่ากับ 10 เท่าของแรงดันอินพุต แล้วทำการจำลองการทำงานของวงจร โดยใช้โปรแกรม OrCAD เมื่อให้แรงดันอินพุต(Vin) มีขนาด 20 มิลลิโวลต์ จะได้สัญญาณทางด้านเอาต์พุต(Vout) ประมาณ 200 มิลลิโวลต์ ดังรูปที่ 2.9



รูปที่ 2.9 ผลการจำลองการทำงานของวงจรขยายสัญญาณโดยทรานซิสเตอร์แสง

2.3.3 วงจรขยายสัญญาณผลต่าง

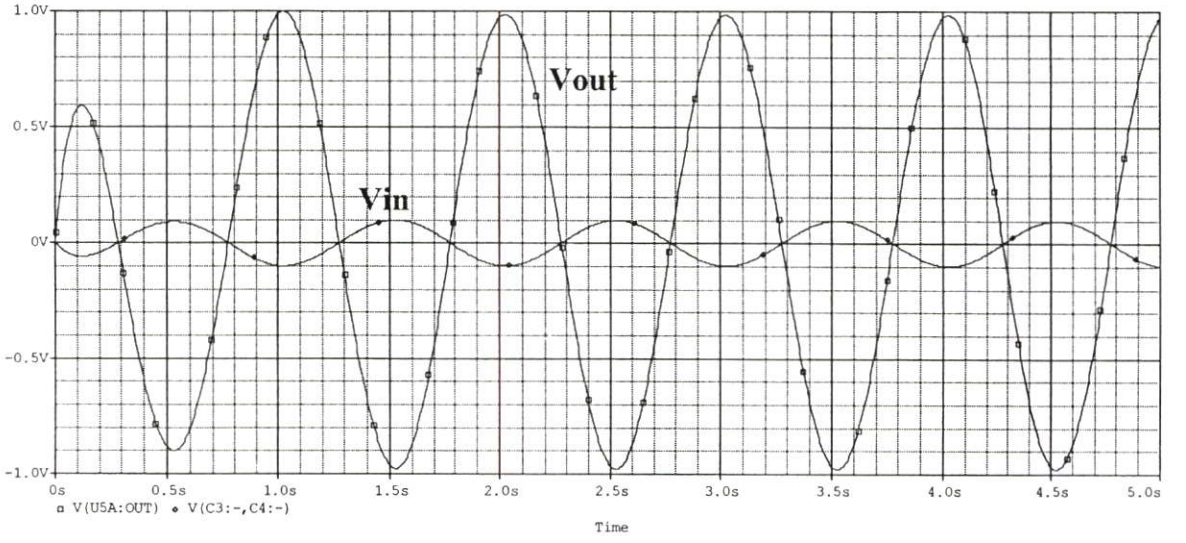
วงจรขยายสัญญาณผลต่าง ซึ่งเป็นวงจรขยายสัญญาณภาคสุดท้าย มีลักษณะวงจรดังในรูปที่ 2.10 จะเห็นว่ามีขั้วอินพุตสองขั้ว และมีขั้วเอาต์พุตเพียงขั้วเดียว วิทยานิพนธ์ฉบับนี้ได้กำหนดให้มีอัตราขยายสัญญาณเท่ากับ 10 จากวงจรถ้ากำหนดให้ A คือออปแอมป์ในอุดมคติ ดังนั้นอัตราขยายแรงดันของวงจรขยายสัญญาณผลต่าง จะมีค่าเท่ากับ



รูปที่ 2.10 วงจรขยายสัญญาณผลต่าง

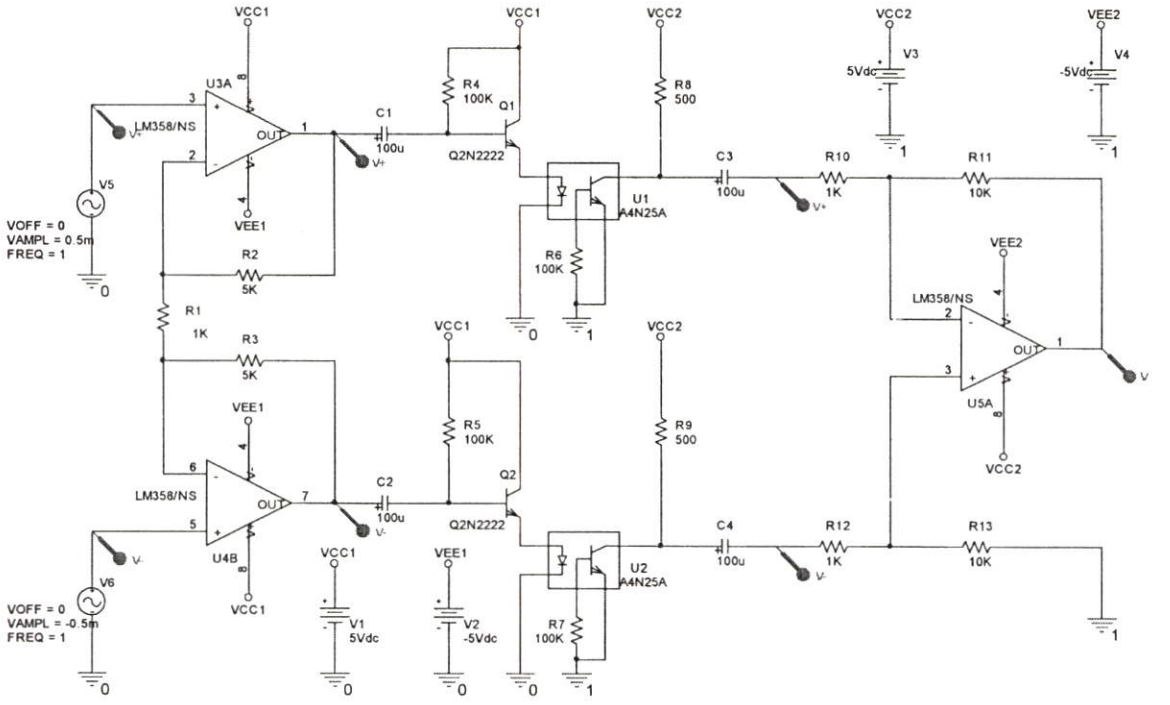
$$A_{V2} = -\frac{V_{OUT}}{(V_{IN1} - V_{IN2})} = -\frac{R_F}{R} \quad (2.5)$$

จากการออกแบบวงจรขยายสัญญาณผลต่าง ของวิทยานิพนธ์ฉบับนี้กำหนดให้มีอัตราขยายแรงดันเท่ากับ 10 เท่าของแรงดันอินพุต เมื่อกำหนดให้ความต้านทาน(R_F) มีค่าเท่า 10 กิโลโอห์ม และความต้านทานค่าคงที่(R) ที่ใช้กำหนดค่าอัตราขยายแรงดัน มีค่าเท่ากับ 1 กิโลโอห์ม แล้วทำการจำลองการทำงานของวงจร โดยใช้โปรแกม OrCAD เมื่อให้แรงดันอินพุต(Vin) มีขนาด 200 มิลลิโวลต์ จะได้สัญญาณทางด้านเอาต์พุต(Vout) ประมาณ 2 โวลต์ ดังรูปที่ 2.11

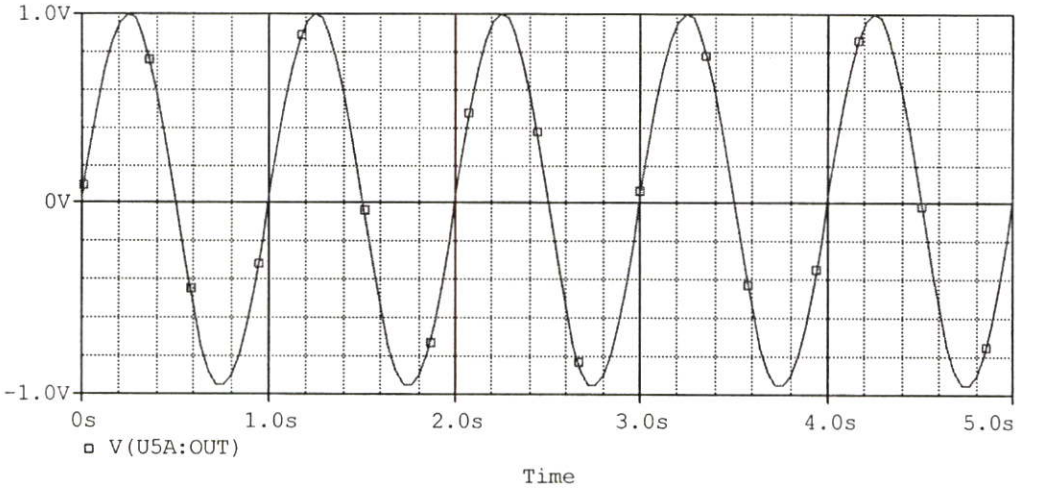


รูปที่ 2.11 ผลการจำลองการทำงานของวงจรขยายสัญญาณผลต่าง

จากการออกแบบวงจรขยายสัญญาณอินสทรูเมนเตชันแบบเชื่อมโยงทางแสง ที่มีอัตราขยายสัญญาณ 1000 เท่า เมื่อทำการจำลองวงจรทั้งหมดร่วมกัน โดยใช้โปรแกรม OrCAD ตามรูปที่ 2.12 แล้วกำหนดสัญญาณอินพุต V5 เท่ากับ 0.5 mV_P กับสัญญาณอินพุต V6 เท่ากับ -0.5 mV_P ที่ความถี่ 1 เฮิรตซ์ทางด้านแหล่งจ่ายไฟฟ้ากระแสตรงของวงจรแบบเชื่อมโยงทางแสงซึ่งมีสองแหล่งจ่าย กำหนดให้แหล่งจ่าย V1 กับ V2 มีจุดกราวด์ร่วมคือหมายเลข 0 หรือเท่ากับ GND_0 ส่วนแหล่งจ่าย V3 กับ V4 มีจุดกราวด์ร่วมคือหมายเลข 1 หรือเท่ากับ GND_1 เมื่อทำการวัดสัญญาณทางด้านเอาต์พุต จะได้ลักษณะของสัญญาณดังแสดงในรูปที่ 2.13

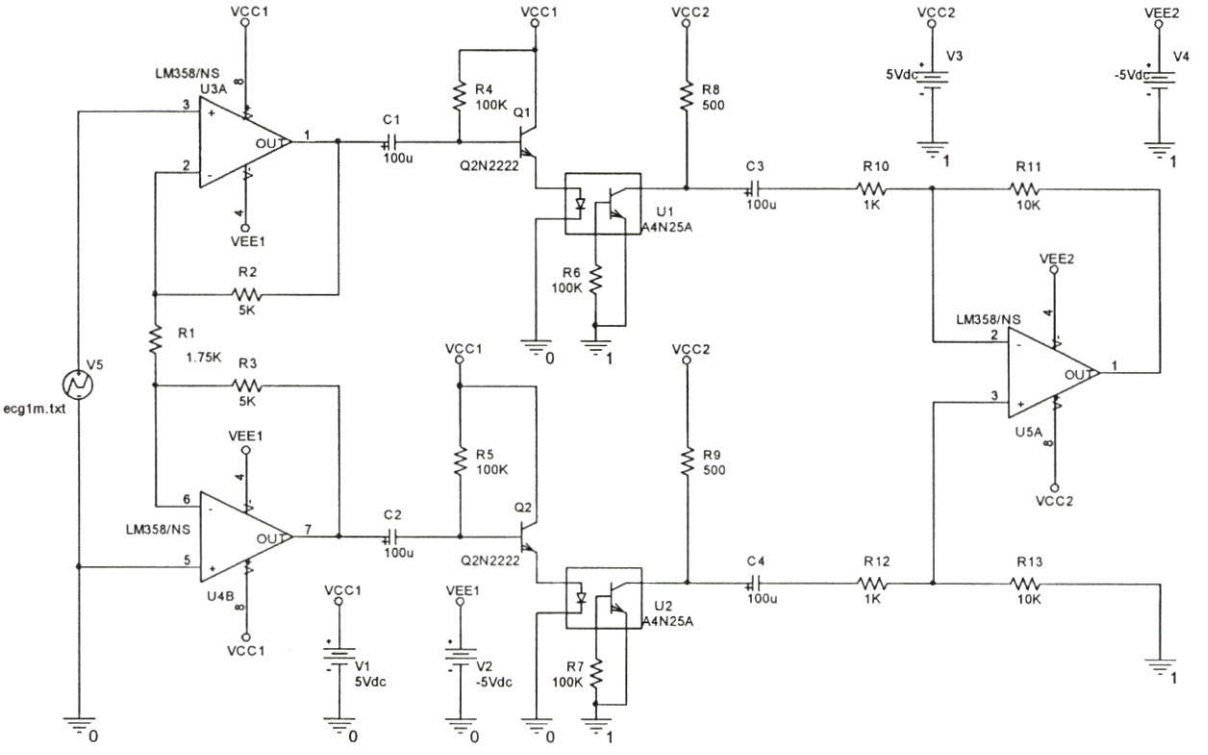


รูปที่ 2.12 วงจรขยายสัญญาณอินสทรูเมนเตชันแบบเชื่อมโยงทางแสง

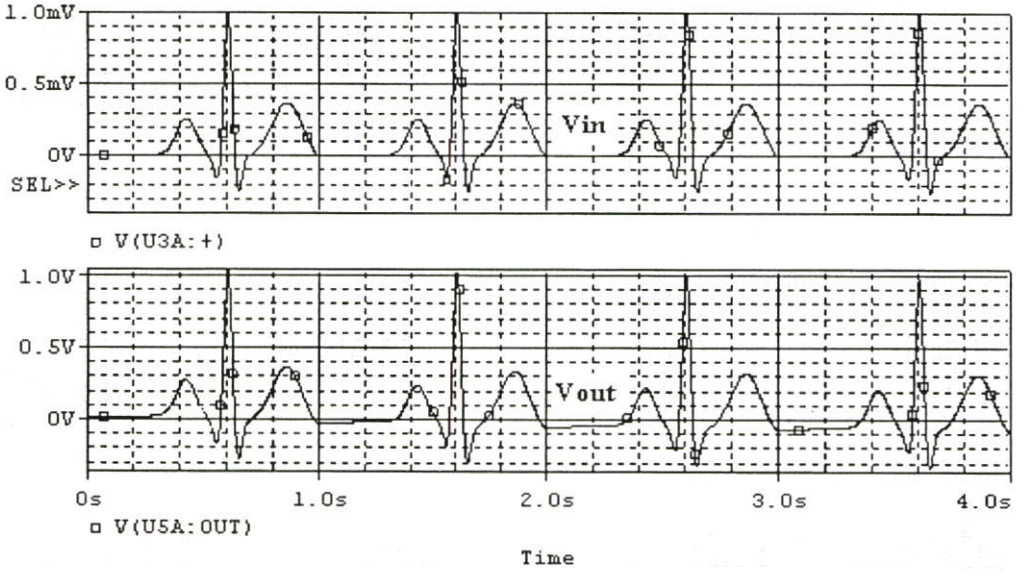


รูปที่ 2.13 ผลการจำลองการทำงานของวงจรขยายสัญญาณอินสทรูเมนเตชันแบบเชื่อมโยงทางแสง

เมื่อใช้สัญญาณคลื่นไฟฟ้าหัวใจต้นแบบ[3] ซึ่งมีขนาดแรงดัน 1 มิลลิโวลต์และคาบเวลาเท่ากับ 1 วินาที ป้อนให้กับขั้วอินพุตของวงจรขยายสัญญาณดังรูปที่ 2.14 แล้วทำการวัดสัญญาณทางขั้วอินพุตเปรียบเทียบกับสัญญาณทางขั้วเอาต์พุตของวงจรขยายสัญญาณ จะได้ลักษณะของสัญญาณดังในรูปที่ 2.15



รูปที่ 2.14 วงจรขยายสัญญาณที่มีอินพุตเป็นคลื่นไฟฟ้าหัวใจต้นแบบ[3]

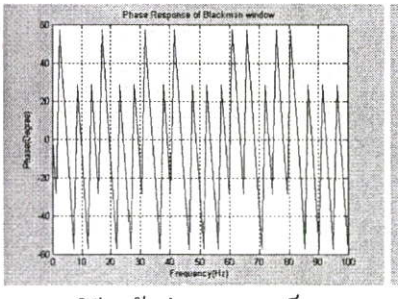


รูปที่ 2.15 ผลการจำลองวงจรขยายสัญญาณคลื่นไฟฟ้าหัวใจ

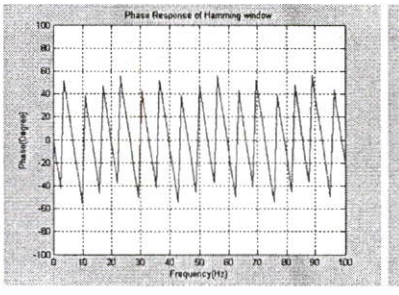
บทที่ 3
วงจรกรองความถี่ดิจิตอลชนิดอิมพัลส์ความยาวไม่จำกัด

ในบทนี้กล่าวถึงวงจรกรองความถี่ดิจิตอล ที่เหมาะสมกับการวัดสัญญาณคลื่นไฟฟ้าหัวใจ โดยพิจารณาที่ผลตอบสนองทางเฟส และผลตอบสนองทางขนาดในช่วงแถบความถี่ผ่าน ต่อมาทำการคำนวณหาสัมประสิทธิ์ แล้วทำการออกแบบวงจรกรองแถบความถี่หยุดผ่านดิจิตอลโดยใช้ภาษาวีเอชดีแอล ลำดับสุดท้ายทำการโปรแกรมลงในวงจรรวมเอฟพีจีเอ

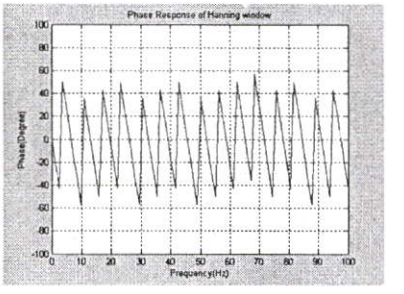
จากการศึกษาเกี่ยวกับวงจรกรองความถี่ดิจิตอล พบว่าวงจรกรองความถี่แบบเอฟไอาร์(FIR) จะให้ผลตอบสนองทางเฟสแบบเชิงเส้นโดยสมบูรณ์ตลอดช่วงความถี่แถบผ่าน จากการจำลองด้วยโปรแกรม MATLAB ของวงจรกรองความถี่แบบเอฟไอาร์จากวิธีหน้าต่าง(Window Method) ของแบล็กแมน(Blackman) , แฮมมิง(Hamming) , ฮานนิง(Hanning) , ไคเซอร์(Kaiser) และตุกคีย์(Tukey) ที่ความถี่กลาง 50 เฮิรตซ์ ค่าคุณภาพเท่ากับ 8 และมีอัตราการลดทอนของแถบหยุดผ่าน 35 dB เมื่อทำการวาดผลตอบสนองทางเฟส จะได้ผลลัพธ์ดังรูปที่ 3.1(ก) จนถึงรูปที่ 3.1(จ) ตามลำดับ จะเห็นว่าผลตอบสนองทางมุมเฟสจะเปลี่ยนแปลงแบบเชิงเส้นโดยสมบูรณ์ ซึ่งอาจไม่เหมาะสมต่อการวัดสัญญาณคลื่นไฟฟ้าหัวใจ



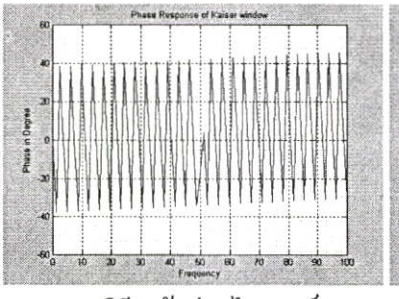
(ก) วิธีหน้าต่างของแบล็กแมน



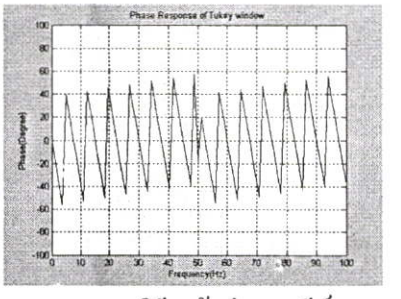
(ข) วิธีหน้าต่างแฮมมิง



(ค) วิธีหน้าต่างฮานนิง



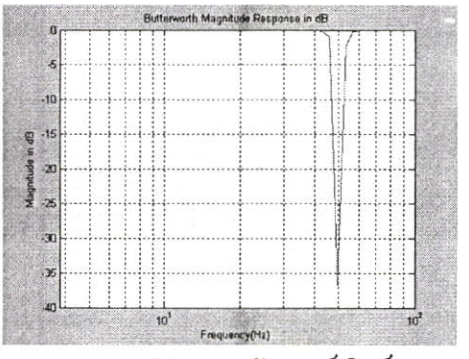
(ง) วิธีหน้าต่างไคเซอร์



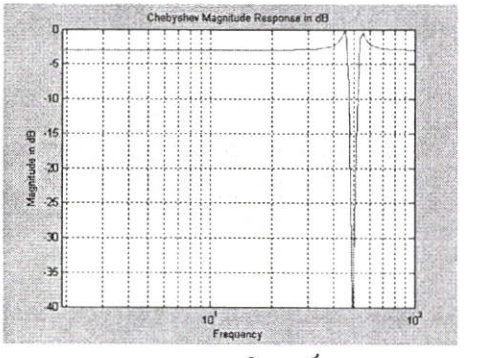
(จ) วิธีหน้าต่างตุกคีย์

รูปที่ 3.1 ผลตอบสนองทางเฟสของวงจรกรองความถี่แบบเอฟไอาร์จากวิธีหน้าต่างแบบต่างๆ

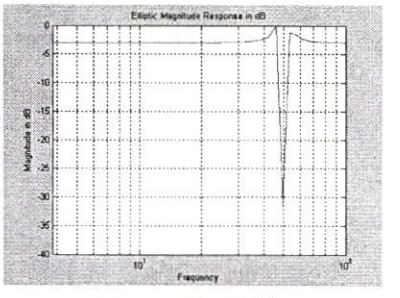
เมื่อพิจารณาวงจรกรองความถี่แบบไอไออาร์(IIR) พบว่าผลตอบสนองทางเฟสในช่วงแถบความถี่ผ่านมีค่าคงที่ อย่างไรก็ตามการออกแบบวงจรกรองความถี่สำหรับการวัดสัญญาณคลื่น ไฟฟ้าหัวใจต้องคำนึงถึงผลตอบสนองทางขนาดของวงจรกรองความถี่ในแต่ละแบบ จากรูปที่ 3.2(ก) จนถึงรูปที่ 3.2(ค) เป็นผลตอบสนองทางขนาดของวงจรกรองแถบความถี่หยุดผ่านแบบบัตเตอร์เวิร์ธ , แบบเชบิเชบว์(Cheby- shev) และแบบอิลลิปติก(Elliptic) ตามลำดับ จะเห็นได้ว่าวงจรกรองความถี่แบบบัตเตอร์เวิร์ธ ให้ผลตอบสนองทางขนาดมีค่าคงที่ตลอดช่วงแถบความถี่ผ่าน ดังนั้นวิทยานิพนธ์ฉบับนี้ จึงเลือกการออกแบบวงจรกรองแถบความถี่หยุดผ่านดิจิตอลแบบบัตเตอร์เวิร์ธ สำหรับการวัดสัญญาณคลื่น ไฟฟ้าหัวใจ



(ก) แบบบัตเตอร์เวิร์ธ

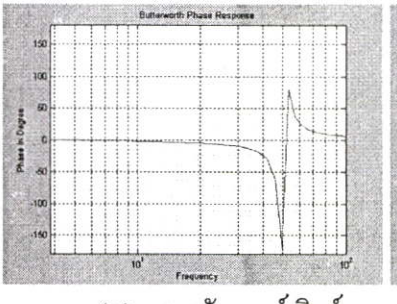


(ข) แบบเชบิเชบว์

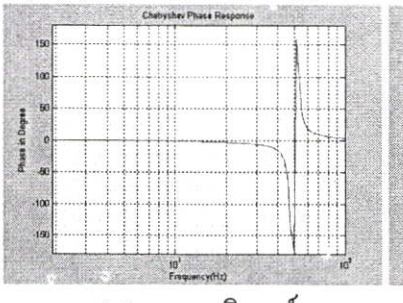


(ค) แบบอิลลิปติก

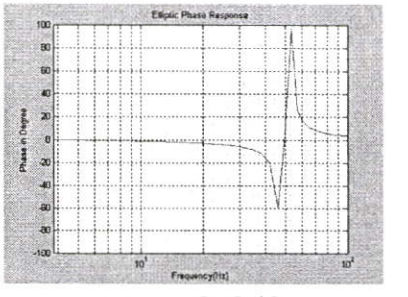
รูปที่ 3.2 ผลตอบสนองทางขนาดของวงจรกรองความถี่แบบไอไออาร์



(ก) แบบบัตเตอร์เวิร์ธ



(ข) แบบเชบิเชบว์

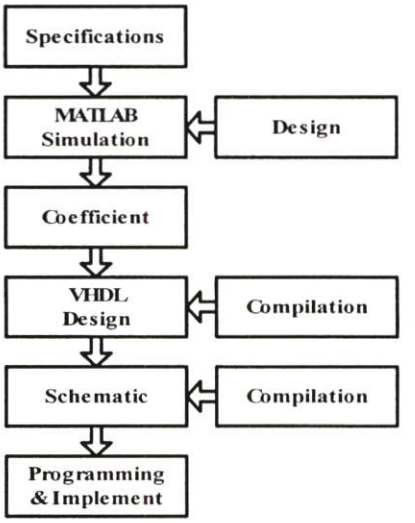


(ค) แบบอิลลิปติก

รูปที่ 3.3 ผลตอบสนองทางเฟสของวงจรกรองความถี่แบบไอไออาร์

3.1 โครงสร้างการออกแบบวงจรกรองความถี่ดิจิตอล

เมื่อพิจารณาถึงความเหมาะสมของวงจรกรองความถี่ดิจิตอลแล้ว ต่อมาทำการสร้างวงจรตามโครงสร้างหรือแผนผังดังแสดงรูปที่ 3.4



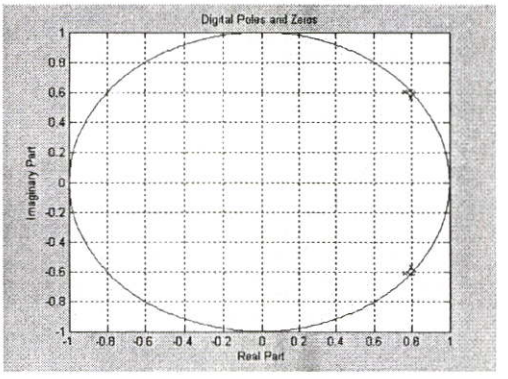
รูปที่ 3.4 โครงสร้างของการออกแบบวงจรกรองความถี่แบบไอไออาร์

จากแผนผังในรูปที่ 3.4 หลังจากได้กำหนดคุณสมบัติของวงจรกรองแถบความถี่หยุดผ่านแล้ว ต่อมาทำการจำลองโดยการเขียนฟังก์ชันด้วยโปรแกรม MATLAB เพื่อคำนวณหาค่าสัมประสิทธิ์ (coefficient) ที่ใช้ในวงจรกรอง , การวาดตำแหน่งของโพล(pole)และซีโร่(zero) , การวาดผลตอบสนองทางความถี่ , ผลตอบสนองทางเฟส , ผลตอบสนองอิมพัลส์(Impulse response) ของวงจรกรองแถบความถี่หยุดผ่าน จากนั้นนำค่าสัมประสิทธิ์ที่ได้เขียนลงบนโปรแกรมด้วยภาษาวีเอชดีแอล(VHDL) เพื่อสร้างวงจรกรองแถบความถี่หยุดผ่านชนิดอิมพัลส์ความยาวไม่จำกัด ลงบนแผนภาพวงจร(Schematic) ก่อนโปรแกรมลงไปในชิพวงจรรวมเอฟพีจีเอ

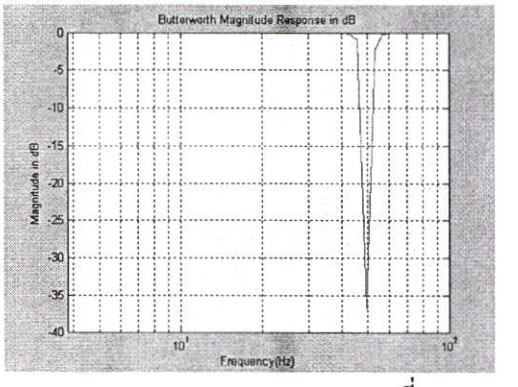
เมื่อกำหนดคุณสมบัติของวงจรกรองแถบความถี่หยุดผ่าน โดยให้ความถี่ตัดด้านต่ำประมาณ 47 เฮิรตซ์ ความถี่ตัดด้านสูงประมาณ 53 เฮิรตซ์ และมีอัตราการลดทอนประมาณ 35 ดีบี ที่ความถี่กลาง 50 เฮิรตซ์ สำหรับวงจรกรองแบบดิจิตอลใช้ความถี่สุ่ม 488 เฮิรตซ์ เมื่อกำหนดค่าต่างๆลงในฟังก์ชัน DBUBSF.M ของโปรแกรม MATLAB แล้วจะได้ตำแหน่งของโพลกับซีโร่ , ผลตอบสนองทางความถี่ , ผลตอบสนองทางเฟส , ผลตอบสนองอิมพัลส์ ของวงจรกรองแถบความถี่หยุดผ่าน เชิงดิจิตอลแบบบัตเตอร์เวิอร์ธ ดังแสดงในรูปที่ 3.5(ก) จนถึงรูปที่ 3.5(ง) ตามลำดับ ส่วนค่าสัมประสิทธิ์สามารถหาค่าโพลีโนเมียลของเศษ (Numerator Polynomial) และโพลีโนเมียลของส่วน(Denominator Polynomial) ได้ดังตารางที่ 3.2

ตารางที่ 3.1 คุณลักษณะของวงจรกรองแถบความถี่หยุดผ่านดิจิตอล แบบบัตเตอร์เวิร์ธ

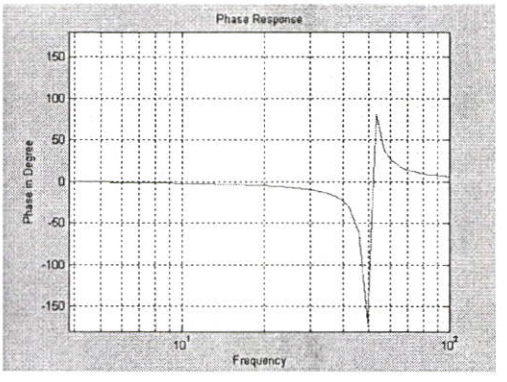
คุณลักษณะ(Specification)	**คุณสมบัติ(Properties)**
ค่าความถี่กลาง(Center Frequency ; f_0)	50 เฮิรตซ์
ค่าคุณภาพ(Quality ; Q)	8.33
อัตราการลดทอนที่ความถี่กลาง	35 ดีบี
ความถี่ของสัญญาณสุ่ม	488 เฮิรตซ์



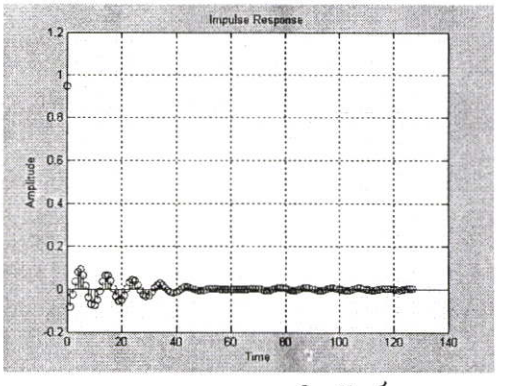
(ก) ตำแหน่งโพลและซีโร่ทางดิจิตอล



(ข) ผลตอบสนองทางความถี่



(ค) ผลตอบสนองทางเฟส



(ง) ผลตอบสนองอิมพัลส์

รูปที่ 3.5 การออกแบบวงจรกรองแถบความถี่หยุดผ่านดิจิตอล แบบบัตเตอร์เวิร์ธ

ตารางที่ 3.2 ตัวประกอบของวงจรกรองแถบความถี่หยุดผ่านดิจิตอล แบบบัตเตอร์เวิร์ธ

โพลีโนเมียล	วงจรกรองความถี่แบบที่ 1			วงจรกรองความถี่แบบที่ 2		
เศษ	$a_0 =$ 1.0000	$a_1 =$ -1.6009	$a_2 =$ 1.0000	$a_0 =$ 1.0000	$a_1 =$ -1.6009	$a_2 =$ 1.0000
ส่วน	$b_0 =$ 1.0000	$b_1 =$ -1.5313	$b_2 =$ 0.9488	$b_0 =$ 1.0000	$b_1 =$ -1.5938	$b_2 =$ 0.9450

3.2 การออกแบบโดยวงจรรวมเอฟพีจีเอ

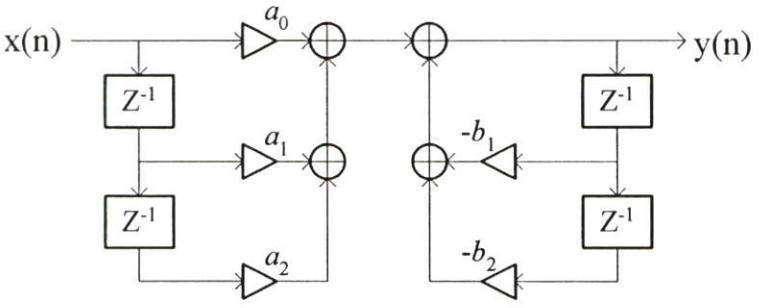
วงจรกรองความถี่ดิจิตอลชนิดอิมพัลส์ความยาวไม่จำกัด สามารถนำไปสร้างใช้งานได้โดยใช้ฟังก์ชันการถ่ายโอนของ *H(z)* ได้โดยตรง การเขียนแผนภาพสำหรับการสร้างวงจรกรองความถี่ เริ่มต้นจากการจัดฟังก์ชันของ *H(z)* อันดับสองตามสมการที่ 3.1

$$H(z) = \frac{a_0 + a_1 z^{-1} + a_2 z^{-2}}{1 + b_1 z^{-1} + b_2 z^{-2}} \qquad (3.1)$$

แผนภาพที่เลือกใช้ของวิทยานิพนธ์ฉบับนี้ ใช้แบบวิธีทางตรงแบบที่ 1 ดังรูปที่ 3.6 ซึ่งได้จากสมการผลต่างของระบบ ถ้าแทน $H(z) = Y(z)/X(z)$ จากนั้นแก้สมการเพื่อหาสมการผลต่างของระบบ[16] ดังสมการที่ 3.2

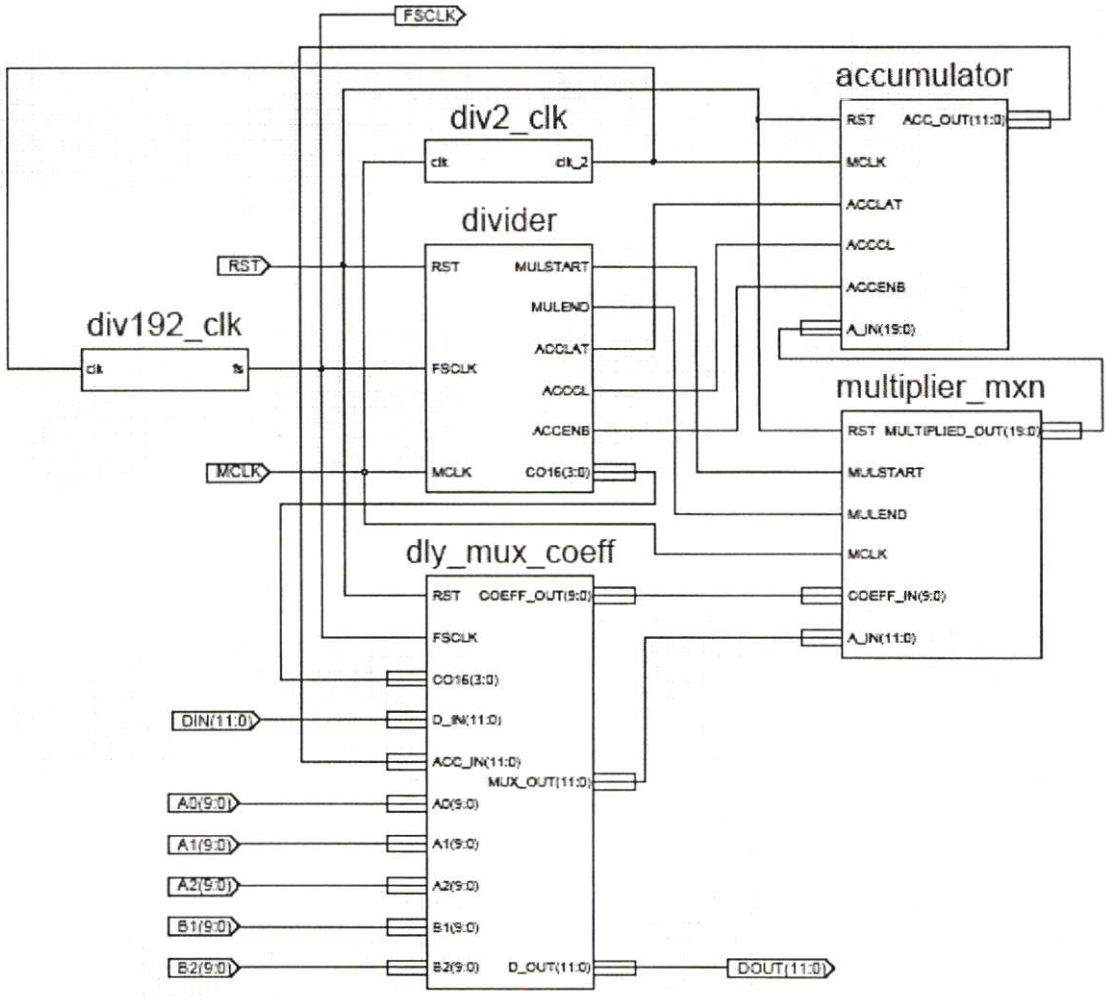
$$\begin{aligned} Y(z)\left\{1 + b_1 z^{-1} + b_2 z^{-2}\right\} &= X(z)\left\{a_0 + a_1 z^{-1} + a_2 z^{-2}\right\} \\ Y(z) &= X(z)\left\{a_0 + a_1 z^{-1} + a_2 z^{-2}\right\} - Y(z)\left\{b_1 z^{-1} + b_2 z^{-2}\right\} \\ y(n) &= a_0 x(n) + a_1 x(n-1) + a_2 x(n-2) - b_1 y(n-1) - b_2 y(n-2) \end{aligned} \qquad (3.2)$$

ซึ่งก็พบว่าสัญญาณเอาต์พุต *y(n)* เกิดจากส่วนที่มาจากสัญญาณอินพุตที่เวลาปัจจุบันและอดีต คือ $a_0x(n) + a_1x(n-1) + a_2x(n-2)$ ลบด้วยส่วนที่มาจากสัญญาณเอาต์พุตในอดีต คือ $b_1y(n-1)-b_2y(n-2)$ ซึ่งสอดคล้องกับที่ได้แสดงในแผนภาพที่ 3.6



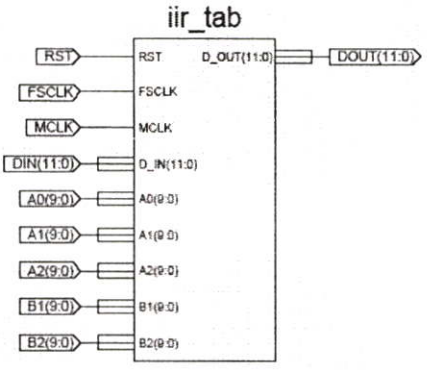
รูปที่ 3.6 แผนภาพการสร้างวงจรกรองความถี่หยุดผ่านแบบไอไออาร์ชนิดทางตรงแบบที่หนึ่ง

เมื่อเขียนแผนภาพของการสร้างวงจรกรองแถบความถี่หยุดผ่านแบบไอไออาร์ ชนิดทางตรงแบบที่หนึ่งอันดับสองแล้ว ต่อมาทำการเขียนโปรแกรมด้วยภาษาวีเอชดีแอล เพื่อสร้างเป็นสัญลักษณ์ (Symbol) ซึ่งประกอบด้วย วงจรหน่วงเวลากับมัลติเพล็กสัมประสิทธิ์(Delay and Multiplex Coefficient ; dly_mux_coeff) , วงจรคูณระหว่างอินพุตกับสัมประสิทธิ์(Multiplier ; multiplier_mxn) , วงจรบวกแบบเต็ม(Full adder ; accumulator) กับวงจรหารความถี่(Clock divider) ซึ่งประกอบไปด้วยสัญญลักษณ์ divider , div193_clk และ div2_clk จากนั้นทำการเชื่อมต่อวงจรตามรูปที่ 3.7



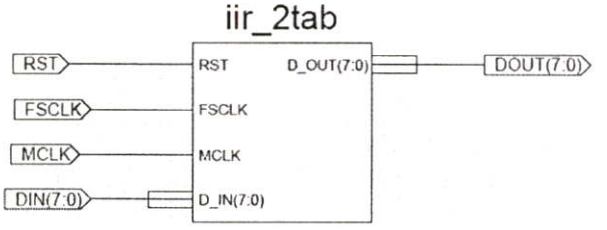
รูปที่ 3.7 วงจรกรองความถี่แบบไอไออาร์ชนิดทางตรงแบบที่หนึ่งอันดับสอง

จากรูปที่ 3.7 ซึ่งเป็นวงจรกรองความถี่แบบไอไออาร์ชนิดทางตรงแบบที่หนึ่งอันดับสอง ที่สร้างจากสัญลักษณ์แล้วเชื่อมต่อกันเป็นวงจร แต่จากการออกแบบตามหัวข้อที่ 3.1 ที่ต้องใช้วงจรกรองความถี่อันดับสองจำนวนสองวงจร ดังนั้นจากรูปวงจรที่ 3.7 สามารถจัดวงจรให้อยู่ในรูปของสัญลักษณ์เพื่อให้การจัดวงจรมีความกระทัดรัดมากขึ้น โดยเรียกใช้อุปกรณ์(component) กับแผนผังขั้ว(Port map) ที่เขียนจากภาษาวีเอชดีแอล ในที่นี้ใช้ชื่อว่า iir_tab.vhd ทำให้ได้สัญลักษณ์ดังแสดงในรูปที่ 3.8



