การออกแบบและการสร้างวงจรกรองสัญญาณเชิงเลขแบบปรับตัวได้ ชนิดผลตอบสนองอิมพัลส์จำกัด โดยใช้โครงสร้างเลขคณิตกระจายด้วยอุปกรณ์ FPGA

Design and Implementation of Distributed Arithmetic Adaptive FIR Filter Using FPGA

โดย

นายเสฐียรพล ปิยะมหาโชติ

นายเอนกชัย นำเจริญวัฒนกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

การออกแบบและการสร้างวงจรกรองสัญญาณเชิงเลขแบบปรับตัวได้ ชนิดผลตอบสนองอิมพัลส์จำกัด โดยใช้โครงสร้างเลขคณิตกระจายด้วยอุปกรณ์ FPGA

Design and Implementation of Distributed Arithmetic Adaptive FIR Filter Using FPGA

โดย

นายเสฐียรพล ปิยะมหาโชติ 44015043

นายเอนกชัย นำเจริญวัฒนกุล 44015047

อาจารย์ที่ปรึกษา

รศ.ดร.กอบชัย เดชหาญ

อาจารย์ศรวัฒน์ ชิวปรีชา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **การออกแบบและการสร้างวงจรกรองสัญญาณเชิงเลขแบบปรับตัวได้ ชนิดผลตอบสนองอิมพัลส์จำกัด โดยใช้โครงสร้างเลขคณิตกระจายด้วยอุปกรณ์ FPGA**

Design and Implementation of Distributed Arithmetic Adaptive FIR Filter using FPGA

ผู้จัดทำ

1. นายเสฐียรพล ปิยะมหาโชติ 44015043

2. นายเอนกชัย นำเจริญวัฒนกุล 44015047

(_____) อาจารย์ที่ปรึกษา

รศ.ดร. กอบชัย เดชหาญ

(_____) อาจารย์ที่ปรึกษา

อาจารย์ศรวัฒน์ ชิวปรีชา

การออกแบบและการสร้างวงจรกรองสัญญาณเชิงเลขแบบปรับตัวได้ ชนิดผลตอบสนองอิมพัลส์จำกัด โดยใช้โครงสร้างเลขคณิตกระจายด้วยอุปกรณ์ FPGA

Design and Implementation of Distributed Arithmetic Adaptive FIR Filter Using FPGA

โดย นายเสฐียรพล ปิยะมหาโชติ 44015043

นายเอนกชัย นำเจริญวัฒนกุล 44015047

อาจารย์ที่ปรึกษา รศ.ดร. กอบชัย เดชหาญ

อาจารย์ศรวัฒน์ ชิวปรีชา

บทคัดย่อ

โครงงานนี้ นำเสนอการออกแบบและสร้างวงจรกรองสัญญาณเชิงเลขแบบปรับตัวได้ชนิดผลตอบสนองอิมพัลส์จำกัด (Adaptive FIR Filter) โดยใช้อัลกอริทึมค่าเฉลี่ยกำลังสองน้อยสุด (Least Mean Square, LMS Algorithm) ผลจำลองการทำงานจะถูกแสดงโดยใช้โปรแกรม MATLAB ในส่วนของการสร้างเป็นฮาร์ดแวร์ จะใช้โครงสร้างเลขคณิตกระจาย (Distributed Arithmetic, DA) ซึ่งเป็นเทคนิคที่ใช้ในการแปลงฟังก์ชันถ่ายโอน (Transfer Function) ซึ่งเป็นสมการที่อยู่ในรูปผลบวกของผลคูณ (Sum of Product) ให้กระจายออกเป็นระดับบิท (Bit Level) และเก็บค่าผลคูณย่อยไว้ในตารางเปิดดู (Look-up table) ทำให้วงจรที่ได้ปราศจากตัวคูณ จากนั้นจะทำการบรรยายพฤติกรรมการทำงานของวงจรด้วยภาษา VHDL (Very High Speed Integrated Circuit Hardware Description Language) วงจรที่ได้ออกแบบจะถูกสังเคราะห์และโปรแกรมลงในอุปกรณ์ FPGA (Field Programmable Gate Array) สำหรับทดสอบการทำงาน

ABSTRACT

This project proposes a design and implementation of adaptive FIR digital filter by using least mean square algorithm. The simulation results will be shown based on MATLAB program simulator, the hardware implementation is based on distributed arithmetic which is the technique using in transfer function transformation. The equations are in the sum of product form for expanding into bit level. The product results are stored in look-up table, this method uses multiplierless. All operations are described by using VHDL, the circuits are designed and synthesized on FPGA in order to test the operation.

สารบัญ

สารบัญรูป

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

หน้า

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาของหัวข้อปริญญานิพนธ์

ในระบบการสื่อสารที่จะทำให้ด้านส่งและด้านรับสามารถสื่อสารกันได้อย่างถูกต้องนั้น สัญญาณที่รับได้ทางด้านรับจะต้องเป็นสัญญาณเดียวกันกับสัญญาณทางด้านส่งที่ส่งออกมา แต่โดยทั่วไปสัญญาณที่ทำการส่งไปถึงปลายทางนั้นพบว่าจะถูกรบกวนทำให้คุณภาพของสัญญาณลดลง การที่จะนำสัญญาณนั้นไปใช้งานจะต้องทำการกรองสัญญาณรบกวนทิ้งก่อน แต่สัญญาณรบกวนมักมีขนาดและความถี่ไม่แน่นอนจึงมีความจำเป็นต้องสร้างตัวกรองความถี่ที่สามารถปรับตัวได้ เพื่อกำจัดสัญญาณที่ไม่ต้องการออกไป และในการออกแบบและสร้างตัวกรองสัญญาณในสมัยก่อนนั้น จะต้องทราบถึงคุณสมบัติของระบบ ว่าต้องการตัวกรองสัญญาณที่มีคุณสมบัติอย่างไร ตอบสนองความถี่ในย่านใด จากนั้นจึงทำการคำนวณ และสร้างตัวกรองสัญญาณ โดยการนำเอาอุปกรณ์ต่าง ๆ มาประกอบกันเข้าเป็นวงจร แล้วทำการทดสอบการทำงานของวงจรเพื่อตรวจสอบการทำงานว่ามีความถูกต้องตามที่คำนวณไว้หรือไม่ ซึ่งปัญหาในการออกแบบลักษณะนี้ ถ้าการทำงานของวงจรไม่ถูกต้องแล้ว ก็จะต้องทำการคำนวณและสร้างใหม่ ซึ่งทำให้เกิดความยุ่งยาก ใช้เวลามาก และถ้าวงจรที่ต้องการออกแบบมีความซับซ้อนมากขึ้นก็จะเป็นเรื่องที่ยากมากในการออกแบบลักษณะนี้ ปัจจุบันได้มีโปรแกรมภาษาระดับสูง ที่ใช้ภาษาบรรยายการทำงานของวงจร (Hardware Description Language : HDL) ซึ่งเป็นภาษาที่ใช้สำหรับออกแบบฮาร์ดแวร์ ทำให้การออกแบบสามารถทำได้สะดวก ลดความยุ่งยากในการนำเอาอุปกรณ์มาเชื่อมต่อให้เป็นวงจร เพียงแต่เขียนโค้ด (Source Code) บรรยายการทำงานของวงจร หลังจากนั้นก็ทำการคอมไพล์ (Compile) แล้วจำลองการทำงาน (Simulation) ดูว่าได้ฟังก์ชั่นการทำงานและไทม์มิ่ง (Timing) ตามที่ต้องการหรือไม่ จากนั้นก็นำซอสโค้ดที่ได้ไปทำการสังเคราะห์ด้วยโปรแกรมสังเคราะห์ (Synthesis Tool) สุดท้ายนำวงจรที่ได้จากการสังเคราะห์ไปทำการแม็ป (Map) ลงไปยัง FPGA (Field Programmable Gate Array) เพื่อเป็นชิปต้นแบบสำหรับการนำไปทดสอบการทำงาน

โครงงานนี้จึงนำเสนอการออกแบบวงจรกรองสัญญาณเชิงเลขแบบปรับตัวได้แบบเอฟไออาร์ (Adaptive FIR Filter) โดยใช้อัลกอริทึมแบบแอลเอ็มเอส (LMS Algorithm) อัลกอริทึมแบบแอลเอ็มเอสเป็นอัลกอริทึมที่ถูกใช้งานอย่างแพร่หลายทางด้านการประมวลผลสัญญาณดิจิตอล เนื่องจากเป็นอัลกอริทึมที่ง่ายต่อการออกแบบ แต่ข้อเสียของอัลกอริทึมแบบแอลเอ็มเอสคือ ต้องเลือกค่าสัมประสิทธิ์การลู่เข้า (Factor of Convergence) ที่ต้องเหมาะสมกับการใช้งาน เพื่อหาจำนวน Order ของ Filter และสัมประสิทธิ์การลู่เข้าโดยทำการซิมูเลชั่น (Simulation) ด้วยโปรแกรมแมทแลบ (MATLAB) ก่อนนำไปเขียนโปรแกรมภาษา VHDL ในส่วนของการสร้างเป็นฮาร์ดแวร์ จะใช้โครงสร้างเลขคณิตกระจาย (Distributed Arithmetic, DA) ซึ่งเป็นเทคนิคที่ใช้ในการแปลงฟังก์ชันถ่ายโอน (Transfer Function) ซึ่งเป็นสมการที่อยู่ในรูปผลบวกของผลคูณ (Sum of Product) ให้กระจายออกเป็นระดับบิท (Bit Level) และเก็บค่าผลคูณย่อยไว้ในตารางเปิดดู (Look-up table) ทำให้วงจรที่ได้ปราศจากตัวคูณ จากนั้นจะทำการบรรยายพฤติกรรมการทำงานของวงจรด้วยภาษา VHDL (Very High Speed Integrated Circuit Hardware

Description Language) วงจรที่ได้ออกแบบจะถูกสังเคราะห์และโปรแกรมลงในอุปกรณ์ FPGA สำหรับทดสอบการทำงาน

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1.2.1 เพื่อศึกษาหลักการกรองสัญญาณโดยใช้ตัวกรองแบบปรับตัวได้

1.2.2 เพื่อศึกษาหลักการปรับตัวของอัลกอริทึมที่ใช้

1.2.3 เพื่อศึกษาและประยุกต์ใช้งานระบบประมวลผลสัญญาณดิจิตอล

1.2.4 เพื่อศึกษาและประยุกต์ใช้งานภาษา VHDL ในระบบประมวลผลสัญญาณดิจิตอล

1.3 ขอบเขตของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้เป็นการออกแบบและสร้างวงจรกรองสัญญาณเชิงเลขแบบปรับตัวได้ชนิดผลตอบสนองอิมพัลส์จำกัด โดยใช้โครงสร้างเลขคณิตกระจาย ในขั้นตอนการออกแบบจะทำการจำลองการทำงานด้วยโปรแกรม MATLAB และในการสร้างจะใช้ภาษา VHDL ในการเขียนบรรยายพฤติกรรมการทำงานของวงจร และทำการบันทึกผลตอบสนองของวงจรกรองที่ได้เทียบกับผลที่ได้จากการจำลองการทำงานด้วยโปรแกรม MATLAB

1.4 เนื้อหาของปริญญานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎีที่นำมาใช้ในการออกแบบและสร้างตัวกรองปรับตัวได้ ซึ่งประกอบด้วยพื้นฐานของวงจรกรองความถี่แบบดิจิตอล วงจรกรองความถี่ดิจิตอลแบบ FIR และ IIR พื้นฐานตัวกรองปรับตัวได้ ตัวกรองปรับตัวได้แบบเอฟไอาร์ และการนำโครงสร้างเลขคณิตกระจายมาประยุกต์ใช้กับวงจรกรองสัญญาณ

บทที่ 3 กล่าวถึงภาษา VHDL ใช้บรรยายพฤติกรรมการทำงานของวงจรดิจิตอล การออกแบบแอลเอ็มเอสอัลกอริทึมโดยใช้โครงสร้างเลขคณิตกระจาย รวมถึงโครงสร้างและการทำงานของวงจรกรองความถี่แบบปรับตัวได้ชนิดผลตอบสนองอิมพัลส์จำกัดโดยใช้โครงสร้างเลขคณิตกระจาย

บทที่ 4 กล่าวถึงการทดลองและผลการทดลองของวงจรกรองสัญญาณแบบปรับตัวได้ ที่ได้จากการจำลองการทำงานด้วยโปรแกรม MATLAB และการออกแบบวงจรต่าง ๆ ด้วยภาษา VHDL พร้อมผลการจำลองการทำงาน

บทที่ 5 เป็นการวิจารณ์ วิเคราะห์และสรุปผล

บทที่ 2
ทฤษฎีและหลักการ

2.1 ทฤษฎีพื้นฐานของวงจรกรองความถี่แบบดิจิตอล

2.1.1 ความหมายของวงจรกรองความถี่แบบดิจิตอล

วงจรกรองความถี่แบบดิจิตอล คือ กระบวนการที่ไปดัดแปลงสเปกตรัมของสัญญาณ ให้มีสเปกตรัมเป็นไปตามข้อกำหนดที่ต้องการ ซึ่งอาจเป็นการเพิ่มค่า หรือลดทอนค่าขนาดของสัญญาณในแถบความถี่ที่กำหนดให้ ซึ่งในการวิเคราะห์และสังเคราะห์วงจรนั้น ต้องใช้เครื่องมือพื้นทางคณิตศาสตร์เข้าช่วย ดังนั้นเราจึงนิยมเรียกว่า วงจรกรองความถี่เชิงเลข

การที่วงจรกรองความถี่เชิงเลขมีการนำมาประยุกต์ใช้งานกันอย่างกว้างขวางนั้น อาจมาจากข้อได้เปรียบหลายประการดังต่อไปนี้

1. ผลตอบสนองความถี่ของวงจรกรุงความถี่ สามารถออกแบบให้มีความใกล้เคียงกับผลตอบสนองความถี่ที่กำหนดให้ หรือผลตอบสนองความถี่ที่ต้องการได้ นอกจากนี้การออกแบบวงจรกรองความถี่ให้มีผลตอบสนองเชิงเส้นทำได้ง่าย

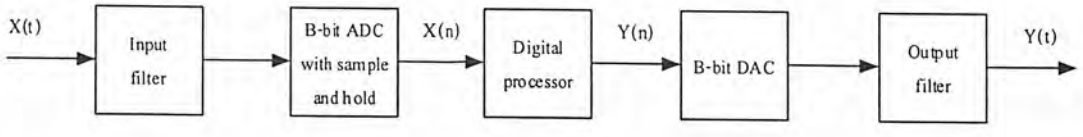
2. คุณสมบัติของวงจรกรองความถี่ที่ออกแบบและสร้างแล้วจะไม่ขยับเลื่อนไปตามสภาพแวดล้อม หรือตามอุณหภูมิ หรือตามระยะเวลาการใช้งาน นอกจากนี้ยังสามารถใช้งานในย่านความถี่ต่ำได้เป็นอย่างดี

3. การประยุกต์ใช้งานเป็นวงจรกรองความถี่แบบปรับตัวได้ (adaptive filter) ทำได้ง่าย

4. ผู้ออกแบบสามารถออกแบบโดยคำนึงถึงความของคำ (wordlength) ของตัวเลขฐานสองที่ต้องการใช้ และยังสามารถออกแบบให้มีผลตอบสนองความถี่ตามที่ต้องการได้

5. ในปัจจุบันถ้าพิจารณาในแง่มุมของเสถียรภาพของวงจรกรองความถี่ ความเชื่อถือได้ ราคา หรือขนาดของวงจรกรองความถี่เชิงเลข สิ่งเหล่านี้กำลังได้รับการพัฒนา และปรับปรุง และมีแนวโน้มว่าจะให้ผลลัพธ์ที่ดีกว่าของวงจรกรองความถี่แบบอนาล็อก (Analog filter) หรือเรียกว่า วงจรกรองความถี่เชิงอุปมาน

วงจรกรองเชิงเลขสามารถเขียนอธิบายในรูปของบล็อกไดอะแกรมได้ดังรูปที่ 2.1 โดยสัญญาณอินพุทซึ่งเป็นสัญญานอนาล็อกจะถูกสุ่ม (sampled) ด้วยช่วงเวลาที่ค่าคงที่ค่าหนึ่ง และสัญญาณที่ถูกสุ่มนี้จะถูกเปลี่ยนให้อยู่ในรูปเลขฐานสอง โดยการแปลงสัญญาณอนาล็อกให้เป็นสัญญาณดิจิตอลหรือสัญญาณเชิงเลข (analog to digital converter)



รูปที่ 2.1 บล็อกไดอะแกรมของวงจรกรองเชิงเลข

หลังจากนั้นเลขฐานสองที่แทนสัญญานอนาล็อกที่เข้ามาทางอินพุทจะถูกกรองโดยวงจรกรองเชิงเลข การกรองจะเป็นการคำนวณทางตัวเลข ซึ่งจะอาศัยวงจรที่ใช้ในระบบคอมพิวเตอร์ ได้แก่ ตัวบวก ตัวคูณ รีจิสเตอร์ และ

อุปกรณ์หน่วยความจำต่าง ๆ ต่อมาค่าเอาท์พุทที่ได้จากวงจรกรองเชิงเลขนี้จะถูกแปลงกลับเป็นสัญญาณอนาล็อกอีกทีหนึ่งเป็นสัญญาณเอาท์พุทที่จะนำไปใช้งานได้ วงจรกรองเชิงเลขสามารถแบ่งได้เป็น 2 ประเภท ตามลักษณะของผลตอบสนองอิมพัลส์ ได้แก่

1. วงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัด

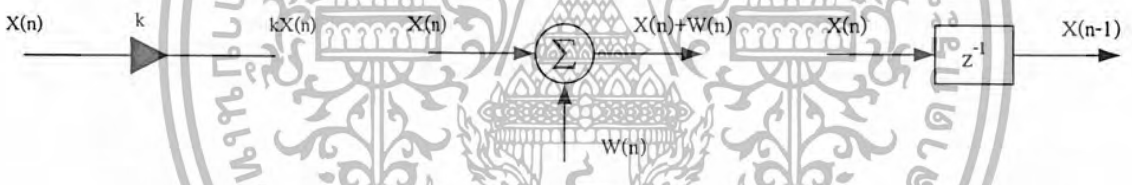
2. วงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์ไม่จำกัด

วงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัด มักเป็นตัวกรองที่ไม่มีการป้อนกลับ เป็นวงจรกรองที่มีโครงสร้างง่าย ๆ และมีเสถียรภาพที่ดี แต่มีข้อเสียที่จะใช้วงจรกรองที่มีอันดับสูงถึงแม้จะต้องการให้มีลักษณะทางความถี่ที่ง่าย ๆ ก็ตาม

ส่วนวงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์ไม่จำกัด เป็นตัวกรองที่มีการป้อนกลับเป็นวงจรกรองที่ใช้อันดับต่ำกว่าวงจรกรองแบบผลตอบสนองอิมพัลส์จำกัด ที่ความต้องการลักษณะทางความถี่เหมือนกัน แต่การออกแบบจะยุ่งยากกว่ามาก และมีปัญหาในเรื่องความมีเสถียรภาพไม่ดีนัก

2.1.2 โครงสร้างของวงจรกรองความถี่แบบดิจิตอล

วงจรกรองความถี่แบบดิจิตอล ประกอบไปด้วยส่วนที่สำคัญ 3 ส่วนคือ การบวก (Adder) การคูณ (Multiplier) และการหน่วง (Unit Delay) เวลาดังแสดงในรูปที่ 2.2 การบวกและการคูณจะใช้แนวความคิดมาจากตัวเลขในหน่วยคอมพิวเตอร์ ส่วนการหน่วงจะทำให้การถึงข้อมูลในอนาคตมีค่าอย่างต่อเนื่อง



รูปที่ 2.2 แสดงองค์ประกอบพื้นฐานทั้งสามที่ใช้เป็นส่วนประกอบของตัวกรองดิจิตอล

การหน่วงเวลานั้นจะแบ่งเป็น 2 ส่วน คือ บวก (positive) และลบ (negative) โดยการหน่วงแบบบวกนี้เป็นอุปกรณ์ที่ทำหน้าที่บันทึกความจำของรีจิสเตอร์ (register) จะเก็บค่าได้ตามระยะเวลาที่กำหนดสำหรับการคำนวณครั้งต่อไป การหน่วงแบบบวกจะแทนค่าด้วย Z^{-1} และสามารถอธิบายความสัมพันธ์ได้ด้วยการแปลงแซด การหน่วงแบบลบ ใช้แทนค่าต่อไปในระดับสัญญาณ แทนค่าด้วย Z จะมีชนิดและการใช้งานที่เหมาะสมอย่างไรก็ตาม การใช้งานก็ไม่สามารถใช้งานได้เสนอไป

วงจรกรองความถี่แบบดิจิตอล สามารถออกแบบให้มีค่าของการคูณที่แน่นอน และมีหลักการในการพิจารณาที่ไม่ยุ่งยากซับซ้อน

2.2 วงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัด หรือ FIR

คำว่า FIR ย่อมาจาก Finite Impulse Response ซึ่งหมายความว่า ผลตอบสนองอิมพัลส์จำกัดนั่นคือ หากเราป้อนสัญญาณอิมพัลส์ให้กับวงจรนี้แล้ว สัญญาณตอบสนองจะมีค่าจำกัด โดยสัญญาณเอาท์พุทของระบบจะขึ้นอยู่

กับสัญญาณอินพุทเท่านั้น จึงเรียกว่า วงจรกรองความถี่แบบไม่มีการป้อนกลับ (non-recursive filter) หรือไม่ป้อนกลับเชิงเลข ซึ่งสามารถเขียนสมการได้ดังนี้

$$y(n) = \sum_{k=-\infty}^{\infty} b_k x(n-k) \tag{2.1}$$

โดยที่ b_k เป็นค่าคงที่ใด ๆ ที่แทนค่าสัมประสิทธิ์ (coefficient) ของวงจรกรองความถี่และในทางปฏิบัติค่าจะมีค่าคงที่ ไม่ถึงกับมีค่าอนันต์ ขึ้นกับอันดับของวงจรกรองความถี่ ที่ต้องการใช้จะได้

$$y(n) = \sum_{k=0}^{N-1} b_k x(n-k) \tag{2.2}$$

และจะได้สมการคอนโวลูชั่น (convolution) เป็น

$$y(n) = \sum_{m=0}^{N-1} h(m)x(n-m) \tag{2.3}$$

เรานำมาเปลี่ยนค่าตัวแปรจะได้เป็น

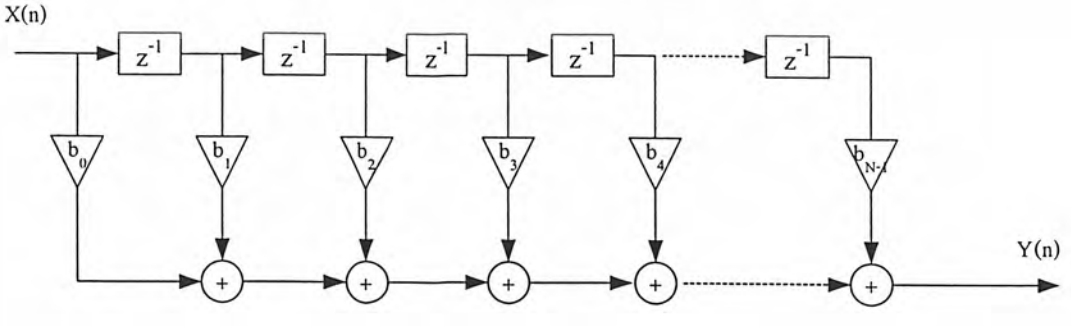
$$y(n) = \sum_{m=n}^{n-N+1} h(n-m)x(m) \tag{2.4}$$

เมื่อ x(n) เป็นอิทพุทและ h(n) เป็นผลตอบสนองอิมพัลส์ลำดับที่ N (length-N impulse response) เมื่อนำมาประยุกต์ใช้งานกับการแปลงแซด จะได้ฟังก์ชันถ่ายโอน

$$H(z) = \sum_{n=0}^{N-1} h(n)z^{-1} \tag{2.5}$$

แทนค่า $z = e^{j\omega}$ จะได้ผลตอบสนองความถี่ของวงจรกรองความถี่ไม่ป้อนกลับเชิงเลข

$$H(\omega) = \sum_{n=0}^{N-1} h(n)e^{-j\omega n} \tag{2.6}$$



รูปที่ 2.3 แสดงโครงสร้างวงจรกรองความถี่ดิจิตอลแบบ FIR

2.3 วงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์ไม่จำกัด หรือ IIR

คำว่า IIR ย่อมาจาก Infinite Impulse Response ซึ่งหมายความว่า ผลตอบสนองอิมพัลส์ไม่จำกัดหรือถึงอนันต์ (Infinite) ทั้งนี้ เพราะวงจรกรองความถี่ชนิดนี้มีคุณสมบัติประจำตัวที่สำคัญ คือ หากเราป้อนสัญญาณอิมพัลส์ให้กับวงจรนี้แล้ว สัญญาณตอบสนองจะไม่สิ้นสุด แต่จะมีไปจนถึงอนันต์ ดังนั้น อาจเรียกวงจรนี้ว่า วงจรกรองความถี่แบบป้อนกลับ (recursive filter) หรือป้อนกลับเชิงเลข เพราะสัญญาณเอาท์พุทจะขึ้นอยู่กับค่าสัญญาณอินพุทที่ป้อนเข้ามาและสัญญาณเอาท์พุทก่อนหน้านั้นโดยทั่วไป ถ้าให้สัมประสิทธิ์ของวงจรกรองความถี่เชิงเลขมีจำนวนจำกัด อาจเขียนเป็นสมการผลต่างสืบเนื่องที่ M (M^{th} order differential equation) สำหรับวงจรกรองความถี่ระบบเวลาจริงได้

$$y(n) = \sum_{k=0}^{M} b_k x(n-k) - \sum_{k=1}^{N} a_k y(n-k) \tag{2.7}$$

ผลตอบสนองความถี่สามารถหาได้จาก สมการผลต่าง โดยจาก

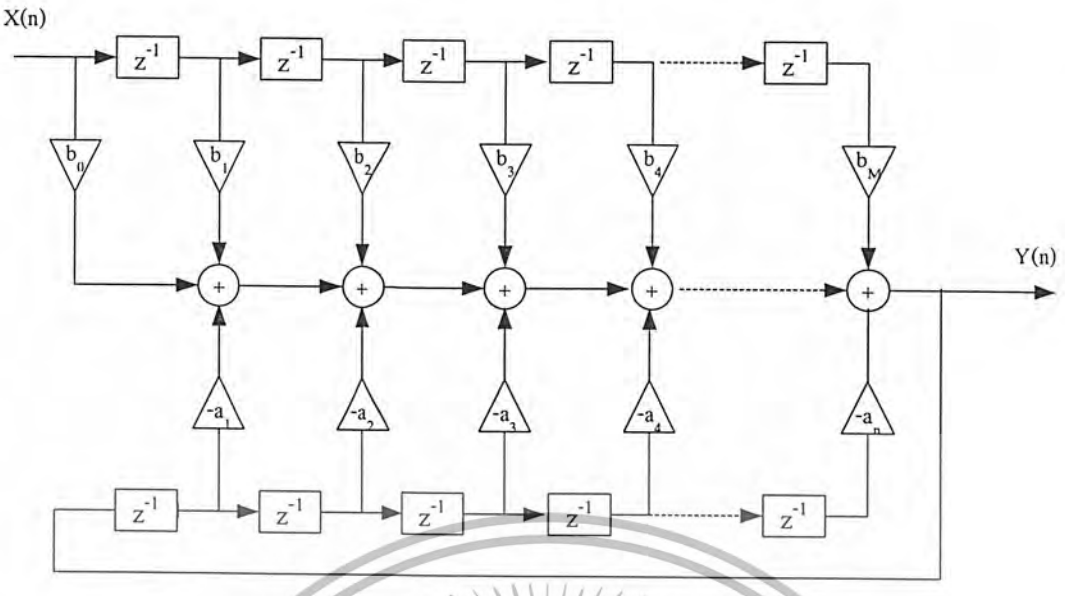
$$y(n) + \sum_{k=1}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k)$$

หรือฟังก์ชันถ่ายโอนเป็น

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}} \tag{2.8}$$

โดยในที่นี้ให้ $b_0 = 1$ และโดยการแทนค่าให้ $z = e^{j\omega}$ เราจะได้ผลตอบสนองความถี่ของวงจรกรองความถี่ป้อนกลับเชิงเลขแบบทั่วไปคือ

$$H(\omega) = \frac{\sum_{k=0}^{M} b_k e^{-j\omega k}}{1 + \sum_{k=1}^{N} a_k e^{-j\omega k}} \tag{2.9}$$



รูปที่ 2.4 แสดงโครงสร้างวงจรกรองความถี่ดิจิตอลแบบ IIR

2.4 ตัวกรองปรับตัวได้ (Adaptive Filter)

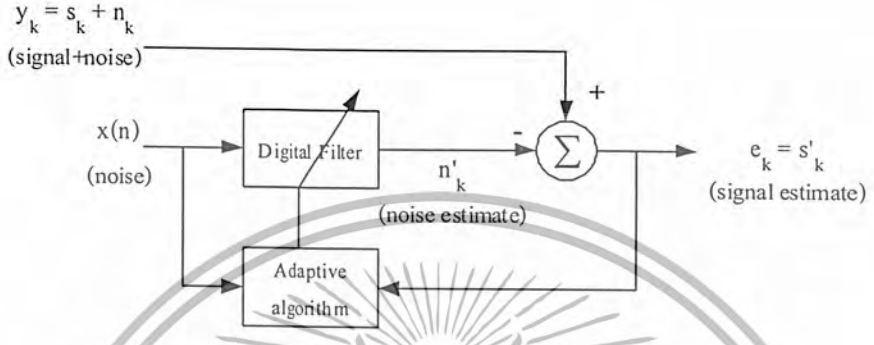
ตัวกรองปรับตัวได้มีความสามารถในการกรองสัญญาณ โดยมีลักษณะการปรับค่าสัมประสิทธิ์โดยอัตโนมัติ ซึ่งการออกแบบตามแนวความคิดที่ว่า จะบรรจุกลุ่มตัวแปร (Parameter) ที่ปรับค่าได้และค่านั้นจะถูกกำหนดโดยอัตโนมัติขึ้นอยู่กับการประมาณค่าคุณสมบัติทางสถิติของสัญญาณที่เกี่ยวข้อง ดังนั้นทฤษฎีของตัวกรองปรับตัวได้จะมีความสัมพันธ์ใกล้ชิดกับทฤษฎีตัวกรองความถี่ดิจิตอลทั่วไป ในการออกแบบต้องการจะจัดกลุ่มของตัวแปรที่ดีที่สุดจากความรู้เกี่ยวกับคุณสมบัติของสัญญาณที่เกี่ยวข้อง ให้เหมาะสมกับบรรทัดฐานและความต้องการ ปัญหาหลักของตัวกรองปรับตัวได้คือการหาอัลกอริทึม (Algorithm) ที่จะปรับค่าตัวแปรได้ดีในสถานการณ์ที่ความรู้เกี่ยวกับคุณสมบัติของสัญญาณที่เกี่ยวข้องมีน้อย ซึ่งจะให้การลู่เข้า (Converge) รวดเร็วเช่นเดียวกับในกรณีออกแบบจากกรณีรู้คุณสมบัติของสัญญาณ โดยทั่วไปนั้นตัวกรองปรับตัวได้จะทำงานโดยใช้กระบวนการทำงานซ้ำ ๆ (Iterative) ในการกรองและปรับค่าตัวแปรไปในเวลาเดียวกัน ตัวกรองปรับตัวได้มีสองลักษณะคือ

1. ตัวกรองปรับตัวได้แบบลูปเปิด (Open Loop) จะทำการเรียนรู้สถิติของสัญญาณที่เกี่ยวข้องและนำผลที่ได้ป้อนให้แก่อัลกอริทึมโดยไม่มีการป้อนกลับ (Non-Recursive) การกระทำลักษณะนี้เพื่อจะให้ได้การกรองที่ดีจะต้องการกลุ่มตัวแปรจำนวนมาก ทำให้ต้องสร้างฮาร์ดแวร์ที่ซับซ้อนซึ่งมีราคาแพง ตัวกรองพวกนี้ได้แก่ ตัวกรองปรับตัวได้แบบเอฟไอาร์

2. ตัวกรองปรับตัวได้แบบลูปปิด (Close Loop) จะทำการเรียนรู้สถิติของสัญญาณที่เกี่ยวข้องและปรับปรุงกลุ่มของค่าตัวแปรปัจจุบันจากสัญญาณที่เข้ามาใหม่และสัญญาณเอาท์พุทที่ได้จากการทำงานรอบที่แล้ว การทำงานเช่นนี้จะเป็นลักษณะของการป้อนกลับ (Recursive) มีข้อดีคือต้องการกลุ่มตัวแปรจำนวนน้อย ทำให้สามารถใช้ฮาร์ดแวร์ที่มีความซับซ้อนไม่มากนักได้ ตัวกรองพวกนี้ได้แก่ ตัวกรองปรับตัวได้แบบไอไอาร์

2.4.1 แนวคิดของตัวกรองปรับตัวได้

ต่อไปจะแสดงแนวคิดของตัวกรองปรับตัวได้โดยใช้การอธิบายจากการประยุกต์ใช้เป็นตัวลดสัญญาณรบกวน (Noise Canceller) พบว่าประกอบด้วยสองส่วนคือส่วนตัวกรองดิจิตอล (Digital Filter) และส่วนที่ใช้ปรับค่าสัมประสิทธิ์ (Adaptive Algorithm) ซึ่งจะใช้ในการปรับค่าหรือปรับปรุงค่าสัมประสิทธิ์ของตัวกรองดิจิตอล ดังรูป



รูปที่2.5 แสดงตัวลดสัญญาณรบกวน

จากรูป 2.5 สัญญาณอินพุท 2 สัญญาณ y_k และ x(n) ถูกป้อนอย่างต่อเนื่องโดยสัญญาณ y_k ประกอบด้วยสัญญาณที่ต้องการ s_k และสัญญาณรบกวน n_k โดยสมมติว่าสัญญาณ x(n) เป็นสัญญาณที่จะใช้ในการตรวจสอบสัญญาณรบกวนที่ปะปนมาและมีความสัมพันธ์กับสัญญาณรบกวน n_k จะพบว่าจากการนำสัญญาณ x_k ไปผ่านตัวกรองดิจิตอลเพื่อทำการประมาณค่าเป็นสัญญาณรบกวนโดยประมาณ n'_k ดังนั้นสัญญาณที่ต้องการจะหาได้จากการหักล้างของเอาท์พุทของตัวกรองดิจิตอลจากสัญญาณ y_k ตามสมการ

$$s'_k = y_k - n'_k = s'_k + n_k - n'_k$$

วัตถุประสงค์ที่สำคัญใน Noise Canceling ก็เพื่อจะสร้างค่าประมาณที่ดีที่สุดของสัญญาณที่ถูกรบกวน s'_k กลับมา

2.4.2 กรรมวิธีการพัฒนาอัลกอริทึมของตัวกรองปรับตัวได้แบบต่าง ๆ

ในการพัฒนาแต่ละอัลกอริทึมจะพบว่าแต่ละอัลกอริทึมมีคุณลักษณะเฉพาะตัว จึงมีความสำคัญมากในการที่จะทำความเข้าใจในความสามารถและขีดจำกัดเพื่อให้การเลือกใช้งานมีประสิทธิภาพ

1. กรรมวิธีการแบบสตอแคสติก เกรเดียนท์ (Stochastic Gradient Apoproach)

เป็นกรรมวิธีที่วัดค่าความผิดพลาดโดยใช้ค่าเฉลี่ยผลต่างกำลังสองและพยายามลดค่าเฉลี่ยผลต่างกำลังสองโดยการทำการหาค่าจุดต่ำสุดบนเออร์เรอร์เพอร์ฟอร์มานซ์เซอร์เฟส (Error Performance Surface) ซึ่งจุดต่ำสุดจะให้ค่าสัมประสิทธิ์ที่ทำให้ค่าเฉลี่ยผลต่างกำลังสองมีค่าต่ำสุด การหาอัลกอริทึมเพื่อใช้ในการปรับค่าสัมประสิทธิ์จะทำการแก้สมการของวีนเนอร์และฮอร์พ (Weiner-Hopf Equation) อัลกอริทึมในกลุ่มนี้ที่นิยมใช้คือลีสต์มีนสแควร์ (Least Mean Square) หรือเรียกอีกอย่างว่าแอลเอ็มเอสอัลกอริทึม (LMS Algorithm)

2. กรรมวิธีการประมาณแบบลีสต์สแควร์ (Least Square Estimation)

กรรมวิธีนี้ใช้วิธีการลดคอสต์ ฟังก์ชั่น (Cost Function) โดยอาจใช้วิธีการของรีเคอร์ซีฟ ลีสต์สแควร์ (Recursive Least Square:RLS) ซึ่งสามารถแบ่งออกเป็น 3 อัลกอริทึม ดังนี้

1. รีเคอร์ซีฟ ลีสต์สแควร์ อัลกอริทึมมาตรฐาน (Standard RLS Algorithm)
2. รีเคอร์ซีฟ ลีสต์สแควร์ อัลกอริทึมโดยการถอดราก (Square Root RLS Algorithm)
3. รีเคอร์ซีฟ ลีสต์สแควร์ อัลกอริทึมแบบเร็ว (Fast RLS Algorithm)

2.4.3 การวัดประสิทธิภาพในระบบปรับตัว

ในระบบปรับตัวทุกระบบมีความจำเป็นที่จะต้องทำการวัดประสิทธิภาพ โดยทำเปรียบเทียบระหว่างอัลกอริทึมต่าง ๆ ในตัวกรองปรับตัวก็เช่นกัน นอกจากจะต้องทำการเปรียบเทียบประสิทธิภาพของตัวกรองแล้ว ยังต้องทำการเปรียบเทียบประสิทธิภาพของอัลกอริทึมอีก เพื่อช่วยในการตัดสินใจเลือกใช้อัลกอริทึมที่ให้การตอบสนองที่ดี การวัดประสิทธิภาพของอัลกอริทึมมีหลายวิธีการดังจะได้กล่าวต่อไปนี้

1. ความเร็วในการลู่เข้า (Convergence Rate)

อัตราการลู่เข้าของระบบปรับตัวเป็นคุณสมบัติที่สำคัญมากซึ่งจะต้องทำการวัดเพื่อให้ได้ตามความต้องการของงานที่จะนำไปใช้ซึ่งโดยทั่วไปแล้วความเร็วในการลู่เข้านั้นสามารถเปรียบเทียบประสิทธิภาพของอัลกอริทึมได้ อย่างไรก็ตามการเลือกใช้อัลกอริทึมไม่จำเป็นต้องใช้อัลกอริทึมที่มีความเร็วสูงสุดเนื่องจากการเพิ่มความเร็วราคาในการสร้างและความซับซ้อนของวงจรก็จะสูงขึ้นเช่นกัน การเลือกใช้จึงขึ้นอยู่กับความจำเป็นในการใช้งานมากกว่า

2. ค่าความผิดพลาดเฉลี่ยกำลังสอง (Mean Square Error)

จะวัดในลักษณะของค่าความผิดพลาดเฉลี่ยกำลังสองที่น้อยที่สุด (Minimum Mean Square Error : MMSE) ซึ่งใช้ในการวัดความสามารถในการทำงานของระบบในการลดสัญญาณรบกวน (Eliminating Noise), ทำนายสัญญาณ (Signal Prediction) หรือแยกแยะระบบ (System Identifying) แล้วแต่ว่าเป็นระบบอะไร โดยทั่วไปแล้วค่าความผิดพลาดเฉลี่ยกำลังสองขึ้นกับหลายปัจจัย เช่น โครงสร้างของตัวกรอง ความไวของสัมประสิทธิ์ ตลอดจนสัญญาณรบกวนทั่ว ๆ ไป

3. ความถูกต้องในการประมาณค่าตัวแปร (Parameter Estimation Accuracy)

ความถูกต้องในการประมาณค่าตัวแปรเป็นปัจจัยที่มีความสำคัญมากเมื่อมีการนำระบบปรับตัวไปใช้ในการหาคุณลักษณะของระบบที่ไม่ทราบ (System Identifying) ซึ่งความถูกต้องในการประมาณค่าตัวแปรมีมาก จะทำให้ได้แบบจำลองของระบบที่ต้องการใกล้เคียงความจริงมากขึ้น

4. ความซับซ้อนในการประมวลผล (Computation Complexity)

ความซับซ้อนในการประมวลผลมีความสำคัญมากในการนำระบบปรับตัวไปใช้งานจริง โดยทั่วไปต้องการความซับซ้อนในการประมวลผลที่ต่ำ ทำให้การสร้างมีราคาถูก นอกจากนั้นหากนำไปโปรแกรมลงในตัวประมวลผลสัญญาณสำเร็จรูปแล้วนั้นก็ต้องคำนึงถึงความสามารถในการทำงานของตัวประมวลผลเปรียบเทียบกับความซับซ้อนในการประมวลผลของอัลกอริทึมด้วย

5. เสถียรภาพ (Stability)

ความมีเสถียรภาพของระบบมีความสำคัญอย่างมากในการใช้งานตัวกรองปรับตัวได้โดยเฉพาะตัวกรองแบบไอไออาร์ เนื่องจากอาจเกิดการเลื่อนของโพลออกนอกวงกลมหนึ่งหน่วย ทำให้เกิดการออสซิเลตและทำให้ค่าลู่ออก (Diverge) จนไม่สามารถทำงานได้ การออกแบบจึงต้องคำนึงถึงเสถียรภาพของตัวกรองด้วย

6. ความคงทนของระบบ (Robustness)

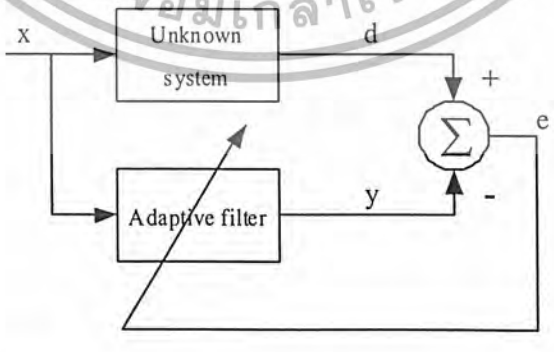
เนื่องจากค่าอินพุทของตัวกรองปรับตัวได้มักมีค่าต่าง ๆ ที่ไม่แน่นอนจึงมีความจำเป็นที่ตัวกรองจะต้องมีความทนต่อความแปรปรวนของสัญญาณ โดยทั่วไปความคงทนของระบบเป็นค่าที่ยากต่อการวัดในการทดลองโดยทั่วไปจึงไม่ทำการวัดคุณลักษณะนี้

2.4.4 การประยุกต์ใช้งานตัวกรองปรับตัวได้

ตัวกรองปรับตัวได้มีคุณสมบัติในการปรับตัวได้อย่างอัตโนมัติทำให้ถูกนำมาประยุกต์ใช้ในอุปกรณ์หลาย ๆ อย่าง ต่อไปจะทำการอธิบายตัวอย่างการนำตัวกรองปรับตัวได้ไปประยุกต์ใช้งาน

1. การประยุกต์ในการจำลองคุณลักษณะของระบบที่ไม่ทราบ (System Identification)

สมมติว่ามีระบบที่ไม่เป็นที่รู้จักและต้องการพิจารณาผลตอบสนองของระบบนั้นต่อสัญญาณที่ป้อนให้กับระบบ โดยสมมุติว่าระบบนี้ไม่เปลี่ยนแปลงไปตามเวลาและมีคุณสมบัติเชิงเส้น และต้องการที่จะพัฒนาแบบจำลอง (Model) สำหรับระบบนี้โดยใช้ตัวกรองปรับตัว เพื่อที่จะสร้างสัญญาณเอาท์พุทให้เหมือนกับเอาท์พุทของระบบที่ไม่เป็นที่รู้จัก โดยการป้อนอินพุทที่เหมือนกันให้แก่ระบบทั้งสองและทำการเปรียบเทียบเอาท์พุททั้งสอง เพื่อสร้างสัญญาณผิดพลาด ซึ่งเป็นความแตกต่างระหว่างสัญญาณทั้งสอง อาจกล่าวได้ว่าการทำงานของตัวกรองปรับตัวเป็นการพยายามปรับผลตอบสนองให้เหมือนกับผลตอบสนองของระบบที่ไม่เป็นที่รู้จัก เพื่อที่จะทำให้สัญญาณผิดพลาดมีน้อยที่สุด ซึ่งวิธีการที่ถูกใช้บ่อย ๆ ในทางปฏิบัติสำหรับการทำให้ค่าความผิดพลาดน้อยที่สุดคือ ค่าความผิดพลาดเฉลี่ยกำลังสอง และสำหรับในกรณีที่สัญญาณอินพุทของระบบที่ไม่เป็นที่รู้จักนั้นคงที่และระบบที่ไม่เป็นที่รู้จักไม่เปลี่ยนแปลงไปตามเวลาจะทำให้ค่าความผิดพลาดลดลงสู่ศูนย์ (Convergence) ในขณะนั้นเองกล่าวได้ว่าตัวกรองปรับตัวทำการจำลองคุณลักษณะของระบบที่ไม่ทราบได้แล้ว



