การปรับแต่งช่องสัญญาณโดยใช้ FPGA
CHANNEL EQUALIZATION USING FPGA

โดย
นางสาวสุพรรษา ครองกิติชู
นางสาวสุภาวัญญ์ ปิยจารุลักษณ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมโทรคมนาคม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2548

การปรับแต่งช่องสัญญาณโดยใช้ FPGA

CHANNEL EQUALIZATION USING FPGA

โดย

นางสาวสุพรรษา ครองกิติชู 45010859

นางสาวสุภาวัญญ์ ปิยจารุลักษณ์ 45010863

อาจารย์ที่ปรึกษา

รศ.ดร.กอบชัย เดชหาญ

อาจารย์ สมปอง วิเศษพาณิชกิจ

รายงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมโทรคมนาคม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง **การปรับแต่งช่องสัญญาณโดยใช้ FPGA**

CHANNEL EQUALIZATION USING FPGA

ผู้จัดทำ

1. **นางสาวสุพรรษา ครองกิติชู 45010859**
2. **นางสาวสุภาวัญญ์ ปิยจารุลักษณ์ 45010863**

.... อาจารย์ที่ปรึกษา

(รศ.ดร.กอบชัย เดชหาญ)

.... อาจารย์ที่ปรึกษา

(อ.สมปอง วิเศษพาณิชกิจ)

การปรับแต่งช่องสัญญาณโดยใช้ FPGA

CHANNEL EQUALIZATION USING FPGA

โดย นางสาวสุพรรษา ครองกิติชู 45010859

นางสาวสุภาวัญญ์ ปิยจารุลักษณ์ 45010863

อาจารย์ที่ปรึกษา รศ.ดร.กอบชัย เดชหาญ

และ อ.สมปอง วิเศษพาณิชกิจ

บทคัดย่อ

โครงงานนี้เพื่อศึกษาและทำการทดลองการปรับแต่งช่องสัญญาณ (Channel Equalization) สำหรับใช้ทดสอบในช่องสัญญาณ AWGN ซึ่งจะใช้โปรแกรม MATLAB ในการจำลองกระบวนการทำงาน โดยขั้นตอนการออกแบบจะใช้วิธีอธิบายการทำงานวงจรด้วย VHDL ลงบนอุปกรณ์ FPGA เพื่อเป็นต้นแบบในการใช้งานต่อไป

ABSTRACT

This project studies and experimental channel equalization for testing in AWGN channel. All results are simulated with MATLAB program simulator. The processor will be implemented on FPGA by using VHDL language.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สามารถสำเร็จลุล่วงลงได้ด้วยความช่วยเหลือ ให้คำปรึกษาและคำแนะนำ รวมทั้งชี้แนะแนวทางเป็นอย่างดีตลอดระยะเวลาจากอาจารย์ที่ปรึกษา รศ.ดร.กอบชัย เดชหาญ และ อาจารย์สมปอง วิเศษพาณิชกิจ ซึ่งเป็นประโยชน์อย่างมากในการทำปริญญานิพนธ์ ขอขอบคุณพี่ๆ และ เพื่อนๆ ในห้องโปรเจคท์ทุกคนที่คอยช่วยเหลือในทุกๆ ด้านตลอดมา

นางสาวสุพรรษา ครองกิติชู
นางสาวสุภาวัญญ์ ปิยจารุลักษณ์

สารบัญ

หน้า

บทที่ 1 บทนำ 1

บทที่ 2 ทฤษฎีและหลักการ 3

2.1 โครงสร้างของระบบสื่อสาร 3

2.2 ช่องสัญญาณในการสื่อสาร 4

2.2.1 ปัจจัยที่มีส่วนเกี่ยวข้องกับความเร็วบนตัวกลางที่ใช้ส่งข้อมูล 4

2.2.2 การแทรกสอดระหว่างสัญลักษณ์ (Intersymbol interference: ISI) 4

2.2.3 ตัวกลางส่งข้อมูลแบบมีสายนำทาง (Guided Media) 5

2.2.4 ความสามารถสูงสุดในการส่งสัญญาณของช่องสัญญาณ 7

2.3 สัญญาณรบกวน 7

2.3.1 คุณสมบัติทั่วไปของสัญญาณรบกวน 7

2.3.1.1 ค่าเฉลี่ยของสัญญาณรบกวน 8

2.3.1.2 ค่ากำลังสองเฉลี่ยของสัญญาณรบกวน 8

2.3.1.3 ค่าความแปรปรวนของสัญญาณรบกวน 8

2.3.2 สัญญาณรบกวนแบบเกาส์เซียน 9

2.4 อิควอไลเซชัน (Equalization) 10

2.5 ชนิดของอิควอไลเซอร์ 12

2.5.1 อิควอไลเซอร์แบบเชิงเส้น (Linear equalization) 12

2.5.1.1 อิควอไลเซอร์แบบเชิงเส้นแบบเทรน 13

2.5.1.2 อิควอไลเซอร์แบบเชิงเส้นแบบบอด 13

2.5.2 อิควอไลเซอร์แบบไม่เป็นเชิงเส้น (Nonlinear equalizer) 14

2.5.2.1 อิควอไลเซอร์แบบไม่เป็นเชิงเส้นแบบเทรน 14

2.5.2.2 อิควอไลเซอร์แบบไม่เป็นเชิงเส้นแบบบอด 15

2.6 ทฤษฎีพื้นฐานของวงจรกรองความถี่แบบดิจิตอล 16

2.6.1 ความหมายของวงจรกรองความถี่แบบดิจิตอล 16

2.6.2 ส่วนประกอบที่สำคัญในวงจรกรองความถี่แบบดิจิตอล 17

2.6.3 วงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัด หรือ FIR 18

2.7 วงจรกรองความถี่แบบปรับตัวได้ (Adaptive Filter) 20

2.7.1 โครงสร้างพื้นฐานของวงจรกรองความถี่แบบปรับตัวได้ 21

2.7.2 กรรมวิธีการพัฒนาอัลกอริทึมของวงจรกรองความถี่แบบปรับตัวได้ 22

2.7.3 การวัดประสิทธิภาพในระบบปรับตัว 22

2.7.4 การประยุกต์ใช้งานวงจรกรองปรับตัวได้ 24

2.7.5 วงจรกรองความถี่ปรับตัวได้แบบเอฟไออาร์ (Adaptive FIR Filter) 26

สารบัญ (ต่อ)

	หน้า
2.8 ระบบดิจิตอลเบื้องต้น	28
2.8.1 ระบบเลขฐานสิบ (Decimal Number Systems)	28
2.8.2 ระบบเลขฐานแปด (Octal Number Systems)	28
2.8.3 ระบบเลขฐานสิบหก (Hexadecimal Number Systems)	29
2.8.4 ระบบเลขฐานสอง (Binary Number Systems)	29
2.8.5 การแปลงเลขฐานสองแบบทูคอมพลีเมนท์ (Two's complement)	30
2.8.6 รูปแบบตัวแทนของจำนวนเลขที่มีเครื่องหมายกำกับ	30
2.9 อัลกอริทึมแบบบ็อก-วูเลย์ (Baugh-Wooley Algorithm)	31
บทที่ 3 การคำนวณและการสร้าง	35
3.1 การออกแบบระบบจำลองการสื่อสาร	35
3.1.1 สัญญาณพีเอเอ็ม 4 ระดับ	35
3.1.2 ช่องสัญญาณที่มีสัญญาณรบกวน (The Noisy Channel)	35
3.1.3 การใช้อิควอไลเซอร์ในช่องสัญญาณ	36
3.2 วิธีการวัดสมรรถนะและความหมายของตัววัดสมรรถนะต่างๆ	36
3.2.1 อัตราส่วนสัญญาณต่อสัญญาณรบกวน (Signal to noise ratio: SNR)	36
3.2.2 อัตราความผิดพลาดสัญลักษณ์ (Symbol error rate: SER)	36
3.2.3 อัตราความผิดพลาดบิท (Bit error rate: BER)	36
3.3 การออกแบบการจำลองการทำงานด้วยโปรแกรมแมทแลบ	37
3.4 การใช้ภาษาวีเอชดีแอลออกแบบสัญญาณพีเอเอ็ม 4 ระดับ	38
3.5 การใช้ภาษาวีเอชดีแอลออกแบบการจำลองช่องสัญญาณแบบ AWGN	39
3.5.1 การสร้างสัญญาณไบนารีแบบสุ่มโดยใช้วงจรนับ LFSR	40
3.5.2 วิธีโพลาร์	40
3.5.3 วิธีเซนทรัลลิมิต	41
3.6 การใช้ภาษาวีเอชดีแอลออกแบบอิควอไลเซอร์	42
บทที่ 4 การทดลองและผลการทดลอง	43
4.1 ผลการทดลองการจำลองการทำงานด้วยโปรแกรมแมทแลบ	43
4.1.1 เมื่อสัญญาณอินพุทเป็นสัญญาณพีเอเอ็ม 2 ระดับ	44
4.1.2 เมื่อสัญญาณอินพุทเป็นสัญญาณพีเอเอ็ม 4 ระดับ	48
4.1.3 เมื่อสัญญาณอินพุทเป็นสัญญาณคิวพีเอสเค	52
4.2 ผลการทดลองการออกแบบวงจรโดยใช้ภาษาวีเอชดีแอล	53
4.2.1 วงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัด	53
4.2.2 วงจรกลับบิทเครื่องหมาย	54

สารบัญ (ต่อ)

หน้า

4.2.3 วงจรหารความถี่ 55

4.2.4 ส่วนการสร้างแบบจำลองช่องสัญญาณ 56

4.2.5 วงจร LFSR 57

4.2.6 วงจรรวมบิท 62

4.2.7 วงจรเก็บค่าและเรียกค่าข้อมูล 63

4.2.8 วงจรตัดบิท 64

4.2.9 วงจรคูณ 65

4.2.10 วงจรทูคอมพลีเมนท์ 66

4.2.11 วงจรบัฟเฟอร์ 67

4.2.12 วงจรบวก 68

4.2.13 ส่วนการสร้างสัญญาณไบนารี่แบบสุ่มด้วยวิธีโพลาร์ 69

4.2.14 ส่วนการสร้างเซนทรัลลิมิต 70

4.2.15 ส่วนการสร้างสัญญาณ AWGN 71

4.2.16 ส่วนการสร้างแบบจำลองช่องสัญญาณ AWGN 72

4.2.17 วงจรกรองความถี่ผ่านแบบปรับตัวได้ 73

4.2.18 ส่วนการสร้างวงจรอิควอไลเซอร์ 74

4.3 การนำโปรแกรมที่ออกแบบโดยภาษาวีเอชดีแอลไปสร้างเป็นวงจรอิควอไลเซอร์ 75

4.3.1 การออกแบบการทดลองสำหรับทดสอบวงจรอิควอไลเซอร์ที่ออกแบบ 76

4.3.2 ผลการทดลองวงจรแปลงสัญญาณเชิงอนาล็อกเป็นสัญญาณดิจิตอล และวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณเชิงอนาล็อก 77

4.3.3 ผลการทดลองวงจรสร้างสัญญาณพีเอเอ็ม 4 ระดับ 77

4.3.4 ผลการทดลองแบบจำลองช่องสัญญาณแบบ AWGN 78

4.3.5 ผลการทดลองวงจรกรองความถี่ผ่านแบบปรับตัวได้อันดับที่ 8 80

4.3.6 ผลการทดลองวงจรอิควอไลเซอร์ 82

4.3.6.1 เมื่อนำสัญญาณผ่านช่องสัญญาณแบบที่สุ่มขึ้น 82

4.3.6.2 เมื่อนำสัญญาณผ่านช่องสัญญาณแบบโพรอากิส-เอ 84

บทที่ 5 บทวิจารณ์และบทสรุป 86

ภาคผนวก

กิตติกรรมประกาศ

หนังสืออ้างอิง

สารบัญรูปภาพ

หน้า

รูปที่ 2.1 แบบจำลองระบบสื่อสาร 3

รูปที่ 2.2 แสดงตัวอย่างการรบกวนระหว่างสัญลักษณ์ของพัลส์รูปสี่เหลี่ยมทั้งสองขณะส่งผ่านช่องสัญญาณที่มีแบนด์วิดท์จำกัด 5

รูปที่ 2.3 สายคู่บิดเกลียว 6

รูปที่ 2.4 โครงสร้างรายละเอียดของสายโคแอกเชียล 6

รูปที่ 2.5 โครงสร้างรายละเอียดของสายใยแก้วนำแสง 7

รูปที่ 2.6 แสดงพีดีเอฟแบบเกาส์เซียน 10

รูปที่ 2.7 แสดงการใช้อิควอไลเซอร์แก้ปัญหาการผิดเพี้ยนของสัญญาณ 10

รูปที่ 2.8 แสดงความหนาแน่นสเปกตรัมพลังงานของพัลส์สี่เหลี่ยม 11

รูปที่ 2.9 แสดงอิควอไลเซอร์แบบเชิงเส้น 12

รูปที่ 2.10 แสดงการใช้อิควอไลเซอร์ที่เป็นจำนวนเชิงซ้อน สำหรับสัญญาณ QPSK โดยจะมีการแบ่งส่วนที่เป็นจำนวนจินตภาพและจำนวนจริง 12

รูปที่ 2.11 แสดงอิควอไลเซอร์แบบเชิงเส้นใน decision-directed mode ซึ่งจะถูกใช้เหมือน Complex FIR filter 12

รูปที่ 2.12 โครงสร้างพื้นฐานของระบบอิควอไลเซอร์แบบเทรน 13

รูปที่ 2.13 โครงสร้างพื้นฐานของระบบอิควอไลเซอร์แบบบอด 13

รูปที่ 2.14 โครงสร้างพื้นฐานของระบบอิควอไลเซอร์แบบไม่เป็นเชิงเส้น 14

รูปที่ 2.15 โครงสร้างพื้นฐานของระบบอิควอไลเซอร์แบบไม่เป็นเชิงเส้นแบบเทรน 15

รูปที่ 2.16 โครงสร้างพื้นฐานของระบบอิควอไลเซอร์แบบไม่เป็นเชิงเส้นแบบบอด 15

รูปที่ 2.17 บล็อกไดอะแกรมของวงจรกรองสัญญาณดิจิตอล 16

รูปที่ 2.18 ส่วนประกอบที่สำคัญในวงจรกรองความถี่แบบดิจิตอล 17

รูปที่ 2.19 โครงสร้างวงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัดรูปแบบตรง 19

รูปที่ 2.20 โครงสร้างวงจรกรองความถี่ดิจิตอลแบบ FIR รูปแบบทรานสโพสแบบตรง (TDF) 19

รูปที่ 2.21 โครงสร้างพื้นฐานของวงจรกรองความถี่แบบปรับตัวได้ 21

รูปที่ 2.22 แสดงการประยุกต์ใช้ตัวกรองปรับตัวในการจำลองคุณลักษณะของระบบที่ไม่ทราบ 24

รูปที่ 2.23 แสดงการใช้อิควอไลเซอร์แบบปรับตัวได้ในระบบส่งสัญญาณ 25

รูปที่ 2.24 แสดงโครงสร้างระบบกำจัดสัญญาณรบกวนโดยใช้ตัวกรองปรับตัวได้ 26

รูปที่ 2.25 แสดงอุปกรณ์เพิ่มประสิทธิภาพสาย 26

รูปที่ 2.26 แสดงวิธีการคูณโดยทั่วไปแบบ Two's complement $m \times n$ บิท 31

รูปที่ 2.27 แสดงวิธีการคูณโดยคิดเครื่องหมายบวก/ลบในส่วนของบิทผลลัพธ์ 32

รูปที่ 2.28 แสดงวิธีการคูณที่มีผลลัพธ์ออกมาเป็นบวกทั้งหมด 33

รูปที่ 2.29 อัลกอริทึมการคูณสำหรับเลขทูคอมพลีเมนท์หรือ อัลกอริทึมแบบบ็อก-วูเลย์ 34

สารบัญรูปภาพ (ต่อ)

หน้า

รูปที่ 3.1 แสดงระบบจำลองการสื่อสาร 35
รูปที่ 3.2 แสดงการเขียนสัญญาณในรูป Constellation ของสัญญาณพีเอเอ็ม 4 ระดับ 35
รูปที่ 3.3 แสดงแบบจำลองช่องสัญญาณที่มีสัญญาณรบกวน 36
รูปที่ 3.4 แผนผังการจำลองระบบสื่อสารด้วยโปรแกรมแมทแลบ 37
รูปที่ 3.5 แผนผังการวัดสมรรถนะของระบบ 38
รูปที่ 3.6 แสดงโครงสร้างของการออกแบบสัญญาณพีเอเอ็ม 4 ระดับ 38
รูปที่ 3.7 แผนผังการจำลองช่องสัญญาณแบบ AWGN ด้วยวิธีโพลาร์ 39
รูปที่ 3.8 แสดงแผนผังการทำงานของวิธีโพลาร์ 40
รูปที่ 3.9 กราฟความสัมพันธ์ระหว่าง W กับ S 41
รูปที่ 3.10 แสดงวิธีเซนทรัลลิมิต (N=4) 42
รูปที่ 3.11 แสดงโครงสร้างของวงจรอิควอไลเซอร์ 42
รูปที่ 4.1 แสดงผลตอบสนองอิมพัลส์และผลตอบสนองทางความถี่ของช่องสัญญาณที่สุ่มขึ้น 43
รูปที่ 4.2 แสดงผลตอบสนองอิมพัลส์และผลตอบสนองทางความถี่ของช่องสัญญาณแบบโพรอากิส-เอ 43
รูปที่ 4.3 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = 20 44
รูปที่ 4.4 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = 0 44
รูปที่ 4.5 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = -10 45
รูปที่ 4.6 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณโพรอากิส-เอที่มีค่า SNR = 20 45
รูปที่ 4.7 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณโพรอากิส-เอที่มีค่า SNR = 0 46
รูปที่ 4.8 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณโพรอากิส-เอที่มีค่า SNR = -10 46
รูปที่ 4.9 แสดง Constellation ของสัญญาณพีเอเอ็ม 2 ระดับ 47
รูปที่ 4.10 แสดงค่า BER ของสัญญาณพีเอเอ็ม 2 ระดับ 47
รูปที่ 4.11 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = 20 48
รูปที่ 4.12 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = 0 48
รูปที่ 4.13 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = -10 49
รูปที่ 4.14 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณโพรอากิส-เอที่มีค่า SNR = 20 49
รูปที่ 4.15 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณโพรอากิส-เอที่มีค่า SNR = 0 50

สารบัญรูปภาพ (ต่อ)

หน้า

รูปที่ 4.16 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณโพรอากิส-เอ ที่มีค่า SNR = -1 50

รูปที่ 4.17 แสดง Constellation ของสัญญาณพีเอเอ็ม 4 ระดับ 51

รูปที่ 4.18 แสดงBER ของสัญญาณพีเอเอ็ม 4 ระดับ 51

รูปที่ 4.19 แสดง Constellation ของสัญญาณคิวพีเอสเค 52

รูปที่ 4.20 แสดงผลที่ได้เมื่อป้อนสัญญาณอินพุทแบบคิวพีเอสเค ในช่องสัญญาณทดลอง 52

รูปที่ 4.21 แสดงสัญลักษณ์ของวงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัด 53

รูปที่ 4.22 แสดงผลการจำลองการทำงานของวงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัด 53

รูปที่ 4.23 แสดงสัญลักษณ์ของวงจรกลับบิทเครื่องหมาย 54

รูปที่ 4.24 แสดงผลการจำลองการทำงานของวงจรกลับบิทเครื่องหมาย 54

รูปที่ 4.25 แสดงสัญลักษณ์ของวงจรหารความถี่ 55

รูปที่ 4.26 แสดงผลการจำลองการทำงานของวงจรหารความถี่ 55

รูปที่ 4.27 แสดงสัญลักษณ์ของวงจรแบบจำลองช่องสัญญาณ 56

รูปที่ 4.28 แสดงผลการจำลองการทำงานของวงจรแบบจำลองช่องสัญญาณ 56

รูปที่ 4.29 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 2 57

รูปที่ 4.30 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 2 57

รูปที่ 4.31 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 3 58

รูปที่ 4.32 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 3 58

รูปที่ 4.33 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 5 58

รูปที่ 4.34 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 5 58

รูปที่ 4.35 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 6 59

รูปที่ 4.36 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 6 59

รูปที่ 4.37 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 7 59

รูปที่ 4.38 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 7 59

รูปที่ 4.39 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 8 60

รูปที่ 4.40 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 8 60

รูปที่ 4.41 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 9 60

รูปที่ 4.42 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 9 60

รูปที่ 4.43 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 11 61

รูปที่ 4.44 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 11 61

รูปที่ 4.45 แสดงสัญลักษณ์ของวงจรรวมบิท 4 บิท 62

สารบัญรูปภาพ (ต่อ)

หน้า

รูปที่ 4.46 แสดงผลการจำลองการทำงานของวงจรรวมบิท 4 บิท 62

รูปที่ 4.47 แสดงสัญลักษณ์ของวงจรเก็บค่าและเรียกค่าข้อมูล 63

รูปที่ 4.48 แสดงผลการจำลองการทำงานของวงจรเก็บค่าและเรียกค่าข้อมูล 63

รูปที่ 4.49 แสดงสัญลักษณ์ของวงจรตัดบิท 64

รูปที่ 4.50 แสดงผลการจำลองการทำงานของวงจรตัดบิท 64

รูปที่ 4.51 แสดงสัญลักษณ์ของวงจรคูณ 65

รูปที่ 4.52 แสดงผลการจำลองการทำงานของวงจรคูณ 65

รูปที่ 4.53 แสดงสัญลักษณ์ของวงจรทูคอมพลีเมนท์ 66

รูปที่ 4.54 แสดงผลการจำลองการทำงานของวงจรทูคอมพลีเมนท์ 66

รูปที่ 4.55 แสดงสัญลักษณ์ของวงจรบัฟเฟอร์ 67

รูปที่ 4.56 แสดงผลการจำลองการทำงานของวงจรบัฟเฟอร์ 67

รูปที่ 4.57 แสดงสัญลักษณ์ของวงจรบวก 68

รูปที่ 4.58 แสดงผลการจำลองการทำงานของวงจรบวก 68

รูปที่ 4.59 แสดงสัญลักษณ์ของวงจรสัญญาณไบนารี่แบบสุ่มด้วยวิธีโพลาร์ 69

รูปที่ 4.60 แสดงผลการจำลองการทำงานของวงจรสัญญาณไบนารี่แบบสุ่มด้วยวิธีโพลาร์ 69

รูปที่ 4.61 แสดงสัญลักษณ์ของวงจรเซนทรัลลิมิต 70

รูปที่ 4.62 แสดงผลการจำลองการทำงานของวงจรเซนทรัลลิมิต 70

รูปที่ 4.63 แสดงสัญลักษณ์ของวงจรสร้างสัญญาณ AWGN 71

รูปที่ 4.64 แสดงผลการจำลองการทำงานของวงจรสร้างสัญญาณ AWGN 71

รูปที่ 4.65 แสดงสัญลักษณ์ของวงจรแบบจำลองช่องสัญญาณ AWGN 72

รูปที่ 4.66 แสดงผลการจำลองการทำงานของวงจรแบบจำลองช่องสัญญาณ AWGN 72

รูปที่ 4.67 แสดงสัญลักษณ์ของวงจรกรองความถี่ผ่านแบบปรับตัวได้ 73

รูปที่ 4.68 แสดงผลการจำลองการทำงานของวงจรกรองความถี่ผ่านแบบปรับตัวได้ 73

รูปที่ 4.69 แสดงส่วนการสร้างวงจรอิควอไลเซอร์ 74

รูปที่ 4.70 แสดงผลการจำลองการทำงานของวงจรอิควอไลเซอร์ 74

รูปที่ 4.71 วงจรแปลงสัญญาณเชิงอนาล็อกเป็นสัญญาณดิจิตอล 75

รูปที่ 4.72 วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณเชิงอนาล็อก 75