รูปที่ 3.8 สัญลักษณ์ของวงจรกรองความถี่หยุดผ่านแบบไอไออาร์ชนิดทางตรงแบบที่หนึ่งอันดับสอง

ลำดับต่อไปทำการเชื่อมต่อวงจรกรองความถี่อันดับสองของทั้งสองวงจรเข้าด้วยกัน โดยใช้ภาษาวีเอชดีแอล แล้วกำหนดค่าสัมประสิทธิ์ตามตารางที่ 3.2 ในที่นี้ใช้ชื่อว่า iir_2tab.vhd หรือสร้างเป็นสัญลักษณ์ได้ดังแสดงในรูปที่ 3.9 ขั้นตอนต่อไปเป็นการกำหนดขั้วใช้งานได้ดังไปตามตารางที่ 3.3 ลำดับสุดท้ายทำการโปรแกรมผ่านทางพอร์ตขนาน(Parallel Port) ไปยังชิพวงจรรวมเอฟพีจีเอ



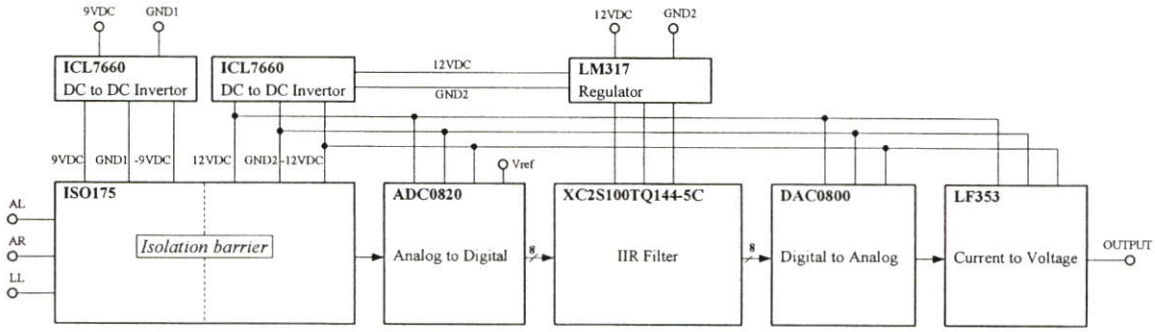
รูปที่ 3.9 สัญลักษณ์ของวงจรกรองแถบความถี่หยุดผ่านแบบไอไออาร์

ตารางที่ 3.3 การเชื่อมต่อขั้วของสัญลักษณ์กับชิพวงจรรวมเอฟพีจีเอ

Port Name	Port Direction	Location	Pad to Setup	Clock to Pad
D_IN<0>	INPUT	p86		N/A
D_IN<1>	INPUT	p95		N/A
D_IN<2>	INPUT	p96		N/A
D_IN<3>	INPUT	p87		N/A
D_IN<4>	INPUT	p99		N/A
D_IN<5>	INPUT	p94		N/A
D_IN<6>	INPUT	p100		N/A
D_IN<7>	INPUT	p93		N/A
D_OUT<0>	OUTPUT	p123	N/A	
D_OUT<1>	OUTPUT	p124	N/A	
D_OUT<2>	OUTPUT	p126	N/A	
D_OUT<3>	OUTPUT	p129	N/A	
D_OUT<4>	OUTPUT	p130	N/A	
D_OUT<5>	OUTPUT	p132	N/A	
D_OUT<6>	OUTPUT	p133	N/A	
D_OUT<7>	OUTPUT	p134	N/A	
FSCLK	INPUT	p101	N/A	N/A
MCLK	INPUT	p91	N/A	N/A
RST	INPUT	p69	N/A	N/A

3.3 ระบบสมบูรณ์ของวงจรขจัดความถี่รบกวน

จากรูปที่ 3.10 เป็นแผนผังการเชื่อมต่อวงจรกรองแถบความถี่หยุดผ่านดิจิตอลกับวงจรแปลงสัญญาณ โดยทางอินพุตประกอบด้วยวงจรขยายสัญญาณแบบแยกโดดที่มีอัตราขยายแรงดัน 1000 เท่า จากนั้นส่งผ่านสัญญาณให้กับ วงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล ในที่นี้ใช้วงจรรวมเบอร์ ADC0820 ซึ่งจะได้ข้อมูลดิจิตอลป้อนไปยังวงจรรวมเอฟพีจีเอเพื่อทำการประมวลผล ส่วนทางด้านเอาต์พุตของวงจรรวมเอฟพีจีเอ ซึ่งก็คือข้อมูลคลื่นไฟฟ้าหัวใจที่ผ่านการกรองความถี่เพื่อขจัดสัญญาณรบกวนออกแล้ว จะถูกส่งสัญญาณไปยังวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอกเชิงกระแส ในที่นี้ใช้วงจรรวมเบอร์ DAC0800 ลำดับสุดท้ายทำการผ่านสัญญาณไปที่วงจรแปลงกระแสเป็นแรงดันเพื่อนำไปใช้งานต่อไป

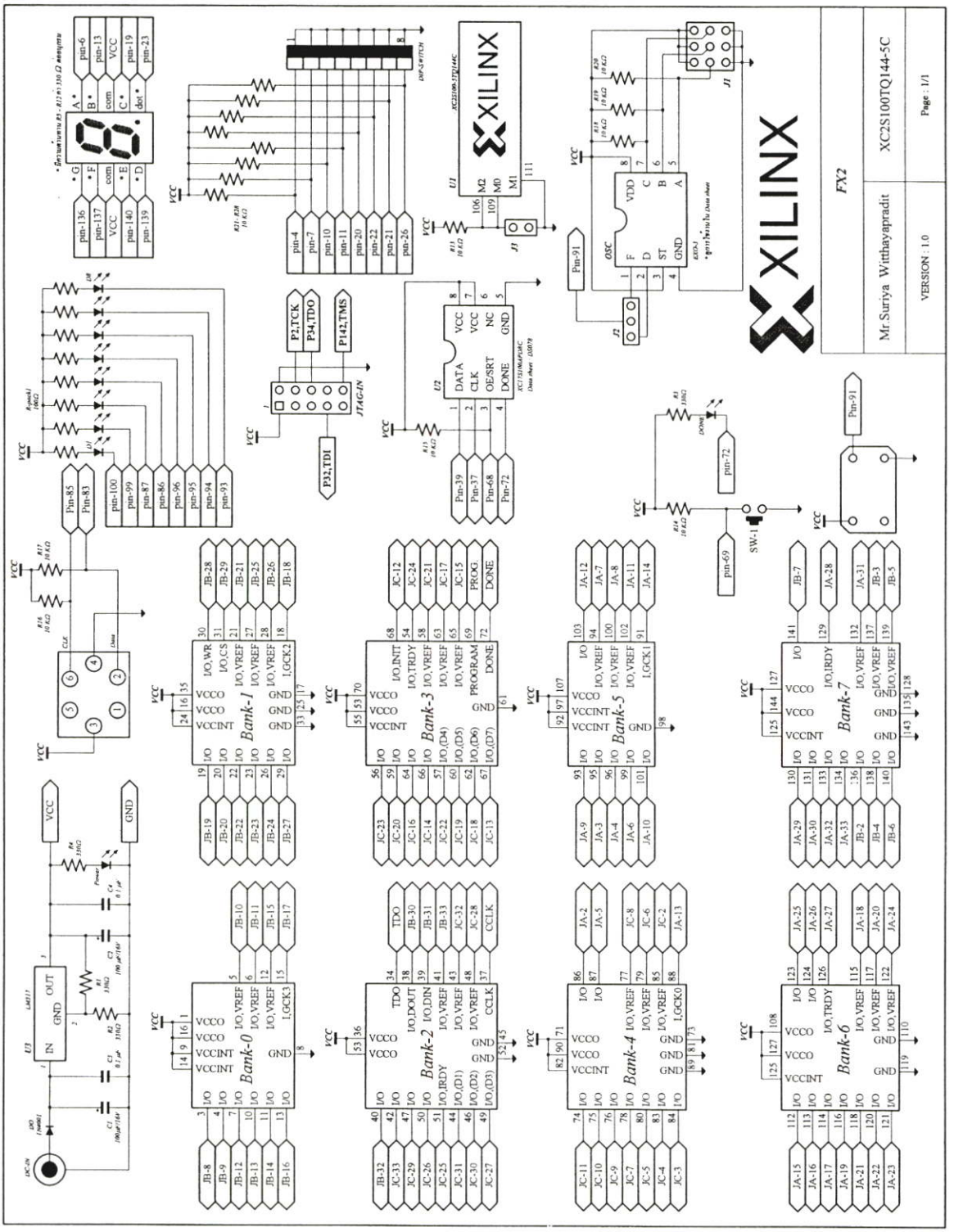


รูปที่ 3.10 การเชื่อมต่อวงจรกรองความถี่กับวงจรแปลงสัญญาณ

สำหรับรูปที่ 3.11 เป็นการแสดงการใช้ทรัพยากรภายในชิพวงจรรวม ส่วนรูปที่ 3.12 แสดงวงจรของชุดทดลองวงจรรวมเอฟพีจีเอ XC2S100TQ144-5C

```
Design Summary
--------------
   Number of errors:      0
   Number of warnings:    1
   Number of Slices:                  324 out of  1,200   27%
   Number of Slices containing
      unrelated logic:                  0 out of    324    0%
   Number of Slice Flip Flops:       260 out of  2,400   10%
   Total Number 4 input LUTs:        492 out of  2,400   20%
      Number used as LUTs:                          471
      Number used as a route-thru:                   21
   Number of bonded IOBs:             17 out of     92   18%
      IOB Flip Flops:                                 8
   Number of GCLKs:                    2 out of      4   50%
   Number of GCLKIOBs:                 2 out of      4   50%
Total equivalent gate count for design:  5,942
Additional JTAG gate count for IOBs:  912
```

รูปที่ 3.11 การใช้ทรัพยากรของชิพวงจรรวมเอฟพีจีเอ

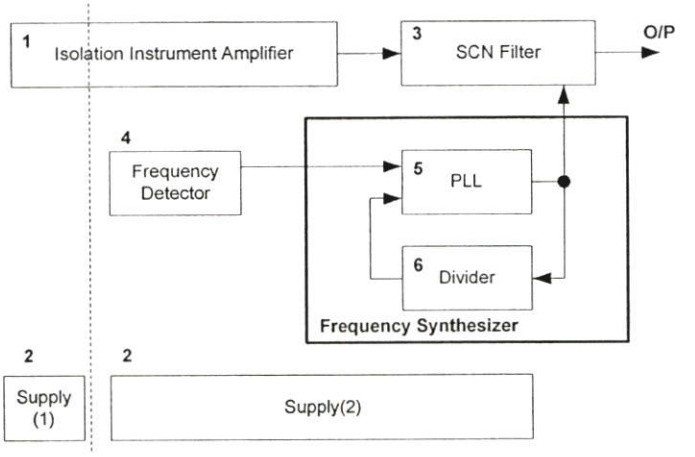


รูปที่ 3.12 วงจรของชุดทดลองวงจรรวมเอฟพีจีเอ XC2S100TQ144-5C

บทที่ 4
วงจรกรองแถบความถี่หยุดผ่านแบบติดตาม

การนำเสนอของบทนี้ ได้นำเสนอวงจรกรองแถบความถี่หยุดผ่านแบบติดตาม ชนิดลูฟเปิด (Open Loop) คือไม่ใช้หลักการป้อนกลับ แต่ใช้หลักการนำสัญญาณรบกวนจริงๆเข้ามาเปรียบเทียบ โดยการทำงานประกอบด้วยวงจรตรวจจับสัญญาณรบกวน (Frequency Detector) , วงจรสังเคราะห์ความถี่ (Frequency Synthesizer) ซึ่งประกอบด้วยวงจรเฟสล็อกลูฟ (Phase Locked Loop) และวงจรหารความถี่ (Frequency Divider) กับวงจรกรองแถบความถี่แบบหยุดผ่านแบบสวิตช์ตัวเก็บประจุ

สำหรับคุณสมบัติของวงจรกรองแถบความถี่หยุดผ่านแบบติดตามนี้ กำหนดให้มีอัตราการลดทอนของสัญญาณรบกวนไม่ต่ำกว่า -35 dB และมีค่าคุณภาพ(Quality factor ; Q)ไม่น้อยกว่า 8 และมีช่วงความถี่ตัดที่สามารถติดตามได้ ต้องมีช่วงความถี่ไม่น้อยกว่า 40 เฮิรตซ์ ถึง 70 เฮิรตซ์ เพื่อนำไปใช้กับการวัดคลื่นไฟฟ้าหัวใจ ซึ่งมีองค์ประกอบของการออกแบบทั้งหมดดังนี้



รูปที่ 4.1 แผนภาพของวงจรกรองแถบความถี่หยุดผ่านแบบติดตามสำหรับวัดคลื่นไฟฟ้าหัวใจ

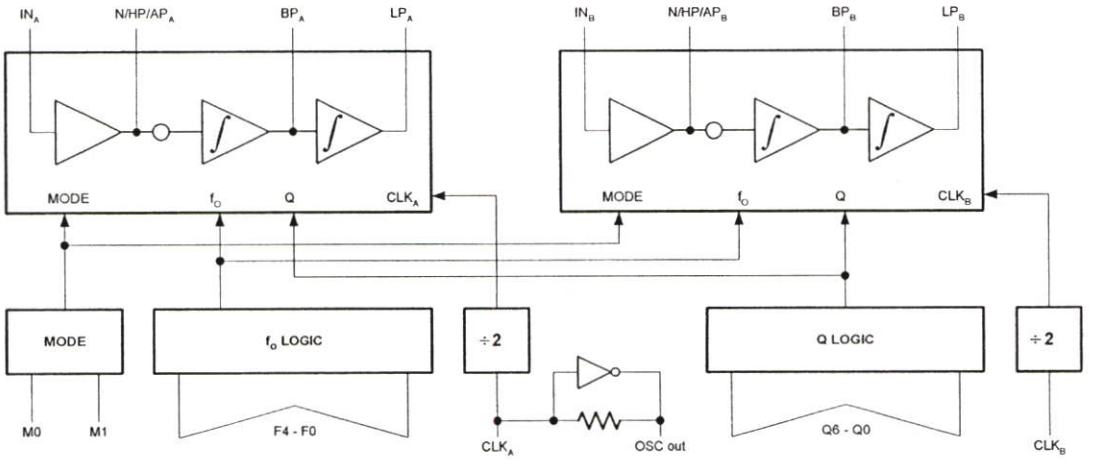
จากแผนภาพในรูปที่ 4.1 อธิบายแต่ละส่วนของวงจรตามรายละเอียดดังนี้ วงจรขยายสัญญาณคลื่นไฟฟ้าหัวใจ(1) ใช้วงจรขยายอินสตรูเมนเตชันแบบแยกโดด ที่มีอัตราขยาย 1000 เท่า เพื่อขยายสัญญาณคลื่นไฟฟ้าจากร่างกายผู้ป่วย ให้กับวงจรกรองแถบความถี่หยุดผ่าน โดยมีวงจรแหล่งจ่ายแรงดันไฟตรงแบบทวิ(Dual Supply) มีสองชุด(2) ใช้กับวงจรส่วนหน้าของวงจรขยายแบบแยกโดดหนึ่งชุด ส่วนแหล่งจ่ายชุดที่สองใช้กับวงจรขยายส่วนสุดท้ายของวงจรขยายแบบแยกโดดกับวงจรกรองแถบความถี่หยุดผ่านแบบติดตาม เพื่อป้องกันอันตรายตามมาตรฐานความปลอดภัยของสมาคมโรคหัวใจแห่งประเทศสหรัฐอเมริกา มิให้เกิดกับตัวผู้ป่วยขณะทำการตรวจวัดคลื่นหัวใจ

วงจรกรองแถบความถี่หยุดผ่าน(3) แบบที่ใช้สวิตช์ตัวเก็บประจุ คือใช้สัญญาณนาฬิกาเป็นตัวเปิด-ปิด สวิตช์ให้กับตัวเก็บประจุภายในของวงจรรวม MAX263 โดยมีการใช้งานแบบขั้วโปรแกรม(pin programmable) เพื่อกำหนดค่าคุณภาพกับค่าอัตราส่วนของสัญญาณนาฬิกาที่ใช้เปิด-ปิดสวิตช์ของวงจรกรองความถี่ ซึ่งสัญญาณนาฬิกาที่ใช้ในการควบคุมนี้ได้จากการสังเคราะห์ความถี่

วงจรตรวจจับความถี่ของสนามไฟฟ้า(4) เป็นวงจรเปรียบเทียบแรงดัน เพื่อทำการเปลี่ยนลักษณะของสัญญาณรบกวนเป็นสัญญาณรูปสี่เหลี่ยมให้กับภาคสังเคราะห์ความถี่ โดยที่ภาคสังเคราะห์ความถี่ประกอบไปด้วยวงจรเฟสล็อกลูฟ(5) กับวงจรหารความถี่(6) ใช้ในการทวีความถี่เป็น 154 เท่าของความถี่รบกวน

4.1 วงจรรวมแบบสวิตช์ตัวเก็บประจุชนิดขั้วโปรแกรม

ในงานวิจัยทั่วไปนิยมใช้วงจรรวมแบบสวิตช์ตัวเก็บประจุ เบอร์ MF10 แต่จากการออกแบบด้วยวงจรรวมดังกล่าว ปรากฏว่าค่าคุณภาพมีค่าต่ำ อีกทั้งอัตราการขจัดสัญญาณรบกวนมีค่าต่ำเช่นกัน ดังนั้นวิทยานิพนธ์ฉบับนี้เลือกใช้วงจรรวมแบบสวิตช์ตัวเก็บประจุเบอร์ MAX263 ซึ่งมีโครงสร้างแบบสวิตช์ประจุชนิดขั้วโปรแกรมใช้กับวงจรกรองความถี่หลายรูปแบบ(Pin Programmable Universal Filters) ตามแผนผังดังรูปที่ 4.2



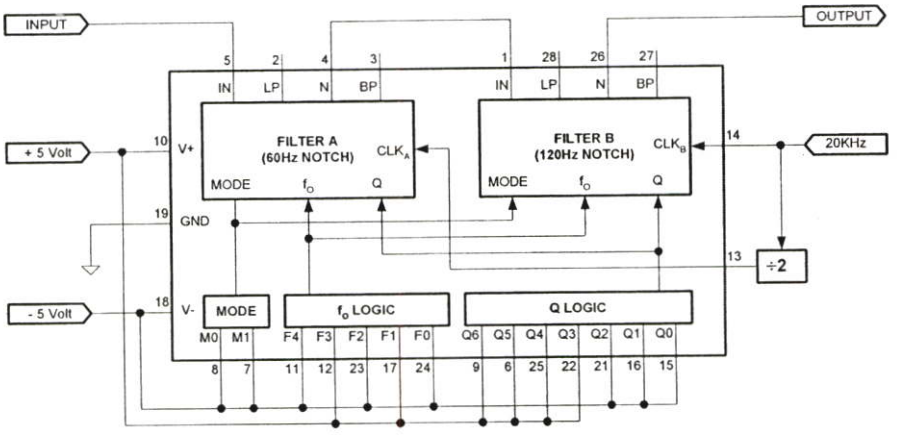
รูปที่ 4.2 แผนผังภายในของวงจรรวมแบบสวิตช์ตัวเก็บประจุชนิดขั้วโปรแกรม

จากแผนผังภายในของวงจรรวมแบบสวิตช์ตัวเก็บประจุชนิดขั้วโปรแกรม จะเห็นว่ามีวงจรกรองความถี่อยู่สองวงจรด้วยกัน วงจรแรกมีขั้วอินพุตคือ IN_A ส่วนขั้วเอาต์พุตมีสามขั้วด้วยกันตามชนิดของวงจรกรองความถี่ ขั้วเอาต์พุตแรกสำหรับวงจรกรองแถบความถี่น็อตช์(Notch Filter;N) กับวงจรกรองความถี่สูงผ่าน(High-pass Filter;HP) กับวงจรกรองความถี่ผ่าน(All-pass Filter;AP) ส่วนขั้ว

เอาต์พุตถัดมา ใช้กับวงจรกรองแถบความถี่ผ่าน(Band-pass Filter;BP) สุดท้ายขั้วเอาต์พุตของวงจรกรองความถี่ต่ำผ่าน(Low-pass Filter;LP)

ต่อมาเป็นอินพุตของขั้วโปรแกรม แบ่งออกเป็นสี่ชุดด้วยกันคือ ขั้วสำหรับเลือกโหมด(Mode) ของผลตอบสนองความถี่มีสองขั้วคือ M0 กับขั้ว M1 ชุดที่สองเป็นขั้วสำหรับเลือกค่าความถี่ตัดมีทั้งหมด 5 ขั้วด้วยกันคือ F0 – F4 ชุดที่สามเป็นขั้วสำหรับเลือกค่าคุณภาพ มีทั้งหมด 7 ขั้วด้วยกันคือ Q0 – Q6

สุดท้ายคือขั้วอินพุตสำหรับความถี่ของสัญญาณนาฬิกา CLK_A และ CLK_B ที่ใช้ในการสวิตช์ของตัวเก็บประจุของวงจรกรองความถี่บล็อก A กับบล็อก B ตามลำดับ



รูปที่ 4.3 วงจรกรองแถบความถี่น็อตช์ที่ความถี่กลาง 60 เฮิรตซ์กับความถี่ตัด 120 เฮิรตซ์

จากรูปที่ 4.3 เป็นการใช้วงจรรวมแบบสวิตช์ตัวเก็บประจุ สำหรับวงจรกรองแถบความถี่น็อตช์ที่มีความถี่กลาง 60 เฮิรตซ์กับความถี่ตัด 120 เฮิรตซ์ โดยใช้โหมด 0 และมีค่าคุณภาพเท่ากับ 8 ดังที่จะกล่าวถึงการใช้งานอย่างละเอียดดังนี้

4.1.1 การเลือกโหมดของผลตอบสนองความถี่

การเลือกโหมดของผลตอบสนองความถี่ มีขั้วอินพุต M0 กับ M1 ป้อนระดับแรงดันสูง(High) หรือระดับแรงดันต่ำ(Low) ให้กับขั้วอินพุต เพื่อเลือกผลตอบสนองทางความถี่ของวงจรกรองความถี่แต่ละชนิด ซึ่งวิทยานิพนธ์ฉบับนี้เลือกโหมดที่ 1 (M0=0 , M1=0) เพราะว่า นอกจากให้ผลตอบสนองแบบแถบความถี่หยุดผ่านแล้ว ขนาดของแถบความถี่ผ่านด้านต่ำ(H_{ON1}) และขนาดของแถบผ่านด้านสูง(H_{ON2}) มีระดับที่เท่าเทียมกัน ดังแสดงตามตารางที่ 4.1 ส่วนโหมดที่ 2 นั้นให้ขนาดของแถบความถี่ผ่านด้านต่ำด้วยอัตราส่วนครึ่งเท่าของผลตอบสนองทางขนาดแถบผ่านด้านสูง สำหรับโหมดที่ 3A สามารถปรับผลตอบสนองทางขนาดของแถบความถี่ผ่านด้านต่ำ และผลตอบสนองทางขนาดของแถบผ่านด้านสูงได้ แต่ต้องเพิ่มวงจรรวมสัญญาณภายนอก

ตารางที่ 4.1 การเลือกโหมดของผลตอบสนองความถี่

โหมด	M1,M0	ฟังก์ชันตัวกรอง	f_N	H_{OLP}	H_{OBP}	H_{ON1} ($f \rightarrow 0$)	H_{ON2} ($f \rightarrow f_{CLK}/4$)	อื่นๆ
1	0,0	LP,BP,**N**	f_0	-1	-Q	-1	-1	
2	0,1	LP,BP,**N**	$f_0\sqrt{2}$	-0.5	$-Q/\sqrt{2}$	-0.5	-1	
3	1,0	LP,BP,HP		-1	-Q			$H_{OHP} = -1$
3A	1,0	LP,BP,HP,**N**	$f_0\sqrt{\frac{R_H}{R_L}}$	-1	-Q	$+\frac{R_G}{R_L}$	$+\frac{R_G}{R_H}$	$H_{OHP} = -1$
4	1,1	LP,BP,AP		-2	-2Q			$H_{OAP} = -1$ $f_Z=f_0, Q_Z=Q$

4.1.2 การเลือกอัตราส่วนของความถี่สัญญาณนาฬิกาต่อความถี่กลาง

การใช้โครงข่ายสวิตช์ประจุสำหรับวงจรกรองความถี่จะต้องคำนึงถึง ความถี่ของสัญญาณนาฬิกา(Clock Frequency ; f_{CLK}) และค่าความถี่กลางของวงจรกรองความถี่(Center Frequency ; f_O) สำหรับวงจรรวมเบอร์ MAX263 ได้กำหนดค่าความถี่ของสัญญาณนาฬิกาและค่าความถี่กลางเป็นค่าอัตราส่วน ดังสมการที่ 4.1

$$f_{CLK} / f_O = \pi \cdot (N + 32) \tag{4.1}$$

โดยที่

f_{CLK} หมายถึง ความถี่ของสัญญาณนาฬิกา มีหน่วยเป็นเฮิรตซ์

f_O หมายถึง ความถี่กลาง มีหน่วยเป็นเฮิรตซ์

π หมายถึง ค่าคงที่มีค่าเท่ากับ 3.14159

N หมายถึง เลขจำนวนเต็มลำดับ

การออกแบบวงจรกรองแถบความถี่หยุดผ่านของวิทยานิพนธ์ฉบับนี้ มีวัตถุประสงค์ให้วงจรกรองแถบความถี่หยุดผ่าน มีความสามารถติดตามค่าความถี่กลางได้ และมีความแม่นยำสูง ดังนั้นการพิจารณาเลือกค่าอัตราส่วนของความถี่สัญญาณนาฬิกาต่อความถี่กลาง เพื่อให้ได้ค่าความถี่สัญญาณนาฬิกาและความถี่กลางที่ใกล้เคียงจำนวนเต็มมากที่สุด เมื่อค่าของ N มีค่าตั้งแต่ 0 จนถึง 31 จะเห็นว่าในแบบที่ 1 ค่าอัตราส่วนของความถี่สัญญาณนาฬิกาต่อความถี่กลางเท่ากับ 153.98 มีความเหมาะสมที่สุด ซึ่งสอดคล้องกับค่า N เท่ากับ 17 ทำการแปลงค่า N = 17 ให้อยู่ในรูปเลขฐานสอง ได้ค่าเท่ากับ 10001 ตามขั้วอินพุต F4 ถึง F0 ตามลำดับ

เมื่อกำหนดให้ขั้วอินพุต F4 จนถึง F0 มีค่า 10001 ตามตารางที่ 4.2 แล้ว จะเห็นว่าจากสมการที่ 4.1 เมื่อทำการเปลี่ยนค่าความถี่ของสัญญาณนาฬิกา ทำให้ค่าความถี่กลางเปลี่ยนค่าด้วย

ตารางที่ 4.2 การเลือกอัตราส่วนของความถี่สัญญาณนาฬิกาต่อความถี่กลาง

อัตราส่วน f_{CLK}/f_0	รหัสโปรแกรม					
โหมด 1 (M0=0,M1=0)	**N**	**F4**	**F3**	**F2**	**F1**	**F0**
150.80	16	1	0	0	0	0
153.98	**17**	**1**	**0**	**0**	**0**	**1**
157.08	18	1	0	0	1	0
160.22	19	1	0	0	1	1

4.1.3 การกำหนดค่าคุณภาพภาพของวงจรกรองแถบความถี่หยุดผ่าน

การออกแบบวงจรกรองแถบความถี่หยุดผ่าน สำหรับการวัดสัญญาณคลื่นไฟฟ้าหัวใจ ควรมีค่าคุณภาพสูง หรือแถบความถี่ที่ต้องการขจัดมีแถบที่เล็กที่สุด โดยที่แถบของความถี่คำนวณได้จากสมการที่ 4.2 และค่าคุณภาพของวงจรกรองแถบความถี่หยุดผ่าน คำนวณได้จากสมการที่ 4.3

$$BW = f_{C2} - f_{C1} \tag{4.2}$$

$$Q = \frac{f_O}{BW} \tag{4.3}$$

โดยที่

BW หมายถึง แถบของความถี่หรือแบนด์วิดท์(Band width)

f_{C1} หมายถึง ค่าความถี่ตัดด้านต่ำ(Low frequency Cut-off)

f_{C2} หมายถึง ค่าความถี่ตัดด้านสูง(High frequency Cut-off)

Q หมายถึง ค่าคุณภาพของวงจรกรองความถี่

สำหรับวงจรรวมเบอร์ MAX263 นี้ สามารถกำหนดค่าคุณภาพได้ตั้งแต่ 0.5 จนถึง 64 ซึ่งระบุเป็นเลขจำนวนเต็มลำดับ(N) ดังสมการที่ 4.4 จากค่าคุณภาพของวงจรกรองแถบความถี่หยุดผ่านของวิทยานิพนธ์ฉบับนี้ได้กำหนดให้มีค่าเท่ากับ 8 เมื่อแทนค่าในสมการที่ 4.4 จะได้ค่าจำนวนเต็มลำดับ

เท่ากับ 120 ซึ่งแปลงค่าเป็นเลขฐานสองได้เท่ากับ 1111000 เพื่อกำหนดเป็นขั้วอินพุต Q6 จนถึง Q0 ตามลำดับ ดังตารางที่ 4.3

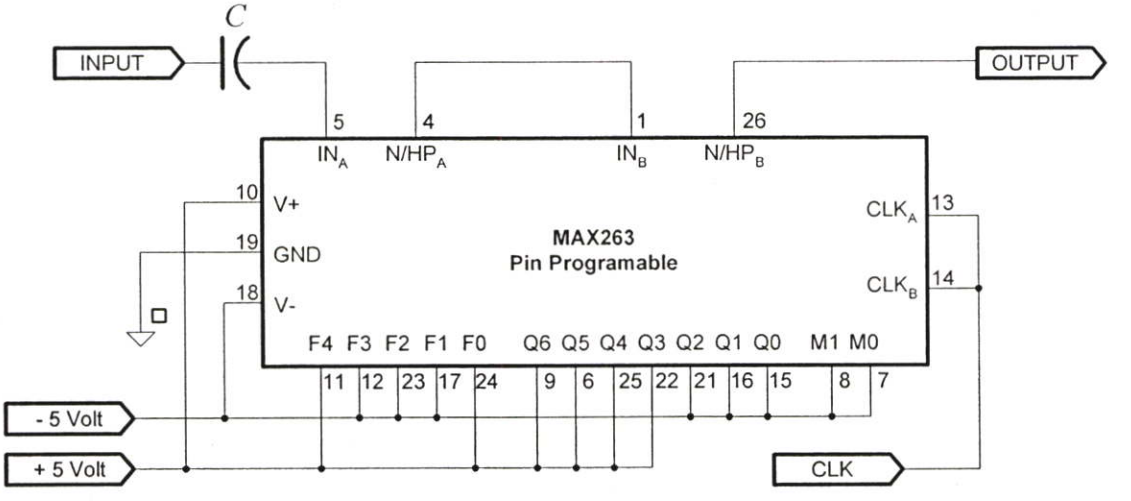
$$N = 128 - \left(\frac{64}{Q}\right) \tag{4.4}$$

ตารางที่ 4.3 การกำหนดค่าตัวประกอบคุณภาพของวงจรกรองความถี่

ค่าตัวประกอบคุณภาพ(Q)	รหัสโปรแกรม							
โหมด 1 (M0=0,M1=0)	**N**	**Q6**	**Q5**	**Q4**	**Q3**	**Q2**	**Q1**	**Q0**
8.00	120	1	1	1	1	0	0	0

4.1.4 วงจรกรองหยุดแถบความถี่โดยใช้สวิตช์ตัวเก็บประจุ

ถ้าให้ความถี่ของสัญญาณนาฬิกาของวงจรกรองความถี่ A เท่ากับสัญญาณนาฬิกาของวงจรกรองความถี่ B ทำให้คุณสมบัติของวงจรกรองความถี่ทั้งสองเหมือนกันทุกประการ เมื่อนำวงจรกรองความถี่ A กับ B ซึ่งเป็นวงจรกรองความถี่อันดับสอง เชื่อมต่ออนุกรมกันทำให้ได้วงจรกรองความถี่อันดับสี่ จากการออกแบบวงจรกรองความถี่ที่ได้กล่าวมาแล้วในหัวข้อที่ 4.1.1 ถึง 4.1.3 จะได้ลักษณะของวงจรกรองความถี่แบบติดตามของวงจรรวมสวิตช์ตัวเก็บประจุดังรูปที่ 4.4



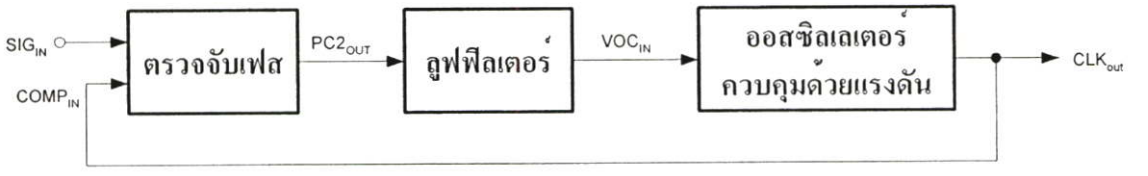
รูปที่ 4.4 วงจรกรองแถบความถี่หยุดผ่านอันดับสี่ โดยใช้โครงข่ายสวิตช์ตัวเก็บประจุ

4.2 วงจรสังเคราะห์ความถี่

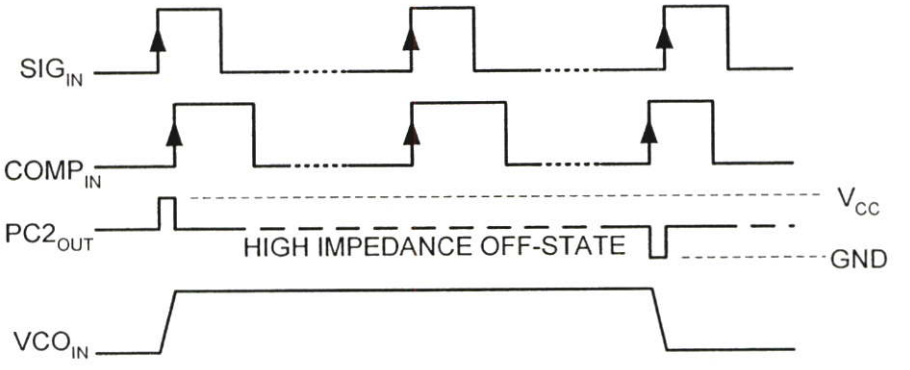
วงจรสังเคราะห์ความถี่ใช้กันมากทางด้านการสื่อสาร ทำหน้าที่ทวีความถี่เป็นจำนวนเท่าของความถี่ทางด้านอินพุต วงจรสังเคราะห์ความถี่ประกอบด้วยวงจรเฟสล็อกลูฟ และวงจรหารความถี่

4.2.1 วงจรเฟสล็อกลูฟ

วงจรเฟสล็อกลูฟประกอบด้วย วงจรตรวจจับเฟส(Phase Detector) หรือวงจรเปรียบเทียบเฟส (Phase Comparator) โดยทำการเปรียบเทียบสัญญาณนาฬิกา CLKin กับสัญญาณที่ผ่านการป้อนกลับมาจากขั้วเอาต์พุตของวงจรออสซิลเลเตอร์ควบคุมด้วยแรงดัน สัญญาณเอาต์พุตของวงจรเปรียบเทียบเฟสจะส่งผ่านไปยังวงจรกรองความถี่ต่ำผ่าน หรือที่เรียกว่า ลูฟฟิลเตอร์(loop filter) เพื่อแปลงผลต่างของการเปรียบเทียบเฟสให้เป็นแรงดันไฟฟ้ากระแสตรง เพื่อป้อนให้กับวงจรออสซิลเลเตอร์ควบคุมด้วยแรงดัน เขียนเป็นแผนผังดังรูปที่ 4.5 สำหรับการเปรียบเทียบเฟสซึ่งมีสองแบบ โดยแบบที่ 1 ใช้กับสัญญาณอินพุตที่มีค่าวัฏจักร(Duty Cycle) เท่ากับ 50 เปอร์เซ็นต์ ในที่นี้เลือกเปรียบเทียบเฟสแบบที่ 2 (Phase Comparator 2) เพราะใช้การเปรียบเทียบมุมเฟส ตามแผนภาพในรูปที่ 4.6

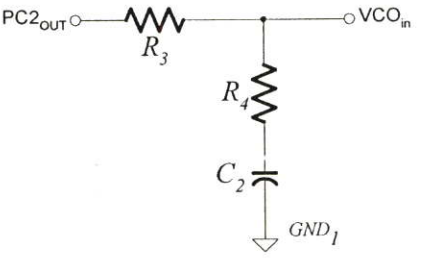


รูปที่ 4.5 แผนผังของวงจรเฟสล็อกลูฟ



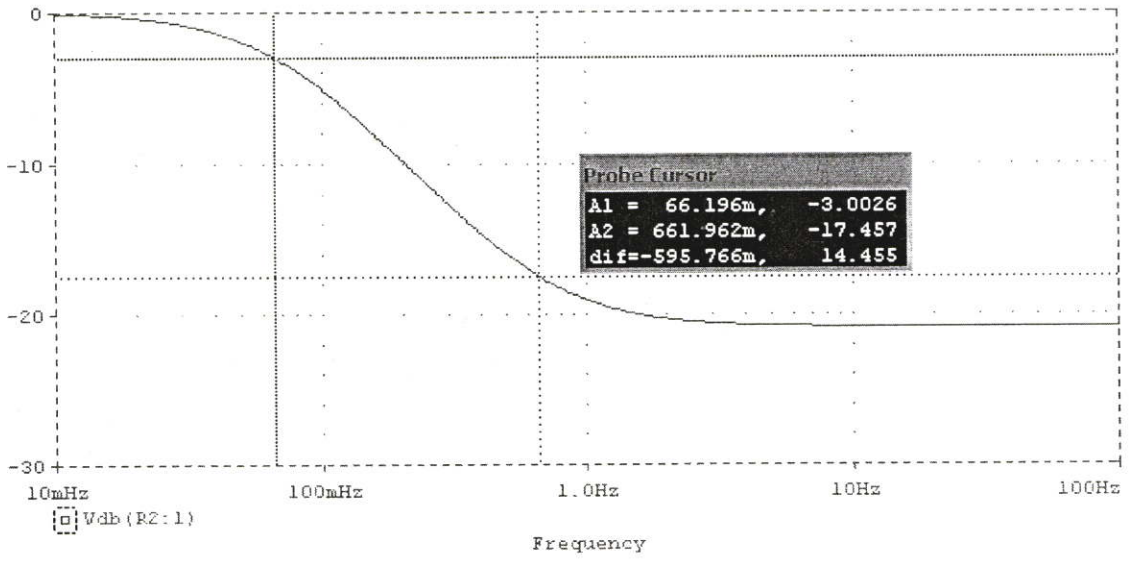
รูปที่ 4.6 แผนภาพของการเปรียบเทียบเฟสของวงจรตรวจจับเฟส

วงจรกรองความถี่ต่ำผ่านหรือลูฟฟิลเตอร์ สามารถใช้ได้ทั้งแบบแอกทีฟและแบบพาสซีฟ สำหรับกรองสัญญาณการตรวจจับเฟสไปเป็นแรงดันไฟตรง จำเป็นต้องออกแบบให้มีความถี่ตัดต่ำมาก ในที่นี้เลือกใช้แบบพาสซิฟ ตามรูปที่ 4.7 ซึ่งเขียนสมการถ่ายโอนได้ดังสมการที่ 4.5 กำหนดให้ความต้านทาน R_3 มีค่าเท่ากับ 1MΩ ความต้านทาน R_4 มีค่าเท่ากับ 100KΩ และตัวเก็บประจุ C_2 มีค่าเท่ากับ 22 ไมโครฟารัด เมื่อทำการจำลองเพื่อหาผลตอบสนองทางความถี่ ได้ผลตอบสนองดังรูปที่ 4.8



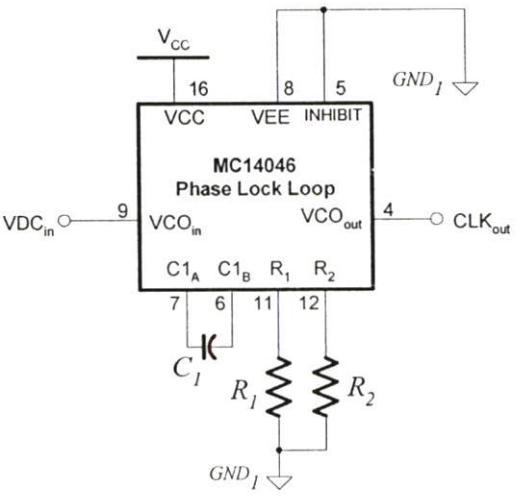
รูปที่ 4.7 วงจรกรองความถี่ต่ำแบบพาสซีฟ

$$F(s) := \frac{R_4 sC + 1}{(R_3 + R_4)sC + 1} \tag{4.5}$$



รูปที่ 4.8 ผลตอบสนองทางความถี่ของวงจรลูฟฟิลเตอร์

การออกแบบวงจรออสซิลเลเตอร์ควบคุมด้วยแรงดัน ที่ประยุกต์ใช้จากวงจรรวมเบอร์ MC14046 ดังรูปที่ 4.9 ได้กำหนดขอบเขตของความถี่ด้านต่ำและความถี่ด้านสูงได้จากสมการที่ 4.6 และสมการที่ 4.7 ตามลำดับ