รูปที่ 2.6 แสดงการประยุกต์ใช้ตัวกรองปรับตัวในการจำลองคุณลักษณะของระบบที่ไม่ทราบ

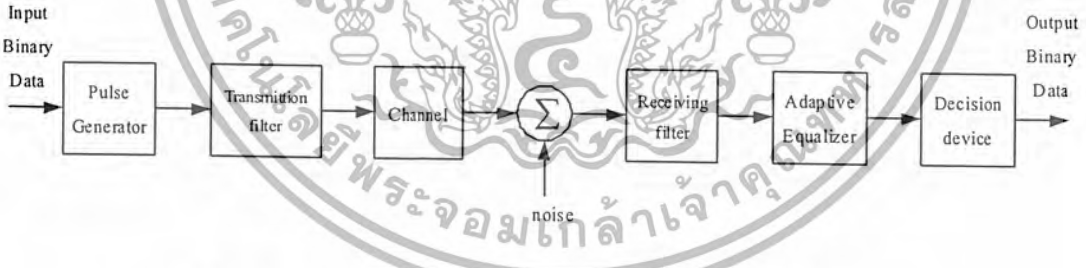
2. การพัฒนาอิควอไลเซอร์แบบปรับตัวได้ (Adaptive Equalization)

ระบบการส่งข้อมูลทั่วไปจะพยายามใช้ประโยชน์จากช่องสัญญาณที่มีอยู่ให้มีประสิทธิภาพที่สุด โดยออกแบบระบบการส่งข้อมูลด้วยอัตราสูงที่สุดเท่าที่เป็นไปได้ ภายใต้ความน่าเชื่อถือที่กำหนด ซึ่งปกติถูกวัดอยู่ในรูปของอัตราการผิดพลาดหรือความน่าจะเป็นเฉลี่ยของการผิดพลาดของสัญลักษณ์ (Average probability of symbol error) การส่งข้อมูลดิจิตอลผ่านช่องทางการสื่อสารแบบเชิงเส้นถูกจำกัดด้วยองค์ประกอบสองอย่าง

1. การแทรกซ้อนระหว่างสัญลักษณ์ เกิดเป็นการแทรกซ้อนของสัญลักษณ์ที่ถูกส่ง ซึ่งเป็นผลมาจากการเบี่ยงเบนของการตอบสนองความถี่ในช่องทาง
2. การรบกวนจากเทอร์มอลน้อยส์ เกิดจากสัญญาณรบกวนซึ่งเกิดจากอุณหภูมิ ซึ่งโดยทั่วไปมักเกิดในอุปกรณ์ต่าง ๆ ของระบบสื่อสาร

สำหรับช่องทางสื่อสารที่จำกัดเช่น โทรศัพท์ปกติจะพบว่าการแทรกซ้อนระหว่างสัญลักษณ์เป็นองค์ประกอบสำคัญที่ใช้พิจารณาในการกำหนดอัตราส่งของข้อมูล

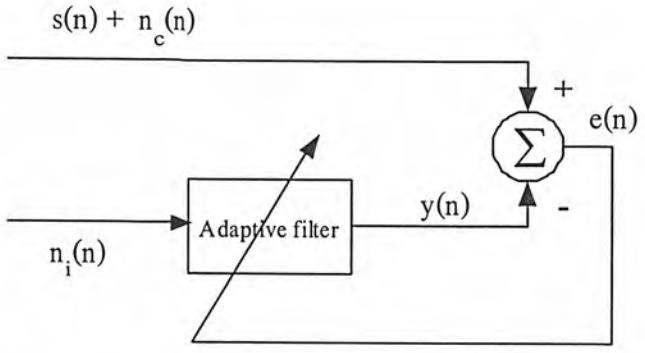
เพื่อที่จะแก้ปัญหาการแทรกซ้อนระหว่างสัญลักษณ์ต้องมีตัวกรองโดยถ้าคุณลักษณะของช่องสัญญาณเป็นที่รู้อย่างแน่นอนแล้ว จะสามารถออกแบบตัวกรองทั้งภาคส่งและภาครับได้ ซึ่งจะลดผลของการแทรกซ้อนระหว่างสัญลักษณ์ลงได้และยังช่วยลดสัญญาณรบกวนที่เพิ่มขึ้นมาแต่อย่างไรก็ตามในทางปฏิบัติ เราพบว่าคุณลักษณะของช่องสัญญาณสื่อสารนั้นเป็นแบบสุ่มฉะนั้นการใช้ตัวกรองภาคส่งและภาครับที่ออกแบบ โดยคุณลักษณะของช่องสัญญาณเฉลี่ยจะไม่เพียงพอที่จะลดการแทรกซ้อนระหว่างสัญลักษณ์ จึงจำเป็นที่จะต้องมีอิควอไลเซอร์แบบปรับตัวได้ ซึ่งจะลดการแทรกซ้อนระหว่างสัญลักษณ์ได้ดีขึ้น โดยพื้นฐานสำหรับการอิควอไลซ์ ของระบบการส่งข้อมูลที่มีอยู่ คือการเพิ่มขนาดสัญญาณที่บางย่าน (Pre-Equalization) ที่ภาคส่งและการลดขนาดสัญญาณที่บางย่าน (Post-Equalization) ที่ภาครับ



รูปที่ 2.7 แสดงการใช้อิควอไลเซอร์แบบปรับตัวได้ในระบบส่งสัญญาณ

3. การกำจัดสัญญาณรบกวน (Noise Cancellation)

การนำตัวกรองปรับตัวได้มาประยุกต์ใช้ในการกำจัดสัญญาณรบกวนนั้น โดยใช้สัญญาณรบกวนที่มีความสัมพันธ์กับสัญญาณรบกวนที่ปนมาในสัญญาณที่ต้องการ ในการกำจัดสัญญาณรบกวน โดยการป้อนสัญญาณและโครงสร้างของระบบกำจัดสัญญาณรบกวนเป็นไปตามรูป



รูปที่ 2.8 แสดงโครงสร้างระบบกำจัดสัญญาณรบกวนโดยใช้ตัวกรองปรับตัวได้

จากรูปป้อนสัญญาณรบกวน $n_i(n)$ ซึ่งมีความสัมพันธ์กับสัญญาณรบกวน $n_c(n)$ ให้แก่ระบบจะได้เอาท์พุทของตัวกรองปรับตัว $y(n)$ แล้วจึงนำไปหักล้างกับสัญญาณรบกวน $n_c(n)$ ซึ่งปะปนอยู่กับสัญญาณที่ต้องการ $s(n)$ ทำให้ได้สัญญาณที่ต้องการ $e(n)$ ซึ่งเป็นสัญญาณที่มีสัญญาณรบกวนน้อยลงและสัญญาณนี้จะถูกป้อนให้กับตัวกรองเพื่อใช้ในการเปรียบเทียบในการปรับค่าสัมประสิทธิ์อีกทีหนึ่ง

4. การพัฒนาอุปกรณ์เพิ่มประสิทธิภาพสาย (Adaptive Line Enhancer)

อุปกรณ์เพิ่มประสิทธิภาพสาย เป็นอุปกรณ์ที่สามารถใช้แยกสัญญาณแบนด์แคบซึ่งฝังอยู่ในสัญญาณแบนด์กว้าง ซึ่งขึ้นกับการใช้งาน สัญญาณที่ต้องการจะแยกออกอาจเป็นสัญญาณที่สนใจหรือสัญญาณรบกวนที่ไม่ต้องการ ตัวอย่างของการใช้เช่น การลดสัญญาณรบกวนความถี่ 50-60 Hz ออกจากสัญญาณของอุปกรณ์ทางการแพทย์

รูปที่ 2.9 แสดงอุปกรณ์เพิ่มประสิทธิภาพสาย

2.5 ตัวกรองปรับตัวได้แบบเอฟไอาร์ (Adaptive FIR Filter)

ในการศึกษาตัวกรองปรับตัวนั้น โดยทั่วไปมักจะเริ่มจากการศึกษาตัวกรองปรับตัวได้แบบเอฟไอาร์เสียก่อน เนื่องจากตัวกรองปรับตัวแบบเอฟไอาร์มีคุณสมบัติบางประการที่ดีกว่า กล่าวคือ เสถียรภาพของระบบที่ดีกว่าเนื่องจากสัมประสิทธิ์ของตัวกรองที่มีค่าจำกัด อัลกอริทึมที่ใช้ในการปรับตัวก็ง่ายกว่า นอกจากนั้นแล้วประสิทธิภาพของตัวกรองก็สามารถมองได้ในลักษณะของการลู่เข้าและเสถียรภาพ ต่อไปจะแสดงการหาอัลกอริทึมสำหรับตัวกรองปรับตัวแบบเอฟไอาร์ โดยใช้อัลกอริทึมแบบลีสต์มีนสแควร์หรืออัลกอริทึมแบบแอลเอ็มเอส

ตัวกรองแบบเอฟไอาร์ ซึ่งมีสัญญาณอินพุทเป็น x(n) มีสมการทั่วไปคือ

$$y(n) = \sum_{i=0}^{N-1} w_i(n)x(n-i) \tag{2.10}$$

ให้สัญญาณที่ต้องการเป็น แล้วสัญญาณความผิดพลาด สามารถนิยามโดย

$$e(n) = d(n) - y(n) \tag{2.11}$$

สำหรับความต้องการโดยทั่วไปของตัวกรองปรับตัว การเลือกสัมประสิทธิ์ $w_0(n)$, $w_1(n)$,..., $w_i(n)$ นั้นจะอยู่บนพื้นฐานของการลดสัญญาณความผิดพลาดให้น้อยที่สุด โดยยึดเอาค่าเฉลี่ยกำลังสองในการเปรียบเทียบ

โดยทั่วไปนั้นแอลเอ็มเอสอัลกอริทึมจะใช้วิธีการปรับสัมประสิทธิ์แบบ สตีปเปสต์ เดสเซนท์ (Steepest Descent) ซึ่งก็คือ

$$W(n+1) = W(n) - \mu\nabla e^2(n) \tag{2.12}$$

เมื่อ $W(n) = [w_0(n) \quad w_1(n) \quad \ldots \quad w_{N-1}(n)]^T$ เป็นเมตริกซ์สัมประสิทธิ์ μ เป็นค่าขนาดสเตป และ ∇ เป็นค่าเกรเดียนท์เวคเตอร์ซึ่งกำหนดโดย

$$\nabla = \left[\frac{\partial}{\partial w_0} \quad \frac{\partial}{\partial w_1} \quad \cdots \quad \frac{\partial}{\partial w_{N-1}}\right]^T \tag{2.13}$$

ดังนั้นเกรเดียนท์ตัวที่ i ของเกรเดียนท์เวคเตอร์ $\nabla e^2(n)$ คือ

$$\frac{\partial e^2(n)}{\partial w_i} = 2e(n)\frac{\partial e(n)}{\partial w_i} \tag{2.14}$$

แทนค่า $e(n) = d(n) - y(n)$ ลงในสมการข้างต้นจะได้เป็น

$$\frac{\partial e^2(n)}{\partial w_i} = -2e(n)\frac{\partial y(n)}{\partial w_i} \tag{2.15}$$

แทนค่า $y(n) = \sum_{i=0}^{N-1} w_i(n)x(n-i)$ ลงในสมการข้างต้นจะได้เป็น

$$\frac{\partial e^2(n)}{\partial w_i} = -2e(n)x(n-i) \tag{2.16}$$

ดังนั้นจากสมการข้างต้นจะพบว่า

$$\nabla e^2(n) = -2e(n)X(n) \tag{2.17}$$

เมื่อ $X(n) = [x(n) \quad x(n-1) \quad \dots \quad x(n-N+1)]^T$ หลังจากทำการแทนค่าแล้วจะได้สมการปรับสัมประสิทธิ์เป็น

$$W(n+1) = W(n) + 2\mu e(n)X(n) \tag{2.18}$$

จากสมการทั้งหมดเราจะได้อัลกอริทึมโดยสรุปเป็น

อินพุท: เวคเตอร์สัมประสิทธิ์ $W(n) = [w_0(n) \quad w_1(n) \quad \dots \quad w_{N-1}(n)]^T$

อินพุทเวคเตอร์ $X(n) = [x(n) \quad x(n-1) \quad \dots \quad x(n-N+1)]^T$

สัญญาณที่ต้องการ d(n)

เอาท์พุท:ฟิลเตอร์เอาร์พุท y(n)

เวคเตอร์สัมประสิทธิ์ที่ปรับแล้ว W(n+1)

1. การกรองสัญญาณ

$y(n) = W^T(n)X(n)$

2. การประมาณค่าความผิดพลาด

$e(n) = d(n) - y(n)$

3. การปรับสัมประสิทธิ์

$W(n+1) = W(n) + 2\mu e(n)X(n)$

2.6 หลักการเบื้องต้นของโครงสร้างเลขคณิตกระจาย

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

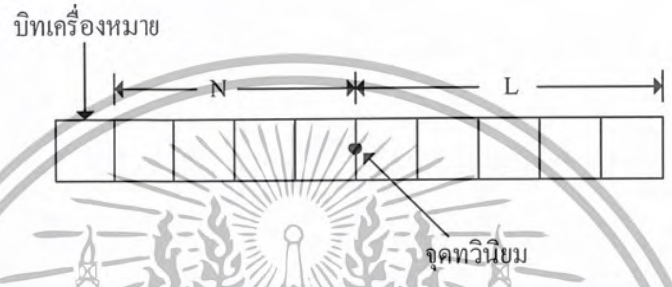
2.6.1 ระบบตัวเลข

สำหรับระบบเชิงเลข ตัวเลขต่างๆจะถูกแทนด้วยเลขฐานสอง ซึ่งโดยทั่วไปมีรูปแบบที่นิยมใช้กันอยู่ 2 รูปแบบ คือ รูปแบบจำนวนโดยตรง (Fixed point format) และ รูปแบบจำนวนอิงดรรชนี (Floating point format) ซึ่งรูปแบบจำนวนโดยตรงจะมีวงจรฮาร์ดแวร์ที่ใช้ในการคำนวณที่ง่ายกว่า แต่ให้ค่าจากการคูณค่อนข้างจำกัด ส่วน

รูปแบบจำนวนอิงดรรชนีจะสามารถแทนค่าของสัญญาณ คือให้ย่านพลวัตร (Dynmic range) ได้มากกว่า แต่ต้องใช้วงจรฮาร์ดแวร์ที่สลับซับซ้อน แพงกว่า และให้ความเร็วในการประมวลผลที่ลดลง

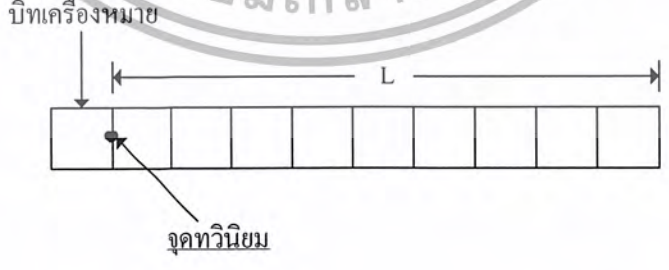
1. รูปแบบจำนวนโดยตรง

รูปแบบจำนวนโดยตรงปกติจะประกอบไปด้วย 3 ส่วน คือ บิทเครื่องหมาย (Sign bit) 1 บิท บิทจำนวนเต็ม (Integer bit) N บิท และบิทเศษส่วน (Fractional bit) L บิท โดยจะมีจุดทวินิยม (Binary point) อยู่ระหว่างบิทจำนวนเต็มและบิทเครื่องหมายดังแสดงในรูปที่ 2.10



รูปที่ 2.10 แสดงการจัดรูปแบบจำนวนโดยตรงที่ประกอบด้วยบิทจำนวนเต็มและบิทเศษส่วน

จำนวนบิท N เป็นตัวกำหนดย่านพลวัตรที่ต้องการ โดยถ้าเลือกให้มีจำนวนน้อยอาจทำให้เกิดการล้น (Overflow) จากการคำนวณได้ แต่ถ้าเลือกให้มีจำนวนมากความเที่ยงตรงก็จะน้อยลง ซึ่งในการสร้างวงจรกรองสัญญาณเชิงเลขโดยการแทนด้วยรูปแบบจำนวนโดยตรงนั้น นิยมที่จะทำมาตราส่วน (Scaling) เพื่อให้ขนาดของสัญญาณมีค่าอยู่ระหว่าง $-1 \le x < 1$ คือมีบิทเครื่องหมาย 1 บิท และบิทเศษส่วน L บิท ดังแสดงในรูปที่ 2.11



รูปที่ 2.11 แสดงการจัดรูปแบบจำนวนโดยตรงที่มีแต่บิทเศษส่วน

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

ตารางที่ 2.1 แสดงคุณสมบัติที่สำคัญของรูปแบบจำนวนโดยตรง

คุณลักษณะ	ขนาดและเครื่องหมาย	ส่วนเติมเต็มสอง	ส่วนเติมเต็มหนึ่ง
ค่าช่วง	$-(1-2^{-L}) \le x \le (1-2^{-L})$	$-1 \le x \le (1-2^{-L})$	$-(1-2^{-L}) \le x \le (1-2^{-L})$
การแทนค่าเลขศูนย์	0.000 และ 1.000	0.000	0.000 และ 1.111
กฎทางคณิตสาสตร์	ง่าย แต่ต้องมีการแยกเก็บขนาดและเครื่องหมาย	ง่าย ต่อการคำนวณเลขจำนวนลบ	ง่าย แต่ควรระมัดระวังในการคำนวณเมื่อมีการทดเกิดขึ้น
ความเหมาะสมสำหรับรูปแบบจำนวนโดยตรง	ไม่ดี	ดีมาก	ดี

ใน 3 รูปแบบนี้ตัวเลขแบบส่วนเติมเต็มสองเป็นที่นิยมใช้กันมากในระบบการประมวลผลสัญญาณเชิงเลข ทั้งนี้เนื่องมาจาก

1. มีการแทนค่าเลขศูนย์ได้เพียงค่าเดียว

2. การสร้างวงจรฮาร์ดแวร์สำหรับการบวก ลบ และคูณ ของเลขส่วนเติมเต็มสองทำได้ง่ายโดยในการคูณสามารถใช้หลักการเลื่อนและบวก (Shift and add)

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

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

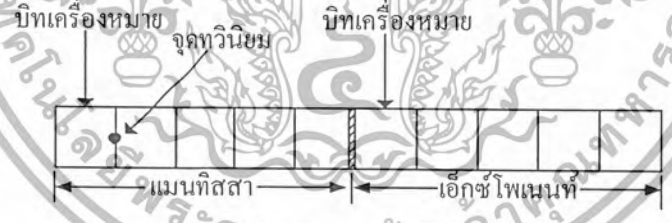
2 รูปแบบจำนวนอิงดรรชนี

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

$$X = M \times 2^{e} \tag{2.19}$$

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

M และ e เรียกว่า แมนทิสสา (Mantissa) และ เอ็กซ์โพเนนท์ (Exponent) ตามลำดับ ตัวอย่างเช่น จำนวน 0.00110101 และ 01001.11 สามารถแทนได้โดย 0.110101×2^{-2} และ 0.100111×2^{4} ตามลำดับ ส่วนจำนวนที่มีค่าเป็นลบก็ทำในลักษณะเดียวกัน รูปแบบจำนวนอิงดรรชนีสามารถแสดงได้ดังรูปที่ 2.12 โดยแบ่งเป็น 2 ส่วน คือ ส่วนหนึ่งสำหรับแมนทิสสา และอีกส่วนสำหรับเอ็กซ์โพเนนท์



รูปที่ 2.12 แสดงการจัดรูปแบบจำนวนอิงดรรชนี

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

2.6.2 ทฤษฎีเลขคณิตกระจาย

จากที่ได้กล่าวมาแล้วว่า โครงสร้างเลขคณิตกระจายมีพื้นฐานอยู่บนการคูณแบบเลขส่วนเติมเต็มสอง ดังนั้นในหัวข้อนี้จะได้อธิบายถึงหลักการของการคูณเลขส่วนเติมเต็มสอง

ให้เลขส่วนเติมเต็มสองของ X ซึ่งแทนด้วย $\overline{X}$ และนิยามโดย

$$\bar{x} = \begin{cases} x5 & \text{ถ้า} X \geq 0 \\ 2-|x| & \text{ถ้า} X < 0 \end{cases} \tag{2.20}$$

โดย X เป็นเลขที่เป็นเศษส่วน (Fractional number)

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

$$\overline{X} = X_0.X_1X_2...X_L \tag{2.21}$$

ค่าของ $\overline{X}$ ในรูปของเลขฐานสิบสามารถหาได้ดังนี้

$$X = -X_0 + \sum_{i=1}^{L} X_i 2^{-i} \tag{2.22}$$

จากนั้นพิจารณาผลคูณต่อไปนี้

$$Y = Xm \tag{2.23}$$

ให้ $\overline{Y}$, $\overline{X}$ และ $\overline{m}$ เป็นเลขส่วนเติมเต็มสองของ Y, X และ m ตามลำดับ จากนั้นพิจารณาจากสมการที่ (2.22) และ สมการที่ (2.23) จะได้

$$\begin{aligned} Y &= -Y_0 + \sum_{i=1}^{L} Y_i 2^{-i} \\ &= -X_0 m + \sum_{i=1}^{L} X_i m 2^{-i} \end{aligned} \tag{2.24}$$

ดังนั้น

$$\begin{aligned} \overline{Y} &= \text{ส่วนเติมเต็มสองของ} \left(-X_0 m + 2^{-1}X_1 m + 2^{-2}X_2 m + 2^{-3}X_3 m + ... + 2^{-L}X_L m\right) \\ &= \text{ส่วนเติมเต็มสองของ} \left(-X_0 m + 2^{-1}(X_1 m + ... + 2^{-1}(X_{L-1} m + 2^{-1}(X_L m)))\right) \end{aligned} \tag{2.25}$$

ต่อไปพิจารณาส่วนเติมเต็มสองของ $2^{-1}U$ โดย

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

สำหรับ $U \geq 0$ (หรือ $U_0 = 0$)

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

และสำหรับ $U < 0$ (หรือ $U_0 = 1$)

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

ดังนั้นสรุปได้ว่า

$$\text{ส่วนเติมเต็มสองของ } (2^{-1}U) = \begin{cases} 2^{-1}\overline{U} & \text{ถ้า } U_0 = 0 \\ 1 + 2^{-1}\overline{U} & \text{ถ้า } U_0 = 1 \end{cases} \quad (2.26)$$

สมการที่ (2.26) นี้แสดงให้เห็นได้ว่า ส่วนเติมเต็มสองของ $(2^{-1}U)$ เป็นการเลื่อนข้อมูลของ $\overline{U}$ ไปทางขวา 1 บิท

$$\therefore \text{ส่วนเติมเต็มสองของ } (2^{-1}U) = 2_2^{-1}\overline{U} \quad (2.27)$$

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

จากนั้นพิจารณาสมการที่ (2.25) และสมการที่ (2.26) จะได้ว่า

$$\begin{aligned} \overline{Y} &= -X_0\overline{m} + 2^{-1}X_1\overline{m} + 2^{-2}X_2\overline{m} + 2^{-3}X_3\overline{m} + ... + 2^{-L}X_L\overline{m} \\ &= -X_0\overline{m} + 2^{-1}(X_1\overline{m} + ... + 2^{-1}(X_{L-1}\overline{m} + 2^{-1}(X_L\overline{m}))) \end{aligned} \quad (2.28)$$

ซึ่งจากสมการที่ (2.28) จะเห็นได้ว่าผลคูณจากสมการที่ (2.23) สามารถหาได้โดยการใช้หลักการเลื่อนและบวก (Shift and add) โดยผลลัพธ์ที่ได้จากการคูณแบบเลขส่วนเติมเต็มสอง สามารถหาได้ตามขั้นตอนดังนี้

1. เคลียร์ค่าข้อมูลในแอคคิวมูเลเตอร์รีจิสเตอร์
2. บวก $X_L\overline{m}$ กับค่าที่อยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์
3. เลื่อนค่าที่อยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์ไปทางขวา 1 บิท

4. ทำซ้ำข้อ 2 และ 3 สำหรับค่า $X_{L-1},...,X_1$

5. ลบค่า $X_0\overline{m}$ ออกจากค่าที่อยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์ (ลบแบบเลขส่วนเติมเต็มสอง)

ตัวอย่างการทำงานตามอัลกอริธึมนี้

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

$m = -0.390625$

$\overline{m} = 2 - |m|$ m เป็นเลขลบ

$= 2 - 0.390625$

$= 1.609375$

$\therefore \overline{m} = 1.100111$

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

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

โดยมีขั้นตอนการทำงาน ดังตารางต่อไปนี้

ตารางที่ 2.2 แสดงขั้นตอนการคูณเลขส่วนเติมเต็มสอง

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

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

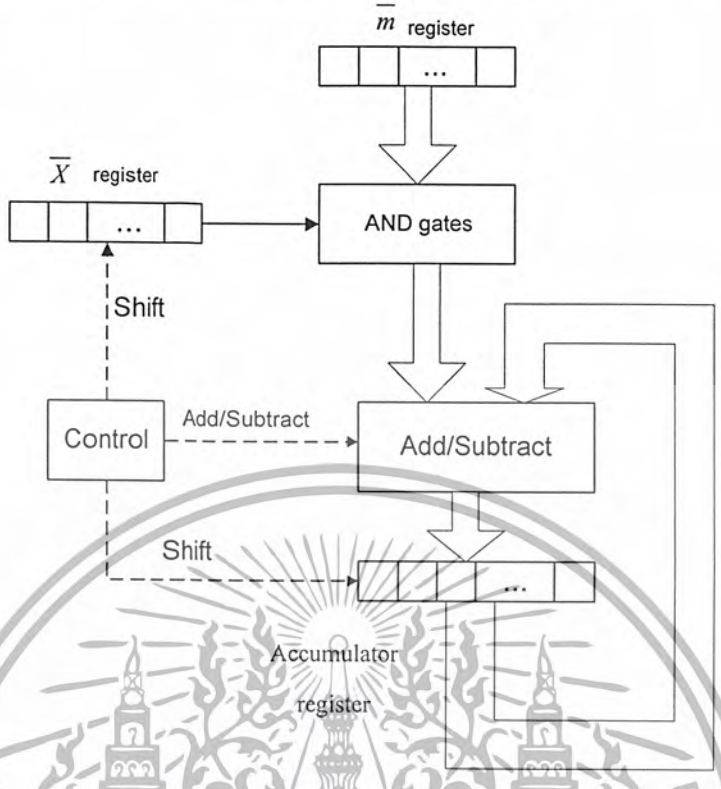
จะได้

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

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

$$= -0.3173828125$$

จากอัลกอริธึมดังกล่าวสามารถออกแบบการทำงานและสร้างวงจรแสดงได้ดังรูปที่ 2.13



รูปที่ 2.13 แสดงการคูณแบบเลขส่วนเติมเต็มสองโดยใช้เลขคณิตกระจาย

ที่ผ่านมาเป็นหลักการคูณแบบเลขส่วนเติมเต็มสอง ส่วนทฤษฎีเลขคณิตกระจายก็อาศัยหลักการดังกล่าวมาใช้ โดยทำการกระจายสมการที่อยู่ในรูปผลบวกของผลคูณให้แตกออกมาอยู่ในระดับบิท (Bit level) พิจารณาผลบวกของผลคูณต่อไปนี้

$$Y = \sum_{i=0}^{N} m_i X_i \tag{2.29}$$

โดย m_i เป็นค่าสัมประสิทธิ์ซึ่งทีค่าคงที่

X_i เป็นข้อมูลอินพุท

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

$$X_i = -X_{i0} + \sum_{j=1}^{L} X_{ij} 2^{-j} \tag{2.30}$$

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

X_{i0} = บิทแสดงเครื่องหมาย

X_{iL} = บิทที่มีนัยสำคัญต่ำสุด (LSB)

$L+1$ = จำนวนบิทที่แทนข้อมูลอินพุท

แทนค่า X_i ในสมการที่ (2.30) ลงในสมการที่ (2.29) จะได้

$$Y = \sum_{i=0}^{N} m_i \left[-X_{i0} + \sum_{j=1}^{L} X_{ij} 2^{-j} \right] \tag{2.31}$$

จัดเทอมของผลบวกใหม่จะได้

$$\begin{aligned} Y &= -X_{i0} \sum_{i=0}^{N} m_i + \sum_{j=1}^{L} X_{ij} 2^{-j} \sum_{i=0}^{N} m_i \\ &= -\sum_{i=0}^{N} X_{i0} m_i + \sum_{i=0}^{N} \sum_{j=1}^{L} X_{ij} m_i 2^{-j} \end{aligned} \tag{2.32}$$

จากนั้นทำการกระจายออกให้เป็นระดับบิท ได้ดังนี้

$$\begin{aligned} Y = & -(X_{00} m_0 + X_{10} m_1 + X_{20} m_2 + \ldots + X_{N0} m_N) \\ & + 2^{-1}(X_{01} m_0 + X_{11} m_1 + X_{21} m_2 + \ldots + X_{N1} m_N) \\ & + 2^{-2}(X_{02} m_0 + X_{12} m_1 + X_{22} m_2 + \ldots + X_{N2} m_N) \\ & + \ldots + 2^{-L}(X_{0L} m_0 + X_{1L} m_1 + X_{2L} m_2 + \ldots + X_{NL} m_N) \end{aligned} \tag{2.33}$$

สมการที่ (2.33) นี้ถูกกระจายออกให้อยู่ในรูปผลบวกของผลคูณระหว่างสัมประสิทธิ์กับข้อมูลอินพุทในระดับบิท ซึ่งเป็นนิยามของการคำนวณแบบเลขคณิตกระจาย และเมื่อเปรียบเทียบสมการที่ (2.33) กับสมการที่ (2.28) จะเห็นว่าการคำนวณหาค่า Y ก็ใช้เลขคณิตกระจายนั่นเอง เพียงแต่นำค่าผลคูณย่อย (Partial product) ที่คำนวณไว้ล่วงหน้าแล้วสำหรับแต่ละค่าที่สอดคล้องกับแต่ละบิทของข้อมูลอินพุทไปเก็บไว้ในตารางเปิดดู ซึ่งเป็นหน่วยความจำ EPROM และใช้ข้อมูลอินพุทเป็นแอดเดรสของหน่วยความจำ เพื่อนำค่าในตารางเปิดดูมาผ่านขั้นตอนการคำนวณตามบูทอัลกอริธึม ซึ่งค่าในตารางเปิดดู สามารถแสดงได้ดังนี้

ตารางที่ 2.3 แสดงค่าผลคูณย่อยที่เก็บไว้ในตารางเปิดดูที่กำหนดโดยข้อมูลอินพุท

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

2.6.3 การนำโครงสร้างเลขคณิตกระจายมาใช้กับวงจรกรองสัญญาณเชิงเลข

ในการนำโครงสร้างเลขคณิตกระจายมาใช้ในการออกแบบสำหรับวงจรกรองสัญญาณเชิงเลขนั้น เพื่อความสะดวกจะใช้สมการผลต่างสืบเนื่องอันดับที่สอง (Second order difference equation) มาพิจารณาเพื่อเป็นพื้นฐานในการสร้างวงจรกรองที่มีอันดับที่สูงขึ้นต่อไป

พิจารณาสมการผลต่างสืบเนื่องอันดับสองดังนี้

$$Y(n) = a_0X(n) + a_1X(n-1) + a_2X(n-2) - b_1Y(n-2) - b_2Y(n-2) \qquad (2.34)$$

แทนลำดับสัญญาณอินพุท X(n) และลำดับสัญญาณเอาต์พุท Y(n) ด้วยเลขส่วนเติมเต็มสองได้ดังนี้

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

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

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

X(n) และ Y(n) สามารถแสดงได้โดย

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

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

นำค่า $X(n)$ และ $Y(n)$ แทนลงในสมการที่ (2.36) ได้

$$\begin{aligned} Y(n) = \sum_{i=1}^{L} 2^{-1}[&a_0X_i(n) + a_1X_i(n-1) + a_2X_i(n-2) - b_1Y_i(n-1) \\ &- b_2Y_i(n-2)] - [a_0X_0(n) + a_1X_0(n-1) + a_2X_0(n-2) \\ &- b_1Y_0(n-1) - b_2Y_0(n-2)] \end{aligned} \tag{2.35}$$

คูณ 2^{-1} ทั้ง 2 ข้างได้

$$\begin{aligned} 2^{-1}Y(n) = \sum_{i=1}^{L} 2^{-1}[&2^{-1}a_0X_i(n) + 2^{-1}a_1X_i(n-1) + 2^{-1}a_2X_i(n-2) - 2^{-1}b_1Y_i(n-1) \\ &- 2^{-1}b_2Y_i(n-2)] - [2^{-1}a_0X_0(n) + a_1X_0(n-1) + 2^{-1}a_2X_0(n-2) \\ &- 2^{-1}b_1Y_0(n-1) - b_2Y_0(n-2)] \end{aligned} \tag{2.36}$$

พิจารณาสมการที่ (2.36) และสมการที่ (2.27) จะได้

$$\begin{aligned} 2^{-1}\overline{Y}(n) = \sum_{i=1}^{L} 2^{-1}[&2^{-1}\bar{a}_0X_i(n) + 2^{-1}\bar{a}_1X_i(n-1) + 2^{-1}\bar{a}_2X_i(n-2) - 2^{-1}\bar{b}_1Y_i(n-1) \\ &- 2^{-1}\bar{b}_2Y_i(n-2)] - [2^{-1}\bar{a}_0X_0(n) + \bar{a}_1X_0(n-1) + 2^{-1}\bar{a}_2X_0(n-2) \\ &- 2^{-1}\bar{b}_1Y_0(n-1) - \bar{b}_2Y_0(n-2)] \end{aligned} \tag{2.37}$$

เพราะฉะนั้น

$$\overline{Y}(n) = \sum_{i=1}^{L} 2_2^{-i}F_i - F_o \tag{2.38}$$

โดย

$$F_i = \bar{a}_0X_i(n) + \bar{a}_1X_i(n-1) + \bar{a}_2X_i(n-2) - \bar{b}_1Y_i(n-1) - \bar{b}_2Y_i(n-2) \tag{2.39}$$

$$F_o = \bar{a}_0X_0(n) + \bar{a}_1X_0(n-1) + \bar{a}_2X_0(n-2) - \bar{b}_1Y_0(n-1) - \bar{b}_2Y_o(n-2) \tag{2.40}$$

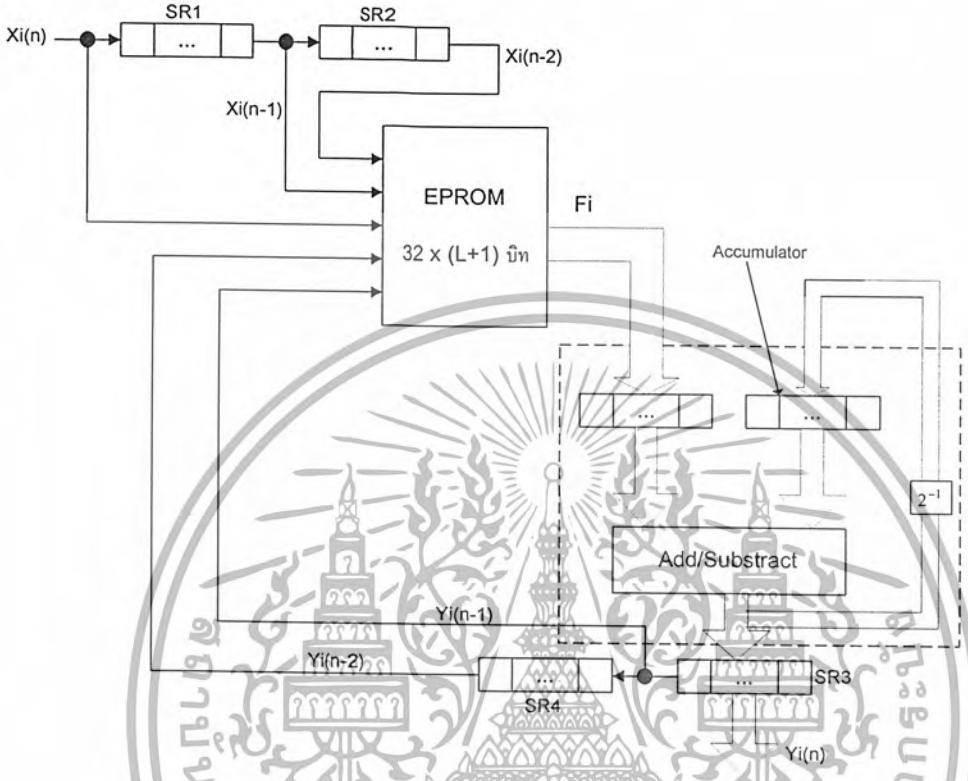
ส่วนเติมเต็มสองของ Y(n) สามารถหาได้โดยใช้อัลกอริธึมดังนี้

1. เคลียร์ค่าของข้อมูลในแอคคิวมูเลเตอร์รีจิสเตอร์
2. คำนวณค่า F_i สำหรับ $i = L$
3. บวกค่า F_i กับค่าที่บรรจุอยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์
4. เลื่อนค่าที่บรรจุอยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์ไปทางขวา1บิต (เลื่อนข้อมูลแบบเลขส่วนเติมเต็มสอง)
5. ทำซ้ำข้อ2 ถึง 4 สำหรับ i = L-1, L-2,...,1

6. คำนวณค่า F_o

7. ลบค่า F_o ออกจากค่าที่บรรจุอยู่ในแอคคิวมูเลเตอร์รีจิสเตอร์ (ลบแบบเลขส่วนเติมเต็มสอง)

โดยอัลกอริธึมที่กล่าวมาสามารถออกแบบการทำงานและสร้างวงจรดังแสดงได้ดังรูป



รูปที่ 2.14 แสดงโครงสร้างวงจรกรองสัญญาณป้อนกลับเชิงเลขอันดับที่สอง

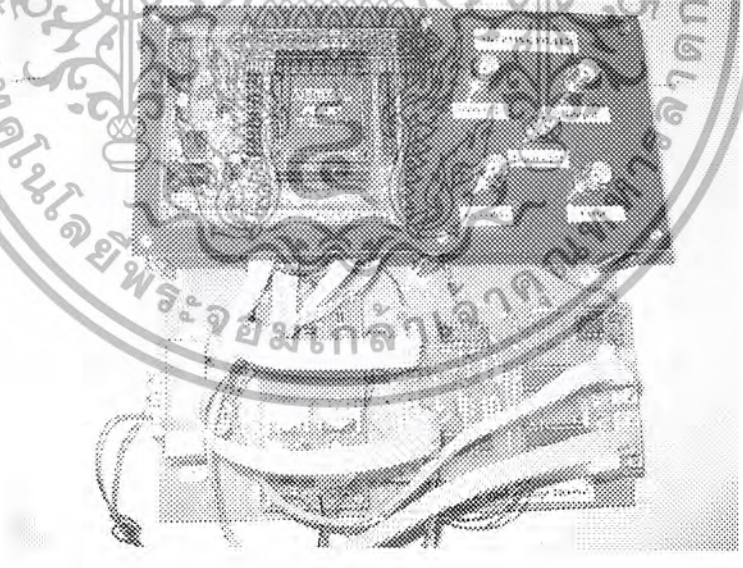
จากรูปที่ 2.14 ส่วนที่อยู่ในเส้นประนิยมเรียกกันว่าวงจรสเกลลิ่งแอคคิวมูเลเตอร์ (Scaling Accumulator) โดยค่าที่อยู่ในแอคคิวมูเลเตอร์ ก่อนที่จะส่งไปบวกกับค่าผลลัพธ์จาก EPROM (หรือ partial sum ตัวต่อไป) จะต้องทำการเลื่อนข้อมูลไปทางขวา 1 บิทก่อน ดังที่กล่าวมาแล้ว ซึ่งการเลื่อนข้อมูลไปทางขวา 1 บิทนี้ เขียนแทนด้วยการคูณด้วย 2^{-1} และผลจากสมการที่ (2.28) นำมาสร้างเป็นตารางเปิดดูบรรจุไว้ใน EPROM ค่าในตารางเปิดดูเป็นค่าของ F_i ซึ่งเกิดจากตัวแปรที่เป็นลำดับสัญญาณอินพุท 5 ตัว ดังนั้นค่าของ F_i จะมีค่า $2^5 = 32$ ค่า โดยขนาดของ EPROM จะมีขนาด $32 \times (L+1)$ บิท

บทที่ 3
การออกแบบและการสร้าง

3.1 การออกแบบวงจรเชิงเลขด้วยอุปกรณ์ FPGA

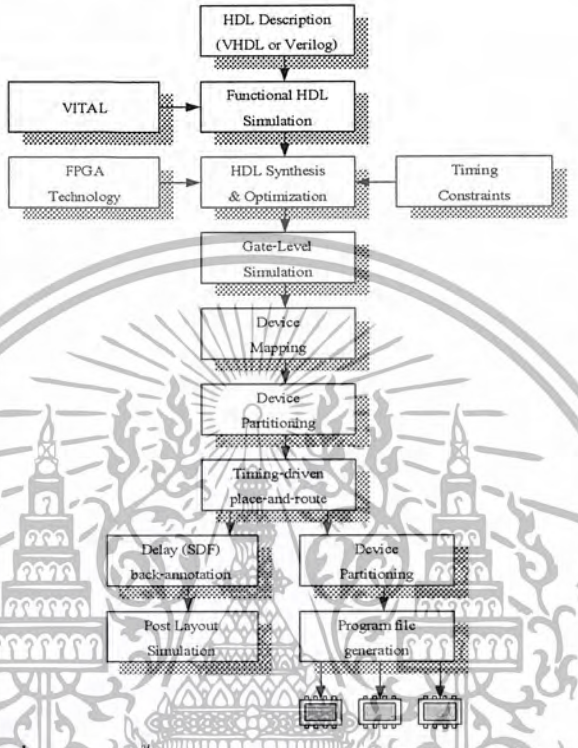
อุปกรณ์ FPGA (Field Programmable Gate Array) เป็นอุปกรณ์ที่ใช้ในการโปรแกรมวงจรที่ได้ออกแบบลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชั่นการทำงานตามที่ออกแบบไว้ ในการทำ FPGA ซึ่งเป็นวิธีการออกแบบ IC (Integrated Circuit) แบบ Semicustom อีกวิธีหนึ่ง เมื่อเทียบกับการทำ ASICs (Application Specific Integrated Circuits) แล้วนั้นก็มีทั้งข้อดีและข้อเสีย คือ การทำ FPGA จะมีข้อจำกัดในด้านขนาดของวงจรเพราะภายในอุปกรณ์ FPGA จะมีจำนวนเกท (gate) ให้ใช้จำนวนจำกัด และการทำ FPGA ก็เหมาะสำหรับการทำผลิตภัณฑ์ต้นแบบหรือเพื่อผลิตในปริมาณต่ำ ส่วนข้อดีของการทำ FPGA ก็คือระยะเวลาที่ใช้ในการทำตั้งแต่เขียนรหัส (code) อธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลด (download) นั้นน้อยกว่าการทำ ASIC มากและการตรวจสอบหรือแก้ไขการออกแบบก็ทำได้สะดวก

การทำ FPGA ในปัจจุบันมีประสิทธิภาพและความสะดวกมากขึ้น ทั้งนี้ก็เนื่องจากทางบริษัทผู้ผลิตอุปกรณ์ FPGA ได้เพิ่มความสามารถของอุปกรณ์ FPGA โดยเพิ่มจำนวนองค์ประกอบภายใน หรือปรับปรุงโครงสร้างสถาปัตยกรรมภายในและยังได้เพิ่มประสิทธิภาพของซอฟต์แวร์ที่ใช้ทำ PPR (Partitioning, Placement and Routing) สำหรับอุปกรณ์นั้นๆด้วย ลักษณะของตัวFPGAและการนำไปใช้งานแสดงดังในรูปที่ 3.1