รูปที่ 4.73 การเตรียมอุปกรณ์สำหรับการทดลองเพื่อทดสอบวงจรอิควอไลเซอร์ที่ออกแบบขึ้น 76

รูปที่ 4.74 วงจรรวมสำหรับการทดลองเพื่อทดสอบวงจรอิควอไลเซอร์ที่ออกแบบขึ้น 76

รูปที่ 4.75 แสดงสัญญาณเอาท์พุทจากการแปลงสัญญาณเชิงอนาล็อกเป็นสัญญาณดิจิตอล และสัญญาณดิจิตอลเป็นสัญญาณเชิงอนาล็อก 77

รูปที่ 4.76 แสดงสัญญาณพีเอเอ็ม 4 ระดับจากวงจรเก็บค่าและเรียกค่าข้อมูล 78

สารบัญรูปภาพ (ต่อ)

หน้า

รูปที่ 4.77 แสดงสัญญาณเอาท์พุทจากแบบจำลองช่องสัญญาณแบบ AWGN เมื่อป้อนสัญญาณ Sinusoidal 78

รูปที่ 4.78 แสดงสัญญาณเอาท์พุทจากแบบจำลองช่องสัญญาณแบบ AWGN เมื่อป้อนสัญญาณรูปสี่เหลี่ยม 79

รูปที่ 4.79 แสดงสัญญาณเอาท์พุทจากแบบจำลองช่องสัญญาณแบบ AWGN เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับ 79

รูปที่ 4.80 สัญญาณเอาท์พุทที่ผ่านตัวกรองความถี่แบบปรับตัวได้อันดับที่ 8 โดยสัญญาณอินพุทเป็นสัญญาณรูปสี่เหลี่ยมความถี่ 100 Hz 80

รูปที่ 4.81 สัญญาณเอาท์พุทที่ผ่านตัวกรองความถี่แบบปรับตัวได้อันดับที่ 8 โดยสัญญาณอินพุทเป็นสัญญาณรูปสี่เหลี่ยมความถี่ 500 Hz 80

รูปที่ 4.82 สัญญาณเอาท์พุทที่ผ่านตัวกรองความถี่แบบปรับตัวได้อันดับที่ 8 โดยสัญญาณอินพุทเป็นสัญญาณรูปสี่เหลี่ยมความถี่ 1 kHz 81

รูปที่ 4.83 สัญญาณเอาท์พุทที่ผ่านตัวกรองความถี่แบบปรับตัวได้อันดับที่ 8 โดยสัญญาณอินพุทเป็นสัญญาณรูปสี่เหลี่ยมความถี่ 2 kHz 81

รูปที่ 4.84 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=23 82

รูปที่ 4.85 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=17 83

รูปที่ 4.86 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=11 83

รูปที่ 4.87 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=23 84

รูปที่ 4.88 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=17 85

รูปที่ 4.89 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=11 85

สารบัญตาราง

หน้า

ตารางที่ 2.1 แสดงค่าน้ำหนักของเลขฐานสิบตำแหน่งต่างๆ 28
ตารางที่ 2.2 แสดงค่าน้ำหนักของเลขฐานแปดตำแหน่งต่างๆ 29
ตารางที่ 2.3 แสดงค่าน้ำหนักของเลขฐานสิบหกตำแหน่งต่างๆ 29
ตารางที่ 2.4 แสดงค่าน้ำหนักของเลขฐานสองตำแหน่งต่างๆ 29

บทที่ 1
บทนำ

1.1 ความเป็นมาของปริญญานิพนธ์

ในปัจจุบันเทคโนโลยีทางด้านการสื่อสารและคอมพิวเตอร์ได้ก้าวหน้าไปอย่างรวดเร็ว และบทบาทสำคัญของความก้าวหน้าเหล่านั้น ก็มาจากการเปลี่ยนแปลงของเทคโนโลยีหลายอย่างที่เคยทำด้วยวงจรอนาล็อกมาเป็นวงจรดิจิตอล เนื่องจากในการประมวลผลสัญญาณดิจิตอลนั้น มีข้อดีเหนือกว่าการใช้วงจรในระบบอนาล็อกมาก เช่น ให้ความถูกต้องแม่นยำมากกว่า สามารถควบคุมได้ง่าย รวมทั้งในการออกแบบหรือเปลี่ยนแปลงแก้ไขก็สามารถทำได้ง่ายกว่าวงจรอนาล็อก ระบบการสื่อสารข้อมูลแบบดิจิตอลได้เข้ามามีบทบาทต่อการดำเนินการของชีวิตของมนุษย์ซึ่งไม่สามารถหลีกเลี่ยงได้ เพื่อใช้ในการติดต่อสื่อสารข้อมูลและได้ทำการพัฒนาระบบการสื่อสารอย่างต่อเนื่องถึงปัจจุบัน และในปัจจุบันได้มีการนำเอาการประมวลผลสัญญาณดิจิตอลมาประยุกต์ใช้งานกันอย่างแพร่หลาย ทั้งในด้านการประมวลผลสัญญาณเสียง การประมวลผลสัญญาณแบบหลายมิติ การวิเคราะห์สัญญาณคลื่นสมอง (EEG) และสัญญาณคลื่นหัวใจ (ECG) การกำจัดสัญญาณรบกวนในระบบการสื่อสาร เป็นต้น

ในการสื่อสารข้อมูลนั้น เราไม่สามารถควบคุมสภาพแวดล้อมต่างๆของระบบได้ทั้งหมด จึงทำให้เกิดการผิดพลาดของข้อมูลที่รับได้ เช่น ในการส่งผ่านข้อมูลอาจจะทำให้เกิดการลดทอนหรือการสูญเสียของสัญญาณ ซึ่งสาเหตุหนึ่งที่ทำให้เกิดปัญหานี้ขึ้นจะเกิดจากการแทรกสอดระหว่างสัญลักษณ์ (Intersymbol Interference: ISI) ที่เกิดเนื่องจากช่องสัญญาณที่ส่งผ่านและสัญญาณรบกวนที่เกิดขึ้นในระบบ ซึ่งปัญหาดังกล่าวสามารถแก้ไขได้หลายวิธี วิธีหนึ่งในการแก้ปัญหาคือ การใช้อิควอไลเซอร์ต่อเข้าที่ภาครับ เพื่อให้ได้ข้อมูลที่ถูกต้องและตรงกัน จะเห็นได้ว่าคุณลักษณะของวงจรอิควอไลเซอร์จะมีการเปลี่ยนแปลงอยู่ตลอดเวลา และสามารถจัดการปัญหาของการแทรกสอดระหว่างสัญลักษณ์ได้อย่างมีประสิทธิภาพ วงจรอิควอไลเซอร์ที่สามารถปรับเปลี่ยนคุณสมบัติไปตามสภาพของช่องสัญญาณที่เปลี่ยนแปลงตามเวลานั้น จำเป็นจะต้องมีการปรับเปลี่ยนค่าสัมประสิทธิ์เพื่อให้เหมาะสมกับสภาพของช่องสัญญาณตลอดเวลา

โครงงานนี้จึงนำเสนอการออกแบบวงจรอิควอไลเซอร์ โดยใช้หลักการของวงจรกรองแบบปรับตัวได้ (Adaptive filter) ที่ใช้อัลกอริทึมค่าเฉลี่ยกำลังสองน้อยสุด (LMS Algorithm) ซึ่งมีการใช้งานอย่างแพร่หลายในด้านการประมวลผลสัญญาณดิจิตอล ในการสร้างนั้นจะทำการซิมูเลชั่นด้วยโปรแกรมแมทแลบ (MATLAB) ก่อนนำไปเขียนโปรแกรมภาษาวีเอชดีแอล (VHDL: Very High Speed Integrated Circuit Hardware Description Language) ในส่วนของการสร้างเป็นฮาร์ดแวร์ จะใช้โครงสร้างวงจรกรองความถี่ดิจิตอลชนิดผลตอบสนองอิมพัลส์จำกัดรูปแบบทรานสโพสแบบตรง (Transposed Direct Form) และหลักการการคูณแบบบู๊อก-วูเลย์ (baugh-wooley) จากนั้นจะทำการบรรยายพฤติกรรมการทำงานด้วยภาษาวีเอชดีแอล วงจรที่ได้ออกแบบจะถูกสังเคราะห์และโปรแกรมลงในอุปกรณ์เอฟพีจีเอ (FPGA : Field Programmable Gate Array) สำหรับทดสอบการทำงาน

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาพื้นฐานระบบการสื่อสารข้อมูล
2. เพื่อศึกษาหลักการและประยุกต์การใช้งานระบบประมวลผลสัญญาณเชิงเลข
3. เพื่อศึกษาและประยุกต์ใช้งานวงจรกรองความถี่แบบปรับตัวได้
4. เพื่อศึกษาหลักการของอัลกอริทึมค่าเฉลี่ยกำลังสองน้อยสุด
5. เพื่อศึกษาและประยุกต์ใช้งานภาษาวีเอชดีแอลในระบบการประมวลผลสัญญาณเชิงเลข

1.3 ขอบเขตของปริญญานิพนธ์

โครงงานนี้เพื่อศึกษาและทำการทดลองการปรับแต่งช่องสัญญาณ (Channel Equalization) สำหรับใช้ทดสอบในช่องสัญญาณ AWGN ซึ่งจะใช้โปรแกรมแมทแลบ ในการจำลองกระบวนการทำงาน โดยขั้นตอนการออกแบบจะใช้วิธีอธิบายการทำงานวงจรด้วยภาษาวีเอชดีแอล ลงบนอุปกรณ์เอฟพีจีเอ เพื่อเป็นต้นแบบในการใช้งานต่อไป

1.4 เนื้อหาของปริญญานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎีและหลักการของระบบการสื่อสาร รวมถึงทฤษฎีที่นำมาใช้ในการออกแบบและการสร้างวงจรอิควอไลเซอร์ปรับค่าได้

บทที่ 3 กล่าวถึงการออกแบบและการสร้างวงจรอิควอไลเซอร์ที่ใช้อัลกอริทึมค่าเฉลี่ยกำลังสองน้อยสุด รูปแบบทรานสโพสแบบตรงโดยใช้หลักการการคูณแบบบ็อก-วูเลย์และการสร้างสัญญาณไบนารีแบบสุ่มโดยใช้วงจรนับ LFSR

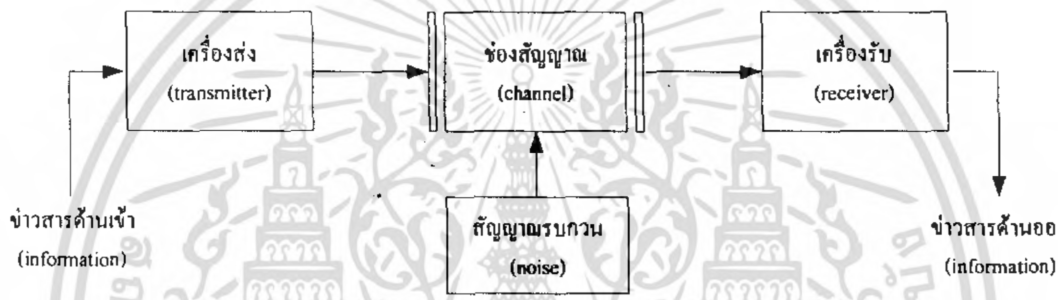
บทที่ 4 กล่าวถึงการทดลองและผลการทดลองที่ได้ จากการจำลองการทำงานโดยใช้โปรแกรมแมทแลบและโปรแกรม MAX PLUS+II ด้วยภาษาวีเอชดีแอล และผลการทำงานบนอุปกรณ์เอฟพีจีเอ

บทที่ 5 บทวิจารณ์และบทสรุป

บทที่ 2
ทฤษฎีและหลักการ

2.1 โครงสร้างของระบบสื่อสาร

โดยทั่วไปเราสามารถแสดงแบบจำลองของระบบสื่อสารได้ดังในรูปที่ 2.1 โดยพิจารณาเริ่มจากทางด้านซ้ายมือ ข่าวสารด้านเข้าที่ป้อนเข้าสู่ระบบสื่อสารคือ เครื่องส่ง (Transmitter) อุปกรณ์ส่วนนี้มีหน้าที่ในการแปลงสัญญาณให้อยู่ในรูปที่เหมาะสมกับช่องสื่อสารที่จะส่งออก เรียกกระบวนการที่ใช้ในการแปลงสัญญาณนี้โดยทั่วไปว่า การมอดูเลต (Modulation) การมอดูเลตสัญญาณมีได้หลายรูปแบบ เช่น การมอดูเลตเชิงแอมพลิจูด การมอดูเลตเชิงความถี่ และการมอดูเลตเชิงเฟส



รูปที่ 2.1 แบบจำลองระบบสื่อสาร

ส่วนถัดมาของแบบจำลองคือ ช่องสัญญาณ (Channel) ในส่วนนี้ก็คือสื่อนำสัญญาณที่พาข่าวสารจากแหล่งกำเนิดไปสู่ปลายทางภาครับ สื่อนำสัญญาณที่ใช้มีตั้งแต่แบบที่มีสายนำสัญญาณ เช่น สายใยแก้วนำแสง สายคู่ตีเกลียว สายโคแอกเชียล ไปจนถึงการส่งสัญญาณประเภทไม่มีสายนำสัญญาณ แต่อาศัยการแพร่ของคลื่นแม่เหล็กไฟฟ้าที่ออกจากสายอากาศไปในบรรยากาศ ซึ่งช่องสัญญาณส่วนใหญ่จะได้รับผลกระทบจากสัญญาณรบกวน และในสภาพที่สัญญาณรบกวนมีความรุนแรงมากอาจจะส่งผลให้ข่าวสารที่ส่งผ่านในช่องสัญญาณเกิดความผิดเพี้ยนของสัญญาณ การลดทอนต่อสัญญาณ ฯลฯ เมื่อสัญญาณได้รับการส่งจนถึงจุดหมายปลายทาง จะได้รับการแปลงกลับโดยเครื่องรับ (Receiver) กระบวนการแปลงสัญญาณในขั้นตอนนี้เรียกว่า กระบวนการดีมอดูเลต (Demodulation) สัญญาณที่ได้จะถูกแปลงกลับคืนให้กลายเป็นข่าวสารที่มีรูปแบบตามต้องการ

ปัญหาที่เกิดกับระบบสื่อสารส่วนใหญ่จะมีสาเหตุมาจากช่องสัญญาณ ดังนั้นอาจกล่าวได้ว่าถ้าไม่มีปัญหาจากช่องสัญญาณที่ใช้แล้ว ปัญหาต่างๆในระบบสื่อสารก็จะหมดไปเกือบทั้งสิ้น

2.2 ช่องสัญญาณในการสื่อสาร

ช่องสัญญาณในการสื่อสารเป็นแบบช่องสัญญาณไร้สาย (Wireless Channel) หรือช่องสัญญาณมีสาย (Wireline Channel) หรือเป็นทั้งสองอย่าง ในบางกรณีจะกล่าวถึงการลดทอนแบบเชิงเส้นและไม่เป็นเชิงเส้น ส่วนประกอบของสัญญาณรบกวนแบบสุ่ม การแทรกสอด ในการส่งข่าวสารที่มีความเร็วสูงและมีความน่าเชื่อถือซึ่งเป็นไปได้ยาก แต่ภายใต้เงื่อนไขก็สามารถเป็นไปได้เนื่องจากใช้พื้นฐานในเรื่องต่างๆ เช่น ทฤษฎีข่าวสาร กระบวนการของสัญญาณ ทฤษฎีของระบบเชิงเส้นและคณิตศาสตร์ และสำหรับคุณสมบัติและคุณภาพของการส่งผ่านข้อมูลจะพิจารณาทั้งในส่วนของตัวกลาง และคุณสมบัติของสัญญาณ ในส่วนของตัวกลางที่ใช้ส่งข้อมูลแบบใช้สายนั้นถือเป็นส่วนสำคัญยิ่งที่จะต้องพิจารณาข้อจำกัดต่างๆ ในการส่งผ่านข้อมูล

2.2.1 ปัจจัยที่มีส่วนเกี่ยวข้องกับความเร็วบนตัวกลางที่ใช้ส่งข้อมูล

1. แบนด์วิดท์ (Bandwidth) คือ แถบความถี่ของช่องสัญญาณ ซึ่งหากมีช่องสัญญาณขนาดใหญ่ ก็จะส่งผลให้ภายในหนึ่งหน่วยเวลาจะสามารถเคลื่อนย้ายปริมาณข้อมูลได้จำนวนมาก นอกจากนี้ความหมายของแบนด์วิดท์โดยทั่วไปคือ ความสามารถในการส่งข้อมูลของช่องสัญญาณของสายส่ง

2. ความสูญเสียต่อการส่งผ่าน (Transmission Impairments) หมายถึง การอ่อนตัวของสัญญาณ ซึ่งการอ่อนตัวของสัญญาณจะเกี่ยวข้องกับระยะทางในการส่งผ่านข้อมูล ตัวอย่างเช่น ตัวกลางที่ใช้ในการส่งผ่านแบบใช้สายอย่างสายคู่บิดเกลียวนั้น จะมีความสูญเสียต่อการส่งผ่านข้อมูลผ่านตัวกลางมากกว่าสายเคเบิล กล่าวคือสายเคเบิลสามารถเชื่อมโยงในระยะทางที่ไกลกว่า หรือมีการอ่อนตัวของสัญญาณที่น้อยกว่า และในทางกลับกันสายเคเบิลก็จะมีข้อจำกัดในระยะทางมากกว่าสายใยแก้วนำแสง

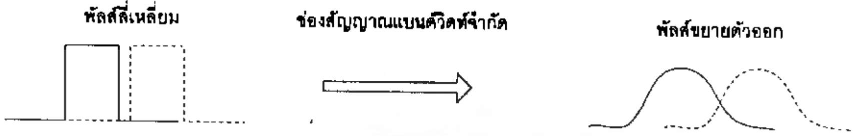
3. การรบกวนของสัญญาณ (Interference) คือ การรบกวนของสัญญาณที่คาบเกี่ยวกันในความถี่อาจส่งผลให้เกิดการบิดเบือนของสัญญาณได้ไม่ว่าจะเป็นตัวกลางส่งข้อมูลแบบมีสายหรือแบบไม่มีสาย ก็สามารถถูกรบกวนด้วยสัญญาณต่างๆได้เช่นกัน เช่น คลื่นวิทยุที่อาจถูกรบกวนหรือถูกแทรกแซงจากคลื่นอื่นๆ หรืออุปกรณ์ที่มีสนามแม่เหล็ก สายคู่เกลียวชนิดไม่มีชีลด์ ซึ่งภายในจะประกอบด้วยสายบิดจำนวนมากหลายๆ คู่ที่มัดอยู่รวมกันภายในฉนวนหุ้มภายนอก ก็จะมีการรบกวนของสัญญาณได้ง่าย ซึ่งวิธีแก้ก็คืออาจใช้สายคู่บิดเกลียวชนิดมีชีลด์ ป้องกันสัญญาณรบกวน

4. จำนวนโหนดที่เชื่อมต่อ (Number of Receivers) คือ ตัวกลางที่ใช้ส่งข้อมูลแบบมีสาย สามารถนำมาเชื่อมต่อในรูปแบบจุดต่อจุด หรือแบบหลายจุดที่ใช้สายส่งข้อมูลร่วมกัน ซึ่งการเชื่อมต่อที่ใช้สายส่งข้อมูลร่วมกัน

2.2.2 การแทรกสอดระหว่างสัญลักษณ์ (Intersymbol interference: ISI)

ในการส่งสัญญาณดิจิตอลโดยใช้พัลส์สี่เหลี่ยมจัดว่าเป็นวิธีที่ตรงไปตรงมา ข้อมูลแต่ละบิทหรือแต่ละสัญลักษณ์จะส่งในเวลาของมันเองที่แยกจากข้อมูลบิทหรือสัญลักษณ์อื่นอย่างเด็ดขาด ดังนั้นรูปสัญลักษณ์ที่พิจารณาทางเวลาของการส่งแต่ละบิทหรือสัญลักษณ์ที่ติดกัน จะไม่มีการซ้อนทับหรือเกยกันเลย ในสภาวะเช่นนี้มีความหมายว่าการส่งข้อมูลแต่ละบิทหรือแต่ละสัญลักษณ์จะไม่มีการรบกวนหรือแทรกแซงระหว่างกันแต่อย่างใด อย่างไรก็ตามพัลส์สี่เหลี่ยมมีสเปกตรัมความถี่ที่กว้างมาก ฉะนั้นหากส่งพัลส์สี่เหลี่ยมช่องสัญญาณที่มีแบนด์วิดท์จำกัด พัลส์สี่เหลี่ยมที่รับได้ที่ภาครับปลายทางจะมีความผิดเพี้ยน

ไปจากเดิมดังตัวอย่างในรูปที่ 2.2 สังเกตว่าพัลส์ที่ส่งออกมีการแผ่ออกทางเวลา และเริ่มมีการเกยกันกับบิทข้างเคียง ผลที่เกิดขึ้นนี้มีชื่อว่า การแทรกสอดระหว่างสัญลักษณ์ เพื่อสะดวกในการอธิบายและง่ายต่อการทำความเข้าใจ เนื้อหาที่จะกล่าวต่อไปจะขอกล่าวใช้คำว่าบิทแทนการใช้คำว่าสัญลักษณ์ ถึงแม้คำว่าสัญลักษณ์จะให้ความหมายที่ครอบคลุมได้กว้างกว่า



รูปที่ 2.2 แสดงตัวอย่างการรบกวนระหว่างสัญลักษณ์ของพัลส์รูปสี่เหลี่ยมทั้งสองขณะส่งผ่านช่องสัญญาณที่มีแบนด์วิดท์จำกัด

จากการที่ช่องสัญญาณสื่อสารในทางปฏิบัติมีขนาดแบนด์วิดท์จำกัด การส่งบิทข้อมูลโดยใช้พัลส์สี่เหลี่ยมจึงก่อปัญหาของการแทรกสอดระหว่างสัญลักษณ์อย่างรุนแรง ซึ่งส่งผลให้การตัดสินใจบิทที่ภาครับมีความผิดพลาดได้โดยง่าย

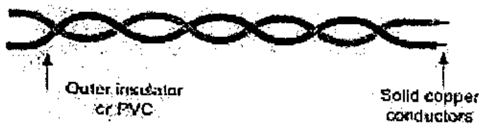
2.2.3 ตัวกลางส่งข้อมูลแบบมีสายนำทาง (Guided Media)

ตัวกลางส่งข้อมูลแบบมีสายซึ่งเปรียบเสมือนกับท่อส่ง ด้วยการส่งข้อมูลผ่านท่อไปยังอุปกรณ์ต่างๆ ตัวอย่าง ตัวกลางที่ใช้ส่งข้อมูลแบบมีสาย เช่น สายเคเบิลชนิดต่างๆไม่ว่าจะเป็นสายคู่บิดเกลียว สายโคแอกเซียล หรือสายใยแก้วนำแสง โดยสัญญาณจะเดินทางไปตามตัวกลางที่เชื่อมต่อโดยตรง แต่ก็มีข้อจำกัดทางกายภาพของตัวกลางส่งข้อมูลแต่ละชนิด เช่น ในระบบเครือข่ายท้องถิ่น สายใยแก้วนำแสงสามารถเชื่อมโยงระยะทางไกลได้หลายไมล์หรือหลายกิโลเมตร ในขณะที่สายคู่บิดเกลียวมีข้อจำกัดด้านการเชื่อมโยงในระยะทางที่สั้นกว่ามาก คือเพียงระยะทาง 100 เมตรเท่านั้น ซึ่งหากมีความต้องการเชื่อมโยงในระยะไกลกว่านี้ก็จำเป็นต้องใช้อุปกรณ์ที่เรียกว่า เครื่องทวนสัญญาณเพื่อให้สัญญาณสามารถส่งทอดไปในระยะทางไกลออกไปได้

1. สายคู่เกลียว (Twisted-Pair Cable)