รูปที่ 4.9 วงจรออสซิลเลเตอร์ควบคุมด้วยแรงดัน

$$f_{min} = \frac{1}{R_2(C_1 + 32pF)} \tag{4.6}$$

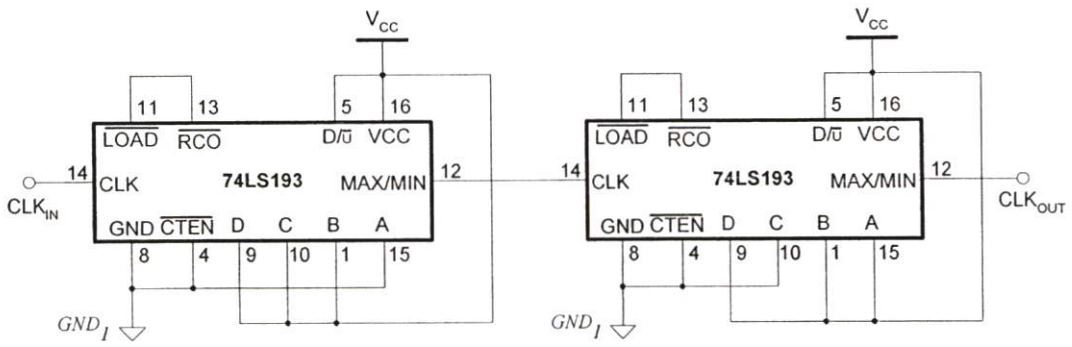
$$f_{max} = \frac{1}{R_1(C_1 + 32pF)} \tag{4.7}$$

4.2.2 วงจรหารความถี่สัญญาณนาฬิกา

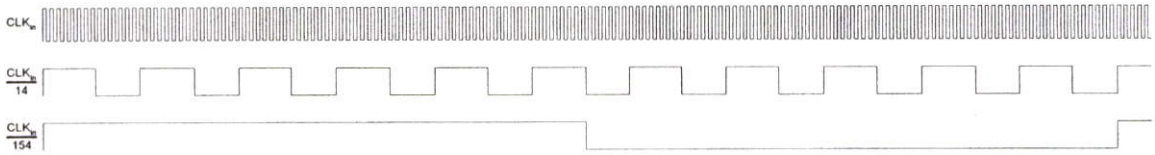
จากคุณสมบัติของวงจรกรองแถบความถี่หยุดผ่านแบบติดตามที่ต้องการขจัดสัญญาณรบกวนที่มีความถี่ 40 เฮิรตซ์ ถึง 70 เฮิรตซ์ ดังนั้นความถี่ของสัญญาณนาฬิกาจึงมีขอบเขตของความถี่ในช่วง 6.16 กิโลเฮิรตซ์จนถึง 10.78 กิโลเฮิรตซ์ ฉนั้นค่าความต้านทาน R_2 กับความต้านทาน R_1 ตามรูปที่ 4.9 มีค่าโดยประมาณ 470 กิโลโอห์ม กับ 240 กิโลโอห์ม ตามลำดับ

สัญญาณนาฬิกา ซึ่งมีขอบเขตของความถี่ในช่วง 6.16 กิโลเฮิรตซ์ ถึง 10.78 กิโลเฮิรตซ์ จะถูกลดความถี่ลงมา โดยใช้วงจรหารความถี่ของสัญญาณนาฬิกา ซึ่งกำหนดให้หารความถี่ลงมาที่ 154 เท่า ทำให้ได้ความถี่ทางเอาต์พุตของวงจรหารความถี่มีขอบเขตในช่วงความถี่ 40 เฮิรตซ์ จนถึง 70 เฮิรตซ์ ตามลำดับ

การหารความถี่จำนวน 154 เท่านั้น ต้องใช้การหารขนาด 8 บิต ในที่นี้ใช้วงจรหารขนาด 4 บิต สองวงจร ดังแสดงในรูปที่ 4.10 โดยวงจรที่หนึ่งจะทำการหารความถี่ของสัญญาณนาฬิกาลงมาที่ 14 เท่า กระทำโดยกำหนดให้ขั้วอินพุต DCBA มีค่าเป็น 1110 ส่วนวงจรที่สองหารความถี่ลงมาอีก 11 เท่า กระทำโดยกำหนดให้ขั้วอินพุต DCBA มีค่าเป็น 1011 จะได้แผนภาพทางเวลาของสัญญาณทางเอาต์พุตของวงจรหารความถี่ของสัญญาณนาฬิกา ดังรูปที่ 4.11



รูปที่ 4.10 วงจรหารความถี่สัญญาณนาฬิกา



รูปที่ 4.11 แผนภาพทางเวลาของวงจรหารความถี่สัญญาณนาฬิกา

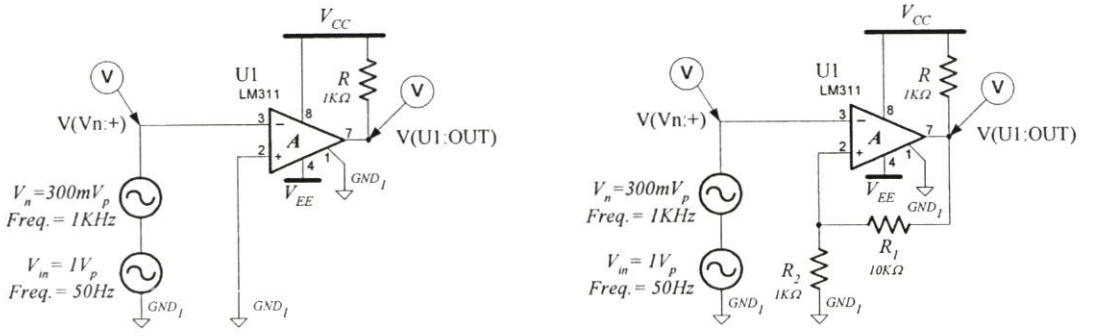
4.2.3 วงจรตรวจจับความถี่ของสัญญาณรบกวน

ความถี่จากสายส่งกำลังไฟฟ้า จะมีค่าความถี่ 50 เฮิรตซ์ หรือ 60 เฮิรตซ์ เมื่อวัดสัญญาณด้วยออสซิลโลสโคป จะเห็นว่า สัญญาณที่วัดได้ มิได้เป็นฟังก์ชันของไซนูซอยด์อย่างสมบูรณ์ เมื่อทำการปรับสัญญาณให้เป็นสัญญาณรูปสี่เหลี่ยม โดยใช้วงจรเปรียบเทียบแรงดัน กรณีที่ใช้การเปรียบเทียบแบบพื้นฐาน ดังแสดงในรูปที่ 4.12(ก) จะทำให้การเปลี่ยนค่าความถี่ผิดพลาดได้ ซึ่งจะเห็นได้ดังรูปที่ 4.13(ก) ดังนั้นวงจรเปรียบเทียบแรงดันควรเลือกใช้แบบฮิสเทอริซีส (Hysteresis Comparator) ดังรูปที่ 4.12(ข) จะทำให้การปรับสัญญาณเป็นรูปสี่เหลี่ยมมีค่าความถี่ตรงกับความถี่ของสายส่งกำลัง ดังแสดงในรูปที่ 4.13(ข) พร้อมกันนี้ได้นำเสนอผลการทดลองวัดสัญญาณของวงจรเปรียบเทียบแรงดันทั้งสองแบบในบทที่ 5

จากวงจรเปรียบเทียบแรงดันแบบฮิสเทอริซีสของรูปที่ 4.12(ข) สามารถคำนวณหาค่าขีดจำกัดบน(Upper Limit ; ULP) ของการเปลี่ยนสถานะทางด้านเอาต์พุตได้ ดังสมการที่ 4.8

$$V_{ULP} = \left(\frac{R_2}{R_1 + R_2} \right) V_{OUT(MAX)} \tag{4.8}$$

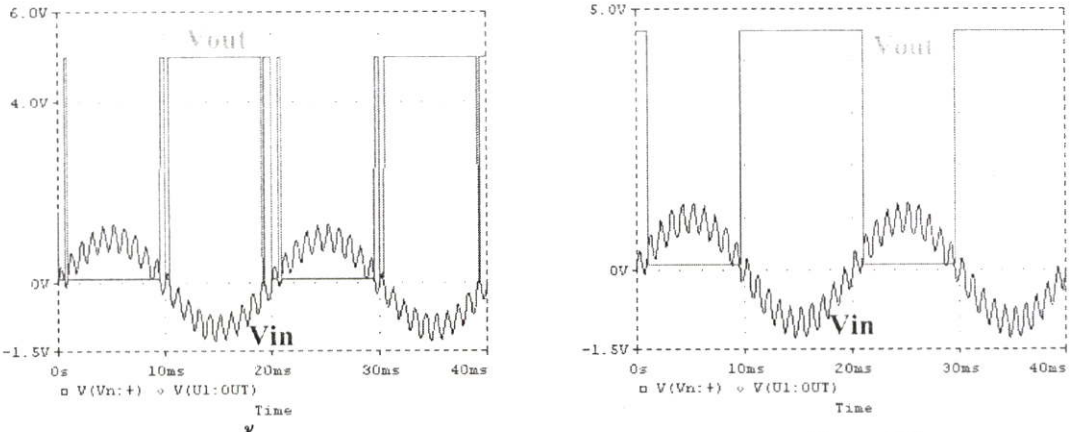
ถ้ากำหนดให้แรงดัน V_{CC} เท่ากับ 5 โวลต์ และ V_{EE} เท่ากับ -5 โวลต์ และค่าแรงดันขีดจำกัดบนเท่ากับ 0.5 โวลต์ จะได้ค่าความต้านทาน R_1 เท่ากับ 10 KΩ และความต้านทาน R_2 เท่ากับ 1KΩ หลังจากได้จำลองการทำงานโดยใช้โปรแกรม P-Spice จะได้ผลการจำลองดังรูปที่ 4.13



(ก) วงจรเปรียบเทียบแรงดันแบบพื้นฐาน (ข) วงจรเปรียบเทียบแรงดันแบบฮิสเทอริซีส

รูปที่ 4.12 วงจรเปรียบเทียบแรงดันที่ใช้จำลองการทำงาน

เมื่อกำหนดขนาดของอินพุต(Vin) ตามรูปที่ 4.12 แล้วจำลองการทำงานด้วยโปรแกรม P-Spice เพื่อสังเกตุผลลัพธ์ทางเอาต์พุต(Vout) โดยพิจารณาทางโดเมนเวลาทำให้ได้ผลลัพธ์ของวงจรเปรียบเทียบแรงดันของแบบพื้นฐานกับแบบฮิสเทอริซิส ดังรูปที่ 4.13(ก) กับ 4.13(ข) ตามลำดับ



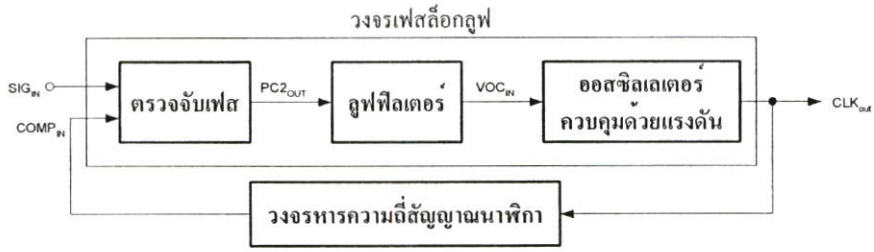
(ก) แบบพื้นฐาน (ข) แบบฮิสเทอริซีส

รูปที่ 4.13 ผลการจำลองวงจรเปรียบเทียบแรงดัน

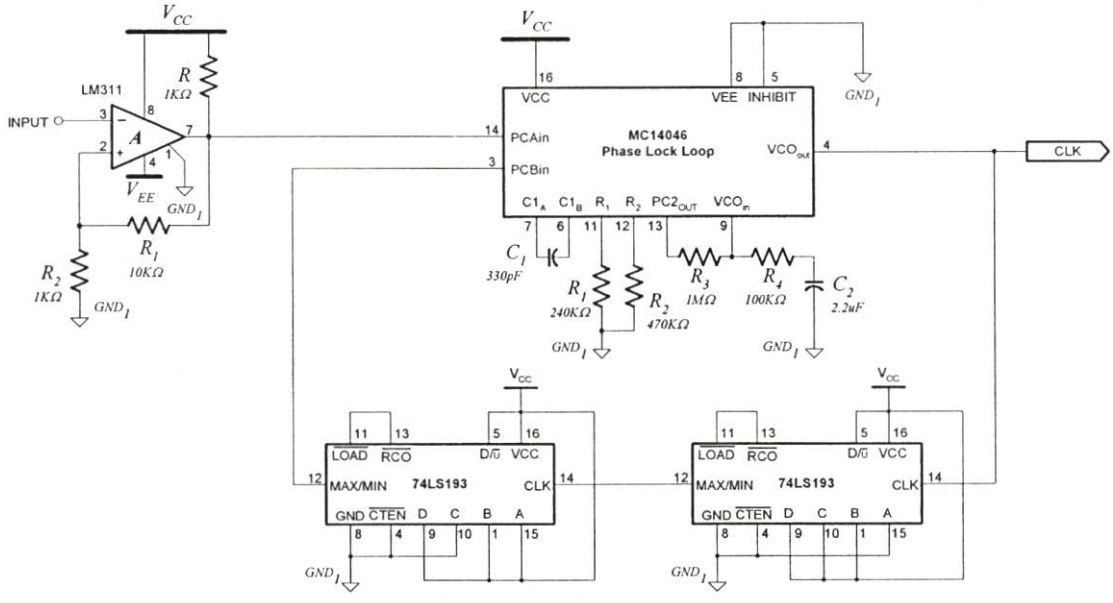
การสังเคราะห์ความถี่โดยใช้วงจรเฟสล็อกลูฟ สามารถเขียนเป็นแผนผังได้ดังรูปที่ 4.14 ซึ่งประกอบด้วยวงจรเฟสล็อกลูฟกับวงจรหารความถี่สัญญาณนาฬิกา โดยสัญญาณเอาต์พุตของวงจรสังเคราะห์ความถี่ ซึ่งเป็นสัญญาณเอาต์พุตของวงจรออสซิลเลเตอร์ควบคุมด้วยแรงดันจะถูกหารความถี่

ลงมาเพื่อเปรียบเทียบกับเฟสของสัญญาณอินพุต SIG_{IN} ซึ่งเป็นสัญญาณแบบคาบเวลา เมื่อสัญญาณอินพุตทั้งสองของวงจรตรวจจับเฟสมีความต่างเฟส จะได้สัญญาณเอาต์พุตของวงจรตรวจจับเฟส เป็นสัญญาณรูปสี่เหลี่ยมและส่งต่อไปยังวงจรกรองความถี่ต่ำผ่าน เพื่อกรองให้เป็นสัญญาณไฟฟ้ากระแสตรงแล้วป้อนให้กับวงจรกำเนิดสัญญาณนาฬิกาต่อไป

เมื่อทำการรวมวงจรเฟสล็อกลูฟ ซึ่งประกอบไปด้วยวงจรตรวจจับเฟส วงจรกรองความถี่ต่ำผ่านและวงจรออสซิลเลเตอร์ควบคุมด้วยแรงดัน เข้ากับวงจรหารความถี่สัญญาณนาฬิกา และวงจรตรวจจับความถี่ของสัญญาณรบกวน จะได้วงจรสังเคราะห์ความถี่เพื่อทำหน้าที่ทวีความถี่ 154 เท่าของความถี่จากสายส่งกำลัง เพื่อใช้ในการควบคุมสัญญาณนาฬิกาของวงจรรวมโครงข่ายสวิตช์ประจุต่อไป ดังแสดงในรูปที่ 4.15



รูปที่ 4.14 แผนผังของวงจรสังเคราะห์ความถี่



รูปที่ 4.15 วงจรสังเคราะห์ความถี่

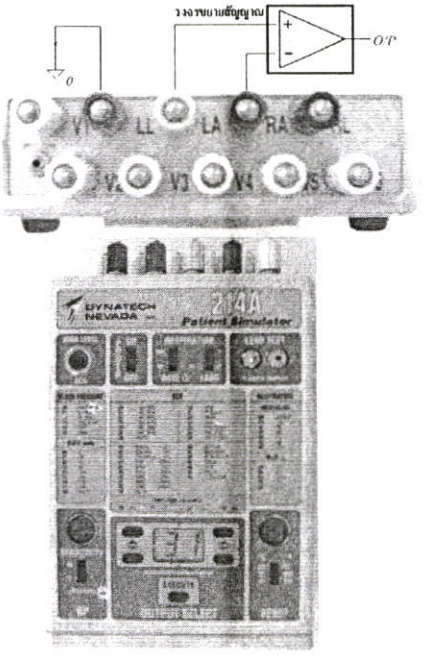
บทที่ 5
ผลการทดลองและการวิเคราะห์

ในบทนี้กล่าวถึง ผลการทดสอบวงจรขยายสัญญาณแบบแยกโดดทั้งแบบวงจรรวมชิพเดี่ยว และวงจรขยายสัญญาณแบบเชื่อมโยงทางแสง , การวัดผลตอบสนองทางความถี่ กับผลตอบสนองทางมุมเฟสของวงจรกรองแถบความถี่หยุดผ่านทั้งแบบดิจิตอลและเชิงแอนะลอก ลำดับต่อมาทำการนำสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบรวมกับความถี่รบกวนป้อนให้กับวงจรกรองแถบความถี่หยุดผ่าน แล้วทำการวัดสัญญาณที่ผ่านการขจัดความถี่รบกวนออกแล้ว ลำดับสุดท้ายทำการวัดหาค่าผิดพลาดแบบกำลังสองเฉลี่ย(Mean Square Error ; MSE) ซึ่งมีรายละเอียดต่างๆดังต่อไปนี้

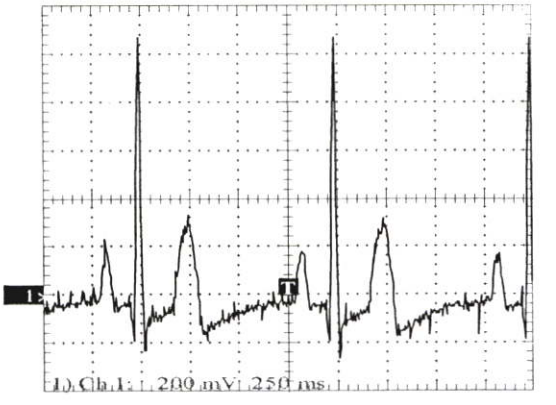
5.1 การทดสอบวงจรขยายสัญญาณอินสทรูเมนเตชันแบบแยกโดด

5.1.1 วงจรขยายสัญญาณแบบแยกโดดชนิดชิพเดี่ยว

เมื่อทำการเชื่อมต่อวงจรขยายสัญญาณเข้ากับแหล่งจ่ายแรงดันแล้ว และเชื่อมต่ออินพุตของวงจรขยายสัญญาณเข้ากับเครื่องจำลองสัญญาณคลื่นไฟฟ้าหัวใจ(ECG Patient Simulator) ยี่ห้อ DYNATECH NEVADA รุ่น 214A Patient Simulator ดังรูปที่ 5.1(ก) ทำการวัดสัญญาณทางด้านเอาต์พุต จะได้ลักษณะสัญญาณไฟฟ้าขนาด 1.2 โวลต์และมีคาบเวลาเท่ากับ 1 วินาที ดังรูปที่ 5.1(ข)



(ก) เครื่องจำลองสัญญาณคลื่นไฟฟ้าหัวใจ

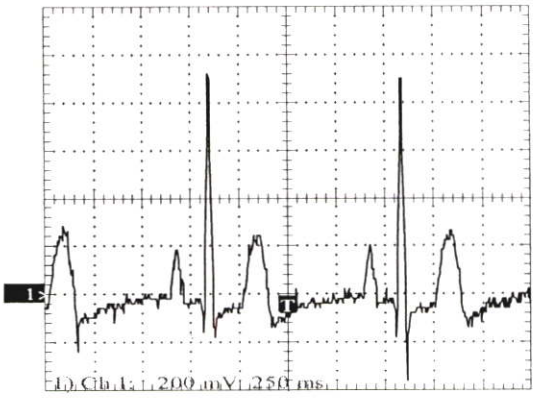


(ข) สัญญาณเอาต์พุตของวงจรขยายแบบแยกโดด

รูปที่ 5.1 ผลการทดลองวงจรขยายสัญญาณแบบแยกโดด

5.1.2 วงจรขยายสัญญาณแบบแยกโดดชนิดเชื่อมโยงทางแสง

ลำดับต่อมา ทำการเชื่อมต่อวงจรขยายสัญญาณแบบเชื่อมโยงทางแสงตามรูปที่ 2.11 เข้ากับแหล่งจ่ายแรงดัน ทำการต่ออินพุตของวงจรขยายสัญญาณเข้ากับเครื่องจำลองสัญญาณคลื่นไฟฟ้าหัวใจ จากนั้นทำการวัดสัญญาณทางด้านเอาต์พุต จะได้ลักษณะสัญญาณไฟฟ้าขนาด 1.1 โวลต์และมีคาบเวลาเท่ากับ 1 วินาที ดังแสดงในรูปที่ 5.2



รูปที่ 5.2 สัญญาณทางเอาต์พุตของวงจรขยายสัญญาณแบบเชื่อมโยงทางแสง

จากการทดลองวงจรขยายสัญญาณทั้งสองวงจร พบว่ารูปสัญญาณคลื่นไฟฟ้าหัวใจมีความสอดคล้องกันมาก โดยการทดลองต่อไปนี้ใช้วงจรขยายสัญญาณแบบแยกโดดสำหรับวงจรกรองความถี่แบบดิจิตอล ส่วนวงจรขยายสัญญาณแบบเชื่อมโยงทางแสงใช้กับวงจรกรองความถี่แบบติดตาม

ก่อนทำการวัดคุณภาพของวงจรกรองแถบความถี่หยุดผ่าน วิทยานิพนธ์ฉบับนี้ขอนำเสนอการสร้างสัญญาณเพื่อใช้ในการทดสอบคุณสมบัติของวงจรกรองที่ได้ออกแบบไว้ดังนี้

5.2 การสร้างสัญญาณคลื่นไฟฟ้าหัวใจเพื่อทดสอบวงจรกรองแถบความถี่หยุดผ่าน

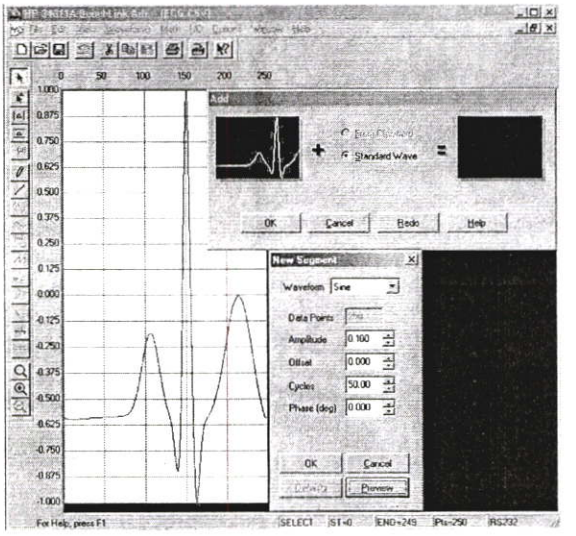
การสร้างสัญญาณคลื่นไฟฟ้าหัวใจเพื่อทดสอบ เป็นการนำสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบรวมกับความถี่รบกวน เพื่อกำหนดเป็นสัญญาณอินพุตของวงจรกรองแถบความถี่หยุดผ่าน ในที่นี้นำเสนอสองแนวทางดังนี้

5.2.1 การสร้างสัญญาณคลื่นไฟฟ้าหัวใจเพื่อการทดสอบผ่านโปรแกรมคอมพิวเตอร์

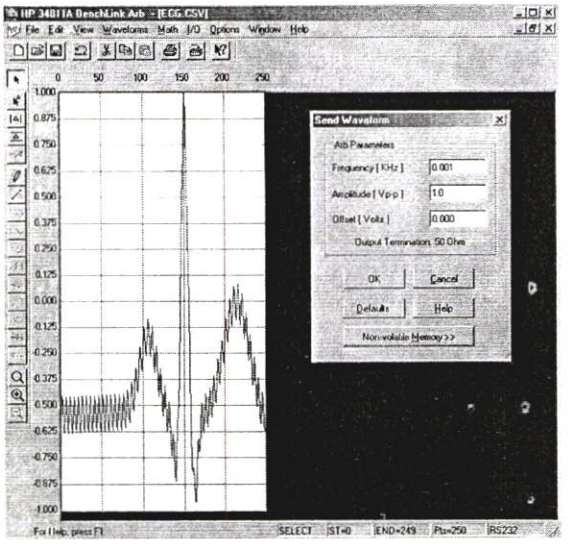
ในขั้นตอนนี้ใช้สมการเชิงคณิตศาสตร์เพื่อสร้างสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบ[3] ซึ่งสร้างจากฟังก์ชันของโปรแกรม MATLAB แล้วจัดข้อมูลให้อยู่ในรูปแบบแถว(Row) และสดมภ์(Column) ลำดับต่อมาจัดเก็บเป็นแฟ้มข้อมูลชนิดที่คั่นด้วยจุลภาค(Comma delimited ; CSV) จากนั้นเปิดโปรแกรม

Benchlink แล้วนำเข้า(Import) สัญญาณที่จัดเก็บไว้ จะได้รูปสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบเท่ากับหนึ่งวงรอบสัญญาณ

ลำดับต่อมาทำการผสมสัญญาณคลื่นไฟฟ้าหัวใจเข้ากับสัญญาณรบกวน โดยเลือกเมนู Math กับ Add ตามลำดับ แล้วทำการป้อนค่าองค์ประกอบของสัญญาณรบกวน ดังรูปที่ 5.3 ซึ่งจากรูปที่แสดงได้กำหนดองค์ประกอบของสัญญาณรบกวน เป็นสัญญาณรูปไซน์ ขนาดแรงดัน 100 มิลลิโวลต์ มีค่าแรงดันออฟเซ็ตเท่ากับศูนย์ ที่ความถี่ 50 เฮิรตซ์ จะได้สัญญาณคลื่นไฟฟ้าหัวใจที่ผสมกับสัญญาณรบกวน จากนั้นส่งสัญญาณดังกล่าวผ่านพอร์ตการสื่อสารตามมาตรฐาน RS-232 ไปยังเครื่องกำเนิดสัญญาณหลายรูปแบบ (Arbitrary Function generator) โดยเลือกการส่งที่เมนู I/O กับ Send Waveform ตามลำดับ ดังรูปที่ 5.4

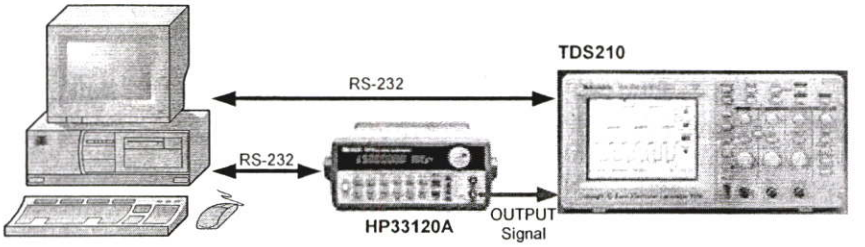


รูปที่ 5.3 สัญญาณคลื่นไฟฟ้าหัวใจต้นแบบรวมกับองค์ประกอบของสัญญาณรบกวน



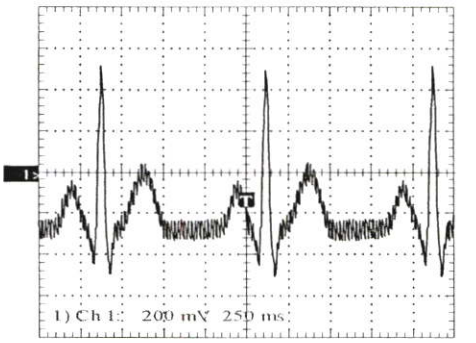
รูปที่ 5.4 สัญญาณคลื่นไฟฟ้าหัวใจเพื่อการทดสอบส่งไปยังเครื่องกำเนิดสัญญาณ

วิธีการสร้างสัญญาณคลื่นไฟฟ้าหัวใจเพื่อการทดสอบแบบวิธีการนี้ มีข้อดีที่สามารถปรับองค์ประกอบของสัญญาณรบกวนได้ง่าย เมื่อได้สัญญาณเพื่อการทดสอบแล้ว ขั้นตอนสุดท้ายเป็นการส่งไปที่เครื่องกำเนิดสัญญาณหลายรูปแบบ ยี่ห้อ HEWLETT PACKARD รุ่น HP33120A ดังในรูปที่ 5.5

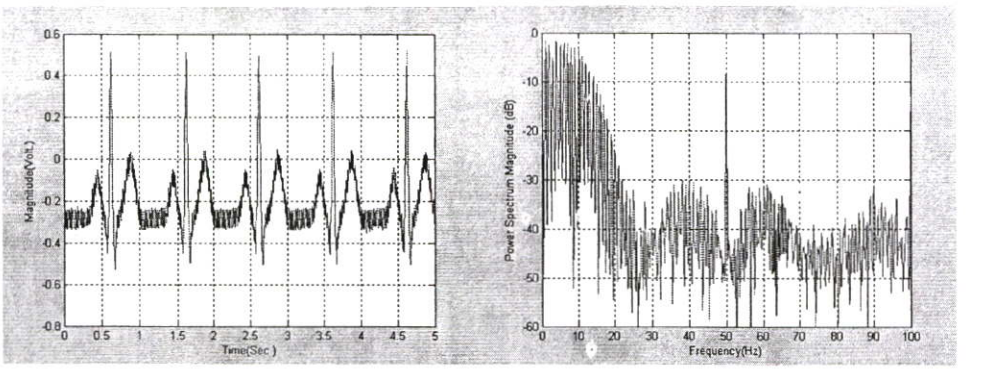


รูปที่ 5.5 การเชื่อมต่อสัญญาณระหว่างคอมพิวเตอร์กับเครื่องกำเนิดสัญญาณ

จากรูปที่ 5.5 เมื่อใช้ออสซิลโลสโคป(Oscilloscope) วัดสัญญาณที่ขั้วเอาต์พุตของเครื่องกำเนิดสัญญาณ แล้วใช้โปรแกรม WaveStar ทำการบันทึกรูปสัญญาณ จะได้สัญญาณคลื่นไฟฟ้าหัวใจเพื่อการทดสอบวงจรกรองแถบความถี่หยุดผ่านดังแสดงในรูปที่ 5.6 และเมื่อนำข้อมูลจากการบันทึกนี้ไปวิเคราะห์องค์ประกอบของสัญญาณ โดยใช้โปรแกรม MATLAB จะได้ลักษณะของสัญญาณทางโดเมนเวลา และโดเมนความถี่ดังรูปที่ 5.7 เมื่อคำนวณหาอัตราส่วนของสัญญาณต่อสัญญาณรบกวน (Signal to noise Ratio ; SNR) เท่ากับ 25.94 ดีบี กับค่าผิดพลาดแบบกำลังสองเฉลี่ยเท่ากับ 165.6



รูปที่ 5.6 สัญญาณคลื่นไฟฟ้าหัวใจเพื่อการทดสอบ

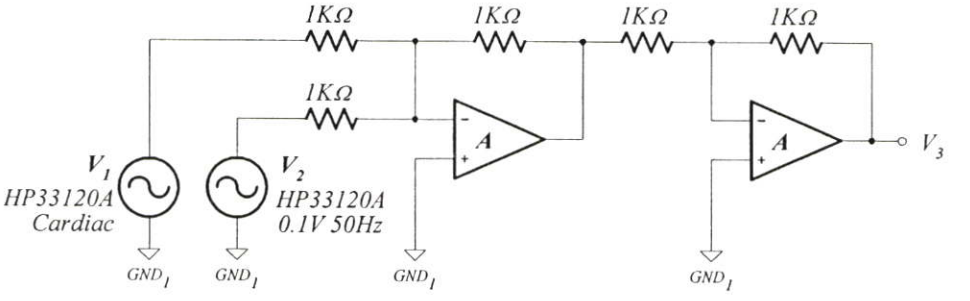


รูปที่ 5.7 ลักษณะของสัญญาณคลื่นไฟฟ้าหัวใจทางโดเมนเวลา กับโดเมนความถี่

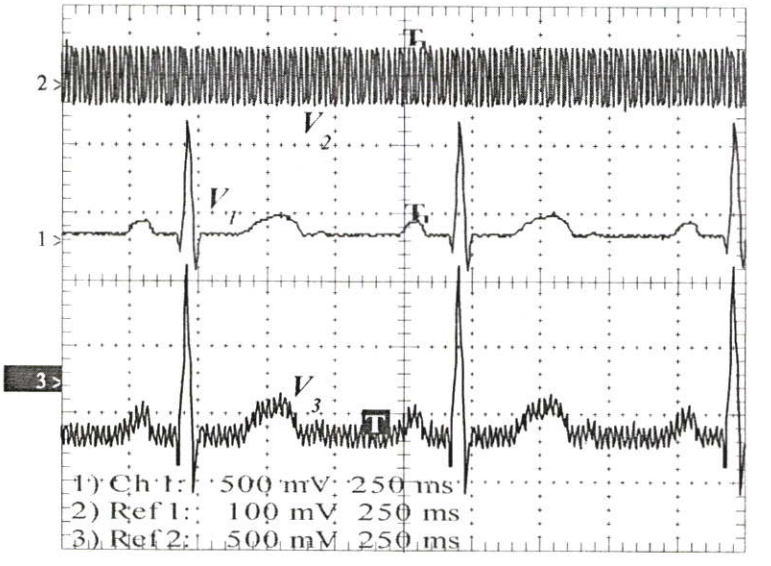
จากรูปที่ 5.7 เมื่อพิจารณาทางโดเมนความถี่ จะเห็นได้ว่าที่ความถี่ 50 เฮิรตซ์ มีขนาดของสัญญาณรบกวนประมาณ 35 – 38 ดีบี

5.2.2 การสร้างเพื่อการทดสอบจากวงจรรวมสัญญาณ

การใช้วงจรรวมสัญญาณ สำหรับสร้างสัญญาณคลื่นไฟฟ้าหัวใจเพื่อการทดสอบวงจรกรองความถี่ ใช้เครื่องกำเนิดสัญญาณสองแหล่งด้วยกัน โดยแหล่งจ่ายที่หนึ่งใช้สร้างสัญญาณคลื่นไฟฟ้าหัวใจ และแหล่งจ่ายที่สองใช้สร้างสัญญาณรบกวน จากรูปที่ 5.8 กำหนดให้ V_1 คือสัญญาณคลื่นไฟฟ้าหัวใจ โดยใช้เครื่องกำเนิดสัญญาณ HP33120A ที่มีรูปแบบของสัญญาณคลื่นไฟฟ้าหัวใจอยู่ภายในและกำหนดให้ V_2 คือสัญญาณรบกวนที่มีขนาดแรงดัน 0.1 โวลต์ ที่ความถี่ 50 เฮิรตซ์ เมื่อทำการวัดสัญญาณ V_1 ด้วยออสซิลโลสโคปคือสัญญาณหมายเลข 1 ตามมาตรวัด Ch1 แล้ววัดสัญญาณ V_2 ด้วยออสซิลโลสโคปคือสัญญาณหมายเลข 2 ตามมาตรวัด Ref1 กับวัดสัญญาณเอาต์พุต V_3 ด้วยออสซิลโลสโคปคือสัญญาณหมายเลข 3 ตามมาตรวัด Ref2 ดังรูปที่ 5.9

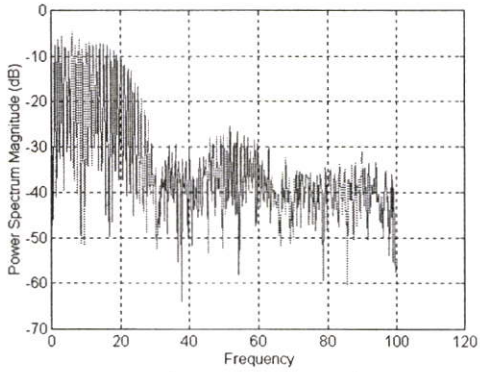


รูปที่ 5.8 วงจรรวมสัญญาณคลื่นไฟฟ้าหัวใจกับสัญญาณรบกวน

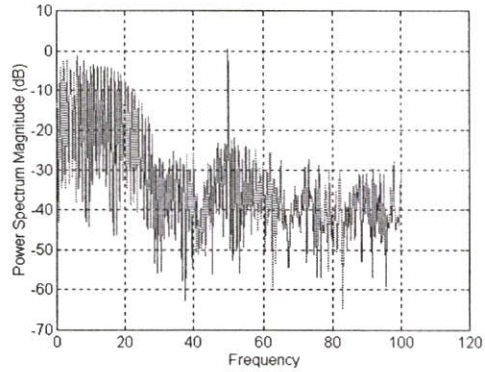


รูปที่ 5.9 สัญญาณทางด้านอินพุตของวงจรรวมสัญญาณ

การวัดสัญญาณไฟฟ้า รูปที่ 5.9 สัญญาณด้านบนคือสัญญาณรบกวนขนาด 0.1 โวลต์ความถี่ 50 เฮิรตซ์ สัญญาณตรงกลางคือสัญญาณคลื่นไฟฟ้าหัวใจจากเครื่องกำเนิดสัญญาณ HP33120A ขนาดแรงดัน 1 โวลต์ความถี่ 1 เฮิรตซ์ สัญญาณด้านล่างคือสัญญาณด้านเอาต์พุตของวงจรรวมสัญญาณ เมื่อทำการวิเคราะห์หาองค์ประกอบของความถี่ของสัญญาณคลื่นไฟฟ้าหัวใจ และสัญญาณทางด้านเอาต์พุตได้ดังแสดงในรูปที่ 5.10



(ก) โดเมนความถี่ของสัญญาณคลื่นไฟฟ้าหัวใจ



(ข) โดเมนความถี่ของสัญญาณด้านเอาต์พุต

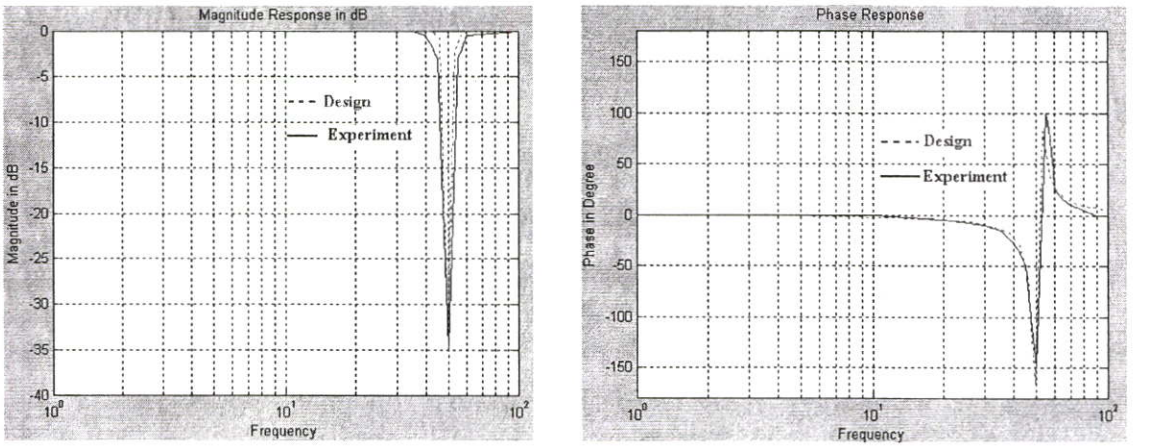
รูปที่ 5.10 โดเมนความถี่ของสัญญาณคลื่นไฟฟ้าด้านเอาต์พุต

การสร้างสัญญาณเพื่อใช้ทดสอบวงจรกรองความถี่ วิทยานิพนธ์ฉบับนี้ เลือกใช้การสร้างผ่านโปรแกรมคอมพิวเตอร์ เพราะสามารถปรับเปลี่ยนคุณสมบัติของสัญญาณรบกวนได้ง่าย ไม่ว่าจะใช้การปรับเปลี่ยนในโปรแกรม MATLAB หรือการปรับเปลี่ยนจากโปรแกรม BenchLink จะมีความสะดวกมากกว่า

ส่วนแนวทางที่สองต้องมีวงจรรวมสัญญาณเพิ่มเติม และต้องใช้เครื่องกำเนิดสัญญาณสองแหล่งด้วยกัน ทำให้ยุ่งยากต่อการทดลอง อีกทั้งการปรับตั้งค่าความถี่รบกวนจะต้องมีความแม่นยำจำเป็นต้องใช้เครื่องกำเนิดสัญญาณที่มีประสิทธิภาพสูง

5.3 วงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอล

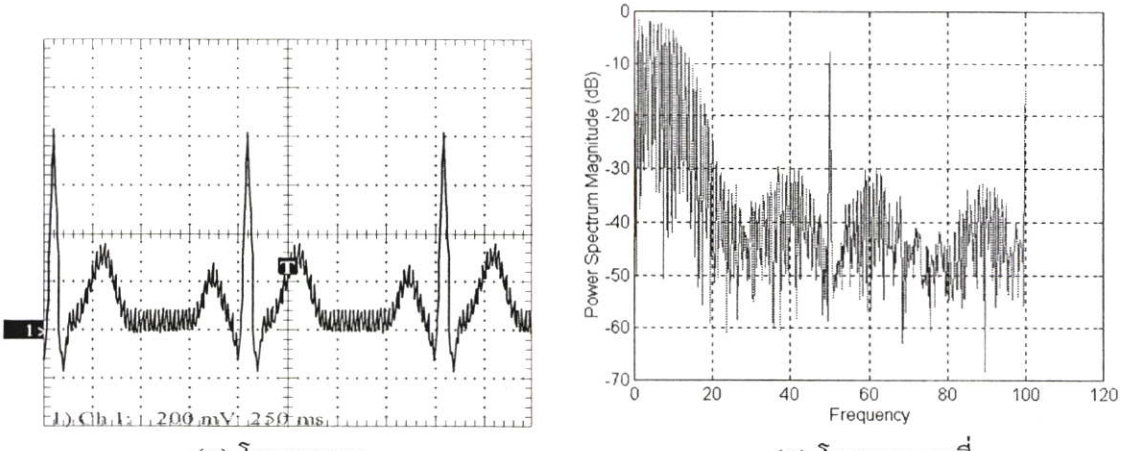
จากการโปรแกรมวงจรกรองแถบความถี่หยุดผ่านลงชิพวงจรรวมเอฟพีจีเอแล้ว ต่อไปทำการเชื่อมต่อวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล กับวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก แล้วทำการป้อนสัญญาณรูปไซน์กำหนดให้ความถี่ 1 เฮิรตซ์ถึงความถี่ 100 เฮิรตซ์ ที่ขนาดแรงดัน 1 โวลต์ ที่ขั้วอินพุตของวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอล ทำการวัดขนาดของแรงดันและเฟสของสัญญาณทางด้านขั้วเอาต์พุตของวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณแอนะลอก จะได้ผลตอบสนองทางความถี่ กับผลตอบสนองทางมุมเฟส ดังรูปที่ 5.11



(ก) ผลตอบสนองทางขนาด (ข) ผลตอบสนองทางมุมเฟส

รูปที่ 5.11 ผลตอบสนองของวงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอล

ลำดับต่อมาทำการป้อนสัญญาณคลื่นไฟฟ้าหัวใจทดสอบให้กับวงจรกรองแถบความถี่ โดยใช้สัญญาณคลื่นไฟฟ้าหัวใจขนาดแรงดัน 1 โวลต์รวมกับสัญญาณรบกวนขนาด 100 มิลลิโวลต์ที่ความถี่ 50 เฮิรตซ์ ทำการวัดสัญญาณคลื่นไฟฟ้าหัวใจทางด้านอินพุตของวงจรกรองความถี่ ดังแสดงในรูปที่ 5.12(ก) และนำค่าที่วัดได้ทำการวิเคราะห์หาโดเมนความถี่ โดยใช้โปรแกรม MATLAB ทำการวิเคราะห์สัญญาณคลื่นไฟฟ้าหัวใจทดสอบได้โดเมนความถี่ของสัญญาณคลื่นไฟฟ้าหัวใจทดสอบ ดังแสดงในรูปที่ 5.12(ข) ซึ่งจากรูปจะเห็นว่าขนาดของสัญญาณรบกวนมีค่าประมาณ 35 ดีบี

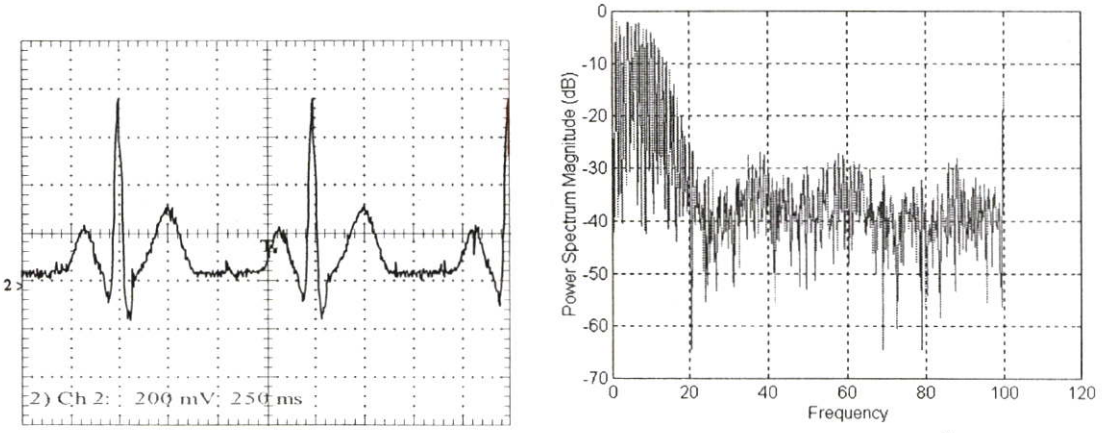


(ก) โดเมนเวลา (ข) โดเมนความถี่

รูปที่ 5.12 สัญญาณคลื่นไฟฟ้าหัวใจทางด้านอินพุตของวงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอล ที่ SNR = 25.94 ดีบี กับค่าของ MSE = 165.6

ลำดับต่อมาทำการวัดสัญญาณคลื่นไฟฟ้าหัวใจทางด้านเอาต์พุตของวงจรกรองความถี่ จะได้สัญญาณที่ผ่านการขจัดความถี่รบกวนออกแล้วดังรูปที่ 5.13(ก) และนำค่าที่วัดได้ทำการวิเคราะห์หา

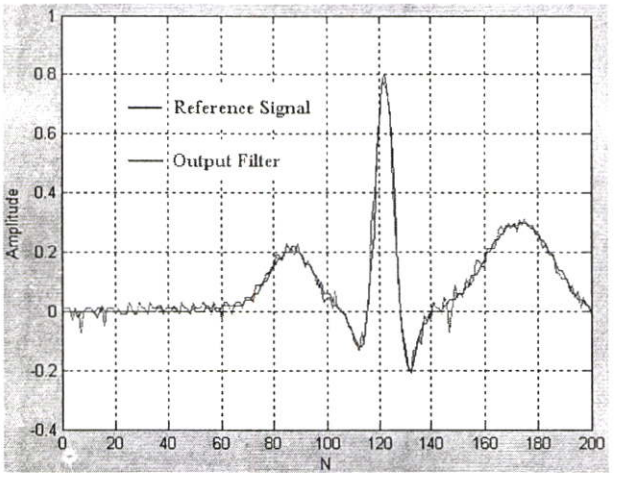
โดเมนทางความถี่ โดยใช้โปรแกรม MATLAB ทำการวิเคราะห์สัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการขจัดสัญญาณรบกวนออกแล้ว จะได้โดเมนทางความถี่ของสัญญาณคลื่นไฟฟ้าหัวใจทางด้านเอาต์พุตของวงจรกรองความถี่ ดังแสดงในรูปที่ 5.13(ข) ซึ่งจากรูปจะเห็นว่าขนาดของสัญญาณรบกวนถูกขจัดออกไปจากสัญญาณคลื่นไฟฟ้าหัวใจ



(ก) โดเมนเวลา (ข) โดเมนความถี่

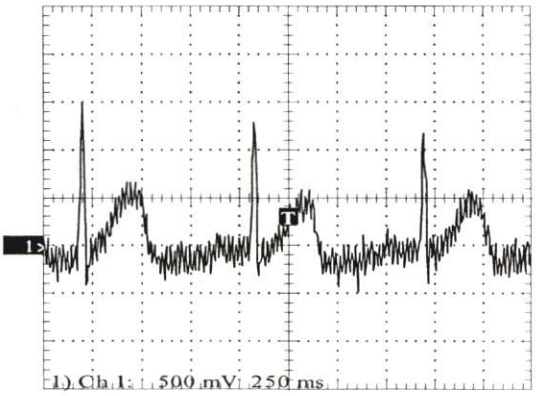
รูปที่ 5.13 สัญญาณคลื่นไฟฟ้าหัวใจด้านเอาต์พุตของวงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอล

นำสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบ และสัญญาณคลื่นไฟฟ้าหัวใจ ที่ผ่านการขจัดสัญญาณรบกวนออกแล้ว คำนวณหาค่าผิดพลาดแบบกำลังสองเฉลี่ย รูปที่ 5.14 แสดงให้เห็นการเปรียบเทียบสัญญาณคลื่นไฟฟ้าหัวใจทั้งสอง มีค่าผิดพลาดแบบกำลังสองเฉลี่ยเท่ากับ 45.82 และวัดขนาดของอัตราส่วนสัญญาณต่อสัญญาณรบกวนเท่ากับ 31.52 ดีบี

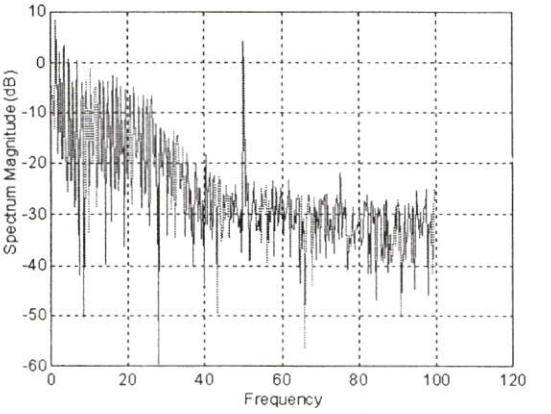


รูปที่ 5.14 ผลการเปรียบเทียบกับสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบกับสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการขจัดสัญญาณรบกวนออกแล้ว

สุดท้ายทำการทดสอบการวัดกับร่างกายจริง โดยเชื่อมต่อวงจรขยายสัญญาณที่มีอัตราขยาย 1000 เท่า เข้ากับวงจรกรองแถบความถี่หยุดผ่าน จากนั้นทำการวัดสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการขยายสัญญาณ จะได้สัญญาณคลื่นไฟฟ้าหัวใจดังแสดงในรูปที่ 5.15 และทำการวัดสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการขจัดสัญญาณรบกวนออกแล้ว จะได้สัญญาณคลื่นไฟฟ้าหัวใจดังแสดงในรูปที่ 5.16

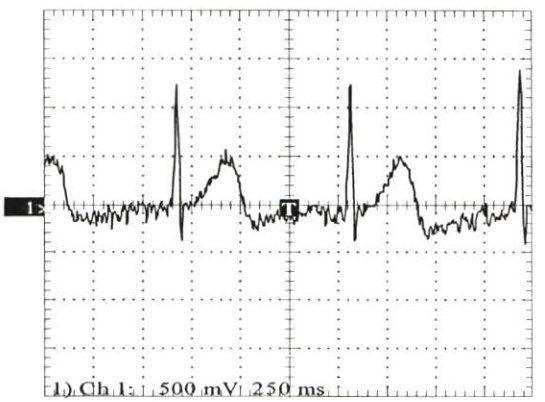


(ก) โดเมนเวลา

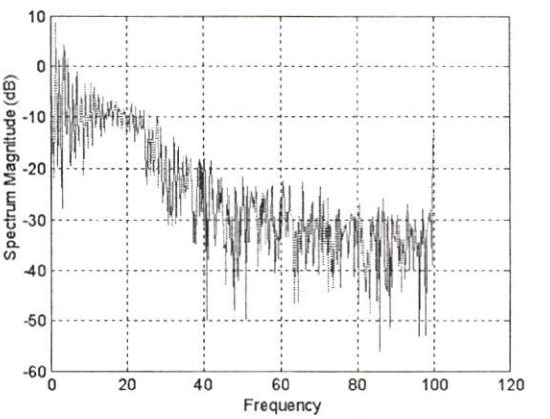


(ข) โดเมนความถี่

รูปที่ 5.15 สัญญาณคลื่นไฟฟ้าหัวใจด้านเอาต์พุตของวงจรขยายสัญญาณจากการวัดกับร่างกายจริง



(ก) โดเมนเวลา



(ข) โดเมนความถี่

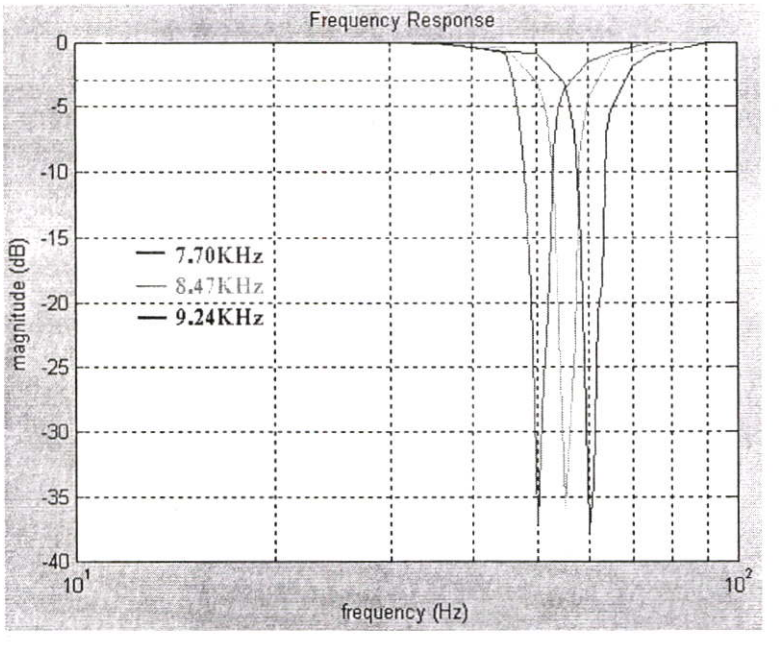
รูปที่ 5.16 สัญญาณคลื่นไฟฟ้าหัวใจเอาต์พุตของวงจรกรองแถบความถี่หยุดผ่านจากการวัดกับร่างกายจริง

5.4 วงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก

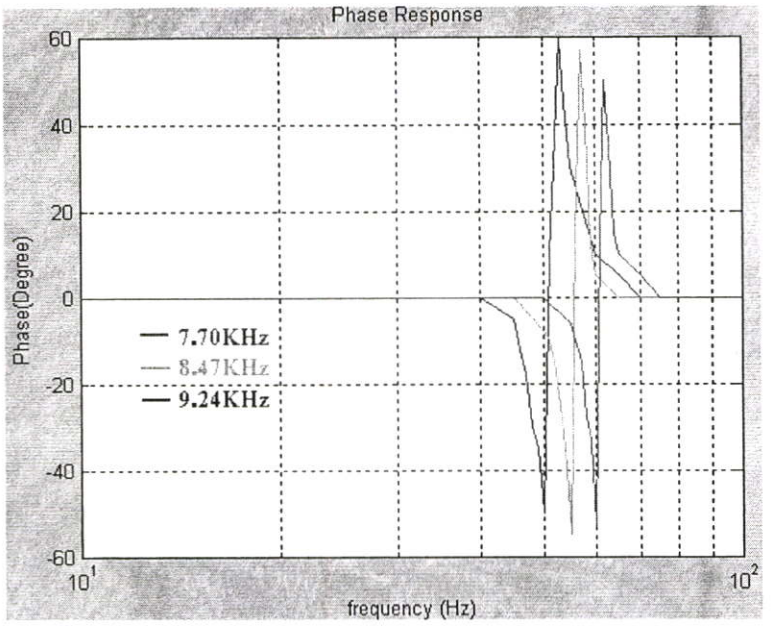
วงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก ในที่นี้นำเสนอวงจรกกรองความถี่ชนิดติดตามขจัดความถี่ของสัญญาณรบกวน การทดสอบวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอกของวิทยานิพนธ์ฉบับนี้ แบ่งออกเป็นสองส่วนคือ ส่วนของวงจรกรองความถี่ชนิดโครงข่ายสวิตช์ประจุที่ใช้สัญญาณนาฬิกาเป็นตัวปรับค่าของความถี่ตัด และส่วนของวงจรสังเคราะห์ความถี่ซึ่งถูกใช้เป็นตัวปรับค่าความถี่ของสัญญาณนาฬิกา ดังรายละเอียดต่อไปนี้

5.4.1 วงจรกรองแถบความถี่หยุดผ่านชนิดโครงข่ายสวิตช์ประจุ

จากคุณสมบัติของวงจรกรองแถบความถี่หยุดผ่านที่ได้กล่าวไว้แล้วในบทที่ 4 กำหนดให้ค่าคุณภาพเท่ากับ 8 ซึ่งสอดคล้องการกำหนดขั้วอินพุตของ Q6 – Q0 เป็นเลขฐานสองเท่ากับ 1111000 ส่วนความถี่ตัดขึ้นอยู่กับค่าอัตราส่วนระหว่างความถี่ของสัญญาณนาฬิกากับค่าของความถี่ตัด กำหนดอัตราส่วนเท่ากับ 154 สอดคล้องการกำหนดขั้วอินพุตของ F4 – F0 เป็นเลขฐานสองเท่ากับ 10001 ป้อนสัญญาณนาฬิกาที่ความถี่ 7.70 กิโลเฮิรตซ์ , 8.47 กิโลเฮิรตซ์ และ 9.24 กิโลเฮิรตซ์ จะได้ผลตอบสนองทางความถี่ทั้งขนาดและเฟส ดังแสดงในรูปที่ 5.17 และ 5.18 ตามลำดับ ซึ่งจะเห็นว่าสัญญาณนาฬิกาที่ความถี่ทั้งสามค่า สอดคล้องกับความถี่ตัดของวงจรกรองแถบความถี่หยุดผ่านเท่ากับ 50 เฮิตร์ซ 55 เฮิรตซ์ และ 60 เฮิรตซ์

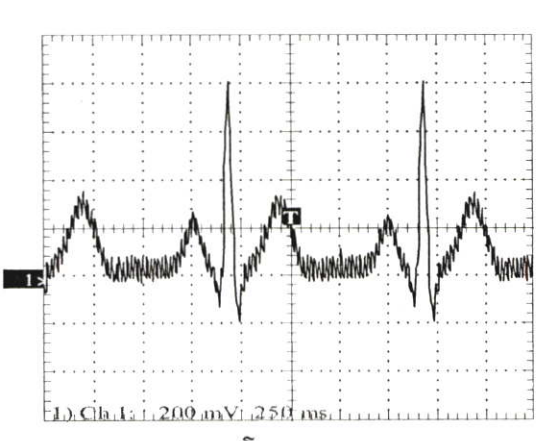


รูปที่ 5.17 ผลตอบสนองทางขนาดของวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก

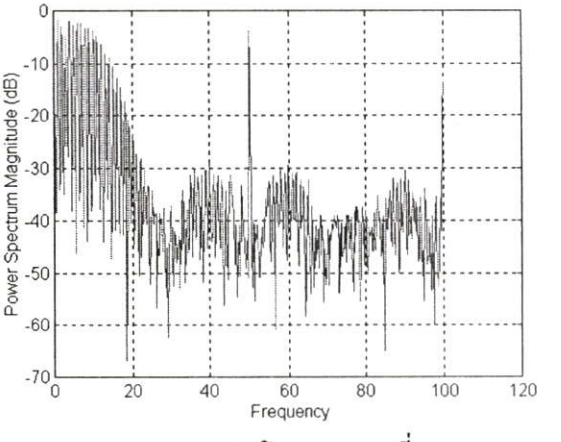


รูปที่ 5.18 ผลตอบสนองทางมุมเฟสของวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก

ทำการป้อนสัญญาณคลื่นไฟฟ้าหัวใจทดสอบให้กับวงจรกรองความถี่ โดยใช้สัญญาณคลื่นไฟฟ้าหัวใจต้นแบบรวมกับสัญญาณรบกวนขนาด 100 มิลลิโวลต์ ที่ความถี่ 50 เฮิรตซ์ , 55 เฮิรตซ์ และ 60 เฮิรตซ์ จะได้สัญญาณคลื่นไฟฟ้าหัวใจทดสอบทั้งในโดเมนเวลา และโดเมนความถี่ ดังรูปที่ 5.19 , 5.20 และ 5.21 ตามลำดับ

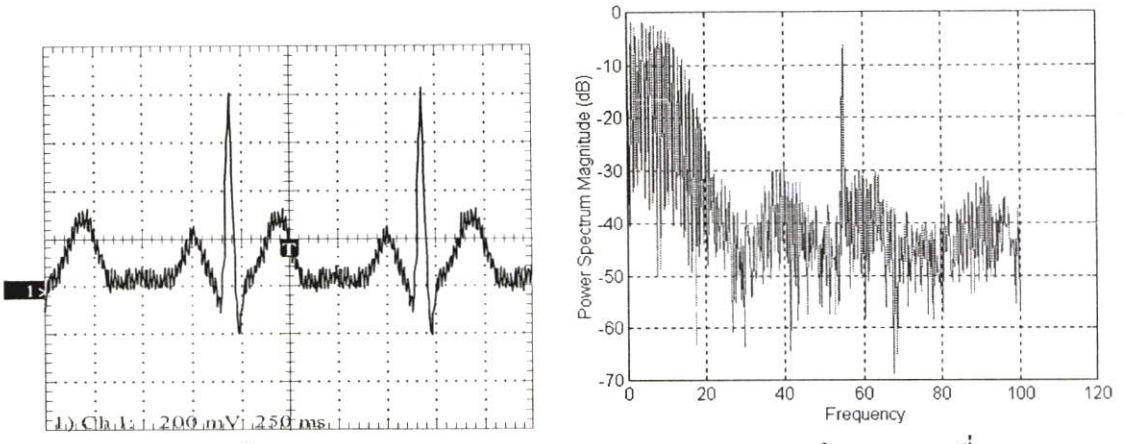


(ก) โดเมนเวลา



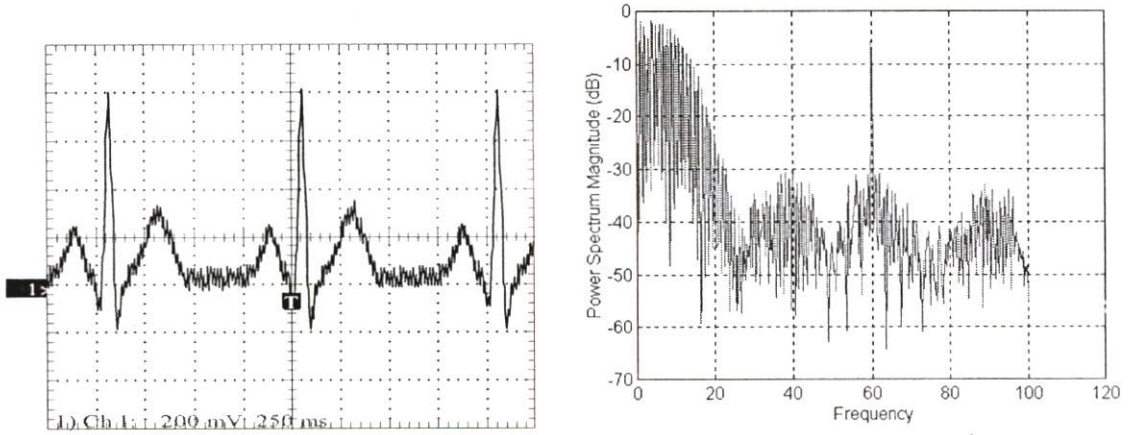
(ข) โดเมนความถี่

รูปที่ 5.19 สัญญาณคลื่นไฟฟ้าหัวใจทดสอบที่มีความถี่รบกวน 50 เฮิรตซ์ และ SNR เท่ากับ 25.94 ดีบี



(ก) โดเมนเวลา (ข) โดเมนความถี่

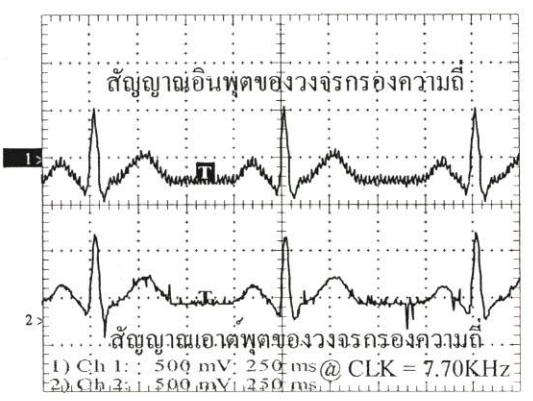
รูปที่ 5.20 สัญญาณคลื่นไฟฟ้าหัวใจทดสอบที่มีความถี่รบกวน 55 เฮิรตซ์ และ SNR เท่ากับ 25.94 ดีบี



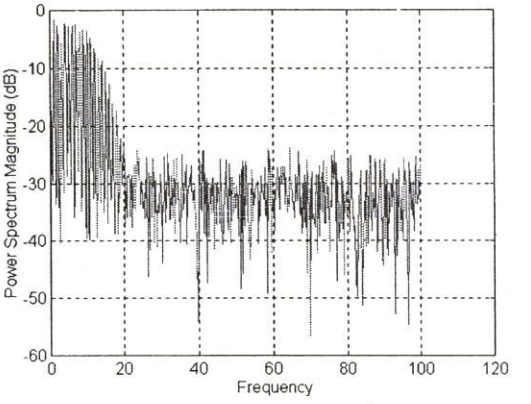
(ก) โดเมนเวลา (ข) โดเมนความถี่

รูปที่ 5.21 สัญญาณคลื่นไฟฟ้าหัวใจทดสอบที่มีความถี่รบกวน 60 เฮิรตซ์ และ SNR เท่ากับ 25.94 ดีบี

ต่อมาทำการป้อนสัญญาณนาฬิกาให้กับวงจรกรองความถี่ โดยใช้ค่าความถี่ 7.70 กิโลเฮิรตซ์ , 8.47 กิโลเฮิรตซ์ และ 9.24 กิโลเฮิรตซ์ ตามลำดับ จะได้สัญญาณคลื่นไฟฟ้าหัวใจทางด้านเอาต์พุตที่ผ่านการกรองความถี่ ในโดเมนความถี่และโดเมนเวลา ที่ความถี่รบกวน 50 เฮิรตซ์ , 55 เฮิรตซ์ และ 60 เฮิรตซ์ ดังแสดงในรูปที่ 5.22 , 5.23 และ 5.24 ตามลำดับ



(ก) โดเมนเวลา

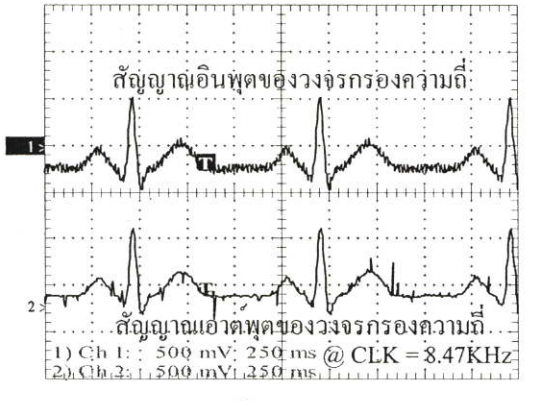


(ข) โดเมนความถี่

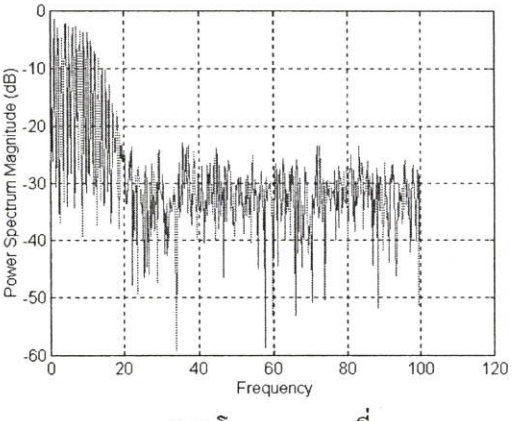
รูปที่ 5.22 สัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการกรองที่ความถี่รบกวน 50 เฮิรตซ์

(ก) สัญญาณด้านบนเป็นสัญญาณคลื่นไฟฟ้าหัวใจทดสอบและรูปสัญญาณด้านล่างเป็นสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการกรองความถี่

(ข) สัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการกรองความถี่ในโดเมนความถี่



(ก) โดเมนเวลา

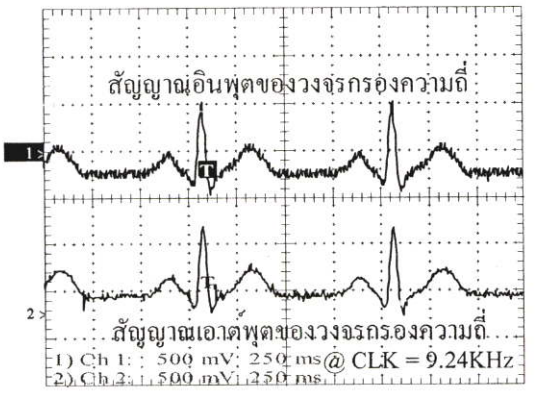


(ข) โดเมนความถี่

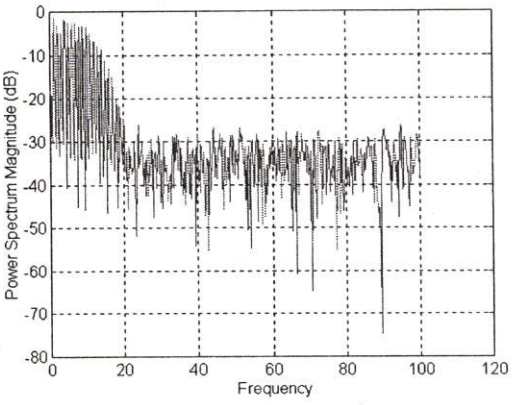
รูปที่ 5.23 สัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการกรองที่ความถี่รบกวน 55 เฮิรตซ์

(ก) สัญญาณด้านบนเป็นสัญญาณคลื่นไฟฟ้าหัวใจทดสอบและรูปสัญญาณด้านล่างเป็นสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการกรองความถี่

(ข) สัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการกรองความถี่ในโดเมนความถี่



(ก) โดเมนเวลา



(ข) โดเมนความถี่

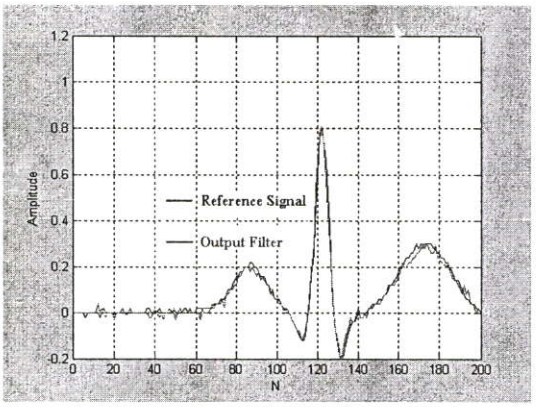
รูปที่ 5.24 สัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการกรองที่ความถี่รบกวน 60 เฮิรตซ์

(ก) สัญญาณด้านบนเป็นสัญญาณคลื่นไฟฟ้าหัวใจทดสอบและรูปสัญญาณด้านล่างเป็นสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการกรองความถี่

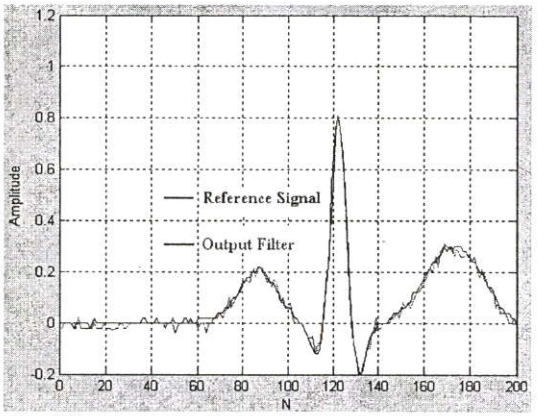
(ข) สัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการกรองความถี่ในโดเมนความถี่

สัญญาณคลื่นไฟฟ้าหัวใจเพื่อทดสอบ ของรูปที่ 5.22(ก) ถึง 5.24 (ก) สัญญาณด้านบนซึ่งเป็นอินพุตของวงจรกรองความถี่ ที่มี SNR เท่ากับ 25.94 ดีบี ส่วนสัญญาณด้านล่างคือสัญญาณที่ผ่านการขจัดสัญญาณรบกวนออกแล้ว สังเกตุได้ว่าวงจรกรองแถบความถี่หยุดผ่านสามารถขจัดสัญญาณรบกวนที่ความถี่ 50 เฮิรตซ์ , 55 เฮิรตซ์ และ 60 เฮิรตซ์ ออกไปได้ เมื่อนำสัญญาณที่ผ่านการขจัดความถี่ออกแล้ววิเคราะห์องค์ประกอบในโดเมนเวลาไม่พบขนาดของสัญญาณรบกวน ตามรูปที่ 5.22(ข) ถึง 5.24 (ข)

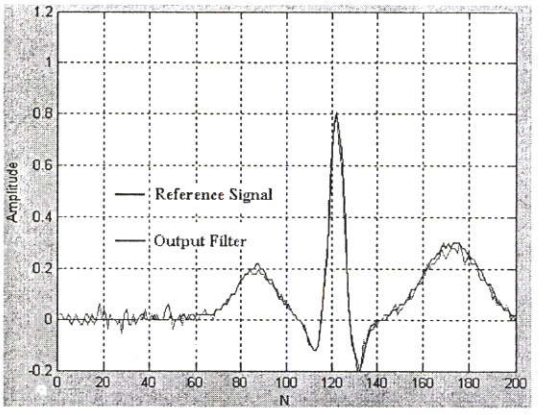
ลำดับถัดไป ทำการคำนวณหาค่าผิดพลาดแบบกำลังสองเฉลี่ย โดยนำสัญญาณคลื่นไฟฟ้าหัวใจทางด้านเอาต์พุตของวงจรกรองความถี่ โดยที่ป้อนความถี่สัญญาณนาฬิกา 7.70 กิโลเฮิรตซ์ , 8.47 กิโลเฮิรตซ์ และ 9.24 กิโลเฮิรตซ์ เปรียบเทียบกับสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบ จะได้ค่าผิดพลาดกำลังสองเฉลี่ยเท่ากับ 44.59 , 48.37 กับ 50.22 ตามลำดับ และทำการวัดขนาดของอัตราส่วนสัญญาณต่อสัญญาณรบกวน มีค่าเท่ากับ 31.64 ดีบี , 31.28 ดีบี กับ 31.12 ดีบี ตามลำดับ รูปที่ 5.25 , 5.26 และ 5.27 แสดงให้เห็นการเปรียบเทียบสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบ กับสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการขจัดสัญญาณรบกวนออกแล้วในโดเมนเวลา



รูปที่ 5.25 ผลการเปรียบเทียบสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบที่ผ่านการขจัดสัญญาณรบกวนออกแล้ว ที่สัญญาณนาฬิกา 7.70 กิโลเฮิรตซ์



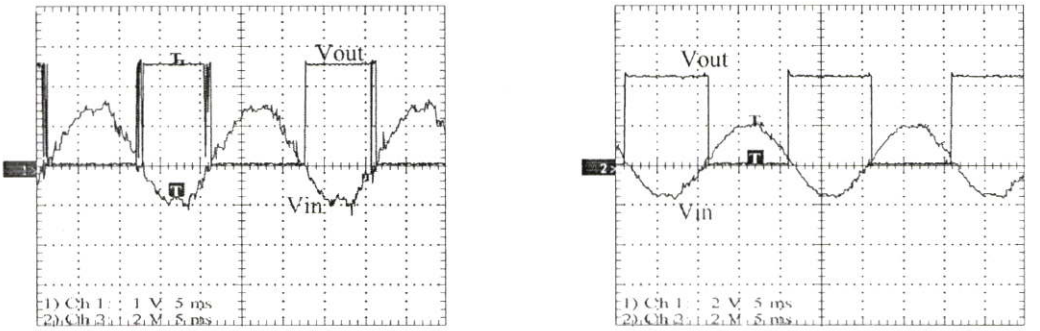
รูปที่ 5.26 ผลการเปรียบเทียบสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบที่ผ่านการขจัดสัญญาณรบกวนออกแล้ว ที่สัญญาณนาฬิกา 8.47 กิโลเฮิรตซ์



รูปที่ 5.27 ผลการเปรียบเทียบสัญญาณคลื่นไฟฟ้าหัวใจต้นแบบที่ผ่านการขจัดสัญญาณรบกวนออกแล้ว ที่สัญญาณนาฬิกา 9.24 กิโลเฮิรตซ์

5.4.2 วงจรสังเคราะห์ความถี่

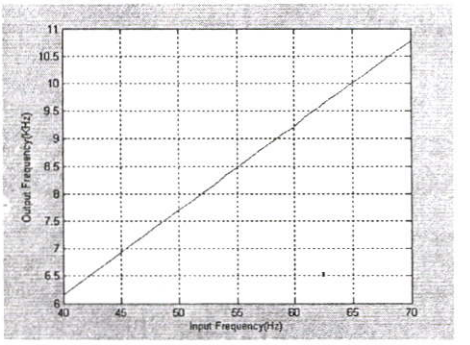
สัญญาณรบกวนจากสายส่งกำลังถูกปรับเปลี่ยนเป็นสัญญาณรูปสี่เหลี่ยม โดยใช้วงจรเปรียบเทียบแรงดันแบบฮิสเทอร์ริซิส จากนั้นป้อนผ่านไปยังขั้วอินพุตของวงจรสังเคราะห์ความถี่ ทำหน้าที่ทวีคูณความถี่ของสัญญาณรบกวนจากสายส่งกำลัง โดยที่อัตราส่วนทวีคูรเท่ากับ 154 เท่า กรณีที่สัญญาณรบกวนจากสายส่งกำลังมีความถี่ 50 เฮิรตซ์ จะถูกทวีความถี่เป็น 7.70 กิโลเฮิรตซ์ ทำนองเดียวกันถ้าสัญญาณรบกวนจากสายส่งกำลังมีความถี่ 60 เฮิรตซ์ จะถูกทวีความถี่เป็น 9.24 กิโลเฮิรตซ์ รูปที่ 5.28 แสดงให้เห็นผลทดลองวัดสัญญาณที่เหนี่ยวนำเข้ามาทางอินพุต และสัญญาณทางด้านเอาต์พุตของวงจรเปรียบเทียบแรงดัน



(ก) วงจรเปรียบเทียบแรงดันแบบพื้นฐาน (ข) วงจรเปรียบเทียบแรงดันแบบฮิสเทอร์ริซิส

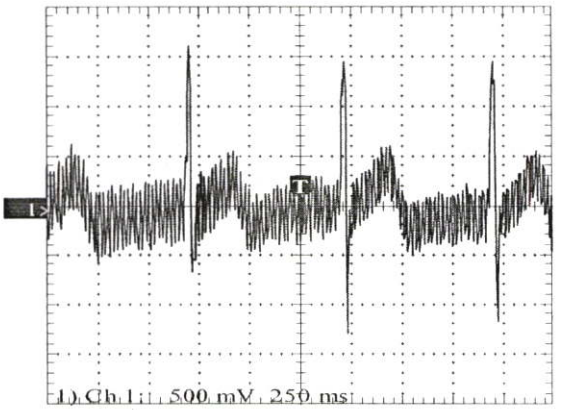
รูปที่ 5.28 สัญญาณที่เหนี่ยวนำเข้ามาทางอินพุตและสัญญาณทางด้านเอาต์พุตของวงจรเปรียบเทียบแรงดันทั้งสองแบบ

ทำการทดสอบคุณสมบัติของวงจรสังเคราะห์ความถี่ โดยทำการป้อนสัญญาณรูปสี่เหลี่ยมจากเครื่องกำเนิดสัญญาณ HP33120A ไปยังขั้วอินพุตของวงจรสังเคราะห์ความถี่ และทำการวัดความถี่ของสัญญาณทางด้านเอาต์พุต จะได้ความถี่ทวีเป็น 154 เท่าของความถี่อินพุต ทำการแสดงความสัมพันธ์ระหว่างความถี่อินพุตกับค่าความถี่ทางเอาต์พุต ตั้งแต่ 40 เฮิรตซ์ จนถึง 70 เฮิรตซ์ จะได้กราฟแสดงความสัมพันธ์ดังรูปที่ 5.29

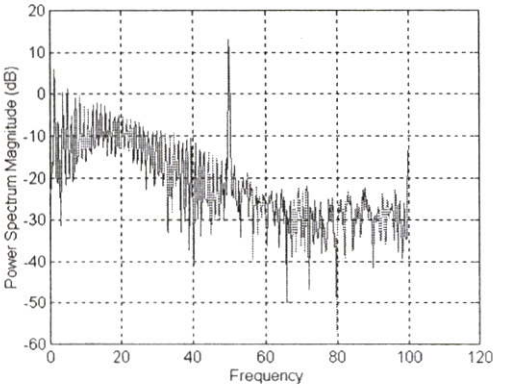


รูปที่ 5.29 ความสัมพันธ์ระหว่างค่าความถี่อินพุตกับค่าความถี่เอาต์พุต

สุดท้ายทำการทดสอบการวัดสัญญาณคลื่นไฟฟ้าหัวใจกับร่างกายจริง โดยเชื่อมต่อวงจรขยายสัญญาณที่มีอัตราขยาย 1000 เท่า เข้ากับวงจรกรองแถบความถี่หยุดผ่าน จากนั้นทำการวัดสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการขยายสัญญาณ จะได้สัญญาณคลื่นไฟฟ้าหัวใจ ดังแสดงในรูปที่ 5.30 และทำการวัดสัญญาณคลื่นไฟฟ้าหัวใจ ที่ผ่านการขจัดสัญญาณรบกวนออกแล้ว จะได้ลักษณะของสัญญาณคลื่นไฟฟ้าหัวใจ ดังแสดงในรูปที่ 5.31

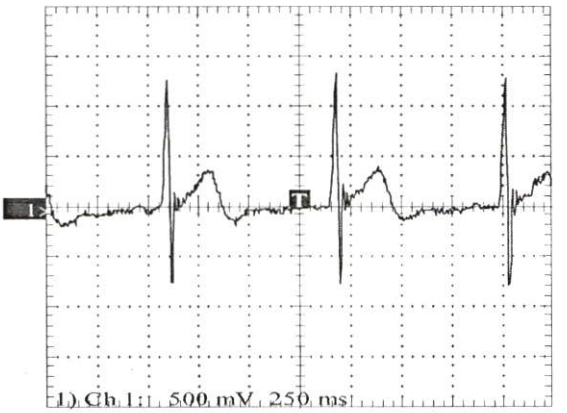


(ก) โดเมนเวลา

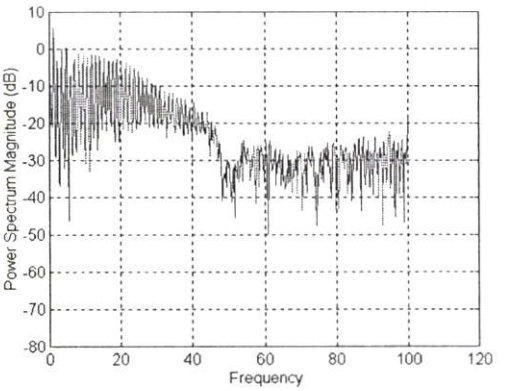


(ข) โดเมนความถี่

รูปที่ 5.30 สัญญาณคลื่นไฟฟ้าหัวใจด้านเอาต์พุตของวงจรขยายสัญญาณจากการวัดกับร่างกายจริง



(ก) โดเมนเวลา



(ข) โดเมนความถี่

รูปที่ 5.31 สัญญาณคลื่นไฟฟ้าหัวใจเอาต์พุตของวงจรกรองแถบความถี่หยุดผ่านจากการวัดกับร่างกายจริง

บทที่ 6
สรุปผลการวิจัย และข้อเสนอแนะ

วงจรกรองแถบความถี่หยุดผ่าน สำหรับการวัดสัญญาณคลื่นไฟฟ้าหัวใจนั้น จะต้องมีการขจัดความถี่รบกวนที่เกิดจากสายส่งกำลัง ออกไปจากสัญญาณคลื่นไฟฟ้าหัวใจได้หมดไป อีกทั้งต้องมีคุณลักษณะของวงจรที่ดีคือ ไม่ส่งผลกระทบทางขนาดหรือแรงดันไฟฟ้าของคลื่นไฟฟ้าหัวใจ และไม่ส่งผลกระทบต่อมุมเฟสของสัญญาณคลื่นไฟฟ้าหัวใจ