รูปที่ 3.1 แสดงลักษณะของตัว FPGA และการนำไปใช้งาน

สำหรับตัวอุปกรณ์ FPGA นั้นก็มีโครงสร้างพื้นฐาน เทคโนโลยีที่ใช้สร้าง ตลอดจนเทคนิควิธีการโปรแกรมที่แตกต่างกันสำหรับผู้ผลิตแต่ละราย นอกจากนั้นอุปกรณ์ FPGA ของแต่ละผู้ผลิตก็มีโครงสร้างและความสามารถที่แตกต่างกันบางส่วน ในการใช้งานนั้นอุปกรณ์ FPGA สามารถนำไปประยุกต์ใช้งานได้ เช่น การประมวลผลสัญญาณเชิงเลข (DSP : Digital Signal Processing) การออกแบบไมโครคอนโทรลเลอร์ เป็นต้น โดยมีขั้นตอนในการออกแบบ ดังแสดงในรูปที่ 3.2



รูปที่ 3.2 แสดงขั้นตอนการออกแบบโดยใช้อุปกรณ์ FPGA

3.2 การออกแบบโดยใช้ภาษาอธิบายการทำงานของฮาร์ดแวร์

ในการออกแบบวงจรเชิงเลขนั้นทำได้โดยการวาดวงจร หรือใช้ภาษาอธิบายฮาร์ดแวร์ ในขั้นตอนนี้เป็นขั้นตอนที่ไม่แตกต่างกันระหว่างการออกแบบด้วย FPGA และ ASIC ในกรณีที่ใช้ภาษาอธิบายฮาร์ดแวร์ แต่ในกรณีที่ออกแบบโดยวิธีการวาดวงจรจะแตกต่างกันโดยที่การทำวิธีนี้จะต้องคำนึงถึงเทคโนโลยีที่จะใช้ซึ่งแต่ละเทคโนโลยีก็มีความแตกต่างกันไป จะเห็นได้ว่าการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ทำได้สะดวกกว่า เพราะการทำด้วยวิธีนี้ไม่ต้องคำนึงถึงเทคโนโลยีที่จะใช้ (Technology independence) และที่สำคัญการออกแบบด้วยวิธีนี้สามารถที่จะแก้ไขโมเดล (Model) หรือเปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่าเพราะไม่ต้องวาดวงจรใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยี

ในการเขียนโค้ด สิ่งที่ต้องคำนึงถึงคือเขียนอย่างไรจึงจะสามารถสังเคราะห์เป็นวงจรได้และให้คุณสมบัติของวงจรตามที่กำหนด เพราะลักษณะการเขียนโค้ดจะมีผลโดยตรงกับวงจรที่ได้ เนื่องจากในการ

สังเคราะห์วงจรนั้นซอร์ฟแวร์สังเคราะห์วงจร (Synthesis Tools) จะทำการสังเคราะห์ตามโค้ดที่เขียน ถ้าอธิบายการทำงานของวงจรเดียวกันแต่เขียนโค้ดในลักษณะที่ต่างกันเมื่อสังเคราะห์แล้วจะได้วงจรที่ต่างกัน และจากวงจรที่ต่างกัน เมื่อนำไปทำต้นแบบด้วย FPGA หรือการทำ ASIC แล้วจะได้ไอซีที่มีคุณสมบัติต่างกัน ทั้งในด้านของขนาดหรือความเร็ว (area and time) ส่วนการเขียนโค้ดลักษณะใดเพื่อให้ได้ผลลัพธ์ที่ดีที่สุดนั้นก็ขึ้นอยู่กับประสบการณ์ในการออกแบบ

3.3 การจำลองการทำงานของวงจร (Simulation)

ขั้นตอนนี้เป็นขั้นตอนที่สำคัญเพราะเป็นขั้นตอนที่ใช้ตรวจสอบฟังก์ชั่นการทำงานของวงจรว่าถูกต้องหรือไม่ มีข้อผิดพลาดตรงไหน เพื่อที่จะได้ทำการแก้ไขให้ถูกต้อง ในขั้นตอนนี้จะใช้ซอร์ฟแวร์สำหรับทำการจำลองการทำงานของวงจร เช่น V-System และ ModelSim ของบริษัท Model Technology

3.4 การสังเคราะห์วงจร

ในขั้นตอนนี้จะใช้ซอร์ฟแวร์สังเคราะห์วงจร (Synthesis tools) ทำการสังเคราะห์โค้ดเพื่อให้ได้เป็นวงจรขึ้นมา แต่ต้องตรวจสอบด้วยว่าซอร์ฟแวร์นั้นๆสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการใช้หรือไม่ โดย FPGA ที่นิยมใช้งานเช่นของบริษัท Xilinx ตระกูล XC4000 และบริษัท Altera ตระกูล FLEX 10 K ซอร์ฟแวร์สังเคราะห์วงจรที่นิยมใช้เช่นโปรแกรม Leonardo Spectrum ของบริษัท Exemplar Logic ซึ่งในขั้นตอนนี้ซอร์ฟแวร์สังเคราะห์วงจรจะแปลงโค้ดและทำการออปติไมซ์ (Optimization) เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ นอกจากนี้ยังสามารถกำหนดข้อบังคับสำหรับวงจรได้เช่น ข้อบังคับในเรื่องของเวลา (time constraints) หรือข้อบังคับในเรื่องของพื้นที่ ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอนออปติไมซ์เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนด ส่วนสำคัญในการออปติไมซ์คือการเทียบ (mapping) วงจรให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้างสถาปัตยกรรมภายในอุปกรณ์ FPGA ในกรณีของ Xilinx ตระกูล XC4000 และ Altera ตระกูล FLEX 10 K จะเทียบโดยใช้วิธี LUT (Look Up Table) เมื่อทำการสังเคราะห์วงจรเสร็จแล้วซอร์ฟแวร์สังเคราะห์วงจรก็จะมีการรายงานผลว่าวงจรที่ออกแบบไปนั้นเป็นอย่างไร เช่น มีความหน่วง (delay) เท่าไร ใช้ทรัพยากรต่างๆใน FPGA อะไรบ้าง เป็นต้น

3.5 การแบ่งวงจร (Partitioning)

ขั้นตอนนี้เป็นการแบ่งวงจรที่ได้จากการสังเคราะห์ให้เป็นส่วนย่อยๆ สำหรับลงใน CLBs, IOBs หรือองค์ประกอบอื่นๆ ภายในอุปกรณ์ FPGA สำหรับเกณฑ์ที่ใช้ในการแบ่งคือให้แต่ละส่วนที่จะแยกออกจากกันมีจำนวนสัญญาณที่เชื่อมต่อระหว่างกันน้อยที่สุดเท่าที่จะทำได้เพื่อช่วยลดความหนาแน่นในตอนทำการเชื่อมต่อสัญญาณ (routing) ในขั้นตอนนี้จะใช้ซอร์ฟแวร์ทำ โดยซอร์ฟแวร์จะเทียบส่วนประกอบของวงจรเช่น เกท (gate), ฟลิปฟลอป (flipflop) ลงในทรัพยากรต่างๆ ที่มีอยู่ภายในอุปกรณ์ FPGA (CLBs, IOBs, BUFT

และ edge decoder) หลังจากทำขั้นตอนนี้เสร็จแล้วสามารถที่จะทราบว่าวงจรใช้จำนวนทรัพยากรภายในอุปกรณ์ FPGA ไปเท่าไร ส่วนซอร์ฟแวร์ที่ใช้ในขั้นตอนนี้ขึ้นอยู่กับตัว FPGA ที่ใช้งานเช่น FPGA ของบริษัท Xilinx จะใช้ Xilinx Foundation Series 2.1i ซึ่งซอร์ฟแวร์ตัวนี้จะรวมเอาซอร์ฟแวร์ย่อยอื่นๆอีก เพื่อให้การทำ PPR (Partitioning, Placement and Routing) เป็นไปอย่างต่อเนื่อง ส่วน FPGA ของบริษัท Altera จะใช้ Altera MAX+II

3.6 การวางอุปกรณ์ (Placement)

ขั้นตอนนี้เป็นการเลือกทำเลที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (partitioning) มาแล้วว่าควรจะอยู่ในตำแหน่งใดในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด เช่นวงจรส่วนไหนควรอยู่ใกล้กันเพื่อจะได้ค้นหาเส้นทาง (route) ได้ง่ายหรือช่วยลดความหน่วง จะเห็นได้ว่าตำแหน่งภายในอุปกรณ์ FPGA นั้นมีความสำคัญเพราะถ้าจัดวางวงจรลงในตำแหน่งที่ไม่เหมาะสมแล้วจะทำให้ความหน่วงเพิ่มขึ้นหรือตัว Router ทำการค้นหาเส้นทางสัญญาณได้ไม่หมด

3.7 การเชื่อมต่อสัญญาณ (Routing)

ในขั้นตอนนี้เป็นการเชื่อมต่อสัญญาณระหว่างองค์ประกอบต่างๆ ภายในอุปกรณ์ FPGA เช่นระหว่าง CLBs หรือระหว่าง CLBs กับ IOBs ขั้นตอนนี้จะทำต่อเนื่องจากการวางอุปกรณ์ ในกรณีที่ทำการวางอุปกรณ์ไว้ไม่ดีซอร์ฟแวร์ก็จะทำการเชื่อมต่อสัญญาณได้ไม่หมดหรือเกิดความหน่วงเกินค่าที่กำหนดในข้อบังคับ โดยสามารถทำขั้นตอนนี้ได้โดยใช้ซอร์ฟแวร์ เช่นกัน หรือจะทำการเชื่อมต่อสัญญาณด้วยตัวเอง (manual layout) ก็ได้แต่ทางที่ดีควรใช้ซอร์ฟแวร์ทำดีกว่าโดยให้ทำการค้นหาเส้นทางหลายๆครั้งเพื่อหาครั้งที่ดีที่สุด นอกจากนั้นการกำหนดข้อบังคับทางเวลา (time constraints) จะช่วยให้ผลที่ได้จากการทำการเชื่อมต่อสัญญาณดีขึ้นได้

3.8 การโปรแกรมอุปกรณ์ FPGA (Configuration)

หลังจากที่วงจรผ่านขั้นตอนต่างๆจนกระทั่งผ่านการทำ PPR (Partitioning, Placement and Routing) แล้วนั้น ถึงตอนนี้ก็สามารถที่จะดาวน์โหลด (download) ลงในอุปกรณ์ FPGA ได้แล้ว ในการดาวน์โหลดนี้ก่อนอื่นต้องแปลงแบบวงจรรวมที่ได้ให้เป็นข้อมูลวงจร (Configuration data) ซึ่งอยู่ในรูปของบิทสตรีม (Bit-stream) ก่อนแล้วจึงดาวน์โหลดลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชั่นการทำงานตามวงจรที่ออกแบบไว้

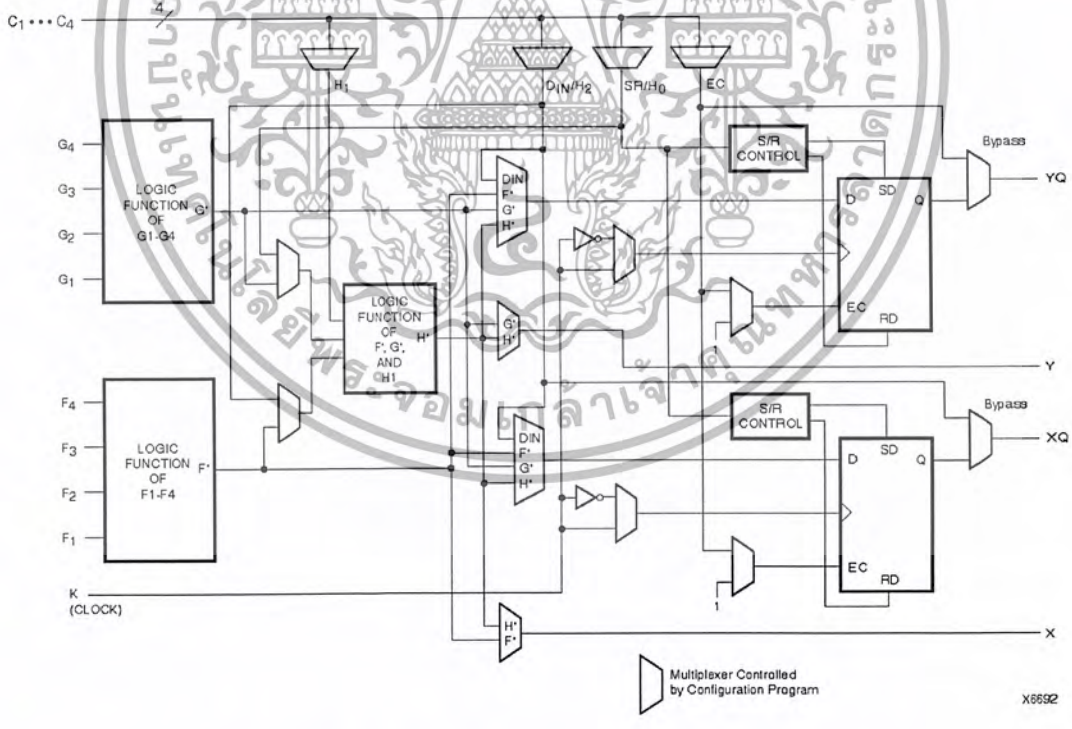
จากที่อธิบายมาทั้งหมดจะเห็นได้ว่าการออกแบบเพื่อทำ FPGA นั้น ทำได้สะดวกกว่าการทำ ASIC มากเพราะใช้เวลาน้อยกว่ามาก ส่วนสำคัญที่ใช้ในการทำ FPGA คือซอร์ฟแวร์ที่ใช้ตั้งแต่การเขียนโค้ดอธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลดลงในอุปกรณ์ FPGA ซึ่งซอร์ฟแวร์ที่ใช้ต้องเป็นซอร์ฟแวร์ที่ใช้ทำงานต่อเนื่องกัน

3.9 สถาปัตยกรรมภายในของ FPGA

สถาปัตยกรรมภายในของ Xilinx FPGA จะมีลักษณะเป็นตารางของลอจิกบล็อก (Logic Block) และล้อมรอบไปด้วยบล็อกการเชื่อมต่อของไอโอ (I/O Interface Block) การเชื่อมต่อระหว่างซีแอลบี (CLB : Configuration Logic Block) และไอโอบี (IOB : Input Output Block) ทำได้โดยผ่านช่องว่างที่พาดผ่านระหว่างแถว (Row) และคอลัมน์ (Column) หน้าที่ของซีแอลบีและไอโอบีแต่ละตัว การเชื่อมต่อภายใน (Interconnection) จะถูกกำหนดไว้ในโปรแกรมคอนฟิกกูเรชั่น (Configuration program) โดยแสดงรายละเอียดของทั้ง 3 ส่วนประกอบใหญ่ๆ ได้ดังนี้

CLB (Configuration Logic Block)

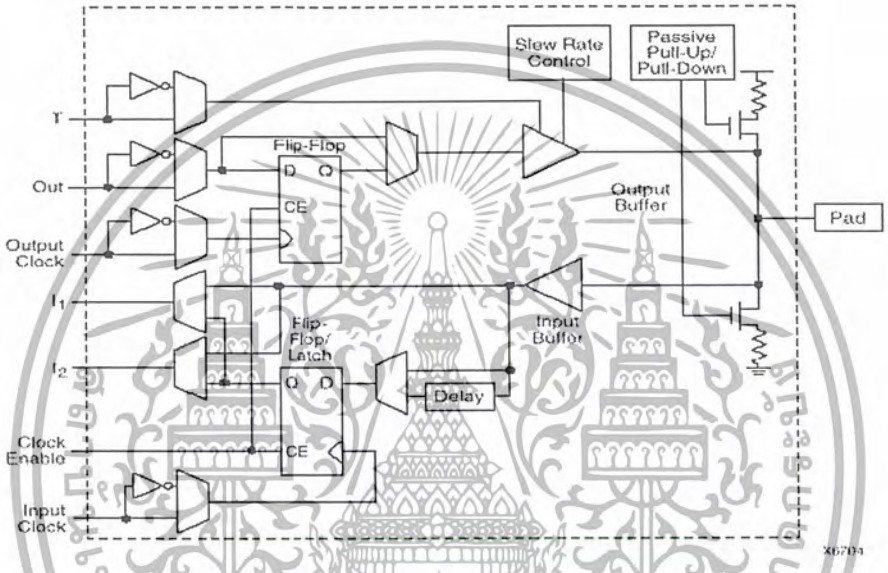
ภายในFPGAจะเป็นเมทริกซ์ของซีแอลบีซึ่งซีแอลบีแต่ละตัวจะประกอบด้วยหน่วยของคอมบิเนชั่นลอจิกที่สามารถโปรแกรมได้ (Programmable Combination logic) และส่วนของรีจิสเตอร์เก็บข้อมูล (Storage register) ส่วนของวงจรคอมบิเนชั่นลอจิกสามารถใช้สร้างวงจรทางด้านฟังก์ชั่น บูลีนของอินพุท ส่วนรีจิสเตอร์รับค่าจากส่วนคอมบิเนชั่นหรือโดยตรงจากเอาท์พุทของซีแอลบี สามารถขับวงจรคอมบิเนชั่นลอจิกโดยตรงผ่านเส้นทางเดินย้อนกลับ (Feedback path)



รูปที่ 3.3 แสดงวงจรภายในของซีแอลบีของ FPGA ตระกูล XC4000

IOB (Input Output Block)

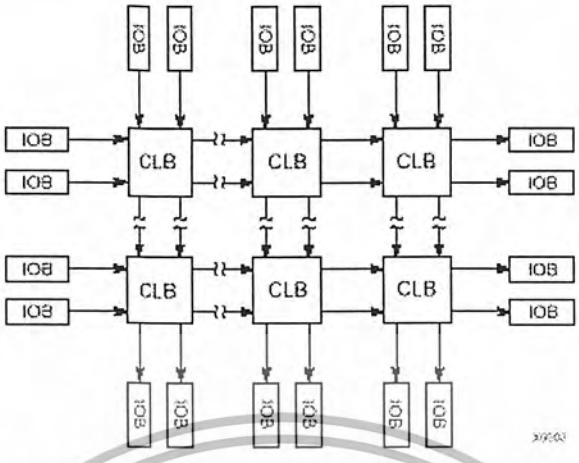
เป็นส่วนติดต่อกับวงจรภายนอกของ FPGA สร้างมาจากส่วนของอุปกรณ์อินพุท/เอาท์พุทที่สามารถโปรแกรมได้ (Programmable Input/Output device) แต่ละตัวสามารถโปรแกรมได้อย่างอิสระโดยจะให้เป็นอินพุท/เอาท์พุทแบบ 3 สถานะ หรือไอโอแบบสองทิศทางก็ได้ โดยอินพุทสามารถโปรแกรมให้รู้จักทั้งระดับสัญญาณทีทีแอล และซีมอสเทรดโฮลของไอโอบีแต่ละตัวมีฟลิปฟลอปสามารถใช้เป็นบัฟเฟอร์สำหรับอินพุทและเอาท์พุท



รูปที่ 3.4 แสดงวงจรภายในของไอโอบีของ FPGA ตระกูล XC4000

Interconnection

ความยืดหยุ่นของการใช้ FPGA มาทำเป็นอุปกรณ์ขึ้นอยู่กับการโปรแกรมทรัพยากรต่างๆที่อยู่ภายในเข้าด้วยกัน การที่จะควบคุมการเชื่อมต่อระหว่างจุดสองจุดภายในชิปจะประกอบไปด้วยเน็ตเวิร์ก 2 ทิศทาง คือ ทางแถวและคอลัมน์ ซึ่งวางอยู่ระหว่าง CLB Programmable switch จะทำการเชื่อมต่ออินพุทและเอาท์พุทของไอโอบีและซีแอลบีที่จุดต่อร่วมระหว่างแถวกับคอลัมน์และสามารถสลับสัญญาณจากเส้นทางไปยังส่วนต่างๆได้



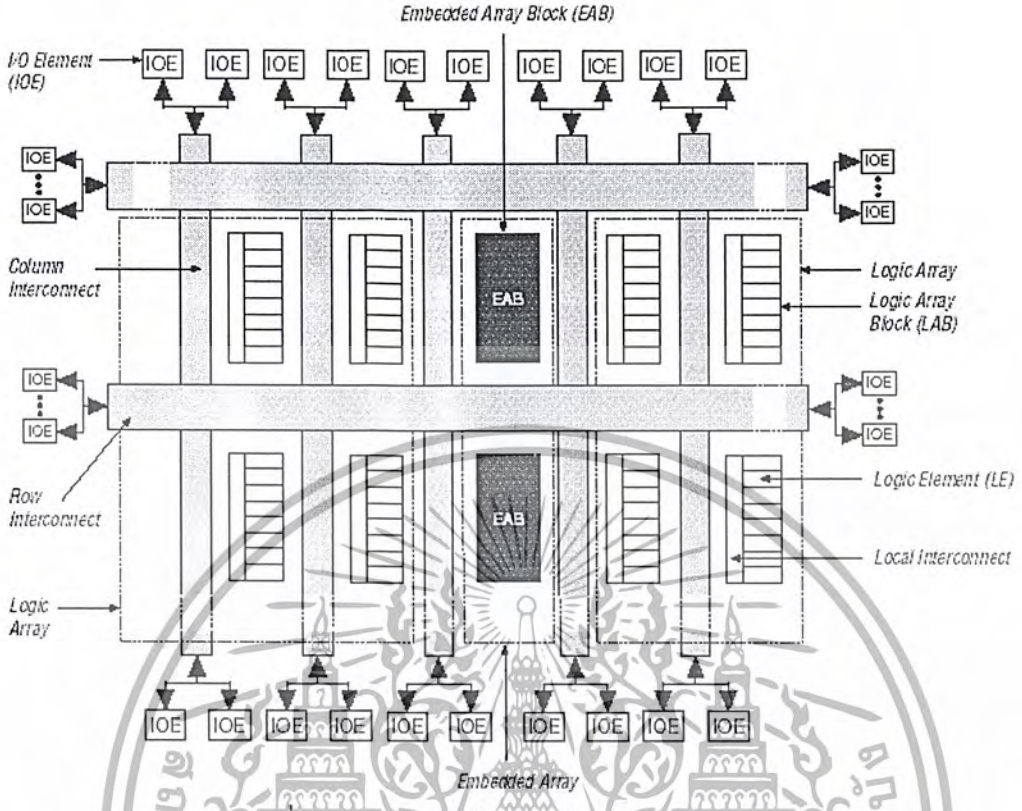
รูปที่ 3.5 แสดง Interconnection ระหว่างไอโอบีกับซีแอลบีของ FPGA ตระกูล XC4000

3.10 โครงสร้างของ FPGA ตระกูล FLEX 10 K

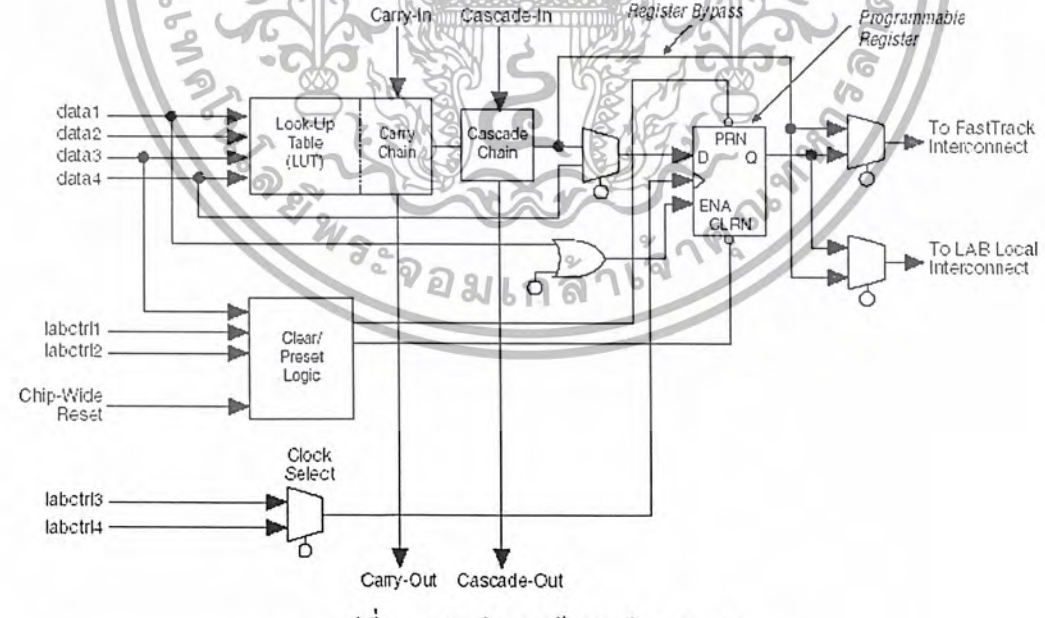
FPGA ของบริษัท Altera ตระกูล FLEX 10 K เป็นอุปกรณ์ที่มีความหนาแน่นเกทประมาณตั้งแต่ 10,000 – 250,000 เกท โดยการจัดโครงสร้าง (Configuration) จะใช้วิธีโหลดโครงสร้างเข้าไปใน SRAM ภายใน ซึ่งหมายความว่าถ้าไม่ได้มีการจ่ายไฟเลี้ยงให้ โครงสร้างที่จัดเอาไว้ก็จะหายไป FPGA ประเภทนี้จะสามารถโปรแกรมซ้ำได้ไม่จำกัดจำนวนครั้ง และการทำงานตามลอจิกฟังก์ชันจะใช้วิธีการเปิดตารางดู (Look Up table : LUT) โดยโครงสร้างของ FPGA ตระกูล FLEX 10 K แสดงดังรูปที่ 3.6 โดยในโครงสร้างของ FPGA ตระกูล FLEX 10 K สามารถที่จะแบ่งเป็นส่วนต่างๆ ได้ดังนี้

Logic Element (LE)

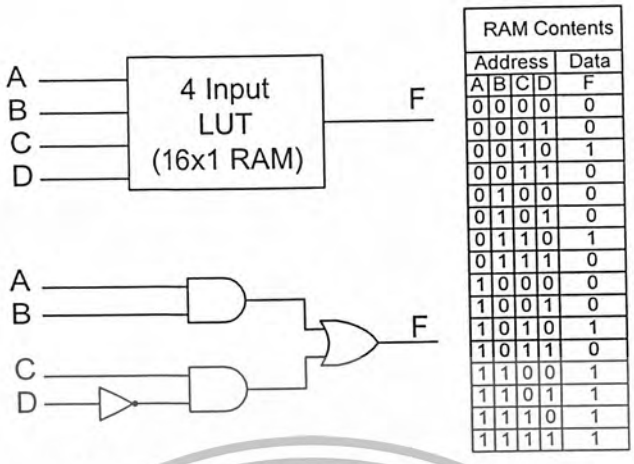
ในรูปที่ 3.7 แสดงโครงสร้างภายในของ LE โดยการกระทำทางบูลีนของลอจิกเกทจะสร้างด้วยวิธีการ LUT โดย LUT คือ 1x16 SRAM ซึ่ง LUT เพียงตัวเดียวสามารถนำมาทำโครงข่ายของลอจิกเกทที่มี 4 อินพุท และ 1 เอาท์พุท โดยโครงข่ายของลอจิกเกทจะถูกแปลงไปเป็นตารางค่าความจริง (Truth Table) ดังแสดงในรูปที่ 3.8



รูปที่ 3.6 แสดงโครงสร้างของ FPGA ตระกูล FLEX 10K



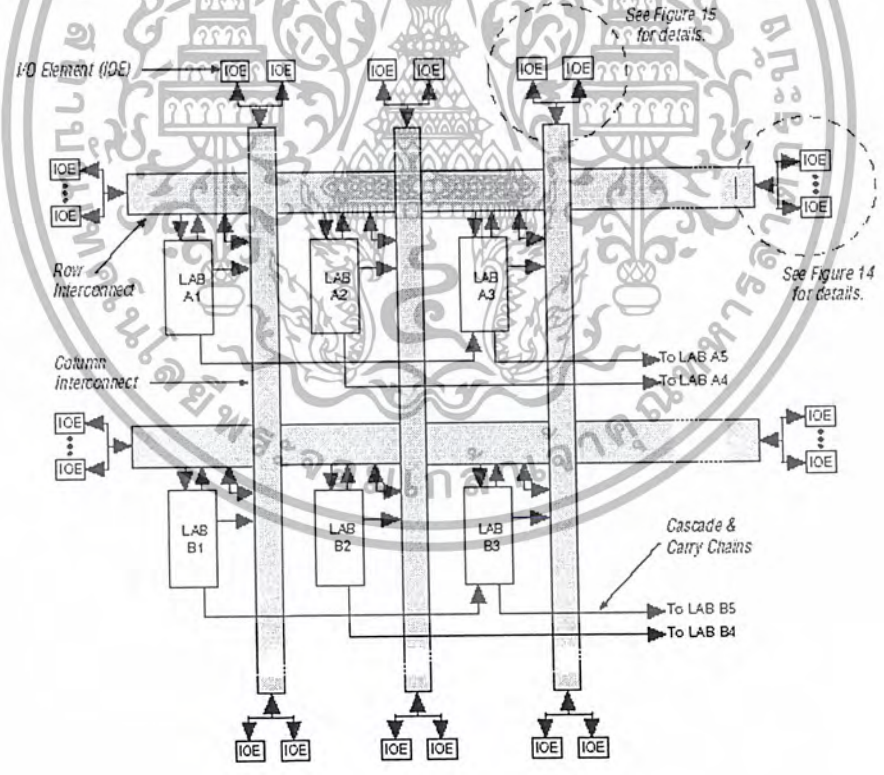
รูปที่ 3.7 แสดงโครงสร้างภายในของ LE



RAM Contents				
Address				Data
A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

รูปที่ 3.8 แสดงการใช้งาน LUT เป็นโครงข่ายของลอจิก

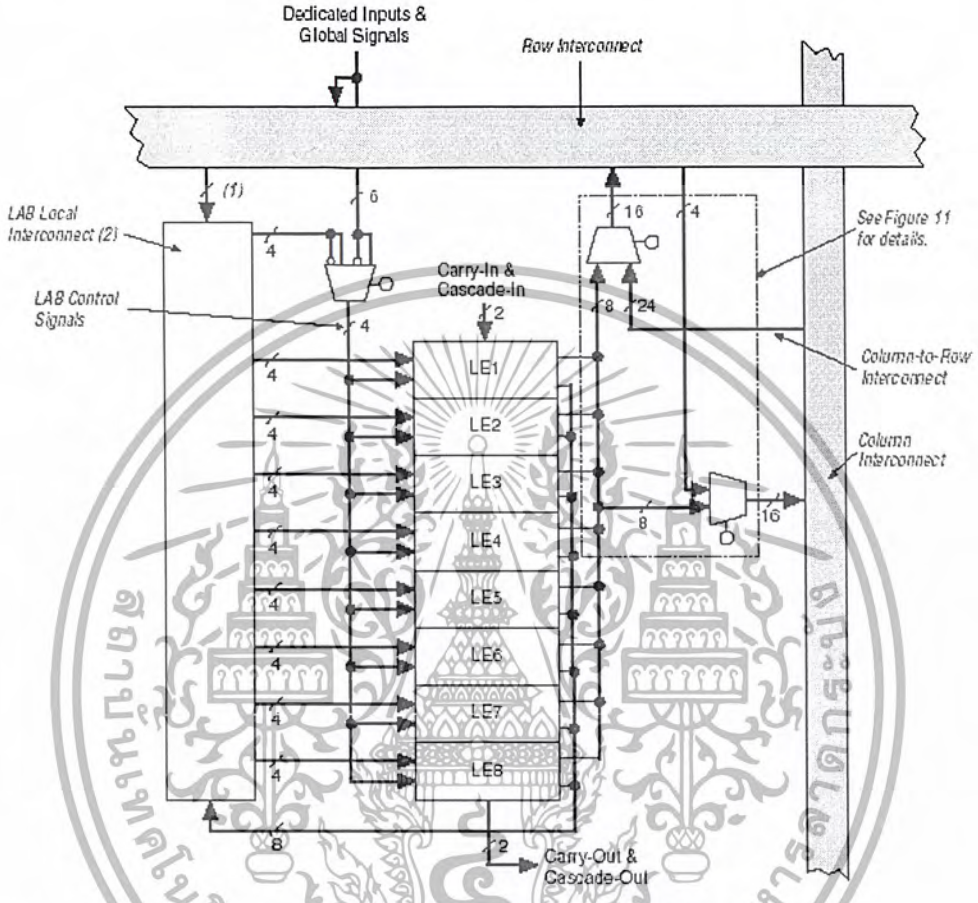
ถ้าโครงข่ายของลอจิกเกทมีความซับซ้อนขึ้นจะต้องใช้ LUT ของแต่ละ LE เป็นจำนวนหลายตัว โดยเอาท์พุทของ LUT จะส่งต่อไปยังฟลิปฟลอปและต่อไปยังโครงข่ายการเชื่อมต่อ (Interconnection Network) ดังแสดงในรูปที่ 3.9



รูปที่ 3.9 แสดงโครงข่ายของการเชื่อมต่อ

Logic Array Block (LAB)

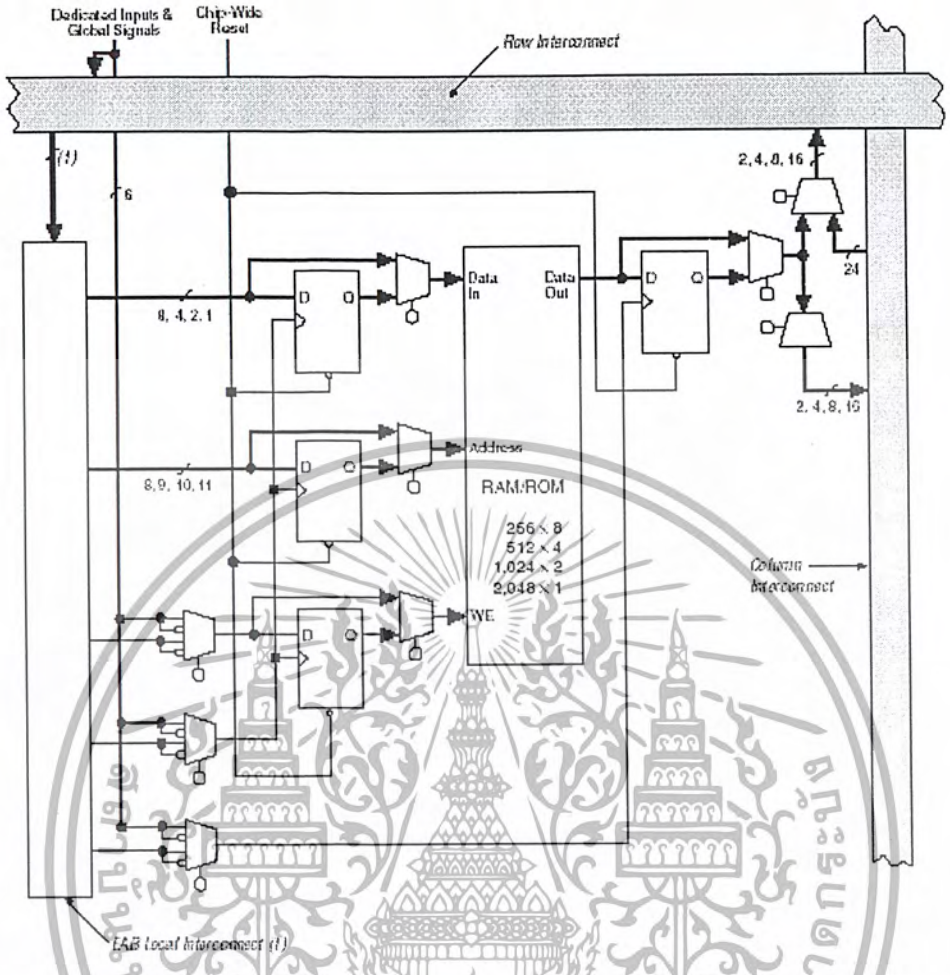
LAB 1 ตัว จะประกอบไปด้วย 8 LE ดังแสดงในรูปที่ 3.10



รูปที่ 3.10 แสดงโครงสร้างภายในของ LAB

Embedded Array Block (EAB)

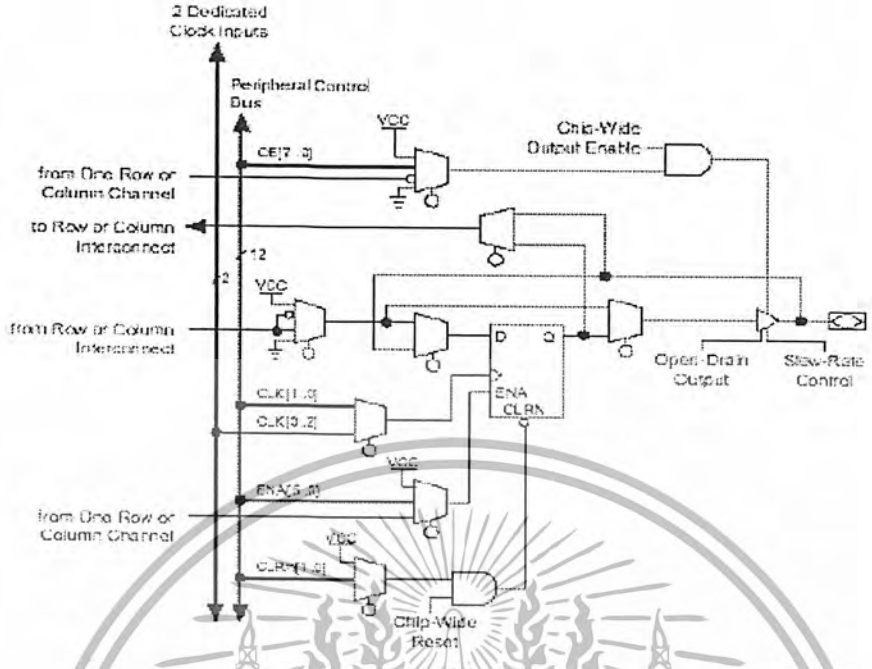
สถาปัตยกรรมโดยทั่วไปของ FLEX 10 K จะมีลักษณะของ LAB ที่มีการจัดเรียงแบบเมตริกซ์ และ EAB ซึ่งมีการเชื่อมต่อผ่านทางแถวและคอลัมน์ โดยในแต่ละแถวจะมี 1 EAB ซึ่ง 1 EAB จะมีขนาด 2048 บิท และสามารถกำหนดความกว้าง (Width) ความลึก (Depth) ของ EAB ได้โดยไม่ส่งผลต่อความเร็ว



รูปที่ 3.11 แสดงโครงสร้างภายใน EAB

Input Out Element (IOE)

IOE จะถูกต่ออยู่กับขา I/O โดยจะประกอบด้วยส่วนของวงจรที่เป็น Tri State และส่วนที่เป็น ฟลิปฟลอป ซึ่งเป็น option ดังแสดงในรูปที่ 3.12



รูปที่ 3.12 แสดงโครงสร้างภายในของ IOE

3.11 ภาษา VHDL และ ส่วนประกอบต่างๆของภาษา

วิวัฒนาการของภาษา VHDL นั้นเริ่มต้นประมาณปี ค.ศ. 1981 โดยที่กระทรวงคลาโหมสหรัฐอเมริกา หรือ DOD (Department of Defense) ได้ทำการพัฒนาโครงการที่มีชื่อว่า VHSIC ซึ่งเป็นการพัฒนาโปรแกรม ซึ่งจัดเป็นภาษาระดับสูงเช่นเดียวกับภาษา C หรือ Pascal แต่สามารถบรรยายพฤติกรรมการทำงานของวงจรเชิงเลข หรือโครงสร้างของวงจรได้ ทั้งนี้เพื่อให้สามารถออกแบบและสร้างวงจรรวมได้รวดเร็วขึ้น

ในการเขียนรูปแบบบรรยายระบบเชิงเลขในลักษณะของการออกแบบจากบนลงล่างจะต้องทำความเข้าใจในเรื่องของโครงสร้างและส่วนประกอบต่างๆของรูปแบบภาษา VHDL เสียก่อน ซึ่งส่วนประกอบที่สำคัญและเป็นพื้นฐานของการเขียนมี 4 หน่วย คือ

- หน่วยการออกแบบเอนทิตี (Entity Design unit)
- หน่วยการออกแบบสถาปัตยกรรม (Architecture Design unit)
- หน่วยการออกแบบแพ็กเก็ต (Package Design unit)
- หน่วยการออกแบบโครงแบบ (Configuration Design unit)

3.11.1 หน่วยการออกแบบเอนทิตี

หน่วยการออกแบบนี้เป็นส่วนที่ใช้สำหรับติดต่อระหว่างภายนอกกับรูปแบบที่เขียนขึ้น โดยเป็นการกำหนดจุดเชื่อมต่อของรูปแบบ กำหนดทิศทางการไหลของสัญญาณ และประเภทของค่าที่สามารถกำหนดให้กับสัญญาณตามจุดต่างๆของข้อมูลที่ไหลผ่านจุดต่อเหล่านั้น รูปที่ 3.13 แสดงให้เห็นถึงโครงสร้างของหน่วยการออกแบบเอนทิตี

```
Entity component_name is
    Input and Output ports
    Physical and other parameters
End [component_name];
```

รูปที่ 3.13 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบเอนทิตี

ส่วนนี้จะขึ้นต้นด้วยคำว่า Entity และ is ระหว่างคำทั้งสองคำเป็นส่วนสำหรับชื่อของรูปแบบที่ต้องการจะเขียน (component_name) หลังจากนั้นจะตามด้วยส่วนที่ใช้กำหนดช่องทางเข้าและออกของข้อมูล (input-output) รวมทั้งพารามิเตอร์อื่นๆ และที่สำคัญคือ หน่วยการออกแบบเอนทิตีจะต้องปิดท้ายด้วยคำว่า End และเครื่องหมายอัฒภาคเสมอ (;)

```
Entity RS_ff is
    port(S,R : in bit;
         Q,QB : out bit);
End RS_ff;
```

(a) หน่วยการออกแบบเอนทิตี ในรูปของภาษา VHDL

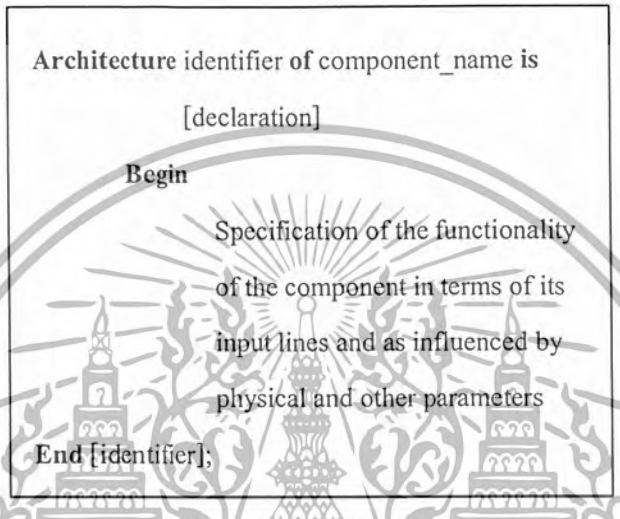
(b) มุมมองของตัวเชื่อมประสาน (Interfacing)

รูปที่ 3.14 แสดงรูปแบบของ RS_flipflop

ในรูปที่ 3.14 เป็นหน่วยการออกแบบเอนทิตีที่บรรยายอุปกรณ์ชื่อ RS_flipflop ในส่วนหัวของเอนทิตีมีการกำหนดจุดต่อ 4 จุด ภายใต้ชุดคำสั่ง port โดยที่ 2 จุดแรกเป็นจุดให้ข้อมูลไหลผ่านเข้า ได้แก่ R, S ซึ่งกำหนดด้วยทิศทางการติดต่อกับโลกภายนอกเป็นการไหลเข้าของข้อมูล (in) ส่วยจุดเอาท์พุทเป็นจุดให้ข้อมูลไหลออก ได้แก่ Q,QB ซึ่งกำหนดด้วยทิศทางการติดต่อกับภายนอกเป็นการไหลออก (out) ส่วนประเภทของข้อมูลที่จะไหลเข้าและออกนั้นเป็นประเภท bit ที่สามารถมีค่าได้เพียงสองค่าเท่านั้น คือ “0” และ “1” เท่านั้น

3.11.2 หน่วยการออกแบบสถาปัตยกรรม

หน่วยการออกแบบสถาปัตยกรรม คือส่วนที่ใช้เขียนบรรยายพฤติกรรมของรูปแบบในมุมมองของการจำลองการทำงาน พฤติกรรมต่างๆที่บรรยายในส่วนนี้ขึ้นอยู่กับข้อมูลที่ผ่านเข้าและออกตรงช่องทางตลอดจนพารามิเตอร์ต่างๆที่กำหนดในหน่วยการออกแบบเอนทิตี รูปที่ 3.15 แสดงให้เห็นถึงโครงสร้างของหน่วยการออกแบบสถาปัตยกรรม