สายคู่เกลียวประกอบด้วยสายทองแดงที่หุ้มด้วยฉนวนพลาสติก หลังจากนั้นก็จะสายสองเส้นดังกล่าวมาถักกันเป็นเกลียวคู่ โดยสายคู่หนึ่งก็จะใช้สำหรับการสื่อสารหนึ่งช่องทาง จำนวนคู่ที่เกิดจากการนำสายสองเส้นมาถักกันเป็นเกลียว ซึ่งอาจมีจำนวนหลายๆคู่ที่นำมารวบเข้าด้วยกันและหุ้มด้วยเปลือกหรือฉนวนภายนอก เช่น สายคู่เกลียวที่ใช้กับเครือข่ายท้องถิ่น ภายในฉนวนห่อหุ้มจะมีจำนวนสาย 4 คู่ด้วยกัน และหากเป็นสายที่นำไปใช้กับการส่งข้อมูลในระยะไกลๆ จำนวนสายภายในอาจมีมากกว่าหนึ่งร้อยคู่ คุณลักษณะสำคัญต่างๆ เช่น มีน้ำหนักเบา และง่ายต่อการติดตั้ง จึงถูกใช้งานอย่างกว้างขวาง เช่น สายเชื่อมต่อจากเครื่องโทรศัพท์ไปสู่ชุมสายโทรศัพท์ก็มักจะเป็นสายคู่ตีเกลียว และมีข้อจำกัด เช่น มีความเร็ว

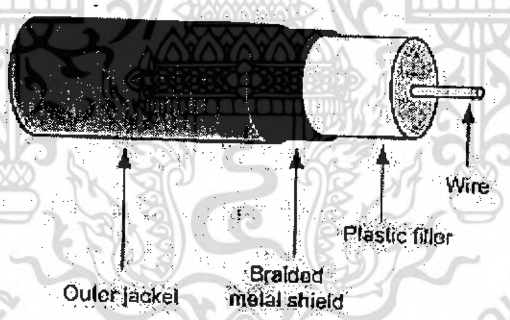
จำกัดและมักใช้งานกับระยะทางสั้นๆ ในกรณีเป็นสายคู่เกลียวชนิดไม่มีชีลด์ก็มักจะไวต่อสัญญาณรบกวนภายนอก



รูปที่ 2.3 สายคู่บิดเกลียว

2. สายโคแอกเชียล (Coaxial Cable)

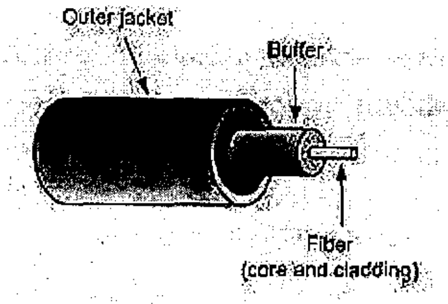
สายโคแอกเชียลจะมีช่วงคยวามถี่หรือแบนด์วิดท์ที่สูงกว่าสายคู่บิดเกลียว และจะมีสายซึ่งมักทำด้วยทองแดงอยู่แกนกลาง ซึ่งสายทองแดงดังกล่าวจะถูกห่อหุ้มด้วยพลาสติก จากนั้นก็จะมีชีลด์ห่อหุ้มอีกชั้นหนึ่งเพื่อป้องกันสัญญาณรบกวน และหุ้มด้วยเปลือกนอกอีกชั้นหนึ่ง จึงทำให้สายโคแอกเชียลนี้เป็นสายที่สามารถป้องกันสัญญาณรบกวนจากคลื่นแม่เหล็กไฟฟ้าได้ดี คุณลักษณะสำคัญต่างๆ เช่น เชื่อมต่อได้ในระยะทางไกล และสามารถป้องกันสัญญาณรบกวนได้ดี แต่ข้อเสียก็ คือ มีราคาแพง สายมีขนาดใหญ่จึงทำให้การติดตั้งเป็นไปได้ยาก



รูปที่ 2.4 โครงสร้างรายละเอียดของสายโคแอกเชียล

3. สายใยแก้วนำแสง (Optical Fiber)

สายใยแก้วนำแสงหรือสายใยแก้วนำแสง เป็นสายที่มีลักษณะโปร่งแสง มีรูปทรงกระบอกในตัน โดยจะมีเส้นผ่านศูนย์กลางประมาณ 125 ไมครอน สายใยแก้วนำแสงจะเป็นแก้วบริสุทธิ์ ซึ่งนำวัตถุดิบมาจากทรายและปนด้วยสารบางอย่างเพื่อให้แก้วมีค่าดัชนีหักเหของแสงตามที่ต้องการ เนื่องจากข้อมูลที่เดินทางผ่านสายใยแก้วนำแสงจะอยู่ในรูปแบบของแสง ดังนั้นข้อมูลที่อยู่ในรูปแบบของสัญญาณไฟฟ้าจึงจำเป็นต้องแปลงให้อยู่ในรูปแบบของแสงเสียก่อน จากนั้นก็จะส่งออกไปเป็นพัลส์ของแสงผ่านสายใยแก้วนำแสง คุณลักษณะสำคัญต่างๆ เช่น มีอัตราค่าลดทอนของสัญญาณต่ำ ไม่มีการรบกวนของสัญญาณไฟฟ้า และมีแบนด์วิดท์สูงมาก อีกทั้งมีขนาดเล็ก น้ำหนักเบา มีความทนทานสูง แต่ข้อเสียก็คือเส้นใยแก้วมีความเปราะบาง แตกหักง่าย และมีราคาสูง



รูปที่ 2.5 โครงสร้างรายละเอียดของสายใยแก้วนำแสง

2.2.4 ความสามารถสูงสุดในการส่งสัญญาณของช่องสัญญาณ (Maximum Channel Capacity)

นอกจากคุณภาพของสัญญาณดิจิตอลที่ส่งหรืออัตราการส่งข้อมูล (บิทต่อวินาที) ของสัญญาณที่ส่งจะขึ้นแบนด์วิดท์ของสัญญาณดังที่กล่าวมาแล้ว ยังขึ้นอยู่กับกำลังในการส่งสัญญาณ ตลอดจนสัญญาณรบกวน ในช่องสัญญาณด้วย ซึ่งปกติแล้วหากเพิ่มกำลังในการส่งสัญญาณจะทำให้สามารถเพิ่มความเร็วของการส่งข้อมูล ตลอดจนเพิ่มระยะทางของการส่งสัญญาณได้ แต่หากใช้กำลังการส่งสัญญาณที่สูงเกินอาจทำลายอุปกรณ์ของระบบสื่อสารได้ หรืออาจเสียค่าใช้จ่ายสูง ซึ่งความสัมพันธ์ของแบนด์วิดท์ กำลังส่งและสัญญาณรบกวน เป็นไปตามกฎของแชนนอน (Shannon) ดังต่อไปนี้

$$C = W\log_2\left(1+\frac{S}{N}\right) \tag{2.1}$$

C คือความสามารถสูงสุดของการส่งข้อมูล (bps) สำหรับสัญญาณข้อมูลดิจิตอลใดๆ ผ่านช่องสัญญาณหนึ่ง

W คือขนาดของแบนด์วิดท์

S/N คืออัตราส่วนระหว่างกำลังในการส่งสัญญาณกับสัญญาณรบกวน (Signal to noise ratio)

2.3 สัญญาณรบกวน

2.3.1 คุณสมบัติทั่วไปของสัญญาณรบกวน

ในธรรมชาติจะมีปริมาณทางกายภาพหลายอย่างที่มีการเปลี่ยนแปลงอยู่ตลอดเวลา โดยหากฎเกณฑ์ที่แน่นอนไม่ได้ ในเรื่องของสัญญาณไฟฟ้าก็เช่นกัน สัญญาณบางชนิดเกิดการเปลี่ยนแปลงอยู่ตลอดเวลาตามธรรมชาติและอาจจะมารบกวนสัญญาณข่าวสารที่ต้องการให้เกิดความผิดเพี้ยน หรือไม่ชัดเจนไปได้ เราเรียกสัญญาณเช่นนี้ว่า สัญญาณรบกวน (noise) ถ้าจะกล่าวโดยทั่วไป สัญญาณรบกวนก็คือสัญญาณที่เราไม่พึงปรารถนา ไม่ว่าจะเกิดขึ้นตามธรรมชาติ หรืออาจจะเกิดขึ้นจากการกระทำของมนุษย์ โดยความจงใจหรือไม่จงใจก็ตาม ถ้าหากว่าเป็นสัญญาณที่เราไม่พึงต้องการซึ่งไม่เกี่ยวข้องกับสัญญาณที่เราต้องการแล้ว เราจะจัดว่ามันเป็นสัญญาณรบกวนทั้งสิ้น ตัวอย่างของสัญญาณรบกวนอันอาจเกิดจากการกระทำของมนุษย์ (man made noise) ได้แก่ สัญญาณจากการจุดระเบิดของหัวเทียนเครื่องยนต์ สัญญาณที่เกิดจากเครื่องใช้ไฟฟ้า เช่น สว่าน หรือเลื่อยไฟฟ้า เป็นต้น สัญญาณรบกวนตามธรรมชาติ

(natural noise) ได้แก่ สัญญาณอันเนื่องจากอุณหภูมิ (thermal noise) และสัญญาณรบกวนจากระบบสุริยะ (solar noise) เป็นต้น

เนื่องจากสัญญาณรบกวนนั้นโดยทั่วไปมีคุณสมบัติที่คาดเดาอะไรโดยแน่นอนไม่ได้ (random) กล่าวคือ คุณสมบัติของสัญญาณรบกวนโดยทั่วไปมักจะเป็นสัญญาณสุ่ม ดังนั้นการที่จะบอกถึงคุณสมบัติอะไรเกี่ยวกับสัญญาณรบกวนนั้น จึงบอกได้แต่คุณสมบัติที่เป็นค่าเชิงสถิติเท่านั้น จากนั้นจึงอาศัยคุณสมบัติเหล่านั้นมาเป็นเครื่องเชื่อมโยงเกี่ยวเนื่อง เพื่ออธิบายเกี่ยวกับคุณสมบัติทางกายภาพของสัญญาณรบกวนนั้นคุณสมบัติทางสถิติที่เราควรรู้ไว้ เมื่อเริ่มศึกษาถึงคุณสมบัติของสัญญาณรบกวนนั้นได้แก่ ค่าเฉลี่ยในลักษณะต่างๆเช่น ค่าเฉลี่ย (average value) หรือค่ามัชฌิม (mean value) ค่ากำลังสองเฉลี่ย (mean square value) และค่าความแปรปรวน (variance) เป็นต้น

ถ้าสมมุติให้ $n(t)$ เป็นฟังก์ชันของเวลาที่ใช้แทนสัญญาณรบกวน ค่าเฉลี่ยของสัญญาณรบกวนในเชิงสถิติ ตามที่ได้อ้างถึงมาแล้วนั้น มีการนิยามดังต่อไปนี้

2.3.1.1 ค่าเฉลี่ยของสัญญาณรบกวน

ค่าเฉลี่ยของสัญญาณรบกวน $n(t)$ เขียนแทนด้วยสัญลักษณ์ $\overline{n(t)}$ มีนิยามดังนี้คือ

$$\overline{n(t)} \equiv \lim_{T\to\infty}\frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} n(t)\,dt \qquad (2.2)$$

ในที่นี้การใช้เครื่องหมายเส้นขีดเหนือค่าฟังก์ชันใด เป็นเครื่องหมายแสดงว่า นั่นคือค่าเฉลี่ยของฟังก์ชันนั้นต่อเวลา ค่าเฉลี่ย $\overline{n(t)}$ นี้บอกให้เรารู้ถึงคุณสมบัติทางกายภาพของสัญญาณรบกวน $n(t)$ นั้นว่ามีระดับไฟตรงที่เป็นค่าเฉลี่ยของสัญญาณรบกวนนั้นอยู่เป็นค่าเท่าใด

ในการหาค่าเฉลี่ยทางปฏิบัติ ค่าระยะเวลา T ใน (2.2) นั้นจะใช้เพียงค่าระยะเวลาที่ยาวนานค่าหนึ่งเท่านั้น ซึ่งถ้าค่าระยะนี้ยาวนานพอสมควรแล้ว ค่า $\overline{n(t)}$ ที่หามาได้ก็จะใช้ประมาณบอกถึงค่าเฉลี่ยตาม (2.2) ได้ใกล้เคียงกับค่าทางทฤษฎี

2.3.1.2 ค่ากำลังสองเฉลี่ยของสัญญาณรบกวน

ค่ากำลังสองเฉลี่ยของสัญญาณรบกวน $\overline{n^2(t)}$ หาได้ดังนี้

$$\overline{n^2(t)} = \lim_{T\to\infty}\frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \left|n(t)\right|^2 dt \qquad (2.3)$$

ค่าเฉลี่ยตาม (2.3) นี้ แสดงถึงปริมาณทางกายภาพ คือ ค่ากำลังเฉลี่ย (Average power) หรือค่ากำลังประสิทธิผล (Effective power) ทั้งหมดของสัญญาณรบกวนนั้นว่ามีค่าเท่าไร

2.3.1.3 ค่าความแปรปรวนของสัญญาณรบกวน

ค่าความแปรปรวนของสัญญาณรบกวน σ_n^2 หาได้จากนิยามดังต่อไปนี้คือ

$$\sigma_n^2 = \lim_{T\to\infty}\frac{1}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} \left\{n(t)-\overline{n(t)}\right\}^2 dt \qquad (2.4)$$

เนื่องจาก $n(t)-\overline{n(t)}$ นั้นแสดงถึงส่วนของ $n(t)$ ที่เบี่ยงเบนไปจากค่าเฉลี่ย $\overline{n(t)}$ ดังนั้นค่า $\left\{n(t)-\overline{n(t)}\right\}$ จึงมีความหมายแทนปริมาณส่วนที่เป็นสลับของสัญญาณรบกวน $n(t)$ ที่ขี่ซ้อนอยู่บนค่าระดับไฟตรงเฉลี่ย $\overline{n(t)}$ ด้วยเหตุนี้ ค่าความแปรปรวนของสัญญาณรบกวน σ_n^2 จึงแสดงให้เรารู้ถึงค่ากำลังเฉลี่ยของส่วนที่เป็นไฟสลับของสัญญาณรบกวน $n(t)$ และค่าความเบี่ยงมาตรฐาน σ_n ของสัญญาณรบกวน $n(t)$ จะบอกให้เรารู้ถึงค่าอาร์เอ็มเอส (RMS) ของสัญญาณรบกวน $n(t)$ นั้น

ค่าทางสถิติตาม (2.2) ถึง (2.4) นั้นมีความสัมพันธ์ดังต่อไปนี้

$$\overline{n^2(t)} = \sigma_n^2 + \left\{\overline{n(t)}\right\}^2 \qquad (2.5)$$

ตามความสัมพันธ์ (2.5) นี้สามารถพิสูจน์ได้โดยอาศัยการบวก $\overline{n(t)}$ เข้าไปกับ $n(t)$ แล้วลบ $\overline{n(t)}$ ออกพร้อมๆกัน ซึ่งจะไม่ทำให้ค่าของ $n(t)$ เปลี่ยนไป แล้วอาศัย (2.3) และทำการกระจายพจน์ จะได้ผลลัพธ์ดังต่อไปนี้ คือ

$$\begin{aligned}\overline{n^2(t)} &= \overline{\left[\left\{n(t)-\overline{n(t)}\right\}+\overline{n(t)}\right]^2} \\ &= \overline{\left\{n(t)-\overline{n(t)}\right\}^2 + 2\overline{n(t)}\left\{n(t)-\overline{n(t)}\right\} + \left\{\overline{n(t)}\right\}^2} \\ &= \overline{\left\{n(t)-\overline{n(t)}\right\}^2} + 2\overline{n(t)}\,\overline{\left\{n(t)-\overline{n(t)}\right\}} + \left\{\overline{n(t)}\right\}^2 \qquad (2.6)\end{aligned}$$

เนื่องจาก $\overline{\left\{n(t)-\overline{n(t)}\right\}}$ มีค่าเท่ากับศูนย์ เพราะฉะนั้นโดยอาศัยคำจำกัดความของ σ_n^2 ตาม (2.4) จะทำให้สามารถสรุปผลจากสมการ (2.6) ได้ว่าคือสมการ (2.5)

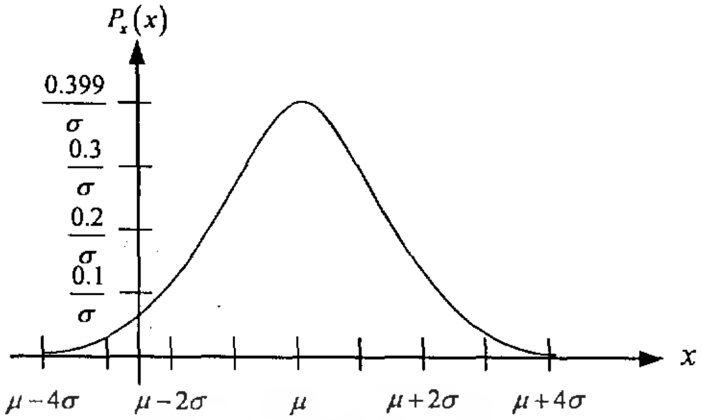
สมการ (2.5) มีความหมายในทางกายภาพว่า กำลังสองทั้งหมดของสัญญาณรบกวนมีค่าเท่ากับกำลังสองเฉลี่ยของส่วนที่เป็นไฟสลับของสัญญาณรบกวน รวมกับกำลังสองของส่วนที่เป็นไฟตรงของสัญญาณรบกวนนั้น

2.3.2 สัญญาณรบกวนแบบเกาส์เซียน

สำหรับสัญญาณรบกวนแบบเกาส์เซียนนั้น จะเป็นรูปแบบที่มีความสำคัญสำหรับการคำนวณเกี่ยวกับระบบสื่อสาร เนื่องมาจากสัญญาณรบกวนแบบ Thermal noise ที่เกิดขึ้นในระบบสื่อสารนั้น จะมีลักษณะในการเกิดที่คล้ายกับรูปแบบของฟังก์ชันเกาส์เซียน โดยที่กระบวนการสุ่มแบบเกาส์เซียนนั้นเป็นกระบวนการสุ่มแบบต่อเนื่อง ซึ่งมีลักษณะของพีดีเอฟ (Probability Density Function) ที่มีลักษณะดังสมการต่อไปนี้

$$P_x(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (2.7)$$

โดยในที่นี้ μ และ σ^2 คือค่าเฉลี่ยและค่าความแปรปรวนของพีดีเอฟ (ตัวแปรสุ่มใต้พีดีเอฟ) ตามลำดับ ค่า $\frac{1}{\sigma\sqrt{2\pi}}$ เป็นค่าน้ำหนักที่ใช้ถ่วงเพื่อทำให้พื้นที่ใต้เส้นโค้งของพีดีเอฟ มีค่าเท่ากับ 1 รูปกราฟของพีดีเอฟแบบเกาส์เซียนดังแสดงในรูปที่ 2.6



รูปที่ 2.6 แสดงพีดีเอฟแบบเกาส์เซียน

2.4 อิควอไลเซชัน (Equalization)

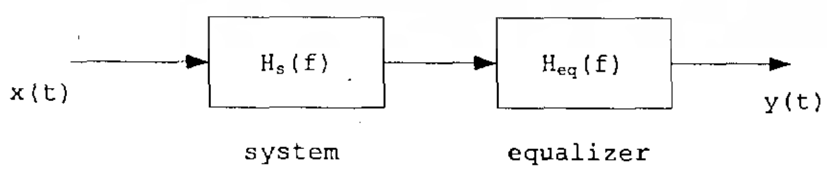
จากปัญหาที่เกิดขึ้นภายในช่องสัญญาณ ในการส่งสัญญาณสื่อสารระบบสัญญาณดิจิตอลดังกล่าว มีผลทำให้เกิดความผิดพลาดของบิทข้อมูลที่รับได้ทางภาครับ เมื่อผ่านกระบวนการดึงสัญญาณคืนกลับมา วิธีการหนึ่งที่สามารถนำมาใช้ในการจัดการกับปัญหาต่างๆก็คือ การใช้อิควอไลเซอร์ (Equalizer) ซึ่งมีหน้าที่ในการปรับแก้ไขรูปสัญญาณ เพื่อให้การดึงสัญญาณเดิมกลับคืนมามีความถูกต้องมากขึ้น เนื่องจากคุณลักษณะของช่องสัญญาณที่มีการเปลี่ยนแปลงไปตามเวลา ดังนั้นโดยปกติแล้ว อิควอไลเซอร์ที่ใช้จะสามารถปรับตัวได้ เพื่อให้เหมาะสมกับสภาพของช่องสัญญาณที่มักจะมีการเปลี่ยนแปลงอยู่เสมอ เรียกว่า อิควอไลเซอร์แบบปรับค่าได้ (Adaptive Equalizer)

ดังนั้นอิควอไลเซอร์ ก็คือวงจรที่ใช้ชดเชยการผิดเพี้ยนของสัญญาณให้กลับไปมีรูปเดิม ดังนั้นถ้าหากจะพิจารณาในโดเมนความถี่ ผลตอบสนองอิมพัลส์ของวงจรอิควอไลเซอร์จะต้องสัมพันธ์กับผลตอบสนองอิมพัลส์ของระบบเชิงเส้นก่อนหน้านั้น ในลักษณะที่ทำให้ผลตอบสนองอิมพัลส์รวมมีคุณสมบัติดังนี้

$$H_s(f)H_{eq}(f) = Ke^{-j2\pi f t_d} \tag{2.8}$$

โดยที่ $H_s(f)$ คือ ผลตอบสนองอิมพัลส์ของระบบ

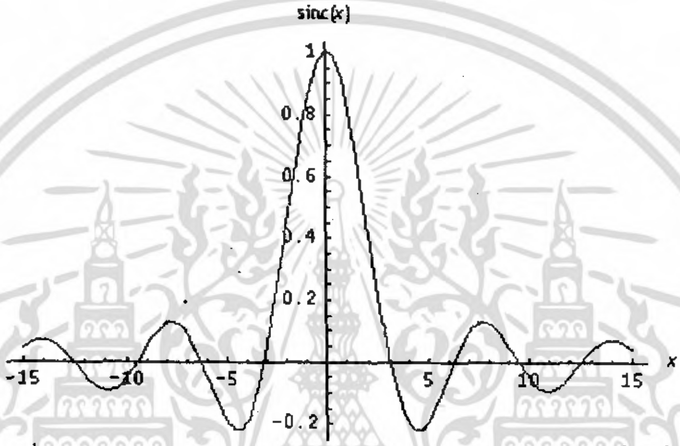
$H_{eq}(f)$ คือ ผลตอบสนองอิมพัลส์ของอิควอไลเซอร์



รูปที่ 2.7 แสดงการใช้อิควอไลเซอร์แก้ปัญหาการผิดเพี้ยนของสัญญาณ

สำหรับอิควอไลเซอร์ที่สมบูรณ์แบบจะต้องเป็นจริงที่ทุกๆความถี่ที่ $X(f) \neq 0$ ซึ่งจากรูปที่ 2.7 แสดงการส่งผ่านสัญญาณที่มีอิควอไลเซอร์ กรณีที่สัญญาณขาเข้ามีแบนด์วิดท์ และ $H_s(f)$ ของระบบไม่

ซับซ้อน การสร้างอิควอไลเซอร์ให้ได้ $H_{eq}(f)$ ก็อาจทำได้ไม่ยากนัก แต่อิควอไลเซอร์ที่ได้ก็มักจะบั่นทอนกำลังของสัญญาณไปอีกไม่น้อย ซึ่งเป็นที่หลีกเลี่ยงไม่ได้ ในกรณีที่สัญญาณขาเข้ามีแบนด์วิดท์กว้าง เช่น เป็นสัญญาณรูปพัลส์สี่เหลี่ยมอิควอไลเซอร์ชนิดสมบูรณ์แบบก็สร้างได้ยากขึ้น ในกรณีเช่นนี้มักจะทำการสร้างอิควอไลเซอร์ให้จัดการกับสเปกตรัมส่วนใหญ่ของสัญญาณได้ เช่น สามารถจัดการกับ 90% ของกำลังของสัญญาณก็อาจจะเพียงพอ เพราะการผิดเพี้ยนก็จะลดลงไปมาก จากรูปที่ 2.8 แสดงความหนาแน่นสเปกตรัมพลังงานของสัญญาณพัลส์สี่เหลี่ยม ดังนั้นในกรณีทั่วไปเรามักจะทำตามลักษณะที่กล่าวมานี้ นอกจากนี้เราอาจจะช่วยให้การออกแบบของอิควอไลเซอร์ง่ายลงโดยส่งสัญญาณต้นทางให้มีแบนด์วิดท์แคบลง โดยใช้วงจรกรองความถี่ช่วย