ในหลายบทความวิจัยได้นำเสนอวงจรกรองความถี่หยุดผ่านแบบดิจิตอล เพราะมีความแม่นยำทางความถี่ตัด โดยเฉพาะชนิดผลตอบสนองอิมพัลส์จำนวนจำกัด เพราะมีคุณสมบัติของมุมเฟสเปลี่ยนแปลงแบบเชิงเส้น แต่วิทยานิพนธ์ฉบับนี้ได้แสดงให้เห็นแล้วว่า การใช้วงจรแถบความถี่หยุดผ่านแบบดิจิตอล ชนิดผลตอบสนองอิมพัลส์จำนวนจำกัดนี้ ไม่เหมาะสมกับการวัดสัญญาณคลื่นไฟฟ้าหัวใจ เพราะส่งผลกระทบต่อมุมเฟสของสัญญาณคลื่นไฟฟ้าหัวใจ จากที่ได้นำเสนอไปแล้วจะเห็นว่าจากแบบจำลองโดยวิธีหน้าต่างของ แบล็กแมน , แฮมมิง , ฮานนิง , ตุกคีย์ และ ไคเซอร์ ต่างก็มีผลตอบสนองทางมุมเฟสของช่วงความถี่แถบผ่านเปลี่ยนแปลงแบบเชิงเส้นต่อเนื่องตลอดเวลา ย่อมส่งผลถึงองค์ประกอบของสัญญาณคลื่นไฟฟ้าหัวใจได้ ถึงแม้ว่าคุณสมบัติของวงจรกรองความถี่ สามารถบังคับความกว้างของแถบเปลี่ยนได้ แต่สรุปโดยรวมแล้วส่งผลเสียมากกว่า

สำหรับวงจรกรองความถี่ดิจิตอล ชนิดผลตอบสนองอิมพัลส์จำนวนไม่จำกัด ซึ่งมีแบบฟังก์ชันการถ่ายโอนเสมือนกับวงจรกรองความถี่แบบแอนะลอก จะมีความเหมาะสมกับการวัดสัญญาณคลื่นไฟฟ้าหัวใจ เพราะมีผลตอบสนองทางมุมเฟสในช่วงแถบความถี่ผ่านที่คงที่ศูนย์องศา แต่ต้องคำนึงถึงผลกระทบทางด้านขนาดหรือแรงดันของสัญญาณไฟฟ้าของผลตอบสนองทางความถี่ โดยวิทยานิพนธ์ฉบับนี้ได้จำลองผลตอบสนองทางความถี่ของฟังก์ชันถ่ายโอนแบบบัตเตอร์เวิร์ธ , เชบิเชพ และ อิลลิปติก ตามที่ได้เสนอไปแล้ว จะเห็นว่าฟังก์ชันการถ่ายโอนของบัตเตอร์เวิร์ธ มีความเหมาะสมกับการวัดสัญญาณคลื่นไฟฟ้าหัวใจมากที่สุด เพราะมีผลตอบสนองทางขนาดในช่วงแถบความถี่ผ่านที่ราบเรียบที่สุด ส่วนฟังก์ชันการถ่ายโอนอื่นๆ จะมีแรงดันกระเพื่อมในช่วงของแถบความถี่ผ่าน

ทางด้านวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก สำหรับการวัดสัญญาณคลื่นไฟฟ้าหัวใจนั้น งานวิจัยทั่วๆไป ไม่เป็นที่นิยมเพราะขาดความแม่นยำทางด้านความถี่ตัด แต่การใช้โครงสร้างสวิตช์ตัวเก็บประจุในวงจรกรองความถี่ กลับมีความแม่นยำไม่ด้อยไปกว่าวงจรกรองความถี่แบบดิจิตอล จากการเปรียบเทียบในผลการทดลอง จะเห็นว่ามีค่าผิดพลาดแบบเฉลี่ยกำลังสองเพียง 44.59 เมื่อเปรียบเทียบกับแบบดิจิตอล ซึ่งมีค่าผิดพลาด 45.82 ดังตารางที่ 6.1

ตารางที่ 6.1 ค่าผิดพลาดกับค่าอัตราสัญญาณต่อสัญญาณรบกวน

PARAMETER	*Digital Filter*	*Analog Switched-Capacitor Filter*		
		@ CLK = 7.70KHz	@ CLK = 8.47KHz	@ CLK = 9.24KHz
MSE	45.82	44.59	48.37	50.22
SNR	31.53 ดีบี	31.64 ดีบี	31.28 ดีบี	31.12 ดีบี

6.1 วงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอล

การออกแบบวงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอลของวิทยานิพนธ์ฉบับนี้เลือกออกแบบจากชนิดผลตอบสนองอิมพัลส์จำนวนไม่จำกัด ใช้โครงสร้างของฟังก์ชันการถ่ายโอนแบบบัตเตอร์เวิร์ธอันดับที่สอง โดยการสร้างแบบจำลองด้วยโปรแกรม MATLAB เพื่อหาผลตอบสนองทางความถี่กับผลตอบสนองทางมุมเฟส พร้อมกับการคำนวณหาค่าสัมประสิทธิ์ของวงจร ลำดับต่อมาทำการออกแบบวงจรกรองความถี่โดยใช้ภาษาวีเอชดีแอลที่มีโครงสร้างแบบทางตรงแบบที่หนึ่ง

หลังจากออกแบบวงจรกรองความถี่จากภาษาวีเอชดีแอลแล้ว ทำการโปรแกรมลงในชิพวงจรรวมเอฟพีจีเอ ซึ่งวิทยานิพนธ์ฉบับนี้ ใช้เบอร์ XC2S100TQ144-5C ตระกูล Spartan-II ของบริษัท Xilinx เชื่อมต่อกับวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณดิจิตอลขนาด 8 บิต และวงจรแปลงสัญญาณดิจิตอลไปเป็นสัญญาณแอนะลอกขนาด 8 บิต เมื่อทำการทดสอบวงจรเพื่อหาผลตอบสนองทางความถี่กับผลตอบสนองทางมุมเฟส ได้ผลลัพธ์ตามที่ได้ออกแบบไว้ ต่อมาทำการทดสอบขจัดสัญญาณรบกวน โดยสร้างสัญญาณรบกวนที่ความถี่ 50 เฮิรตซ์ขนาดแรงดัน 100 มิลลิโวลต์ ผสมกับสัญญาณคลื่นไฟฟ้าหัวใจตามที่ได้นำเสนอวิธีการในบทที่ 5 แล้วนั้น ปรากฏว่าจากการวัดสัญญาณทางด้านเอาต์พุตทั้งในโดเมนเวลากับโดเมนความถี่ วงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอลที่ได้ออกแบบนี้ สามารถขจัดสัญญาณรบกวนได้ตามคุณสมบัติที่ได้ออกแบบไว้

เมื่อทำการวัดค่าผิดพลาดแบบกำลังสองเฉลี่ย โดยนำสัญญาณคลื่นไฟฟ้าหัวใจก่อนการผสมกับสัญญาณรบกวน เปรียบเทียบกับสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการขจัดสัญญาณรบกวนออกไปแล้ว ปรากฏว่ามีค่าประมาณ 45.82 และมีค่าขนาดของอัตราส่วนสัญญาณต่อสัญญาณรบกวนเท่ากับ 31.52 ดีบี

6.2 วงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก

การออกแบบวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก จากการประยุกต์ใช้โครงข่ายสวิตช์ตัวเก็บประจุสำหรับวงจรกรองความถี่ วิทยานิพนธ์ฉบับนี้เลือกใช้วงจรรวมเบอร์ MAX263 ซึ่งมีข้อดีกว่าการใช้วงจรรวมเบอร์ MF10 ที่ออกแบบได้ง่ายกว่า ไม่ต้องมีอุปกรณ์อื่นๆเพิ่มเติม การกำหนด

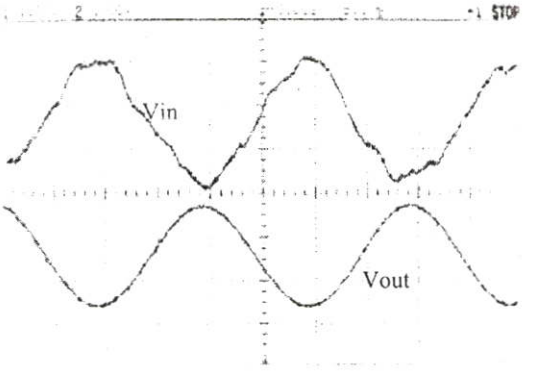
ค่าความถี่ตัดแบบคงที่ กระทำได้โดยการกำหนดขั้วอินพุต F4 – F0 กับความถี่ของสัญญาณนาฬิกาให้มีค่าคงตัว ในกรณีที่ต้องการให้ความถี่ตัดเปลี่ยนค่าตามความถี่ของสัญญาณรบกวน กระทำได้โดยกำหนดให้ขั้วอินพุต F4 – F0 มีค่าคงตัว ส่วนความถี่ของสัญญาณนาฬิกาให้สามารถปรับเปลี่ยนค่าได้ ซึ่งวิทยานิพนธ์ฉบับนี้ได้กำหนดให้ขั้วอินพุต F4 – F0 มีค่าคงตัวของ N เท่ากับ 17 หรือมีค่าในเลขฐานสองเท่ากับ 10001

สำหรับความกว้างของแถบความถี่หยุด ขึ้นอยู่กับค่าคุณภาพของวงจรกรองความถี่ สามารถกำหนดได้จากขั้วอินพุต Q6 – Q0 ให้มีค่าคงตัวที่ N เท่ากับ หรือมีค่าในเลขฐานสองเท่ากับ 111100 เมื่อการออกแบบวงจรกรองแถบความถี่หยุดผ่านของวิทยานิพนธ์ฉบับนี้ สามารถติดตามขจัดความถี่ของสัญญาณรบกวนได้ ดังนั้นความถี่ของสัญญาณนาฬิกาของวงจรรวมโครงข่ายสวิตช์ตัวเก็บประจุ จะต้องปรับเปลี่ยนค่าของความถี่ได้ โดยความถี่ของสัญญาณนาฬิกาที่ปรับเปลี่ยนค่านี้ กำหนดได้จากวงจรสังเคราะห์ความถี่ ที่ทำหน้าที่ทวีความถี่จำนวน 154 เท่าของความถี่รบกวน

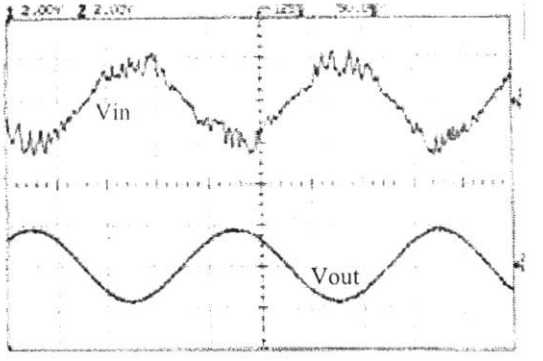
ลำดับต่อมาทำการทดสอบกับการขจัดสัญญาณรบกวน โดยสร้างสัญญาณรบกวนที่ความถี่ 50 เฮิรตซ์ขนาดแรงดัน 100 มิลลิโวลต์ ผสมกับสัญญาณคลื่นไฟฟ้าหัวใจ ปรากฏว่าจากการวัดสัญญาณทางด้านเอาต์พุตทั้งในโดเมนเวลากับโดเมนความถี่ วงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอกที่ได้ออกแบบนี้ สามารถขจัดสัญญาณรบกวนได้ตามคุณสมบัติที่ได้ออกแบบไว้

เมื่อทำการวัดค่าผิดพลาดแบบกำลังสองเฉลี่ย โดยนำสัญญาณคลื่นไฟฟ้าหัวใจก่อนการผสมกับสัญญาณรบกวน เปรียบเทียบกับสัญญาณคลื่นไฟฟ้าหัวใจที่ผ่านการขจัดสัญญาณรบกวนออกไปแล้ว ปรากฏว่ามีค่าประมาณ 44.59 กับค่าขนาดของอัตราส่วนสัญญาณต่อสัญญาณรบกวน เท่ากับ 31.64 ดีบี

จากการพิจารณาทางด้านสมรรถนะ ของวงจรกรองแถบความถี่หยุดผ่านแบบปรับตัวได้ของวิธีการ ASRF ดังรูปที่ 6.1(ก) กับวิธีการ S-PFD ดังรูปที่ 6.1(ข) งานวิจัยดังกล่าวมิได้ระบุถึงอัตราของการลดทอนสัญญาณรบกวน และค่าคุณภาพของวงจรกรองความถี่ ทำให้ไม่สามารถแสดงถึงสมรรถนะอย่างสมบูรณ์ได้ แต่จากการสังเกตุผลการทดลองของงานวิจัยทั้งสองวิธีการ พบว่าวิธีการ ASRF มีอัตราการลดทอนสัญญาณรบกวนได้น้อย อีกทั้งมุมเฟสตรงกันข้ามกับสัญญาณทางอินพุต -180 องศา ส่วนวิธีการ S-PFD จะมีอัตราการลดทอนสัญญาณได้ดีกว่า แต่ก็มีมุมเฟสตรงกันข้ามกับสัญญาณทางอินพุต -180 องศา เช่นกัน ดังนั้นการนำไปใช้ในการวัดคลื่นไฟฟ้าหัวใจ จำเป็นต้องผ่านวงจรปรับแต่งสัญญาณอีกครั้ง เมื่อเปรียบเทียบผลการทดลองกับแบบลูฟเปิดดังรูปที่ 6.2 เห็นได้ว่าโครงสร้างแบบลูฟเปิดของวิทยานิพนธ์ฉบับนี้มีอัตราการขจัดสัญญาณรบกวนได้ดี และมีมุมเฟสทางเอาต์พุตเป็นศูนย์องศา

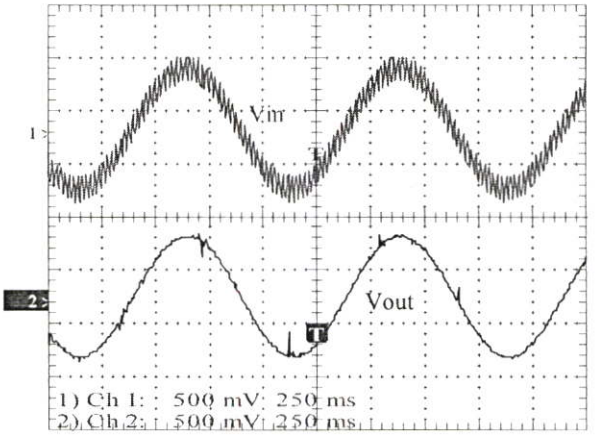


(ก) วิธีการ ASRF[10]



(ข) S-PFD[11]

รูปที่ 6.1 ผลการทดลองของวงจรกรองแถบความถี่หยุดผ่านแบบปรับตัวของวิธีการ ASRF กับ S-PFD



รูปที่ 6.2 ผลการทดลองของวงจรกรองแถบความถี่หยุดผ่านแบบลูฟเปิด

6.3 วงจรขยายสัญญาณแบบแยกโดด

การวัดสัญญาณคลื่นไฟฟ้าหัวใจแบบสองขั้ว(bipolar lead) หรือการวัดสัญญาณแบบลิมบ์หลีด(limb Lead) จะต้องออกแบบวงจรขยายแบบผลต่าง ซึ่งวงจรขยายสัญญาณผลต่างที่ดี นิยมใช้วงจรขยายอินสทรูเมนเตชัน เพราะมีความต้านทานที่อินพุตสูงมาก และมีอัตราขจัดแบบร่วม(Common Mode Rejection Ratio ; CMRR) ที่ดี และสามารถปรับอัตราขยายสัญญาณได้ง่าย แต่สำหรับการวัดสัญญาณคลื่นไฟฟ้าหัวใจ จะต้องแยกสัญญาณออกจากกันระหว่างภาคอินพุตกับภาคขยายสัญญาณด้วยวงจรรวมแบบเดี่ยวชนิดขยายสัญญาณแบบแยกโดด ซึ่งปัจจุบันได้หยุดการผลิต แต่วิทยานิพนธ์ฉบับนี้ได้นำเสนอการขยายสัญญาณผลต่างชนิดแยกโดดสัญญาณ โดยใช้วงจรขยายสัญญาณอินสทรูเมนเตชันกับวงจรขยายผ่านแสง ตามที่ได้เสนอไปแล้วจะเห็นว่า จากผลการจำลองกับการทดลองจริง สามารถนำไปประยุกต์ใช้งานได้

6.4 ข้อเสนอแนะกับการพัฒนา

จากการศึกษาและออกแบบวงจรกรองแถบความถี่หยุดผ่าน ทั้งแบบดิจิตอลและแบบแอนะลอก จะเห็นว่าคุณภาพของวงจรทั้งสองมีผลตอบสนองทางความถี่ , ผลตอบสนองทางมุมเฟส และการขจัดความถี่ของขจัดสัญญาณใกล้เคียงกันมาก เมื่อเปรียบเทียบทางด้านการใช้ทรัพยากร พบว่าวงจรกรองแบบดิจิตอลจะใช้ทรัพยากรมาก วงจรจะมีขนาดใหญ่ และมีราคาแพง

ทางด้านการพัฒนา สำหรับวงจรกรองแถบความถี่หยุดผ่านแบบดิจิตอล จะเห็นว่าวิทยานิพนธ์ฉบับนี้ใช้การคำนวณหาสัมประสิทธิ์จากโปรแกรม MATLAB แล้วนำค่าที่ได้ไปออกแบบวงจรกรองความถี่โดยใช้ภาษาวีเอชดีแอล ดังนั้นถ้าทำการคำนวณหาสัมประสิทธิ์ ณ ความถี่ตัดค่าต่างๆ แล้วนำค่าสัมประสิทธิ์ไปกำหนดเป็นข้อมูลตาราง(Look-Up Table ; LUT) จะทำให้วงจรกรองแถบความถี่หยุดผ่าน สามารถติดตามขจัดความถี่ของสัญญาณรบกวน แบบลูฟเปิดได้

ส่วนวงจรกรองแถบความถี่หยุดผ่านแบบแอนะลอก จะเห็นว่ามีโครงสร้างวงจรเล็กกว่าวงจรแบบลูฟปิด และการออกแบบได้พยายามหลีกเลี่ยงการใช้อุปกรณ์แบบเฉื่อยงาน(passive device) ทำให้การพัฒนาเป็นวงจรรวมแบบเดี่ยวเป็นไปได้ไม่ยุ่งยาก อีกทั้งความแม่นยำทางด้านความถี่ตัดก็ไม่ด้อยไปกว่าวงจรกรองความถี่แบบดิจิตอล

เอกสารอ้างอิง

[1] JOHN G.WEBSTER **Medical Instrumentation Application and Design** third edition , USA., JOHN WILEY & SONS,INC.,1998.

[2] Bernard Widrow , Samuel D. Stearns **Adaptive Signal Processing** Englewood , Prentice-Hall , 1985.

[3] Patrick E. McSharry , Gari D. Clifford , Lionel Tarassenko , and Leonard A. Smith ,"A Dynamical Model for Generating Synthetic Electrocardiogram Signals", **IEEE TRANSACTION ON BIOMEDICAL ENGINEERING** , VOL.50 , NO.3 , MARCH 2003. , pp.289-294

[4] สาวิตร์ ตัณฑนุช , "การกำจัดสัญญาณรบกวน 50 เฮิตรซ์ ด้วยวงจรกรองดิจิตอลที่สร้างจากเอฟพีจีเอในเครื่องวัดคลื่นไฟฟ้าหัวใจ" , **การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 24 (EECON-24)** , 22 – 23 พฤศจิกายน 2544. , หน้า 951 – 956

[5] T. Sansaloni , J. Valls , M.J. Canet , M.Martinez , "FPGA-Based Notch Filter for Cancelling 50Hz Noise on ECG" , **XIII Design of Circuits and Integrated Systems Conference (DCISC99)** , Mallorca 1999.

[6] R. Ramos , A. Manuel , G. Olivar , E. Trullols and J. Del Rio , "Application by means of FPGA of an adaptive canceller 50Hz interference in electrocardiography" , **IEEE Instrumentation and Measurement Technology Conference** , Budapest , Hungary , May 21-23 , 2001. , pp.32-37

[7] Phillip A. Regalia **Adaptive IIR Filtering in Signal Processing and Control**" New York , Marcel-Dekker, 1995.

[8] P.Wattanaluk , C.Benjangkaprasert , O.Sangaroon , K.Janchittrapongvej , " New Variable Step-Size Algorithm for Lattice Form Structure Adaptive IIR Notch Filter" , **ECTI-CON 2006 International Conference** , 2006.

[9] Ying-Wen Bai , et.al. "Adjustable 60Hz Noise Reduction by a Notch Filter for ECG Signals" , **IMTC-2004** , Italy , 18 – 20 May 2004. , pp.1706 – 1711.

[10] L.G. Bustamante, R.H.Strandberg and M.A.Soderstrand , "Switched-Capacitor Adaptive Sampling Rate Notch Filter for Enhancement and Elimination of Bandpass Signals" , **Proceeding of the 40th Midwest Symposium on Circuits and Systems** , Volume 2 , 3-6 August 1997. , pp. 1354 – 1357

[11] L.G. Bustamante, M.A.Soderstrand , " High-Range Switched Capacitor Tracking Filter " , **Circuits and Systems(ISCAS '99). Proceedings of the 1999 IEEE International Symposium** on Volume: 2 , IEEE 1999. , pp. 65-68.

[12] D.V.Bhaskar Rao , Sun-Yuan Kung "Adaptive Notch Filtering for the Retrieval of Sinusoids in Noise" , **IEEE Transactions on Acoustics, Speech and Signal Processing** , VOL. ASSP-32, NO. 4, AUGUST 1984. pp.791 – 802.

[13] Jose Silva-Martinez "A PROGRAMMABLE SWITCHED-CAPACITOR FILTER" , **Circuits and Systems, 1994. ISCAS '94., 1994 IEEE International Symposium** , 30 May-2 Jun 1994. pp.727 – 730.

[14] A.Carusone and D.A.Johns "Analogue adaptive filters : Past and Present" , **Proc. IEE-Circuits , Devices , Syst.** , vol. 47 , no. 1 , Feb. 2000. , pp. 82–90.

[15] Miroslav D. Lutovac , Dejan V. Tosic , Brian L. Evans **FILTER DESIGN FOR SIGNAL PROCESSING Using MATLAB and Mathematica** USA. , Prentice Hall Inc. , 2001.

[16] พรชัย ภววงษ์ศักดิ์ **การประมวลผลสัญญาณดิจิตอลเบื้องต้น** กรุงเทพมหานคร , มหาวิทยาลัยเทคโนโลยีมหานคร 2542

[17] สมศักดิ์ ชุมช่วย **การประมวลสัญญาณเชิงเลขเบื้องต้น** กรุงเทพมหานคร , สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , 2545

[18] R.M.MARSTON **OPTOELECTRONICS CIRCUITS MANUAL** Cornwall , Hartnolls Ltd. , 1988.

[19] Francis Christian . " **Isolation Techniques Using Optical Couplers** " [Online]. Available : http://encon.fke.utm.my/nikd/Internet/opto-couplers.pdf , 2006.

[20] Saeed V. Vaseghi **Advanced Digital Signal Processing and Noise Reduction** Second Edition , New York , JOHN WILEY & SONS.LTD , 2001.

[21] B.Williams , Fred J.Taylor **Electronic Filter Design Handbook : LC , Active and Digital Filter** Second Edition , USA. , McGraw-Hill.Inc , 1988.

[22] K.C.Chang **Digital Systems Design with VHDL and Synthesis an Integrated Approach** , California , IEEE Computer Society , 1999.

[23] Zainalabedin Navabi **VHDL Analysis and Modeling of Digital Systems** , Singapore , McGraw-Hill Inc. , 1993.

[24] James R. Armstrong , F. Gail Gray **VHDL Design Representation and Synthesis** Second Edition , USA. , Prentice-Hall Inc. , 2000.

ภาคผนวก

ภาคผนวก ก.

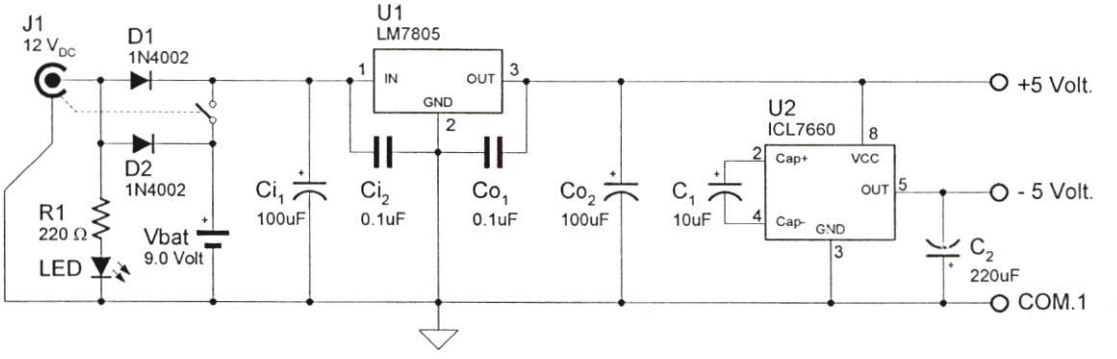
DC POWER SUPPLY

ภาคผนวก ก.
แหล่งจ่ายกำลังไฟตรง

การออกแบบวงจรขยายสัญญาณ สำหรับการวัดสัญญาณคลื่นไฟฟ้าหัวใจ ที่ได้กล่าวมาแล้วในบทที่ 2 เป็นการออกแบบวงจรขยายสัญญาณแบบแยกโดด นั่นคือส่วนของอินพุตของวงจรขยายสัญญาณ จะต้องแยกสัญญาณไฟฟ้ากับขั้วกราวด์ออกจากส่วนของเอาต์พุต เพราะฉะนั้นการออกแบบวงจรแหล่งจ่ายกำลังไฟตรง จำเป็นต้องออกแบบให้มีแหล่งจ่ายกำลังไฟตรงถึงสองแหล่งด้วยกัน อีกทั้งวงจรอิเล็กทรอนิกส์ที่ออกแบบต้องการแหล่งจ่ายกำลังแบบทวิ (Dual Power Supply) หรือมีแรงดันด้านบวก(Positive Voltage) กับแรงดันด้านลบ (Negative Voltage) แต่วิทยานิพนธ์ฉบับนี้ เลือกการออกแบบโดยใช้แหล่งจ่ายกำลังไฟตรงด้านบวกด้านเดียว ผ่านวงจรแปรผันกลับทางไฟตรง (DC to DC Inverter) เพื่อให้ได้แรงดันด้านลบสำหรับวงจรขยายสัญญาณได้ ทำให้การวัดสัญญาณคลื่นไฟฟ้าหัวใจของวิทยานิพนธ์ฉบับนี้ สามารถใช้งานกับแหล่งจ่ายกำลังแบบขั้วเดี่ยว(Single Power Supply) ได้ อีกทั้งได้ออกแบบวงจรเตือนระดับแรงดันไฟต่ำกว่าเกณฑ์ เพื่อรองรับการใช้งานกับแหล่งจ่ายแบบแบตเตอรี ดังการออกแบบดังนี้

ก.1 วงจรแหล่งจ่ายกำลังไฟตรงแบบทวิ

การใช้แหล่งจ่ายกำลังไฟตรงแบบขั้วเดี่ยว จากหม้อแปลงแรงดันไฟตรงขนาดเล็ก ซึ่งมีแรงดันไฟตรงที่ได้กรองสัญญาณแล้ว ผ่านเข้ามายังวงจรคงค่าแรงดัน(Voltage Regulator) เพื่อใช้เป็นแหล่งจ่ายกำลังไฟตรงด้านบวก ในที่นี้เลือกใช้วงจรรวมเบอร์ LM7805 ที่ให้แรงดันไฟฟ้ากระแสตรงด้านบวกเท่ากับ +5 โวลต์ เมื่อทำการแปรผันกลับทางด้านไฟตรงด้านลบ โดยใช้วงจรรวมเบอร์ ICL7660 ทำให้ได้แหล่งจ่ายกำลังไฟตรงแบบทวิ ที่มีทั้งแรงดันไฟฟ้ากระแสตรงด้านบวกมีค่าเท่ากับ +5 โวลต์ กับแรงดันไฟฟ้ากระแสตรงด้านลบมีค่าเท่ากับ -5 โวลต์ ดังแสดงในรูปที่ ก.1



รูปที่ ก.1 แหล่งจ่ายกำลังไฟตรงแบบทวิสำหรับส่วนอินพุตของวงจรขยายสัญญาณ

การแปรผันกลับทางไฟตรงด้านลบ ต้องคำนึงถึงวงจรสมมูลทางแรงดัน โดยเฉพาะค่าความต้านทานทางด้านเอาต์พุต(Output resistance ; R_O) ซึ่งเกิดจากความต้านทานของสวิตช์มอส(MOS Switched) ภายในของวงจรรวม , ความถี่ของการสวิตช์ และค่าความต้านทานสมมูลแบบอนุกรม (equivalent series resistance) ของตัวเก็บประจุ C_1 กับ C_2 โดยสามารถประมาณค่าของความต้านทานทางด้านเอาต์พุตได้ดังนี้

$$R_O \cong 2 \cdot R_{SW} + \frac{1}{0.5 \cdot F_{clk} \cdot C_1} + 5 \cdot ESR_C \qquad \text{(ก.1)}$$

โดยที่

R_O คือ ความต้านทานทางด้านเอาต์พุตของวงจรแปรผันกลับทางไฟตรง

R_{SW} คือ ความต้านทานของสวิตช์มอส

F_{clk} คือ ความถี่ของการสวิตช์

ESR_C คือ ค่าความต้านทานสมมูลแบบอนุกรมของตัวเก็บประจุ

จากคุณสมบัติของวงจรแปรผันกลับทางไฟตรงด้านลบ ที่อุณหภูมิ 25°C จะมีค่าความต้านทานของสวิตช์มอส เท่ากับ 23 โอห์ม และความถี่ของการสวิตช์มีค่าเท่ากับ 10 กิโลเฮิรตซ์ สำหรับค่าความต้านทานสมมูลแบบอนุกรมของตัวเก็บประจุ มีค่าความต้านทานต่ำโดยทั่วๆไปมีค่า ประมาณ 10 โอห์ม เพราะฉะนั้นค่าความต้านทานทางด้านเอาต์พุตของวงจรแปรผันทางไฟตรงด้านลบมีค่าประมาณ 116 โอห์ม

สำหรับค่าแรงดันริปเปิล(ripple) ซึ่งเป็นแรงดันไฟสลับ ที่เกิดจากการสวิตช์ของมอสตามความถี่ของการสวิตช์ สามารถประมาณค่าของแรงดันริปเปิลทางด้านเอาต์พุตได้ดังนี้

$$V_{ripple} = \left(\frac{1}{F_{clk} \cdot C_2} + 2ESR_{C_2} \right) I_{out} \qquad \text{(ก.2)}$$

โดยที่

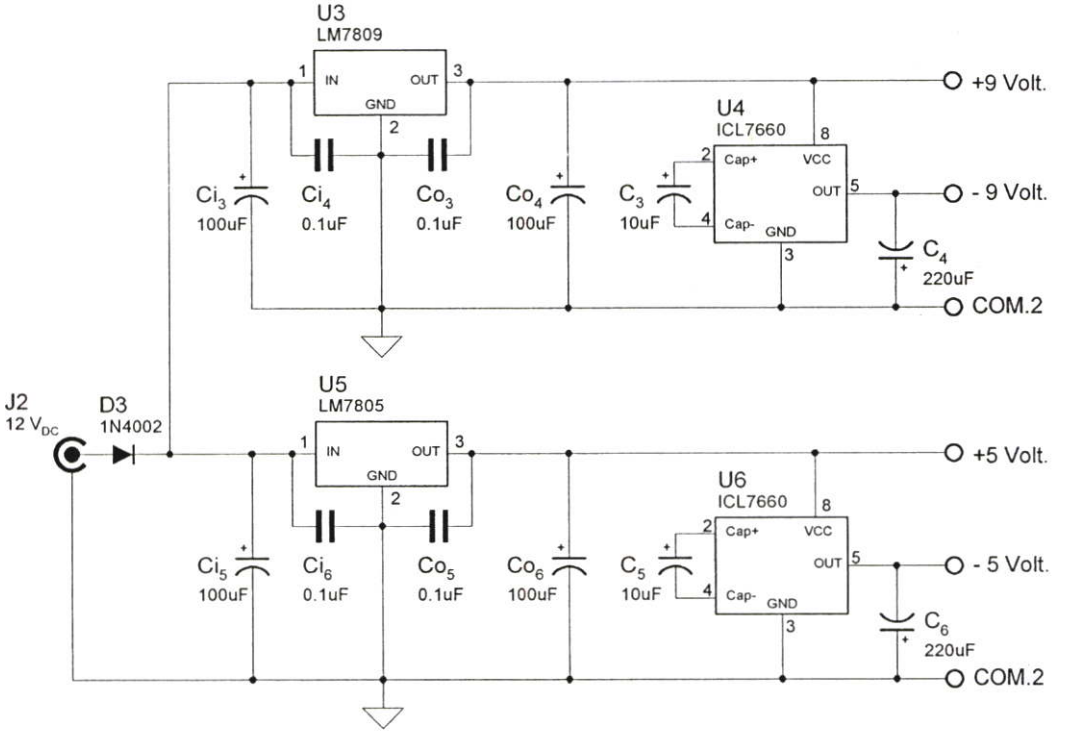
V_{ripple} คือ ค่าแรงดันริปเปิล

F_{clk} คือ ความถี่ของการสวิตช์

ESR_{C2} คือ ค่าความต้านทานสมมูลแบบอนุกรมของตัวเก็บประจุ

I_{out} คือ ค่ากระแสทางขั้วออกของวงจรแปรผันกลับทางไฟตรงด้านลบ

จากวงจรในรูปที่ ก.1 เป็นแหล่งจ่ายกำลังไฟตรงแบบทวิ ที่ใช้สำหรับส่วนอินพุตของวงจรขยายสัญญาณ กรณีที่ใช้กับส่วนเอาต์พุตของวงจรขยายสัญญาณ จะต้องออกแบบแหล่งจ่ายกำลังไฟตรงแบบทวิใหม่ตามรูปที่ ก.2 จะเห็นว่ามีแหล่งจ่ายกำลังไฟตรงแบบทวิสองแหล่งด้วยกันคือค่าแรงดัน ±9 โวลต์ สำหรับส่วนเอาต์พุตของวงจรขยายสัญญาณ กับค่าแรงดัน ±5 โวลต์ที่ใช้กับวงจรรวมโครงข่ายสวิตช์ตัวเก็บประจุ



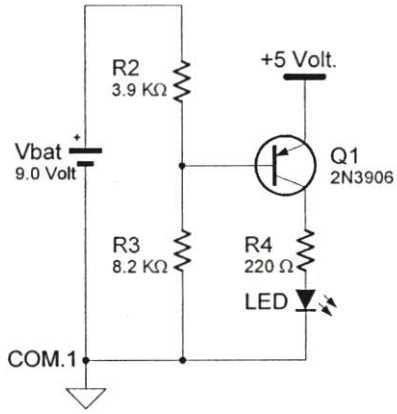
รูปที่ ก.2 แหล่งจ่ายกำลังไฟตรงแบบทวิสำหรับส่วนเอาต์พุตของวงจรขยายสัญญาณ

สำหรับแหล่งจ่ายกำลังไฟตรงของวงจรกรองแถบความถี่หยุดผ่านเชิงเลข มีความต้องการแหล่งจ่ายกำลังแบบทวิเช่นเดียวกันกับวงจรกรองแถบความถี่หยุดผ่านเชิงแอนะลอก เพื่อให้เกิดความสะดวกต่อการดำเนินงาน วิทยานิพนธ์ฉบับนี้จึงเลือกใช้ระดับแรงดันไฟแบบทวิที่ระดับแรงดัน ±5 โวลต์สำหรับส่วนอินพุตของวงจรขยายสัญญาณ กับค่าแรงดัน ±9 โวลต์ สำหรับส่วนเอาต์พุตของวงจรขยายสัญญาณกับวงจรแปลงสัญญาณแอนะลอกเป็นสัญญาณเชิงเลข กับวงจรแปลงสัญญาณเชิงเลขเป็นสัญญาณแอนะลอก

ส่วนแหล่งจ่ายกำลังไฟตรงที่ใช้ในการประมวลผลของชิพวงจรรวมเอฟพีจีเอ เป็นการใช้แหล่งจ่ายแรงดันแบบขั้วเดี่ยว ที่ต้องการระดับแรงดันไฟตรงสองระดับคือ 2.5 โวลต์ กับระดับแรงดัน 3.3 โวลต์ วิทยานิพนธ์ฉบับนี้ได้ออกแบบโดยใช้วงจรรวมคงค่าแรงดันเบอร์ LM317 ตามวงจรรวมในรูปที่ 3.11

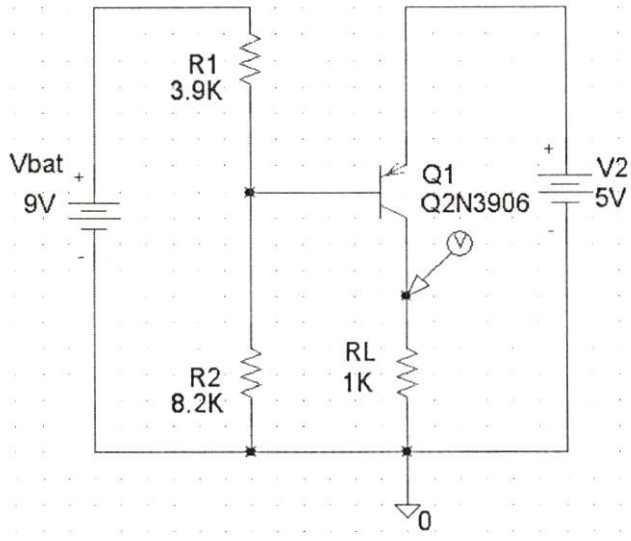
ก.2 วงจรตรวจจับระดับแรงดันต่ำกว่าเกณฑ์(Low Voltage Detector)

การวัดสัญญาณคลื่นไฟฟ้าหัวใจเป็นสิ่งที่สำคัญต่อชีวิตผู้ป่วย การใช้แหล่งจ่ายกำลังไฟ ตรงแบบขั้วเดี่ยวจากแบตเตอรี จำเป็นต้องมีวงจรอิเล็กทรอนิกส์สำหรับตรวจจับระดับแรงดันของแบตเตอรี เพื่อเตือนถึงระดับแรงดันไฟตรงที่ส่งผลให้การวัดสัญญาณคลื่นไฟฟ้าหัวใจ อาจเกิดความผิดพลาดได้ วิทยานิพนธ์ฉบับนี้เลือกการออกแบบการตรวจจับแบบพื้นฐาน เพื่อเกิดความสะดวกต่อการพัฒนาเป็นวงจรรวมแบบชิพเดี่ยว โดยมีวงจรตามรูปที่ ก.3



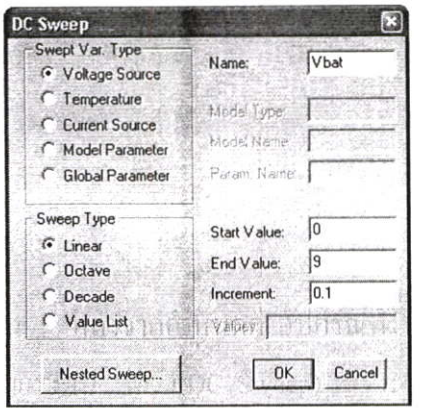
รูปที่ ก.3 วงจรตรวจจับระดับแรงดันต่ำกว่าเกณฑ์

จากรูปที่ ก.3 เมื่อทำการจำลองด้วยโปรแกรมพี-สไปซ์(P-Spice) โดยกำหนดค่าอุปกรณ์ทั้งหมดตามวงจรที่ได้ออกแบบ ส่วนขั้วคอลเลกเตอร์ซึ่งเป็นสัญญาณเอาต์พุต แทนสภาวะการทำงานด้วยความต้านทานโหลด(RL)



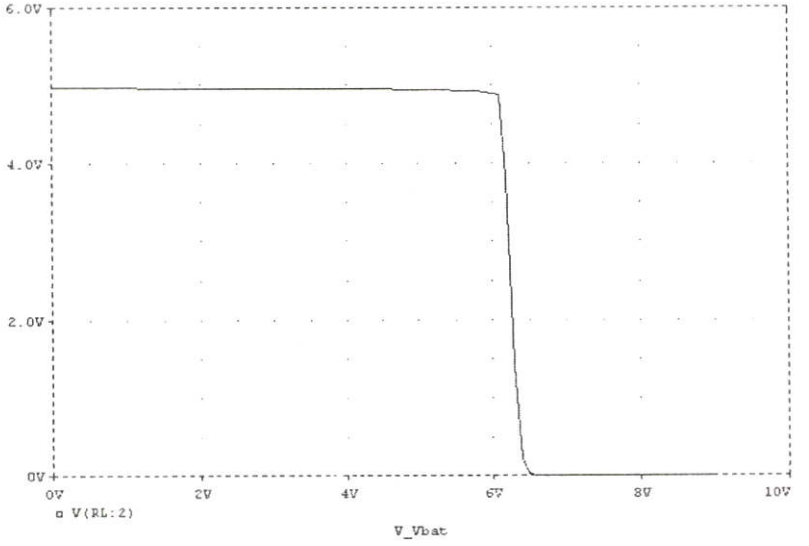
รูปที่ ก.4 การจำลองวงจรตรวจจับระดับแรงดันต่ำกว่าเกณฑ์

ในการจำลองการทำงาน หลังจากกำหนดค่าอุปกรณ์ทั้งหมดแล้ว ได้ตั้งค่าการวิเคราะห์ (Analysis Setup) เป็นแบบแปรค่าแรงดันไฟตรง(DC Sweep) เพราะต้องการจำลองทางด้านไฟตรงทางแรงดันของแบเตอรี โดยทำการปรับค่าของแหล่งจ่ายแรงดันแบตเตอรี ตั้งแต่ค่าแรงดัน 0 โวลต์ จนถึงแรงดัน 9 โวลต์ โดยให้เพิ่มค่าแบบเชิงเส้น(Linear)ครั้งละ 0.1 โวลต์ ตามรูปที่ ก.5



รูปที่ ก.5 การตั้งค่าของการวิเคราะห์แบบแปรค่าแรงดันไฟตรง

จากการจำลองวงจรตรวจจับระดับแรงดันต่ำกว่าเกณฑ์ โดยใช้โปรแกรมพี-สไปซ์ จะเห็นว่าเมื่อแรงดันไฟตรงจากแบตเตอรีลดระดับลงเหลือค่าแรงดันประมาณ 6.2 โวลต์ แรงดันที่ขั้วคอลเลกเตอร์จะมีค่าเท่ากับ 5 โวลต์ ทำให้รับรู้ถึงระดับของแรงดันทางไฟตรงได้



รูปที่ ก.6 ผลการจำลองวงจรตรวจจับระดับแรงดันต่ำกว่าเกณฑ์

ภาคผนวก ข.