รูปที่ 3.15 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบสถาปัตยกรรม

ส่วนของหน่วยการออกแบบสถาปัตยกรรมเริ่มต้นด้วยคำว่า Architecture และตามด้วยชื่อ (identifier) สิ่งที่ต้องกำหนดลงไปได้แก่ สิ่งที่แสดงให้เห็นว่า Architecture นั้นใช้บรรยายหน่วยการออกแบบเอนทิตีใดๆ (of <entity design unit> is) ส่วนที่อยู่ระหว่าง Architecture และ Begin เป็นพื้นที่ส่วนประกาศหน่วยของสถาปัตยกรรมกำหนด (Architecture declaration area) ที่เป็นส่วนเผื่อเลือก (Option) ในบริเวณนี้สามารถใช้เขียนประกาศกำหนดค่าต่างๆที่จะนำไปใช้ภายในสถาปัตยกรรมนั้นได้ อาทิเช่น ประเภท (Type) ต่างๆ (ตัวอย่างเช่น bit, bit_vector), สัญญาณ (signal), ค่าคงที่ (constant), โปรแกรมย่อย (ได้แก่ function และ procedure) และอุปกรณ์ (component) ส่วนที่ใช้บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้าและไหลออกของรูปแบบ (สัญญาณที่กำหนดในชุดคำสั่ง port) นั้น จะถูกบรรยายในบริเวณเนื้อที่ระหว่างคำว่า Begin กับ End ของหน่วยการออกแบบสถาปัตยกรรม และนอกจากนั้นชุดคำสั่งทุกคำสั่งที่อยู่ภายในบริเวณนี้จะเป็นชุดคำสั่งแบบแข่งขนาน (Concurrent statement) เท่านั้น คือทุกๆ statement จะทำงานพร้อมกัน ลำดับก่อนหลังจะไม่มีผลต่อการทำงานของรูปแบบ หน่วยการออกแบบสถาปัตยกรรมจะต้องปิดท้ายด้วยคำสั่ง End และชื่อของสถาปัตยกรรมนั้นๆ โดยทั่วไปการเขียนรูปแบบระบบเชิงเลขด้วยภาษา VHDL สามารถเขียนได้ในลักษณะต่างๆ ดังนี้

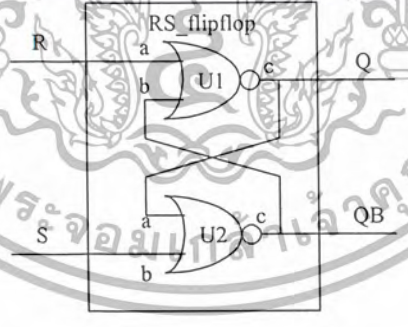
- ลักษณะการไหลของข้อมูล (Dataflow style)
- ลักษณะพฤติกรรม (Behavioral style)
- ลักษณะโครงสร้าง (Structural style)
- ลักษณะผสม (Mixed Model style)

```
Architecture dataflow of RS_ff is
        Begin
                Q  <=  not(QB or R);
                QB <= not(Q or S);
        End dataflow;
```

รูปที่ 3.16 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop

ตามฟังก์ชั่นบูลีน $Q = \overline{QB + R}$ และ $QB = \overline{Q + S}$

รูปที่ 3.16 ส่วนที่บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้า (R, S) กับข้อมูลที่ไหลออก (Q, QB) ประกอบด้วยชุดคำสั่งแบบแข่งขนาน 2 ชุด ซึ่งเขียนเป็นประเภทการไหลของข้อมูล หรือเรียกว่า ระดับการถ่ายโอนข้อมูลระหว่างรีจิสเตอร์ (RTL : Register Transfer Level)



รูปที่ 3.17 แสดงโครงสร้างภายในสถาปัตยกรรมของ RS_flipflop

รูปที่ 3.18 เป็นหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop ในลักษณะโครงสร้าง ซึ่งเปรียบเสมือนการนำอุปกรณ์ที่มีอยู่ในไลบรารี่ (Library) มาต่อเป็นวงจรตามต้องการ โดยใช้นอร์เกท 2 อินพุท (nor2) จำนวนสองตัวมาสร้างตามฟังก์ชั่นบูลีนของรูปที่ 3.16

```
Architecture struc of RS_ff is
        component nor2
            port(a,b : in bit;
                 c :out bit);
        end component;

    Begin
        U1 : nor2 port map(R,QB,Q);
        U2 : nor2 port map(S,Q,QB);
End struc;
```

รูปที่ 3.18 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop ในลักษณะโครงสร้าง

```
Architecture behave of RS_ff is
    Begin
        process(R,S)
            begin
                if R='0' and S='1' then
                    Q <= '1';
                    QB <= '0';
                elsif R='1' and S='0' then
                    Q <= '1';
                    QB <= '0';
                end if;
        end process;
End behave;
```

รูปที่ 3.19 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop ในลักษณะพฤติกรรม

รูปที่ 3.19 เป็นการเขียนบรรยายการทำงานของรูปแบบในลักษณะพฤติกรรม ซึ่งจะเห็นได้ว่ามีลักษณะที่เหมือนกับการเขียนโปรแกรมทั่วไป โดยจะต้องมีการใช้งานส่วนที่เรียกว่า process และการทำงานของรูปแบบจะขึ้นอยู่กับการเปลี่ยนแปลงของสิ่งที่อยู่ภายใน process (อินพุท R, S) ซึ่งเรียกว่า Sensitivity list การเขียนในลักษณะนี้ลำดับก่อนหลังของชุดคำสั่งจะมีผลต่อการทำงานของรูปแบบที่เขียนขึ้น

```
Architecture mixed of RS_ff is
        component nor2
                port(a,b : in bit;
                        c : out bit);
        end component;
Begin
        U1 : nor2 port map(R,QB,Q);
        QB <= not(Q or S);
End mixed;
```

รูปที่ 3.20 แสดงหน่วยการออกแบบสถาปัตยกรรมของ RS_flipflop ในลักษณะผสม

ไม่ว่าจะเขียนบรรยายส่วนของสถาปัตยกรรมของ RS_flipflop ในลักษณะของพฤติกรรม การไหลของข้อมูล โครงสร้าง หรือผสมที่นำเอาแต่ละลักษณะมาเขียนไว้ในส่วนของสถาปัตยกรรมก็ตาม ต่างก็มีพฤติกรรมเดียวกัน และจะให้ผลลัพธ์จากการจำลองการทำงานที่เหมือนกัน ซึ่งถือว่าเป็นข้อดีของภาษา VHDL

3.11.3 หน่วยการออกแบบแพ็กเก็ต

ข้อมูลต่างๆ ตลอดจนโปรแกรมย่อยที่เป็นประโยชน์ต่อการเขียนรูปแบบบรรยายระบบเชิงเลขสามารถเก็บไว้ในส่วนของแพ็กเก็ตได้ และข้อมูลเหล่านี้สามารถเรียกไปใช้ได้โดยหน่วยการออกแบบเอนทิตี หน่วยการออกแบบสถาปัตยกรรม หรือจากหน่วยการออกแบบแพ็กเก็ตอื่นๆ โดยปกติแล้ว แพ็กเก็ตจะแบ่งออกเป็น 2 ส่วน คือการประกาศแพ็กเก็ต (Package Declaration) และส่วนของบอดี้แพ็กเก็ต (Package Body) เนื่องจากแพ็กเก็ตถูกสร้างขึ้นเป็นส่วนแยกต่างหากออกจากรูปแบบที่กำลังเขียนอยู่ ฉะนั้นการที่จะนำแพ็กเก็ตไปใช้นั้นจะต้องมีการเชื่องโยงหรืออ้างอิงเสียก่อน ซึ่งในภาษา VHDL สามารถกระทำได้ด้วยชุดคำสั่ง USE

Package Declaration

ส่วนที่มีความสำคัญที่สุดของแพ็กเก็ต (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ส่วนการประกาศแพ็กเก็ต เพราะจะเป็นส่วนที่กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพ็เก็ตสำหรับนำไปใช้ภายนอกตัวของแพ็กเก็ตเอง สิ่งใดๆที่ถูกประกาศไว้ในส่วนของบอดี้แพ็กเก็ตแต่ไม่ได้ถูกประกาศไว้ในส่วนการประกาศแพ็กเก็ตจะไม่สามารถถูกนำค่าและพฤติกรรมไปใช้ส่วนนอกได้ ซึ่งสามารถเปรียบเทียบได้ดับสิ่งที่ประกาศไว้ในส่วนของการประกาศเอนทิตี คือจุดเชื่อมต่อ หรือพอร์ท ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพ็กเก็ตสามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้ และยังสามารถถูกนำไปใช้จากรูปแบบภายนอกได้ เช่น ใช้สำหรับประกาศชนิด (Type) หรือสัญญาณ เช่นเดียวกันกับส่วนบอดี้แพ็กเก็ตที่ไม่จำเป็นต้องมีส่วนของการประกาศแพ็กเก็ต แต่แพ็กเก็ตนั้นจะไม่สามารถถูกนำไปใช้จากรูปแบบอื่นได้

```
Package package_name is
  Package_declaration_part
End package_name;
```

รูปที่ 3.21 แสดงโครงสร้างโดยทั่วไปของส่วนการประกาศแพ็กเก็ต

Package body

โครงสร้างที่ประกอบด้วยคำสั่งต่างๆในรูปของคำสั่งลำดับ (Sequence) ที่ใช้บรรยายฟังก์ชั่นการทำงานของโปรแกรมย่อย (Subprogram) ทั้งหลาย ที่ชื่อของโปรแกรมย่อยนั้นๆที่ถูกประกาศไปในส่วนของการประกาศแพ็กเก็ตแล้วจะถูกเก็บไว้ในส่วนบอดี้แพ็กเก็ต ทั้งนี้รวมทั้งการกำหนดค่าคงที่ต่างๆ อันได้แก่ค่าคงที่ที่ถูกประกาศชื่อก่อนในส่วนของการประกาศแพ็กเก็ต แต่ถูกกำหนดค่าในส่วนของบอดี้แพ็กเก็ต ฉะนั้นส่วนบอดี้แพ็กเก็ตจึงไม่จำเป็นต้องมีถ้าในส่วนของการประกาศแพ็กเก็ตไม่มีการประกาศชื่อที่เป็นโปรแกรมย่อยหรือค่าคงที่ การเขียนบอดี้แพ็กเก็ตนั้นเป็นไปตามกฎเกณฑ์ที่แสดงในรูปที่ 3.22

```
Package body package_name is
  declarative part
End package_name;
```

รูปที่ 3.22 แสดงโครงสร้างโดยทั่วไปของบอดี้แพ็กเก็ต

3.11.4 หน่วยการออกแบบโครงแบบ

ดังที่ทราบกันแล้วว่ารูปแบบหนึ่งของระบบเชิงเลขไม่ว่าจะเป็นอะไร จะมีหน่วยการออกแบบเอนทิตีได้เพียงหนึ่งเดียวเท่านั้น แต่หน่วยการออกแบบเอนทิตีหนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลายหน่วย ดังนั้นจะต้องมีหน่วยการออกแบบโครงแบบมาเพื่อกำหนดการใช้โครงแบบ (Configuration) ประกอบเอนทิตีกับหน่วยการออกแบบสถาปัตยกรรมหน่วยไหนเข้าด้วยกัน

```
Configuration identifier of entity_name is
        Configuration_declarative_part
End;
```

รูปที่ 3.23 แสดงโครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

3.12 การออกแบบแอลเอ็มเอสอัลกอริทึมโดยใช้โครงสร้างเลขคณิตกระจาย (LMS Adaptive Filter Using Distributed Arithmetic)

ให้ S(k) คืออินพุทเวคเตอร์ และ W(k) คือเวคเตอร์สัมประสิทธิ์ ซึ่งแทนด้วย

$$\mathbf{S}(k) = [s(k), s(k-1), \ldots, s(k-N+1)]^T$$

$$\mathbf{W}(k) = [w_0(k), w_1(k), \ldots, w_{N-1}(k)]^T$$

สัญญาณเอาท์พุทของFIR filter คือ

$$y(k) = \mathbf{S}^T(k)\mathbf{W}(k) = \mathbf{F}^T\mathbf{A}^T(k)\mathbf{W}(k) \qquad (3.1)$$

นิยาม Address matrix A(k) และ Scaling vector F ดังนี้

$$\mathbf{A}(k) = \begin{bmatrix} b_0(k) & \cdots & b_0(k-N+1) \\ b_1(k) & \cdots & b_1(k-N+1) \\ \vdots & \ddots & \vdots \\ b_{B-1}(k) & \cdots & b_{B-1}(k-N+1) \end{bmatrix}^T$$

และ

$$\mathbf{F} = [-2^0, 2^1, \ldots, 2^{-(B-1)}]^T$$

โดยที่ B = word length ของสัญญาณอินพุท

ความสัมพันธ์ระหว่างสัญญาณอินพุท กับ Address matrix คือ

$$s(k) = [b_0(k), b_1(k), \ldots, b_{B-1}(k)]\mathbf{F} \tag{3.2}$$

นิยาม Address vector แทนด้วย

$$\mathbf{A}_{vi}(k) = [b_i(k), b_i(k-1), \ldots, b_i(k-N+1)]^T \quad ; i=0, 1, \ldots, B-1$$

สมการปรับสัมประสิทธิ์ของ LMS Algorithm แทนด้วย

$$\mathbf{W}(k+1) = \mathbf{W}(k)+2\mu e(k)\mathbf{S}(k) \tag{3.3}$$

จากสมการที่ (3.3) คูณทั้งสองข้างของสมการด้วย $\mathbf{A}^T(k)$ จะได้

$$\mathbf{A}^T(k)\,\mathbf{W}(k+1) = \mathbf{A}^T(k)[\mathbf{W}(k)+2\mu e(k)\,\mathbf{S}(k)] \tag{3.4}$$

แทน $\mathbf{S}(k) = \mathbf{A}(k)\mathbf{F}$ ลงในสมการที่ (3.4) จะได้

$$\mathbf{A}^T(k)\,\mathbf{W}(k+1) = \mathbf{A}^T(k)[\mathbf{W}(k)+2\mu e(k)\mathbf{A}(k)\mathbf{F}] \tag{3.5}$$

สัญญาณความผิดพลาด e(k) สามารถนิยามโดย

$$e(k) = d(k) - y(k) \tag{3.6}$$

AFS (Adaptive Function Space) นิยามโดย

$$\mathbf{P}(k) = \mathbf{A}^T(k)\mathbf{W}(k) \tag{3.7}$$

$$= [p_0(k), \ldots, p_{B-1}(k)]^T$$

และ

$$\mathbf{P}(k+1) = \mathbf{A}^T(k)\mathbf{W}(k+1) \tag{3.8}$$

$$= [p_0(k+1), \ldots, p_{B-1}(k+1)]^T$$

i^{th} elements ของ P(k) และ $\mathbf{P}(k+1)$ เป็นค่า partial product ที่สัมพันธ์กับ Address vector $\mathbf{A}_{vi}(k)$ ซึ่งก็คือ i^{th} column vector ของ matrix $\mathbf{A}(k)$

ดังนั้นค่า time index ของ P(k) และ P(k+1) จะสอดคล้องกับค่าของ $\mathbf{W}(k)$ และ $\mathbf{W}(k+1)$ ตามลำดับ ค่า partial product ทั้งหมดจะมีเท่ากับ 2^N โดย N^{th} คือ order ของ filter เราเรียก space นี้ (P(k) และ P(k+1)) ซึ่งมี 2^N partial product นี้ว่า "AFS"

แทนค่าสมการ (3.8) และ (3.7) ลงในสมการที่ (3.5) จะได้

$$\mathbf{P}(k+1) = \mathbf{P}(k)+2\mu e(k)\mathbf{A}^T(k)\mathbf{A}(k)\mathbf{F} \tag{3.9}$$

โดยการใช้สมการที่ (3.7) จะได้สัญญาณ Output y(k) คือ

$$y(k) = \mathbf{F}^T\,\mathbf{P}(k) \tag{3.10}$$

และจากสมการที่ (3.9) ถ้าสมมุติว่า Input Signal คือ white noise with zero mean ดังนั้นค่า expectation values ของ $\mathbf{A}^T(k)\mathbf{A}(k)\mathbf{F}$ คือ

$$E[\mathbf{A}^T(k)\mathbf{A}(k)\mathbf{F}] = 0.25N\mathbf{F} \tag{3.11}$$

จากนั้นแทนค่า $\mathbf{A}^T(k)\mathbf{A}(k)\mathbf{F}$ ลงในสมการที่ (3.9) ด้วยสมการที่ (3.11) ดังนั้นสมการที่ (3.9) จะกลายเป็น

$$\mathbf{P}(k+1) = \mathbf{P}(k)+0.2\mu Ne(k)\mathbf{F} \tag{3.12}$$

3.13 โครงสร้างและการทำงานของวงจรกรองความถี่แบบปรับตัวได้ชนิดผลตอบสนองอิมพัลส์จำกัดโดยใช้โครงสร้างเลขคณิตกระจาย

1. สัญญาณอินพุท SK ซึ่งเป็นสัญญาณที่ทำการแปลงเป็นเลขจำนวนส่วนเติมเต็มสองจะถูกโหลดเข้า PISO (Parallel In Serial Out) โดยขาสัญญาณ (lr) และใช้ clock ในการ shift สัญญาณออกทีละบิท

2. สัญญาณที่ shift ออกมาจาก PISO จะป้อนให้วงจร SISO (Serial In Aerial Out) และจะ shift ด้วยสัญญาณ clock ออกทีละบิทเช่นกัน

3. เอาท์พุทที่ได้จาก PISO และ SISO จะถูกแบ่งไป 2 ทาง ส่วนหนึ่งจะนำไปชี้ Address ของ RAM (Random Access Memory) และอีกส่วนหนึ่งจะป้อนให้วงจร SISO เพื่อทำการวนค่ากลับมาชี้ Address ของ RAM อีกครั้งเมื่อ shift จนครบ 8 bits แล้ว

4. อินพุทที่เข้าขา Address จะไปชี้ค่าสัมประสิทธิ์ที่ถูกเก็บไว้ใน RAM ซึ่งเป็นค่า inner product of constant vector ออกมา output ที่ได้จะถูกส่งต่อไปยังอินพุทของวงจรสเกลลิ่งแอคคิวมูเลเตอร์

5. วงจรสเกลลิ่งแอคคิวมูเลเตอร์จะทำหน้าที่ในการบวกลบเลขส่วนเติมเต็มสอง (2 's complement) แทนการคูณโดยตรง ซึ่งภายในสเกลลิ่งแอคคิวมูเลเตอร์ประกอบด้วยวงจร บวก/ลบ (Add/Sub) และรีจิสเตอร์แอคคิวมูเลเตอร์ ด้านเอาท์พุทของวงจรรีจิสเตอร์แอคคิวมูเลเตอร์ ถูกออกแบบให้เป็นฮาร์ดแวร์สเกลลิ่ง (Hardware Scaling) ด้วย 2^{-1} ก่อนที่จะป้อนกลับไปบวกกับค่าที่ออกมาจากหน่วยความจำต่อไป แล้วจึงนำค่าที่ได้โหลดออก Buffer

6. สัญญาณที่ได้ที่ถูกโหลดออกมาเป็นสัญญาณ Output ส่วนหนึ่งจะถูกนำไปใช้ในการหา error signal (e(k)) โดยนำไปหาผลต่างกับ Desired signal (d(k))

7. สัญญาณความผิดพลาด (error signal) จะถูกนำไปคำนวณสัมประสิทธิ์ค่าใหม่ P(k+1) ตามสมการที่ 3.12 โดยนำสัมประสิทธิ์ค่าเก่า P(k) ไปบวกเข้ากับผลคูณของ $0.2\mu Ne(k)\mathbf{F}$

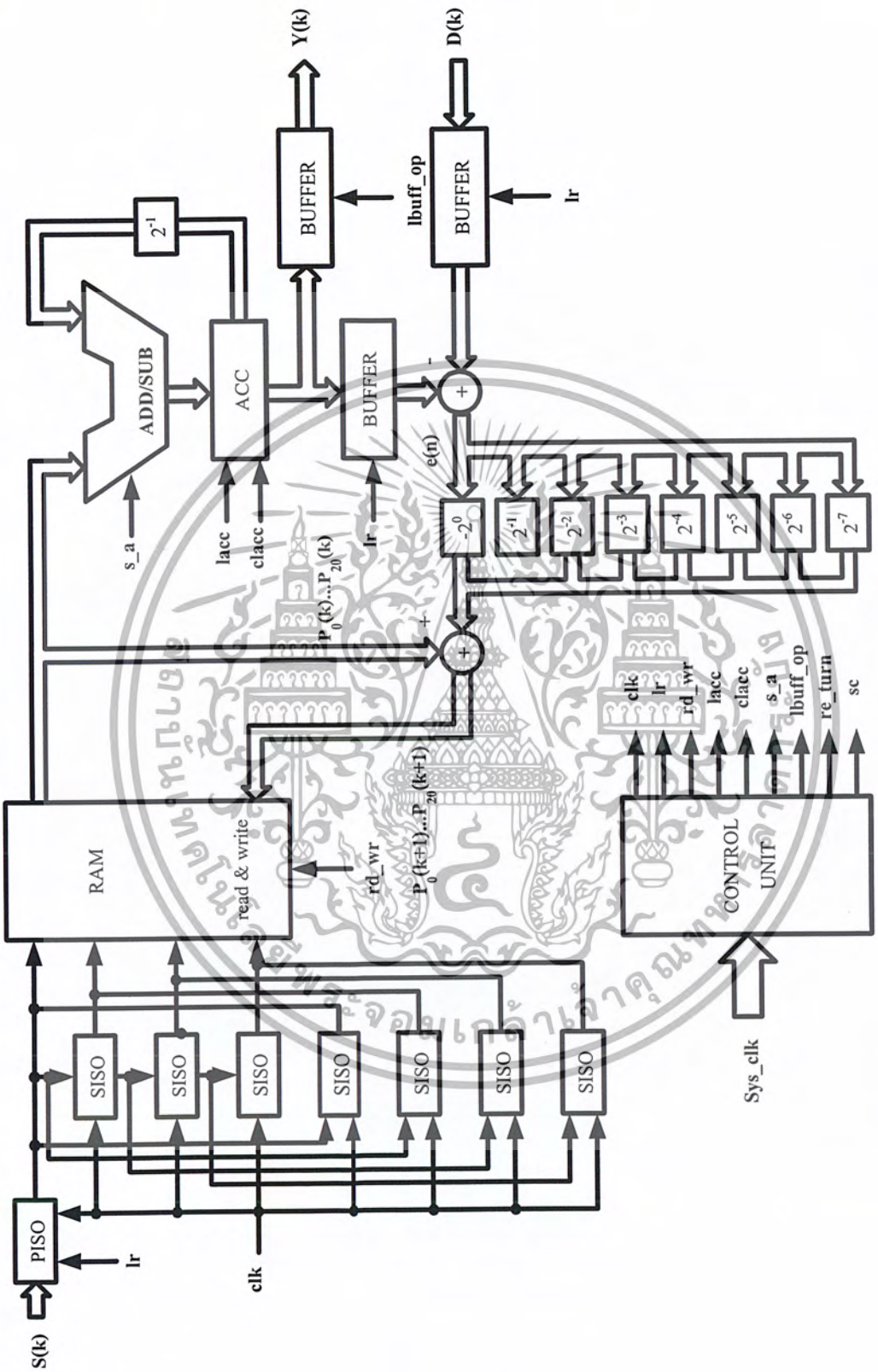
โดย μ คือค่า step size ต้องกำหนดให้เหมาะสม

N คือจำนวนอันดับ (order) ของตัวกรอง

F คือweight ของสัญญาณแต่ละ bit ของ input

เมื่อได้ค่าสัมประสิทธิ์ใหม่แล้วก็จะถูกนำไปเก็บไว้ใน RAM ที่ Address เดิม

8. การทำงานจะวนซ้ำตั้งแต่ขั้นตอนที่ 1-7 ซ้ำต่อไปเรื่อย ๆ จนสามารถ track สัญญาณที่ต้องการได้



รูปที่ 3.24 แสดงโครงสร้างของวงจรกรองสัญญาณแบบปรับตัวได้ โดยใช้โครงสร้างเลขคณิตกระจาย

บทที่ 4
การทดลองและผลการทดลอง

4.1 การซิมูเลชั่นของตัวกรองปรับตัวแบบเอฟไออาร์

การซิมูเลชั่นหาผลตอบสนองต่าง ๆ ของตัวกรองปรับตัวนั้นจะทำการซิมูเลชั่นจากโปรแกรมซึ่งเขียนขึ้นโดยใช้โปรแกรม MATLAB โดยกำหนดเลือดใช้ความถี่สัญญาณเบสแบนด์ ใช้ความถี่การชักตัวอย่างเท่ากับ 8 kHz และใช้จำนวนไอเทอร์เรชั่นเท่ากับ 1000 ไอเทอร์เรชั่น และทำการเปลี่ยนค่าต่าง ๆ คือ

4.1.1 หาผลตอบสนองต่อค่าความผิดพลาดเฉลี่ยกำลังสองน้อยสุด (Mean Square Error : MSE)

โดยทำการเปลี่ยนอันดับของตัวกรองระหว่าง 4, 8 และ 16 และทำการเปลี่ยนขนาด Step size (μ) ต่าง ๆ ดังนี้ 0.0625, 0.03125, 0.015625

4.1.2 หาผลตอบสนองต่อสัญญาณรบกวนรูปไซน์ที่ผสมกับสัญญาณเบสแบนด์

โดยทำการเปลี่ยนอันดับของตัวกรองระหว่าง 4, 8 และ 16 และทำการเปลี่ยนขนาด Step size (μ) ต่าง ๆ ดังนี้ 0.0625, 0.03125, 0.015625 แล้วทำการหาผลตอบสนองต่อสัญญาณรบกวนรูปไซน์ความถี่ 5 kHz ที่ผสมกับสัญญาณเบสแบนด์ความถี่ 100 Hz

4.1.3 หาผลตอบสนองต่อสัญญาณรบกวนสุ่มที่ผสมกับสัญญาณเบสแบนด์

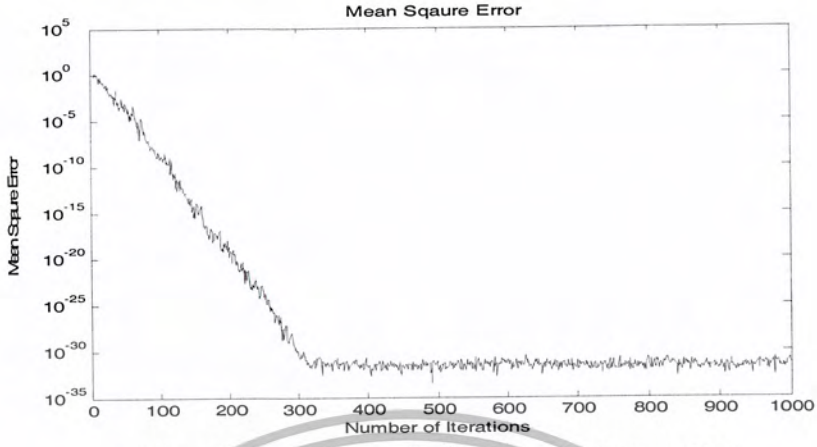
โดยทำการเปลี่ยนอันดับของตัวกรองระหว่าง 4, 8 และ 16 และทำการเปลี่ยนขนาด Step size (μ) ต่าง ๆ ดังนี้ 0.0625, 0.03125, 0.015625 แล้วทำการหาผลตอบสนองต่อสัญญาณรบกวนสุ่มที่ผสมกับสัญญาณเบสแบนด์ความถี่ 100 Hz

4.1.4 หาผลตอบสนองต่อสัญญาณรบกวนรูปไซน์ความถี่ต่าง ๆ ที่ผสมกับสัญญาณเบสแบนด์

โดยเลือกใช้ตัวกรองอันดับ 4 กำหนดขนาด Step size (μ) เท่ากับ 0.0625 แล้วทำการหาผลตอบสนองต่อสัญญาณรบกวนรูปไซน์ที่ความถี่ 1 kHz, 10 kHz และ 100 kHz ที่ผสมกับสัญญาณเบสแบนด์ความถี่ 100 Hz

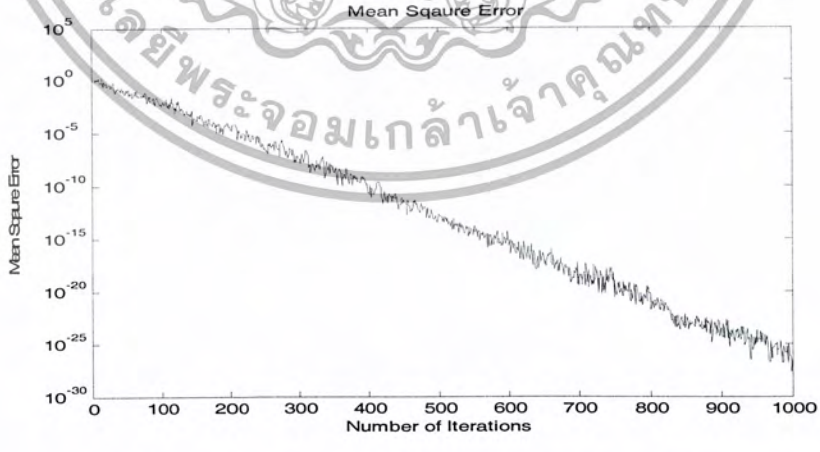
4.2 ผลตอบสนองต่าง ๆ จากการซิมูเลชั่น

4.2.1 ผลตอบสนองต่อค่าความผิดพลาดเฉลี่ยกำลังสองน้อยสุด (Mean Square Error : MSE)



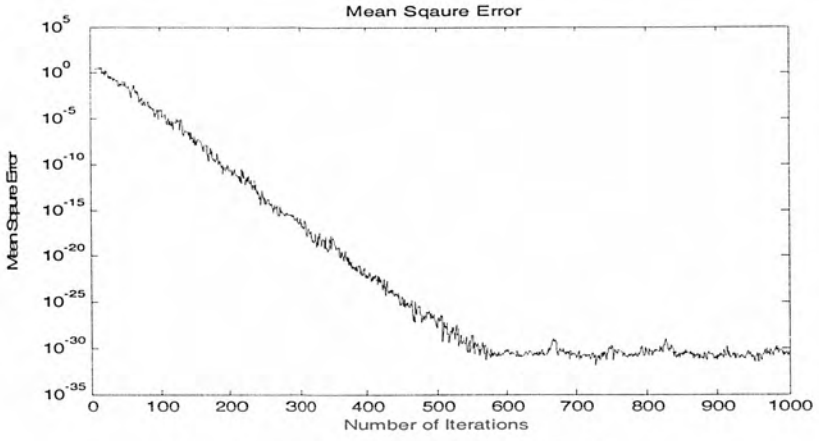
ก) ค่า MSE ของวงจรกรองอันดับ 4 Step size (μ) = 0.0625

ข) ค่า MSE ของวงจรกรองอันดับ 4 Step size(μ) = 0.03125

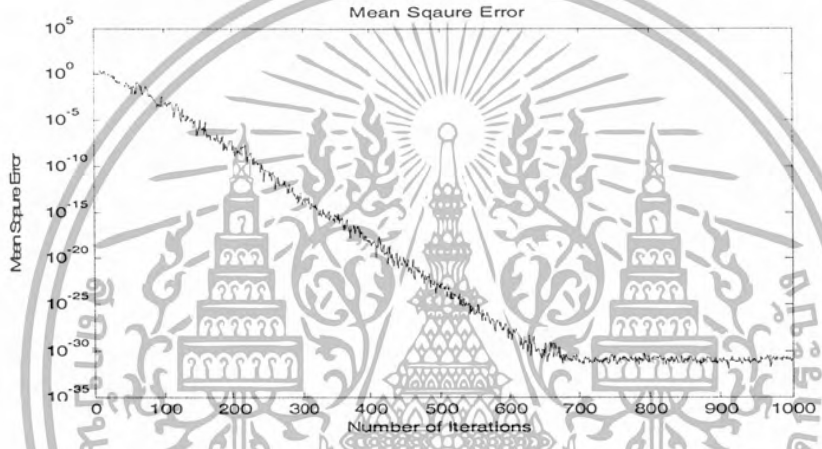


ค) ค่า MSE ของวงจรกรองอันดับ 4 Step size(μ) = 0.015625

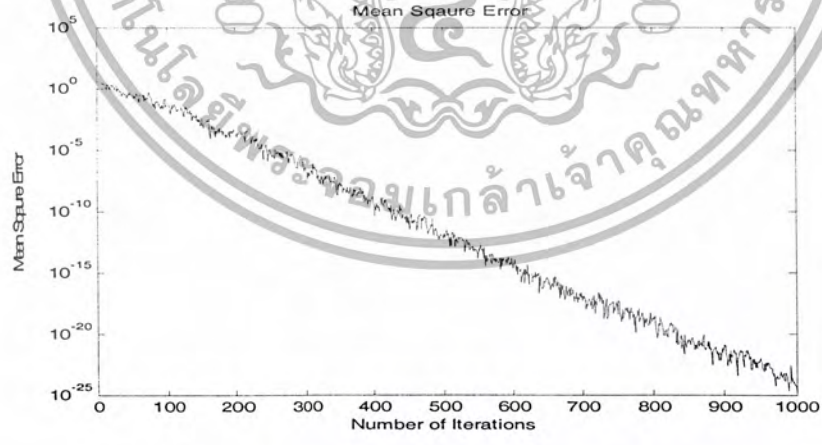
รูปที่ 4.1 ผลตอบสนองต่อค่าความผิดพลาดเฉลี่ยกำลังสองน้อยสุด ของวงจรกรองอันดับ 4 ที่ค่า Step size (μ) ต่าง ๆ



ก) ค่า MSE ของวงจรกรองอันดับ 8 Step size(μ) = 0.0625

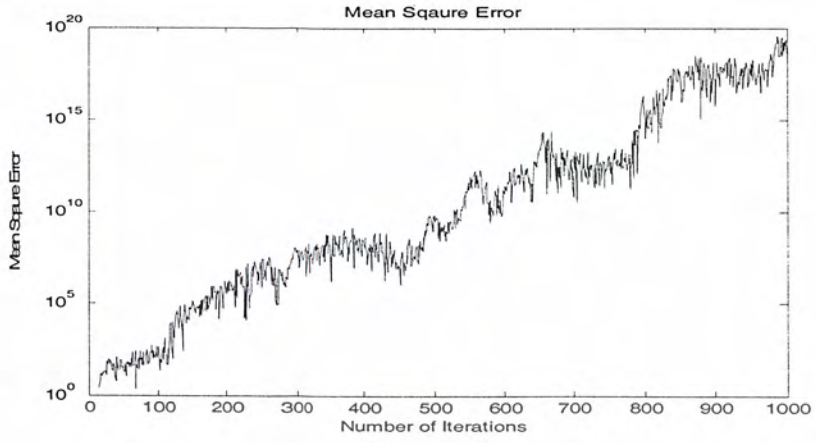


ข) ค่า MSE ของวงจรกรองอันดับ 8 Step size(μ) = 0.03125

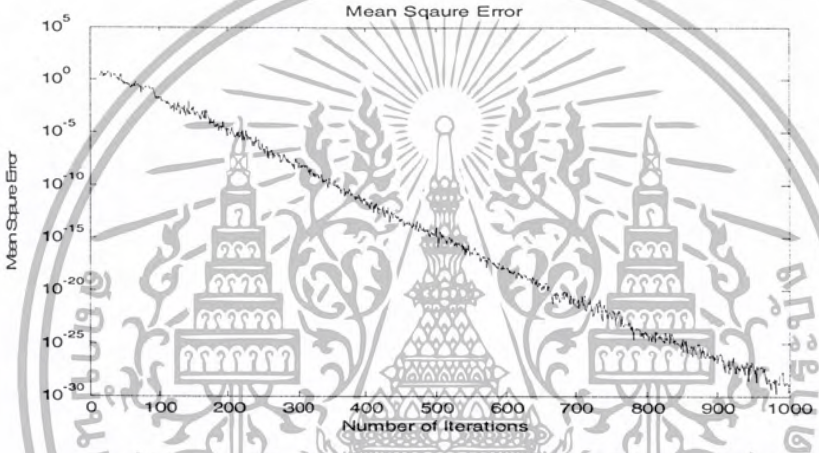


ค) ค่า MSE ของวงจรกรองอันดับ 8 Step size(μ) = 0.015625

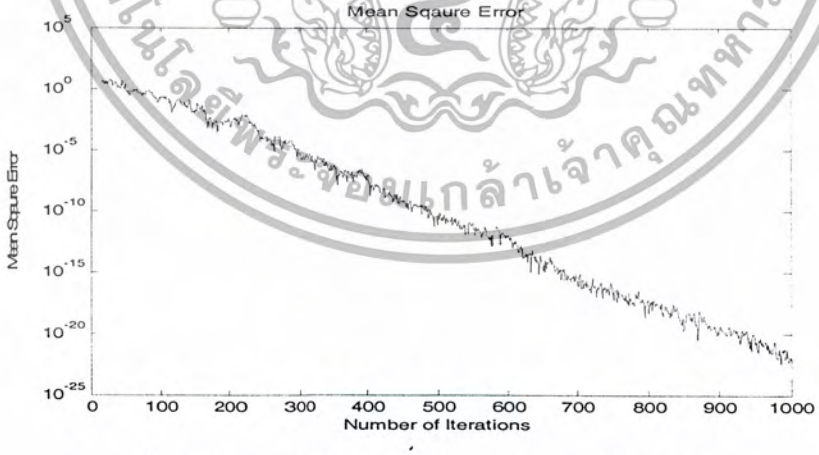
รูปที่ 4.2 ผลตอบสนองต่อค่าความผิดพลาดเฉลี่ยกำลังสองน้อยสุด ของวงจรกรองอันดับ 8 ที่ค่า Step size (μ) ต่าง ๆ



ก) ค่า MSE ของวงจรกรองอันดับ 16 Step size(μ) = 0.0625



ข) ค่า MSE ของวงจรกรองอันดับ 16 Step size(μ) = 0.03125

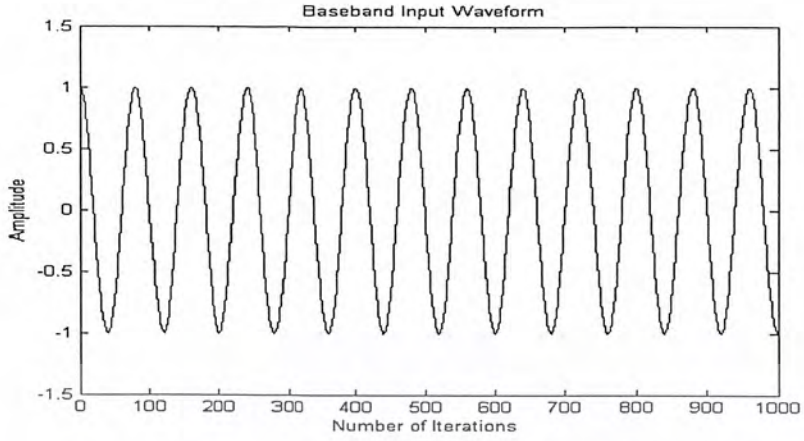


ค) ค่า MSE ของวงจรกรองอันดับ 16 Step size(μ) = 0.015625

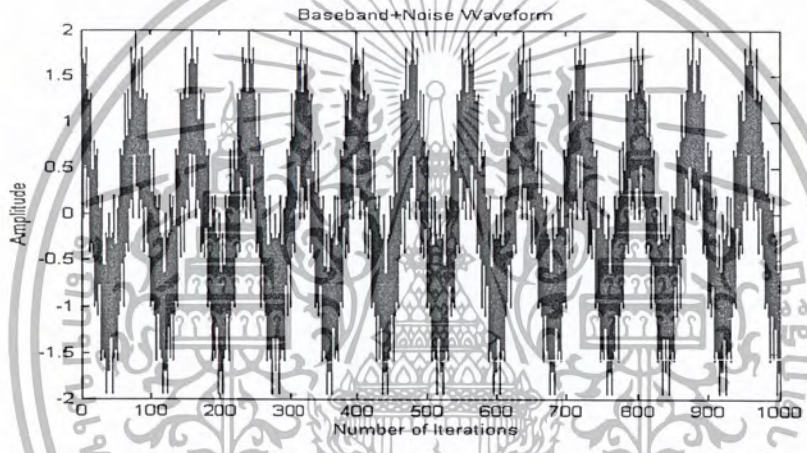
รูปที่ 4.3 ผลตอบสนองต่อค่าความผิดพลาดเฉลี่ยกำลังสองน้อยสุด ของวงจรกรองอันดับ 16 ที่ค่า Step size (μ) ต่าง ๆ

จากผลการ Simulate ด้วยโปรแกรม MATLAB รูปที่ 4.1ถึง 4.3 สังเกตุได้ว่า ค่า Step size(μ) มีค่ามากจะมีการลู่เข้าของสัญญาณได้ดีขึ้น โดยวัดผลจากการลดลงของค่า MSE ที่ order ต่าง ๆ กัน แต่ถ้าทำการเลือกค่า Step size (μ) ที่มีค่ามากเกินไปไม่เหมาะสมกับ order ที่เลือกไว้ จะทำให้ไม่สามารถกรองสัญญาณได้ โดยค่า MSE จะลู่ออก

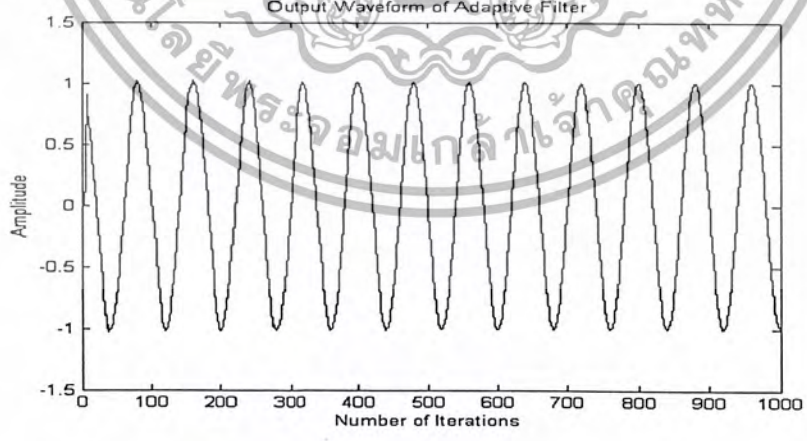
4.2.2 ผลตอบสนองต่อสัญญาณรบกวนรูปไซน์ที่ผสมกับสัญญาณเบสแบนด์



ก) สัญญาณเบสแบนด์

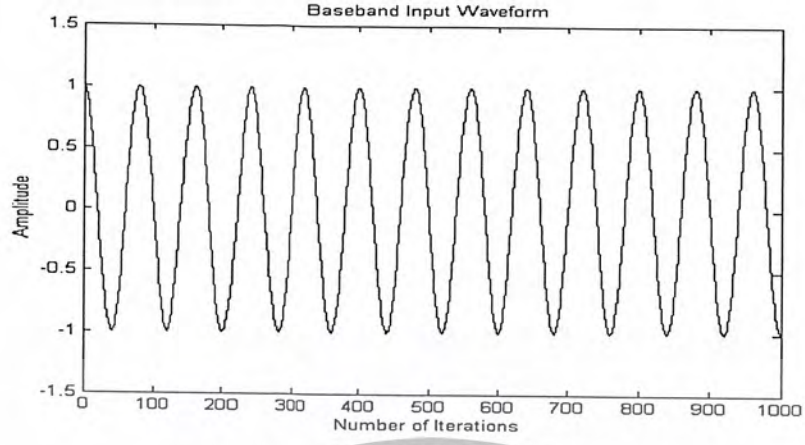


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

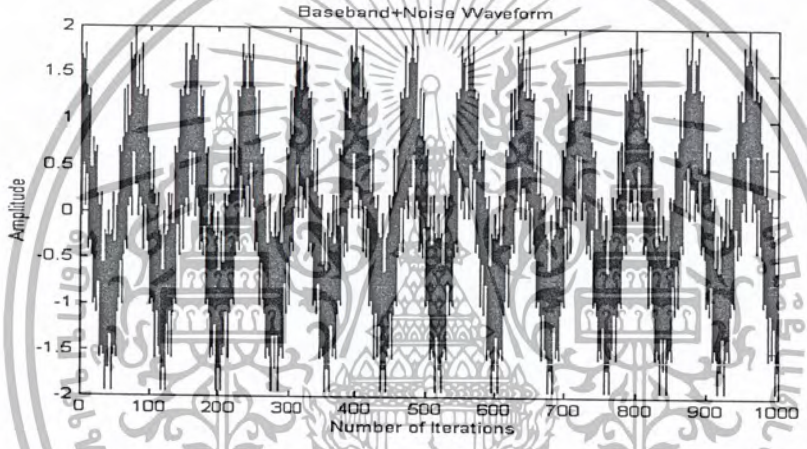


ค) สัญญาณที่กรองได้

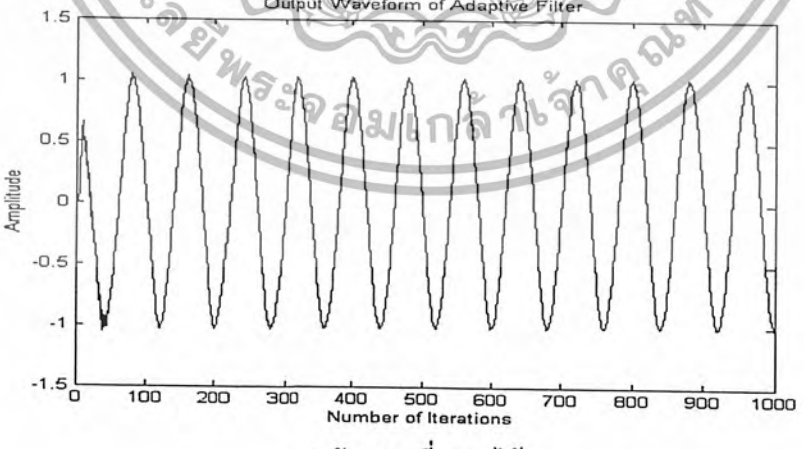
รูปที่ 4.4 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 4 ต่อสัญญาณรบกวนรูปไซน์ความถี่ 5 kHz ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.0625