รูปที่ 2.8 แสดงความหนาแน่นสเปกตรัมพลังงานของพัลส์สี่เหลี่ยม

หลักการในการทำงานของอิควอไลเซอร์แบบปรับค่า จะแบ่งการส่งสัญญาณออกเป็น 2 ช่วงสลับกันตลอด ในช่วงแรกภาคส่งจะส่งชุดบิทที่เรียกว่าลำดับฝึกฝน (Training sequence) ไปให้ภาครับซึ่งรูปแบบของชุดบิทที่ว่านี้ภาครับจะทราบค่าอยู่ล่วงหน้าแล้ว ชุดบิทเหล่านี้อาจจะมีรูปแบบตายตัวหรืออาจจะเป็น pseudorandom binary ก็ได้ ในช่วงที่ภาครับได้รับชุดบิทดังกล่าวภาครับก็จะทำการคำนวณและปรับค่าสัมประสิทธิ์ของวงจรกรองความถี่ ในอิควอไลเซอร์ที่เหมาะสมสำหรับจัดการกับสภาพของช่องสัญญาณในขณะนั้น เมื่อภาคส่งสิ้นสุดการส่งชุดลำดับฝึกฝนก็จะเข้าสู่ช่วงที่สอง ซึ่งเป็นช่วงที่ภาคส่งจะทำการส่งบิทข้อมูลของผู้ใช้ไป ในช่วงนี้ภาครับจะใช้อิควอไลเซอร์ที่มีค่าสัมประสิทธิ์ที่คำนวณไว้มาใช้ในการดึงสัญญาณข้อมูลกลับมาเพื่อให้ได้ค่าที่ถูกต้องมากที่สุด

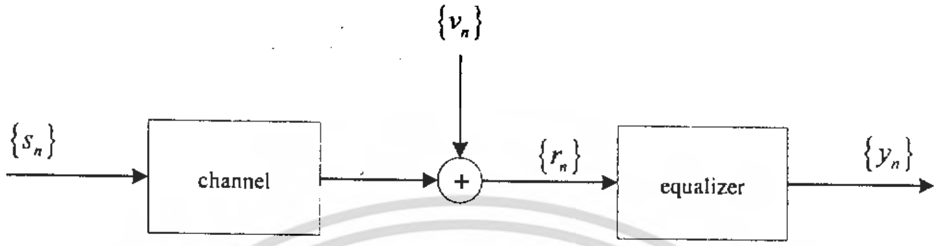
จะเห็นว่าคุณลักษณะของอิควอไลเซอร์จะมีการเปลี่ยนแปลงตามเวลา เพื่อให้เหมาะสมกับสภาพของช่องสัญญาณซึ่งมักมีการเปลี่ยนแปลงอยู่ตลอดเวลา และสามารถแก้ปัญหาของการแทรกสอดระหว่างสัญลักษณ์ได้อย่างมีประสิทธิภาพแต่ปัญหาหลักของของอิควอไลเซอร์แบบปรับค่า คือการหาอัลกอริทึม (Algorithm) ที่จะปรับค่าตัวแปรได้ดีในสถานการณ์ที่ความรู้เกี่ยวกับคุณสมบัติของสัญญาณ

2.5 ชนิดของอิควอไลเซอร์

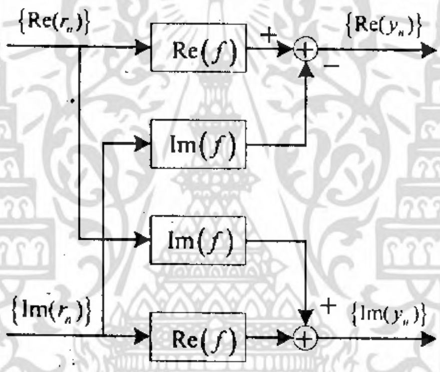
อิควอไลเซอร์สามารถแบ่งออกเป็น 2 ประเภทใหญ่ๆ คือ

2.5.1 อิควอไลเซอร์แบบเชิงเส้น (Linear equalization)

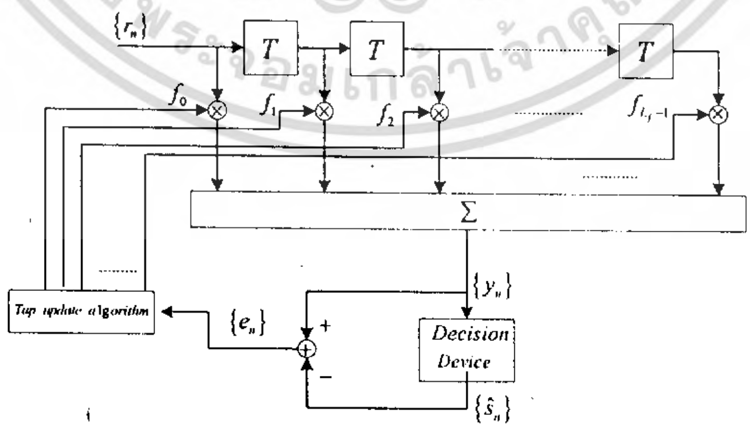
อิควอไลเซอร์แบบเชิงเส้นสามารถถูกใช้ในรูปแบบของจำนวนเชิงซ้อนได้แสดงได้ดังรูปที่ 2.9



รูปที่ 2.9 แสดงอิควอไลเซอร์แบบเชิงเส้น

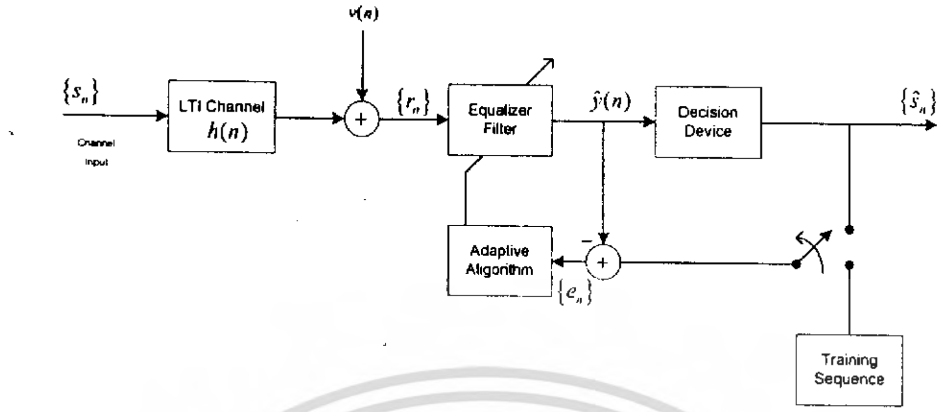


รูปที่ 2.10 แสดงการใช้อิควอไลเซอร์ที่เป็นจำนวนเชิงซ้อน สำหรับสัญญาณ QPSK โดยจะมีการแบ่งส่วนที่เป็นจำนวนจินตภาพและจำนวนจริง



รูปที่ 2.11 แสดงอิควอไลเซอร์แบบเชิงเส้นใน decision-directed mode ซึ่งจะถูกใช้เหมือน Complex FIR filter

2.5.1.1 อิควอไลเซอร์แบบเชิงเส้นแบบเทรน

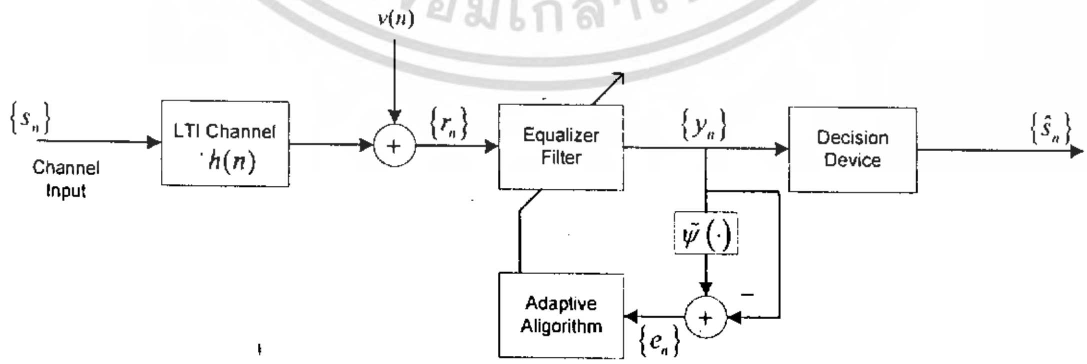


รูปที่ 2.12 โครงสร้างพื้นฐานของระบบอิควอไลเซอร์แบบเทรน

เมื่อพิจารณารูปที่ 2.12 ซึ่งแสดงวงจรอิควอไลเซอร์ปรับค่าได้แบบเทรนซึ่งเราจะต้องทราบถึงผลตอบสนองของสัญญาณที่ต้องการ (Desired response signal) เพื่อนำมาใช้ในการปรับค่าสัมประสิทธิ์ซึ่งถูกนำมาเปรียบเทียบกับผลตอบสนองของสัญญาณที่ต้องการนี้ และใช้ข้อมูลจากการเปรียบเทียบดังกล่าวนี้มาใช้เพื่อนำไปปรับปรุงสมรรถนะของระบบ นอกจากนี้ยังสามารถใช้เป็นลำดับฝึกฝน (training sequence) ให้กับส่วนป้อนกลับของวงจรกรองอีกด้วย ซึ่งวงจรกรองจะนำข้อมูลที่ได้ไปปรับค่าสัมประสิทธิ์ของวงจรอิควอไลเซอร์โดยอาศัยอัลกอริทึมมาตรฐานเช่น อัลกอริทึมค่าเฉลี่ยกำลังสองน้อยสุดหรืออัลกอริทึมอาร์แอลเอส กรณีอัลกอริทึมค่าเฉลี่ยกำลังสองน้อยสุดที่มีการใช้ลำดับฝึกฝนนั้นสามารถแสดงได้ดังนี้

$$h(n) = h(n-1) + 2\mu r(n)e^{*}(n) \tag{2.9}$$

2.5.1.2 อิควอไลเซอร์แบบเชิงเส้นแบบบอด



รูปที่ 2.13 โครงสร้างพื้นฐานของระบบอิควอไลเซอร์แบบบอด

โครงสร้างพื้นฐานของระบบอิควอไลเซอร์แบบบอดสามารถแสดงได้ตามรูปที่ 2.13 โดยมีส่วนประกอบที่สำคัญคือฟังก์ชัน scalar zero-memory nonlinear ψ ซึ่งใช้ในการกำเนิดผลตอบสนองของสัญญาณที่ต้องการ $\psi\left[\tilde{y}(n)\right]$ สำหรับอัลกอริทึมปรับค่าได้
เราจะสามารถเขียนค่าความผิดพลาดได้เป็น

$$e(n) = \psi\left[\tilde{y}(n)\right] - \tilde{y}(n) \qquad (2.10)$$

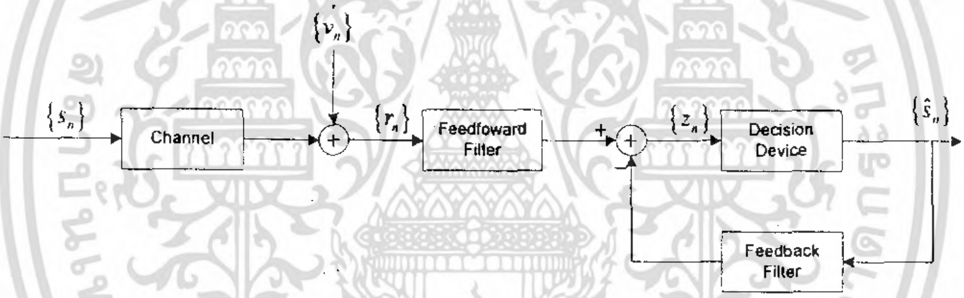
โดยที่

$$\tilde{y}(n) = \sum_{k=-L}^{L} h_k^*(n-1)r(n-k) \equiv h^H(n-1)r(n) \qquad (2.11)$$

$\tilde{y}(n)$ คือ สัญญาณเอาท์พุทของวงจรอิควอไลเซอร์ เกรเดียนร์อัลกอริทึมสำหรับอิควอไลเซอร์แบบบอดกำหนดได้ดังนี้

$$h(n+1) = h(n) + \mu r(n)e^*(n) \qquad (2.12)$$

2.5.2 อิควอไลเซอร์แบบไม่เป็นเชิงเส้น (Nonlinear equalizer)

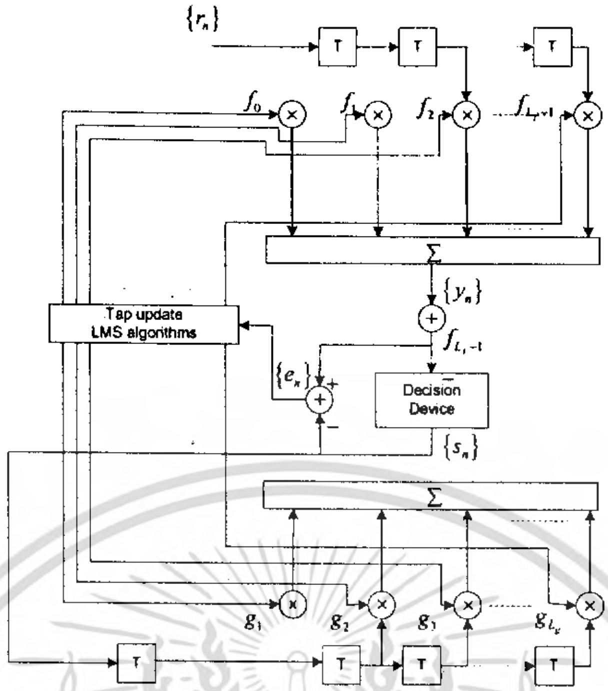


รูปที่ 2.14 โครงสร้างพื้นฐานของระบบอิควอไลเซอร์แบบไม่เป็นเชิงเส้น

จากรูปพบว่าอิควอไลเซอร์แบบไม่เป็นเชิงเส้นจะมีส่วนของวงจรป้อนกลับ (Feedback) ซึ่งจะทำการปรับค่าของส่วนตัดสินให้ทำงานได้อย่างแม่นยำยิ่งขึ้น

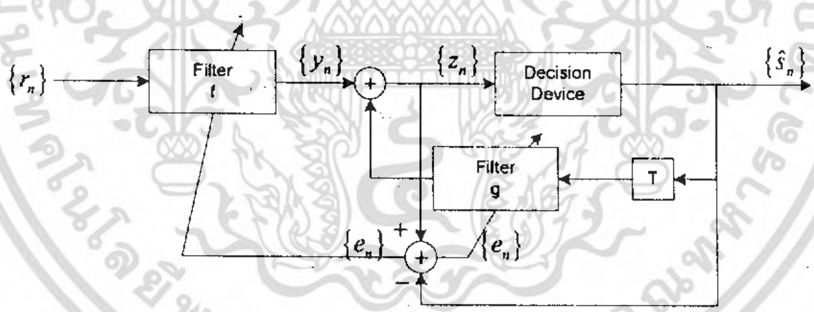
2.5.2.1 อิควอไลเซอร์แบบไม่เป็นเชิงเส้นแบบเทรน

จากรูปที่ 2.15 พบว่ามีส่วนที่เป็นวงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัด 2 ตัว โดยมีความยาวเป็น L_f และ L_g โดยในส่วนที่มีความยาวเป็น L_f จะอยู่ในส่วนของที่เป็นวงจรตรง และในส่วนที่มีความยาวเป็น L_g จะอยู่ในส่วนที่เป็นวงจรป้อนกลับ ซึ่งในที่นี้จะเป็นการใช้อัลกอริทึมค่าเฉลี่ยกำลังสองน้อยสุดหรืออัลกอริทึมแบบแอลเอ็มเอสในการปรับค่าสัมประสิทธิ์ในทั้งสองส่วน



รูปที่ 2.15 โครงสร้างพื้นฐานของระบบอิควอไลเซอร์แบบไม่เป็นเชิงเส้นแบบเทรน

2.5.2.2 อิควอไลเซอร์แบบไม่เป็นเชิงเส้นแบบบอด



รูปที่ 2.16 โครงสร้างพื้นฐานของระบบอิควอไลเซอร์แบบไม่เป็นเชิงเส้นแบบบอด

การทำงานจะมีลักษณะคล้ายกับแบบไม่เป็นเชิงเส้นแบบเทรน คือจะมีทั้งส่วนของวงจรส่วนตรงและวงจรส่วนป้อนกลับ แต่ในที่นี้เราจะใช้อัลกอริทึมสำหรับอิควอไลเซอร์แบบบอด (CMA: Constant Modulus algorithm) ในการปรับค่าสัมประสิทธิ์ซึ่งจะสามารถเขียนได้ว่า

$$\begin{aligned} h(n+1) &= h(n) + \mu \cdot e_n^{cma} \cdot r^*(n) \\ &= h(n) + \mu \cdot z_n\left(\left|z_n\right|^2 - \gamma\right) \cdot r^*(n) \end{aligned} \tag{2.13}$$

2.6 ทฤษฎีพื้นฐานของวงจรกรองความถี่แบบดิจิตอล

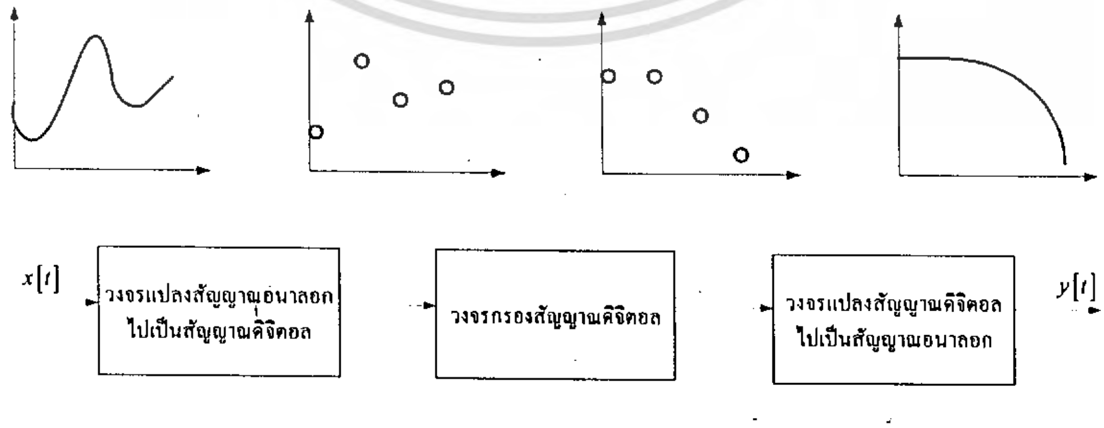
2.6.1 ความหมายของวงจรกรองความถี่แบบดิจิตอล

วงจรกรองความถี่แบบดิจิตอล คือ กระบวนการที่ไปดัดแปลงสเปกตรัมของสัญญาณ ให้มีสเปกตรัมเป็นไปตามข้อกำหนดที่ต้องการ ซึ่งอาจเป็นการเพิ่มค่า หรือลดทอนค่าขนาดของสัญญาณในแถบความถี่ที่กำหนดให้ ซึ่งในการวิเคราะห์และสังเคราะห์วงจรนั้นต้องใช้เครื่องมือพื้นฐานทางคณิตศาสตร์เข้าช่วย ดังนั้นเราจึงนิยมเรียกว่า วงจรกรองความถี่เชิงเลข

การที่วงจรกรองความถี่เชิงเลขมีการนำมาประยุกต์ใช้งานกันอย่างกว้างขวาง อาจมาจากข้อได้เปรียบหลายประการดังต่อไปนี้

1. ผลตอบสนองความถี่ของวงจรกรองความถี่สามารถออกแบบให้มีความใกล้เคียงกับผลตอบสนองความถี่ที่กำหนดให้ หรือผลตอบสนองความถี่ที่ต้องการได้ นอกจากนี้การออกแบบวงจรกรองความถี่ให้มีผลตอบสนองเชิงเส้นได้ง่าย
2. คุณสมบัติของวงจรกรองความถี่ที่ออกแบบและสร้างแล้วจะไม่ขยับเลื่อน (shift) ไปตามสภาพแวดล้อมหรือตามอุณหภูมิ หรือตามระยะเวลาการใช้งาน นอกจากนี้ยังสามารถใช้งานในย่านความถี่ต่ำได้เป็นอย่างดี
3. การประยุกต์ใช้งานเป็นวงจรกรองความถี่ที่ออกแบบปรับค่าได้ (Adaptive filter) ทำได้ง่าย
4. ผู้ออกแบบสามารถออกแบบโดยคำนึงถึงความยาวของคำ (Word length) ของตัวเลขฐานสองที่ต้องการใช้ และยังสามารถออกแบบให้มีผลตอบสนองความถี่ที่ต้องการได้
5. ในปัจจุบัน ถ้าพิจารณาในแง่มุมของเสถียรภาพของวงจรกรองความถี่ ความเชื่อถือได้ ราคาหรือขนาดของวงจรกรองความถี่แบบอนาลอก (Analog filter) หรือเรียกว่าวงจรกรองความถี่เชิงอนุมาน

วงจรกรองสัญญาณดิจิตอลสามารถเขียนอธิบายในรูปของบล็อกไดอะแกรมได้ดังรูปที่ 2.17 โดยสัญญาณอินพุทเป็นสัญญาณซึ่งเป็นสัญญาณอนาลอกจะถูกสุ่ม (Sampled) ด้วยช่วงเวลาที่ค่าคงที่ค่าหนึ่ง และสัญญาณที่ถูกสุ่มนี้จะเปลี่ยนให้อยู่ในรูปเลขฐานสองโดยการแปลงสัญญาณดิจิตอลหรือสัญญาณเชิงเลข (analog to digital converter)



รูปที่ 2.17 บล็อกไดอะแกรมของวงจรกรองสัญญาณดิจิตอล

หลังจากนั้นเลขฐานสองที่แทนสัญญาณอนาลอกที่เข้ามาทางอินพุทจะถูกกรองโดยวงจรกรองสัญญาณดิจิตอล การกรองจะเป็นการคำนวณทางตัวเลขซึ่งจะอาศัยวงจรที่ใช้ในระบบคอมพิวเตอร์ ได้แก่ ตัวบวก (Adder) ตัวคูณ (Multiplier) รีจิสเตอร์ (Shift register) และอุปกรณ์หน่วยความจำ (Memory Device) ต่อมาค่าเอาท์พุตที่ได้จากวงจรกรองสัญญาณดิจิตอลนี้จะถูกแปลงกลับเป็นสัญญาณอนาลอกอีกทีหนึ่งเป็นสัญญาณเอาท์พุตที่นำไปใช้งานได้

วงจรกรองสัญญาณดิจิตอลสามารถแบ่งได้เป็น 2 ประเภท ตามลักษณะของผลตอบสนองอิมพัลส์ ได้แก่