Function & Script MATLAB

Script : blackman.m

```
1.  % Designing FIR bandstop filter using Blackman window
2.  disp('Script for designing FIR bandstop filter using Blackman window')
3.  disp('Written by Mr.Suriya Witthayapradit')
4.  disp('Department of Instrumentation Engineering')
5.  disp('King Mongkut's Institute of Technology Ladkrabang')
6.  N = input('Enter the length (order(N)) of the filter is : ');
7.  fc1 = input('Enter the lower passband cutoff frequency(Hz.) : ');
8.  fc2 = input('Enter the upper passband cutoff frequency(Hz.) : ');
9.  fs = input('Enter sampling rate in Hz. : ');
10. % ------------------------------------------------------------------ %
11. Ts = 1/fs;
12. w1 = 2*pi*fc1*Ts;
13. w2 = 2*pi*fc2*Ts;
14. if rem(N,2) == 0 N = N+1; end;
15. % disp('The length (order) of the filter is')
16. % disp(N)
17. for k = 1:N
18. if (k-1) ~= (N-1)/2
19. hds(k) = -sin(w2*(k-1-(N-1)/2))/(pi*(k-1-(N-1)/2))+sin(w1*(k-1-(N-1)/2))/(pi*(k-1-(N-1)/2));
20. else
21. hds(k) = 1-w2/pi + w1/pi;
22. end
23. end
24. for k = 1:N
25. t(k)=k-1;
26. end
27. for k = 1:N
28. w(k) = 0.42-0.5*cos(2*pi*(k-1)/(N-1))+0.08*cos(4*pi*(k-1)/(N-1));
29. end
30. for k = 1:N
31. h(k) = hds(k)*w(k);
32. end
33. M = 400;
34. for n = 1:M+1
35. f(n) = (n-1)/M;
36. q(n) = 2*pi*f(n);
37. H(n) = h((N-1)/2+1)+2*h(1)*cos(q(n)*(-(N-1)/2));
38. for k = 2:(N-1)/2
39. H(n) = H(n) + 2*h(k)*cos(q(n)*(k-1-(N-1)/2));
40. end
41. H(n) = exp(-j*q(n)*(N-1)/2)*H(n);
42. dB(n) = 20*log10(abs(H(n)));
43. if dB(n) < -200 dB(n) = -200; end
44. pha(n) = angle(H(n))/pi;
45. end
46. figure(1)
47. semilogx(f*fs,dB);
48. axis([0 100 -60 10]);grid on;
49. title('Magnitude Response in dB')
50. xlabel('Frequency(Hz)')
51. ylabel('Magnitude in dB')
52. figure(2)
53. plot(f*fs,pha*180/pi);
54. axis([0 100 -60 60]);grid on;
55. title('Phase Response of Blackman window')
56. xlabel('Frequency(Hz)')
57. ylabel('Phase(Degree)')
```

Script : hamming.m

```
% Designing FIR bandstop filter using Hamming window
disp('Script for designing FIR bandstop filter using Hamming window')
disp('Written by Mr.Suriya Witthayapradit')
disp('Department of Instrumentation Engineering')
disp('King Mongkut's Institute of Technology Ladkrabang')
N = input('Enter the length (order(N)) of the filter is : ');
fc1 = input('Enter the lower passband cutoff frequency(Hz.) : ');
fc2 = input('Enter the upper passband cutoff frequency(Hz.) : ');
fs = input('Enter sampling rate in Hz. : ');
% ------------------------------------------------------------------ %
Ts = 1/fs;
w1 = 2*pi*fc1*Ts;
w2 = 2*pi*fc2*Ts;
if rem(N,2) == 0 N = N+1; end;
% disp('The length (order) of the filter is')
% disp(N)
for k = 1:N
if (k-1) ~= (N-1)/2
hds(k) = -sin(w2*(k-1-(N-1)/2))/(pi*(k-1-(N-1)/2))+sin(w1*(k-1-(N-1)/2))/(pi*(k-1-(N-1)/2));
else
hds(k) = 1-w2/pi + w1/pi;
end
end
for k = 1:N
t(k)=k-1;
end
for k = 1:N
w(k) = 0.54-0.46*cos(2*pi*(k-1)/(N-1));
end
for k = 1:N
h(k) = hds(k)*w(k);
end
M = 400;
for n = 1:M+1
f(n) = (n-1)/M;
q(n) = 2*pi*f(n);
H(n) = h((N-1)/2+1)+2*h(1)*cos(q(n)*(-(N-1)/2));
for k = 2:(N-1)/2
H(n) = H(n) + 2*h(k)*cos(q(n)*(k-1-(N-1)/2));
end
H(n) = exp(-j*q(n)*(N-1)/2)*H(n);
dB(n) = 20*log10(abs(H(n)));
if dB(n) < -200 dB(n) = -200; end
pha(n) = angle(H(n))/pi;
end
figure(1)
semilogx(f*fs,dB);
axis([0  100  -60  10]);grid on;
title('Magnitude Response in dB')
xlabel('Frequency(Hz)')
ylabel('Magnitude in dB')
figure(2)
plot(f*fs,pha*180/pi);
axis([0  100  -100  100]);grid on;
title('Phase Response of Hamming window')
xlabel('Frequency(Hz)')
ylabel('Phase(Degree)')
```

Script : hanning.m

```
% Designing FIR bandstop filter using Hanning window
disp('Script for designing FIR bandstop filter using Hanning window')
disp('Written by Mr.Suriya Witthayapradit')
disp('Department of Instrumentation Engineering')
disp('King Mongkut's Institute of Technology Ladkrabang')
N = input('Enter the length (order(N)) of the filter is : ');
fc1 = input('Enter the lower passband cutoff frequency(Hz.) : ');
fc2 = input('Enter the upper passband cutoff frequency(Hz.) : ');
fs = input('Enter sampling rate in Hz. : ');
% ------------------------------------------------------------------ %
Ts = 1/fs;
w1 = 2*pi*fc1*Ts;
w2 = 2*pi*fc2*Ts;
if rem(N,2) == 0 N = N+1; end;
% disp('The length (order) of the filter is')
% disp(N)
for k = 1:N
if (k-1) ~= (N-1)/2
hds(k) = -sin(w2*(k-1-(N-1)/2))/(pi*(k-1-(N-1)/2))+sin(w1*(k-1-(N-1)/2))/(pi*(k-1-(N-1)/2));
else
hds(k) = 1-w2/pi + w1/pi;
end
end
for k = 1:N
t(k)=k-1;
end
for k = 1:N
w(k) = (1/2)*(1-cos(2*pi*(k-1)/(N-1)));
end
for k = 1:N
h(k) = hds(k)*w(k);
end
M = 400;
for n = 1:M+1
f(n) = (n-1)/M;
q(n) = 2*pi*f(n);
H(n) = h((N-1)/2+1)+2*h(1)*cos(q(n)*(-(N-1)/2));
for k = 2:(N-1)/2
H(n) = H(n) + 2*h(k)*cos(q(n)*(k-1-(N-1)/2));
end
H(n) = exp(-j*q(n)*(N-1)/2)*H(n);
mag(n) = abs(H(n));
dB(n) = 20*log10(abs(H(n)));
if dB(n) < -200 dB(n) = -200; end
pha(n) = angle(H(n))/pi;
end
figure(1)
semilogx(f*fs,dB);
axis([0  100  -60  10]);grid on;
title('Magnitude Response in dB')
xlabel('Frequency(Hz)')
ylabel('Magnitude in dB')
figure(2)
plot(f*fs,pha*180/pi);
axis([0  100  -100  100]);grid on;
title('Phase Response of Hanning window')
xlabel('Frequency(Hz)')
ylabel('Phase(Degree)')
```

Script : Kaiser.m

```
1.  % designing FIR bandstop filter using Kaiser window
2.  A1 = input('ripple in the passband (0 - 1) : ');
3.  A2 = input('ripple in the stopband (0 - 1) : ');
4.  f1lpc = input('Lower passband cutoff frequency in Hz.');
5.  f2upc = input('Upper passband cutoff frequency in Hz.');
6.  f3lsc = input('Lower stopband cutoff frequency in Hz.');
7.  f4usc = input('Upper stopband cutoff frequency in Hz.');
8.  fs = input('Sampling rate in Hz.');
9.  % ------------------------------------------------------------------ %
10. Ts = 1/fs;
11. if f3lsc-f1lpc <= f2upc-f4usc bt = f3lsc-f1lpc; else bt = f2upc-f4usc; end
12. f1lpc = f1lpc+bt/2;
13. f2upc = f2upc-bt/2;
14. w1 = 2*pi*f1lpc*Ts;
15. w2 = 2*pi*f2upc*Ts;
16. if A1 <= A2 am = A1; else am = A2; end
17. aa = -20*log10(am); % aa = stopband attenuation in dB.
18. if aa <= 21 b = 0;
19. elseif aa > 21 & aa <= 50 b = 0.5842*(aa-21)^0.4 + 0.07886*(aa-21);
20. else
21. b = 0.1102*(aa-8.7);
22. end
23. end
24. if aa <= 21 D = 0.9222; else D = (aa-7.95)/14.36; end
25. N = ceil(fs*D/bt+1);
26. if rem(N,2) == 0 N = N+1; end;
27. disp('The length (order) of the filter is')
28. disp(N)
29. for k = 1:N
30. if (k-1) ~= (N-1)/2
31. hds(k) = -sin(w2*(k-1-(N-1)/2))/(pi*(k-1-(N-1)/2))+sin(w1*(k-1-(N-1)/2))/(pi*(k-1-(N-1)/2));
32. else
33. hds(k) = 1-w2/pi + w1/pi;
34. end
35. end
36. for k = 1:N
37. t(k)=k-1;
38. end
39. for k = 1:N
40. w(k) = bessel(0,j*b*sqrt(1-(1-2*(k-1)/(N-1))^2))/bessel(0,j*b);
41. end
42. for k = 1:N
43. t(k) = k-1;
44. z(k) = 0;
45. if w(k) >= 0 Hi(k) = w(k); Lo(k) = 0;
46. else Hi(k) = 0; Lo(k) = -w(k);
47. end
48. end
49. for k = 1:N
50. h(k) = hds(k)*w(k);
51. end
52. for k = 1:N
53. t(k) = k-1;
54. z(k) = 0;
55. if h(k) >= 0 Hi(k) = h(k); Lo(k) = 0;
56. else Hi(k) = 0; Lo(k) = -h(k);
57. end
58. end
```

```
sum1=sum(h);
for k = 1:N
h(k) = h(k)/sum1;
end
for k = 1:N
t(k) = k-1;
z(k) = 0;
if h(k) >= 0 Hi(k) = h(k); Lo(k) = 0;
else Hi(k) = 0; Lo(k) = -h(k);
end
end
M = 400;
for n = 1:M+1
f(n) = (n-1)/M;
q(n) = 2*pi*f(n);
H(n) = h((N-1)/2+1)+2*h(1)*cos(q(n)*(-(N-1)/2));
for k = 2:(N-1)/2
H(n) = H(n) + 2*h(k)*cos(q(n)*(k-1-(N-1)/2));
end
H(n) = exp(-j*q(n)*(N-1)/2)*H(n);
mag(n) = abs(H(n));
dB(n) = 20*log10(abs(H(n)));
if dB(n) < -200 dB(n) = -200; end
pha(n) = angle(H(n))/pi;
end
figure
plot(f*fs,dB);grid;
axis([0  100  -40  10]);grid on;
title('Magnitude Response in dB')
xlabel('Frequency')
ylabel('Magnitude in dB')
figure
plot(f*fs,pha*180/pi);grid;
axis([0  100  -60  60]);grid on;
title('Phase Response of Kaiser window')
xlabel('Frequency')
ylabel('Phase in Degree')
```

Script : tukey.m

```
1.  %designing FIR bandstop filter using Tukey window
2.  N = input('Enter the length (order(N)) of the filter is : ');
3.  a = input('parameter for Tukey window.  0 <= a <= 1 : ');
4.  fc1 = input('Enter the lower passband cutoff frequency(Hz.) : ');
5.  fc2 = input('Enter the upper passband cutoff frequency(Hz.) : ');
6.  fs = input('Enter sampling rate in Hz. : ');
7.  % ------------------------------------------------------------- %
8.  Ts = 1/fs;
9.  w1 = 2*pi*fc1*Ts; % q1bsf = Lower passband cutoff frequency.
10. w2 = 2*pi*fc2*Ts; % q2bsf = Upper passband cutoff frequency.
11. if rem(N,2) == 0 N = N+1; end;
12. % disp('The length (order) of the filter is')
13. % disp(N)
14. for k = 1:N
15. if (k-1) ~= (N-1)/2
16. hds(k) = -sin(w2*(k-1-(N-1)/2))/(pi*(k-1-(N-1)/2))+sin(w1*(k-1-(N-1)/2))/(pi*(k-1-(N-1)/2));
17. else
18. hds(k) = 1-w2/pi + w1/pi;
19. end
20. end
21. for k = 1:N
22. t(k)=k-1;
23. end
24. for k = 1:N
25. if k > (1+a)*(N-1)/2 + 1
26. w(k) = (1/2)*[1 + cos((k-1-(1+a)*(N-1)/2)*pi*2/((1-a)*(N-1)))];
27. elseif k >= (1-a)*(N-1)/2 + 1 & k <= (1+a)*(N-1)/2 + 1 w(k) = 1;
28. else
29. w(k) = (1/2)*[1 + cos((N+1-k-1-(1+a)*(N-1)/2)*pi*2/((1-a)*(N-1)))];
30. end
31. end
32. for k = 1:N
33. h(k) = hds(k)*w(k);
34. end
35. M = 400;
36. for n = 1:M+1
37. f(n) = (n-1)/M;
38. q(n) = 2*pi*f(n);
39. H(n) = h((N-1)/2+1)+2*h(1)*cos(q(n)*(-(N-1)/2));
40. for k = 2:(N-1)/2
41. H(n) = H(n) + 2*h(k)*cos(q(n)*(k-1-(N-1)/2));
42. end
43. H(n) = exp(-j*q(n)*(N-1)/2)*H(n);
44. dB(n) = 20*log10(abs(H(n)));
45. if dB(n) < -200 dB(n) = -200; end
46. pha(n) = angle(H(n))/pi;
47. end
48. figure(1)
49. semilogx(f*fs,dB);axis([0  100  -60  10]);grid on;
50. title('Magnitude Response in dB')
51. xlabel('Frequency(Hz)')
52. ylabel('Magnitude in dB')
53. figure(2)
54. plot(f*fs,pha*180/pi);axis([0  100  -100  100]);grid on;
55. title('Phase Response of Tukey window')
56. xlabel('Frequency(Hz)')
57. ylabel('Phase(Degree)')
```

Function : Butterworth bandstop filter

```
1.  % Function for designing digital Butterworth bandstop filter.
2.  function[dp,dz,h] = dbubsf(amax,amin,f1bsf,f2bsf,f3bsf,f4bsf,fs)
3.  Ts = 1/fs;    % Ts = sampling interval.
4.  q1bsf = 2*pi*f1bsf*Ts;    % q1bsf = digital lower passband cutoff frequency in radians.
5.  q2bsf = 2*pi*f2bsf*Ts;    % q2bsf = digital upper passband cutoff frequency in radians.
6.  q3bsf = 2*pi*f3bsf*Ts;    % q3bsf = digital lower stopband cutoff frequency in radians.
7.  q4bsf = 2*pi*f4bsf*Ts;    % q4bsf = digital upper stopband cutoff frequency in radians.
8.  w1bsf = 2*fs*tan(q1bsf/2);  % w1bsf = prewarped lower passband cutoff frequency in radians/s.
9.  w2bsf = 2*fs*tan(q2bsf/2);  % w2bsf = prewarped upper passband cutoff frequency in radians/s.
10. w3bsf = 2*fs*tan(q3bsf/2);  % w3bsf = prewarped lower stopband cutoff frequency in radians/s.
11. w4bsf = 2*fs*tan(q4bsf/2);  % w4bsf = prewarped upper stopband cutoff frequency in radians/s.
12. if w1bsf*w2bsf > w3bsf*w4bsf w4bsf = w1bsf*w2bsf/w3bsf; end
13. if w1bsf*w2bsf < w3bsf*w4bsf w3bsf = w1bsf*w2bsf/w4bsf; end
14. N = ceil(log((10^(0.1*amin)-1)/(10^(0.1*amax)-1))/(2*log((w2bsf-w1bsf)/(w4bsf-w3bsf)))); % N =
    order of the filter.
15. disp('The order of the filter is')
16. disp(N)
17. % Calculating the normalized lowpass pole locations.
18. % Vector s is the normalized poles.
19. if rem(N,2) == 1 s(1) = -1; else s(1) = 0; end
20. if rem(N,2) == 1 min = (N+1)/2; else min = N/2 + 1; end
21. if N == 1 max = 1; elseif rem(N,2) == 1 max = N-1; else max = N; end
22. if rem(N,2) == 1 factor = 0; else factor = 1/2; end
23. if N == 1 maxa = 1; elseif rem(N,2) == 1 maxa = (N-1)/2; else maxa = N/2; end
24. if rem(N,2) == 1 evena = 0; else evena = 1; end
25. for m = min:max
26. s(m) = cos(pi*(m - factor)/N) + j*sin(pi*(m - factor)/N);
27. end
28. for ka = 1:maxa
29. s(2*ka-evena) = s((N+1+evena+2*(ka-1))/2);
30. s(2*ka+1-evena) = conj(s(2*ka-evena));
31. end
32. % Plotting the normalized analog lowpass poles.
33. for n = 1:201
34. x(n) = (n-101)/100;
35. ty(n) = sqrt(1-x(n).*x(n));
36. by(n) = - ty(n);
37. end
38. figure
39. plot(x,ty)
40. hold on
41. plot(x,by)
42. plot(real(s),imag(s),'bx'); grid;
43. title('Normalized Analog Lowpass Pole Locations')
44. xlabel('Real Part')
45. ylabel('Imaginary Part')
46. disp('Normalized Analog Lowpass Pole Locations')
47. disp(s);
48. % Calculating the normalized analog lowpass zero locations.
49. % Vector szb is the normalized analog zeros.
50. for k = 1:N
51. sz(k) = 10^50;
52. end
53. % Calculating the frequency transformed analog poles and zeros.
54. % Lowpass to bandstop transformation.
55. Wo = 1/(10^(0.1*amax)-1)^(1/(2*N)); % Wo = half-power frequency.
56. w21=(w2bsf-w1bsf)/2;
57. w2m1=w2bsf-w1bsf;
```

```
w12t4=4*w1bsf*w2bsf;
for k = 1:N
s(k) = Wo*s(k);
R1(k) = real(s(k));
I1(k) = imag(s(k));
R2(k) = R1(k)^2;
I2(k) = I1(k)^2;
mag(k) = R2(k) + I2(k);
A(k) = (I2(k)-R2(k))*w2m1^2/mag(k)^2 + w12t4;
B(k) = 2*abs(R1(k))*abs(I1(k))*w2m1^2/mag(k)^2;
RA(k) = (1/2)*sqrt((sqrt(A(k)^2+B(k)^2)-A(k))/2);
RB(k) = j*(1/2)*(sqrt((sqrt(A(k)^2+B(k)^2)+A(k))/2)-I1(k)*w2m1/mag(k));
end
for m = 1:2*N
if rem(m,2) == 1
if I1((m+1)/2) > 0 sfs(m) = w21*R1((m+1)/2)/mag((m+1)/2) + RA((m+1)/2) + RB((m+1)/2); end
if I1((m+1)/2) < 0 sfs(m) = w21*R1((m+1)/2)/mag((m+1)/2) - RA((m+1)/2) + RB((m+1)/2); end
R1a = (w2m1/R1((m+1)/2))^2;
if I1((m+1)/2) == 0 & R1a >= w12t4 sfs(m) = w21/R1((m+1)/2) + (1/2)*sqrt(R1a-w12t4); end
if I1((m+1)/2) == 0 & R1a < w12t4 sfs(m) = w21/R1((m+1)/2) + j*(1/2)*sqrt(-R1a+w12t4); end
else
sfs(m) = conj(sfs(m-1));
if I1(m/2) == 0 & (w2m1/R1(m/2))^2 >= w12t4
sfs(m) = w21/R1(m/2) - (1/2)*sqrt(w2m1^2/R1(m/2)^2-w12t4); end
end
end
if N==1 sfs(1) = w21/R1(1) + (1/2)*sqrt((w2m1/R1(1))^2-w12t4); end
if N==1 sfs(2) = w21/R1(1) - (1/2)*sqrt((w2m1/R1(1))^2-w12t4); end
for m = 1:2*N
if rem(m,2) == 1 szfs(m) = j*sqrt(w1bsf*w2bsf); else szfs(m) = conj(szfs(m-1)); end
end
% Plotting the frequency transformed bandpass poles.
wo = sqrt(w1bsf*w2bsf)/Wo;
for na = 1:201
xa(na) = wo*(na-101)/100;
tya(na) = sqrt(wo^2-xa(na)^2);
bya(na) = - tya(na);
end
figure
plot(xa,tya)
hold on
plot(xa,bya)
plot(real(sfs),imag(sfs),'bx'); grid;
title('Frequency Transformed Bandstop Pole Locations')
xlabel('Real Part')
ylabel('Imaginary Part')
disp('Frequency Transformed Bandstop Pole Locations')
disp(sfs)
disp('Frequency Transformed Bandstop Zero Locations')
disp(szfs)
% Transforming analog poles and zeros to digital poles and zeros using bilinear transformation.
for m = 1:2*N
dp(m) = (1 + (Ts/2)*sfs(m))/(1 - (Ts/2)*sfs(m));
dz(m) = (1 + (Ts/2)*szfs(m))/(1 - (Ts/2)*szfs(m));
end
% Plotting digital poles and zeros on the complex z-plane
figure
plot(x,ty)
hold on
plot(x,by)
```

```
118.plot(real(dp),imag(dp),'bx')
119.plot(real(dz),imag(dz),'ro'); grid;
120.title('Digital Poles and Zeros')
121.xlabel('Real Part')
122.ylabel('Imaginary Part')
123.disp('Digital Pole Locations')
124.disp(dp)
125.disp('Digital Zero Locations')
126.disp(dz)
127.%
128.for i = 1:N
129.d2(i,1)=1; d2(i,2)=-2*real(dp(2*i-1));d2(i,3) = (abs(dp(2*i-1)))^2;
130.n2(i,1)=1; n2(i,2)=-2*real(dz(2*i-1));n2(i,3) = (abs(dz(2*i-1)))^2;
131.end
132.disp('Numerator Second Order Terms')
133.disp(n2)
134.disp('Denominator Second Order Terms')
135.disp(d2)
136.% Plotting the magnitude response and the phase response with respect to the normalized digital
    frequency.
137.M = 128;  % M is the number of points the frequency response is calculated.
138.step = 2*pi/M;
139.for m = 1:M
140.H(m) = (1-dz(1)*exp(j*(-1)*(m-1)*step))*(1-dz(2)*exp(j*(-1)*(m-1)*step))/[(1-dp(1)*exp(j*(-1)*(m-
    1)*step))*(1-dp(2)*exp(j*(-1)*(m-1)*step))];
141.end
142.if N > 1
143.for n = 3:2*N
144.for m = 1:M
145.H(m) = H(m).*[(1-dz(n)*exp(j*(-1)*(m-1)*step))/(1-dp(n)*exp(j*(-1)*(m-1)*step))];
146.end
147.end
148.end
149.max = abs(H(1));
150.for m = 2:M
151.if abs(H(m)) > max max = abs(H(m)); end
152.end
153.disp('maximum gain is')
154.disp(max)
155.b0=1/max;
156.disp('Constant Term b0 =')
157.disp(b0);
158.%Polynomial Form
159.nump(1) = n2(1,1); nump(2) = n2(1,2); nump(3) = n2(1,3);
160.denp(1) = d2(1,1); denp(2) = d2(1,2); denp(3) = d2(1,3);
161.if N > 1
162.for i=2:N nump = conv(nump,n2(i,:)); denp = conv(denp,d2(i,:)); end
163.end
164.disp('Numerator Polynomial')
165.disp(b0*nump)
166.disp('Denominator Polynomial')
167.disp(denp)
168.%
169.for m = 1:M
170.f(m) = (m-1)/M ;
171.H(m) = H(m)/max;
172.mag(m) = abs(H(m));
173.if H(m) == 0 dB(m) = -200; else dB(m) = 20*log10(mag(m)); end
174.if dB(m) < -200 dB(m) = -200; end
175.pha(m) = angle(H(m));
```

```
176.end
177.figure
178.semilogx(f*fs,mag);
179.axis([0  100  0  1]);grid;
180.title('Butterworth Magnitude Response')
181.xlabel('Frequency(Hz)')
182.ylabel('Magnitude(Volt.)')
183.figure
184.% semilogx(f*fs,dB ,':r',y,x,'-b');
185.semilogx(f*fs,dB ,'-r');hold on;
186.axis([0  100  -40  0]);grid;
187.title('Butterworth Magnitude Response in dB')
188.xlabel('Frequency(Hz)')
189.ylabel('Magnitude in dB')
190.figure
191.% semilogx(f*fs,pha*180/pi,':r',q,p,'b');
192.semilogx(f*fs,pha*180/pi,'-r');
193.axis([0  100  -180  180]); grid;
194.title('Phase Response')
195.xlabel('Frequency')
196.ylabel('Phase in Degree')
197.% Calculating and Plotting the Impulse Response of the Filter.
198.h = ifft(H);
199.for m = 1:M
200.k(m) = m-1; z(m) = 0; Lo(m) = 0; Hi(m) = 0;
201.if h(m) >= 0 Hi(m) = h(m); else Lo(m) = h(m); end
202.end
203.maxh = h(1); minh = h(1);
204.for m = 2:M
205.if h(m) > maxh maxh = h(m); end
206.if h(m) < minh minh = h(m); end
207.end
208.figure
209.%errorbar(k,z,Lo,Hi,'w')
210.stem(k,h); grid;
211.title('Impulse Response')
212.xlabel('Time')
213.ylabel('Amplitude')
```

Function : Chebyshev bandstop filter

```
1.  % Function for designing digital Chebyshev bandstop filter.
2.  function[dp,dz,h] = dchbsf(amax,amin,f1bsf,f2bsf,f3bsf,f4bsf,fs)
3.  Ts = 1/fs;    % Ts = sampling interval.
4.  q1bsf = 2*pi*f1bsf*Ts;    % q1bsf = digital lower passband cutoff frequency in radians.
5.  q2bsf = 2*pi*f2bsf*Ts;    % q2bsf = digital upper passband cutoff frequency in radians.
6.  q3bsf = 2*pi*f3bsf*Ts;    % q3bsf = digital lower stopband cutoff frequency in radians.
7.  q4bsf = 2*pi*f4bsf*Ts;    % q4bsf = digital upper stopband cutoff frequency in radians.
8.  w1bsf = 2*fs*tan(q1bsf/2);  % w1bsf = prewarped lower passband cutoff frequency in radians/s.
9.  w2bsf = 2*fs*tan(q2bsf/2);  % w2bsf = prewarped upper passband cutoff frequency in radians/s.
10. w3bsf = 2*fs*tan(q3bsf/2);  % w3bsf = prewarped lower stopband cutoff frequency in radians/s.
11. w4bsf = 2*fs*tan(q4bsf/2);  % w4bsf = prewarped upper stopband cutoff frequency in radians/s.
12. if w1bsf*w2bsf > w3bsf*w4bsf w4bsf = w1bsf*w2bsf/w3bsf; end
13. if w1bsf*w2bsf < w3bsf*w4bsf w3bsf = w1bsf*w2bsf/w4bsf; end
14. N = ceil(acosh(sqrt((10^(0.1*amin)-1)/(10^(0.1*amax)-1)))/acosh((w2bsf-w1bsf)/(w4bsf-w3bsf))); %
    N = order of the filter.
15. disp('Digital Chebyshev (Type I) Bandstop Filter Design')
16. disp(' ')
17. disp('Lower passband cutoff frequency q1bsf =')
18. disp(q1bsf)
19. disp('Upper passband cutoff frequency q2bsf =')
20. disp(q2bsf)
21. disp('Lower stopband cutoff frequency q3bsf =')
22. disp(q3bsf)
23. disp('Upper stopband cutoff frequency q4bsf =')
24. disp(q4bsf)
25. disp('Prewarped lower passband cutoff frequency w1bsf =')
26. disp(w1bsf)
27. disp('Prewarped upper passband cutoff frequency w2bsf =')
28. disp(w2bsf)
29. disp('Prewarped lower stopband cutoff frequency w3bsf =')
30. disp(w3bsf)
31. disp('Prewarped upper stopband cutoff frequency w4bsf =')
32. disp(w4bsf)
33. disp('The order of the filter is')
34. disp(N)
35. % Calculating the normalized lowpass pole locations.
36. % Vector s is the normalized poles.
37. a = (1/N)*asinh(1/sqrt(10^(0.1*amax)-1));
38. if rem(N,2) == 1 s(1) = -sinh(a); end
39. if N == 1 max = 0;
40. elseif rem(N,2) == 1 max = (N-3)/2;
41. else max = (N-2)/2;
42. end
43. end
44. if N > 1
45. for m = 0:max
46. if rem(N,2) == 1
47. s(2*m+2) = -sinh(a)*sin((2*m+1)*pi/(2*N))+j*cosh(a)*cos((2*m+1)*pi/(2*N));
48. s(2*m+3) = conj(s(2*m+2));
49. else
50. s(2*m+1) = -sinh(a)*sin((2*m+1)*pi/(2*N))+j*cosh(a)*cos((2*m+1)*pi/(2*N));
51. s(2*m+2) = conj(s(2*m+1));
52. end
53. end
54. end
55. % Plotting the normalized analog lowpass poles.
56. for n = 1:201
57. x(n) = (n-101)/100;
```

```
ty(n) = sqrt(1-x(n).*x(n));
by(n) = - ty(n);
end
figure
plot(x,ty)
hold on
plot(x,by)
plot(real(s),imag(s),'bx'); grid;
title('Normalized Analog Lowpass Pole Locations')
xlabel('Real Part')
ylabel('Imaginary Part')
disp('Normalized Analog Lowpass Pole Locations')
disp(s)
% Calculating the normalized analog lowpass zero locations.
% Vector szb is the normalized analog zeros.
for k = 1:N
sz(k) = 10^50;
end
% Calculating the frequency transformed analog poles and zeros.
% Lowpass to bandstop transformation.
w21=(w2bsf-w1bsf)/2;
w2m1=w2bsf-w1bsf;
w12t4=4*w1bsf*w2bsf;
for k = 1:N
R1(k) = real(s(k));
I1(k) = imag(s(k));
R2(k) = R1(k)^2;
I2(k) = I1(k)^2;
mag(k) = R2(k) + I2(k);
A(k) = (I2(k)-R2(k))*w2m1^2/mag(k)^2 + w12t4;
B(k) = 2*abs(R1(k))*abs(I1(k))*w2m1^2/mag(k)^2;
RA(k) = (1/2)*sqrt((sqrt(A(k)^2+B(k)^2)-A(k))/2);
RB(k) = j*(1/2)*(sqrt((sqrt(A(k)^2+B(k)^2)+A(k))/2)-I1(k)*w2m1/mag(k));
end
for m = 1:2*N
if rem(m,2) == 1
if I1((m+1)/2) > 0 sfs(m) = w21*R1((m+1)/2)/mag((m+1)/2) + RA((m+1)/2) + RB((m+1)/2); end
if I1((m+1)/2) < 0 sfs(m) = w21*R1((m+1)/2)/mag((m+1)/2) - RA((m+1)/2) + RB((m+1)/2); end
R1a = (w2m1/R1((m+1)/2))^2;
if I1((m+1)/2) == 0 & R1a >= w12t4 sfs(m) = w21/R1((m+1)/2) + (1/2)*sqrt(R1a-w12t4); end
if I1((m+1)/2) == 0 & R1a < w12t4 sfs(m) = w21/R1((m+1)/2) + j*(1/2)*sqrt(-R1a+w12t4); end
else
sfs(m) = conj(sfs(m-1));
if I1(m/2) == 0 & (w2m1/R1(m/2))^2 >= w12t4 sfs(m) = w21/R1(m/2) -
    (1/2)*sqrt(w2m1^2/R1(m/2)^2-w12t4); end
end
end
for m = 1:2*N
if rem(m,2) == 1 szfs(m) = j*sqrt(w1bsf*w2bsf); else szfs(m) = conj(szfs(m-1)); end
end
% Plotting the frequency transformed bandpass poles.
wo = sqrt(w1bsf*w2bsf);
for na = 1:201
xa(na) = wo*(na-101)/100;
tya(na) = sqrt(wo^2-xa(na)^2);
bya(na) = - tya(na);
end
figure
plot(xa,tya)
hold on
```

```
117.plot(xa,bya)
118.plot(real(sfs),imag(sfs),'bx'); grid;
119.title('Frequency Transformed Bandstop Pole Locations')
120.xlabel('Real Part')
121.ylabel('Imaginary Part')
122.disp('Frequency Transformed Bandstop Pole Locations')
123.disp(sfs)
124.disp('Frequency Transformed Bandstop Zero Locations')
125.disp(szfs)
126.% Transforming analog poles and zeros to digital poles and zeros using bilinear transformation.
127.for m = 1:2*N
128.dp(m) = (1 + (Ts/2)*sfs(m))/(1 - (Ts/2)*sfs(m));
129.dz(m) = (1 + (Ts/2)*szfs(m))/(1 - (Ts/2)*szfs(m));
130.end
131.% Plotting digital poles and zeros on the complex z-plane
132.figure
133.plot(x,ty)
134.hold on
135.plot(x,by)
136.plot(real(dp),imag(dp),'bx')
137.plot(real(dz),imag(dz),'ro');grid;
138.title('Digital Poles and Zeros')
139.xlabel('Real Part')
140.ylabel('Imaginary Part')
141.disp('Digital Pole Locations')
142.disp(dp)
143.disp('Digital Zero Locations')
144.disp(dz)
145.%
146.for i = 1:N
147.d2(i,1)=1; d2(i,2)=-2*real(dp(2*i-1));d2(i,3) = (abs(dp(2*i-1)))^2;
148.n2(i,1)=1; n2(i,2)=-2*real(dz(2*i-1));n2(i,3) = (abs(dz(2*i-1)))^2;
149.end
150.disp('Numerator Second Order Terms')
151.disp(n2)
152.disp('Denominator Second Order Terms')
153.disp(d2)
154.% Plotting the magnitude response and the phase response with respect to the normalized digital
    frequency.
155.M = 256;  % M is the number of points the frequency response is calculated.
156.step = 2*pi/M;
157.for m = 1:M
158.H(m) = (1-dz(1)*exp(j*(-1)*(m-1)*step))*(1-dz(2)*exp(j*(-1)*(m-1)*step))/[(1-dp(1)*exp(j*(-1)*(m-
    1)*step))*(1-dp(2)*exp(j*(-1)*(m-1)*step))];
159.end
160.if N > 1
161.for n = 3:2*N
162.for m = 1:M
163.H(m) = H(m).*[(1-dz(n)*exp(j*(-1)*(m-1)*step))/(1-dp(n)*exp(j*(-1)*(m-1)*step))];
164.end
165.end
166.end
167.max = abs(H(1));
168.for m = 2:M
169.if abs(H(m)) > max max = abs(H(m)); end
170.end
171.disp('maximum gain is')
172.disp(max)
173.b0=1/max;
174.disp('Constant Term b0 =')
```

```
175.disp(b0);
176.%Polynomial Form
177.nump(1) = n2(1,1); nump(2) = n2(1,2); nump(3) = n2(1,3);
178.denp(1) = d2(1,1); denp(2) = d2(1,2); denp(3) = d2(1,3);
179.if N > 1
180.for i=2:N nump = conv(nump,n2(i,:)); denp = conv(denp,d2(i,:)); end
181.end
182.disp('Numerator Polynomial')
183.disp(b0*nump)
184.disp('Denominator Polynomial')
185.disp(denp)
186.%
187.for m = 1:M
188.f(m) = (m-1)/M;
189.H(m) = H(m)/max;
190.mag(m) = abs(H(m));
191.if H(m) == 0 dB(m) = -200; else dB(m) = 20*log10(mag(m)); end
192.if dB(m) < -200 dB(m) = -200; end
193.pha(m) = angle(H(m));
194.end
195.figure
196.semilogx(f*fs,mag);
197.axis([0  100  0  1]);grid;
198.title('Chebyshev Magnitude Response')
199.xlabel('Frequency(Hz)')
200.ylabel('Magnitude(Volt.)')
201.figure
202.semilogx(f*fs,dB ,'-r');
203.axis([0  100  -40  0]);grid;
204.title('Chebyshev Magnitude Response in dB')
205.xlabel('Frequency')
206.ylabel('Magnitude in dB')
207.figure
208.semilogx(f*fs,pha*180/pi,'-r');
209.axis([0  100  -180  180]); grid;
210.title('Phase Response')
211.xlabel('Frequency(Hz)')
212.ylabel('Phase in Degree')
213.% Calculating and Plotting the Impulse Response of the Filter.
214.h = ifft(H);
215.for m = 1:M
216.k(m) = m-1;
217.z(m) = 0;
218.if real(h(m)) >= 0 Lo(m) = 0; else Lo(m) = -real(h(m)); end
219.if real(h(m)) >= 0 Hi(m) = real(h(m)); else Hi(m) = 0; end
220.end
221.figure
222.%errorbar(k,z,Lo,Hi,'y')
223.stem(k,h); grid;
224.title('Impulse Response')
225.xlabel('Time')
226.ylabel('Amplitude')
```