ก) สัญญาณเบสแบนด์

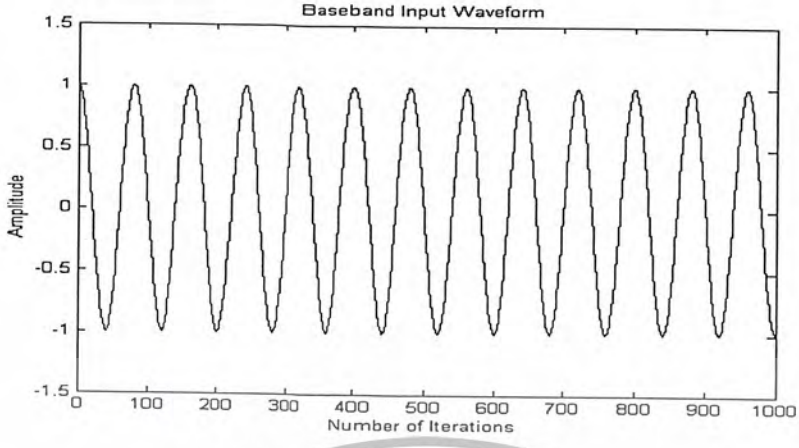


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

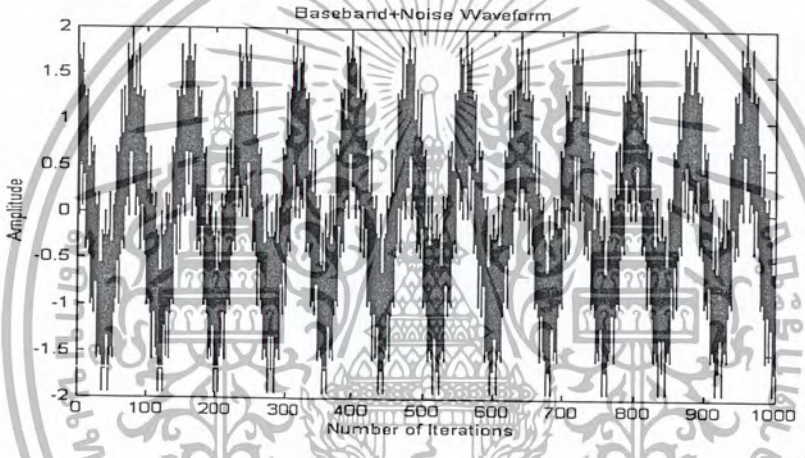


ค) สัญญาณที่กรองได้

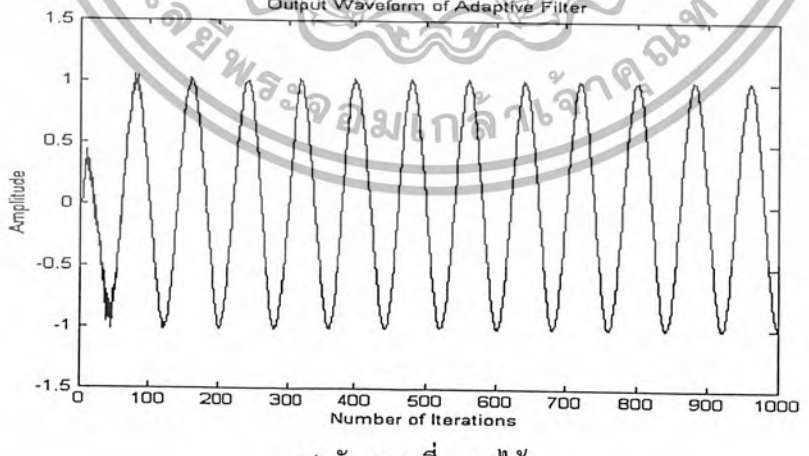
รูปที่ 4.5 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 4 ต่อสัญญาณรบกวนรูปไซน์ความถี่ 5 kHz ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.03125



ก) สัญญาณเบสแบนด์



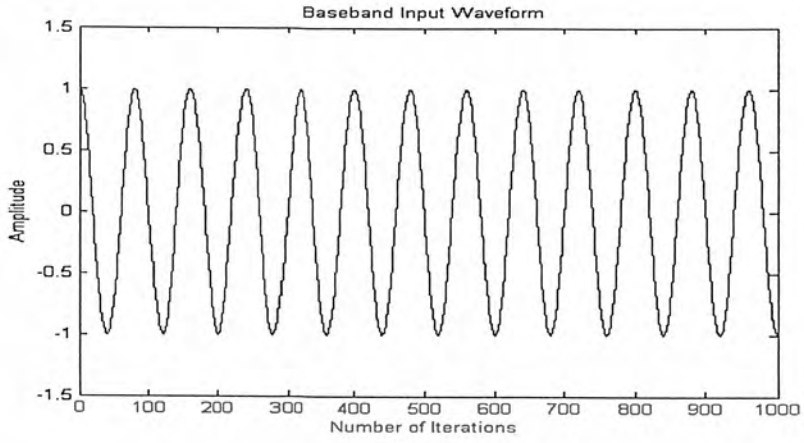
ข) สัญญาณเบสแบนด์ที่ถูกรบกวน



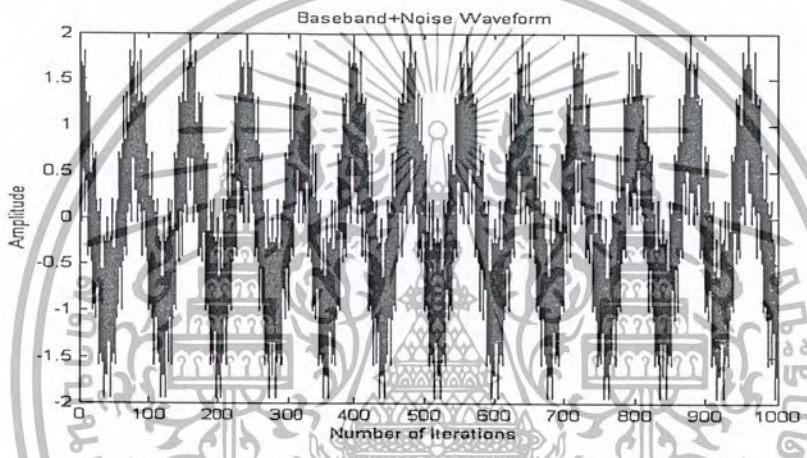
ค) สัญญาณที่กรองได้

รูปที่ 4.6 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 4 ต่อสัญญาณรบกวนรูปไซน์ความถี่ 5 kHz ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.015625

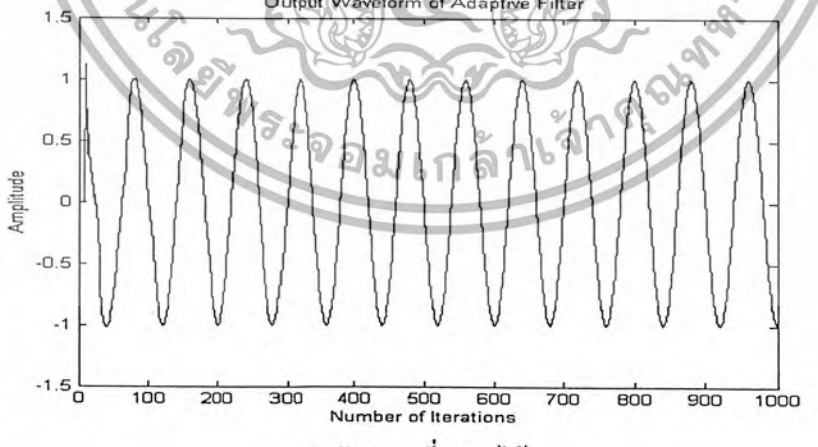
จากผลการ Simulate ด้วยโปรแกรม MATLAB รูปที่ 4.4 ถึง 4.6 เป็นกำหนดค่า order เท่ากับ 4 โดยใช้สัญญาณเบสแบนด์ความถี่ 100 Hz ไปผสมกับสัญญาณรบกวนรูปไซน์ความถี่ 5 kHz ป้อนเข้าวงจรกรอง และทำการเปลี่ยนแปลงค่า Step size (μ) สังเกตุได้ว่าค่า μ ที่มีค่ามากจะทำให้สัญญาณที่กรองได้ มีประสิทธิภาพมากกว่าค่า μ ที่มีค่าน้อย



ก) สัญญาณเบสแบนด์

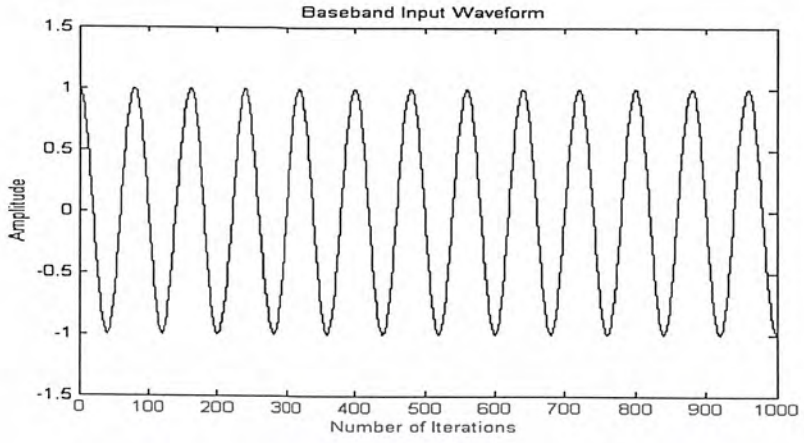


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

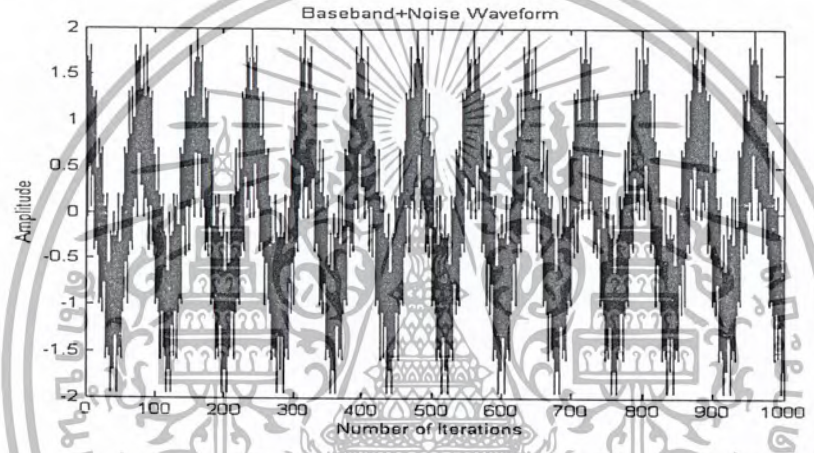


ค) สัญญาณที่กรองได้

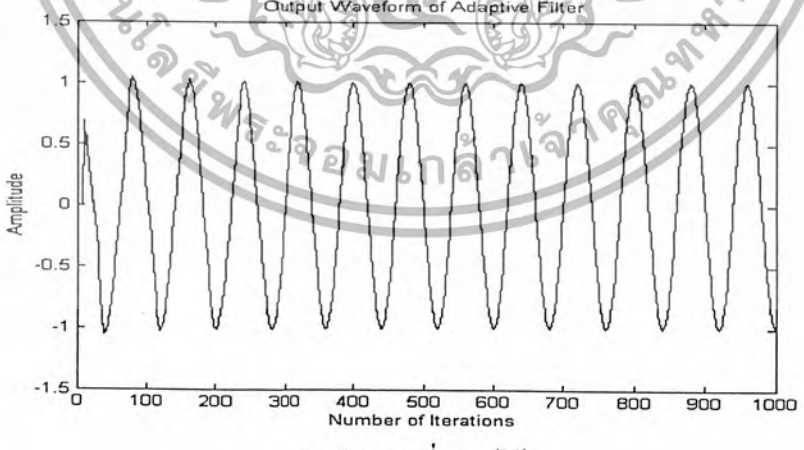
รูปที่ 4.7 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 8 ต่อสัญญาณรบกวนรูปไซน์ ความถี่ 5 kHz ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.0625



ก) สัญญาณเบสแบนด์

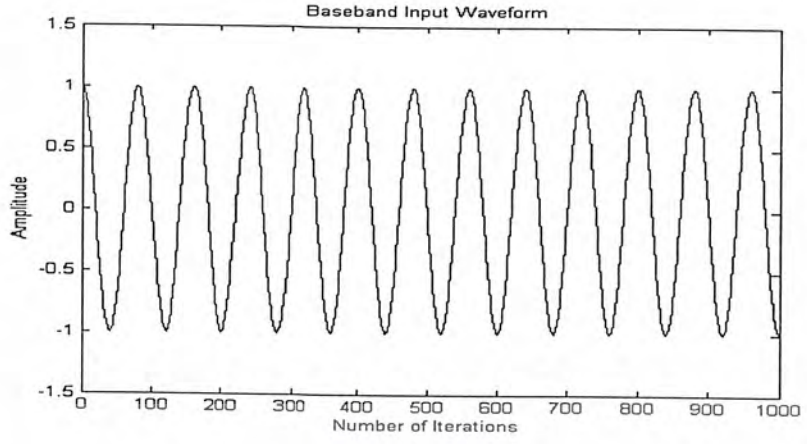


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

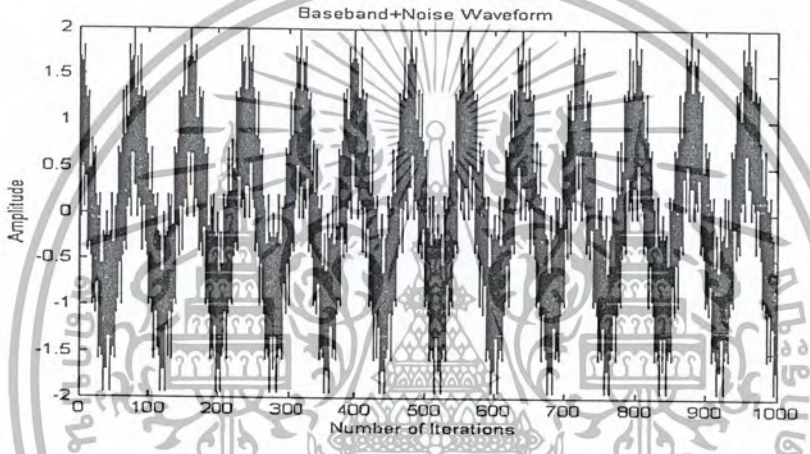


ค) สัญญาณที่กรองได้

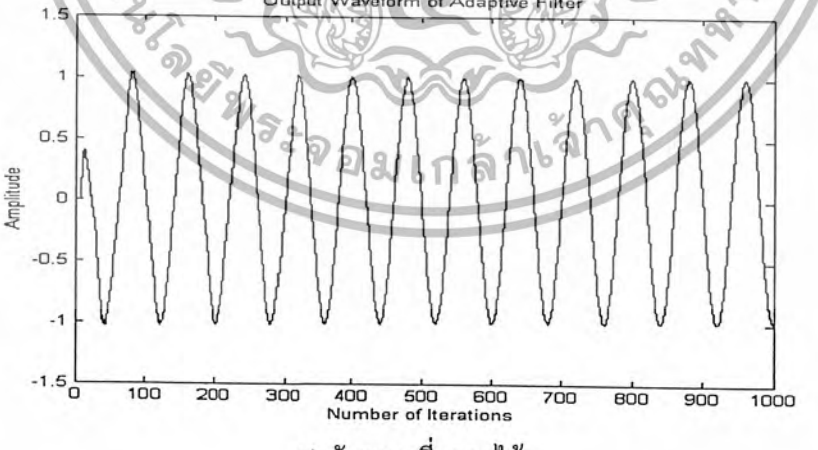
รูปที่ 4.8 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 8 ต่อสัญญาณรบกวนรูปไซน์ ความถี่ 5 kHz ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.03125



ก) สัญญาณเบสแบนด์



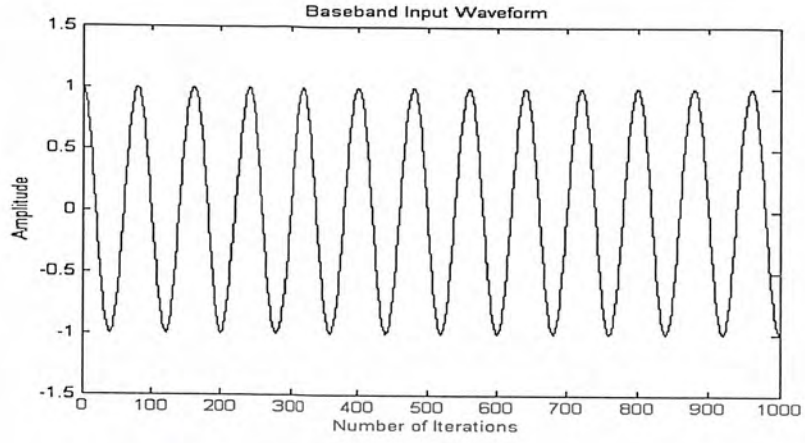
ข) สัญญาณเบสแบนด์ที่ถูกรบกวน



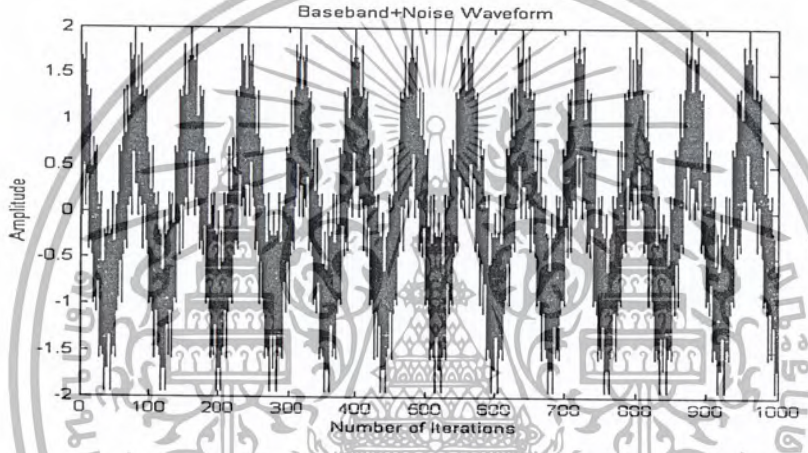
ค) สัญญาณที่กรองได้

รูปที่ 4.9 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 8 ต่อสัญญาณรบกวนรูปไซน์ ความถี่ 5 kHz ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.015625

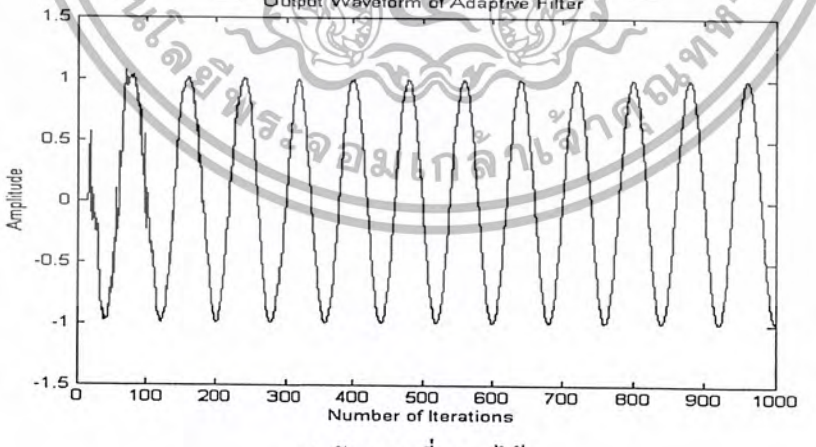
จากผลการ Simulate ด้วยโปรแกรม MATLAB รูปที่ 4.7 ถึง 4.9 เป็นกำหนดค่า order เท่ากับ 8 โดยใช้สัญญาณเบสแบนด์ความถี่ 100 Hz ไปผสมกับสัญญาณรบกวนรูปไซน์ความถี่ 5 kHz ป้อนเข้าวงจรกรอง และทำการเปลี่ยนแปลงค่า Step size (μ) สังเกตุได้ว่าค่า μ ที่มีค่าน้อยจะทำให้สัญญาณที่กรองได้ มีประสิทธิภาพมากกว่าค่า μ ที่มีค่ามาก และใช้เวลาในการลู่เข้าที่เร็วกว่า order 4



ก) สัญญาณเบสแบนด์

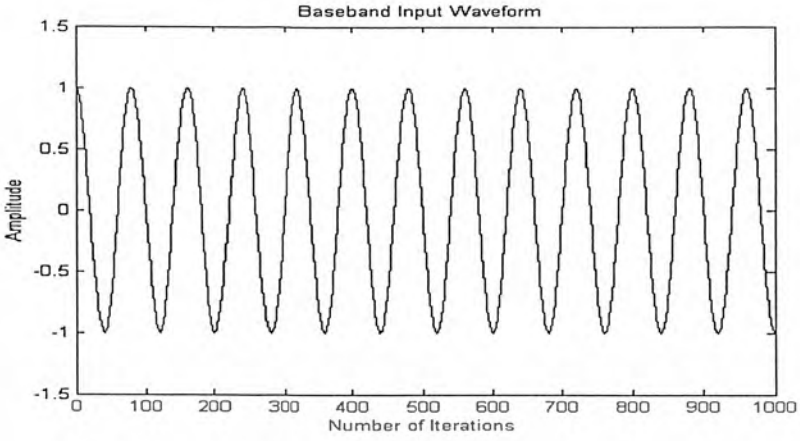


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

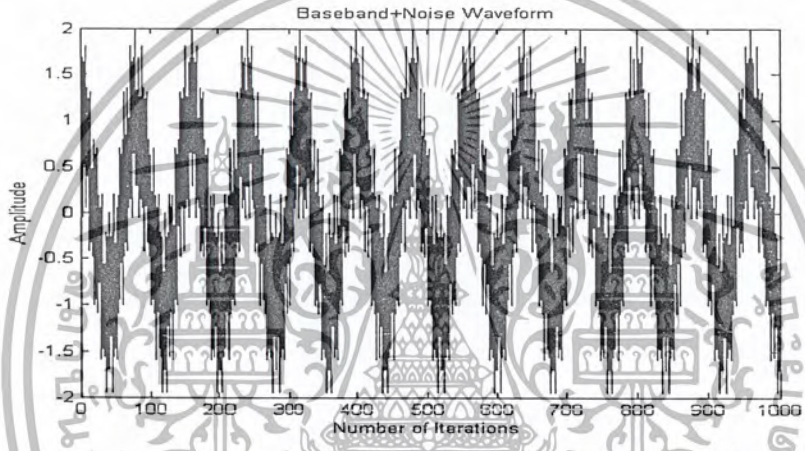


ค) สัญญาณที่กรองได้

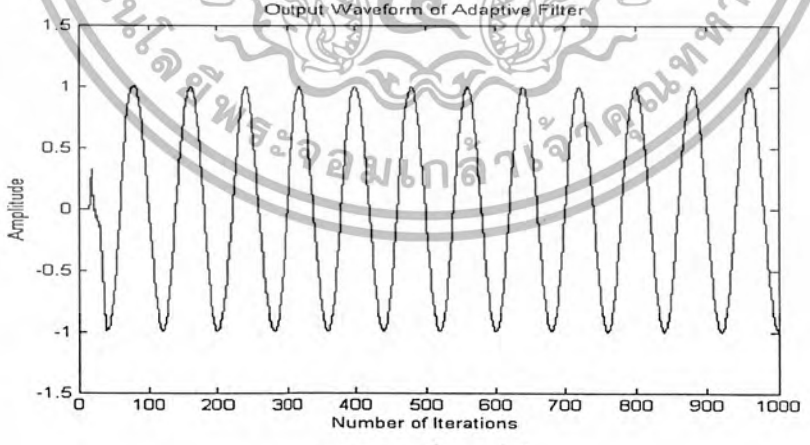
รูปที่ 4.10 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 16 ต่อสัญญาณรบกวนรูปไซน์ ความถี่ 5 kHz ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.0625



ก) สัญญาณเบสแบนด์

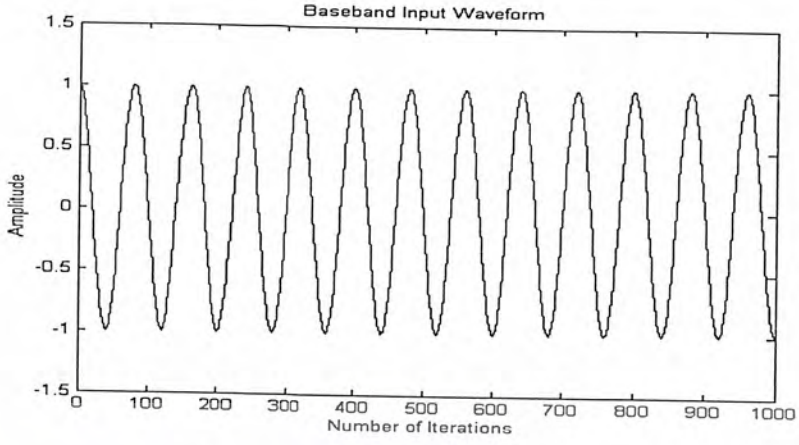


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

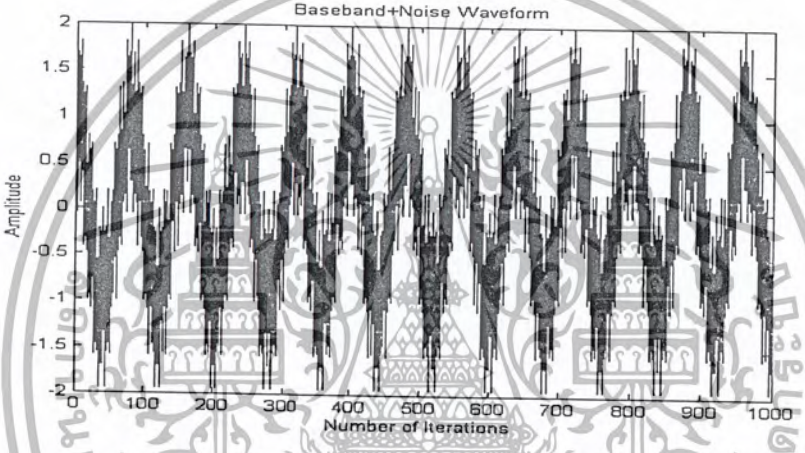


ค) สัญญาณที่กรองได้

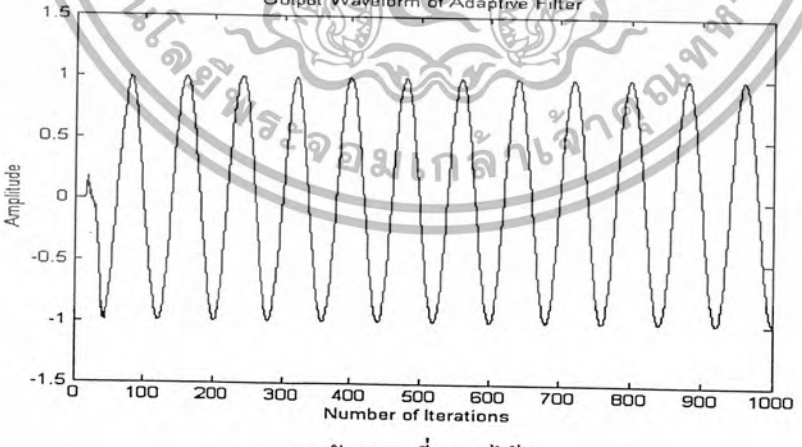
รูปที่ 4.11 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 16 ต่อสัญญาณรบกวนรูปไซน์ ความถี่ 5 kHz ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.03125



ก) สัญญาณเบสแบนด์



ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

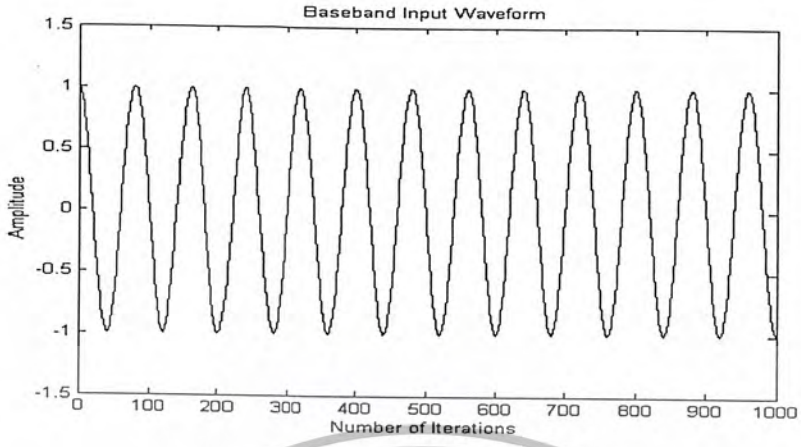


ค) สัญญาณที่กรองได้

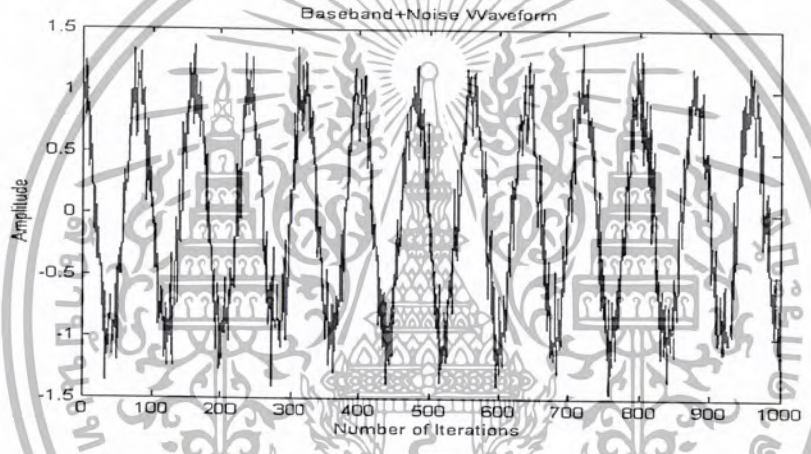
รูปที่ 4.12 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 16 ต่อสัญญาณรบกวนรูปไซน์ ความถี่ 5 kHz ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.015625

จากผลการ Simulate ด้วยโปรแกรม MATLAB รูปที่ 4.10 ถึง 4.12 เป็นกำหนดค่า order เท่ากับ 16 โดยใช้สัญญาณเบสแบนด์ความถี่ 100 Hz ไปผสมกับสัญญาณรบกวนรูปไซน์ความถี่ 5 kHz ป้อนเข้าวงจรกรอง และทำการเปลี่ยนแปลงค่า Step size (μ) สังเกตุได้ว่าค่า μ ที่มีค่าน้อยจะทำให้สัญญาณที่กรองได้ มีประสิทธิภาพมากกว่าค่า μ ที่มีค่ามาก และใช้เวลาในการลู่เข้าที่เร็วกว่า order 4 และ order 8

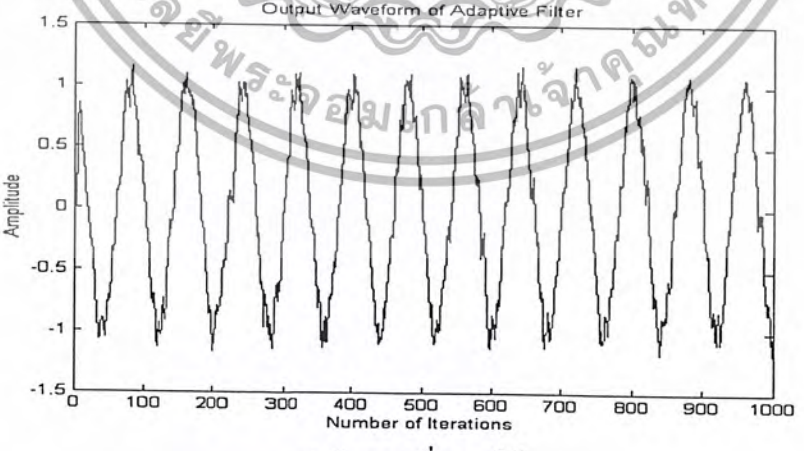
4.2.3 ผลตอบสนองต่อสัญญาณรบกวนสุ่มที่ผสมกับสัญญาณเบสแบนด์



ก) สัญญาณเบสแบนด์

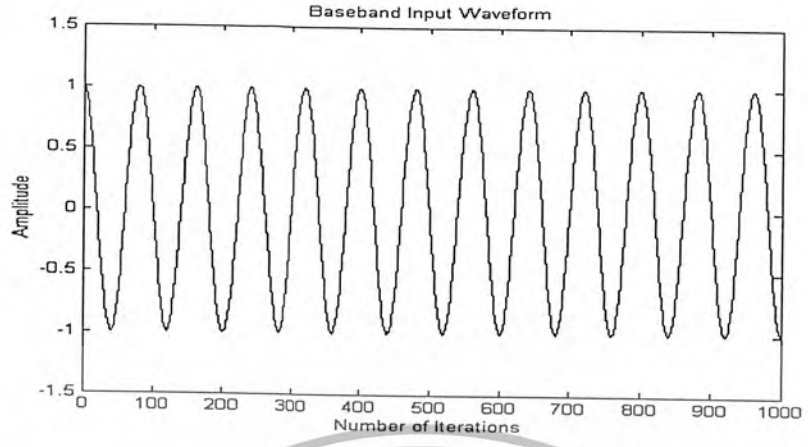


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

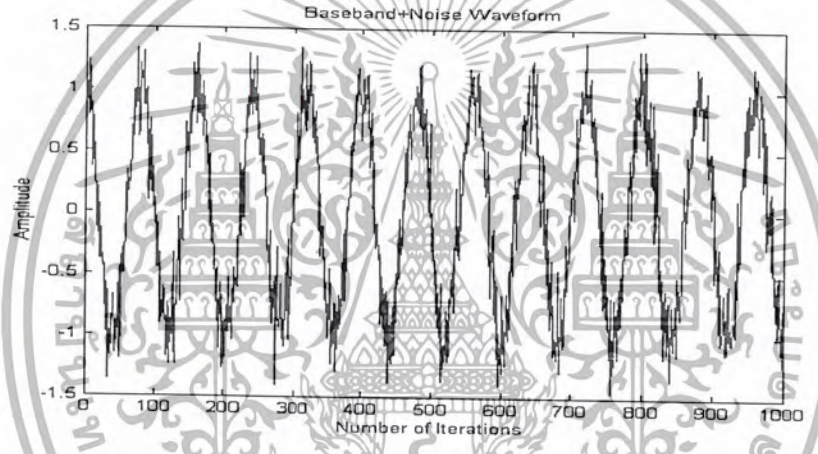


ค) สัญญาณที่กรองได้

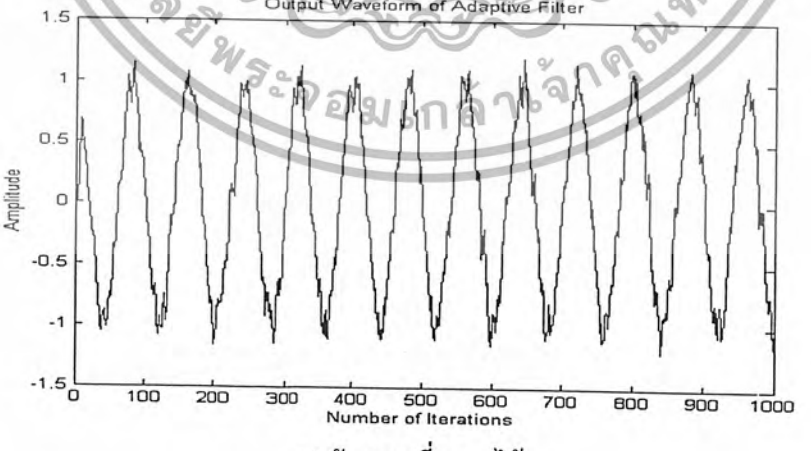
รูปที่ 4.13 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 4 ต่อสัญญาณรบกวนสุ่ม ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.0625



ก) สัญญาณเบสแบนด์

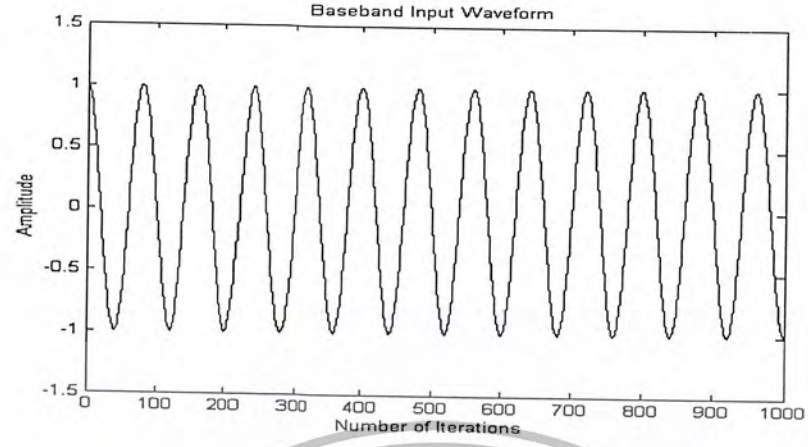


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

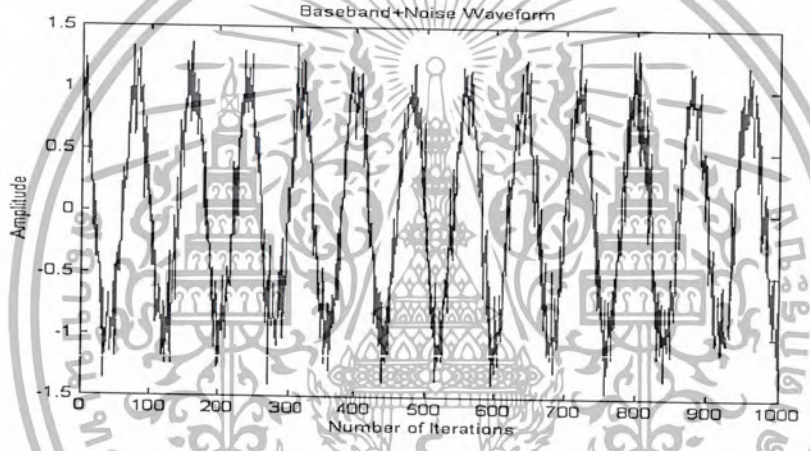


ค) สัญญาณที่กรองได้

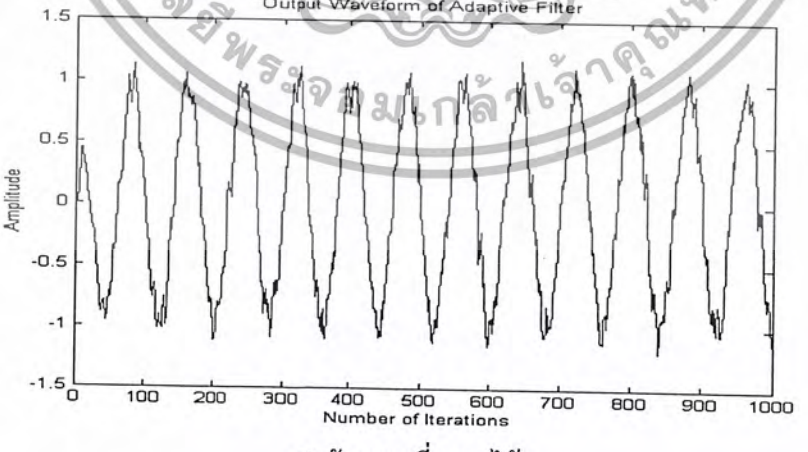
รูปที่ 4.14 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 4 ต่อสัญญาณรบกวนสุ่ม ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.03125



ก) สัญญาณเบสแบนด์

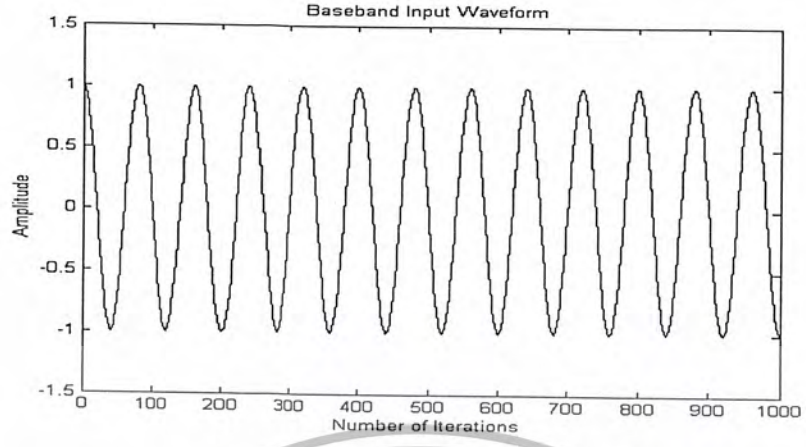


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

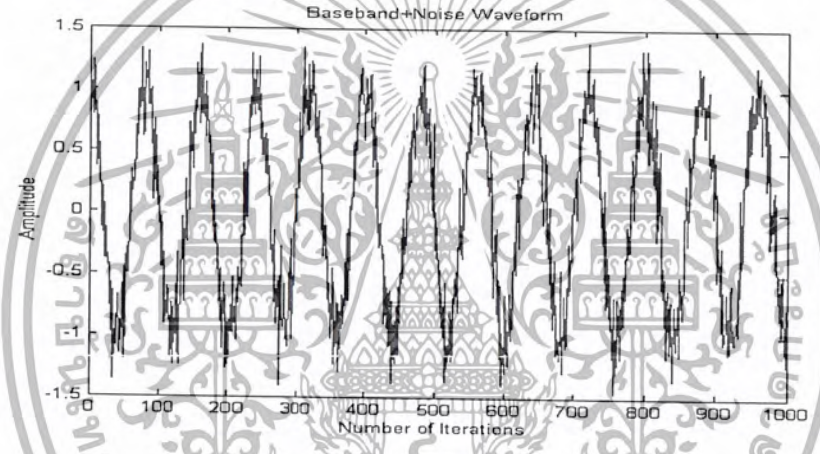


ค) สัญญาณที่กรองได้

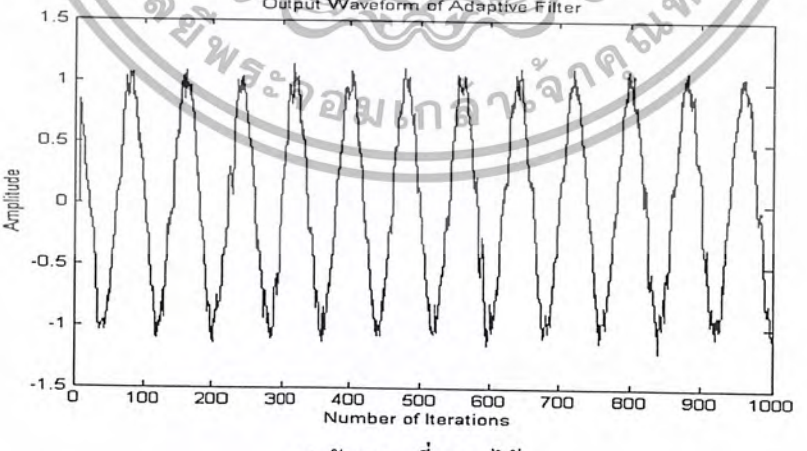
รูปที่ 4.15 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 4 ต่อสัญญาณรบกวนสุ่ม ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.015625