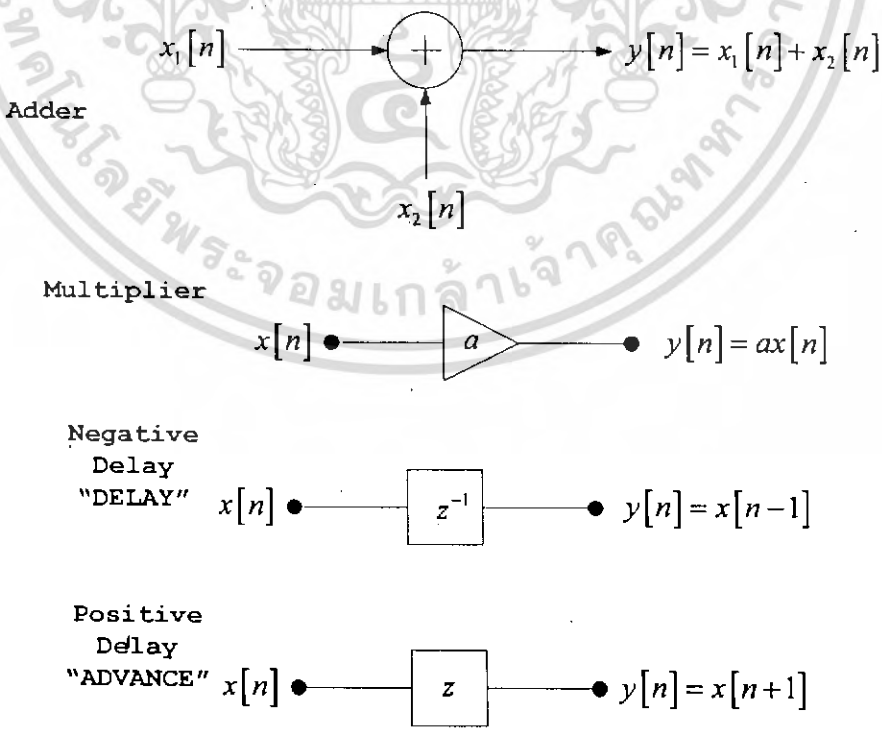
1. วงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัด
2. วงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์ไม่จำกัด

วงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัด มักเป็นตัวกรองที่ไม่มีการป้อนกลับ เป็นวงจรที่มีโครงสร้างง่ายๆและมีเสถียรภาพที่ดี แต่มีข้อเสียที่จะให้วงจรกรองที่มีอันดับสูง ถึงแม้จะต้องการให้มีลักษณะทางความถี่ที่ง่ายๆก็ตาม

วงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์ไม่จำกัด เป็นตัวกรองที่มีการป้อนกลับเป็นวงจรที่ใช้อันดับต่ำกว่าวงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัดที่ต้องการลักษณะทางความถี่เหมือนกันแต่การออกแบบจะยุ่งยากกว่า และมีปัญหาในเรื่องมีเสถียรภาพไม่ดีนัก

2.6.2 ส่วนประกอบที่สำคัญในวงจรกรองความถี่แบบดิจิตอล

วงจรกรองความถี่แบบดิจิตอลประกอบไปด้วยส่วนที่สำคัญ 3 ส่วน คือ การบวก, การคูณและการหน่วง (delay) ดังรูปที่ 2.18 การบวกและการคูณจะใช้แนวความคิดมาจากตัวเลขในหน่วยคอมพิวเตอร์ ส่วนการหน่วงจะทำให้การถึงข้อมูลในอนาคตมีค่าอย่างต่อเนื่อง



รูปที่ 2.18 ส่วนประกอบที่สำคัญในวงจรกรองความถี่แบบดิจิตอล

การหน่วงจะแบ่งออกเป็น 2 ส่วนคือ บวก (positive) และลบ (negative) โดยการหน่วงแบบบวกนี้เป็นอุปกรณ์ที่ทำหน้าที่บันทึกความจำของรีจีสเตอร์ (register) จะเก็บค่าได้ตามระยะเวลาที่กำหนดสำหรับการคำนวณครั้งต่อไป การหน่วงแบบบวกจะแทนค่าด้วย z^{-1} และสามารถอธิบายความสัมพันธ์ได้ด้วยการแปลงแซด การหน่วงแบบลบใช้แทนค่าต่อไป ในระดับสัญญาณแทนค่าด้วย z จะมีชนิดและการใช้งานที่เหมาะสม อย่างไรก็ตามการใช้งานก็ไม่สามารถใช้งานได้เสมอไป

วงจรกรองความถี่แบบดิจิตอล สามารถออกแบบให้มีค่าของการคูณที่แน่นอน และมีหลักการในการพิจารณาที่ไม่ยุ่งยากซับซ้อน

2.6.3 วงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัด หรือ FIR

คำว่า FIR ย่อมาจาก Finite Impulse Response ซึ่งแปลว่า ผลตอบสนองอิมพัลส์จำกัด นั่นคือ หากเราป้อนสัญญาณอิมพัลส์ให้กับวงจรนี้แล้ว สัญญาณตอบสนองจะมีค่าจำกัด โดยสัญญาณเอาท์พุดของระบบจะขึ้นอยู่กับสัญญาณอินพุทเท่านั้น จึงเรียกว่า วงจรกรองความถี่ไม่ป้อนกลับ (Non-recursive filter) หรือ ไม่ป้อนกลับเชิงเลข ซึ่งสามารถเขียนสมการได้ดังนี้

$$y(n) = \sum_{k=-\infty}^{\infty} b_k x(n-k) \tag{2.14}$$

โดยที่ b_k เป็นค่าคงที่ใดๆ ที่แทนค่าสัมประสิทธิ์ (Coefficient) ของวงจรกรองความถี่และในทางปฏิบัติค่า k จะมีค่าคงที่ ไม่ถึงกับมีค่าอนันต์ขึ้นกับอันดับของวงจรกรองความถี่ N ที่ต้องการใช้ จะได้

$$y(n) = \sum_{k=0}^{N-1} b_k x(n-k) \tag{2.15}$$

และจะได้สมการคอนโวลูชั่น (Convolution) เป็น

$$y(n) = \sum_{m=0}^{N-1} h(m)x(n-m) \tag{2.16}$$

เปลี่ยนค่าตัวแปรจะได้

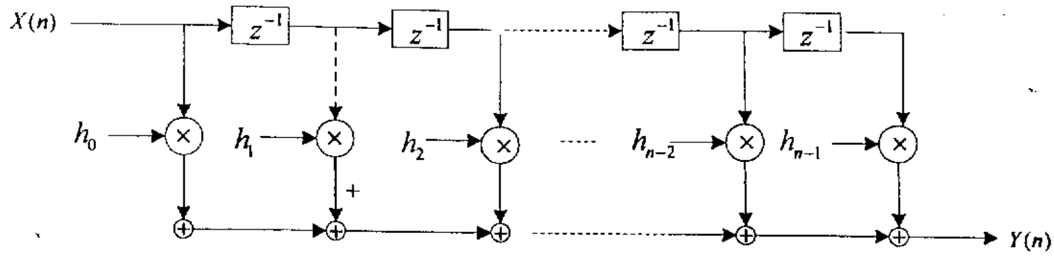
$$y(n) = \sum_{m=n}^{n-N-1} h(n-m)x(m) \tag{2.17}$$

เมื่อ $x(n)$ เป็นอินพุท และ $h(n)$ เป็นผลตอบสนองอิมพัลส์ลำดับที่ N (length-N impulse response) เมื่อนำมาประยุกต์ใช้งานกับการแปลงแซด จะได้ฟังก์ชันถ่ายโอน

$$H(z) = \sum_{n=0}^{N-1} h(n)z^{-n} \tag{2.18}$$

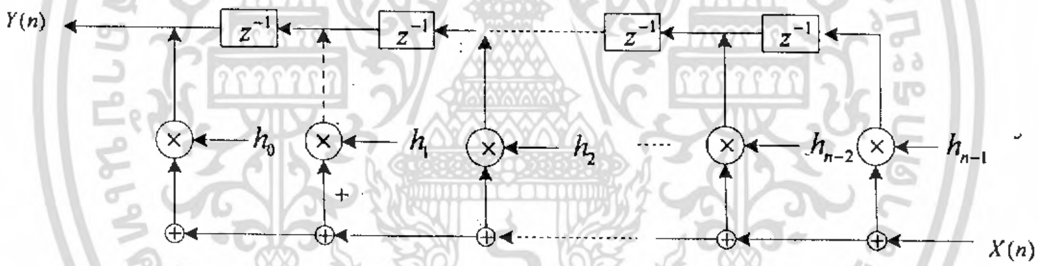
แทนค่า $z = e^{j\omega}$ จะได้ผลตอบสนองความถี่ของวงจรกรองความถี่ไม่ป้อนกลับเชิงเลข

$$H(z) = \sum_{n=0}^{N-1} h(n)e^{-j\omega n} \tag{2.19}$$

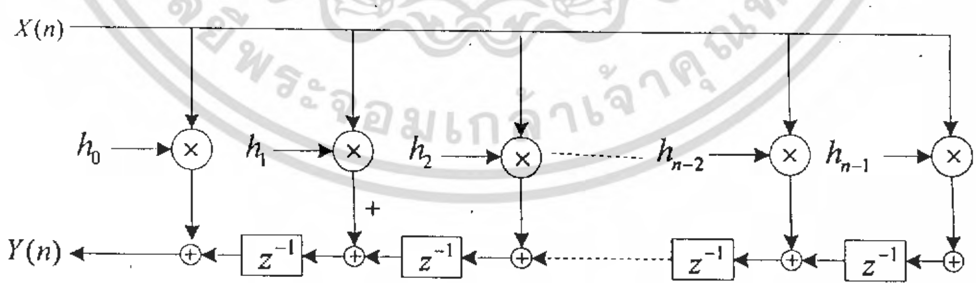


รูปที่ 2.19 โครงสร้างวงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัดรูปแบบตรง

จากรูปที่ 2.19 แสดงโครงสร้างรูปแบบตรงของงจรกรองความถี่ดิจิตอลแบบผลตอบสนองอิมพัลส์จำกัดและเนื่องจากโครงสร้างทรานสโพส จะเป็นการเปลี่ยนจากการเขียนทิศทางการไหลจากซ้ายไปขวา และการเขียนจากขวาไปซ้าย และเปลี่ยนจากอินพุทไปเป็นเอาท์พุท โดยจะสามารถแสดงได้จากรูปที่ 2.20(ก) และสามารถเขียนใหม่ได้เป็นดังรูปที่ 2.20(ข) ซึ่งจะพบว่ามีค่าฟังก์ชั่นถ่ายโอนเท่ากันกับโครงสร้างรูปแบบตรง เนื่องจากมีค่าสัมประสิทธิ์ทั้งตัวตั้งและตัวหารเท่ากันกับโครงสร้างแบบตรง ดังนั้นทำให้เราสามารถเลือกใช้รูปแบบที่ต่างกันนี้ให้เหมาะสมกับงานที่ต่างๆกันออกไปได้



(ก)



(ข)

รูปที่ 2.20 โครงสร้างวงจรกรองความถี่ดิจิตอลแบบ FIR รูปแบบทรานสโพสแบบตรง (TDF)

2.7 วงจรกรองความถี่แบบปรับตัวได้ (Adaptive Filter)

วงจรกรองความถี่ทั่วๆไป เป็นวงจรกรองความถี่แบบสัมประสิทธิ์คงที่ หรือแบบไม่แปรตามเวลา สำหรับวงจรกรองความถี่แบบปรับตัวได้ จะเป็นวงจรกรองความถี่แบบที่มีสัมประสิทธิ์ปรับเปลี่ยนตลอดเวลาโดยมีลักษณะการปรับค่าสัมประสิทธ์โดยอัตโนมัติ โดยใช้เงื่อนไขทางสถิติของสัญญาณและของโมเดลของสิ่งแวดล้อมที่สร้างขึ้นในการหาค่าสัมประสิทธิ์ ตัวกรองแบบปรับตัวได้นี้ทำให้ขอบเขตของการประยุกต์ใช้การประมวลผลสัญญาณดิจิตอลขยายวงออกไปกว้างมาก ตัวกรองความถี่แบบปรับตัวได้นี้จะมีประโยชน์อย่างมากในสภาวะที่เราไม่สามารถใช้ตัวกรองความถี่ดิจิตอลแบบธรรมดาได้หรือใช้ได้แต่ให้ผลไม่ดีนัก ซึ่งจะให้ผลที่ดีขึ้นถ้าใช้ตัวกรองความถี่แบบปรับตัวได้ ยกตัวอย่างเช่น

1) เมื่อสัญญาณรบกวนและสัญญาณที่ต้องการอยู่ในย่านความถี่เดียวกัน ซึ่งถ้าเราใช้ตัวกรองความถี่ที่มีค่าสัมประสิทธิ์คงที่ และต้องการให้สัญญาณรบกวนส่วนใหญ่หมดไป ก็จะทำให้สัญญาณที่ต้องการหายไปด้วย

2) เมื่อไม่รู้ลักษณะที่แน่นอนของสัญญาณรบกวน เช่น ไม่รู้ว่าสัญญาณรบกวนอยู่ในย่านความถี่ใดหรือเข้ามาในทิศทางใดในกรณีของสายอากาศแบบอะเรย์ เป็นต้น

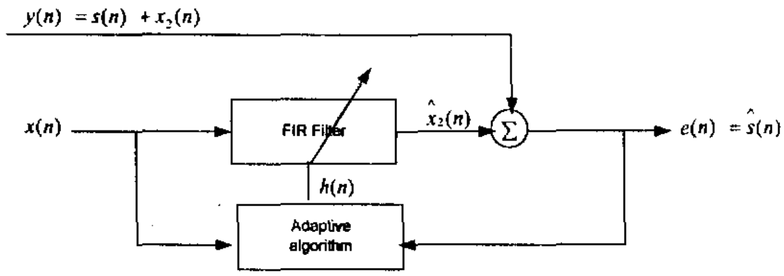
3) เมื่อสัญญาณที่ต้องการมีความผิดเพี้ยน ไม่รู้ลักษณะที่แน่นอน เช่น กรณีของโมเดมหรือโทรศัพท์มือถือที่มีการส่งสัญญาณผ่านช่องสัญญาณ สัญญาณที่ตัวรับได้รับจะถูกทำให้ผิดเพี้ยนไปด้วยฟังก์ชันถ่ายโอนของสัญญาณที่ไม่รู้ค่าแน่นอนและอาจมีค่าแปรตามเวลาด้วย เพราะฉะนั้นจึงต้องใช้ตัวกรองเพื่อทำการกรองเอาความผิดพลาดนี้ออกไป

การออกแบบตามแนวความคิดที่ว่า จะบรรจุกลุ่มตัวแปร (Parameter) ที่ปรับค่าได้และค่านั้นจะถูกกำหนดโดยอัตโนมัติขึ้นอยู่กับการประเมินค่าคุณสมบัติทางสถิติของสัญญาณที่เกี่ยวข้อง ดังนั้นทฤษฎีของวงจรกรองปรับตัวได้จะมีความสัมพันธ์ใกล้ชิดกับทฤษฎีวงจรกรองความถี่ดิจิตอลทั่วไป ในการออกแบบต้องการจะจัดกลุ่มของตัวแปรที่ดีที่สุดจากความรู้เกี่ยวกับคุณสมบัติของสัญญาณที่เกี่ยวข้อง ให้เหมาะสมกับบรรทัดฐานและความต้องการ ปัญหาหลักของวงจรกรองปรับตัวได้คือการหาอัลกอริทึม (Algorithm) ที่จะปรับค่าตัวแปรได้ดีในสถานการณ์ที่ความรู้เกี่ยวกับคุณสมบัติของสัญญาณ โดยทั่วไปนั้นวงจรกรองปรับตัวได้จะทำงานโดยใช้กระบวนการทำงานซ้ำๆ (Iterative) ในการกรองและปรับค่าตัวแปรไปในเวลาเดียวกัน วงจรกรองความถี่แบบปรับตัวได้มีสองลักษณะคือ

1. วงจรกรองปรับตัวได้แบบลูปเปิด (Open Loop) จะทำการเรียนรู้สถิติของสัญญาณที่เกี่ยวข้องและนำผลที่ได้ป้อนให้แก่อัลกอริทึมโดยไม่มีการป้อนกลับ (Non-Recursive) การกระทำลักษณะนี้เพื่อจะให้ได้การกรองที่ดีจะต้องการกลุ่มตัวแปรจำนวนมาก ทำให้ต้องสร้างฮาร์ดแวร์ที่ซับซ้อนซึ่งมีราคาแพง วงจรกรองพวกนี้ได้แก่ วงจรกรองปรับตัวได้แบบผลตอบสนองอิมพัลส์จำกัด

2. วงจรกรองปรับตัวได้แบบลูปปิด (Close Loop) จะทำการเรียนรู้สถิติของสัญญาณที่เกี่ยวข้องและปรับปรุงกลุ่มของค่าตัวแปรปัจจุบันจากสัญญาณที่เข้ามาใหม่ และสัญญาณเอาท์พุทที่ได้จากการทำงานรอบที่แล้ว การทำงานเช่นนี้จะเป็นลักษณะของการป้อนกลับ (Recursive) มีข้อดีคือต้องการกลุ่มตัวแปรน้อย ทำให้สามารถใช้ฮาร์ดแวร์ที่มีความซับซ้อนไม่มากนักได้ วงจรกรองความถี่พวกนี้ได้แก่ วงจรกรองปรับตัวได้แบบผลตอบสนองอิมพัลส์ไม่จำกัด

2.7.1 โครงสร้างพื้นฐานของวงจรกรองความถี่แบบปรับตัวได้



รูปที่ 2.21 โครงสร้างพื้นฐานของวงจรกรองความถี่แบบปรับตัวได้

สามารถอธิบายโครงสร้างพื้นฐานของวงจรกรองความถี่แบบปรับตัวได้ จากรูปที่ 2.21 ได้ ดังนี้

1. เราสามารถวัดสัญญาณมาได้ 2 สัญญาณ คือ x(n) และ y(n) โดยที่ y(n) เป็นสัญญาณที่ผสมกันระหว่าง s(n) และ $x_2(n)$ วัตถุประสงค์ของตัวกรองก็คือ ต้องการแยก s(n) ออกจาก $x_2(n)$ โดยไม่จำเป็นต้องรู้ย่านความถี่หรือลักษณะทางสถิติของสัญญาณทั้งสองล่วงหน้า

2. เงื่อนไขที่จำเป็น คือ s(n) ต้องไม่มีความสัมพันธ์กับ $x_2(n)$ ในทางสถิติ (statistical independent) หรือสัมพันธ์กันน้อยมาก เช่น เป็นสัญญาณเสียงที่มีแหล่งกำเนิด 2 แหล่ง ที่ไม่เกี่ยวข้องกัน ส่วน x(n) ซึ่งเป็นสัญญาณอีกสัญญาณหนึ่งที่วัดได้นั้น ต้องมีความสัมพันธ์กับ $x_2(n)$

3. เราจะนำ x(n) ผ่านตัวกรอง FIR ที่สร้างขึ้นซึ่งมีสัมประสิทธิ์ h(n) และอันดับเท่ากับ N ให้สัญญาณขาออกของตัวกรอง คือ $\hat{x}_2(n)$ ถ้าหากว่าเราสามารถปรับสัมประสิทธิ์ h(n) ให้มีค่าใกล้เคียงกับสัมประสิทธิ์ที่เป็นโมเดลระหว่าง x(n) กับ $x_2(n)$ ได้ $\hat{x}_2(n)$ ก็จะเป็นสัญญาณที่ใกล้เคียงกับ $x_2(n)$ และเมื่อนำ $\hat{x}_2(n)$ ลบออกจาก y(n) สัญญาณที่เหลืออยู่ซึ่งจะเรียกว่า สัญญาณความคลาดเคลื่อน e(n) ก็จะมีค่าใกล้เคียงกับ s(n) นั่นคือเราสามารถแยก $x_2(n)$ ออกจาก s(n) ได้

ตัวที่จะคำนวณหาสัมประสิทธิ์ h(n) ก็คือ อัลกอริทึมปรับตัวได้ ซึ่งรับค่า e(n) และ x(n) จากนั้นคำนวณ h(n) ป้อนให้กับตัวกรอง FIR และจะปรับค่า h(n) ไปเรื่อยๆ ทุกๆขั้นเวลา จนกระทั่ง h(n) ลู่เข้าสู่ค่าที่ถูกต้อง อัลกอริทึมในการปรับตัว (Adaptive Algorithm) มีด้วยกันหลากหลายวิธี เพื่อใช้ในการคำนวณหาค่าสัมประสิทธิ์นี้ การศึกษาถึงอัลกอริทึมเหล่านี้รวมอยู่ในสาขาของการประมวลผลสัญญาณดิจิตอลขั้นสูง ซึ่งบางครั้งก็เรียกว่า การประมวลผลสัญญาณทางสถิติ (statistical signal processing) หรือการประมวลผลสัญญาณแบบปรับตัวได้ (Adaptive signal processing) หรือทฤษฎีการประมาณค่า (estimation theory)

2.7.2 กรรมวิธีการพัฒนาอัลกอริทึมของวงจรกรองความถี่แบบปรับตัวได้

ในการพัฒนาแต่ละอัลกอริทึมจะพบว่าแต่ละอัลกอริทึมมีคุณลักษณะเฉพาะตัว จึงมีความสำคัญมากในการที่จะทำความเข้าใจในความสามารถและขีดจำกัดเพื่อให้การเลือกใช้งานมีประสิทธิภาพ

1. กรรมวิธีการประมาณแบบสตอแคสติก เกรเดียนท์ (Stochastic Gradient Approach) เป็นกรรมวิธีที่วัดค่าความผิดพลาดโดยใช้ค่าเฉลี่ยผลต่างกำลังสองและพยายามลดค่าเฉลี่ยผลต่างกำลังสอง โดยการทำการหาค่าจุดต่ำสุดบนเออร์เรอร์เพอร์ฟอร์มานซ์เซอร์เฟส (Error Performance Surface) ซึ่งค่าต่ำสุดจะให้ค่าสัมประสิทธิ์ที่ทำให้ค่าเฉลี่ยผลต่างกำลังสองมีค่าต่ำสุด การหาอัลกอริทึมเพื่อใช้ในการปรับค่าสัมประสิทธิ์จะทำการแก้สมการของวีนเนอร์และฮอร์พ (Weiner-Hopf Equation) อัลกอริทึมในกลุ่มนี้ที่นิยมใช้คือลีสต์มีนสแควร์ (Least Mean Square) หรือเรียกอีกอย่างว่าแอลเอ็มเอสอัลกอริทึม (LMS Algorithm)

2. กรรมวิธีการประมาณแบบลีสต์สแควร์ (Least Square Estimation) กรรมวิธีนี้ใช้การลดคอสท์ฟังก์ชัน (Cost Function) โดยอาจใช้วิธีการของรีเคอซีฟ ลีสต์สแควร์ (Recursive Least Square) ซึ่งสามารถแบ่งออกเป็น 3 อัลกอริทึมดังนี้

- รีเคอร์ซีฟ ลีสต์สแควร์ อัลกอริทึมมาตรฐาน (Standard RLS Algorithm)
- รีเคอร์ซีฟ ลีสต์สแควร์ อัลกอริทึมโดยการถอดราก (Square Root RLS Algorithm)
- รีเคอร์ซีฟ ลีสต์สแควร์ อัลกอริทึมแบบเร็ว (Fast RLS Algorithm)

ตัวกรองวีนเนอร์จะใช้ค่าสัมประสิทธิ์ $h(n)$ ที่ดีที่สุด ทุกๆค่าเวลา n แต่ปัญหาของตัวกรองวีนเนอร์ก็คือ ค่า $h(n)$ มีการคำนวณที่ยุ่งยากและจำเป็นต้องรู้ค่าพารามิเตอร์ทางสถิติของสัญญาณอินพุทและเอาท์พุทด้วย ซึ่งในทางปฏิบัติมักจะไม่ทราบค่าที่แน่นอน ด้วยเหตุผลนี้ทำให้ตัวกรองวีนเนอร์แทบจะไม่มีการนำมาใช้งานในทางปฏิบัติ