Function : Elliptic bandstop filter

```
% Function for designing digital Elliptic bandstop filter.
function[dp,dz,h] = delbsf(amax,amin,f1bsf,f2bsf,f3bsf,f4bsf,fs)
Ts = 1/fs;    % Ts = sampling interval.
q1bsf = 2*pi*f1bsf*Ts;    % q1bsf = digital lower pass band cutoff frequency in radians.
q2bsf = 2*pi*f2bsf*Ts;    % q2bsf = digital upper pass band cutoff frequency in radians.
q3bsf = 2*pi*f3bsf*Ts;    % q3bsf = digital lower stop band cutoff frequency in radians.
q4bsf = 2*pi*f4bsf*Ts;    % q4bsf = digital upper stop band cutoff frequency in radians.
w1bsf = 2*fs*tan(q1bsf/2);  % w1bsf = prewarped lower pass band cutoff frequency in radians/s.
w2bsf = 2*fs*tan(q2bsf/2);  % w2bsf = prewarped upper pass band cutoff frequency in radians/s.
w3bsf = 2*fs*tan(q3bsf/2);  % w3bsf = prewarped lower stop band cutoff frequency in radians/s.
w4bsf = 2*fs*tan(q4bsf/2);  % w4bsf = prewarped upper stop band cutoff frequency in radians/s.
Ws1 = (w3bsf*(w2bsf-w1bsf))/(-w3bsf^2 + w1bsf*w2bsf); %backward transformation.
Ws2 = (w4bsf*(w2bsf-w1bsf))/(-w4bsf^2 + w1bsf*w2bsf); %backward transformation.
Wslpf = min(abs(Ws1),abs(Ws2)); %Wslpf = stopband cutoff frequency of the normalized lowpass
  fiter.
if w1bsf*w2bsf > w3bsf*w4bsf w4bsf = w1bsf*w2bsf/w3bsf; end
q4absf = 2*atan(w4bsf*Ts/2);  %q4absf = q4bsf corresponding to adjusted w4bsf
if w1bsf*w2bsf < w3bsf*w4bsf w3bsf = w1bsf*w2bsf/w4bsf; end
q3absf = 2*atan(w3bsf*Ts/2);   %q3absf = q3bsf corresponding to adjusted w3bsf
wobsf = sqrt(w1bsf*w2bsf); %wobsf = geometric mean of w1bsf and w2bsf.
% Calculating the order of the normalized lowpass filter.
k = (w4bsf-w3bsf)/(w2bsf-w1bsf);
Wp = sqrt(k);
Ws = 1/sqrt(k);
kp = sqrt(1-k^2);
qo = (1/2)*(1-sqrt(kp))/(1+sqrt(kp));
q = qo + 2*qo^5 + 15*qo^9 + 150*qo^13;
D = (10^(0.1*amin)-1)/(10^(0.1*amax)-1);
N = ceil(log10(16*D)/log10(1/q)); % N = order of the filter.
disp('Digital Elliptic Bandstop Filter Design')
disp(' ')
disp('Lower passband cutoff frequency q1bsf =')
disp(q1bsf)
disp('Upper passband cutoff frequency q2bsf =')
disp(q2bsf)
disp('Lower stopband cutoff frequency q3bsf =')
disp(q3bsf)
disp('Upper stopband cutoff frequency q4bsf =')
disp(q4bsf)
disp('Prewarped lower passband cutoff frequency w1bsf =')
disp(w1bsf)
disp('Prewarped upper passband cutoff frequency w2bsf =')
disp(w2bsf)
disp('Prewarped lower stopband cutoff frequency w3bsf =')
disp(w3bsf)
disp('Prewarped upper stopband cutoff frequency w4bsf =')
disp(w4bsf)
disp('Center frequency = sqrt(w1bsf*w2bsf) = wobsf =')
disp(wobsf)
disp('The order of the filter is')
disp(N)
% Calculating the normalized lowpass pole locations.
% Vector s is the normalized poles.
L = (1/(2*N))*log((10^(0.05*amax)+1)/(10^(0.05*amax)-1));
sumn = sinh(L);
for m = 1:50
sumn = sumn + (-1)^m*q^(m*(m+1))*sinh((2*m+1)*L);
end
```

```
sumd = -q*cosh(2*L);
for m = 2:50
sumd = sumd + (-1)^m*q^m^2*cosh(2*m*L);
end
so = abs(2*q^0.25*sumn/(1+2*sumd));
W = sqrt((1+k*so^2)*(1+so^2/k));
if N == 1 r = 1;
elseif rem(N,2) == 1
r = (N-1)/2;
else
r = N/2;
end
if rem(N,2) == 1 muminus = 0; else muminus = 0.5; end
for p = 1:r
sumna = sin(pi*(p-muminus)/N);
for m = 1:50
sumna = sumna + (-1)^m*q^(m*(m+1))*sin((2*m+1)*pi*(p-muminus)/N);
end
sumda = -q*cos(2*pi*(p-muminus)/N);
for m = 2:50
sumda = sumda + (-1)^m*q^m^2*cos(2*m*pi*(p-muminus)/N);
end
WG(p) = 2*q^0.25*sumna/(1+2*sumda);
V(p) = sqrt((1-k*WG(p)^2)*(1-WG(p)^2/k));
if N == 1 WG(1) = 10^(-50); end
Ao(p) = 1/WG(p)^2;
Bo(p) = [(so*V(p))^2 + (WG(p)*W)^2]/[1+so^2*WG(p)^2]^2;
B1(p) = 2*so*V(p)/[1+so^2*WG(p)^2];
end
if rem(N,2) == 1 Ho = so*Bo(1)/Ao(1); end
if rem(N,2) == 0 Ho = 10^(-0.05*amax)*Bo(p)/Ao(p); end
if N > 1
for p = 2:r
Ho = Ho*Bo(p)/Ao(p);
end
end
if rem(N,2) == 1 s(1) = -so; end
if N > 1
if rem(N,2) == 1
for m = 2:2*r+1
if rem(m,2) == 0
if B1(m/2)^2 >= 4*Bo(m/2) s(m) = -B1(m/2)/2+0.5*sqrt(B1(m/2)^2-4*Bo(m/2));
else
s(m) = -B1(m/2)/2+j*0.5*sqrt(-B1(m/2)^2+4*Bo(m/2));
end
end
if rem(m,2) == 1
if B1((m-1)/2)^2 >= 4*Bo((m-1)/2) s(m) = -B1((m-1)/2)/2-0.5*sqrt(B1((m-1)/2)^2-4*Bo((m-1)/2));
else
s(m) = conj(s(m-1));
end
end
end
else
for m = 1:2*r
if rem(m,2) == 1
if B1((m+1)/2)^2 >= 4*Bo((m+1)/2) s(m) = -B1((m+1)/2)/2+0.5*sqrt(B1((m+1)/2)^2-4*Bo((m+1)/2));
else
s(m) = -B1((m+1)/2)/2+j*0.5*sqrt(-B1((m+1)/2)^2+4*Bo((m+1)/2));
end
```

```
117.end
118.if rem(m,2) == 0
119.if B1(m/2)^2 >= 4*Bo(m/2) s(m) = -B1(m/2)/2-0.5*sqrt(B1(m/2)^2-4*Bo(m/2));
120.else
121.s(m) = conj(s(m-1));
122.end
123.end
124.end
125.end
126.end
127.% Calculating the normalized lowpass zero locations.
128.% Vector sz is the normalized zeros.
129.if N == 1 r = 1;
130.elseif rem(N,2) == 1
131.r = (N-1)/2;
132.else
133.r = N/2;
134.end
135.if rem(N,2) == 1 sz(1) = 10^50; end
136.if rem(N,2) == 1 & N > 1
137.for m = 1:2*r
138.if rem(m,2) == 1 sz(m+1) = j*sqrt(Ao((m+1)/2));
139.else
140.sz(m+1) = conj(sz(m));
141.end
142.end
143.end
144.if rem(N,2) == 0
145.for m = 1:2*r
146.if rem(m,2) == 1 sz(m) = j*sqrt(Ao((m+1)/2));
147.else
148.sz(m) = conj(sz(m-1));
149.end
150.end
151.end
152.disp('Normalized Lowpass Pole Locations')
153.disp(s)
154.disp('Normalized Lowpass Zero Locations')
155.disp(sz(1))
156.for i = 2:N
157.disp(sz(i))
158.end
159.%Adjustments
160.sqrtws = sqrt(Wslpf);
161.for i = 1:N
162.s(i) = sqrtws*s(i);
163.sz(i) = sqrtws*sz(i);
164.end
165.disp('Normalized Lowpass Pole Locations After Adjustments')
166.disp(s)
167.disp('Normalized Lowpass Zero Locations After Adjustments')
168.disp(sz(1))
169.for i = 2:N
170.disp(sz(i))
171.end
172.% Plotting the normalized lowpass poles and zeros.
173.for n = 1:201
174.x(n) = (n-101)/100;
175.ty(n) = sqrt(1-x(n).*x(n));
176.by(n) = - ty(n);
```

```
177.end
178.figure
179.plot(x,ty)
180.hold on
181.plot(x,by)
182.plot(real(s),imag(s),'bx');grid;
183.for i = 1:N
184.if real(sz(i)) < 10^3
185.plot(real(sz(i)),imag(sz(i)),'ro')
186.end
187.end
188.title('Normalized Lowpass Poles and Zeros')
189.xlabel('Real Part')
190.ylabel('Imaginary Part')
191.% Calculating the frequency transformed analog poles and zeros.
192.% Lowpass to bandstop transformation.
193.Wo=sqrt((w2bsf-w1bsf)/(w4bsf-w3bsf));
194.w21=(w2bsf-w1bsf)/2;
195.w2m1=w2bsf-w1bsf;
196.w12t4=4*w1bsf*w2bsf;
197.for k = 1:N
198.s(k) = s(k);
199.R1(k) = real(s(k));
200.I1(k) = imag(s(k));
201.R2(k) = R1(k)^2;
202.I2(k) = I1(k)^2;
203.mag(k) = R2(k) + I2(k);
204.A(k) = (I2(k)-R2(k))*w2m1^2/mag(k)^2 + w12t4;
205.B(k) = 2*abs(R1(k))*abs(I1(k))*w2m1^2/mag(k)^2;
206.RA(k) = (1/2)*sqrt((sqrt(A(k)^2+B(k)^2)-A(k))/2);
207.RB(k) = j*(1/2)*(sqrt((sqrt(A(k)^2+B(k)^2)+A(k))/2)-I1(k)*w2m1/mag(k));
208.end
209.for m = 1:2*N
210.if rem(m,2) == 1
211.if I1((m+1)/2) > 0 sfs(m) = w21*R1((m+1)/2)/mag((m+1)/2) + RA((m+1)/2) + RB((m+1)/2); end
212.if I1((m+1)/2) < 0 sfs(m) = w21*R1((m+1)/2)/mag((m+1)/2) - RA((m+1)/2) + RB((m+1)/2); end
213.R1a = (w2m1/R1((m+1)/2))^2;
214.if I1((m+1)/2) == 0 & R1a >= w12t4 sfs(m) = w21/R1((m+1)/2) + (1/2)*sqrt(R1a-w12t4); end
215.if I1((m+1)/2) == 0 & R1a < w12t4 sfs(m) = w21/R1((m+1)/2) + j*(1/2)*sqrt(-R1a+w12t4); end
216.else
217.sfs(m) = conj(sfs(m-1));
218.if I1(m/2) == 0 & (w2m1/R1(m/2))^2 >= w12t4 sfs(m) = w21/R1(m/2) -
    (1/2)*sqrt(w2m1^2/R1(m/2)^2-w12t4); end
219.if I1(m/2) == 0 & (w2m1/R1(m/2))^2 < w12t4 sfs(m) = w21/R1(m/2) - j*(1/2)*sqrt(-
    w2m1^2/R1(m/2)^2+w12t4); end
220.end
221.end
222.% Transformation of zeros
223.for k = 1:N
224.sz(k) = sz(k);
225.Rz1(k) = real(sz(k));
226.Iz1(k) = imag(sz(k));
227.Rz2(k) = Rz1(k)^2;
228.Iz2(k) = Iz1(k)^2;
229.mag(k) = Rz2(k) + Iz2(k);
230.Az(k) = (Iz2(k)-Rz2(k))*w2m1^2/mag(k)^2 + w12t4;
231.Bz(k) = 2*abs(Rz1(k))*abs(Iz1(k))*w2m1^2/mag(k)^2;
232.RAz(k) = (1/2)*sqrt((sqrt(Az(k)^2+Bz(k)^2)-Az(k))/2);
233.RBz(k) = j*(1/2)*(sqrt((sqrt(Az(k)^2+Bz(k)^2)+Az(k))/2)-Iz1(k)*w2m1/mag(k));
234.if Iz1(k) ~= 0 AAz(k) = ((w2bsf-w1bsf)/Iz1(k))^2 + w12t4; end
```

```
235.end
236.for m = 1:2*N
237.if rem(m,2) == 1
238.if Iz1((m+1)/2) >= 0 szfs(m) = w21*Rz1((m+1)/2)/mag((m+1)/2) + RAz((m+1)/2) + RBz((m+1)/2);
    end
239.if Iz1((m+1)/2) < 0 szfs(m) = w21*Rz1((m+1)/2)/mag((m+1)/2) - RAz((m+1)/2) + RBz((m+1)/2); end
240.if Iz1((m+1)/2) > 0 & Rz1((m+1)/2) == 0 szfs(m) = j*(1/2)*(sqrt(AAz((m+1)/2))-w2m1/Iz1((m+1)/2));
    end
241.if Iz1((m+1)/2) < 0 & Rz1((m+1)/2) == 0 szfs(m) = j*(1/2)*(sqrt(AAz((m+1)/2))-w2m1/Iz1((m+1)/2));
    end
242.else
243.szfs(m) = conj(szfs(m-1));
244.end
245.end
246.% Plotting the frequency transformed bandpass poles and zeros.
247.wo = sqrt(w1bsf*w2bsf)/Wo;
248.for na = 1:201
249.xa(na) = wo*(na-101)/100;
250.tya(na) = sqrt(wo^2-xa(na)^2);
251.bya(na) = - tya(na);
252.end
253.figure
254.plot(xa,tya)
255.hold on
256.plot(xa,bya)
257.plot(real(sfs),imag(sfs),'bx');grid;
258.for i = 1:2*N
259.if real(szfs(i)) < 10^30
260.plot(real(szfs(i)),imag(szfs(i)),'ro')
261.end
262.end
263.title('Frequency Transformed Bandstop Poles and Zeros')
264.xlabel('Real Part')
265.ylabel('Imaginary Part')
266.disp('Frequency Transformed Bandstop Pole Locations')
267.disp(sfs)
268.disp('Frequency Transformed Bandstop Zero Locations')
269.disp(szfs)
270.% Transforming analog poles and zeros to digital poles and zeros using bilinear transformation.
271.for m = 1:2*N
272.dp(m) = (1 + (Ts/2)*sfs(m))/(1 - (Ts/2)*sfs(m));
273.dz(m) = (1 + (Ts/2)*szfs(m))/(1 - (Ts/2)*szfs(m));
274.end
275.% Plotting digital poles and zeros on the complex z-plane
276.figure
277.plot(x,ty)
278.hold on
279.plot(x,by)
280.plot(real(dp),imag(dp),'bx')
281.plot(real(dz),imag(dz),'ro')
282.grid;
283.title('Digital Poles and Zeros')
284.xlabel('Real Part')
285.ylabel('Imaginary Part')
286.disp('Digital Pole Locations')
287.disp(dp)
288.disp('Digital Zero Locations')
289.disp(dz)
290.%
291.for i = 1:N
```

```
292.if rem(N,2) == 1
293.if i == 1
294.d2(i,1) = 1; d2(i,2) = -dp(1)-dp(2); d2(i,3) = (-dp(1))*(-dp(2));
295.n2(i,1) = 1; n2(i,2) = -dz(1)-dz(2); n2(i,3) = (-dz(1))*(-dz(2));
296.else
297.d2(i,1)=1; d2(i,2)=-2*real(dp(2*i-1));d2(i,3) = (abs(dp(2*i-1)))^2;
298.n2(i,1)=1; n2(i,2)=-2*real(dz(2*i-1));n2(i,3) = (abs(dz(2*i-1)))^2;
299.end
300.else
301.d2(i,1)=1; d2(i,2)=-2*real(dp(2*i-1));d2(i,3) = (abs(dp(2*i-1)))^2;
302.n2(i,1)=1; n2(i,2)=-2*real(dz(2*i-1));n2(i,3) = (abs(dz(2*i-1)))^2;
303.end
304.end
305.disp('Numerator Second Order Terms')
306.disp(n2)
307.disp('Denominator Second Order Terms')
308.disp(d2)
309.% Plotting the magnitude response and the phase response with respect to the normalized digital
    frequency.
310.M = 128;  % M is the number of points the frequency response is calculated.
311.step = 2*pi/M;
312.for m = 1:M
313.H(m)  = (1-dz(1)*exp(j*(-1)*(m-1)*step))*(1-dz(2)*exp(j*(-1)*(m-1)*step))/[(1-dp(1)*exp(j*(-1)*(m-
    1)*step))*(1-dp(2)*exp(j*(-1)*(m-1)*step))];
314.end
315.if N > 1
316.for n = 3:2*N
317.for m = 1:M
318.H(m) = H(m).*[(1-dz(n)*exp(j*(-1)*(m-1)*step))/(1-dp(n)*exp(j*(-1)*(m-1)*step))];
319.end
320.end
321.end
322.max = abs(H(1));
323.for m = 2:M
324.if abs(H(m)) > max max = abs(H(m)); end
325.end
326.disp('maximum gain is')
327.disp(max)
328.b01=1/max;
329.disp('1/max = ')
330.disp(b01);
331.%Finding b0
332.H0 = (1 - dz(1))/(1 - dp(1));
333.for i = 2:2*N
334.H0 = H0*(1 - dz(i))/(1 - dp(i));
335.end
336.if rem(N,2) == 1
337.b0 = 1/real(H0);
338.else
339.b0 = 10^(-0.05*amax)/real(H0);
340.end
341.disp('Constant Term b0 =')
342.disp(b0);
343.%Polynomial Form
344.nump(1) = n2(1,1); nump(2) = n2(1,2); nump(3) = n2(1,3);
345.denp(1) = d2(1,1); denp(2) = d2(1,2); denp(3) = d2(1,3);
346.if N > 1
347.for i=2:N nump = conv(nump,n2(i,:)); denp = conv(denp,d2(i,:)); end
348.end
349.disp('Numerator Polynomial')
```

```
350.disp(b0*nump)
351.disp('Denominator Polynomial')
352.disp(denp)
353.for m = 1:M
354.f(m) = (m-1)/M;
355.H(m) = H(m)/max;
356.mag(m) = abs(H(m));
357.if H(m) == 0 dB(m) = -200; else dB(m) = 20*log10(mag(m)); end
358.if dB(m) < -200 dB(m) = -200; end
359.pha(m) = angle(H(m));
360.end
361.figure
362.semilogx(f*fs,mag);
363.axis([0  100  0  1]);grid;
364.title('Elliptic Magnitude Response')
365.xlabel('Frequency(Hz)')
366.ylabel('Magnitude(Volt.)')
367.figure
368.semilogx(f*fs,dB ,'-r');
369.axis([0  100  -40  0]);grid;
370.title('Elliptic Magnitude Response in dB')
371.xlabel('Frequency')
372.ylabel('Magnitude in dB')
373.figure
374.semilogx(f*fs,pha*180/pi,'-r');
375.axis([0  100  -180  180]); grid;
376.title('Phase Response')
377.xlabel('Frequency(Hz)')
378.ylabel('Phase in Degree')
379.% Calculating and Plotting the Impulse Response of the Filter.
380.h = ifft(H);
381.for m = 1:M
382.k(m) = m-1;
383.z(m) = 0;
384.if real(h(m)) >= 0 Lo(m) = 0; else Lo(m) = -real(h(m)); end
385.if real(h(m)) >= 0 Hi(m) = real(h(m)); else Hi(m) = 0; end
386.end
387.figure
388.%errorbar(k,z,Lo,Hi,'y')
389.stem(k,h); grid;
390.title('Impulse Response')
391.xlabel('Time')
392.ylabel('Amplitude')
393.%Verifying
394.Hq1 =abs(1 - dz(1)*exp(-j*q1bsf))/abs(1 - dp(1)*exp(-j*q1bsf));
395.Hq2 =abs(1 - dz(1)*exp(-j*q2bsf))/abs(1 - dp(1)*exp(-j*q2bsf));
396.Hq3 =abs(1 - dz(1)*exp(-j*q3bsf))/abs(1 - dp(1)*exp(-j*q3bsf));
397.Hq3a =abs(1 - dz(1)*exp(-j*q3absf))/abs(1 - dp(1)*exp(-j*q3absf));
398.Hq4 =abs(1 - dz(1)*exp(-j*q4bsf))/abs(1 - dp(1)*exp(-j*q4bsf));
399.Hq4a =abs(1 - dz(1)*exp(-j*q4absf))/abs(1 - dp(1)*exp(-j*q4absf));
400.for p = 2:2*N
401.Hq1 =Hq1*abs(1 - dz(p)*exp(-j*q1bsf))/abs(1 - dp(p)*exp(-j*q1bsf));
402.Hq2 =Hq2*abs(1 - dz(p)*exp(-j*q2bsf))/abs(1 - dp(p)*exp(-j*q2bsf));
403.Hq3 =Hq3*abs(1 - dz(p)*exp(-j*q3bsf))/abs(1 - dp(p)*exp(-j*q3bsf));
404.Hq3a =Hq3*abs(1 - dz(p)*exp(-j*q3absf))/abs(1 - dp(p)*exp(-j*q3absf));
405.Hq4 =Hq4*abs(1 - dz(p)*exp(-j*q4bsf))/abs(1 - dp(p)*exp(-j*q4bsf));
406.Hq4a =Hq4a*abs(1 - dz(p)*exp(-j*q4absf))/abs(1 - dp(p)*exp(-j*q4absf));
407.end
408.Hq1 = -20*log10(abs(b0*Hq1));
409.Hq2 = -20*log10(abs(b0*Hq2));
```

```
410.Hq3 = -20*log10(abs(b0*Hq3));
411.Hq3a = -20*log10(abs(b0*Hq3a));
412.Hq4 = -20*log10(abs(b0*Hq4));
413.Hq4a = -20*log10(abs(b0*Hq4a));
414.disp('Loss in dB at the lower passband cutoff frequency (q1bsf)')
415.disp(Hq1)
416.disp('Loss in dB at the upper passband cutoff frequency (q2bsf)')
417.disp(Hq2)
418.disp('Loss in dB at the lower stopband cutoff frequency (q3bsf)')
419.disp(Hq3)
420.disp('Loss in dB at the upper stopband cutoff frequency (q4bsf)')
421.disp(Hq4)
422.if q3absf ~= q3bsf
423.disp('Loss in dB at the lower stopband cutoff frequency adjusted (q3absf)')
424.disp(Hq3a)
425.end
426.if q4absf ~= q4bsf
427.disp('Loss in dB at the upper stopband cutoff frequency adjusted (q4absf)')
428.disp(Hq4a)
429.end
```

ภาคผนวก ค.

VHDL Code

Accumulator.vhd

```
1     library IEEE;
2       use IEEE.std_logic_1164.all;
3       use IEEE.std_logic_arith.all;
4
5     entity accumulator is
6        Port (
7               RST       : In std_logic; -- reset
8               MCLK      : In std_logic; -- 192Fs
9               A_IN      : In std_logic_vector(19 downto 0); -- an input to adder
10              ACCLAT    : In std_logic; -- accumulation latch
11              ACCCL     : In std_logic; -- accumulation clear
12              ACCENB    : In std_logic; -- accumulation enable
13              ACC_OUT   : Out std_logic_vector(11 downto 0) ); -- accumulated data
14    end accumulator;
15
16    architecture STRUCTURE of accumulator is
17
18      signal A_tmp      : signed( 19 downto 0 );
19      signal B_tmp      : signed( 23 downto 0 );
20      signal C_tmp      : signed( 23 downto 0 );
21      signal ACC_tmp    : signed( 23 downto 0 );
22      signal acc_round  : signed( 23 downto 0 );
23
24    begin -- architecture accumulator
25
26    ---------------------------------------------------------------------
27      gen_A_tmp : process( A_IN )
28      begin -- process gen_A_tmp
29
30        for i in 19 downto 0 loop
31          A_tmp(i) <= A_IN(i);
32        end loop;
33
34      end process gen_A_tmp;
35
36    ---------------------------------------------------------------------
37      B_tmp <= ACC_tmp;
38
39    ---------------------------------------------------------------------
40      C_tmp <= A_tmp + B_tmp; -- adder
41
42    ---------------------------------------------------------------------
43    gen_ACC_tmp : process( RST, MCLK, ACCCL ) -- accumulator
44    begin -- process gen_ACC_tmp
45
46      if(RST = '1' or ACCCL = '1') then
47        ACC_tmp <= (others => '0');
48      elsif( MCLK'event and MCLK = '1' ) then
49        if( ACCENB = '1' ) then
50          ACC_tmp <= C_tmp;
51        end if;
52      end if;
53
54    end process gen_ACC_tmp;
55
56    ---------------------------------------------------------------------
57      acc_round <= C_tmp + 32; -- to round it on sixth bit
58
59    ---------------------------------------------------------------------
60    gen_ACC_OUT : process( RST, MCLK ) -- to pick up accumulated data
61    begin -- process gen_ACC_OUT
62
63      if( RST = '1' ) then
64        ACC_OUT <= (others => '0');
65      elsif( MCLK'event and MCLK = '1' ) then
66        if ( ACCLAT = '1' ) then
67          if (acc_round(23) = '0' and
68                (acc_round(22) = '1'
```

```
69              or acc_round(21) = '1'
70              or acc_round(20) = '1'
71              or acc_round(19) = '1'
72              or acc_round(18) = '1'
73              or acc_round(17) = '1')) then -- overflow
74      --        ACC_OUT <= "01111111";
75            ACC_OUT <= (others => '1');
76            ACC_OUT(11) <= '0';
77          elsif (acc_round(23) = '1' and
78                 (acc_round(22) = '0'
79              or acc_round(21) = '0'
80              or acc_round(20) = '0'
81              or acc_round(19) = '0'
82              or acc_round(18) = '0'
83              or acc_round(17) = '0')) then -- underflow
84      --        ACC_OUT <= "10000000";
85            ACC_OUT <= (others => '0');
86            ACC_OUT(11) <= '1';
87          else
88            for i in 11 downto 0 loop
89              ACC_OUT(i) <= acc_round(i + 6);
90            end loop;
91          end if;
92        end if;
93      end if;
94
95    end process gen_ACC_OUT;
96
97    end; -- architecture accumulator
```

Divider.vhd

```
1     library IEEE;
2       use IEEE.std_logic_1164.all;
3       use IEEE.std_logic_arith.all;
4
5     entity divider is
6        Port (
7               RST        : In std_logic; -- reset
8               FSCLK      : In std_logic; -- Fs (a clock synchronized with input data)
9               MCLK       : In std_logic; -- 384FSCLK or faster (master clock)
10              CO16       : Out std_logic_vector(3 downto 0); -- Jul/25/00
11              MULSTART   : Out std_logic;
12              MULEND     : Out std_logic;
13              ACCLAT     : Out std_logic; -- accumulation latch
14              ACCCL      : Out std_logic; -- accumulation clear
15              ACCENB     : Out std_logic ); -- accumulation enable
16    end divider;
17
18    architecture STRUCTURE of divider is
19
20      signal counter_12_16   : unsigned( 3 downto 0 );
21      signal counter_32      : unsigned( 4 downto 0 );
22      signal counter_2       : unsigned( 1 downto 0 );
23      signal fsclk_delay1    : std_logic;
24      signal fsclk_delay2    : std_logic;
25      signal fsclk_edge_dummy: std_logic;
26
27    begin -- architecture divider
28
29    ------------------------------------------------------------
30    gen_CO16 : process( counter_12_16 ) -- output for dly_mux_coeff
31    begin -- process gen_CO16
32
33      for i in 3 downto 0 loop
34        CO16(i) <= counter_12_16(i);
35      end loop;
36
37    end process gen_CO16;
38
39    ------------------------------------------------------------
40    gen_counter_12_16 : process( RST, MCLK ) -- counter to generate ACCENB
41    begin -- process gen_counter_12_16
42
43      if(RST = '1') then
44        counter_12_16 <= "0000";
45      elsif( MCLK'event and MCLK = '1' ) then
46        if(fsclk_edge_dummy = '1') then
47          counter_12_16 <= "0000";
48        elsif(counter_32 = 31) then
49          counter_12_16 <= counter_12_16 + 1;
50        end if;
51      end if;
52
53    end process gen_counter_12_16;
54
55    ------------------------------------------------------------
56    gen_MULSTART : process( counter_12_16, counter_32 ) -- multiplication start
57    begin -- process gen_MULSTART
58
59      if( counter_12_16 < 5 and counter_32 = 0 ) then
60        MULSTART <= '1';
61      else
62        MULSTART <= '0';
63      end if;
64
65    end process gen_MULSTART;
66
67    ------------------------------------------------------------
68    gen_MULEND : process( counter_12_16, counter_32 ) -- multiplication end
69    begin -- process gen_MULEND
70
71      if( counter_12_16 < 5
72          and counter_32 = 11 ) then
73        MULEND <= '1';
74      else
75        MULEND <= '0';
76      end if;
77
78    end process gen_MULEND;
79
80    ------------------------------------------------------------
```

```
81    gen_ACCENB : process( counter_12_16, counter_32 )
82    begin -- process gen_ACCENB
83
84      if( counter_12_16 > 0 and counter_12_16 < 6
85          and counter_32 = 11 ) then
86        ACCENB <= '1';
87      else
88        ACCENB <= '0';
89      end if;
90
91    end process gen_ACCENB;
92
93    -------------------------------------------------------------
94    gen_ACCLAT : process( counter_12_16, counter_32 )
95    begin -- process gen_ACCLAT
96
97      if( counter_12_16 = 5
98          and counter_32 = 11 ) then
99        ACCLAT <= '1';
100     else
101       ACCLAT <= '0';
102     end if;
103
104   end process gen_ACCLAT;
105
106   -------------------------------------------------------------
107   gen_ACCCL : process( counter_12_16, counter_32 )
108   begin -- process gen_ACCCL
109
110     if( counter_12_16 = 1 and counter_32 = 0 ) then
111       ACCCL <= '1';
112     else
113       ACCCL <= '0';
114     end if;
115
116   end process gen_ACCCL;
117
118   -------------------------------------------------------------
119   gen_fsclk_delay : process( RST, MCLK )
120   begin -- process gen_fsclk_delay
121
122     if( RST = '1' ) then
123       fsclk_delay1 <= '0';
124       fsclk_delay2 <= '0';
125     elsif( MCLK'event and MCLK = '1' ) then
126       fsclk_delay1 <= FSCLK;
127       fsclk_delay2 <= fsclk_delay1;
128     end if;
129
130   end process gen_fsclk_delay;
131
132   -------------------------------------------------------------
133   fsclk_edge_dummy <= fsclk_delay1 and not fsclk_delay2;
134
135   -------------------------------------------------------------
136   gen_counter_32 : process( RST, MCLK ) -- counter to generate ACCCLK
137   begin -- process gen_counter_32
138
139     if(RST = '1') then
140       counter_32 <= "00000";
141     elsif( MCLK'event and MCLK = '1' ) then
142       if(counter_32 = 31 or fsclk_edge_dummy = '1') then
143         counter_32 <= "00000";
144       else
145         counter_32 <= counter_32 + 1;
146       end if;
147     end if;
148
149   end process gen_counter_32;
150
151   -------------------------------------------------------------
152   end; -- architecture divider
```

Dly_mux_coeff.vhd

```
1     library IEEE;
2       use IEEE.std_logic_1164.all;
3       use IEEE.std_logic_arith.all;
4
5     entity dly_mux_coeff is
6        Port (
7               CO16       : In std_logic_vector(3 downto 0); -- Jul/25/0
8               RST        : In std_logic; -- reset
9               FSCLK      : In std_logic; -- Fs (a clock synchronized with input data)
10              D_IN       : In std_logic_vector(11 downto 0); -- data input
11              ACC_IN     : In std_logic_vector(11 downto 0); -- from accumulator
12              A0         : In std_logic_vector( 9 downto 0); -- coeff a0
13              A1         : In std_logic_vector( 9 downto 0); -- coeff a1
14              A2         : In std_logic_vector( 9 downto 0); -- coeff a2
15              B1         : In std_logic_vector( 9 downto 0); -- coeff b1
16              B2         : In std_logic_vector( 9 downto 0); -- coeff b2
17              COEFF_OUT  : Out std_logic_vector( 9 downto 0); -- coefficients
18              MUX_OUT    : Out std_logic_vector(11 downto 0); -- multiplexed data
19              D_OUT      : Out std_logic_vector(11 downto 0) ); -- filterd data output
20    end dly_mux_coeff;
21
22    architecture STRUCTURE of dly_mux_coeff is
23
24      signal d_a1, d_a2, d_b1, d_b2 : std_logic_vector( 11 downto 0 );
25
26    begin -- architecture dly_mux_coeff
27
28    ------------------------------------------------------------------
29    gen_d_a : process( RST, FSCLK ) -- delay input data (corresponds to z^(-1)
30    begin -- process gen_d_a
31
32      if(RST = '1') then
33        d_a1 <= (others => '0');
34        d_a2 <= (others => '0');
35      elsif( FSCLK'event and FSCLK = '1' ) then
36        d_a1 <= D_IN;
37        d_a2 <= d_a1;
38      end if;
39
40    end process gen_d_a;
41
42    ------------------------------------------------------------------
43    gen_d_b : process( RST, FSCLK ) -- delay output data (corresponds to z^(-1)
44    begin -- process gen_d_b
45
46      if(RST = '1') then
47        d_b1 <= (others => '0');
48        d_b2 <= (others => '0');
49      elsif( FSCLK'event and FSCLK = '1' ) then
50        d_b1 <= ACC_IN;
51        d_b2 <= d_b1;
52      end if;
53
54    end process gen_d_b;
55
56    ------------------------------------------------------------------
57      gen_D_OUT : process( d_b1 ) -- filtered output
58      begin -- process gen_D_OUT
59
60        for i in 11 downto 0 loop
61          D_OUT(i) <= d_b1(i);
62        end loop;
63
64      end process gen_D_OUT;
65
66    ------------------------------------------------------------
67      gen_COEFF_OUT : process( CO16, D_IN, d_a1, d_a2, d_b1, d_b2, A0, A1, A2, B1, B2 ) --
      multiplex coefficients because we have only one multiplier
```

```
68      begin -- process gen_COEFF_OUT
69
70        if( CO16 = "0000" ) then
71          COEFF_OUT <= A0;
72          MUX_OUT <= D_IN;
73        elsif( CO16 = "0001" ) then
74          COEFF_OUT <= A1;
75          MUX_OUT <= d_a1;
76        elsif( CO16 = "0010" ) then
77          COEFF_OUT <= A2;
78          MUX_OUT <= d_a2;
79        elsif( CO16 = "0011" ) then
80          COEFF_OUT <= B1; -- b1 needs to be multiplied by (-1)
81          MUX_OUT <= d_b1;
82        elsif( CO16 = "0100" ) then
83          COEFF_OUT <= B2; -- b2 needs to be multiplied by (-1)
84          MUX_OUT <= d_b2;
85        else
86          COEFF_OUT <= (others => '0');
87          MUX_OUT <= (others => '0');
88        end if;
89
90      end process gen_COEFF_OUT;
91
92    ------------------------------------------------------------------
93
94    end; -- architecture dly_mux_coeff
```

Multiplier_mxn.vhd

```
1     library IEEE;
2       use IEEE.std_logic_1164.all;
3       use IEEE.std_logic_arith.all;
4
5     entity multiplier_mxn is
6        Port (
7         RST             : In std_logic; -- reset
8         COEFF_IN        : In std_logic_vector(  9 downto 0 ); -- coefficients
9         A_IN       : In std_logic_vector( 11 downto 0 ); -- multiplexed data
10        MULSTART        : In std_logic; -- multiplication start
11        MULEND          : In std_logic; -- multiplication end
12        MCLK       : In std_logic; -- 384FSCLK or faster
13        MULTIPLIED_OUT : Out std_logic_vector( 19 downto 0 ) ); -- multiplication results
14    end multiplier_mxn;
15
16    architecture STRUCTURE of multiplier_mxn is
17
18      signal abs_COEFF_IN    : unsigned(  8 downto 0 );
19      signal abs_A_IN        : unsigned( 10 downto 0 );
20      signal shifted_A_IN    : unsigned( 19 downto 0 );
21      signal sigma           : unsigned( 19 downto 0 );
22      signal sigma_delay     : unsigned( 19 downto 0 );
23      signal multi_start     : std_logic;
24      signal multiplied_flag : std_logic;
25
26    type state_type is ( S_start,
27                         S_sigma0, S_sigma1, S_sigma2, S_sigma3,
28                         S_sigma4, S_sigma5, S_sigma6, S_sigma7,
29                         S_sigma8, S_sigma9, S_sigma10, S_sigma11,
30                         S_sigma12, S_sigma13, S_sigma14, S_sigma15,
31                         S_sigma16, S_sigma17, S_sigma18, S_sigma19,
32                         S_sigma20, S_sigma21, S_sigma22, S_sigma23,
33                         S_sigma24, S_end );
34      signal current_state : state_type;
35
36    begin -- architecture multiplier_mxn
37
38    ------------------------------------------------------------------
39    gen_multiplied_flag : process( COEFF_IN, A_IN ) -- check polarity of data
40    variable all_zero_c, all_zero_d : boolean;
41    begin -- process gen_multiplied_flag
42
43      all_zero_c := true;
44      all_zero_d := true;
45      for i in  9 downto 0 loop
46        if( COEFF_IN(i) = '1' ) then
47          all_zero_c := false;
48        end if;
49      end loop;
50      for i in 11 downto 0 loop
51        if( A_IN(i) = '1' ) then
52          all_zero_d := false;
53        end if;
54      end loop;
55
56      if ( all_zero_c or all_zero_d ) then
57        multiplied_flag <= '0';
58      else
59        multiplied_flag <= COEFF_IN(  9 ) xor A_IN( 11 );
60      end if;
61
62    end process gen_multiplied_flag;
63    ------------------------------------------------------------------
64      multi_start <= MULSTART;
65
66
67    ------------------------------------------------------------------
68      gen_abs_COEFF_IN : process( COEFF_IN ) -- convert coefficients to absolute value
69      variable tmp_abs : unsigned(  8 downto 0 );
70      variable max_abs : unsigned(  8 downto 0 );
71      begin -- process gen_abs_COEFF_IN
72
73        for i in  8 downto 0 loop
74          max_abs(i) := '1';
75        end loop;
76
77        for i in  8 downto 0 loop -- initialize
78          abs_COEFF_IN(i) <= '0';
79        end loop;
80
```

```
81        if( COEFF_IN( 9) = '0' ) then
82          for i in  8 downto 0 loop
83            abs_COEFF_IN(i) <= COEFF_IN(i);  .
84          end loop;
85        end if;
86        if( COEFF_IN( 9) = '1' ) then
87          for i in  8 downto 0 loop
88            tmp_abs(i) := not COEFF_IN(i);
89          end loop;
90
91          if (tmp_abs /= max_abs) then
92            tmp_abs := tmp_abs + 1;
93          end if;
94
95          for i in  8 downto 0 loop
96            abs_COEFF_IN(i) <= tmp_abs(i);
97          end loop;
98        end if;
99
100     end process gen_abs_COEFF_IN;
101
102   ------------------------------------------------------------------
103     gen_abs_A_IN : process( A_IN ) -- convert the input data to absolute value
104     variable tmp_abs : unsigned( 10 downto 0 );
105     variable max_abs : unsigned( 10 downto 0 );
106     begin -- process gen_abs_A_IN
107
108       for i in 10 downto 0 loop
109         max_abs(i) := '1';
110       end loop;
111
112       for i in 10 downto 0 loop -- initialize
113         abs_A_IN(i) <= '0';
114       end loop;
115
116       if( A_IN(11) = '0' ) then
117         for i in 10 downto 0 loop
118           abs_A_IN(i) <= A_IN(i);
119         end loop;
120       end if;
121       if( A_IN(11) = '1' ) then
122         for i in 10 downto 0 loop
123           tmp_abs(i) := not A_IN(i);
124         end loop;
125
126         if (tmp_abs /= max_abs) then
127           tmp_abs := tmp_abs + 1;
128         end if;
129
130         for i in 10 downto 0 loop
131           abs_A_IN(i) <= tmp_abs(i);
132         end loop;
133       end if;
134
135     end process gen_abs_A_IN;
136
137   ------------------------------------------------------------------
138     gen_shifted_A_IN : process( RST, MCLK )
139     variable bit_num : integer;
140     begin -- process gen_shifted_A_IN
141
142       if( RST = '1' ) then
143         current_state <= S_start;
144       elsif( MCLK'event and MCLK = '1' ) then
145         shifted_A_IN <= (others => '0');
146
147         case current_state is
148         when S_start =>
149         current_state <= S_sigma0;
150
151         when S_sigma0 =>
152         bit_num := 0;
153           if(bit_num >  8) then bit_num :=  8; end if;
154         if( abs_COEFF_IN(bit_num) = '1' ) then
155           for i in 10 downto 0 loop
156             shifted_A_IN(i + bit_num) <= abs_A_IN(i);
157             end loop;
158           end if;
159           if( 10 = 2 ) then
160             current_state <= S_end;
```

```
161           else
162          current_state <= S_sigma1;
163           end if;
164
165         when S_sigma1 =>
166        bit_num := 1;
167           if(bit_num >  8) then bit_num :=  8; end if;
168        if( abs_COEFF_IN(bit_num) = '1' ) then
169          for i in 10 downto 0 loop
170            shifted A IN(i + bit_num) <= abs_A_IN(i);
171             end loop;
172           end if;
173           if( 10 = 3 ) then
174             current_state <= S_end;
175           else
176          current_state <= S_sigma2;
177           end if;
178
179         when S_sigma2 =>
180        bit_num := 2;
181           if(bit_num >  8) then bit_num :=  8; end if;
182        if( abs_COEFF_IN(bit_num) = '1' ) then
183          for i in 10 downto 0 loop
184            shifted_A_IN(i + bit_num) <= abs_A_IN(i);
185             end loop;
186           end if;
187           if( 10 = 4 ) then
188             current_state <= S_end;
189           else
190          current_state <= S_sigma3;
191           end if;
192
193         when S sigma3 =>
194        bit_num := 3;
195           if(bit_num >  8) then bit_num :=  8; end if;
196        if( abs_COEFF_IN(bit_num) = '1' ) then
197          for i in 10 downto 0 loop
198            shifted_A_IN(i + bit_num) <= abs_A_IN(i);
199             end loop;
200           end if;
201           if( 10 = 5 ) then
202             current_state <= S_end;
203           else
204          current_state <= S_sigma4;
```

```
205           end if;
206
207          when S_sigma4 =>
208         bit_num := 4;
209            if(bit_num >  8) then bit_num :=  8; end if;
210         if( abs_COEFF_IN(bit_num) = '1' ) then
211           for i in 10 downto 0 loop
212             shifted_A_IN(i + bit_num) <= abs_A_IN(i);
213              end loop;
214            end if;
215            if( 10 = 6 ) then
216              current_state <= S_end;
217            else
218           current_state <= S_sigma5;
219            end if;
220
221          when S_sigma5 =>
222         bit_num := 5;
223            if(bit_num >  8) then bit_num :=  8; end if;
224         if( abs_COEFF_IN(bit_num) = '1' ) then
225           for i in 10 downto 0 loop
226             shifted_A_IN(i + bit_num) <= abs_A_IN(i);
227              end loop;
228            end if;
229            if( 10 = 7 ) then
230              current_state <= S_end;
231            else
232           current_state <= S_sigma6;
233            end if;
234
235          when S_sigma6 =>
236         bit_num := 6;
237            if(bit_num >  8) then bit_num :=  8; end if;
238         if( abs COEFF IN(bit num) = '1' ) then
239           for i in 10 downto 0 loop
240             shifted_A_IN(i + bit_num) <= abs_A_IN(i);
241              end loop;
242            end if;
243            if( 10 = 8 ) then
244              current_state <= S_end;
245            else
246           current state <= S_sigma7;
247            end if;
248
249          when S_sigma7 =>
250         bit_num := 7;
251            if(bit_num >  8) then bit_num :=  8; end if;
252         if( abs_COEFF_IN(bit_num) = '1' ) then
253           for i in 10 downto 0 loop
254             shifted_A_IN(i + bit_num) <= abs_A_IN(i);
255              end loop;
256            end if;
257            if( 10 = 9 ) then
258              current_state <= S_end;
259            else
260           current_state <= S_sigma8;
261            end if;
262
263          when S_sigma8 =>
264         bit_num := 8;
265            if(bit_num >  8) then bit_num :=  8; end if;
266         if( abs_COEFF_IN(bit_num) = '1' ) then
267           for i in 10 downto 0 loop
268             shifted A IN(i + bit_num) <= abs_A_IN(i);
269              end loop;
270            end if;
271            if( 10 = 10 ) then
272              current_state <= S_end;
```

```
273           else
274           current_state <= S_sigma9;
275           end if;
276
277          when S_sigma9 =>
278         bit_num := 9;
279            if(bit_num >  8) then bit_num :=  8; end if;
280         if( abs_COEFF_IN(bit_num) = '1' ) then
281           for i in 10 downto 0 loop
282             shifted_A_IN(i + bit_num) <= abs_A_IN(i);
283             end loop;
284           end if;
285           if( 10 = 11 ) then
286             current_state <= S_end;
287           else
288           current_state <= S_sigma10;
289           end if;
290
291          when S_sigma10 =>
292         bit_num := 10;
293            if(bit_num >  8) then bit_num :=  8; end if;
294         if( abs_COEFF_IN(bit_num) = '1' ) then
295           for i in 10 downto 0 loop
296             shifted_A_IN(i + bit_num) <= abs_A_IN(i);
297             end loop;
298           end if;
299           if( 10 = 12 ) then
300             current_state <= S_end;
301           else
302           current_state <= S_sigma11;
303           end if;
304
305          when S_sigma11 =>
306         bit_num := 11;
307            if(bit_num >  8) then bit_num :=  8; end if;
308         if( abs_COEFF_IN(bit_num) = '1' ) then
309           for i in 10 downto 0 loop
310             shifted_A_IN(i + bit_num) <= abs_A_IN(i);
311             end loop;
312           end if;
313           if( 10 = 13 ) then
314             current_state <= S_end;
315           else
316           current_state <= S_sigma12;
317           end if;
318
319          when S_sigma12 =>
320         bit_num := 12;
321            if(bit_num >  8) then bit_num :=  8; end if;
322         if( abs_COEFF_IN(bit_num) = '1' ) then
323           for i in 10 downto 0 loop
324             shifted_A_IN(i + bit_num) <= abs_A_IN(i);
325             end loop;
326           end if;
327           if( 10 = 14 ) then
328             current_state <= S_end;
329           else
330           current_state <= S_sigma13;
331           end if;
332
333          when S_sigma13 =>
334         bit_num := 13;
335            if(bit_num >  8) then bit_num :=  8; end if;
336         if( abs_COEFF_IN(bit_num) = '1' ) then
337           for i in 10 downto 0 loop
338             shifted_A_IN(i + bit_num) <= abs_A_IN(i);
339             end loop;
340           end if;
```

```
341           if( 10 = 15 ) then
342             current_state <= S_end;
343           else
344          current state <= S_sigma14;
345           end if;
346
347         when S_sigma14 =>
348        bit_num := 14;
349           if(bit_num >  8) then bit_num :=  8; end if;
350        if( abs_COEFF_IN(bit_num) = '1' ) then
351          for i in 10 downto 0 loop
352           shifted A IN(i + bit_num) <= abs_A_IN(i);
353            end loop;
354           end if;
355           if( 10 = 16 ) then
356             current_state <= S_end;
357           else
358          current_state <= S_sigma15;
359           end if;
360
361         when S_sigma15 =>
362        bit_num := 15;
363           if(bit_num >  8) then bit_num :=  8; end if;
364        if( abs_COEFF_IN(bit_num) = '1' ) then
365          for i in 10 downto 0 loop
366           shifted_A_IN(i + bit_num) <= abs_A_IN(i);
367            end loop;
368           end if;
369           if( 10 = 17 ) then
370             current_state <= S_end;
371           else
372          current_state <= S_sigma16;
373           end if;
374
375         when S_sigma16 =>
376        bit_num := 16;
377           if(bit_num >  8) then bit_num :=  8; end if;
378        if( abs_COEFF_IN(bit_num) = '1' ) then
379          for i in 10 downto 0 loop
380            shifted_A_IN(i + bit_num) <= abs_A_IN(i);
381             end loop;
382           end if;
383           if( 10 = 18 ) then
384             current_state <= S_end;
385           else
386          current_state <= S_sigma17;
387           end if;
388
389         when S sigma17 =>
390        bit_num := 17;
391           if(bit_num >  8) then bit_num :=  8; end if;
392        if( abs_COEFF_IN(bit_num) = '1' ) then
393          for i in 10 downto 0 loop
394            shifted_A_IN(i + bit_num) <= abs_A_IN(i);
395             end loop;
396           end if;
397           if( 10 = 19 ) then
398             current_state <= S_end;
399           else
400          current_state <= S_sigma18;
401           end if;
402
403         when S_sigma18 =>
404        bit num := 18;
405           if(bit_num >  8) then bit_num :=  8; end if;
406        if( abs_COEFF_IN(bit_num) = '1' ) then
407          for i in 10 downto 0 loop
408            shifted_A_IN(i + bit_num) <= abs_A_IN(i);
```

```
409             end loop;
410           end if;
411           if( 10 = 20 ) then
412             current_state <= S_end;
413           else
414          current_state <= S_sigma19;
415           end if;
416
417         when S_sigma19 =>
418        bit_num := 19;
419           if(bit_num >  8) then bit_num :=  8; end if;
420        if( abs COEFF IN(bit num) = '1' ) then
421          for i in 10 downto 0 loop
422            shifted_A_IN(i + bit_num) <= abs_A_IN(i);
423             end loop;
424           end if;
425           if( 10 = 21 ) then
426             current_state <= S_end;
427           else
428          current_state <= S_sigma20;
429           end if;
430
431         when S_sigma20 =>
432        bit_num := 20;
433           if(bit_num >  8) then bit_num :=  8; end if;
434        if( abs_COEFF_IN(bit_num) = '1' ) then
435          for i in 10 downto 0 loop
436            shifted_A_IN(i + bit_num) <= abs_A_IN(i);
437             end loop;
438           end if;
439           if( 10 = 22 ) then
440             current_state <= S_end;
441           else
442          current state <= S_sigma21;
443           end if;
444
445         when S_sigma21 =>
446        bit_num := 21;
447           if(bit_num >  8) then bit_num :=  8; end if;
448        if( abs_COEFF_IN(bit_num) = '1' ) then
449          for i in 10 downto 0 loop
450            shifted A IN(i + bit_num) <= abs_A_IN(i);
451             end loop;
452           end if;
453           if( 10 = 23 ) then
454             current_state <= S_end;
455           else
456          current_state <= S_sigma22;
457           end if;
458
459         when S_sigma22 =>
460        bit_num := 22;
461           if(bit_num >  8) then bit_num :=  8; end if;
462        if( abs_COEFF_IN(bit_num) = '1' ) then
463          for i in 10 downto 0 loop
464            shifted_A_IN(i + bit_num) <= abs_A_IN(i);
465             end loop;
466           end if;
467           if( 10 = 24 ) then
468             current_state <= S_end;
469           else
470          current_state <= S_sigma23;
471           end if;
472
473         when S_sigma23 =>
474        bit_num := 23;
475           if(bit_num >  8) then bit_num :=  8; end if;
476        if( abs_COEFF_IN(bit_num) = '1' ) then
477          for i in 10 downto 0 loop
478            shifted_A_IN(i + bit_num) <= abs_A_IN(i);
479             end loop;
480           end if;
```

```
481           if( 10 = 25 ) then
482             current_state <= S_end;
483           else
484             current_state <= S_sigma24;
485           end if;
486
487         when S_sigma24 =>
488        bit_num := 24;
489           if(bit_num >  8) then bit_num :=  8; end if;
490        if( abs_COEFF_IN(bit_num) = '1' ) then
491          for i in 10 downto 0 loop
492            shifted_A_IN(i + bit_num) <= abs_A_IN(i);
493             end loop;
494           end if;
495        current_state <= S_end;
496
497         when S_end =>
498        if(multi_start = '1') then
499          current_state <= S_start;
500           else
501          current_state <= S_end;
502           end if;
503
504         when others =>
505        current_state <= S_start;
506
507         end case;
508       end if;
509
510     end process gen_shifted_A_IN;
511
512   ---------------------------------------------------------------------
513     gen_sigma_delay : process( RST, multi_start, MCLK )
514     begin -- process gen_sigma_delay
515
516       if( RST = '1' or multi_start = '1') then
517         sigma_delay <= (others => '0');
518       elsif( MCLK'event and MCLK = '1' ) then
519         sigma_delay <= sigma;
520       end if;
521
522     end process gen_sigma_delay;
523
524   ---------------------------------------------------------------------
525     sigma <= shifted_A_IN + sigma_delay;
526
527   ---------------------------------------------------------------------
528     gen_MULTIPLIED_OUT : process( RST, MCLK ) -- pick up multiplication results
529     begin -- process gen_MULTIPLIED_OUT
530
531       if( RST = '1' ) then
532         MULTIPLIED_OUT <= (others => '0');
533       elsif( MCLK'event and MCLK = '1' ) then
534         if( MULEND = '1' ) then
535           if( multiplied_flag = '0' ) then
536             MULTIPLIED_OUT( 19 ) <= '0';
537             for i in 18 downto 0 loop
538               MULTIPLIED_OUT( i ) <= sigma(i +  1);
539             end loop;
540           else
541             MULTIPLIED_OUT( 19 ) <= '1';
542             for i in 18 downto 0 loop
543               MULTIPLIED_OUT( i ) <= not sigma(i +  1);
544             end loop;
545           end if;
546         end if;
547       end if;
548
549     end process gen_MULTIPLIED_OUT;
550
551   ---------------------------------------------------------------------
552   end; -- architecture multiplier_mxn
```