ก) สัญญาณเบสแบนด์

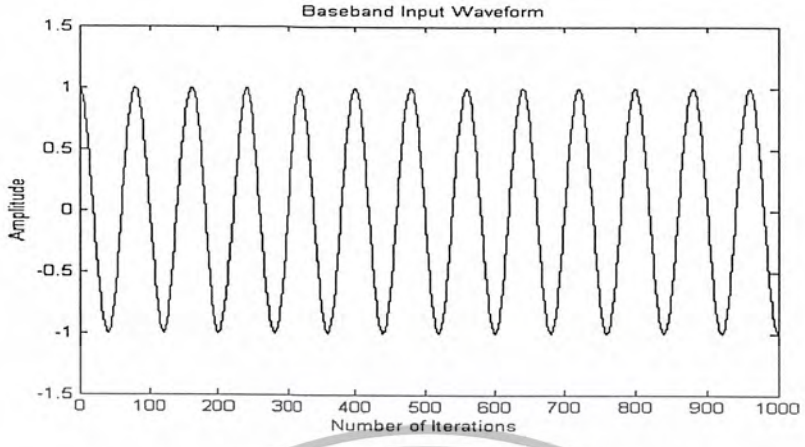


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

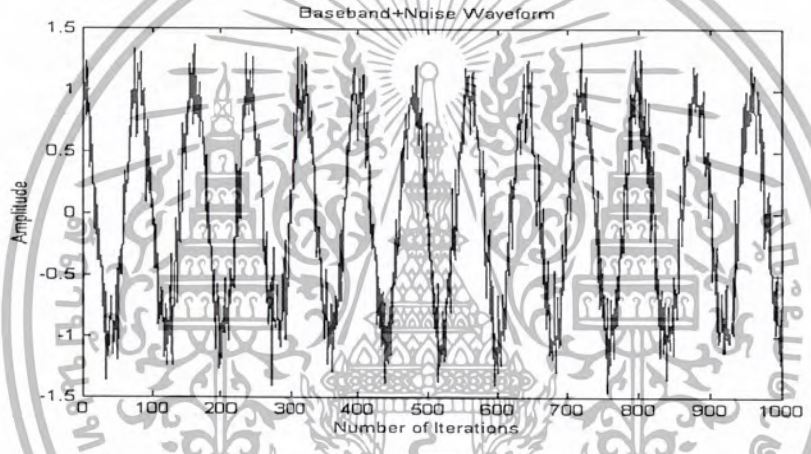


ค) สัญญาณที่กรองได้

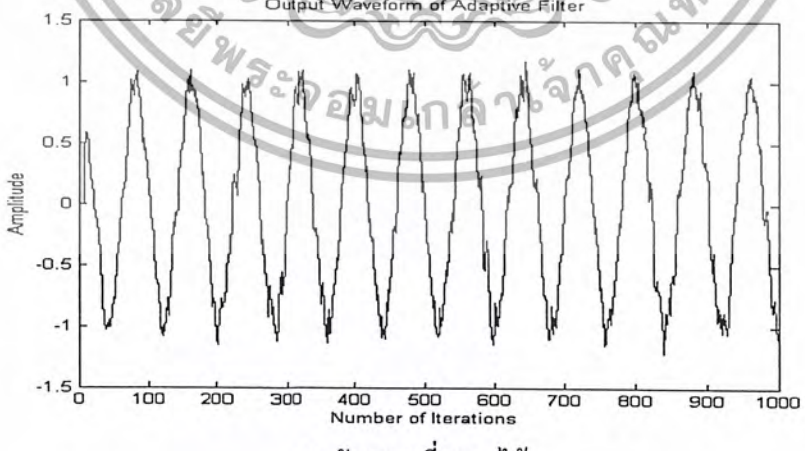
รูปที่ 4.16 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 8 ต่อสัญญาณรบกวนสุ่ม ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.0625



ก) สัญญาณเบสแบนด์

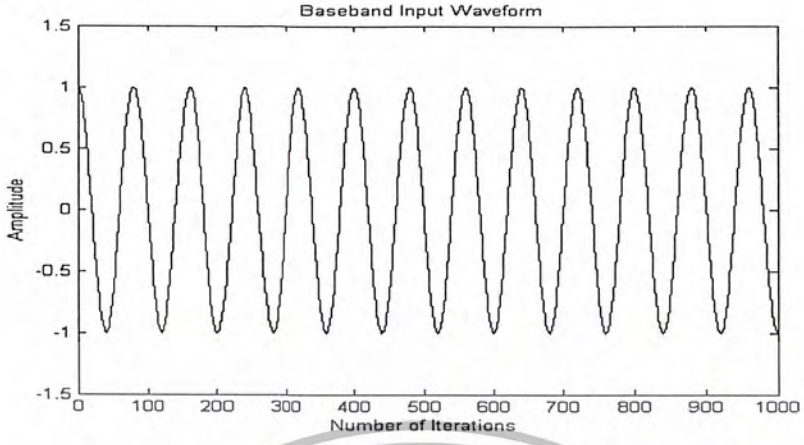


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

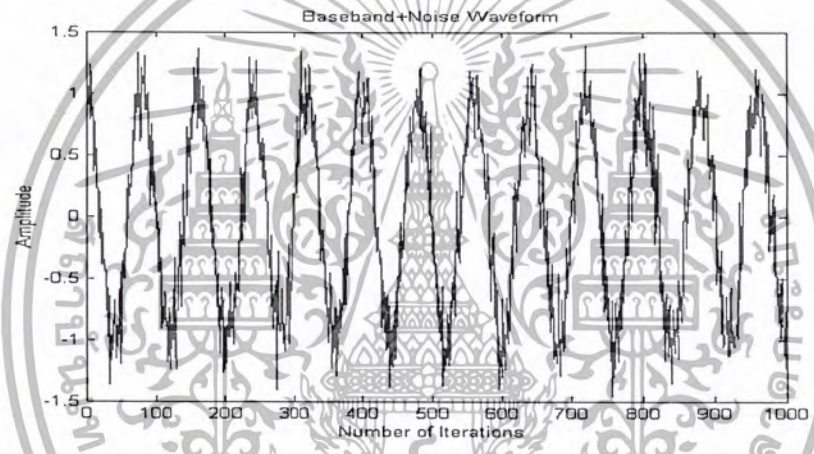


ค) สัญญาณที่กรองได้

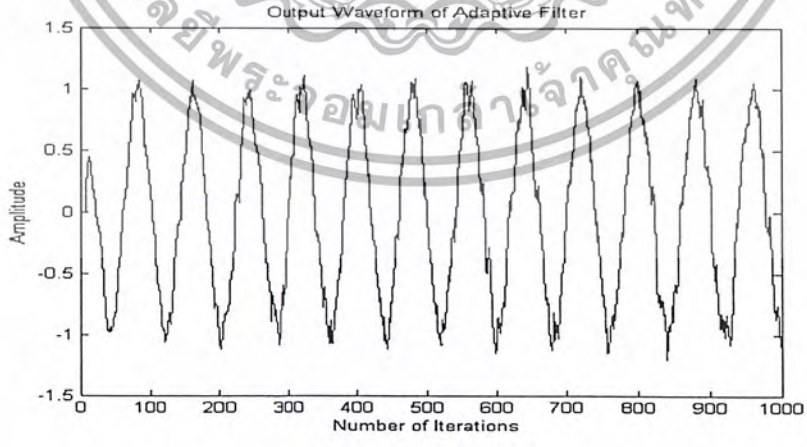
รูปที่ 4.17 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 8 ต่อสัญญาณรบกวนสุ่ม ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.03125



ก) สัญญาณเบสแบนด์

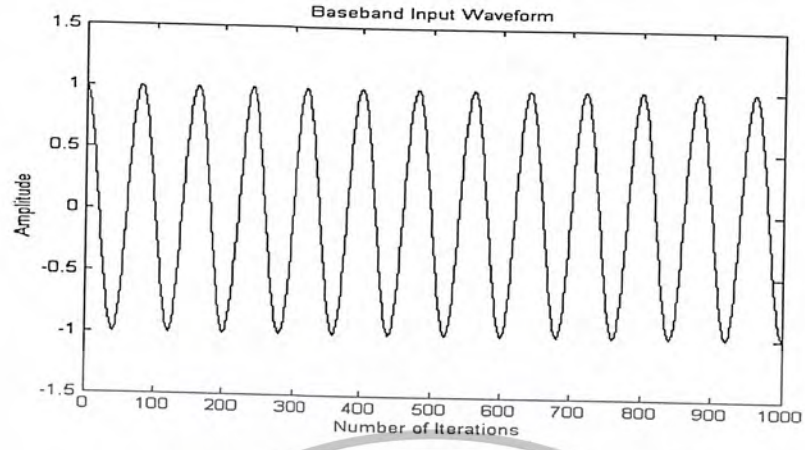


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

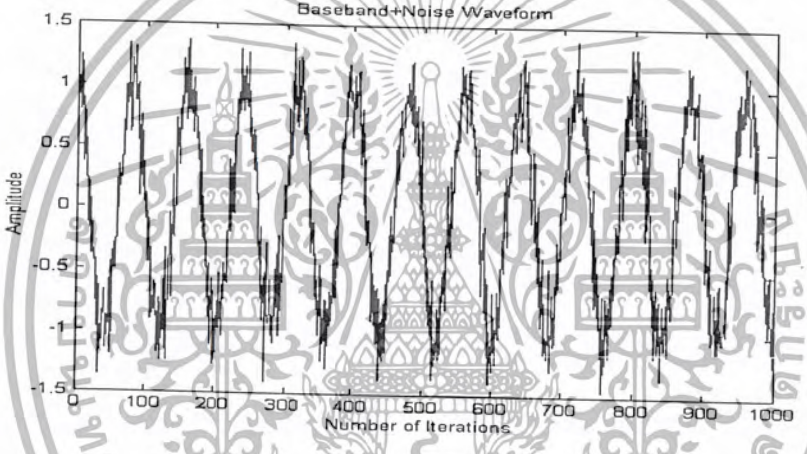


ค) สัญญาณที่กรองได้

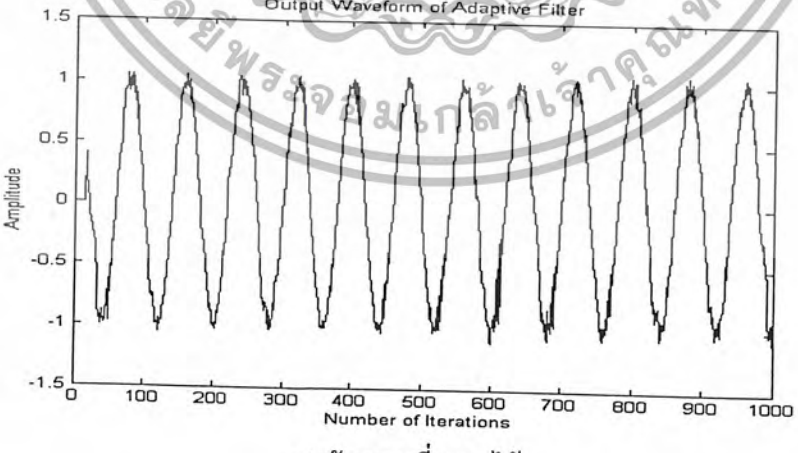
รูปที่ 4.18 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 8 ต่อสัญญาณรบกวนสุ่ม ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.015625



ก) สัญญาณเบสแบนด์

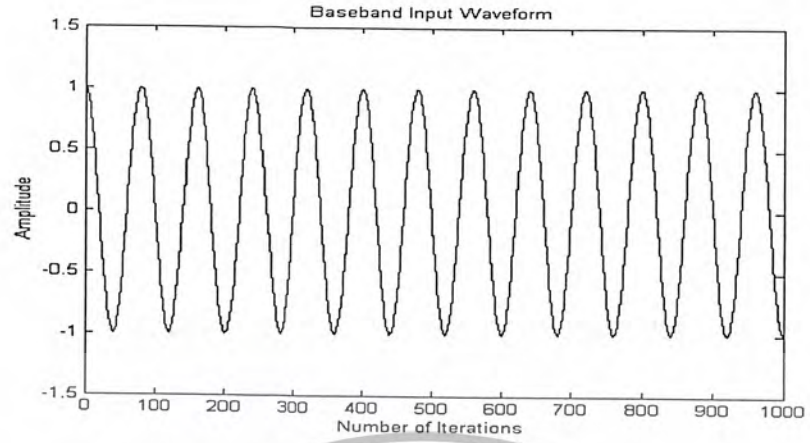


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

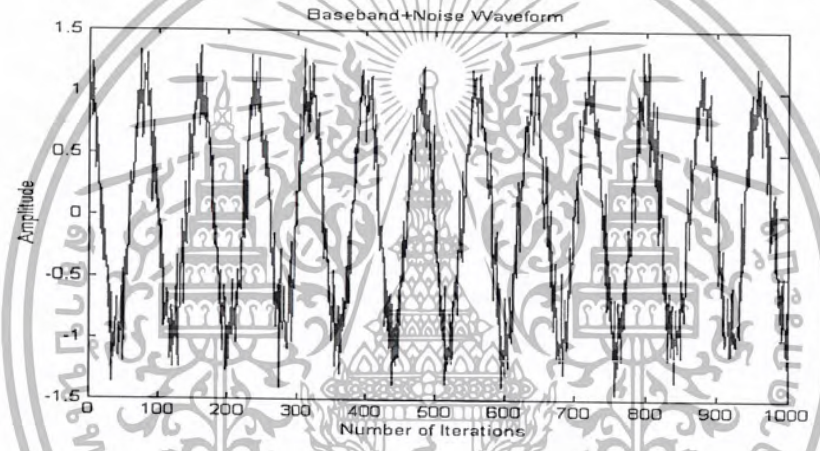


ค) สัญญาณที่กรองได้

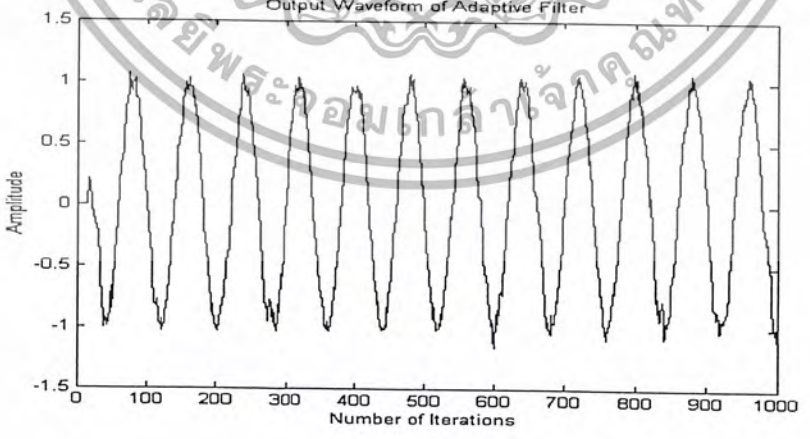
รูปที่ 4.19 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 16 ต่อสัญญาณรบกวนสุ่ม ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.0625



ก) สัญญาณเบสแบนด์

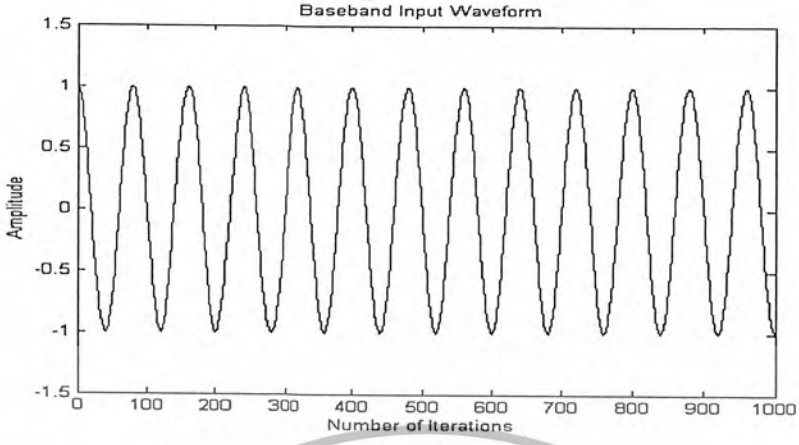


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

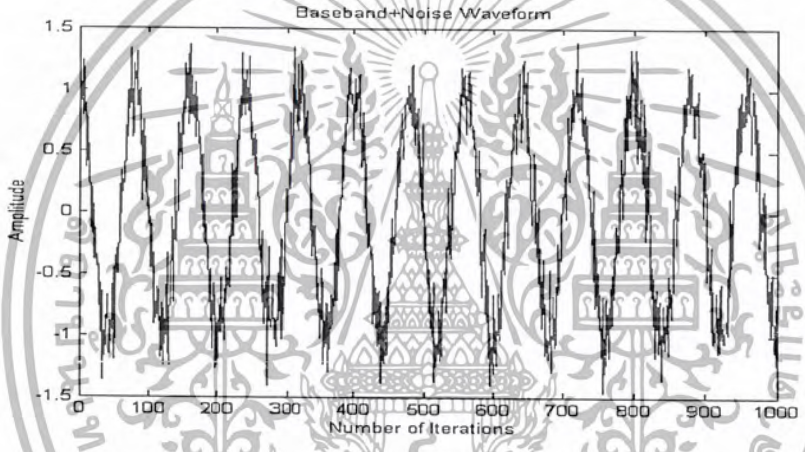


ค) สัญญาณที่กรองได้

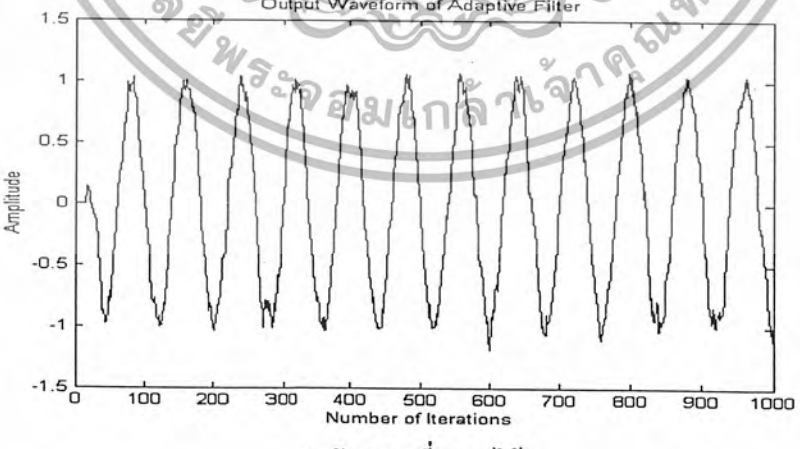
รูปที่ 4.20 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 16 ต่อสัญญาณรบกวนสุ่ม ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.03125



ก) สัญญาณเบสแบนด์



ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

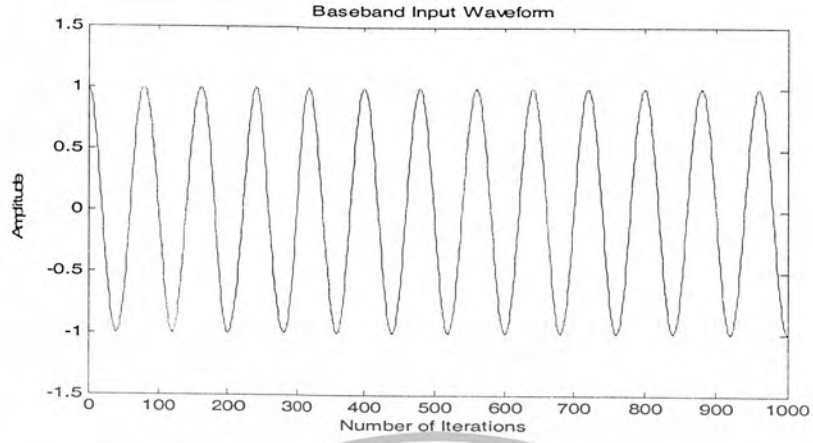


ค) สัญญาณที่กรองได้

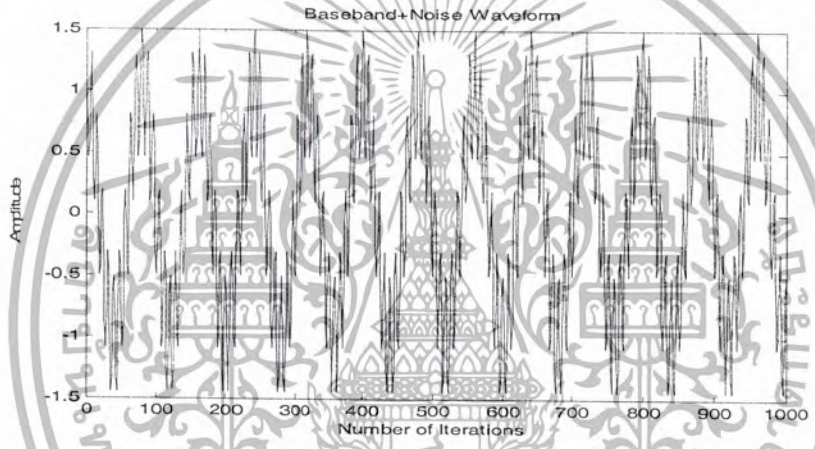
รูปที่ 4.21 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 16 ต่อสัญญาณรบกวนสุ่ม ความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.015625

จากผลการ Simulate ด้วยโปรแกรม MATLAB รูปที่ 4.13 ถึง 4.21 เป็นการป้อนสัญญาณรบกวนสุ่มผสมกับสัญญาณเบสแบนด์ โดยทำการเปลี่ยน order และเปลี่ยนค่า Step size (μ) ด้วย สังเกตุได้ว่ายิ่งจำนวน order ยิ่งสูงจะทำให้การกรองสัญญาณมีประสิทธิภาพดียิ่งขึ้น แต่ต้องเลือกค่า μ ที่เหมาะสมกับจำนวน order ของวงจรกรองด้วย

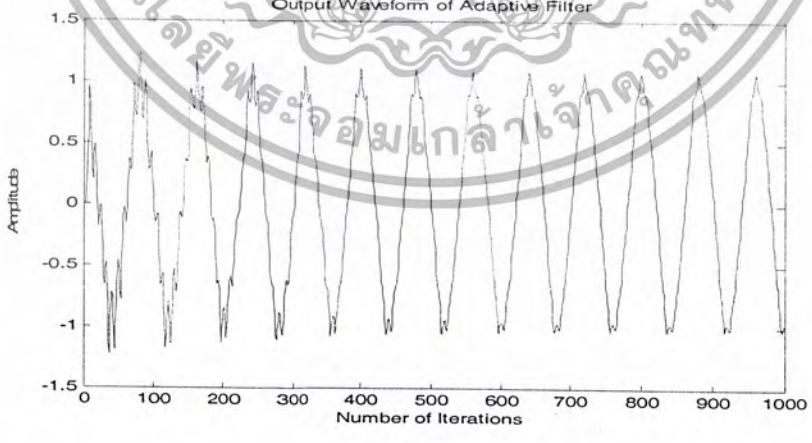
4.2.4 หาผลตอบสนองต่อสัญญาณรบกวนรูปไซน์ความถี่ต่าง ๆ ที่ผสมกับสัญญาณเบสแบนด์



ก) สัญญาณเบสแบนด์

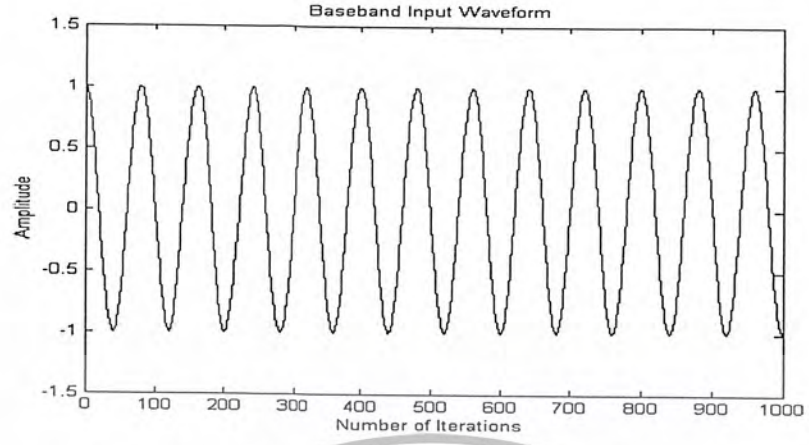


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

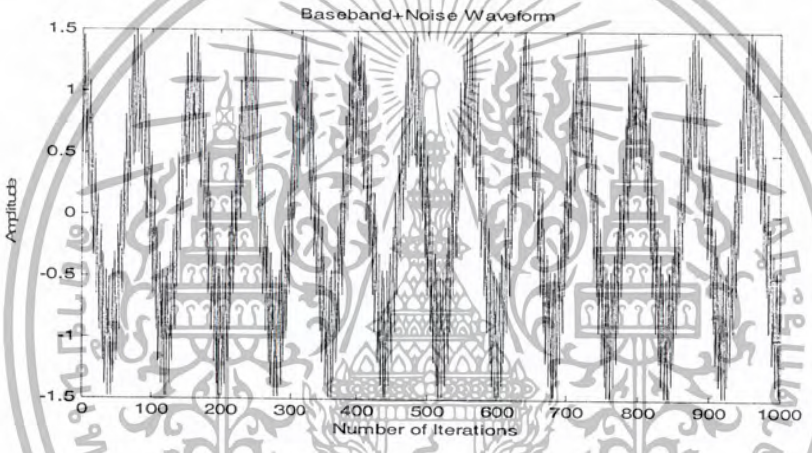


ค) สัญญาณที่กรองได้

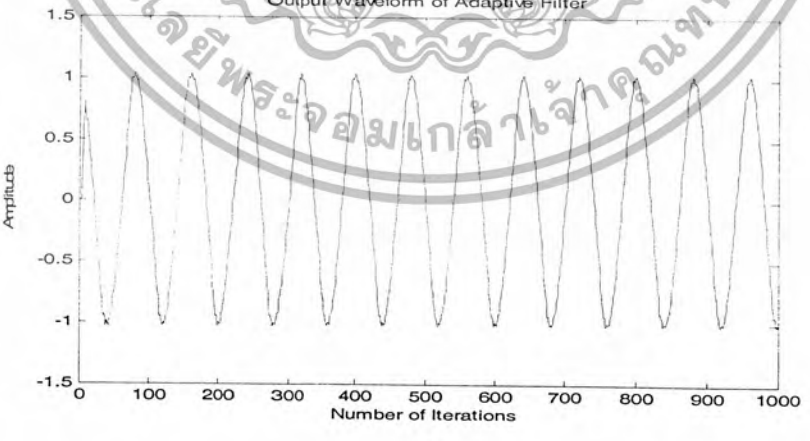
รูปที่ 4.22 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 4 ต่อสัญญาณรบกวนสุ่มความถี่ 1 kHz ผสมกับความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.0625



ก) สัญญาณเบสแบนด์

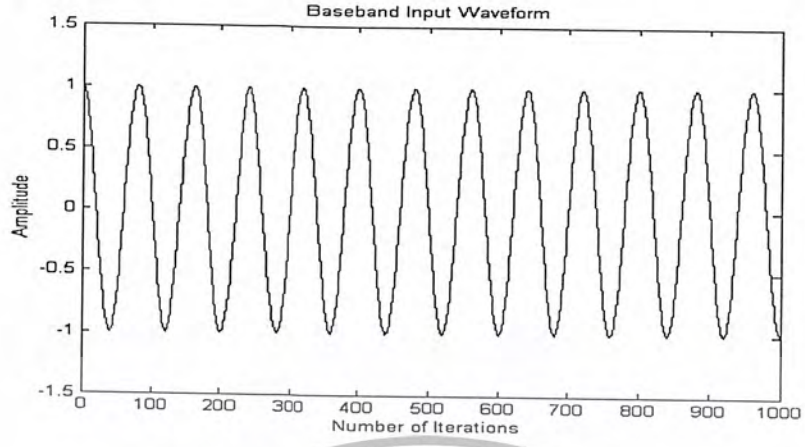


ข) สัญญาณเบสแบนด์ที่ถูกรบกวน

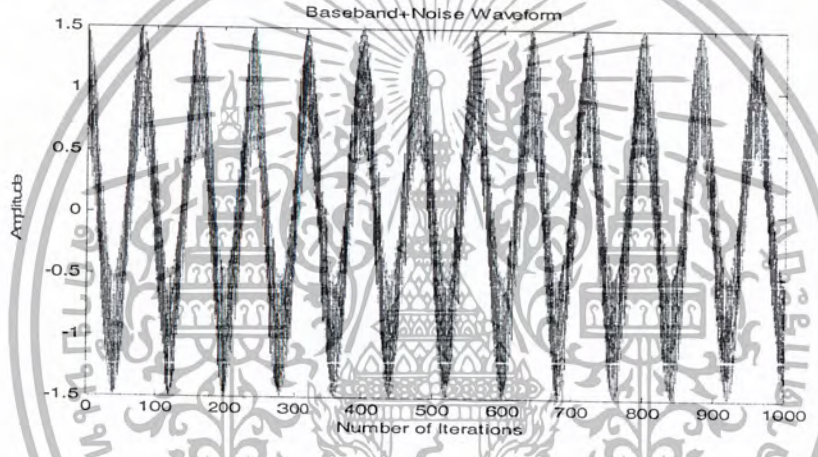


ค) สัญญาณที่กรองได้

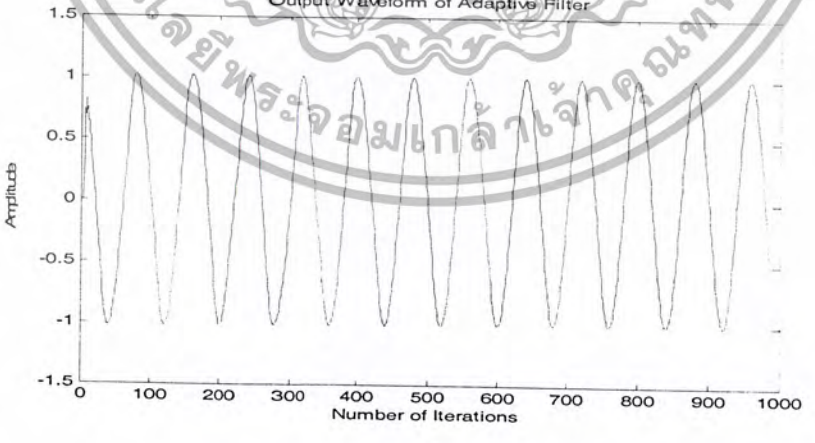
รูปที่ 4.23 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 4 ต่อสัญญาณรบกวนสุ่มความถี่ 10 kHz ผสมกับความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.0625



ก) สัญญาณเบสแบนด์



ข) สัญญาณเบสแบนด์ที่ถูกรบกวน



ค) สัญญาณที่กรองได้

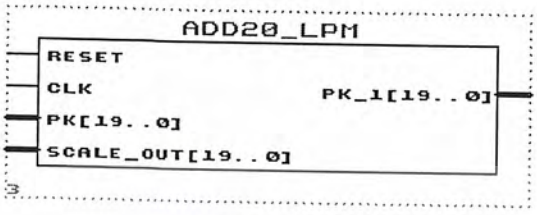
รูปที่ 4.24 ผลตอบสนองของตัวกรองปรับตัวได้แบบเอฟไอาร์อันดับ 4 ต่อสัญญาณรบกวนสุ่มความถี่ 100 kHz ผสมกับความถี่สัญญาณเบสแบนด์ 100 Hz ที่ค่า Step size (μ) = 0.0625

จากผลการ Simulate ด้วยโปรแกรม MATLAB รูปที่ 4.22 ถึง 4.24 เป็นการป้อนสัญญาณรบกวนรูปไซน์ที่ความถี่ 1 kHz, 10 kHz, 100 kHz ผสมกับสัญญาณเบสแบนด์ โดยใช้ order เท่ากับ 4 และ Step size (μ) เท่ากับ 0.0625 พบว่าผลตอบสนองต่อความถี่ต่าง ๆ ของตัวกรองปรับตัวแบบเอฟไออาร์ มีผลการตอบสนองในการกำจัดสัญญาณรบกวนได้ดี และสังเกตได้ว่าจะมีการลู่เข้าที่ดีขึ้น เมื่อสัญญาณรบกวนมีความถี่แตกต่างกับสัญญาณเบสแบนด์มากขึ้น

4.3 การออกแบบวงจรโดยใช้ภาษา VHDL

4.3.1 วงจรบวก (Adder)

เป็นวงจรที่ทำหน้าที่บวกค่าขนาด 20 bit โดยใช้สัญญาณ clk เป็นตัวควบคุมในการบวกและขาสัญญาณ reset ในการกำหนดสัญญาณที่ output ให้เป็นลอจิก "0" ทั้งหมด

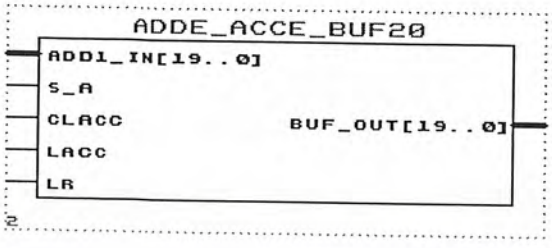


รูปที่ 4.25 แสดง Symbol ของวงจรบวก (Adder)

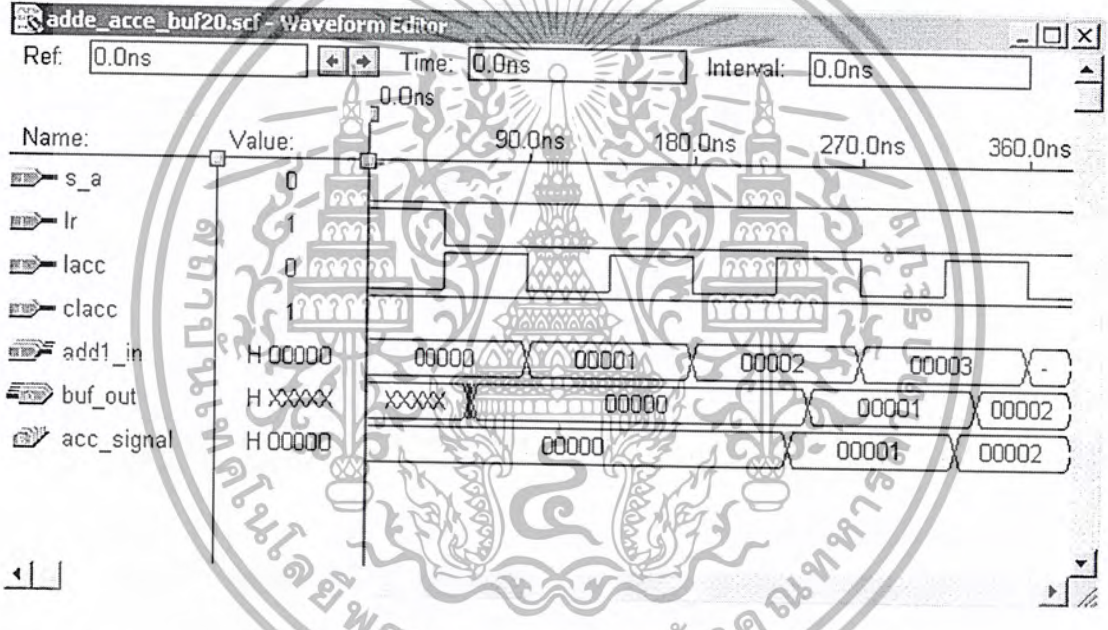
รูปที่ 4.26 แสดง Symbol ของวงจรบวก (Adder)

4.3.2 วงจรสเกลลิ่งแอคคิวมูเลเตอร์ (Scaling Accumulator)

เป็นวงจรที่รวมเอาวงจรบวก แอคคิวมูเลเตอร์ และบัฟเฟอร์เอาไว้ เพื่อทำหน้าที่ในการบวกลบเลขแบบส่วนเติมเต็มสองแทนการคูณ โดยตรง



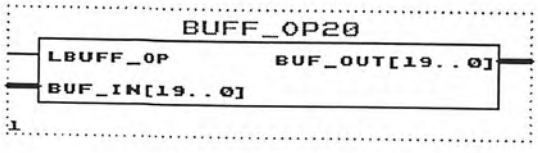
รูปที่ 4.27 แสดง Symbol ของวงจรสเกลลิ่งแอคคิวมูเลเตอร์ (Scaling Accumulator)



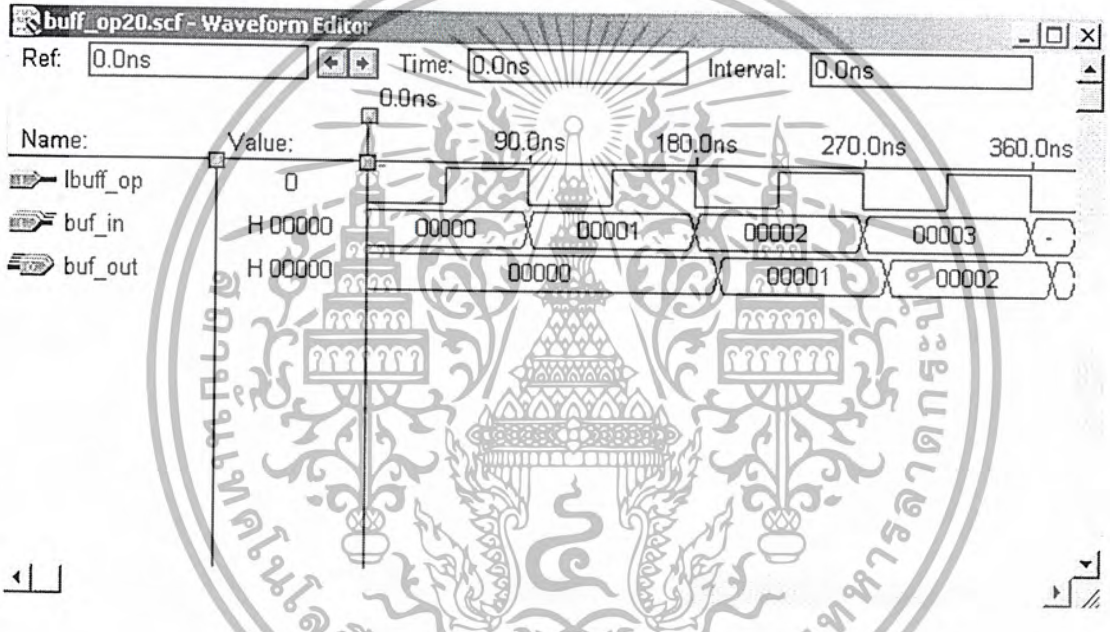
รูปที่ 4.28 แสดงผลการจำลองการทำงานของวงจรรีจิสเตอร์แอคคิวมูเลเตอร์ (Scaling Accumulator)

4.3.3 วงจรบัฟเฟอร์ (Buffer)

เป็นวงจรที่ใช้เป็นตัวกันชนระหว่างวงจรภายในและใช้ในการโหลดค่าออกเอาท์พุท โดยใช้ขาสัญญาณ lbuff_op ในการโหลดค่าออกเอาท์พุท



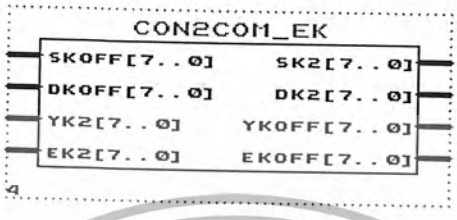
รูปที่ 4.29 แสดง Symbol ของวงจรบัฟเฟอร์ (Buffer)



รูปที่ 4.30 แสดงผลการจำลองการทำงานของวงจรบัฟเฟอร์ (Buffer)

4.4.4 วงจรอินเวอร์สบิทเครื่องหมาย (Convert Sign Bit)

เป็นวงจรที่ใช้ในการกลับบิท MSB (Most Significant Bit) จากลอจิก "0" ให้เป็นลอจิก "1" และจากลอจิก "1" ให้เป็นลอจิก "0" ของอินพุทที่เป็นรูปแบบของสัญญาณออฟเซทให้เป็นสัญญาณแบบส่วนเติมเต็มสองมาใช้ในการประมวลผลและนำเอาท์พุทที่ได้จากการประมวลแล้วซึ่งเป็นเลขส่วนเติมเต็มสองแปลงกลับให้เป็นสัญญาณออฟเซทตามเดิม

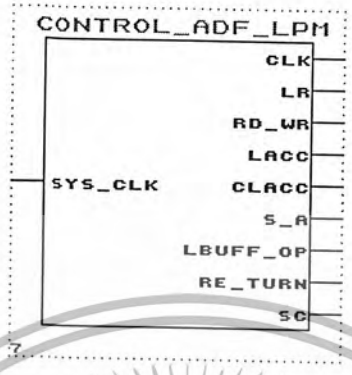


รูปที่ 4.31 แสดง Symbol ของวงจรอินเวอร์สบิทเครื่องหมาย (Convert Sign Bit)

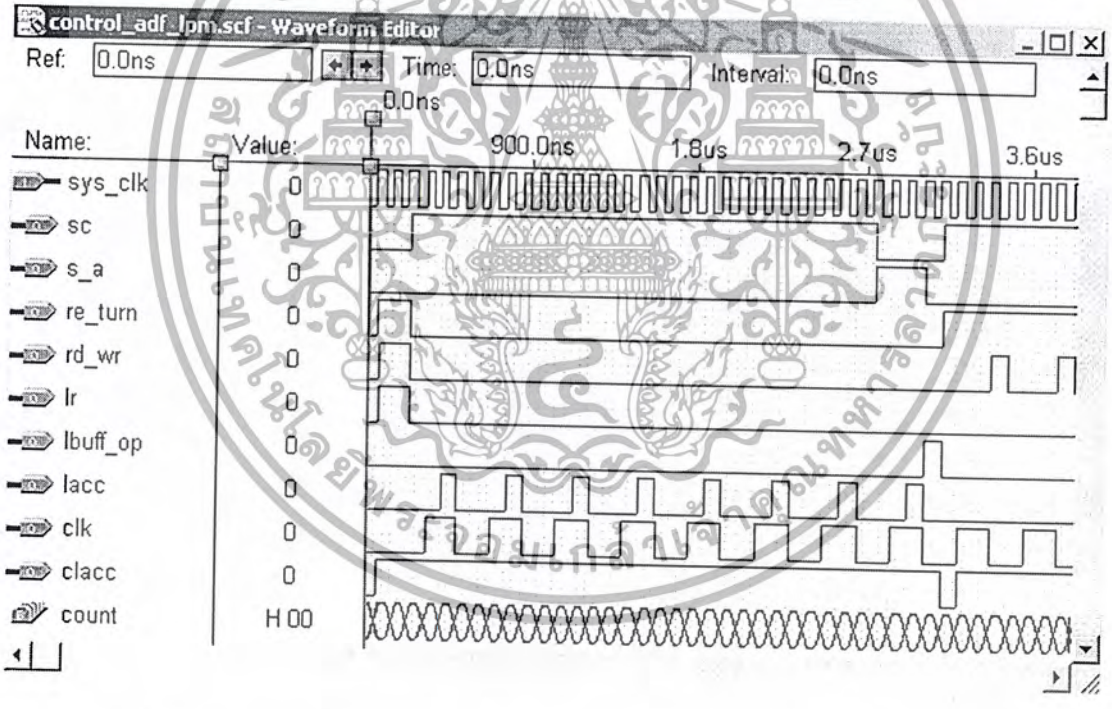
รูปที่ 4.32 แสดงผลการจำลองการทำงานของวงจรอินเวอร์สบิทเครื่องหมาย (Convert Sign Bit)

4.4.5 วงจรสร้างสัญญาณควบคุม (Control Unit)

เป็นวงจรที่กำเนิดสัญญาณควบคุมเพื่อใช้ในการควบคุมยูนิตต่าง ๆ ทั้งหมดโดยจะใช้ clk ของระบบเป็นสัญญาณอ้างอิง และให้สัญญาณควบคุมทั้ง 9 ขา



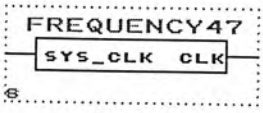
รูปที่ 4.33 แสดง Symbol ของวงจรสร้างสัญญาณควบคุม (Control Unit)