อย่างไรก็ตามตัวกรองวีนเนอร์นี้มีประโยชน์มากในแง่ทฤษฎี เพราะมันคือเป้าหมายที่ตัวกรองปรับตัวได้แบบอื่นๆต้องการไปถึง อัลกอริทึมปรับตัวได้ที่นิยมได้แก่ LMS, RLS และ Kalman ล้วนแล้วแต่มีผลตอบสนองที่พิสูจน์ได้ว่าลู่เข้าสู่ผลตอบสนองของตัวกรองวีนเนอร์ทั้งสิ้น กล่าวคือ อัลกอริทึมเหล่านี้ไม่ได้ให้ผลตอบสนองที่เหมือนกับผลตอบสนองของตัวกรองวีนเนอร์ทุกๆขั้นเวลา แต่ถ้าถูกใช้ในสภาวะที่ลักษณะทางสถิติของสัญญาณอินพุทเปลี่ยนแปลงไม่เร็วนัก เมื่อเวลาผ่านไปสักระยะหนึ่ง มันจะสามารถให้ผลตอบสนองที่ลู่เข้าสู่ผลตอบสนองของตัวกรองวีนเนอร์ได้ ลักษณะสำคัญของการคำนวณสัมประสิทธิ์ด้วยอัลกอริทึมเหล่านี้ที่ปรับปรุงจากตัวกรองวีเนอร์ คือ

1) ไม่จำเป็นต้องรู้ค่าทางสถิติของสัญญาณอินพุท
2) มีการคำนวณที่สามารถปรับไปใช้การคำนวณแบบเวลาจริงได้

2.7.3 การวัดประสิทธิภาพในระบบปรับตัว

ในระบบปรับตัวทุกระบบมีความจำเป็นที่จะต้องทำการวัดประสิทธิภาพ โดยทำการเปรียบเทียบระหว่างอัลกอริทึมต่างๆในตัวกรองปรับตัวก็เช่นกัน นอกจากจะต้องทำการเปรียบเทียบสัมประสิทธิ์ของ

ตัวกรองแล้วยังต้องทำการเปรียบเทียบประสิทธิภาพของอัลกอริทึมอีก เพื่อช่วยในการตัดสินใจเลือกใช้อัลกอริทึมที่ให้การตอบสนองที่ดี การวัดประสิทธิภาพของอัลกอริทึมมีหลายวิธีดังจะกล่าวต่อไปนี้

1. ความเร็วในการลู่เข้า (Convergence Rate)

อัตราการลู่เข้าของระบบปรับตัวเป็นคุณสมบัติที่สำคัญมาก ซึ่งจะต้องทำการวัดเพื่อให้ได้ตามความต้องการของงานที่จะนำไปใช้ ซึ่งโดยทั่วไปแล้วความเร็วในการลู่เข้านั้นสามารถเปรียบเทียบประสิทธิภาพของอัลกอริทึมได้ อย่างไรก็ตามการเลือกใช้อัลกอริทึมไม่จำเป็นต้องใช้อัลกอริทึมที่มีความเร็วสูงสุดเนื่องจากการเพิ่มความเร็ว ราคาในการสร้างและความซับซ้อนของวงจรก็จะสูงขึ้นเช่นการเลือกใช้จึงขึ้นอยู่กับความจำเป็นในการใช้งานมากกว่า

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 ตลอดจนสัญญาณรบกวนทั่วไป

3. ความถูกต้องในการประมาณค่าตัวแปร (Parameter Estimation Accuracy)

ความถูกต้องในการประมาณค่าตัวแปรเป็นปัจจัยที่มีความสำคัญมาก เมื่อมีการนำระบบปรับตัวไปใช้ในการหาคุณลักษณะของระบบที่ไม่ทราบ (System Identifying) ซึ่งความถูกต้องในการประมาณค่าตัวแปรมีมาก จะทำให้ได้แบบจำลองของระบบที่ต้องการใกล้เคียงความจริงมากขึ้น

4. ความซับซ้อนในการประมวลผล (Computation Complexity)

ความซับซ้อนในการประมวลผลมีความสำคัญมากในการนำระบบปรับตัวไปใช้งานจริง โดยทั่วไปต้องการความซับซ้อนในการประมวลผลที่ต่ำ ทำให้การสร้างมีราคาถูก นอกจากนั้นหากนำไปโปรแกรมลงในตัวประมวลผลสัญญาณสำเร็จรูปแล้วนั้นก็ต้องคำนึงถึงความสามารถในการทำงานของตัวประมวลผลเปรียบเทียบกับความซับซ้อนในการประมวลผลของอัลกอริทึมด้วย

5. เสถียรภาพ (Stability)

ความมีเสถียรภาพของระบบมีความสำคัญมากในการใช้งานตัวกรองปรับตัวได้ โดยเฉพาะตัวกรองแบบไอไออาร์ เนื่องจากอาจเกิดการเลื่อนของโพลออกนอกวงกลมหนึ่งหน่วย ทำให้เกิดการออสซิเลตและทำให้ค่าลู่ออก (Diverge) จนไม่สามารถทำงานได้ การออกแบบจึงต้องคำนึงถึงเสถียรภาพของตัวกรองด้วย

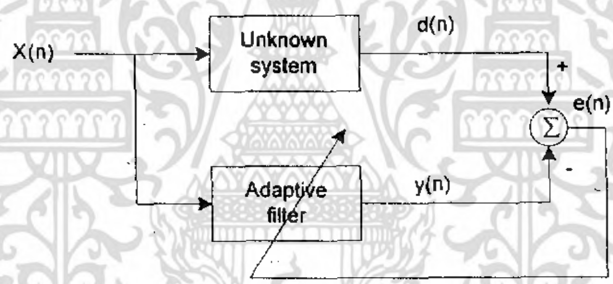
6. ความคงทนของระบบ (Robustness)

เนื่องจากค่าอินพุทของตัวกรองปรับตัวได้มักมีค่าต่างๆที่ไม่แน่นอน จึงมีความจำเป็นที่ตัวกรองจะต้องมีความทนต่อความแปรปรวนของสัญญาณ โดยทั่วไปความคงทนของระบบเป็นค่าที่ยากต่อการวัดในการทดลอง โดยทั่วไปจึงไม่ทำการวัดคุณลักษณะนี้

2.7.4 การประยุกต์ใช้งานวงจรกรองปรับตัวได้

ตัวกรองปรับตัวได้มีคุณสมบัติในการปรับตัวได้อย่างอัตโนมัติทำให้ถูกนำมาประยุกต์ใช้อุปกรณ์หลายๆอย่าง ต่อไปจะทำการอธิบายตัวอย่างการนำตัวกรองปรับตัวได้ไปประยุกต์ใช้งาน

1. การประยุกต์ในการจำลองคุณลักษณะของระบบที่ไม่ทราบ (System Identification) สมมติว่ามีระบบที่ไม่เป็นที่รู้จัก และต้องการพิจารณาผลตอบสนองของระบบนั้นต่อสัญญาณที่ป้อนให้กับระบบ โดยสมมติว่าระบบนี้ไม่เปลี่ยนแปลงไปตามเวลาและมีคุณสมบัติเชิงเส้น และต้องการที่จะพัฒนาแบบจำลอง (Model) สำหรับระบบนี้โดยใช้ตัวกรองปรับตัวได้ เพื่อที่จะสร้างสัญญาณเอาท์พุทให้เหมือนกับเอาท์พุทของระบบที่ไม่เป็นที่รู้จัก โดยการป้อนอินพุทที่เหมือนกันให้แก่ระบบทั้งสองและทำการเปรียบเทียบเอาท์พุททั้งสอง เพื่อสร้างสัญญาณผิดพลาด ซึ่งเป็นความแตกต่างระหว่างสัญญาณทั้งสอง อาจกล่าวได้ว่า การทำงานของตัวกรองปรับตัวได้เป็นการพยายามปรับผลตอบสนองให้เหมือนกับผลตอบสนองของระบบที่ไม่เป็นที่รู้จัก เพื่อที่จะทำให้สัญญาณผิดพลาดมีน้อยที่สุดคือ ค่าความผิดพลาดเฉลี่ยกำลังสอง และสำหรับในกรณีที่สัญญาณอินพุทของระบบที่ไม่เป็นที่รู้จักนั้นคงที่และระบบที่ไม่เป็นที่รู้จักไม่เปลี่ยนแปลงไปตามเวลาจะทำให้ค่าความผิดพลาดลดลงสู่ศูนย์ (Convergence) ในขณะนั้นเองกล่าวได้ว่าตัวกรองปรับตัวทำการจำลองคุณลักษณะของระบบที่ไม่ทราบได้แล้ว



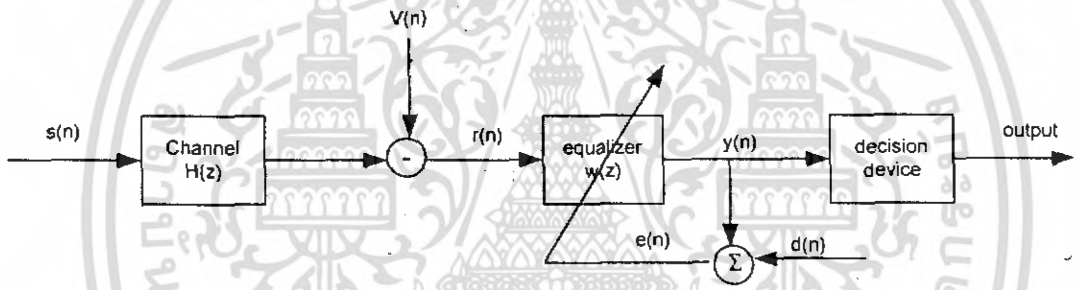
รูปที่ 2.22 แสดงการประยุกต์ใช้ตัวกรองปรับตัวในการจำลองคุณลักษณะของระบบที่ไม่ทราบ

2. การพัฒนาอิควอไลเซอร์แบบปรับตัวได้ (Adaptive Equalization) ระบบการส่งข้อมูลทั่วไปจะพยายามใช้ประโยชน์จากช่องสัญญาณที่มีอยู่ให้มีประสิทธิภาพที่สุด โดยออกแบบระบบการส่งข้อมูลด้วยอัตราสูงที่สุดเท่าที่จะเป็นไปได้ ภายใต้ความน่าเชื่อถือที่กำหนด ซึ่งปกติถูกวัดอยู่ในรูปของอัตราการผิดพลาด หรือความน่าจะเป็นเฉลี่ยของการผิดพลาด หรือความน่าจะเป็นเฉลี่ยของการผิดพลาดของสัญลักษณ์ (Average probability of symbol error) การส่งข้อมูลดิจิตอลผ่านช่องทางการสื่อสารแบบเชิงเส้นถูกจำกัดด้วยองค์ประกอบสองอย่าง

1) การแทรกซ้อนระหว่างสัญลักษณ์ เกิดเป็นการแทรกซ้อนของสัญญาณที่ถูกส่ง ซึ่งเป็นผลมาจากการเบี่ยงเบนของการตอบสนองความถี่ในช่องทาง

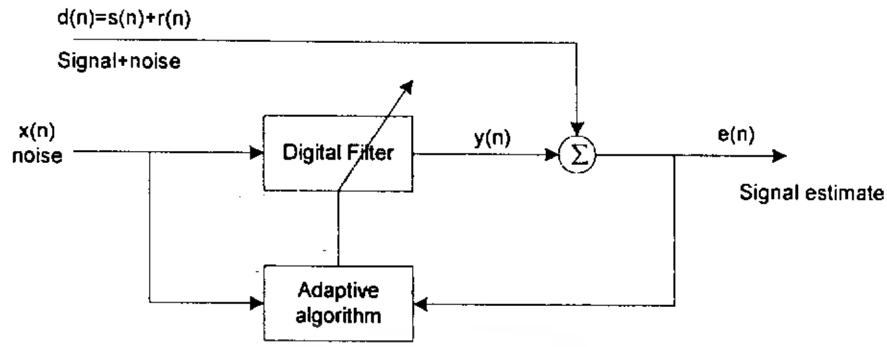
2) การรบกวนจากเทอร์มอลน้อยส์ เกิดจากสัญญาณรบกวนซึ่งเกิดจากอุณหภูมิ ซึ่งโดยทั่วไปมักเกิดในอุปกรณ์ต่างๆของระบบสื่อสารสำหรับช่องทางสื่อสารที่จำกัด เช่น โทรศัพท์ปกติจะพบว่าการแทรกซ้อนระหว่างสัญลักษณ์เป็นองค์ประกอบสำคัญที่ใช้พิจารณาในการกำหนดอัตราส่งของข้อมูล

เพื่อที่จะแก้ปัญหาการแทรกซ้อนระหว่างสัญลักษณ์ต้องมีตัวกรอง โดยถ้าคุณลักษณะของช่องสัญญาณเป็นที่รู้อย่างแน่นอนแล้ว จะสามารถออกแบบตัวกรองทั้งภาคส่งและภาครับได้ ซึ่งจะลดผลของการแทรกซ้อนระหว่างสัญลักษณ์ลงได้และยังช่วยลดสัญญาณรบกวนที่เพิ่มขึ้นมา แต่อย่างไรก็ตามในทางปฏิบัติเราพบว่าคุณลักษณะของช่องสัญญาณสื่อสารนั้นเป็นแบบสุ่ม ฉะนั้นในการใช้ตัวกรองภาคส่งและภาครับที่ออกแบบ โดยคุณลักษณะของช่องสัญญาณเฉลี่ยจะไม่เพียงพอที่จะลดการแทรกซ้อนระหว่างสัญลักษณ์ จึงจำเป็นที่จะต้องมีอิควอไลเซอร์แบบปรับตัวได้ ซึ่งจะลดการแทรกซ้อนระหว่างสัญลักษณ์ จึงจำเป็นที่จะต้องมีอิควอไลเซอร์แบบปรับตัวได้ ซึ่งจะลดการแทรกซ้อนระหว่างสัญลักษณ์ได้ดีขึ้น โดยพื้นฐานสำหรับการอิควอไลซ์ของระบบการส่งข้อมูลที่มีอยู่ คือการเพิ่มขนาดสัญญาณที่บางย่าน (Pre-Equalization) ที่ภาคส่งและการลดขนาดสัญญาณที่บางย่าน (Post-Equalization) ที่ภาครับ



รูปที่ 2.23 แสดงการใช้อิควอไลเซอร์แบบปรับตัวได้ในระบบส่งสัญญาณ

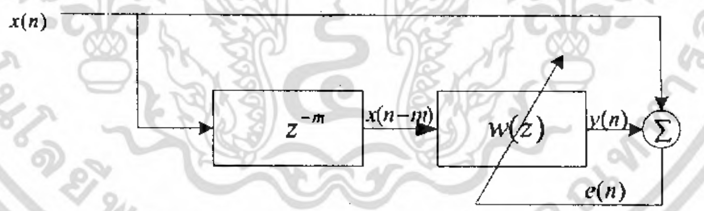
3. การกำจัดสัญญาณรบกวน (Noise Cancellation) การนำตัวกรองปรับตัวได้มาประยุกต์ใช้ในการกำจัดสัญญาณรบกวนนั้น โดยใช้สัญญาณรบกวนที่มีความสัมพันธ์กับสัญญาณรบกวนที่ปนมาในสัญญาณที่ต้องการ ในการกำจัดสัญญาณรบกวนโดยการป้อนสัญญาณและโครงสร้างของระบบกำจัดสัญญาณรบกวนเป็นไปตามรูปที่ 2.24



รูปที่ 2.24 แสดงโครงสร้างระบบกำจัดสัญญาณรบกวนโดยใช้ตัวกรองปรับตัวได้

จากรูปป้อนสัญญาณรบกวน $x(n)$ ซึ่งมีความสัมพันธ์กับสัญญาณรบกวน r(n) ให้แก่ระบบจะได้เอาท์พุทของตัวกรองปรับตัว y(n) แล้วจึงนำไปหักล้างกับสัญญาณรบกวน r(n) ซึ่งปะปนอยู่กับสัญญาณที่ต้องการ s(n) ทำให้ได้สัญญาณที่ต้องการ e(n) ซึ่งเป็นสัญญาณที่มีสัญญาณรบกวนน้อยลงและสัญญาณนี้จะถูกป้อนให้กับตัวกรองเพื่อใช้ในการเปรียบเทียบในการปรับค่าสัมประสิทธิ์อีกทีหนึ่ง

1. การพัฒนาอุปกรณ์เพิ่มประสิทธิภาพสาย (Adaptive Line Enhancer) อุปกรณ์เพิ่มประสิทธิภาพสาย เป็นอุปกรณ์ที่สามารถใช้แยกสัญญาณแบนด์แคบซึ่งฝังอยู่ในสัญญาณแบนด์กว้าง ซึ่งขึ้นกับการใช้งานสัญญาณที่ต้องการแยกออกอาจเป็นสัญญาณที่สนใจหรือสัญญาณรบกวนที่ไม่ต้องการ ตัวอย่างของการใช้งานเช่น การลดสัญญาณรบกวนความถี่ 50-60 Hz ออกจากสัญญาณของอุปกรณ์ทางการแพทย์



รูปที่ 2.25 แสดงอุปกรณ์เพิ่มประสิทธิภาพสาย

2.7.5 วงจรกรองความถี่ปรับตัวได้แบบเอฟไออาร์ (Adaptive FIR Filter)

ในการศึกษาวงจรกรองแบบปรับตัวได้นั้น โดยทั่วไปมักจะเริ่มจากการศึกษาวงจรกรองปรับตัวได้แบบเอฟไออาร์เสียก่อน เนื่องจากตัวกรองปรับตัวแบบเอฟไออาร์มีคุณสมบัติบางประการที่ดีกว่า กล่าวคือ เสถียรภาพของระบบที่ดีกว่าเนื่องจากสัมประสิทธิ์ของตัวกรองที่มีค่าจำกัด อัลกอริทึมที่ใช้ในการปรับตัวก็ง่ายกว่า นอกจากนั้นแล้วประสิทธิภาพของตัวกรองก็สามารถมองได้ในลักษณะของการลู่เข้าและเสถียรภาพต่อไปจะแสดงการหาอัลกอริทึมสำหรับวงจรกรองปรับตัวได้แบบเอฟไออาร์ โดยใช้อัลกอริทึม

ค่าเฉลี่ยกำลังสองน้อยสุดหรืออัลกอริทึมแบบแอลเอ็มเอส ซึ่งเป็นอัลกอริทึมที่ง่ายที่สุดเมื่อเทียบกับตัวอื่นๆและมีการใช้งานอย่างกว้างขวาง

ตัวกรองแบบเอฟไออาร์ ซึ่งมีสัญญาณอินพุทเป็น $x(n)$ มีสมการทั่วไปคือ

$$y(n)=\sum_{k=0}^{N-1}h_k(n)x(n-k) \tag{2.20}$$

ให้สัญญาณที่ต้องการเป็น d(n) แล้วสัญญาณความผิดพลาดสามารถนิยามโดย

$$e(n)=d(n)-y(n) \tag{2.21}$$

สำหรับความต้องการ โดยทั่วไปของตัวกรองปรับตัว การเลือกสัมประสิทธิ์ $h_0(n),h_1(n),\ldots,h_{N-1}(n)$ นั้นจะอยู่บนพื้นฐานของการลดสัญญาณความผิดพลาดให้น้อยที่สุด โดยยึดเอาค่าเฉลี่ยกำลังสองในการเปรียบเทียบ

โดยทั่วไปนั้นแอลเอ็มเอสอัลกอริทึมจะใช้วิธีการปรับสัมประสิทธิ์แบบสตีปเปสต์ เดสเซนท์ (Steepest Desent) ซึ่งก็คือ

$$\bar{h}(n+1)=\bar{h}(n)-\mu\nabla e^2(n) \tag{2.22}$$

เมื่อ $\bar{h}(n)=[h_0(n)\ h_1(n),\ \ldots\ ,h_{N-1}(n)]^T$ เป็นเมตริกซ์สัมประสิทธิ์ μ จะเป็นค่าสัมประสิทธิ์การลู่เข้า (Step size) และ ∇ เป็นค่าเกรเดียนท์เวคเตอร์ซึ่งกำหนดโดย

$$\nabla=\left[\frac{\partial}{\partial h_0}\ \frac{\partial}{\partial h_1}\ \cdots\ \frac{\partial}{\partial h_{N-1}}\right]^T \tag{2.23}$$

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

$$\frac{\partial e^2(n)}{\partial h_k}=2e(n)\frac{\partial e(n)}{\partial h_k} \tag{2.24}$$

ค่า e(n) = d(n)-y(n) ลงในสมการข้างต้นจะได้เป็น

$$\frac{\partial e^2(n)}{\partial h_k}=-2e(n)\frac{\partial y(n)}{\partial h_k} \tag{2.25}$$

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

$$\frac{\partial e^2(n)}{\partial h_k}=-2e(n)x(n-k) \tag{2.26}$$

ดังนั้นจากสมการข้างต้นจะพบว่า

$$\nabla e^2(n)=-2e(n)\bar{x}(n) \tag{2.27}$$

เมื่อ $\bar{x}(n) = [x(n)\ \ x(n-1)\ \ \ldots\ \ x(n-N+1)(n)]^T$ หลังจากทำการแทนค่าแล้วจะได้สมการปรับสัมประสิทธิ์เป็น

$$\bar{h}(n+1) = \bar{h}(n) + 2\mu e(n)\bar{x}(n) \tag{2.28}$$

จากสมการทั้งหมดเราจะได้อัลกอริทึมโดยสรุปเป็น

อินพุท : เวคเตอร์สัมประสิทธิ์ $\bar{h}(n) = [h_0(n)\ \ h_1(n)\ \ \ldots\ \ h_{N-1}(n)]^T$

อินพุทเวคเตอร์ $\bar{x}(n) = [x(n)\ \ x(n-1)\ \ \ldots\ \ x(n-N+1)(n)]^T$

สัญญาณที่ต้องการ d(n)

เอาท์พุท : ฟิลเตอร์เอาท์พุท y(n)

เวคเตอร์สัมประสิทธิ์ที่ปรับแล้ว $\bar{h}(n+1)$

1. การกรองสัญญาณ

$$y(n) = \bar{h}^T(n)\bar{x}(n)$$

2. การประมาณค่าความผิดพลาด

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

3. การปรับสัมประสิทธิ์

$$\bar{h}(n+1) = \bar{h}(n) + 2\mu e(n)\bar{x}(n)$$

2.8 ระบบดิจิตอลเบื้องต้น

2.8.1 ระบบเลขฐานสิบ (Decimal Number Systems)

ในระบบเลขฐานสิบเป็นระบบจำนวนที่เรารู้จักและทำความเข้าใจได้ง่ายที่สุด เพราะว่าเป็นระบบที่เราใช้งานกันอยู่ทุกวันนี้ ระบบเลขฐานสิบจะประกอบด้วยสัญลักษณ์ของตัวเลข จำนวน 10 ตัว ได้แก่ 0, 1, 2, 3, 4, 5, 6, 7, 8 และ 9 โดยทั่วไปสัญลักษณ์ของตัวเลขแต่ละตัวในระบบเลขจะถูกเรียกว่า "เลขโดด" และ "ดิจิต" (Digit) จึงอาจกล่าวได้ว่าระบบเลขฐานสิบมี 10 ดิจิต หลักที่มีค่าต่ำสุดหน้าจุดทศนิยมคือเลขหลักหน่วย มีน้ำหนัก (weight) เท่ากับ 1 (10^0) ดังตารางที่ 2.1

ตารางที่ 2.1 แสดงค่าน้ำหนักของเลขฐานสิบตำแหน่งต่างๆ

น้ำหนัก	10,000	1,000	100	10	1	0.1	0.01	0.001	0.0001
ตำแหน่ง	10^4	10^3	10^2	10^1	10^0	10^{-1}	10^{-2}	10^{-3}	10^{-4}

2.8.2 ระบบเลขฐานแปด (Octal Number Systems)

ในระบบเลขฐานแปดจะใช้หลักการเช่นเดียวกับระบบเลขฐานสิบ แต่ต่างกันตรงที่ฐานเลขเป็น 8 เนื่องจากประกอบด้วยสัญลักษณ์ของตัวเลข จำนวน 8 ตัว ได้แก่ 0, 1, 2, 3, 4, 5, 6 และ 7 จึงอาจกล่าวได้ว่าระบบเลขฐานแปดมี 8 ดิจิต มีค่าน้ำหนักดังตารางที่ 2.2