iir_tab.vhd

```
library IEEE;
  use IEEE.std_logic_1164.all;
  use IEEE.std_logic_arith.all;

entity iir_tab is
   Port (
          RST       : In std_logic; -- reset
          FSCLK     : In std_logic; -- Fs (a clock synchronized with input data)
          D_IN      : In std_logic_vector(11 downto 0); -- data input
          MCLK      : In std_logic; -- 384FSCLK or faster (master clock)
          A0        : In std_logic_vector( 9 downto 0); -- coeff a0
          A1        : In std_logic_vector( 9 downto 0); -- coeff a1
          A2        : In std_logic_vector( 9 downto 0); -- coeff a2
          B1        : In std_logic_vector( 9 downto 0); -- coeff b1
          B2        : In std_logic_vector( 9 downto 0); -- coeff b2
          D_OUT     : Out std_logic_vector(11 downto 0) ); -- filtered data output
end iir_tab;

architecture STRUCTURE of iir_tab is

  signal coeff_signal    : std_logic_vector( 9 downto 0);
  signal mul_signal      : std_logic_vector(19 downto 0);
  signal acc_signal      : std_logic_vector(11 downto 0);
  signal accenb_signal   : std_logic;
  signal acclat_signal   : std_logic;
  signal acccl_signal    : std_logic;
  signal mux_signal      : std_logic_vector(11 downto 0);
  signal mulstart_signal : std_logic;
  signal mulend_signal   : std_logic;
  signal co16_signal     : std_logic_vector(3 downto 0);

component divider
Port (
          RST       : In std_logic;
          FSCLK     : In std_logic;
          MCLK      : In std_logic;
          CO16      : Out std_logic_vector(3 downto 0);
          MULSTART  : Out std_logic;
          MULEND    : Out std_logic;
          ACCLAT    : Out std_logic;
          ACCCL     : Out std_logic;
          ACCENB    : Out std_logic );
end component;

component multiplier_mxn
Port (
          RST       : In std_logic;
          COEFF_IN  : In std_logic_vector(  9 downto 0 );
          A_IN      : In std_logic_vector( 11 downto 0 );
          MULSTART  : In std_logic;
          MULEND    : In std_logic;
          MCLK      : In std_logic;
          MULTIPLIED_OUT : Out std_logic_vector( 19 downto 0 ) );
end component;

component accumulator
Port (
          RST       : In std_logic;
          MCLK      : In std_logic;
          A_IN      : In std_logic_vector(19 downto 0);
          ACCLAT    : In std_logic;
          ACCCL     : In std_logic;
          ACCENB    : In std_logic;
          ACC_OUT   : Out std_logic_vector(11 downto 0) );
end component;

component dly_mux_coeff
Port (
          CO16      : In std_logic_vector(3 downto 0);
          RST       : In std_logic;
          FSCLK     : In std_logic;
          D_IN      : In std_logic_vector(11 downto 0);
          ACC_IN    : In std_logic_vector(11 downto 0);
          A0        : In std_logic_vector( 9 downto 0);
          A1        : In std_logic_vector( 9 downto 0);
          A2        : In std_logic_vector( 9 downto 0);
          B1        : In std_logic_vector( 9 downto 0);
          B2        : In std_logic_vector( 9 downto 0);
          COEFF_OUT : Out std_logic_vector( 9 downto 0);
          MUX_OUT   : Out std_logic_vector(11 downto 0);
```

```
81              D_OUT      : Out std_logic_vector(11 downto 0) );
82    end component;
83
84    begin -- architecture iir_tab
85
86    divider_1: divider port map (
87              RST => RST,
88              FSCLK => FSCLK,
89              MCLK => MCLK,
90              CO16 => co16_signal,
91              MULSTART => mulstart_signal,
92              MULEND => mulend_signal,
93              ACCLAT => acclat_signal,
94              ACCCL => acccl_signal,
95              ACCENB => accenb_signal );
96
97    multiplier_mxn_1: multiplier_mxn port map (
98              RST => RST,
99              COEFF_IN => coeff_signal,
100             A_IN => mux_signal,
101             MULSTART => mulstart_signal,
102             MULEND => mulend_signal,
103             MCLK => MCLK,
104             MULTIPLIED_OUT => mul_signal);
105
106   accumulator_1: accumulator port map (
107             RST => RST,
108             MCLK => MCLK,
109             A_IN => mul_signal,
110             ACCLAT => acclat signal,
111             ACCCL => acccl_signal,
112             ACCENB => accenb_signal,
113             ACC_OUT => acc_signal);
114
115   dly_mux_coeff_1: dly_mux_coeff port map (
116             CO16 => co16_signal,
117             RST => RST,
118             FSCLK => FSCLK,
119             D_IN => D_IN,
120             ACC_IN => acc_signal,
121             A0 => A0,
122             A1 => A1,
123             A2 => A2,
124             B1 => B1,
125             B2 => B2,
126             COEFF_OUT => coeff_signal,
127             MUX_OUT => mux_signal,
128             D_OUT => D_OUT);
129
130   end; -- architecture iir_tab
```

iir_2tab.vhd

```
1     library IEEE;
2       use IEEE.std_logic_1164.all;
3       use IEEE.std_logic_arith.all;
4
5     entity iir_2tab is
6        Port (
7               RST       : In std_logic; -- reset
8               FSCLK     : In std_logic; -- Fs (a clock synchronized with input data)
9               D_IN      : In std_logic_vector( 7 downto 0); -- input data
10              MCLK      : In std_logic; -- 384FSCLK or faster (master clock)
11              D_OUT     : Out std_logic_vector( 7 downto 0) ); -- filtered data output
12    end iir_2tab;
13
14    architecture STRUCTURE of iir_2tab is
15
16    component iir_tab
17    Port (
18              RST       : In std_logic;
19              FSCLK     : In std_logic;
20              D_IN      : In std_logic_vector(11 downto 0);
21              MCLK      : In std_logic;
22              A0        : In std_logic_vector( 9 downto 0);
23              A1        : In std_logic_vector( 9 downto 0);
24              A2        : In std_logic_vector( 9 downto 0);
25              B1        : In std_logic_vector( 9 downto 0);
26              B2        : In std_logic_vector( 9 downto 0);
27              D_OUT     : Out std_logic_vector(11 downto 0) );
28    end component;
29
30      signal a0_sig     : std_logic_vector( 9 downto 0);
31      signal a1_sig     : std_logic_vector( 9 downto 0);
32      signal a2_sig     : std_logic_vector( 9 downto 0);
33      signal b1_sig     : std_logic_vector( 9 downto 0);
34      signal b2_sig     : std_logic_vector( 9 downto 0);
35      signal c0_sig     : std_logic_vector( 9 downto 0);
36      signal c1_sig     : std_logic_vector( 9 downto 0);
37      signal c2_sig     : std_logic_vector( 9 downto 0);
38      signal d1_sig     : std_logic_vector( 9 downto 0);
39      signal d2_sig     : std_logic_vector( 9 downto 0);
40      signal d_in_sig   : std_logic_vector(11 downto 0);
41      signal d_out_sig  : std_logic_vector(11 downto 0);
42      signal d_inter_sig: std_logic_vector(11 downto 0);
43
44    begin -- architecture iir_t2tab
45
46    -- Note: we need two more bits for coefficients
47    -- and four more bits for in/out data internally,
48    -- so we compensate those in this architecture
49
50      a0_sig <= "0001111011";
51      a1_sig <= "1100111010";
52      a2_sig <= "0001111011";
53      b1_sig <= "0011000100"; -- polarity has been inverted
54      b2_sig <= "1110000110"; -- polarity has been inverted
55
56      c0_sig <= "0001111110";
57      c1_sig <= "1100110110";
58      c2_sig <= "0001111110";
59      d1_sig <= "0011001100"; -- polarity has been inverted
60      d2_sig <= "1110000110"; -- polarity has been inverted
61
62    ------------------------------------------------------------
63    gen_d_in_sig : process(D_IN) -- compensate two bits for D_IN
64    begin -- process gen_d_in_sig
65
66      if( D_IN( 7) = '0' ) then
67        d_in_sig <= "0000" & D_IN;
68      else
69         d_in_sig <= "1111" & D_IN;
70       end if;
71
72    end process gen_d_in_sig;
73
74    ------------------------------------------------------------
75    gen_d_out_sig : process(d_out_sig) -- compensate two bits for D_OUT
76    begin -- process gen_d_out_sig
77
78      if( d_out_sig(11) = '0' and
79          (d_out_sig(10) = '1' or
80           d_out_sig( 9) = '1' or
```

```
81          d_out_sig( 8) = '1' or
82          d_out_sig( 7) = '1') ) then -- overflow
83        D_OUT <= (others => '1');
84        D_OUT( 7) <= '0';
85      elsif( d_out_sig(11) = '1' and
86            (d_out_sig(10) = '0' or
87             d_out_sig( 9) = '0' or
88             d_out_sig( 8) = '0' or
89             d_out_sig( 7) = '0') ) then -- underflow
90        D_OUT <= (others => '0');
91        D_OUT( 7) <= '1';
92      else
93        for i in  7 downto 0 loop
94          D_OUT(i) <= d_out_sig(i);
95        end loop;
96      end if;
97
98    end process gen_d_out_sig;
99
100   iir_tab_a: iir_tab port map ( -- 1st iir filter
101             RST => RST,
102             FSCLK => FSCLK,
103             D_IN => d_in_sig,
104             MCLK => MCLK,
105             A0 => a0_sig,
106             A1 => a1_sig,
107             A2 => a2_sig,
108             B1 => b1_sig,
109             B2 => b2_sig,
110             D_OUT => d_inter_sig);
111
112   iir_tab_b: iir_tab port map ( -- 2nd iir filter
113             RST => RST,
114             FSCLK => FSCLK,
115             D_IN => d_inter_sig,
116             MCLK => MCLK,
117             A0 => c0 sig,
118             A1 => c1_sig,
119             A2 => c2_sig,
120             B1 => d1_sig,
121             B2 => d2_sig,
122             D_OUT => d_out_sig);
123
124   end; -- architecture iir_2tab
```

div192_clk.vhd

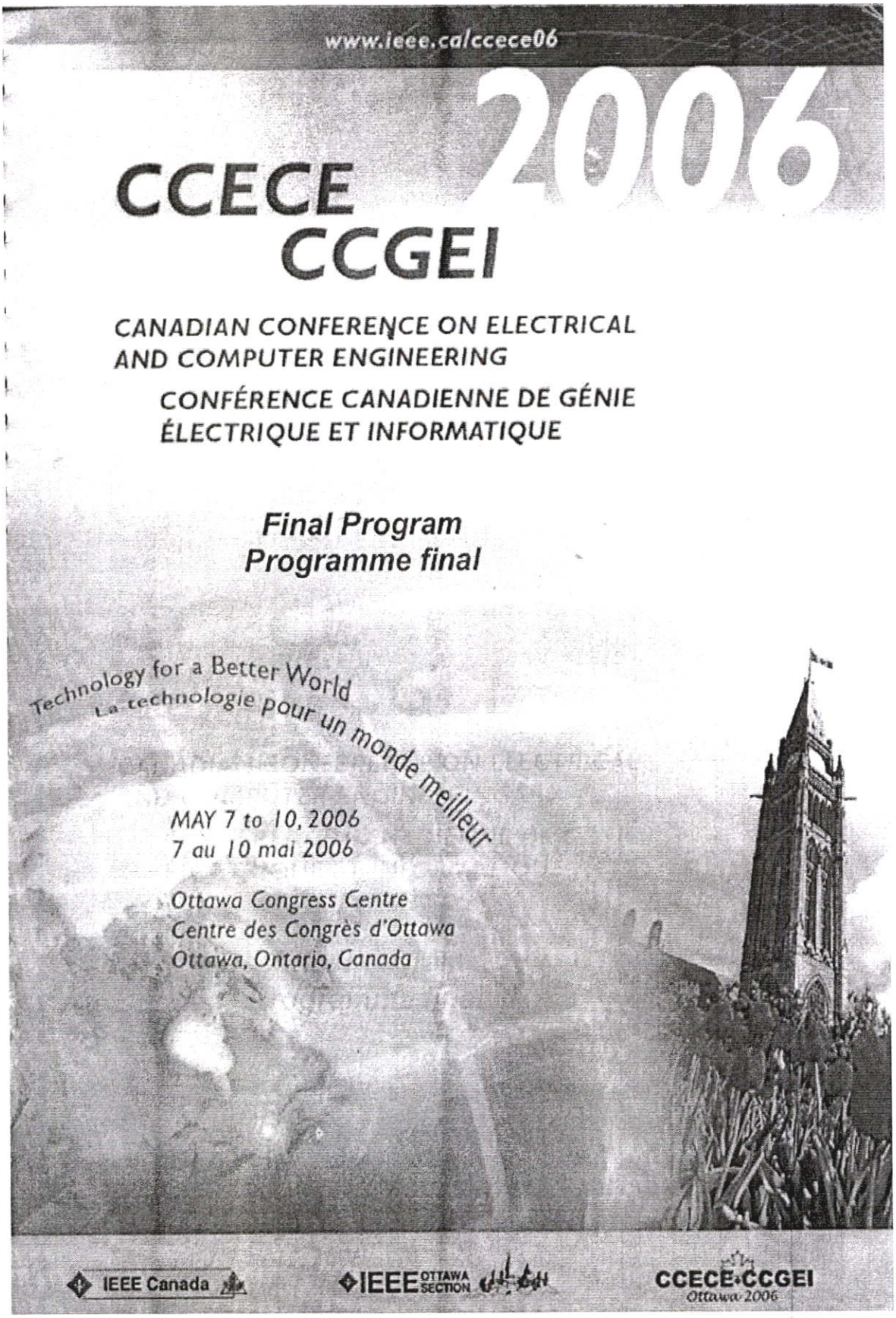
```
1     Library ieee;
2     Use ieee.std_logic_1164.all;
3     Use ieee.std_logic_arith.all;
4
5     Entity div192_clk is
6          Port (
7               clk  :    in   bit;
8               fs   :    out  bit
9               );
10    End  div192_clk;
11
12    Architecture div_192 of div192_clk is
13    Begin
14
15    Process (clk)
16    Variable cnt   :    integer range 0 to 192 := 0;
17    Begin
18    if clk'event and clk = '1' then
19         if cnt > 192 then
20              cnt := 0;
21         elsif cnt > 86 then
22              fs   <= '1';
23              cnt := cnt + 1;
24         else
25              fs   <= '0';
26              cnt := cnt + 1;
27         end if;
28
29    end if;
30    End process;
31    End ;
32
```

div2_clk.vhd

```
1     Library ieee;
2     Use ieee.std_logic_1164.all;
3     Use ieee.std_logic_arith.all;
4
5     Entity div2_clk is
6     Port       (
7                clk       :    in   bit;
8                clk_2     :    out  bit
9                );
10    End div2_clk;
11
12    Architecture clk_2 of div2_clk is
13    Begin
14
15    Process (clk)
16    Variable  cnt  :    integer range 0 to 1 := 0;
17    Begin
18         if clk'event and clk = '1' then
19
20               if cnt > 0 then
21                    clk_2     <= '1';
22                    cnt := cnt + 1;
23               else
24                    clk_2     <= '0';
25                    cnt := cnt + 1;
26               end if;
27         end if;
28    End process;
29    End ;
```

ภาคผนวก ง.

ผลงานวิจัยที่ได้รับการตีพิมพ์เผยแพร่


www.ieee.ca/ccece06
2006
CCECE
CCGEI
CANADIAN CONFERENCE ON ELECTRICAL AND COMPUTER ENGINEERING
CONFÉRENCE CANADIENNE DE GÉNIE ÉLECTRIQUE ET INFORMATIQUE
Final Program
Programme final
Technology for a Better World
La technologie pour un monde meilleur
MAY 7 to 10, 2006
7 au 10 mai 2006
Ottawa Congress Centre
Centre des Congrès d'Ottawa
Ottawa, Ontario, Canada
IEEE Canada
IEEE OTTAWA SECTION
CCECE•CCGEI
Ottawa 2006

TRACKING NOTCH FILTER FOR ELECTROCARDIOGRAPH MEASUREMENT

S. Witthayapradit
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, 10520 Thailand
email: zsuriya@gmail.com

S. Chitwong
Faculty of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, 10520 Thailand
email: kcsakrey@kmitl.ac.th

S. Tumthong
Faculty of Science and Technology
Rajamangala University of Technology Suvarnabhumi, Nonthaburi Campus
Nonthaburi, 11000 Thailand
email: suwut@rmutsb.ac.th

Abstract

This paper presents a tracking notch filter with opened-loop architecture for elimination of electric field frequency which can also interfere into ECG waveform. For measuring ECG waveform having the isolated instrumentation amplifier, at here, the quality factor (Q) of this filter is assigned about 8 and the attenuation of frequency noise is around of −35 dB and the gain of amplifier is of 1000. The frequency from frequency noise detector is multiplied by the frequency synthesizer. The clock signal from the frequency synthesizer is used for the switched capacitor network. The performance of the proposed filter is tested with ECG waveform included frequency noise of 50Hz , 55Hz and 60Hz and also with measuring in Lead I of bipolar lead included noise signal of 50Hz for both cases, that is, before and after through the notch filter circuit in the time and frequency domain. The measured results have been successfully tested. In this paper, the discrete and passive devices are used for the designed circuit.

Keywords: ECG; Tracking notch filter; Line filter.

1. Introduction

First, the tracking notch filter utilizes sequential phase-frequency detector−S-PFD, band pass filter, summing amplifier, multiplication, integrator, and voltage controlled oscillator−VCO as important components [1]. Second, switched-capacitor adaptive sample-rate filter−SC-ASRF which is consisted of band pass filter, high pass filter, summing amplifier, multiplication, integrator, and VCO is used [2]. Both mentioned methods are used as closed loop and also large circuit. For this paper, tracking notch filter is used as open loop and noise frequency is introduced for frequency synthesizer. The designed tracking notch filter is then small circuit which is consisted of switched-capacitor network−SCN and frequency synthesizer to filter out noise in ECG signal shown in Fig.1.

This paper is organized as following. In section 2, the designed tracking notch filter is represented and described. The experiments and results together with conclusions are given in section 3 and 4, respectively.

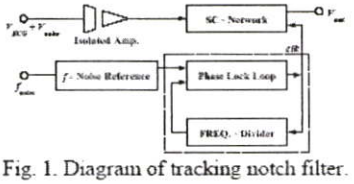


Fig. 1. Diagram of tracking notch filter.

2. Design of Tracking Notch Filter

2.1. The 4-Order Band Stop Filter

In this paper, integrated circuit as MAX263 which operate as switched-capacitor network is selected for band stop filter shown in Fig. 2. The characteristics of band stop filter consist of center frequency f_o, quality factor Q which are set by logic, and attenuation. The structure of integrated circuit consists of the two 2-order band stop filter. Each is connected in series, the order of series connected band stop filter is then of 4 and also the maximum attenuation is of -40dB.

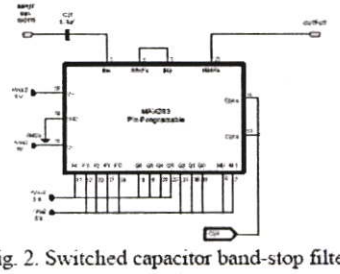


Fig. 2. Switched capacitor band-stop filter.

2.2.1 Center frequency. It is adjusted to track noise frequency. Equation [1] is used to assign center frequency. At here, ration is of 150 times, that is, the input logic F4-F0 of integrated circuit band stop filter is 10000. Adjustment of the center frequency from 40Hz to 70Hz corresponds with the clock frequency from 6kHz to 10.5kHz.

$$N = \frac{f_{clk}}{f_o} \qquad [1]$$

where N is ratio, f_{clk} clock frequency, and f_o center frequency.

1-4244-0038-4 2006 IEEE
CCECE/CCGEI, Ottawa, May 2006

2.2.2 Quality factor. The band width of band stop filter is dependent on quality factor and center frequency as equation [2]. The quality factor is of 8 by which input logic Q6-Q0 of integrated circuit band stop filter is 1111000.

$$Q = \frac{f_o}{BW} \qquad [2]$$

where Q is quality factor and BW bandwidth of band stop filter.

2.2. Frequency Synthesizer

Function of frequency synthesizer is multiplying input frequency, 40Hz to 70Hz, from noise frequency detector into 6kHz to 10.5kHz as clock frequency for controlling switched-capacitor. As mentioned, the multiplication ratio is then of 150 times. Frequency synthesizer circuit consists of phase lock loop by which the minimum and maximum frequency of voltage control oscillator is as equation [3] and frequency divider shown in Fig. 3.

$$f_{min} = \frac{1}{R_2(C_1 + 32\text{pF})} \text{ and } f_{max} = \frac{1}{R_1(C_1 + 32\text{pF})} \qquad [3]$$

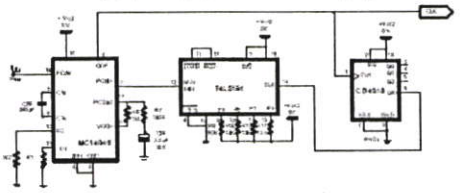


Fig. 3. Frequency synthesizer.

2.3. Noise Frequency Detector

Inducing of electric field via signal cable is used to detect noise frequency as input signal for voltage comparator. Hysteresis comparator shown in Fig. 4 is implemented as voltage comparator by which ranging of transition voltage is assigned as equation [4]

$$V_{upper} = \frac{R_2}{R_1 + R_2}[+V_{out(max)}] \text{ and } V_{lower} = \frac{R_2}{R_1 + R_2}[-V_{out(max)}] \qquad [4]$$

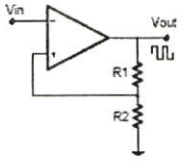


Fig. 4. Hysteresis comparator.

2.4. Isolated Instrument Amplifier

To amplify ECG waveform having amplitude of 1mV approximately, isolated instrument amplifier is applied and gain is of 1000 times. At here, we selected the integrated circuit as ISO175 of Burr-Brown shown in Fig. 5. AL input is connected to electrode on left hand, AR input is connected to electrode on right hand, and GND1 is connected to electrode on left leg.

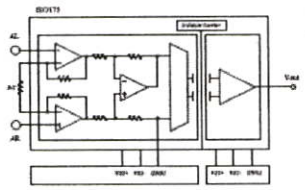


Fig. 5. Isolated instrument amplifier.

3. Experiments and Results

3.1. Band Stop Filter

Each of the clock frequency of 7.5kHz, 8.25kHz and 9.0kHz, respectively, is fed to band stop filter. Let input signal be sinusoidal waveform, whose amplitude and frequency are of 1Vp-p and of 10Hz to 100Hz. Output signals are measured and recorded and the frequency response is then plotted as Fig. 6. From Fig. 6, that is, the clock frequency of 7.5kHz, 8.25kHz and 9.0kHz correspond with center frequency of 50Hz, 55Hz and 60Hz, respectively.

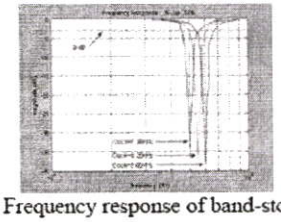


Fig. 6. Frequency response of band-stop filter.

3.2. Frequency Synthesizer

As mentioned, let the multiplication ratio be 150 times. The input signal frequency from noise frequency detector as the clock frequency ranging from 40Hz to 70Hz fed into frequency synthesizer. The output signal frequency is measured and relation between the input and output signal frequency is plotted as shown in Fig. 7 by which data1 means the data from experiment and data2 means the data from calculation. We see that the relation between the input and output signal frequency is linear enough.

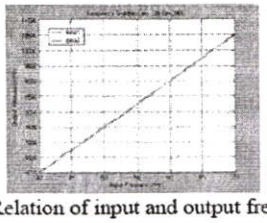


Fig. 7. Relation of input and output frequency.

3.3. Filtering with Noise Frequency added Standard ECG Waveform

To filter the noise frequency at each frequency, noise frequency detector, frequency synthesizer, isolated instrument amplifier and 4-order band stop filter are connected as the tracking notch filter. By using MATLAB program, the standard ECG waveform [3], whose amplitude is of 1.2V shown in Fig. 8, is created and added by noise signal. The noise added standard ECG waveform is conversed for BenchLink program in order to download onto arbitrary generator of HP33120A which generate real ECG signal shown in Fig. 9.

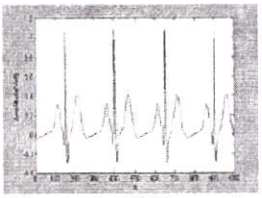


Fig. 8. The standard ECG waveform [3].

From Fig. 9, signal No. 3 is the standard ECG signal whose amplitude is of 1V and period is of 1ms. Signal No.1 is the noise added standard ECG signal by which amplitude and frequency of noise is of 200mV and 50Hz, respectively. Signal No.2 is the filtered ECG signal from the tracking notch filter.

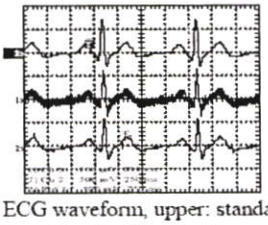
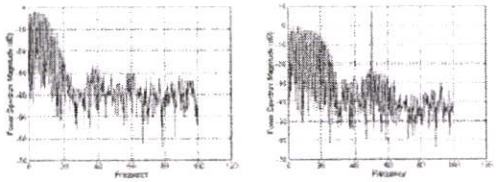
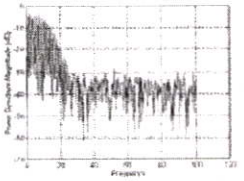


Fig. 9. Real ECG waveform, upper: standard ECG [3]
Middle: ECG combined with noise frequency of 50Hz.
Lower: Filtered ECG.

Also we use WAVESTAR program to record the real ECG waveform for analyzing spectral by using MATLAB program. Fig. 10 shows all spectral of real ECG signal corresponding with signal shown in Fig. 9. From Fig. 10(c), see that spectral of the filtered ECG signal at frequency of 50Hz is eliminated.



(a) (b)



(c)

Fig. 10. Spectral from analyzing by using MATLAB.

To confirm result in term of MSE, the standard ECG waveform from Fig. 8 and the filtered ECG waveform from Fig. 9 which is recorded by WAVESTAR program, both shown together in Fig. 11 by which dot line is the standard ECG waveform and solid line is the filtered ECG wavform, are used to calculate MSE. The MSE is of 3.44%.

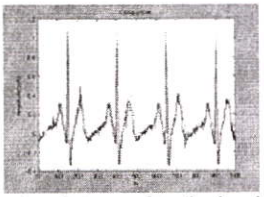
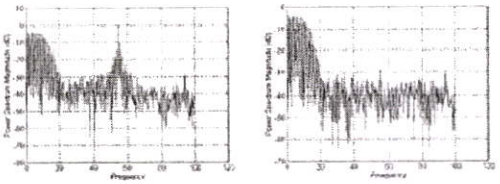


Fig. 11. Comparison between Standard and Filtered ECG.

For other experiment, let noise frequency be of 55Hz and 60Hz. The experiment is performed the same as in case of noise frequency of 50Hz. Results of filtering noise frequency of 55Hz show in Fig. 12 to 14. Fig. 13(b) shows that spectral of the filtered ECG signal at frequency of 55Hz is eliminated when comparing with Fig. 13(a). The MSE is of 2.34%.

Fig. 12. Upper: ECG combined with noise frequency of 55Hz.
Lower: Filtered ECG.



(a) (b)

Fig. 13. Spectral from analyzing by MATLAB.

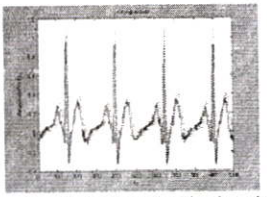


Fig. 14. Comparison between standard and filtered ECG.

Results of filtering noise frequency of 60Hz show in Fig. 15 to 17. Fig. 16(b) shows that spectral of the filtered ECG signal at frequency of 60Hz is eliminated when comparing with Fig. 16(a). The MSE is of 1.44%.

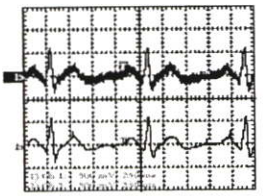
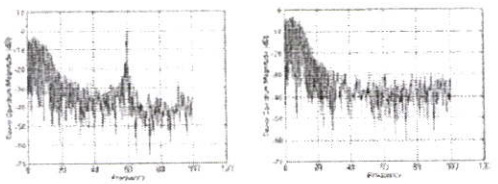


Fig. 15. Upper: ECG combined with noise frequency of 60Hz. Lower: Filtered ECG.



(a) (b)

Fig. 16. Spectral from analyzing by MATLAB.

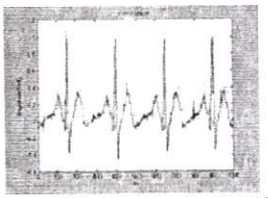
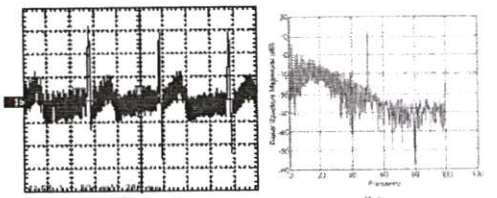


Fig. 17. Comparison between Standard and Filtered ECG.

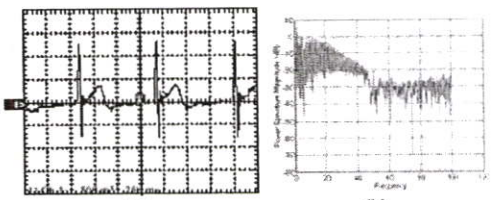
3.4. Filtering with Real ECG Signal

Using electrode as standard position to measure ECG signal in lead I, such ECG signal is connected to the isolated instrument amplifier and output signal from one is fed to the designed tracking notch filter. The unfiltered and filtered ECG waveform is measured and recorded by WAVESTAR program. Fig. 18(a) and (b) show the unfiltered ECG waveform and its spectral, respectively. Also, Fig. 19(a) and (b) show the filtered ECG waveform and its spectral, respectively. Comparing Fig. 18 and 19 shows that the designed tracking notch filter can filter out the noise frequency seeing in both time and frequency domain.



(a) (b)

Fig. 18. (a) Unfiltered ECG (b) its Spectral.



(a) (b)

Fig. 19. (a) Filtered ECG (b) its Spectral.

4. Conclusions

From frequency response of the designed tracking notch filter, attenuation at each center frequency is approximately of -35dB to -38dB which is good enough. The difference of the calculated and measured frequency from frequency synthesizer for which the multiplication is of 150 times is very close. The performance by experiment with the noise frequency added standard ECG waveform have shown the filtered ECG waveform in time domain without noise, frequency domain without spectral of noise frequency, and in term of MSE seeing that it is low enough. Also, the performance of the real ECG signal has shown the filtered ECG waveform in time domain without noise and in frequency domain without spectral of noise frequency, only.

One of advantage of the designed tracking notch filter is the small circuit size when comparing with the closed loop band stop filter and can use discrete component and passive device. In future, this filter will develop as a single chip.

References

[1] L. Bustamante and M. A. Soderstrand, "High-Rang Switched Capacitor Tracking," *IEEE International Symposium on Circuits and Systems*, vol. 1, pp. 688-691, May, 1999.

[2] L. Bustamante, R. H. Strandberg, and M. A. Soderstrand, "Switched-Capacitor Adaptive Sampling Rate Notch Filter for Enhancement and Elimination of Bandpass Signal," *Proceeding Of the 40th Midwest Symposium on Circuits and System*, vol. 2, pp. 1354-1357, August, 1997.

[3] P. E. McSharry, G. D. Clifford, L. Tarassenko, and L. A. Smith, "A Dynamical Model for Generating Synthesis Electrocardiogram Signals," *IEEE Trans. On Biomedical Engineering*, vol. 50, no. 3, pp. 289-294, March, 2003.

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายสุริยา วิทยาประดิษฐ์
วัน เดือน ปีเกิด สถานที่เกิด	23 พฤศจิกายน พ.ศ.2510 ที่ อ.เมือง จ.นครศรีธรรมราช
ที่อยู่ปัจจุบัน	111/232 ซอย คาริเบียน 3 หมู่ที่ 8 ถ. ฉลองกรุง
	แขวง ลำปลาทิว เขต ลาดกระบัง กรุงเทพมหานคร 10520
ประวัติการศึกษา	2536 อุตสาหกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์
	วิทยาลัยมหานคร
ความชำนาญเฉพาะด้าน	1. การวัดสัญญาณคลื่นไฟฟ้าหัวใจ
	2. การประยุกต์ใช้งานวงจรอิเล็กทรอนิกส์
	3. ออกแบบชุดทดลองเกี่ยวกับเอฟพีจีเอ
	4. ระบบติดตามการเคลื่อนที่โดยใช้ระบบจีพีเอส
ประสบการณ์ทำงาน	บริษัทเวฟเกต จำกัด พ.ศ. 2530 - 2532
	บริษัทถาวรคอมพิวเตอร์ พ.ศ. 2536 - 2538
	มหาวิทยาลัยเทคโนโลยีมหานคร พ.ศ. 2533 - ปัจจุบัน

ผลงานวิจัยการประชุมวิชาการ

1. สุริยา วิทยาประดิษฐ์ , ศุภชัย นำเกียรติสกุล , สุเจตน์ จันทรังษ์ , "การรู้จำลายมือเขียนภาษาไทย" , การประชุมวิชาการทางไฟฟ้า ครั้งที่-16 , สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง , พ.ศ. 2536.
2. Suriya Witthayapradit , Pullop Tangbovonpechet , Pichet Chopaka , Panom Petchjatuporn ,"A Prototype In - Vechicle Navigation System Based On GPS" , p.430-431, EECON-18 , Mahanakorn University of Technology 1995.
3. สุริยา วิทยาประดิษฐ์ , พนม เพชรจตุพร ,"แนวทางประยุกต์ใช้ระบบจีพีเอสกับระบบขนส่งมวลชน" , การประชุมใหญ่ทางวิชาการประจำปี 2540 , วิศวกรรมสถานแห่งประเทศไทย ในพระบรมราชูปถัมภ์ 20-23 พฤศจิกายน 2540.
4. S. Witthayapradit , S. Chitwong , S. Tumthong ,"TRACKING NOTCH FILTER FOR ELECTROCARDIOGRAPH MEASUREMENT" , CCECE/CCGEI, IEEE Ottawa CANADA , May 2006.

ผลงานวิจัยวารสาร

1. สุริยา วิทยาประดิษฐ์ , ณัฐวุฒิ ควงลัดดา , "Digital Training ที่สร้างจากชิพตระกูล CPLD"(ตอนที่ 1) , เซมิคอนดักเตอร์ อิเล็กทรอนิกส์ , ฉบับที่ 260 , มีนาคม 2547.
2. สุริยา วิทยาประดิษฐ์ , ณัฐวุฒิ ควงลัดดา , "Digital Training ที่สร้างจากชิพตระกูล CPLD"(ตอนที่ 2 ตอนจบ) , เซมิคอนดักเตอร์ อิเล็กทรอนิกส์ , ฉบับที่ 261 , เมษายน 2547.
3. สุริยา วิทยาประดิษฐ์ , พีระพล ยุวภูษิตานนท์ ,"เรียนรู้ระบบดิจิตอล สำหรับการออกแบบวงจรกรองความถี่แบบ FIR จากโปรแกรม MATLAB" , เซมิคอนดักเตอร์ อิเล็กทรอนิกส์ , ฉบับที่ 263 , มิถุนายน 2547.