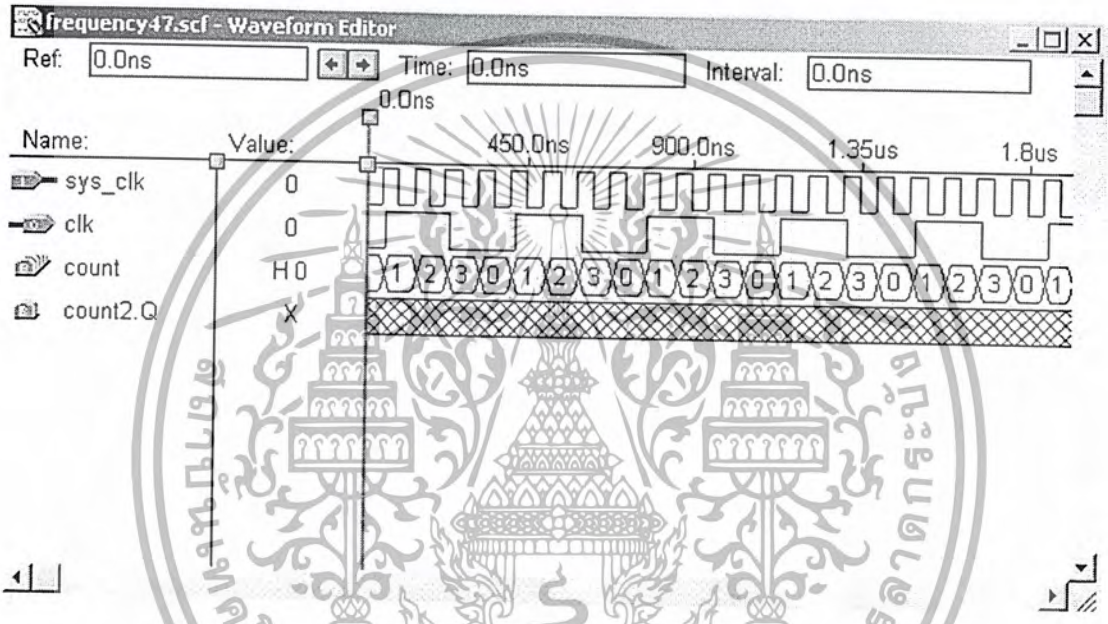
รูปที่ 4.34 แสดงผลการจำลองการทำงานของวงจรสร้างสัญญาณควบคุม (Control Unit)

4.4.6 วงจรหารความถี่ (Frequency Divider)

เป็นวงจรหารความถี่จาก oscillator ให้มีความถี่ที่เหมาะสมเพื่อใช้ในการกำหนดค่าสัญญาณความถี่แซมปลิ้ง (Sampling Frequency)



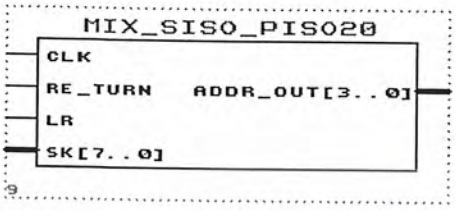
รูปที่ 4.35 แสดง Symbol ของวงจรหารความถี่ (Frequency Divider)



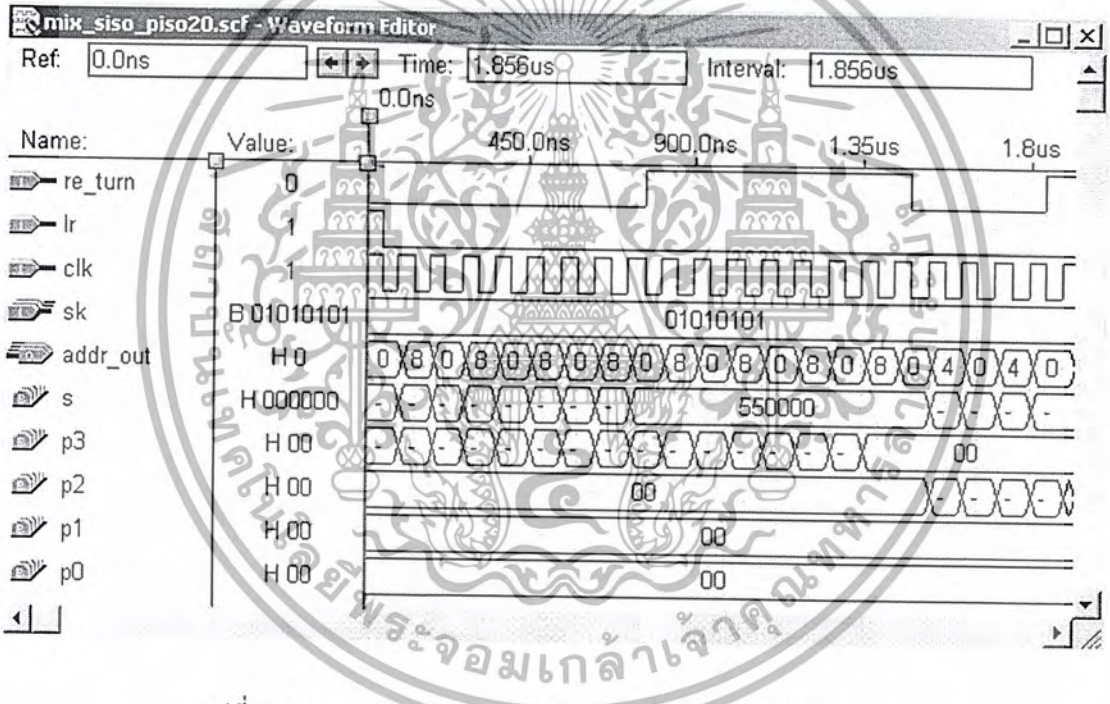
รูปที่ 4.36 แสดงผลการจำลองการทำงานของวงจรหารความถี่ (Frequency Divider)

4.4.7 วงจร SISO และ PISO (Serial In Serial Out and Parallel In Serial Out)

เป็นวงจรที่รวมทั้งตัว shift สัญญาณเข้าด้วยกัน โดยทำหน้าที่โหลดค่าสัญญาณอินพุท และทำการ shift ค่าสัญญาณออกเอาท์พุททีละบิท โดยใช้ขาสัญญาณ clk ในการ shift สัญญาณ ขาสัญญาณ re_turn ใช้ในการควบคุมการนำค่าออกเอาท์พุทแบบวนซ้ำ



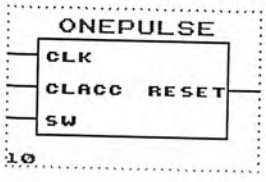
รูปที่ 4.37 แสดง Symbol ของวงจร SISO และ PISO



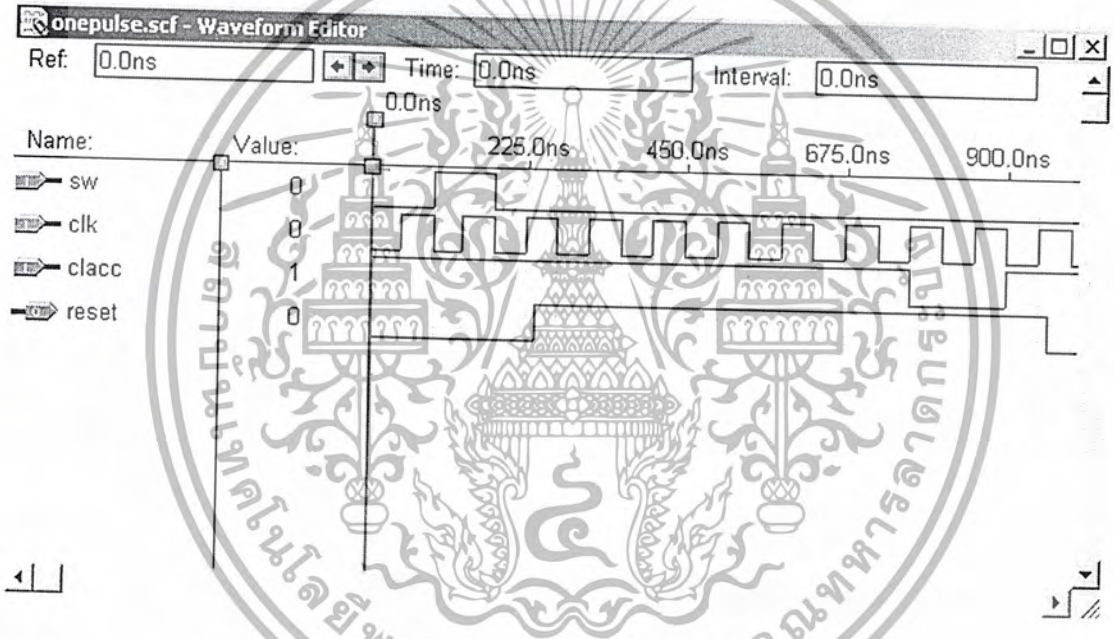
รูปที่ 4.38 แสดงผลการจำลองการทำงานของวงจร SISO และ PISO

4.4.8 วงจรวันพัลส์ (One Pulse)

เป็นวงจรที่ทำหน้าที่ควบคุมสัญญาณ reset จากภายนอก ให้มีจังหวะที่เหมาะสมในการเคลียร์ค่าสัญญาณต่าง ๆ ในวงจร โดยขาสัญญาณ clk กำหนดให้สัญญาณเป็นไปตามต้องการ และขาสัญญาณ sw_reset กับขา clacc จะเป็นตัวกำหนดเงื่อนไขในการกำหนดลอจิกของขา reset ที่เอาท์พุท



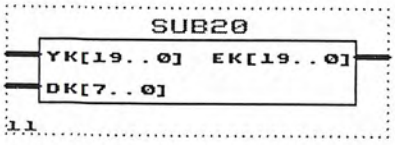
รูปที่ 4.39 แสดง Symbol ของวงจรวันพัลส์ (One Pulse)



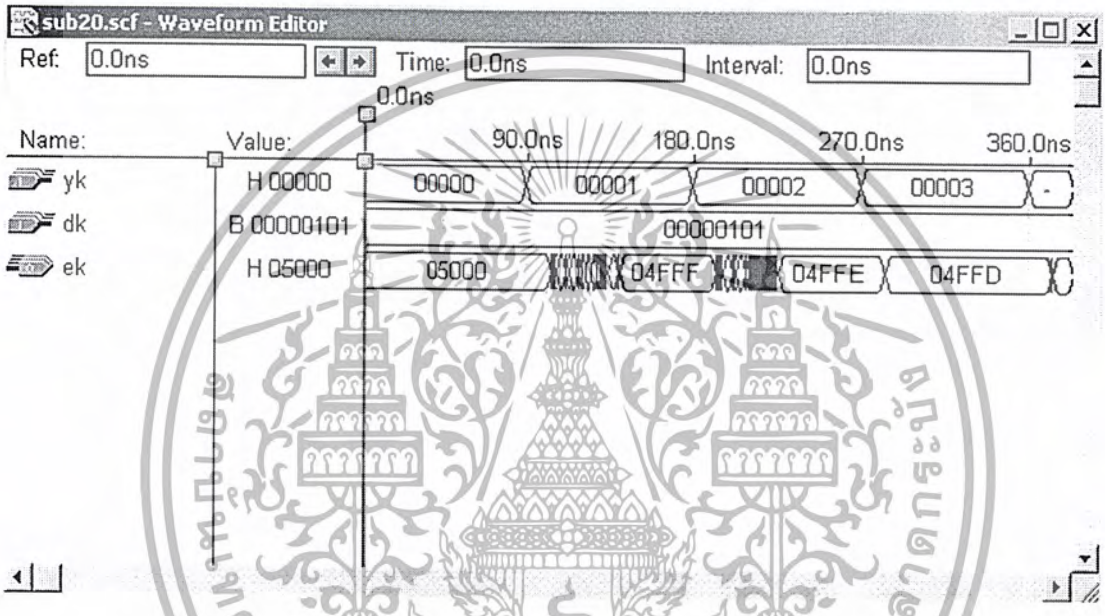
รูปที่ 4.40 แสดงผลการจำลองการทำงานของวงจรวันพัลส์ (One Pulse)

4.4.9 วงจรลบสัญญาณ (Subtractor)

เป็นวงจรที่ทำหน้าที่ลบสัญญาณขนาด 20 bit โดยทำการลบค่าสัญญาณตลอดเวลา



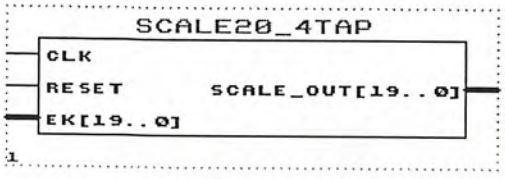
รูปที่ 4.41 แสดง Symbol ของวงจรลบสัญญาณ (Subtractor)



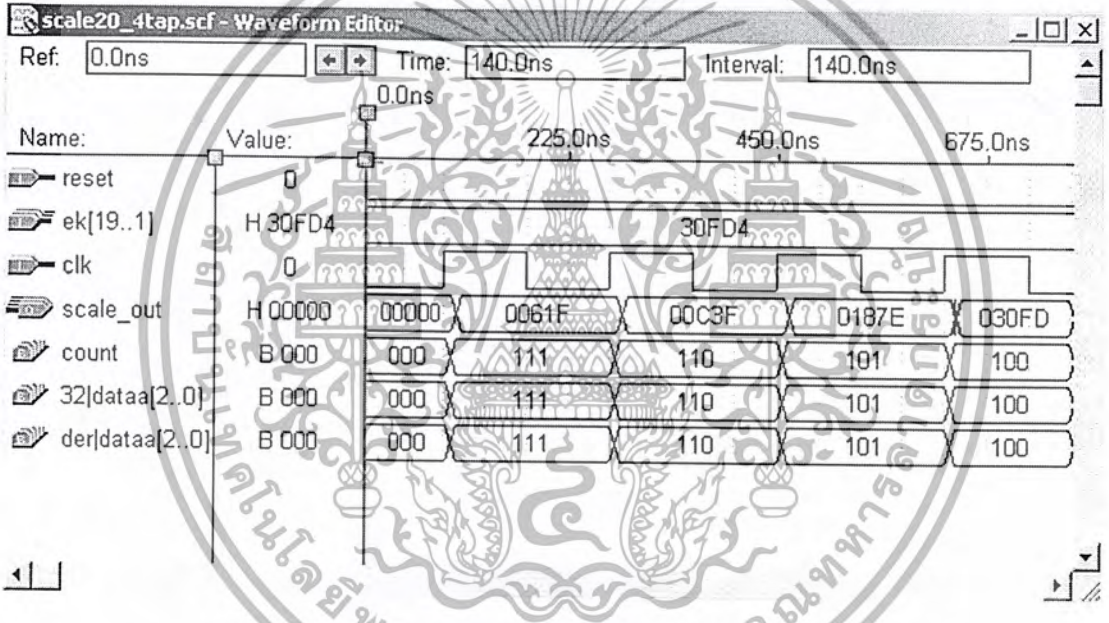
รูปที่ 4.42 แสดงผลการจำลองการทำงานของวงจรลบสัญญาณ (Subtractor)

4.4.10 วงจรปรับน้ำหนัก (Weighting)

เป็นวงจรที่ใช้ในการปรับค่าน้ำหนัก (weight) ซึ่งจะถูกกำหนดโดยแต่ละบิทที่อินพุทชี้ใน RAM ขาสัญญาณ clk จะเป็นตัวกำหนดจังหวะในการปรับน้ำหนัก ส่วนขาสัญญาณ Reset จะใช้ในการเคลียร์ค่าของวงจรนับ เพื่อให้การปรับน้ำหนักตรงกับทฤษฎี



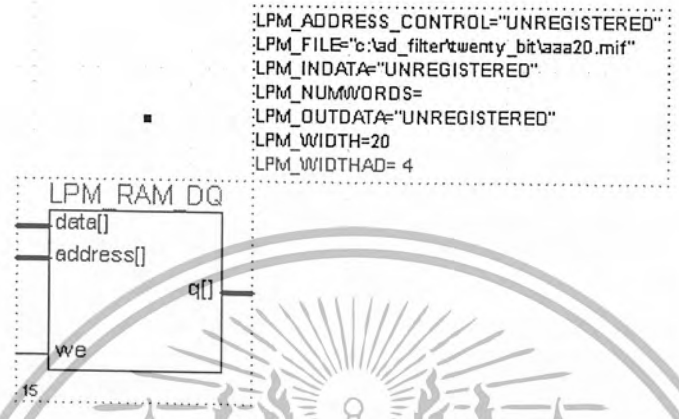
รูปที่ 4.43 แสดง Symbol ของวงจรปรับน้ำหนัก (Weighting)



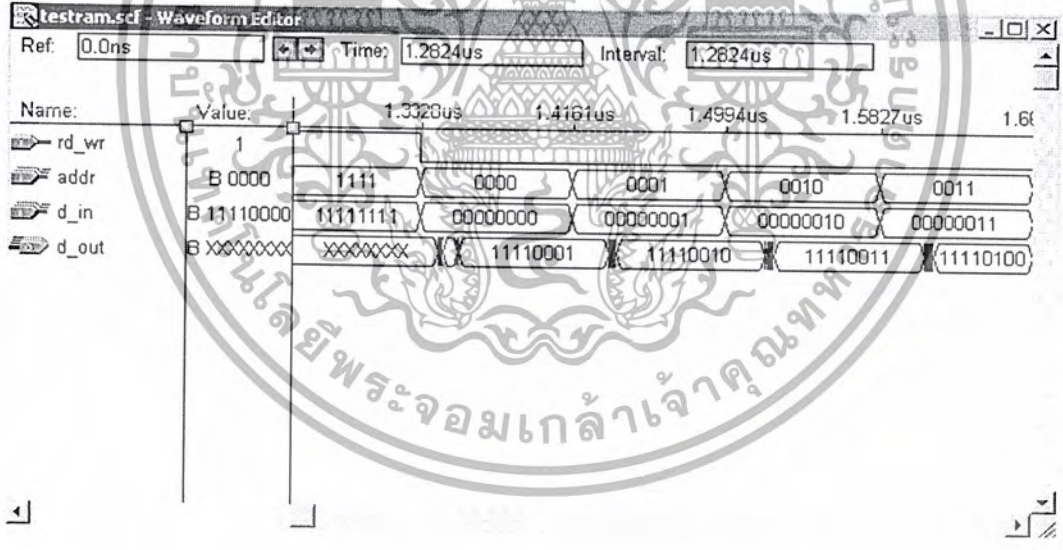
รูปที่ 4.44 แสดงผลการจำลองการทำงานของวงจรปรับน้ำหนัก (Weighting)

4.4.11 RAM (Random Access Memory)

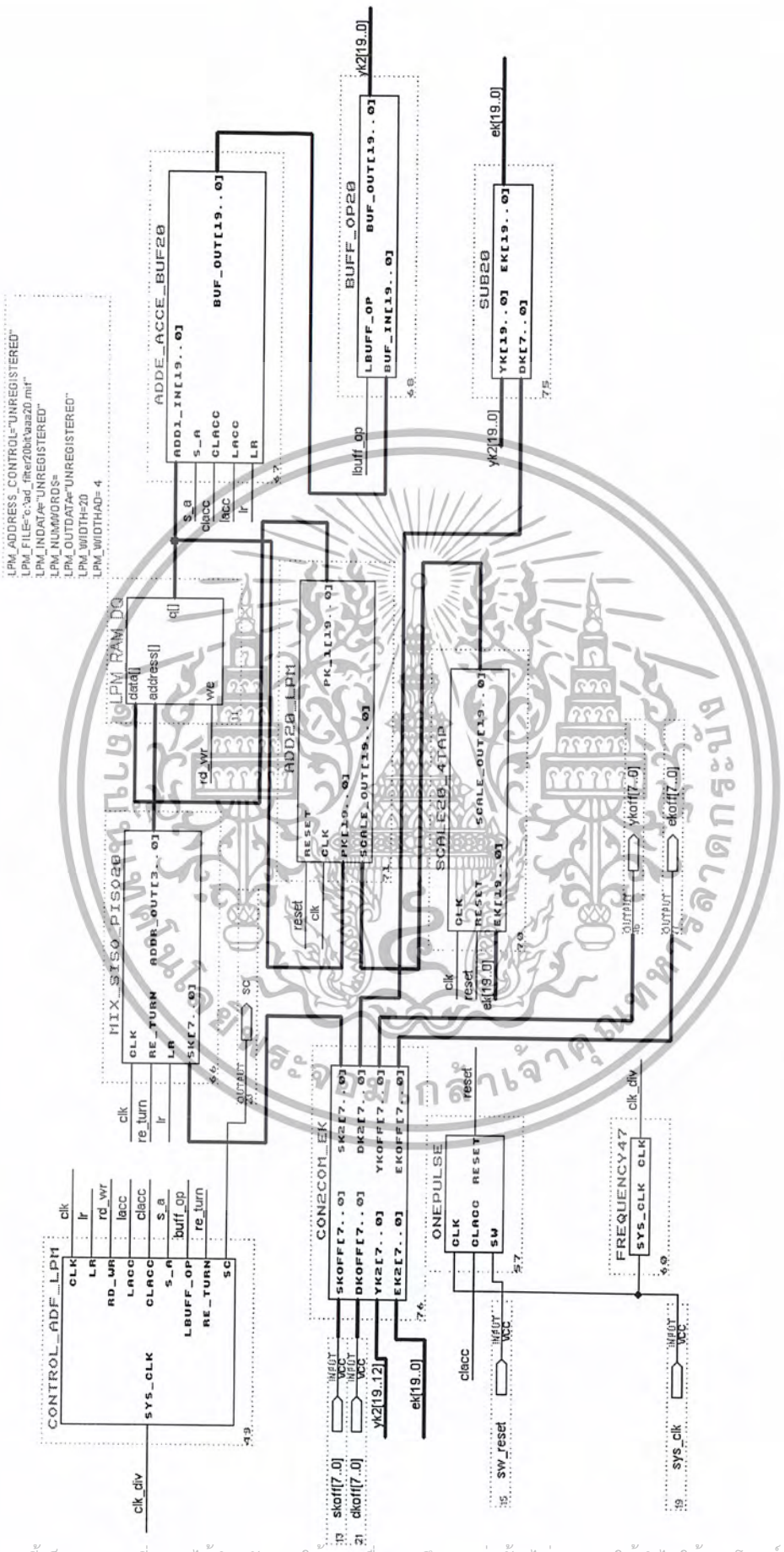
เป็นตัวที่ทำหน้าที่เก็บค่าสัมประสิทธิ์ของตัวกรองแบบปรับตัวได้ โดยที่ถ้าสัญญาณ RD/WR = "0" จะเป็นการอ่านค่าสัมประสิทธิ์ที่อยู่ใน RAM ออกไปใช้งาน และถ้าสัญญาณ RD/WR = "1" จะเป็นการเขียนค่าสัมประสิทธิ์ใหม่เข้าไปเก็บไว้ใน RAM



รูปที่ 4.45 แสดง Symbol ของ RAM



รูปที่ 4.46 แสดงผลการจำลองการทำงานของ RAM

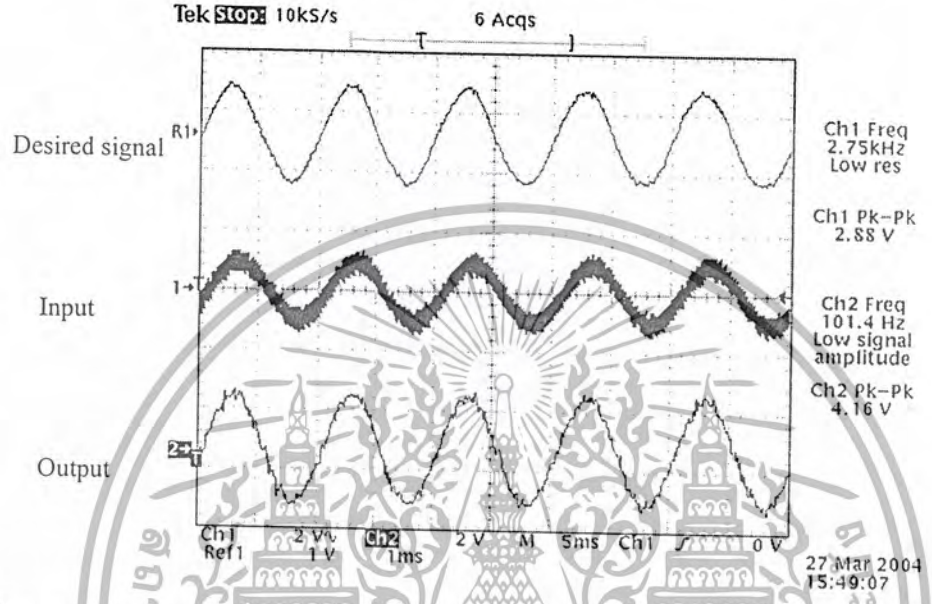


รูปที่ 4.47 แสดงวงจรกรองสัญญาณแบบปรับตัวได้ที่สร้างด้วยภาษา VHDL

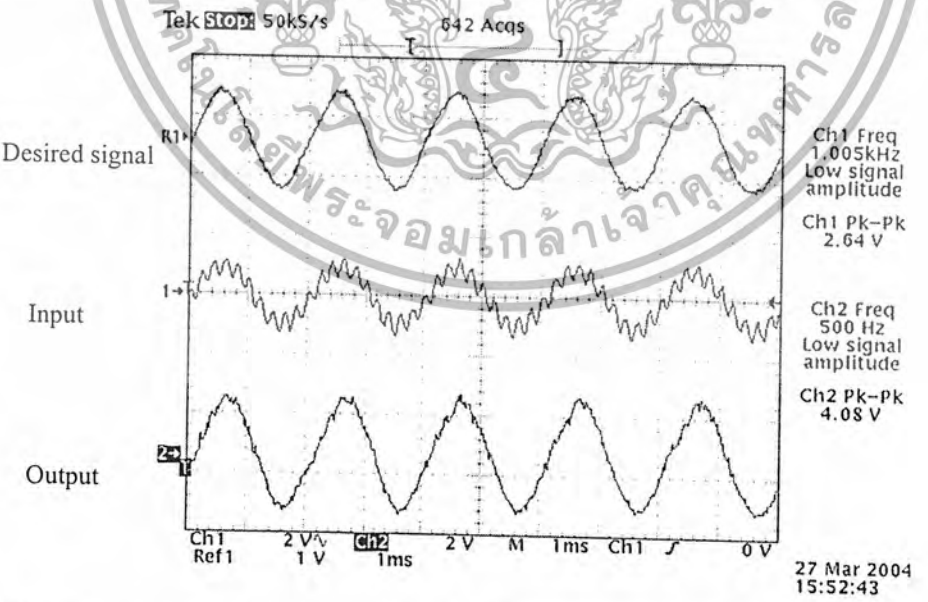
4.4 ผลตอบสนองของตัวกรองแบบปรับตัวได้ที่เขียนด้วยภาษา VHDL

4.4.1 ผลตอบสนองของตัวกรองแบบปรับตัวได้ที่ค่า Step size (μ) ต่าง ๆ

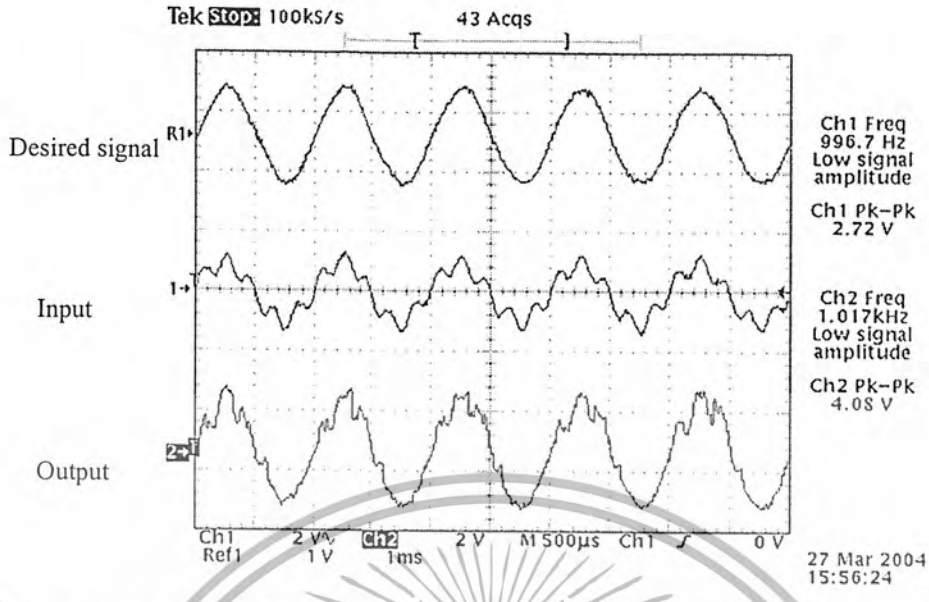
ทำการป้อนสัญญาณอินพุทซึ่งเป็นสัญญาณเบสแบนด์รูปไซน์ความถี่ต่าง ๆ คือ 100 Hz, 500 Hz, 1kHz ขนาด 1 Volt ที่ถูกรบกวนด้วยสัญญาณรบกวนรูปไซน์ความถี่ 5 kHz ขนาด 0.6 Volt ใช้ตัวกรองปรับตัวแบบได้อันดับ 4 ที่ค่า Step size (μ) ต่าง ๆ คือ 0.25, 0.125, 0.0625 ตามลำดับ



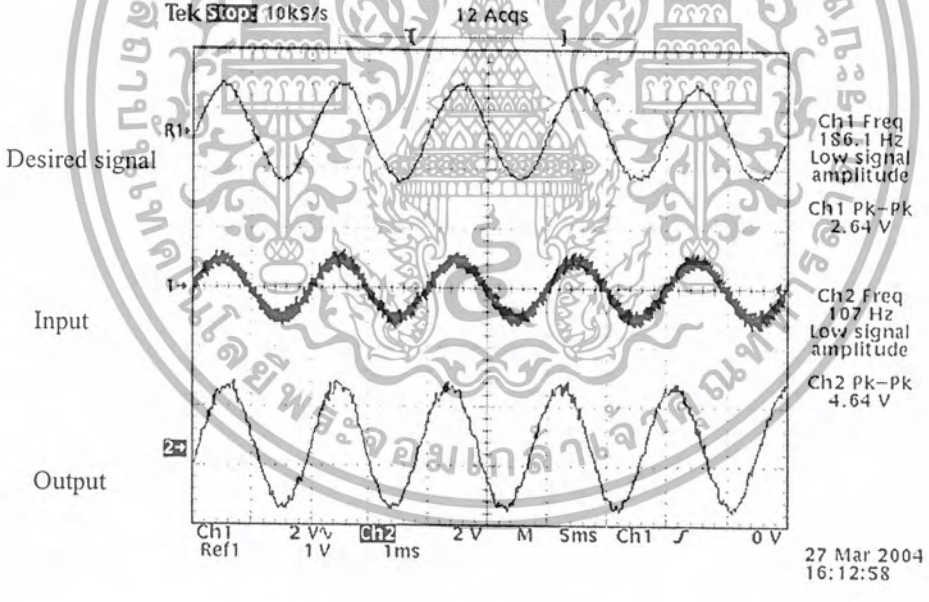
รูปที่ 4.48 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 100 Hz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ที่ค่า Step size (μ) = 0.25



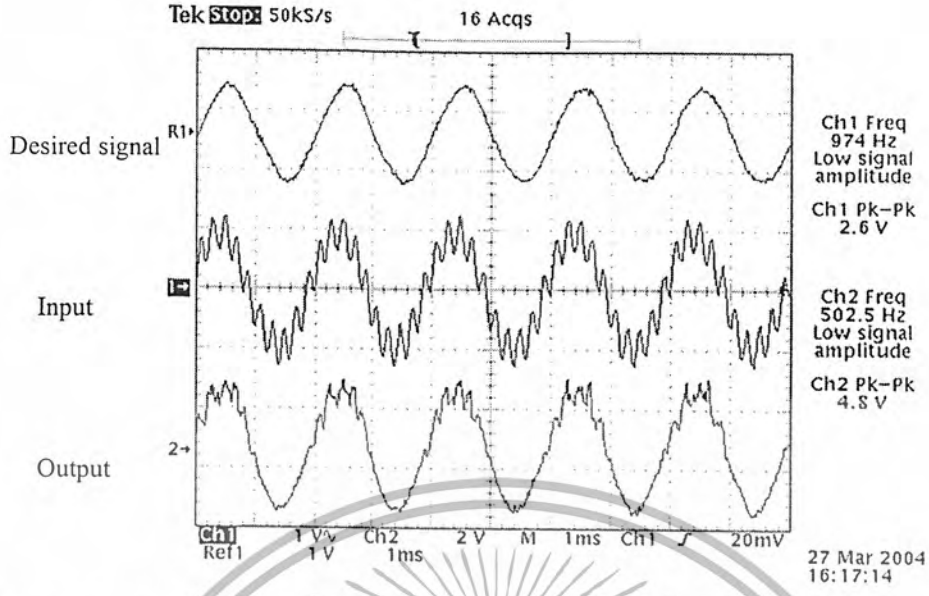
รูปที่ 4.49 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 500 Hz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ที่ค่า Step size (μ) = 0.25



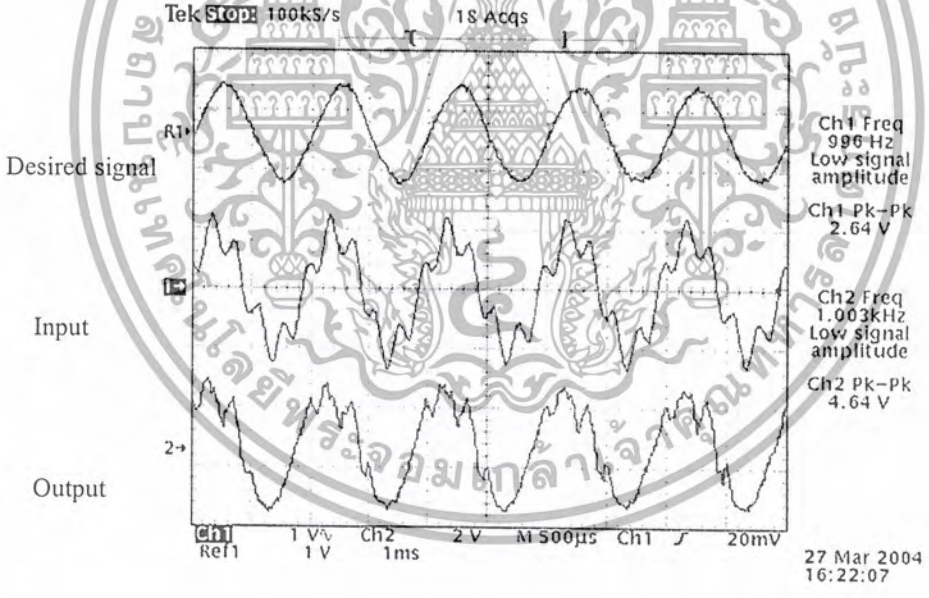
รูปที่ 4.50 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 1 kHz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ที่ค่า Step size (μ) = 0.25



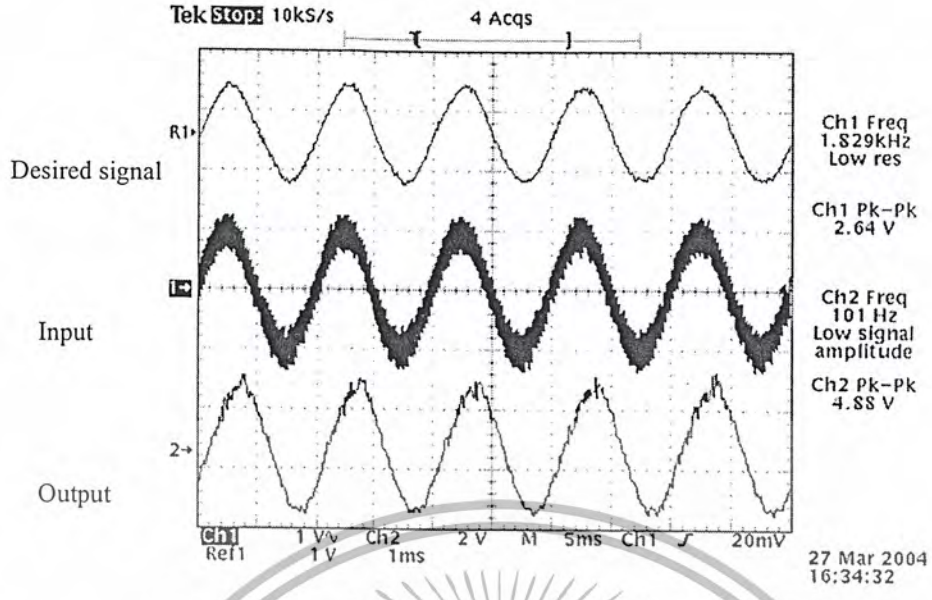
รูปที่ 4.51 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 100 Hz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ที่ค่า Step size (μ) = 0.125



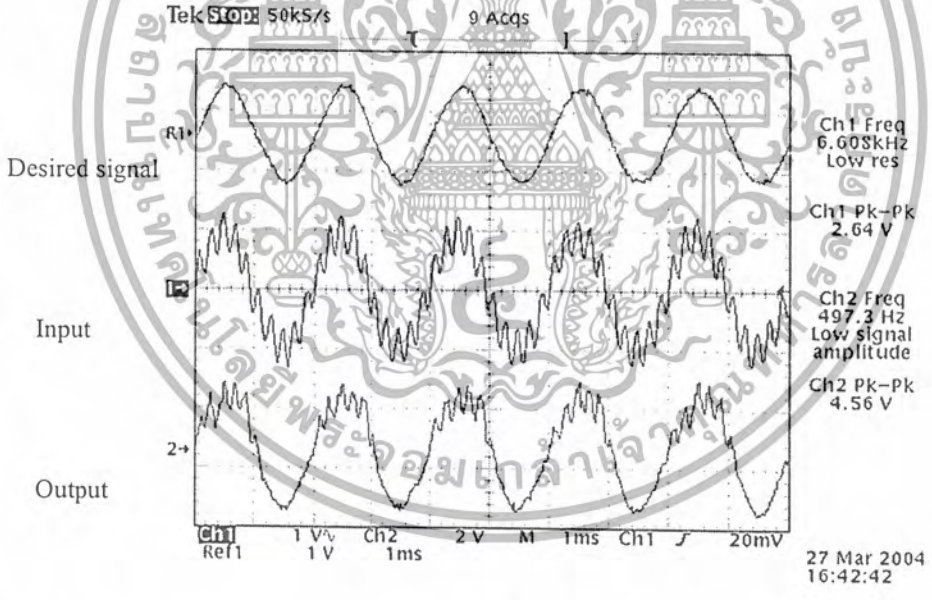
รูปที่ 4.52 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 500 Hz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ที่ค่า Step size (μ) = 0.125



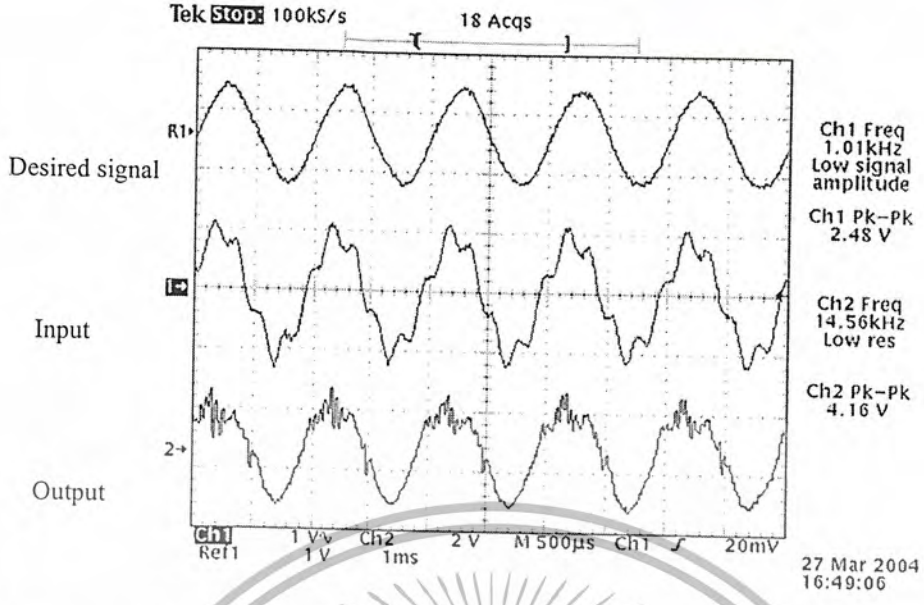
รูปที่ 4.53 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 1 kHz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ที่ค่า Step size (μ) = 0.125



รูปที่ 4.54 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 100 Hz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ที่ค่า Step size (μ) = 0.0625



รูปที่ 4.55 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 500 Hz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ที่ค่า Step size (μ) = 0.0625

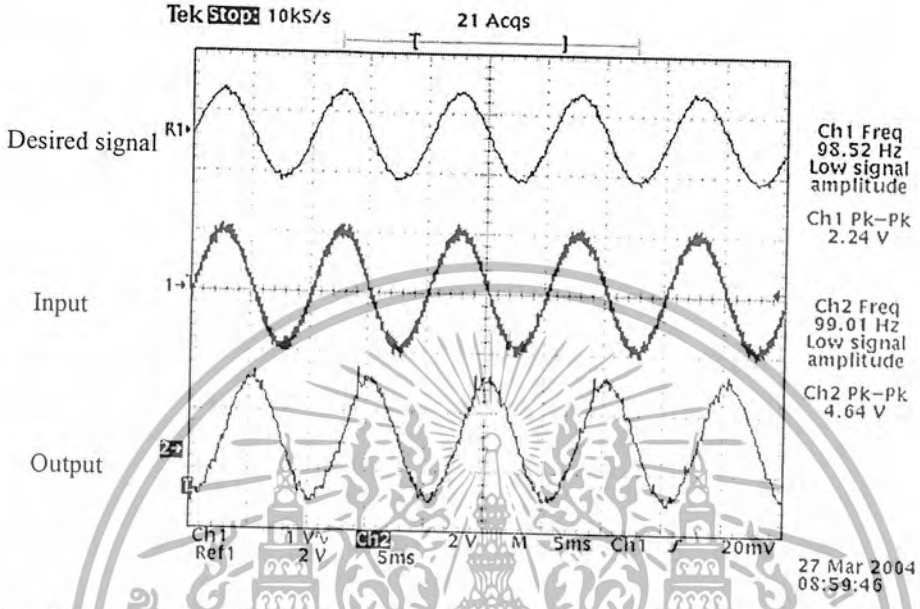


รูปที่ 4.56 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 1 kHz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ที่ค่า Step size (μ) = 0.0625

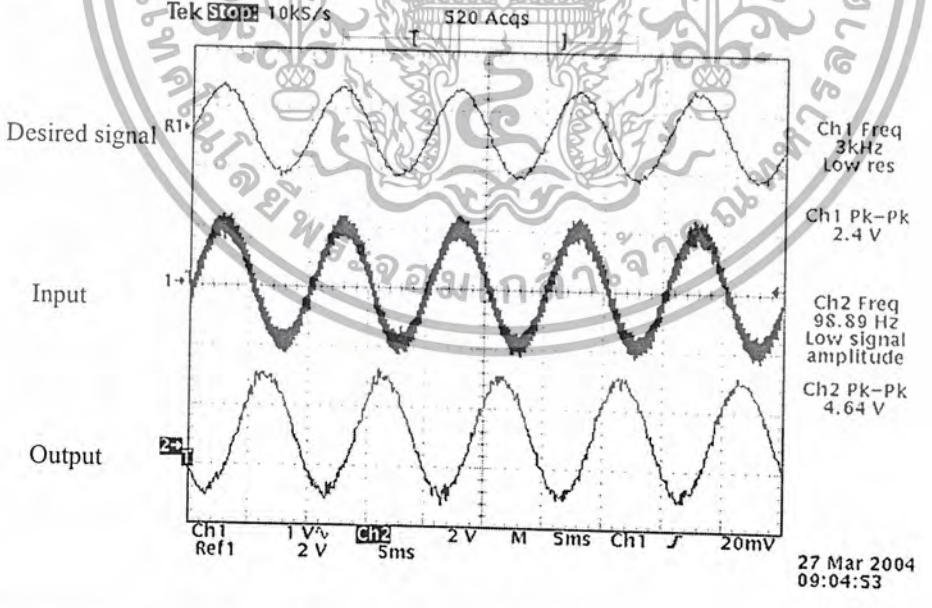
จากรูปที่ 4.48 ถึง 4.56 สามารถสังเกตได้ว่าค่าสเตปที่เหมาะสมกับสัญญาณรูปแบบหนึ่งแต่อาจจะไม่เหมาะกับสัญญาณอีกรูปแบบหนึ่งก็ได้ ค่าสเตปที่เหมาะสมโดยจะพิจารณาจากผลการทดลองสัญญาณในช่วง 100 Hz ถึง 1 kHz ว่าคุณภาพในการกรองสัญญาณที่ค่าสเตปค่าไหนดีกว่ากัน ค่าสเตปที่เหมาะสมหรือไม่เหมาะสม สังเกตุได้จากจากผลการทดลองคือ แอมปลิจูดและเฟส ของทั้งสัญญาณ desired signal อีกทั้งจำนวนอันดับของวงจรกรองก็มีผลด้วยเช่นกัน ถ้าขนาดของสเตปมีมากเกินไปอาจทำให้ไม่สามารถกรองสัญญาณสัญญาณได้เลย