ตารางที่ 2.2 แสดงค่าน้ำหนักของเลขฐานแปดตำแหน่งต่างๆ

น้ำหนัก	512	64	8	1	0.125	0.015625	1.953125^{-3}
ตำแหน่ง	8^3	8^2	8^1	8^0 .	8^{-1}	8^{-2}	8^{-3}

การแปลงเลขฐานสิบเป็นเลขฐานแปดสามารถทำได้โดยการนำเลขฐานสิบมาหารด้วยแปด และนำผลหารหลังจุดทศนิยมไปคูณด้วยแปด และนำผลหารหน้าจุดทศนิยมไปหารด้วยแปด จนกระทั่งผลลัพธ์จากการหารหน้าจุดทศนิยมเป็นศูนย์ คำตอบคือ ผลคูณของเลขหลังจุดทศนิยมทุกตัวโดยผลการคูณตัวแรกจะเป็นบิทต่ำสุดของเลขฐานแปด

2.8.3 ระบบเลขฐานสิบหก (Hexadecimal Number Systems)

ในระบบเลขฐานสิบหกจะคล้ายกับระบบเลขฐานเลขแปด ซึ่งเป็นการประยุกต์ใช้ในการโปรแกรมคอมพิวเตอร์ หรืออุปกรณ์ควบคุมโปรแกรมได้ (Programmable Control) รวมทั้งการโปรแกรมคอมพิวเตอร์ชนิดแผ่นพิมพ์เดียว ระบบเลขฐานสิบหกจะประกอบไปด้วยตัวเลขและตัวอักษรรวมกันจำนวน 16 ตัว ได้แก่ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, และ F ซึ่งมีจำนวน 16 ดิจิต นั่นคือตัว A แทนเลข 10 ตัว B แทนเลข 11 และตัว F แทนเลข 15 มีค่าน้ำหนักดังตารางที่ 2.3

ตารางที่ 2.3 แสดงค่าน้ำหนักของเลขฐานสิบหกตำแหน่งต่างๆ

น้ำหนัก	4096	256	16	1	0.0625	3.90625^{-3}	2.441406^{-4}
ตำแหน่ง	16^3	16^2	16^1	16^0 .	16^{-1}	16^{-2}	16^{-3}

ดังนั้นการแปลงเลขฐานสิบหกเป็นเลขฐานสิบจึงต้องใช้ตารางค่าน้ำหนักเป็นตัวกำหนด

2.8.4 ระบบเลขฐานสอง (Binary Number Systems)

ในระบบเลขฐานสองนั้นจะประกอบด้วยตัวเลขเพียง 2 ตัว คือ “0” และ “1” ซึ่งในวิชาดิจิตอลเบื้องต้น หรือส่วนที่เกี่ยวกับวงจรดิจิตอล จะต้องใช้สัญญาณดิจิตอลที่รู้จักกันในชื่อของสัญญาณ “0” กับ “1” หรือสัญญาณที่มีแรงดันสองระดับ คือระดับลอจิกสูง (ลอจิก 1 หรือ H) และระดับลอจิกต่ำ (ลอจิก 0 หรือ L) เลขฐานสองแต่ละหลักเรียกว่า 1 บิท (bit: binary digit) มีค่าน้ำหนักดังตารางที่ 2.4

ตารางที่ 2.4 แสดงค่าน้ำหนักของเลขฐานสองตำแหน่งต่างๆ

น้ำหนัก	16	8	4	2	1	0.5	0.25	0.125	0.0625
ตำแหน่ง	2^4	2^3	2^2	2^1	2^0 .	2^{-1}	2^{-2}	2^{-3}	2^{-4}

ถ้าหากนำตัวเลขฐานสองมากกว่า 1 บิทมาพิจารณา เช่น เลขฐานสอง 2 บิท ซึ่งจะมีผลทำให้เกิดจำนวนของการเปลี่ยนแปลงเป็น 4 สถานะ ส่วนในกรณีถ้าเป็นเลขฐานสอง 3 บิท ซึ่งจะเกิดการเปลี่ยนแปลง 8 สถานะ จึงสรุปเป็นสมการคณิตศาสตร์และความสัมพันธ์ของจำนวนหลัก ซึ่งจะใช้แสดงสถานะของการเปลี่ยนแปลงในระบบเลขฐานสองได้ดังนี้

$$\text{จำนวนสถานการณ์เปลี่ยนแปลง} = 2^n \tag{2.29}$$

โดยที่ 2 คือเลขฐานสอง n คือจำนวนบิท

โดยทั่วไปแล้วเลขฐานสองจำนวน n บิทคือ $a_{n-1}a_{n-2}...a_1a_0$ เมื่อถูกนำมาเขียนแทนเลขจำนวนเต็มบวก A จะสามารถคำนวณหาค่าได้ดังนี้

$$A = \sum_{i=0}^{n-1} 2^i a_i \tag{2.30}$$

จากสมการที่ (2.30) เป็นรูปแบบที่ไม่มีเครื่องหมาย และจะเรียก a_{n-1} ว่าเป็น MSB (most-significant bit) และ a_0 เป็น LSB (least-significant bit)

2.8.5 การแปลงเลขฐานสองแบบทูคอมพลีเมนท์ (Two's complement)

ในระบบเลขฐานสองนั้นจำนวนจะมีสองชนิดซึ่งคือตัวเลขที่เป็นเลขบวก และเลขลบซึ่งจะถูกเขียนด้วยระบบของใช้สัญลักษณ์และไม่ใช้สัญลักษณ์ (Signed and unsigned) โดยที่แบบที่ใช้สัญลักษณ์นั้นจะใช้ได้ทั้งเลขบวกและเลขลบ และถ้าแบบไม่ใช้สัญลักษณ์จะใช้กับค่าบวกเท่านั้น โดยการเลือกใช้แต่ละชนิดนั้นขึ้นอยู่กับว่าระบบของเราที่ใช้จะเหมาะกับแบบใดมากกว่ากัน

ยกตัวอย่างเช่น ใช้แบบไม่ใช้สัญลักษณ์ 8-บิท หมายถึง ทุกๆ 8 บิทถูกใช้และช่วงของจำนวนสามารถถูกใช้จาก 0 (00000000) ถึง 255 (11111111)

ถ้าใช้แบบสัญลักษณ์ 8-บิท ช่วงของจำนวนตัวเลขจะสามารถเขียนได้เป็น -127 (11111111) ถึง +127 (01111111) โดยที่สัญลักษณ์จะเกิดขึ้นในบิทมากสุด (MSB) ของจำนวนบวกจะเป็น 0 และจำนวนลบจะเป็น 1

ข้อได้เปรียบของแบบที่ใช้สัญลักษณ์ คือ ถ้าการปฏิบัติการโดยการลบถูกนำมาใช้ในระบบสามารถทำได้ง่าย โดยการสละบิทมากที่สุดในระบบ ซึ่งถ้าใช้ในแบบไม่ใช้สัญลักษณ์ในระบบทุกๆ บิทจะถูกใช้ ซึ่งจะยากต่อการทำการลบ ซึ่งการทำทูคอทพลีเมนท์จะเป็นการประยุกต์ค่า ซึ่งจะทำให้รูปแบบของการลบสามารถกระทำได้ในรูปแบบของการบวก

ในระบบฮาร์ดแวร์แบบเลขฐานสองถูกนำมาใช้เยอะดังนั้นองค์ประกอบต่างๆ ใช้การบวกมากกว่าการลบ ถ้าการลบถูกนำมาใช้ ราคาของระบบนั้นๆ จะสูงขึ้นเมื่อเปรียบเทียบกับองค์ประกอบที่ใช้การบวก

2.8.6 รูปแบบตัวแทนของจำนวนเลขที่มีเครื่องหมายกำกับ

มีวิธีการมากมายที่สามารถนำมาใช้ในการเขียนเลขฐานสองแทนเลขจำนวนเต็มบวก และเลขจำนวนเต็มลบ ซึ่งวิธีการทั้งหมดจะเกี่ยวข้องกับข้อมูลที่มีค่าสูงสุดของเลขฐานสองจำนวนนั้นๆ(MSB) และเรียกบิทนั้นว่าบิทเครื่องหมาย (sign bit) ถ้าบิทเครื่องหมายเป็นเลขจำนวนเต็มบวก และถ้ามีค่าเป็น "1" ก็แสดงว่าเป็นเลขจำนวนลบ

รูปแบบที่ง่ายที่สุดในการนำบิทเครื่องหมายมาใช้เรียกว่า รูปแบบเครื่องหมายจำนวน (Sign magnitude) ในระบบนี้ตัวเลขทางขวาจำนวน n-1 บิทจะเป็นตัวเลขที่บอกขนาดเลขจำนวนเต็มนั้นในกรณีทั่วไปเลขจำนวน A จะสามารถคำนวณหาค่าได้ดังนี้

$$A = \begin{cases} \sum_{i=0}^{n-2} 2^i a_i & if \quad a_{n-1} = 0 \\ -\sum_{i=0}^{n-2} 2^i a_i & if \quad a_{n-1} = 1 \end{cases} \tag{2.31}$$

2.9 อัลกอริทึมแบบบ็อก-วูเลย์ (Baugh-Wooley Algorithm)

การคูณเลขฐานที่มีจำนวน m บิท ($Y = y_{m-1}, \ldots, y_0$) และ จำนวน n บิท ($X = x_{n-1}, \ldots, x_0$) จะได้ผลลัพธ์เป็นจำนวน n+m บิท ดังรูปที่ 2.26

$$\begin{array}{cccccccccccc}
 & & & y_{m-1} & \ldots & & y_4 & y_3 & y_2 & y_1 & y_0 \\
 & & & & & x_{n-1} & \ldots & & x_2 & x_1 & x_0 \\
\hline
 & & & x_0 y_{m-1} & \ldots & & x_0 y_4 & x_0 y_3 & x_0 y_2 & x_0 y_1 & x_0 y_0 \\
 & & x_1 y_{m-2} & x_1 y_{m-2} & \ldots & x_1 y_4 & x_1 y_3 & x_1 y_2 & x_1 y_1 & x_1 y_0 & \\
 & x_2 y_{m-1} & x_2 y_{m-2} & \ldots & x_2 y_4 & x_2 y_3 & x_2 y_2 & x_2 y_1 & x_2 y_0 & & \\
 & \cdot & & & & & & & & & \\
 \cdot & & \cdot & & & & & & & & \\
 x_{n-1} y_{m-1} & \ldots & x_{n-1} y_2 & x_{n-1} y_1 & x_{n-1} y_0 & & & & & & \\
\hline
\end{array}$$

$$P_{m+n-1} \; P_{m+n-2} \; \ldots \; P_{m+1} \quad P_m \quad P_{n+1} \quad P_n \quad P_{n-1} \quad \ldots \quad P_3 \quad P_2 \quad P_1 \quad P_0$$

รูปที่ 2.26 แสดงวิธีการคูณโดยทั่วไปแบบ Two's complement $m \times n$ บิท

ในการคูณ Two's complement ของตัวตั้งและตัวคูณที่มีเครื่องหมาย โดยเมื่อให้ตัว Y_v เป็นตัวตั้งของ Y และ ให้ตัว X_v เป็นตัวคูณของ X สำหรับค่า Y_v, X_v จะได้ว่า

$$Y_v = -y_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} y_i 2^i \tag{2.32}$$

$$X_v = -x_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \tag{2.33}$$

ผลลัพธ์การคูณคือ P_v เป็นผลลัพธ์ คือ

$$\begin{aligned} P_v &= -p_{m+n-1} 2^{m+n-1} + \sum_{i=0}^{m+n-2} p_i 2^i = Y_v X_v \\ &= \left(-y_{m-1} 2^{m-1} + \sum_{i=0}^{m-2} y_i 2^i \right) \left(-x_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i \right) \\ &= \left(x_{n-1} y_{m-1} 2^{m+n-2} + \sum_{i=0}^{n-2} \sum_{j=0}^{m-2} x_i y_j 2^{i+j} \right) - \left(\sum_{i=0}^{m-2} x_{n-1} y_i 2^{n-1} + \sum_{i=0}^{n-2} y_{m-1} x_i 2^{m-1+i} \right) \end{aligned} \tag{2.34}$$

เมื่อสร้าง P โดยการรวมส่วนที่เป็นบิทเครื่องหมายเข้าไปด้วยแล้วทำให้เราจะต้องพิจารณาในส่วนที่เป็นเครื่องหมาย โดยเครื่องหมายอยู่ในบิท $x_{n-1}y_i$ ซึ่ง $i=0,...,m-2$ และ $y_{m-1}x_i$ ซึ่ง $i=0,...,n-2$ จะเป็นลบ เมื่อนำบิทของส่วนนี้มาเขียนใหม่จะสามารถแสดงได้ดังรูปที่ 2.27 โดยบิทที่เป็นส่วนลบทั้งหมดจะถูกวางอยู่ใน 2 แถวสุดท้าย และผลลัพธ์ จะได้มาโดยการรวม $n-2$ แถวแรกของส่วน และการลบ 2 แถวสุดท้าย

$$
\begin{array}{ccccccccccccccc}
 & & & & & & & y_{m-1} & \cdots & y_4 & y_3 & y_2 & y_1 & y_0 \\
 & & & & & & & & x_{n-1} & & \cdots & x_2 & x_1 & x_0 \\
\hline
 & & & & & & x_0y_{m-2} & \cdots & & x_0y_4 & x_0y_3 & x_0y_2 & x_0y_1 & x_0y_0 \\
 & & & & & x_1y_{m-2} & \cdots & x_1y_4 & x_1y_3 & x_1y_2 & x_1y_1 & x_1y_0 & & \\
 & & & & x_2y_{m-2} & \cdots & x_2y_4 & x_2y_3 & x_2y_2 & x_2y_1 & x_2y_0 & & & \\
 & x_{n-2}y_{m-2} & 0 & x_{n-2}y_{m-2} & \cdots & x_{n-2}y_2 & x_{n-2}y_1 & x_{n-2}y_0 & & & & & & \\
0 & 0 & x_{n-1}y_{m-2} & x_{n-1}y_{m-3} & \cdots & x_{n-1}y_2 & x_{n-1}y_1 & x_{n-1}y_0 & & & & & & \\
0 & 0 & x_{n-2}y_{m-1} & x_{n-3}y_{m-1} & & & & & & & & & & \\
\hline
P_{n+m-1} & P_{n+m-2} & P_{n+m-3} & P_{n+m-4} & \cdots & P_{n+1} & P_n & P_{n-1} & \cdots & P_3 & P_2 & P_1 & P_0
\end{array}
$$

รูปที่ 2.27 แสดงวิธีการคูณโดยคิดเครื่องหมายบวก/ลบในส่วนของบิทผลลัพธ์

ค่าการลบของเลข Two's complement $Z=(z_{k-1},...,z_0)$ จะได้ว่า Z_v

$$-Z_v = 1-\overline{z}_{k-1}2^{k-1}+\sum_{i=0}^{k-2}\overline{z}_i 2^i \tag{2.35}$$

โดยที่ $\overline{z_i}$ คือ คอมพลีเมนต์ของ $\overline{z_i}$

ดังนั้นการลบของ

$$2^{n-1}\left(-0\cdot 2^m+0\cdot 2^{m-1}+\sum_{i=0}^{m-2}x_{n-1}y_i 2^i\right) \tag{2.36}$$

สามารถเขียนได้เป็น

$$2^{n-1}\left(-1\cdot 2^m+1\cdot 2^{m-1}+1+\sum_{i=0}^{m-2}\overline{x_{n-1}y_i}\,2^i\right) \tag{2.37}$$

ในแถวรองสุดท้ายที่แสดง

$$0\ \ 0\ \ x_{n-1}y_{m-2}\ \ x_{n-1}y_{m-3}\ \cdots\ x_{n-1}y_0$$

จะสามารถเขียนได้เป็น

$$1\ 1\ \overline{x_{n-1}y_{m-2}}\ \overline{x_{n-1}y_{m-3}}\ \ldots\ \overline{x_{n-1}y_0}$$

และในแถวสุดท้ายที่แสดง

$$0\ 0\ x_{n-2}y_{m-1}\ \ x_{n-3}y_{m-1}\ldots\ x_0y_{m-1}$$

จะเขียนได้เป็น

$$1\ 1\ \overline{x_{n-2}y_{m-1}}\ \ \overline{x_{n-3}y_{m-1}}\ldots\overline{x_0y_{m-1}}$$

และ '1' จะถูกเพิ่มในหลักที่ p_{n-1} และ p_{m-1}

การลบของสมการที่ (2.36) จะได้สมการที่ (2.37) ซึ่งสังเกตได้ว่าจะได้ค่าเป็น

$$\begin{cases} 0 & for\ \ x_{n-1}=0 \\ 2^n\left(-2^m+2^{m-1}+1+\sum_{i=0}^{m-2}\overline{y}2^i\right) & for\ \ x_{n-1}=1 \end{cases} \tag{2.38}$$

จากสมการที่ (2.38) สามารถเขียนสมการที่ (2.37) ใหม่ได้ว่า

$$2^{n-1}\left(-2^m+2^{m-1}+\overline{x}_{n-1}2^{m-1}+x_{n-1}+\sum_{i=0}^{m-2}x_{n-1}\overline{y}_i2^i\right) \tag{2.39}$$

ดังนั้นสมการที่ (2.37) จึงเทียบเท่ากับสมการที่ (2.39) และจะสามารถเขียนในแถวสุดท้ายใหม่และมีการบวกค่าคงที่ จะได้เป็นดังรูปที่ 2.28 ต่อจากบิท partial แถวสุดท้ายในรูปที่ 2.27

$$\begin{array}{l}
\qquad\qquad\qquad\qquad y_{m-1}\ \ldots\ y_4\ \ y_3\ \ y_2\ \ y_1\ \ y_0 \\
\qquad\qquad\qquad\qquad\qquad x_{n-1}\ \ldots\ x_2\ \ x_1\ \ x_0 \\
\hline
\qquad\qquad x_0y_{m-2}\ \ldots\qquad x_0y_2\ \ x_0y_1\ \ x_0y_0 \\
\qquad x_1y_{m-2}\ \ldots\qquad x_1y_2\ \ x_1y_1\ \ x_1y_0 \\
x_2y_{m-2}\ \ldots\qquad x_2y_1\ \ x_2y_0 \\
x_{n-1}y_{m-1}\ \ 0\ \ x_{n-2}y_{m-2}\ \ \ldots\ \ x_{n-2}y_2\ \ x_{n-2}y_1\ \ x_{n-2}y_0 \\
\overline{x}_{n-1}\ \ \overline{x_{n-1}y_{m-2}}\ \ \overline{x_{n-1}y_{m-3}}\ \ \ldots\ \ \overline{x_{n-1}y_2}\ \ \overline{x_{n-1}y_1}\ \ \overline{x_{n-1}y_0} \\
\overline{y}_{m-1}\ \ \overline{x_{n-2}y_{m-1}}\ \ \overline{x_{n-3}y_{m-1}} \\
1\qquad\qquad y_{m-1}\qquad\qquad x_{n-1} \\
\hline
P_{n+m-1}\ P_{n+m-2}\ \ P_{n+m-3}\ \ P_{n+m-4}\ \ \ldots\ \ P_{m+1}\ \ldots\ P_{n+1}\ \ P_n\ \ P_{n-1}\ \ \ldots\ \ P_3\ \ P_2\ \ P_1\ \ P_0
\end{array}$$

รูปที่ 2.28 แสดงวิธีการคูณที่มีผลลัพธ์ออกมาเป็นบวกทั้งหมด

ลักษณะหลักทั่วไปของบิทผลลัพธ์ในรูปที่ 2.28 มีข้อดีดังนี้

1. บิทผลลัพธ์จะหาได้จาก AND ของบิทตัวคูณและตัวตั้ง

2. บิทผลลัพธ์ทุกตัวจะมีค่าสัมประสิทธิ์เป็นบวก

ผลลัพธ์จะหาได้จากฟังก์ชั่น AND และฟังก์ชั่น ADDไม่จำเป็นต้องมีการลบ และ NAND ฟังก์ชั่นสำหรับ $\overline{x_i y_i}$ อย่างตัวการทำงานของอัลกอริทึมแบบบ็อก-วูเลย์ เมื่อตัวตั้ง (a) และตัวคูณ (b) มีจำนวนบิทเท่ากัน จะสามารถเขียนได้ดังนี้

สำหรับ N คือ จำนวนบิทของเลขฐานสอง ที่เป็น two's complement แทนด้วย A ซึ่งสามารถเขียนได้ดังนี้

$$A = -2^{N-1} a_{N-1} + \sum_{i=0}^{N-2} 2^i a_i \tag{2.40}$$

เมื่อนำ Two's complement 2 ชุดคูณกัน จะได้ว่า

$$P = \left(-2^{N-1} a_{N-1} + \sum_{i=0}^{N-2} 2^i a_i\right)\left(-2^{N-1} b_{N-1} + \sum_{i=0}^{N-2} 2^i b_i\right)$$

$$= \sum_{i=0}^{N-2}\sum_{j=0}^{n-2} 2^{i+j} a_i b_j + 2^{2N-2} a_{N-1} b_{N-1}$$

$$+\sum_{i=0}^{N-2}\sum_{j=0}^{n-2} 2^{i+j} a_i b_j + 2^{2N-2} a_{N-1} b_{N-1} + \left(\sum_{j=0}^{N-2} 2^j \overline{a_{N-1} b_j} + \sum_{j=0}^{N-2} 2^j \overline{b_{N-1} a_j}\right) 2^{N-1} + 2^N - 2^{2N-1} \tag{2.41}$$

ตัวอย่างเช่น เราจะสามารถการทำงานในกรณีที่ทูคอมพลีเมนท์แบบ 4 บิทคูณ 4 บิท (4*4) ได้ดังรูปที่ 2.29

$$\begin{array}{cccccccc}
 & & & & a_3 & a_2 & a_1 & a_0 \\
 & & & & b_3 & b_2 & b_1 & b_0 \\
\hline
 & & & 1 & \overline{a_3 b_0} & a_2 b_0 & a_1 b_0 & a_0 b_0 \\
 & & & \overline{a_3 b_1} & a_2 b_1 & a_1 b_1 & a_0 b_1 & \\
 & & & & 1 & & & \\
 & & \overline{a_3 b_2} & a_2 b_2 & a_1 b_2 & a_0 b_2 & & \\
1 & a_3 b_3 & \overline{a_2 b_3} & \overline{a_1 b_3} & \overline{a_0 b_3} & & & \\
\hline
P_7 & P_6 & P_5 & P_4 & P_3 & P_2 & P_1 & P_0
\end{array}$$

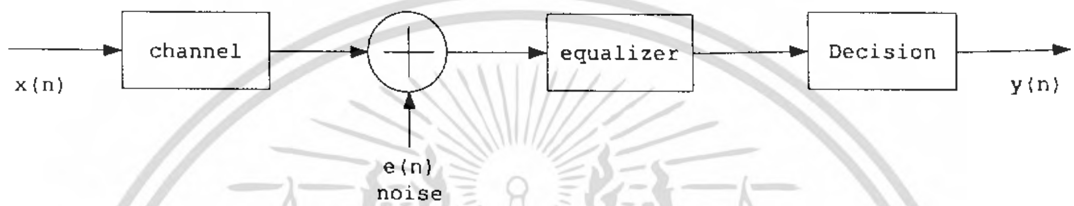
รูปที่ 2.29 อัลกอริทึมการคูณสำหรับเลขทูคอมพลีเมนท์หรือ อัลกอริทึมแบบบ็อก-วูเลย์

จากรูปที่ 2.29 ใน 3 แถวแรกจะใช้ NAND 1 ตัว และใช้ AND 3 ตัว และในแถวที่ 4 จะใช้ AND 1 ตัว และ NAND 3 ตัว โดยผลลัพธ์ใน 3 แถวแรกจะเป็นส่วนของขนาด (Magnitude part: PM) และส่วนที่เหลือ (ในแถวที่ 4) จะเป็นส่วนของเครื่องหมาย (Sign bit: PS)