4.4.2 ผลตอบสนองต่อสัญญาณรบกวนรูปไซน์ที่ผสมกับสัญญาณเบสแบนด์

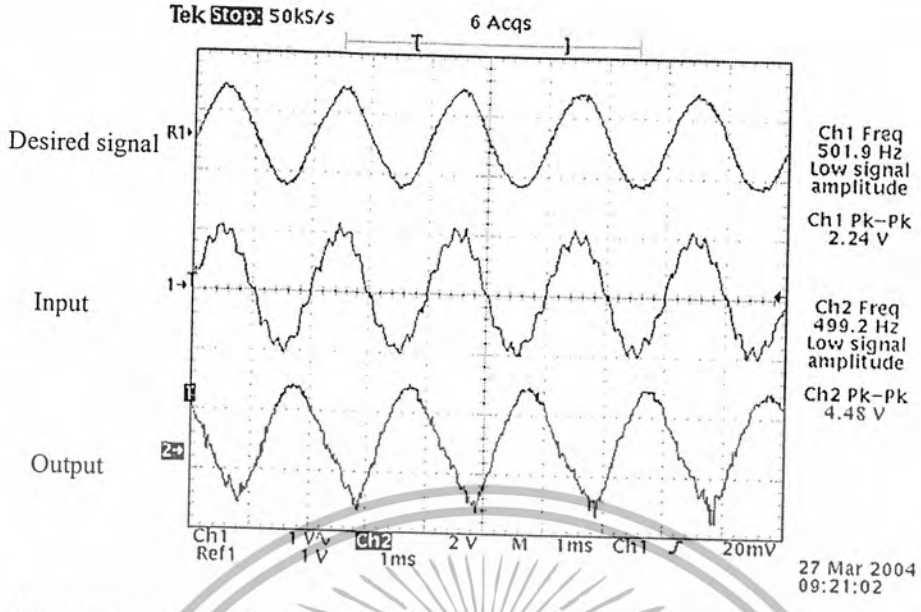
ทำการป้อนสัญญาณอินพุทซึ่งเป็นสัญญาณเบสแบนด์รูปไซน์ความถี่ต่าง ๆ คือ 100 Hz, 500 Hz, 1kHz ขนาด 1 Volt ที่ถูกรบกวนด้วยสัญญาณรบกวนรูปไซน์ความถี่ 5 kHz ขนาด 0.2 Volt กับ 0.4 Volt ใช้ตัวกรองปรับตัวแบบได้อันดับ 4 โดยเลือกใช้ค่า Step size (μ) = 0.25



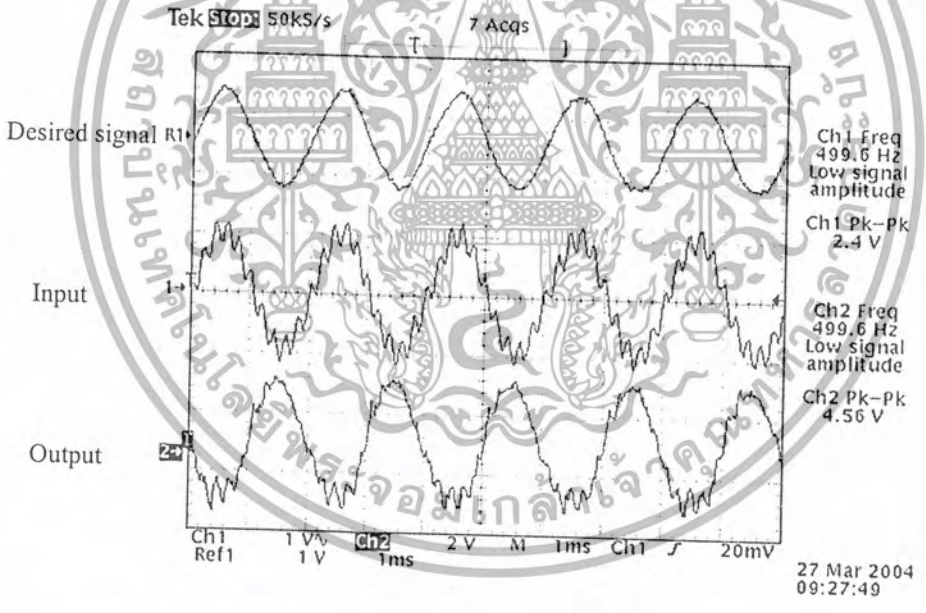
รูปที่ 4.57 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 100 Hz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ขนาด 0.2 Volt



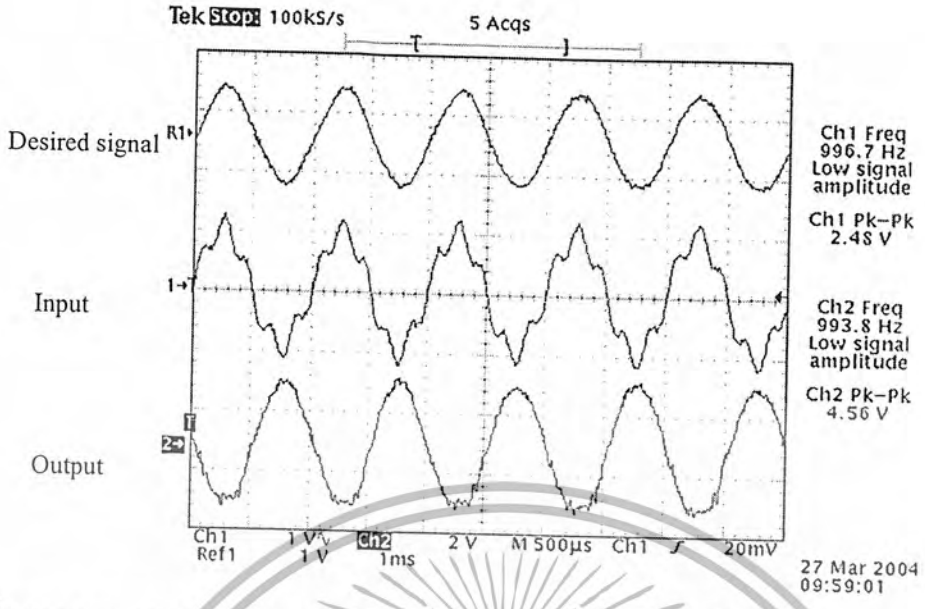
รูปที่ 4.58 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 100 Hz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ขนาด 0.4 Volt



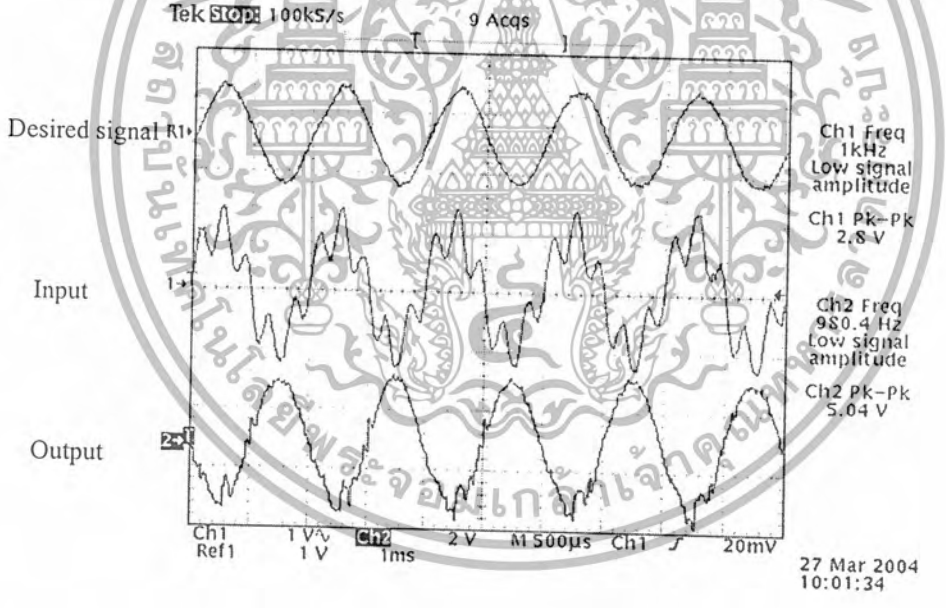
รูปที่ 4.59 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 500 Hz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ขนาด 0.2 Volt



รูปที่ 4.60 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 500 Hz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ขนาด 0.4 Volt



รูปที่ 4.61 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 1 kHz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ขนาด 0.2 Volt

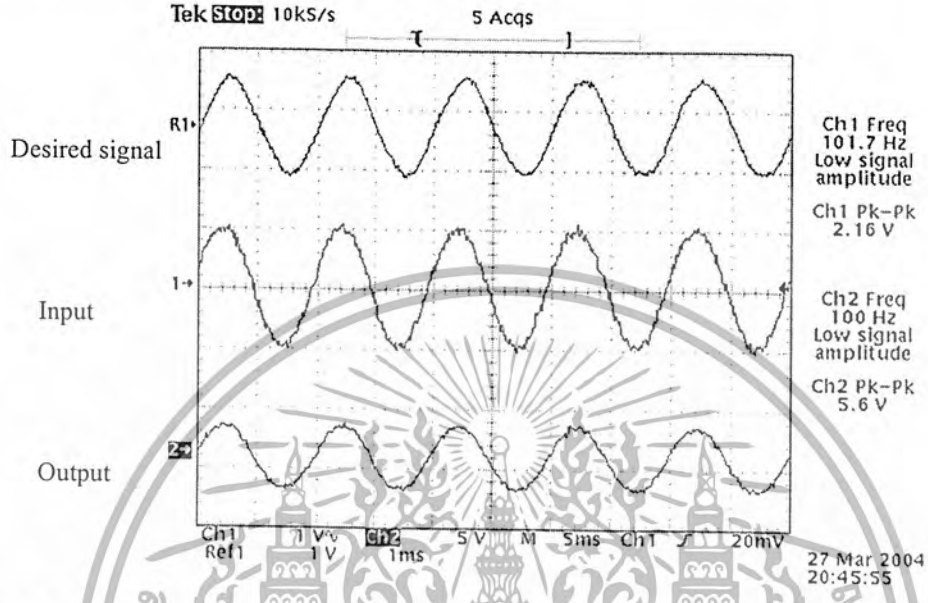


รูปที่ 4.62 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 1 kHz ซึ่งถูกรบกวนด้วยสัญญาณรูปไซน์ความถี่ 5 kHz ขนาด 0.4 Volt

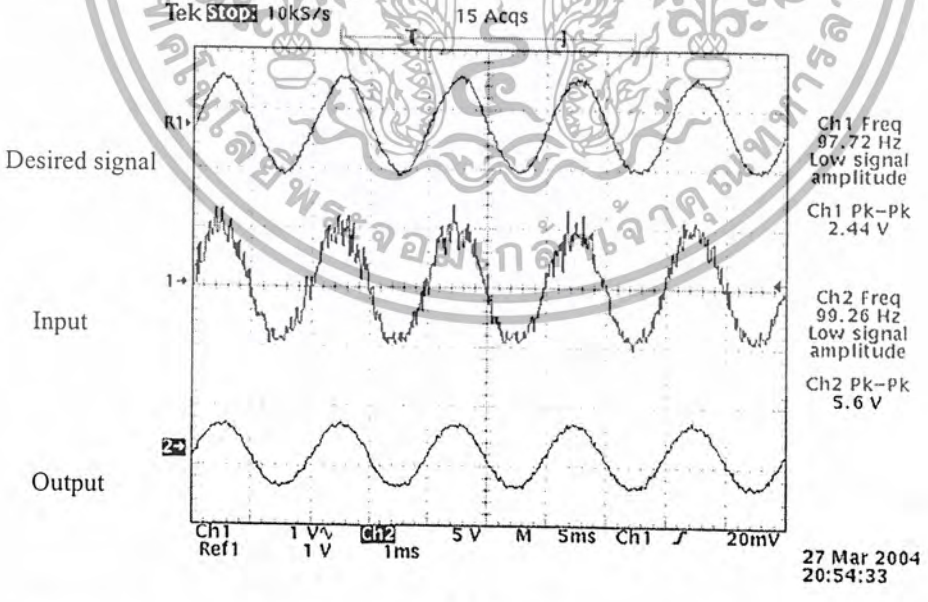
จากรูปที่ 4.57 ถึง 4.62 พบว่าตัวกรองสัญญาณแบบปรับตัวได้อันดับที่ 4 มีขนาดสเตปเท่ากับ 0.25 มีผลตอบสนองค่อนข้างดี จากการทดลองสังเกตได้ว่ายิ่งความถี่สัญญาณเบสแบนด์มีความเกี่ยวพัน (Correlation) กับสัญญาณรบกวนมาก ก็จะยิ่งทำให้คุณภาพของสัญญาณแย่ลง แต่ถ้าสัญญาณเบสแบนด์มีความเกี่ยวพัน (Correlation) กับสัญญาณรบกวนน้อยลงก็จะทำให้คุณภาพของสัญญาณดีขึ้น อีกตัวแปรหนึ่งที่สำคัญคือ แอมปลิจูดของสัญญาณรบกวน เมื่อสัญญาณรบกวนมีแอมปลิจูดมากขึ้น ผลลัพธ์ของการกรองสัญญาณจะมีประสิทธิภาพแย่ลง กว่าการมีแอมปลิจูดของสัญญาณน้อยกว่า

4.4.3 ผลตอบสนองต่อสัญญาณรบกวน (Noise) ที่ผสมกับสัญญาณเบสแบนด์

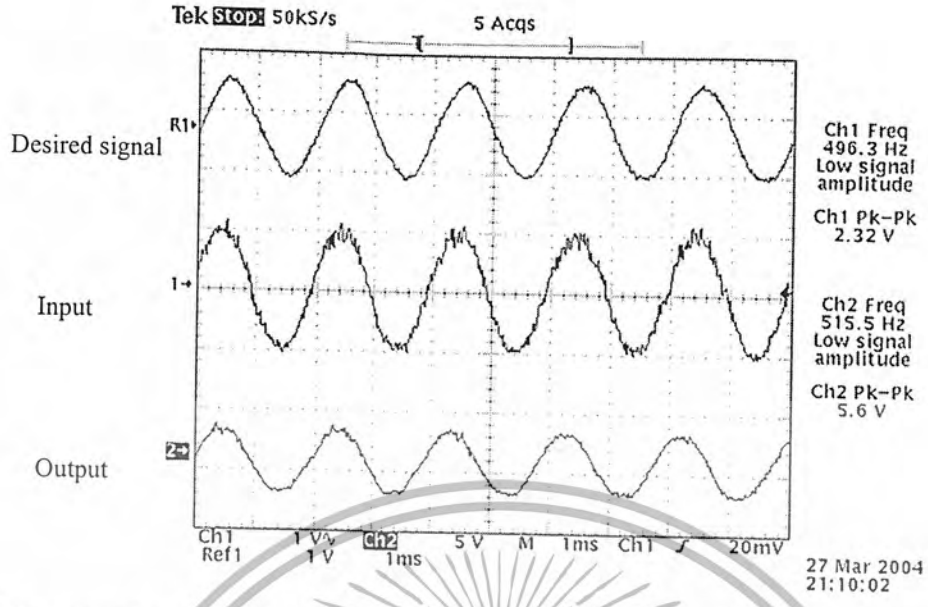
ทำการป้อนสัญญาณอินพุทซึ่งเป็นสัญญาณเบสแบนด์รูปไซน์ความถี่ต่าง ๆ คือ 100 Hz, 500 Hz, 1kHz ขนาด 1 Volt ที่ถูกรบกวนด้วยสัญญาณรบกวน (Noise) ขนาด 1 Volt กับ 2 Volt ใช้ตัวกรองปรับตัวแบบได้อันดับ 4 โดยเลือกใช้ค่า Step size (μ) = 0.25



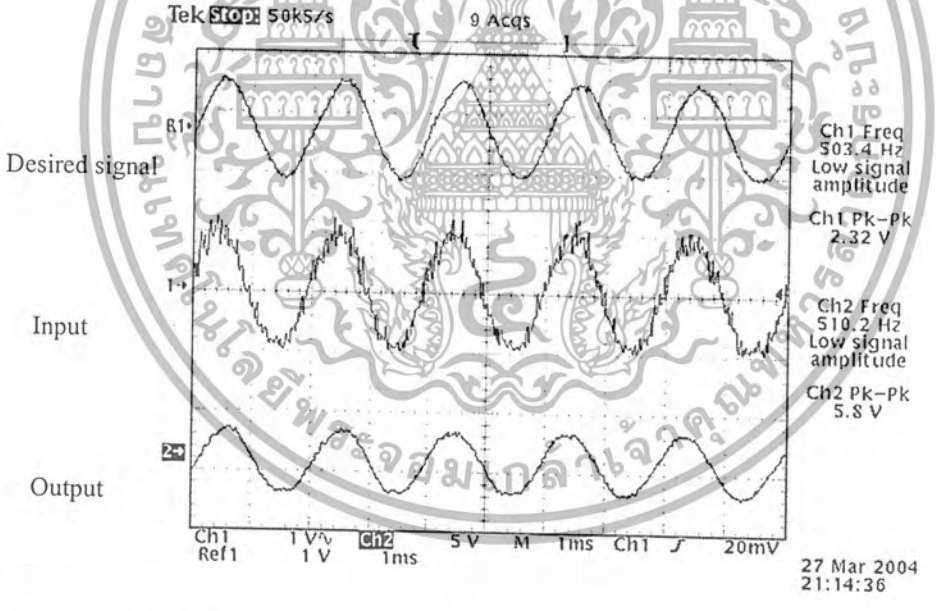
รูปที่ 4.63 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 100 Hz ซึ่งถูกรบกวนด้วยสัญญาณรบกวน (Noise) ขนาด 1 Volt



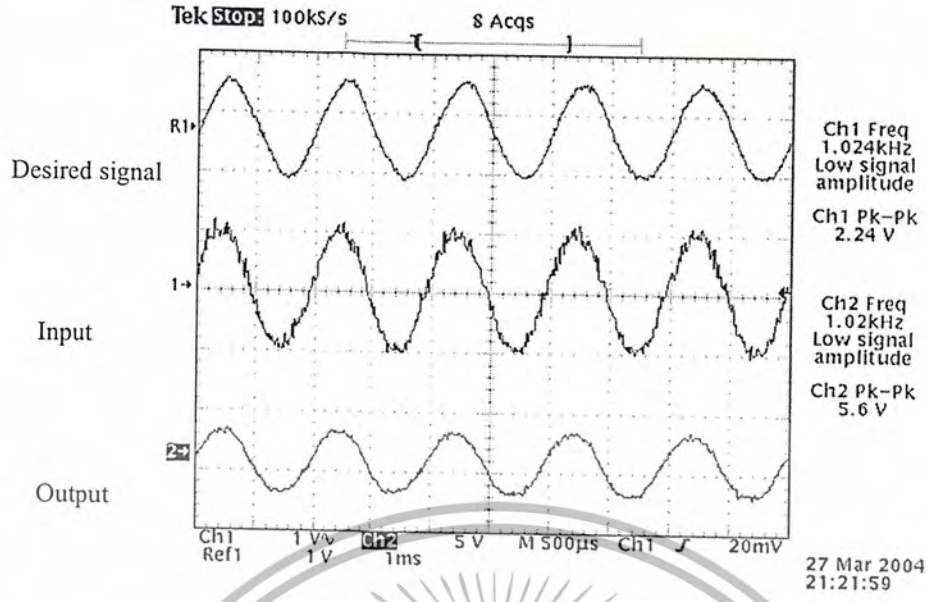
รูปที่ 4.64 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 100 Hz ซึ่งถูกรบกวนด้วยสัญญาณรบกวน (Noise) ขนาด 2 Volt



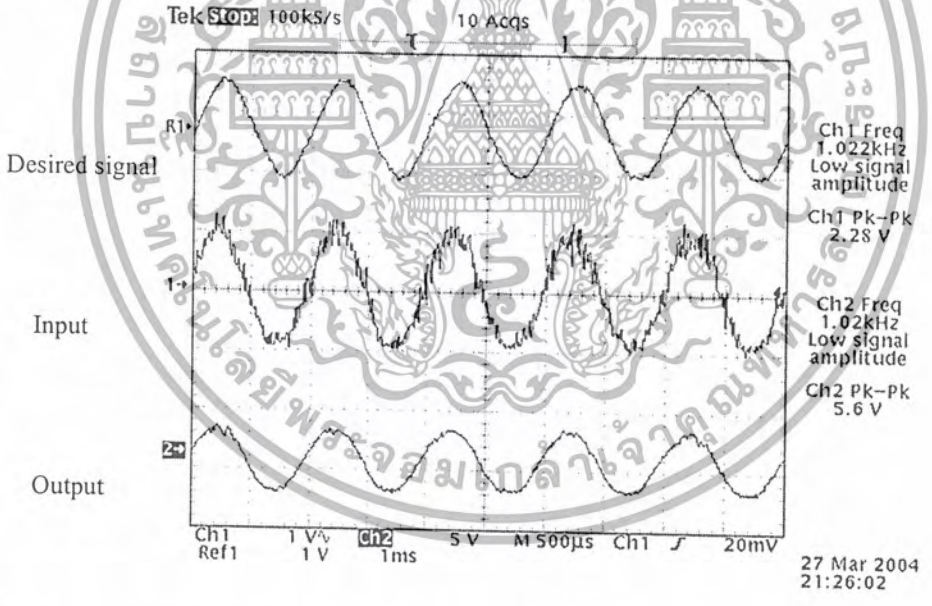
รูปที่ 4.65 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 500 Hz ซึ่งถูกรบกวนด้วยสัญญาณรบกวน (Noise) ขนาด 1 Volt



รูปที่ 4.66 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 500 Hz ซึ่งถูกรบกวนด้วยสัญญาณรบกวน (Noise) ขนาด 2 Volt



รูปที่ 4.67 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 1 kHz ซึ่งถูกรบกวนด้วยสัญญาณรบกวน (Noise) ขนาด 1 Volt

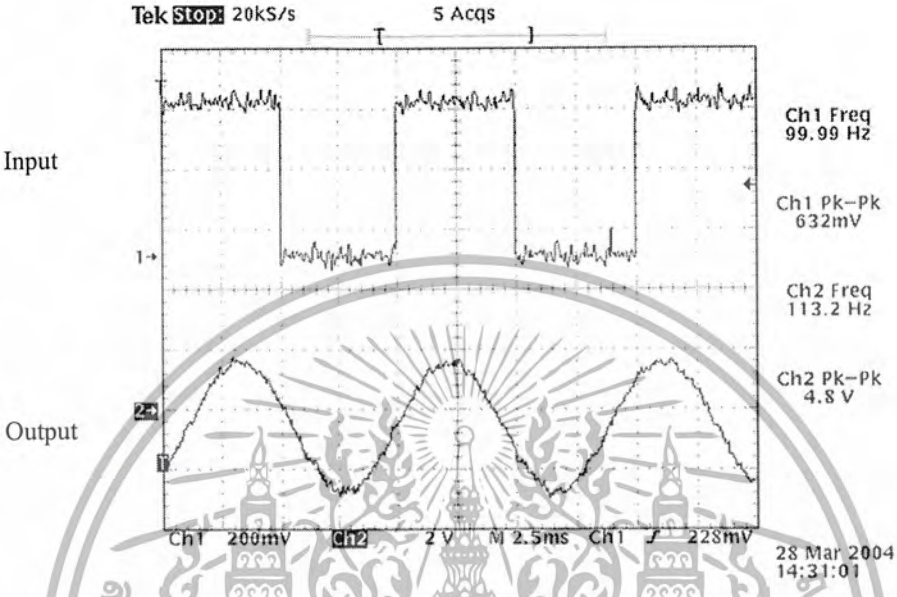


รูปที่ 4.68 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณเบสแบนด์รูปไซน์ความถี่ 1 kHz ซึ่งถูกรบกวนด้วยสัญญาณรบกวน (Noise) ขนาด 2 Volt

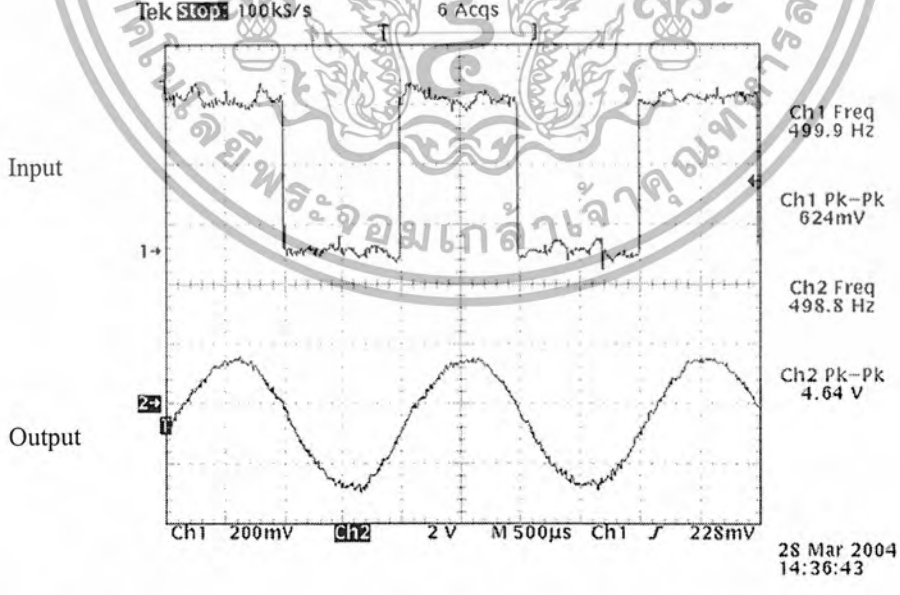
จากรูปที่ 4.63 ถึง 4.68 พบว่าตัวกรองสัญญาณแบบปรับตัวได้อันดับที่ 4 มีขนาดสเตปเท่ากับ 0.25 มีผลตอบสนองของวงจรดีพอสมควร โดยสัญญาณรบกวนแบบสุ่มจะมีองค์ประกอบของทุกความถี่สัญญาณ ทำให้สัญญาณเบสแบนด์มีความเกี่ยวพันกับสัญญาณรบกวน ทำให้คุณภาพการกรองสัญญาณแย่ลง อีกตัวแปรหนึ่งที่สำคัญคือ ขนาดแอมปลิจูดของสัญญาณรบกวนแบบสุ่มจะส่งผลโดยตรงต่อคุณภาพของสัญญาณ ถ้าสัญญาณรบกวนแบบสุ่มมีขนาดแอมปลิจูดมากขึ้นคุณภาพของสัญญาณที่กรองได้จะแย่ลงเช่นกัน

4.4.4 ผลตอบสนองต่อสัญญาณรูปไซน์ที่เป็นองค์ประกอบของสัญญาณสี่เหลี่ยม (Square wave)

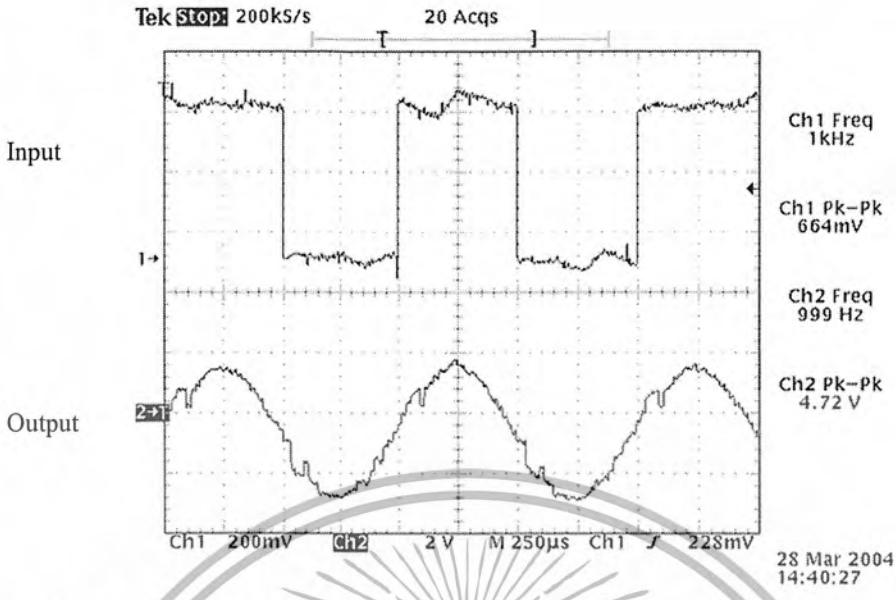
ทำการป้อนสัญญาณอินพุทซึ่งเป็นสัญญาณสี่เหลี่ยมความถี่100 Hz, 500 Hz, 1 kHz, 2 kHz แล้วทำการปรับ Desired Signal ที่ความถี่ Fundamental ต่าง ๆ คือ 100 Hz, 500 Hz, 1 kHz, 2 kHz ใช้ตัวกรองปรับตัวแบบได้อันดับ 4 โดยเลือกใช้ค่า Step size (μ) = 0.25



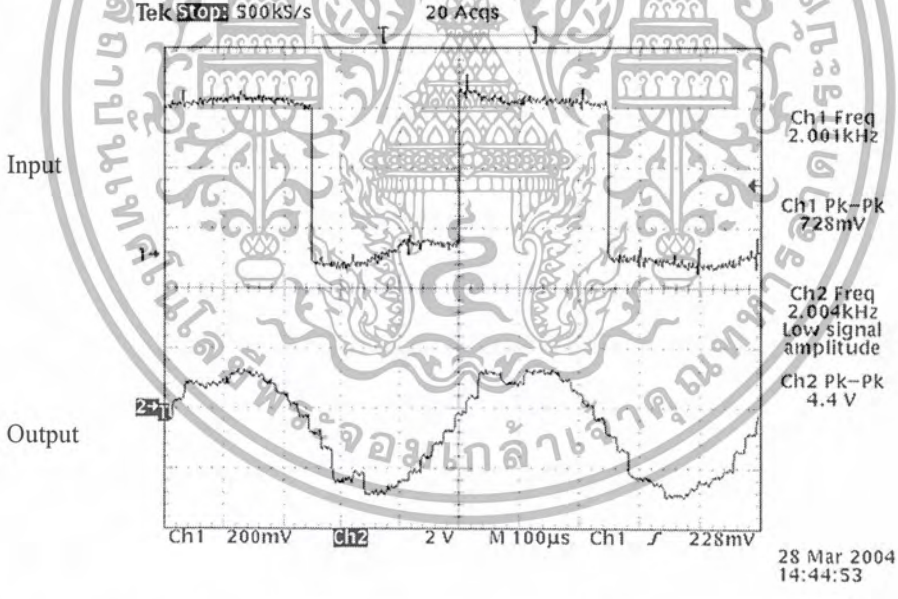
รูปที่ 4.69 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณสี่เหลี่ยมความถี่ 100 Hz โดยปรับ desired signal เท่ากับ 100 Hz



รูปที่ 4.70 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณสี่เหลี่ยมความถี่ 500 Hz โดยปรับ desired signal เท่ากับ 500 Hz



รูปที่ 4.71 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณสี่เหลี่ยมความถี่ 1 kHz โดยปรับ desired signal เท่ากับ 1 kHz



รูปที่ 4.72 ผลตอบสนองของตัวกรองแบบปรับตัวได้ต่อสัญญาณสี่เหลี่ยมความถี่ 2 kHz โดยปรับ desired signal เท่ากับ 2 kHz

จากรูปที่ 4.69 ถึง 4.72 พบว่าตัวกรองสัญญาณแบบปรับตัวได้อันดับที่ 4 มีขนาดสเตปเท่ากับ 0.25 มีผลตอบสนองทางความถี่ที่ดี จากการทดลองป้อนสัญญาณ square wave ซึ่งเป็นผลรวมของสัญญาณฮาโมนิกส์คี่ โดยมี fundamental frequency เท่ากับ 100 Hz, 500 Hz, 1 kHz และ 2 kHz ถ้าปรับ desired signal ให้มีความถี่เท่ากับ fundamental frequency ของอินพุทจะทำให้ผลลัพธ์ของการกรองสัญญาณมีคุณภาพดี

บทที่ 5
บทวิจารณ์และบทสรุป

จากการออกแบบวงจรกรองเชิงเลขแบบปรับตัวได้โดยใช้โครงสร้างเลขคณิตกระจายด้วยวิธี Adaptive Function Space ด้วยจากการใช้โครงสร้างเลขคณิตกระจายที่ไม่ใช้ตัวคูณทำให้ประหยัดเกทที่ใช้งาน อีกทั้งใช้วิธี adaptive function space ทำให้สมการของ LMS Algorithm ออกแบบเป็นฮาร์ดแวร์ได้ง่ายขึ้นซึ่งเป็นข้อดี ทำให้ฮาร์ดแวร์ที่ใช้มีขนาดเล็กลง ส่วนผลจากการ Simulation ด้วยโปรแกรม MATLAB ทำให้ทราบว่าจำนวนอันดับ (order) และ step size (μ) จะมีความสัมพันธ์กัน ถ้าเลือกค่าที่เหมาะสมจะทำให้ผลตอบสนองในการกรองสัญญาณดีขึ้น วงจรกรองจะมีการลู่เข้าดีขึ้น อย่างไรก็ตามการที่ตัวกรองจะทำงานได้ดีนั้นขึ้นอยู่กับตัวแปรอื่น ๆ อีก เช่น ขนาดของสัญญาณอินพุท ค่ากำลังงานของสัญญาณรบกวน ปัจจัยทั้งหลายเหล่านี้ทำให้คุณภาพของสัญญาณเปลี่ยนแปลง อีกปัจจัยหนึ่งคือ มีความเกี่ยวเนื่องสัมพันธ์กัน (Correlation) ของสัญญาณเบสแบนด์และสัญญาณรบกวนมากน้อยแค่ไหน ถ้ามีความเกี่ยวเนื่องสัมพันธ์กันมาก การจะกรองสัญญาณที่เราต้องการออกจากส่งสัญญาณที่มีความสัมพันธ์กันจำเป็นต้องใช้จำนวนอันดับของวงจรสูงขึ้น

Input	desired signal	step size (μ)			
s(k)	d(k)	0.5	0.25	0.125	0.0625
square wave ความถี่ 5 kHz	100 Hz	แย่	พอใช้	ดี	ดี
	500 Hz	แย่	พอใช้	ดี	ดี
	1 kHz	แย่	ดี	ดี	ดี
	2 kHz	แย่	ดี	ดี	ดี
	5 kHz	แย่	ดี	ดี	ดี

ตารางที่ 5.1 แสดงผลตอบสนองของวงจรกรองปรับตัวได้ต่อค่า step size (μ) ต่าง ๆ โดยป้อนสัญญาณอินพุทเป็น square wave

หมายเหตุ คุณภาพของสัญญาณที่ได้จากวงจรกรองปรับตัวได้ จะแบ่งออกเป็น 4 ระดับ

- ดี หมายถึง สัญญาณที่กรองได้มี noise ปนเล็กน้อย
- พอใช้ หมายถึง สัญญาณที่กรองได้มี noise ปนปานกลาง แต่คุณภาพยังดีอยู่
- แย่ หมายถึง สัญญาณที่กรองได้มี noise ปนมากขึ้น จนสัญญาณผิดเพี้ยน
- แย่มาก หมายถึง ไม่สามารถกรองสัญญาณได้

Input s(k)		desired signal	step size (μ)			
Baseband	Noise (0.2V)	d(k)	0.5	0.25	0.125	0.0625
100 Hz	5 kHz	100 Hz	พอใช้	ดี	ดี	ดี
500 Hz	5 kHz	500 Hz	พอใช้	พอใช้	พอใช้	พอใช้
1 kHz	5 kHz	1 kHz	แย่	แย่	พอใช้	พอใช้
2 kHz	5 kHz	2 kHz	แย่	แย่	พอใช้	แย่
5 kHz	5 kHz	5 kHz	แย่มาก	แย่	แย่	แย่

ตารางที่ 5.2 แสดงผลตอบสนองของวงจรกรองปรับตัวได้ต่อค่า step size (μ) ต่าง ๆ โดยป้อนสัญญาณอินพุทเป็น Baseband+ Noise (0.2V)

Input s(k)		desired signal	step size (μ)			
Baseband	Noise (0.2V)	d(k)	0.5	0.25	0.125	0.0625
100 Hz	5 kHz	100 Hz	พอใช้	ดี	ดี	ดี
500 Hz	5 kHz	500 Hz	พอใช้	พอใช้	พอใช้	แย่
1 kHz	5 kHz	1 kHz	แย่	แย่	แย่	แย่
2 kHz	5 kHz	2 kHz	แย่มาก	แย่	แย่	แย่มาก
5 kHz	5 kHz	5 kHz	แย่มาก	แย่มาก	แย่มาก	แย่มาก

ตารางที่ 5.3 แสดงผลตอบสนองของวงจรกรองปรับตัวได้ต่อค่า step size (μ) ต่าง ๆ โดยป้อนสัญญาณอินพุทเป็น Baseband+ Noise (0.4V)

สำหรับปัญหาในการออกแบบและการสร้างคือ การซิมูเลชั่นสำหรับตัวกรองปรับตัวได้ต้องซิมูเลชั่นตรวจจับสัญญาณไปตามแกนเวลาซึ่งยากกว่าการซิมูเลชั่นด้วยตัวกรองธรรมดาทั่วไปและการออกแบบในวงจรด้วยภาษา VHDL ความละเอียดของของวงจรที่ใช้ประมวลผลภายในถ้ามีจำนวนของบิทน้อยค่าความละเอียดจะต่ำจะทำให้ไม่สามารถตรวจจับสัญญาณที่ต้องการได้ การออกแบบวงจรกรองทำได้แค่อันดับที่ 4 เนื่องจากข้อจำกัดของอุปกรณ์ ซึ่งมีจำกัดเพียงแค่ 10,000 เกท

การประยุกต์ใช้งาน

โดยภาพรวมการออกแบบวงจรกรองเชิงเลขแบบปรับตัวได้โดยใช้โครงสร้างเลขคณิตกระจายด้วยอุปกรณ์ FPGA เน้นการทดลองเบื้องต้นเกี่ยวกับการวัดคุณลักษณะของวงจรกรองแบบปรับตัวได้เมื่อป้อนสัญญาณในลักษณะต่าง ๆ กันแล้ววงจรกรองสามารถตรวจจับสัญญาณตามสัญญาณที่เราต้องการได้ ส่วนการนำไปประยุกต์ใช้งาน เช่น Echo Cancellation for Telephony, Spectral Line Enhancement

ภาคผนวก

โปรแกรมพล็อตค่า Mean Square Error

```
clear all;close all;
no_of_data = 2000;      % set number of data points used for training
order = 8; % set the adaptive filter order
mu = 0.0625; % set the step size constant
x=randn(no_of_data,1); % input assumed to be white noise
h=rand(order,1); % system picked randomly
d=filter(h,1,x); % generate output (desired signal)
% Initialize The LMS Algorithm
w=zeros(order,1);
J=zeros(no_of_data,1);
for i = 1:runs
% The LMS Adaptation
for n= order : no_of_data
    D = x(n : -1 : n-order+1);
    d_hat(n) = w'*D;
    e(n) = d(n)- d_hat(n);
    w = w + 2*mu*e(n)*D;
    w_error(n) = norm(h-w);
    J(n) = J(n)+ e(n)^2;
end;
end;
J = J/runs;
% Plot results
figure;
plot(20*log10(abs(e)));
title('Learning Curve');
xlabel('Number of Iterations');
ylabel('Output Estimation Error in dB');
figure;
semilogy(w_error);
title('Weight Estimation Error');
xlabel('Number of Iterations');
ylabel('Weight Errir in dB');
figure;
%plot(J,'r')
semilogy(J,'r');
title('Mean Squre Error');
xlabel('Number of Iterations');
ylabel('MSE');
```

โปรแกรมที่ใช้ในการซิมูเลชั่นของตัวกรองปรับตัวได้แบบเอฟไอาร์

```
clear all;clc;close all;
randn('seed',0);
rand('seed',0);
no_of_data = 4000;      % set number of data points used for training
order =2; % set the adaptive filter order
runs = 10;
fs =4000;
t = 1/(fs);
%baseamp = 10;
freqbase =1000;%input('Type in Baseband Frequency (Hz) = ');
freqbase1 = input('Type in noise(sin) Frequency (Hz) = ');
baseband = 0.7*cos(2*pi*freqbase*t*[1:no_of_data]);
%noise = 1*randn(1,no_of_data);
noise = 0.3*cos(2*pi*freqbase1*t*[1:no_of_data]);
mu = input('Step Size = ');% set the step size constant
J=zeros(no_of_data,1);
for i = 1:runs
x = baseband+noise; % input assumed to be white noise
x_in = x';
%h=rand(order,1); % system picked randomly
d=baseband; % generate output (desired signal)
h = rand(order,1);
%d = filter(h,1,baseband);
% Initialize The LMS Algorithm
w=zeros(order,1);
% The LMS Adaptation
for n= order : no_of_data
    D = x_in(n : -1 : n-order+1);
    d_hat(n) = w'*D;
    e(n) = d(n)-d_hat(n);
    w = w + 2*mu*e(n)*D;
    w_error(n) = norm(h-w);
    J(n) = J(n)+ e(n)^2;
 end
end
J = J/runs;
% Plot results
%figure;
%plot(abs(e))
%plot(20*log10(abs(e)));
title('Learning Curve');
xlabel('Number of Iterations');
ylabel('Output Estimation Error in dB');
%figure;
%semilogy(w_error);
title('Weight Estimation Error');
xlabel('Number of Iterations');
ylabel('Weight Error in dB');
%figure;
%plot(J,'r')
%semilogy(J,'r');
title('Mean Squre Error');
xlabel('Number of Iterations');
ylabel('MSE');
%hold on
```

```
figure;
plot(x)
title('Baseband+Noise Waveform');
xlabel('Number of Iterations');
ylabel('Amplitude');
figure;
plot(d_hat)
title('Output Waveform of Adaptive Filter');
xlabel('Number of Iterations');
ylabel('Amplitude');
figure;
plot(baseband)
title('Baseband Input Waveform');
xlabel('Number of Iterations');
ylabel('Amplitude');
figure;
plot(e)
title('Error Signal');
xlabel('Number of Iterations');
ylabel('Amplitude');
```

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ สามารถสำเร็จลุล่วงลงได้ด้วยความช่วยเหลือ ให้คำแนะนำ ให้คำปรึกษา และชี้แนะแนวทางวิธีการจาก รศ.ดร.กอบชัย เดชหาญ และ อาจารย์ศรวัฒน์ ชิวปรีชา ซึ่งเป็นประโยชน์อย่างมากในการทำปริญญานิพนธ์ และขอขอบคุณเพื่อน ๆ ทุกคนสำหรับความมีน้ำใจที่คอยช่วยเหลือ ติชมชิ้นงานให้สำเร็จลุล่วงไปด้วยดี กราบขอบพระคุณทุก ๆ ท่านเป็นอย่างสูงไว้ ณ ที่นี้ด้วย

สุดท้ายนี้ขอขอบพระคุณ บิดา มารดา ที่สนับสนุนลูก ๆ เช่น การเงิน ความรัก และคำด่า ที่ช่วยให้เราเติบโตมาในจนทุกวันนี้

เสฐียรพล ปิยะมหาโชติ
เอนกชัย นำเจริญวัฒนกุล

บรรณานุกรม

1. C.F.N. Cowan and P.M. Grant, "Adaptive Filters," Prentice Hall, 1985.
2. S. Haykin, "Adaptive Filter," Prentice Hall, 2002
3. A.V. Oppenheim, R.W. Schafer, J.R. Buck, "Discrete-Time Signal Processing," Prentice Hall, 1999
4. C.F.N. Cowan and J. Mavor, "New Digital Adaptive Filter Implementation using Distributed Arithmetic Techniques," IEE Proc., vol.128, Pt.F, No.4, pp.225-230, Aug. 1981.
5. C.H. Wei and J.J. Lou, "Multimemory Block Structure for Implementing a Digital Adaptive Filter using Distributed Arithmetic," IEE Proc., vol.133, Pt.G, No.1, pp.19-26, Feb. 1986.
6. E. C. Ifeachor and B. W. Jervic, "Digital Signal Processing," Adison-wesley, 1998, 541-576.
7. K. Takahashi, "Analysis of the Convergence Condition of LMS Adaptive Digital Filter Using Distributed Arithmetic," IEICE Trans. Fundamental , vol.E85-A, No.6 , pp.1249-1256, June 2002