บทที่ 3
การคำนวณและการสร้าง

3.1 การออกแบบระบบจำลองการสื่อสาร

ในโครงงานนี้จะทำการทดลองระบบการสื่อสาร โดยเริ่มต้นจากการนำสัญญาณอินพุท x(n) เมื่อผ่านเข้าสู่ช่องสัญญาณที่มีสัญญาณรบกวนรวมอยู่ด้วย จะทำให้สัญญาณที่ได้รับนั้นผิดเพี้ยนไปจากเดิม จึงต้องนำมาเข้าสู่วงจรอิควอไลเซอร์ และวงจรดีซิชั่น ดังแสดงในรูปที่ 3.1



รูปที่ 3.1 แสดงระบบจำลองการสื่อสาร

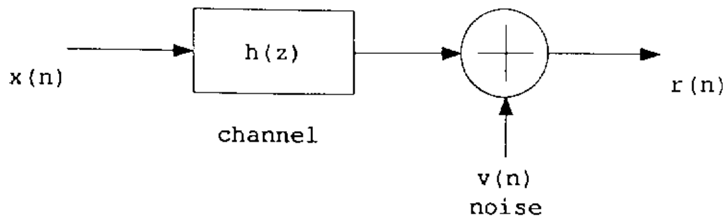
3.1.1 สัญญาณพีเอเอ็ม 4 ระดับ

สัญญาณอินพุทที่ใช้จะเป็นสัญญาณในระบบสื่อสารแบบดิจิตอล จากการมอดูเลตสัญญาณดิจิตอล คือ สัญญาณของ Pulse Amplitude Modulation (PAM) ซึ่งจะสามารถเขียนให้อยู่ในรูปของ constellation ได้ดังรูปที่ 3.2

รูปที่ 3.2 แสดงการเขียนสัญญาณในรูป Constellation ของสัญญาณพีเอเอ็ม 4 ระดับ

3.1.2 ช่องสัญญาณที่มีสัญญาณรบกวน (The Noisy Channel)

ช่องสัญญาณในการสื่อสารในระบบเชิงเส้นจะแสดงด้วยฟังก์ชั่น $h(z)$ และแหล่งกำเนิดสัญญาณรบกวนแบบเกาส์เซียนซึ่งเขียนแทนด้วย $v(n)$ สำหรับสัญญาณที่ได้รับจากการรวมสัญญาณรบกวนและ $x(n)$ จะได้เป็น $r(n)$ ดังแสดงได้ดังรูปที่ 3.3



รูปที่ 3.3 แสดงแบบจำลองช่องสัญญาณที่มีสัญญาณรบกวน

3.1.3 การใช้อีควอไลเซอร์ในช่องสัญญาณ

ตัวอิควอไลเซอร์แบบปรับค่าได้อยู่ในส่วนประกอบที่สำคัญในชนิดของวงจรกรองความถี่แบบปรับตัวได้เชิงเส้น มีแบบจำลองเป็นฟังก์ชันถ่ายโอนผกผันของช่องสัญญาณ

3.2 วิธีการวัดสมรรถนะและความหมายของตัววัดสมรรถนะต่างๆ

3.2.1 อัตราส่วนสัญญาณต่อสัญญาณรบกวน (Signal to noise ratio: SNR)

ค่าอัตราส่วนสัญญาณต่อสัญญาณรบกวนของสัญญาณที่ออกจากเครื่องรับ ซึ่งมีนิยามในหน่วยเดซิเบล (Decibel: dB)

$$SNR_k = 10\log\left(\frac{A^2}{\sigma^2}\right) \tag{3.1}$$

เมื่อ A คือ ขนาดของสัญญาณ

σ^2 คือ ค่าความแปรปรวนของสัญญาณรบกวน ซึ่งคือกำลังของสัญญาณรบกวนนั่นเอง

3.2.2 อัตราความผิดพลาดสัญลักษณ์ (Symbol error rate: SER)

อัตราความผิดพลาดสัญลักษณ์หรือความน่าจะเป็นของความผิดสัญลักษณ์(Symbol Error Probability: SEP) เป็นค่าคงที่แสดงถึงสมรรถนะของระบบ ซึ่งจะพิจารณาที่การดีเทกต์สัญลักษณ์ที่ได้ โดยพิจารณาในสภาวะต่างๆ เช่น เมื่อมีการเปลี่ยนแปลงของอัตราส่วนสัญญาณต่อสัญญาณรบกวน

$$SER = 1-\left(1-P_{\sqrt{M}}\right)^2 \tag{3.2}$$

เมื่อ $P_{\sqrt{M}}$ คือ ความน่าจะเป็นความผิดพลาดสัญลักษณ์

$$P_{\sqrt{M}} = \frac{2\left(\sqrt{M}-1\right)}{\sqrt{M}}Q\left(\sqrt{\frac{3E_{avg}}{(M-1)N_o}}\right) \tag{3.3}$$

โดยที่ M คือ จำนวนตำแหน่งของสัญญาณ

E_{avg}/N_o คือ ค่าเฉลี่ย SNR ต่อสัญลักษณ์

3.2.3 อัตราความผิดพลาดบิท (Bit error rate: BER)

อัตราส่วนความผิดพลาดบิทหรือความน่าจะเป็นของความผิดพลาด(Bit Error Probability: BEP) เป็นค่าที่แสดงถึงสมรรถนะของระบบที่พิจารณาในสภาวะต่างๆ เช่น เมื่อมีการเปลี่ยนแปลงของสัญญาณต่อสัญญาณรบกวน

3.3 การออกแบบการจำลองการทำงานด้วยโปรแกรมแมทแลบ

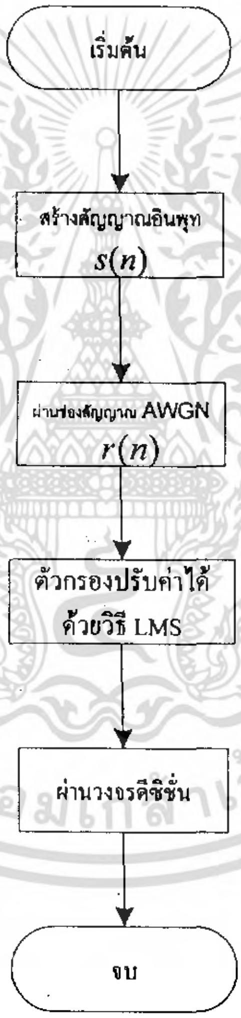
ในการจำลองการทำงานทั้งระบบ มีขั้นตอนการทำงานดังนี้

1. ทำการสร้างสัญญาณอินพุท โดยใช้สัญญาณพีเอเอ็ม 2 ระดับ สัญญาณพีเอเอ็ม 4 ระดับ และสัญญาณคิวพีเอสเค (Quadrature Phase Shift Keying: QPSK)

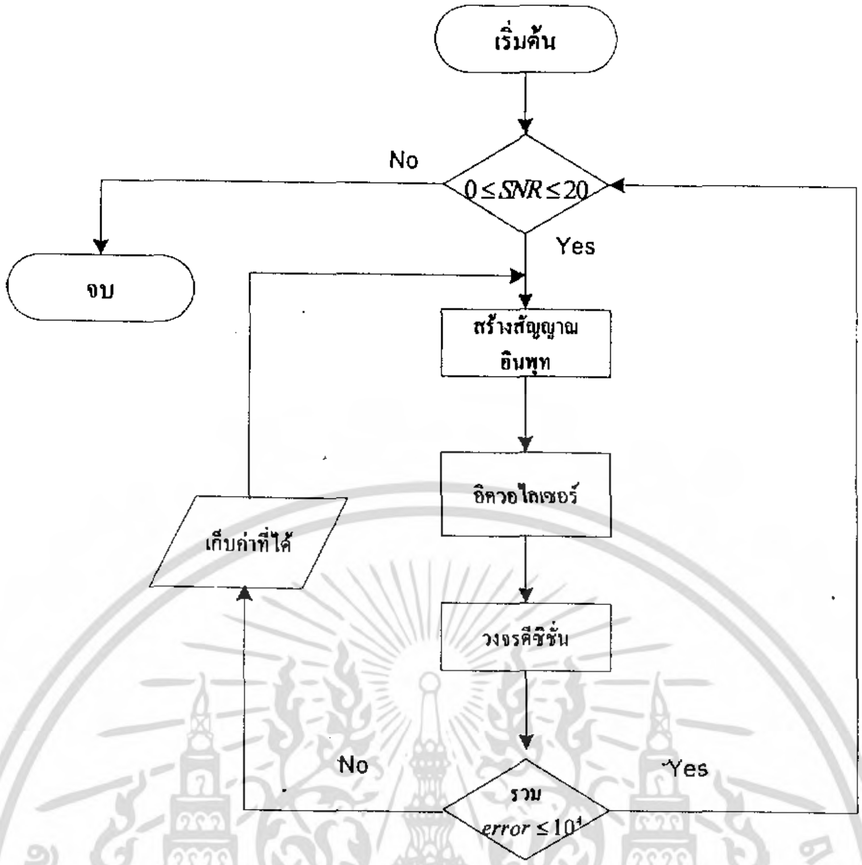
2. นำสัญญาณอินพุทผ่านช่องสัญญาณแบบ AWGN

3. นำสัญญาณผ่านเข้าสู่ระบบอิควอไลเซอร์ คือ เอาสัญญาณที่รับมาจากช่องสัญญาณ AWGN เข้ามาผ่านวงจรกรองที่มีการปรับค่าได้ โดยการหาค่าความผิดพลาดเพื่อนำไปปรับค่าสัมประสิทธิ์โดยใช้อัลกอริทึมแบบแอลเอ็มเอส

4. นำสัญญาณผ่านวงจรดีซิชั่น



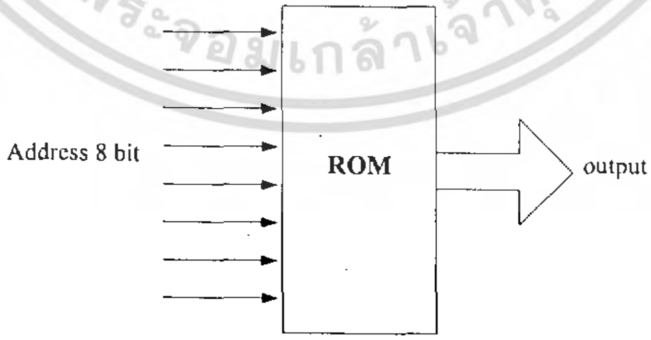
รูปที่ 3.4 แผนผังการจำลองระบบสื่อสารด้วยโปรแกรมแมทแลบ



รูปที่ 3.5 แผนผังการวัดสมรรถนะของระบบ

3.4 การใช้ภาษาวีเอชดีแอลออกแบบสัญญาณพีเอเอ็ม 4 ระดับ

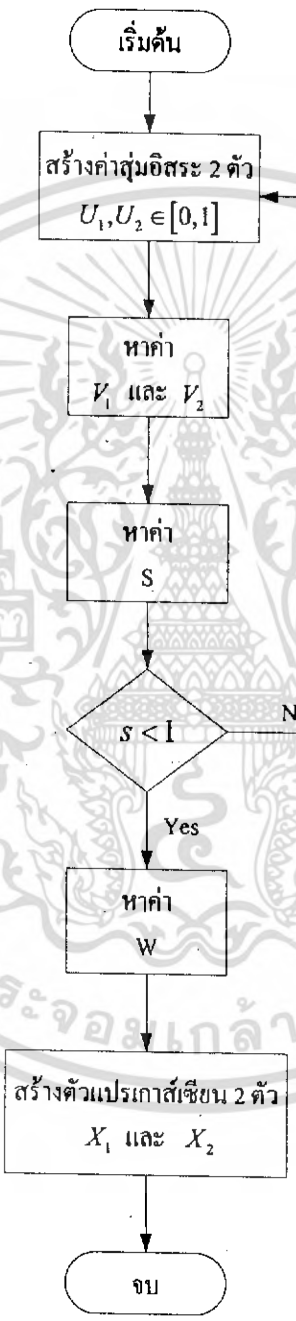
นำสัญญาณอินพุท 8 บิทที่ได้จากวงจรแปลงสัญญาณเชิงอนาล็อกเป็นสัญญาณดิจิตอลเข้าไปชี้ค่าตำแหน่งข้อมูล (address) ไปอ่านค่าผลลัพธ์ของแต่ละแอดเดรสที่อยู่ในวงจรเก็บค่าและเรียกค่าข้อมูล (ROM) ซึ่งมีการเก็บค่าที่แบ่งระดับสัญญาณไว้ 4 ค่า จะได้สัญญาณพีเอเอ็ม 4 ระดับ



รูปที่ 3.6 แสดงโครงสร้างของการออกแบบสัญญาณพีเอเอ็ม 4 ระดับ

3.5 การใช้ภาษาวีเอชดีแอลออกแบบการจำลองช่องสัญญาณแบบ AWGN

ในการจำลองช่องสัญญาณแบบ AWGN นั้นจะสร้างเพื่อทดสอบวงจรอิควอไลเซอร์ โดยวิธีที่เรานำมาใช้นี้ ประกอบด้วยวิธีโพลาร์และวิธีเซนทรัลลิมิต ซึ่งวิธีนี้จะปรับปรุงวิธี Box-Muller โดยการกำจัดฟังก์ชั่นตรีโกณออกไป และใช้ตัวแปร 2 ตัวที่เป็นอิสระ อัลกอริทึมแบบโพลาร์นี้จะทำได้รวดเร็วกว่าอัลกอริทึมแบบ Box-Muller เนื่องจากว่า จะมีฟังก์ชั่นที่น้อยกว่า โดยมีขั้นตอนการทำงานดังนี้



รูปที่ 3.7 แผนผังการจำลองช่องสัญญาณแบบ AWGN ด้วยวิธีโพลาร์

1. สร้างค่าจากการสุ่มอิสระ 2 ตัว โดยกำหนดให้เป็น U_1 และ U_2 ให้มีค่าอยู่ในช่วง [0,1]

2. กำหนดให้ $V_1 = (2 \times U_1) - 1$ และ $V_2 = (2 \times U_2) - 1$

3. กำหนดให้ $S = V_1^2 + V_2^2$

4. ถ้า $S \geq 1$ สั่งให้เริ่มกลับไปทำงานตั้งแต่ในข้อ 2

5. ทำงานเป็นลูป จนกระทั่ง $S < 1$

6. กำหนดให้ $W = \sqrt{-2\ln(s)/s}$

7. สร้างตัวแปรเกาส์เซียน 2 ตัว คือ

$$X_1 = V_1 * W$$
$$X_2 = V_2 * W$$

3.5.1 การสร้างสัญญาณไบนารีแบบสุ่มโดยใช้วงจรนับ LFSR

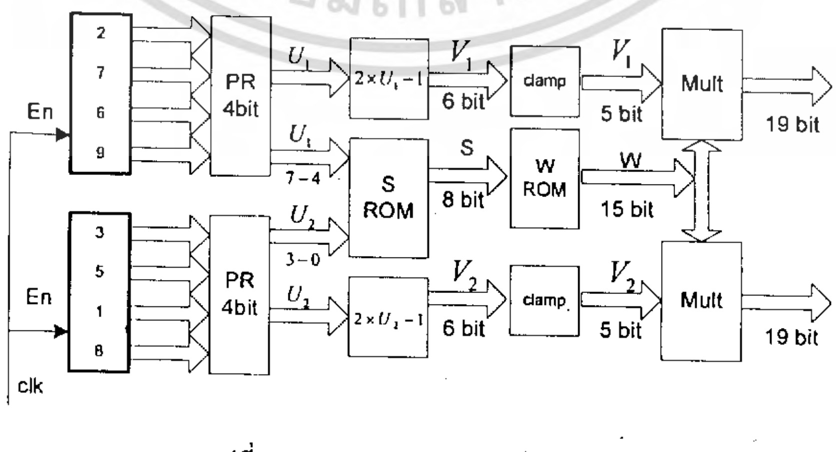
LFSR (Linear feedback shift register) เป็นทางเลือกหนึ่งในการสร้างวงจรนับไบนารี ซึ่งสามารถลดจำนวนวงจรลอจิกให้น้อยลงได้ดีกว่าวิธีอื่นๆ ผลของวงจรนับไบนารีของ LFSR โดยปกติแล้วจะมีลำดับของการนับเป็น $2^m - 1$ สภาวะ และมีลักษณะเป็นไบนารีแบบสุ่ม ทำให้ Register เหล่านี้มีชื่อเรียกอื่นๆ เช่น pseudo-random sequence generator (PRSG)

วัตถุประสงค์หลักๆ ในการใช้ LFSR ก็เพื่อให้มีการนับจำนวนใกล้เคียงกับสภาวะที่เป็นไปได้ $(2^m - 1)$ ให้มากที่สุด และมีการประยุกต์ใช้งานในเรื่องการเข้ารหัสแบบดิจิตอล (digital coding)

คุณสมบัติของ LFSR ในการออกแบบ

- LFSR ที่มีฟลิปฟลอบ (flip - flop) m ตัว สามารถที่จะสร้างจำนวนสภาวะ (state) เป็น $2^m - 1$ โดยที่สภาวะที่เป็นศูนย์จะไม่ยอมให้เกิดขึ้นเนื่องจาก LFSR จะไม่ทำงานได้ (หรือ counter locks up)
- เลือกใช้โพลิโนเมียลที่มั่นใจแล้วว่า $2^m - 1$ สภาวะจะต้องไม่ซ้ำกัน โดยจำเป็นจะต้องรู้ต้นแบบของโพลิโนเมียล
- ในการสร้างวงจรนับต้องคำนึงถึงโพลิโนเมียลที่เหมาะสม เช่น วงจรนับ 35 ต้องใช้โพลิโนเมียลที่มีอันดับเท่ากับ 6 ผลลัพธ์ที่เป็นไปได้คือ $2^6 - 1 = 63$

3.5.2 วิธีโพลาร์



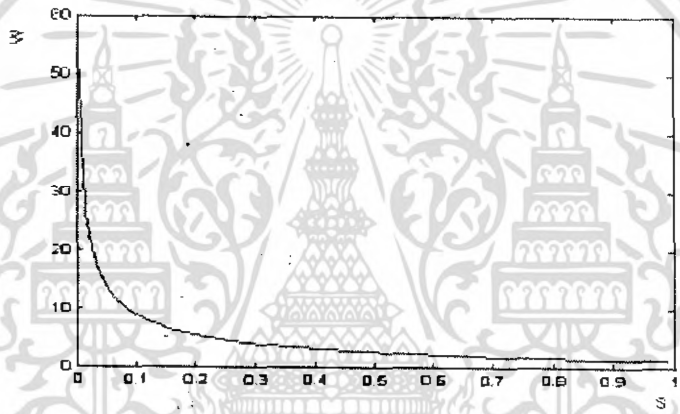
รูปที่ 3.8 แสดงแผนผังการทำงานของวิธีโพลาร์

1. ใช้ 8 LFSRs ที่เป็นอิสระเพื่อสร้างค่าจากการสุ่มอิสระ 2 ตัว คือ U_1 และ U_2 ซึ่งแต่ละตัวประกอบด้วย 4 บิท โดยบิททั้งหมดนี้จะอยู่ในส่วนที่เป็นเศษ ดังนั้นค่าที่เกิดขึ้นของ U_1 และ U_2 จะอยู่ในช่วง [0, 0.9375] ([.0000 , .1111] เมื่อเป็นระบบฐานสอง) ส่วนค่า V_1 และ V_2 จะต้องขึ้นอยู่กับค่า U_1 และ U_2 จากสมการข้างต้น และในการคิดค่า S จะได้ว่า

$$S = (2 * U_1 - 1)^2 + (2 * U_2 - 1)^2 \tag{3.4}$$

โดยค่า S จะถูกเก็บข้อมูลใน ROM ถ้าค่า $S \geq 1$ จะถูกเซตเป็นศูนย์

2. ในส่วนของค่า W ที่ถูกเก็บใน ROM จากรูปที่ 3.9 แสดงกราฟความสัมพันธ์ระหว่าง W กับ S โดยเมื่อค่า S อยู่ในช่วง [0.00390625, 0.99609375] และ ค่า W จะอยู่ในช่วง [54.2835, 0.0886] ในระบบฐานสองเมื่อเราใช้ W แทนด้วย 13 บิท จะได้ว่า 5 บิทสำหรับส่วนจำนวนเต็ม และ 8 บิทสำหรับส่วนเศษ

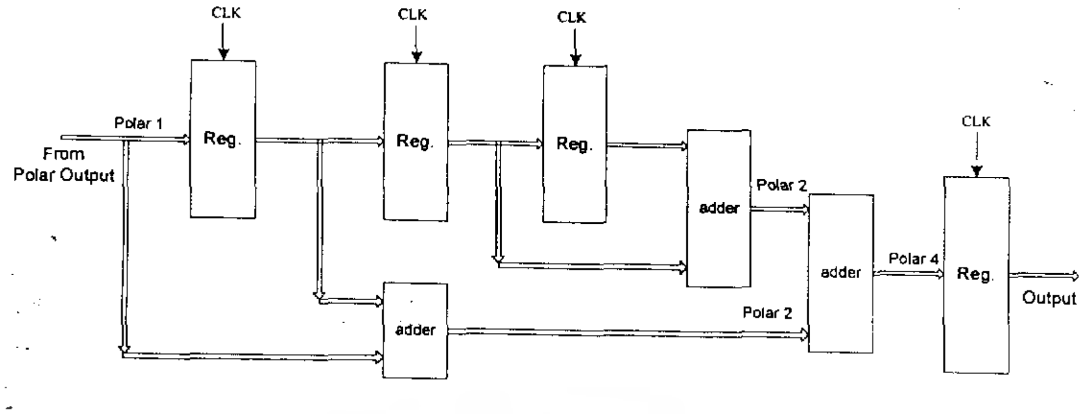


รูปที่ 3.9 กราฟความสัมพันธ์ระหว่าง W กับ S

3. การสร้างผลลัพธ์ จะทำได้จากค่า X_1 และ X_2 ซึ่งขึ้นอยู่กับค่า W, V_1 และ V_2 ซึ่งผลลัพธ์ที่ได้จะมี 19 บิท โดย 1 บิทจะเป็นบิทแสดงเครื่องหมาย 6 บิทสำหรับส่วนจำนวนเต็ม และ 12 บิทสำหรับส่วนเศษ

3.5.3 วิธีเซนทรัลลิมิต

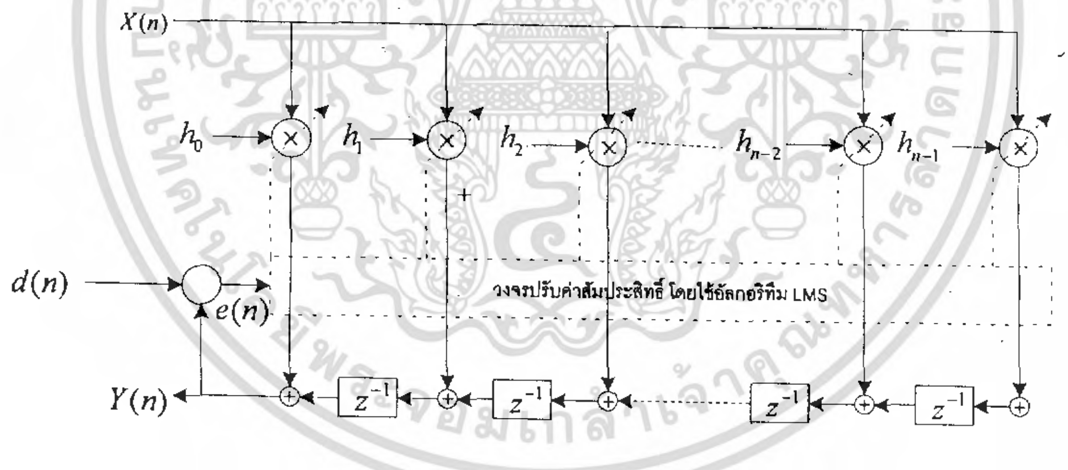
ทฤษฎีเซนทรัลลิมิตกล่าวถึงผลของทฤษฎีความน่าจะเป็นว่า พีดีเอฟของผลลัพธ์ของตัวแปรที่เป็นอิสระหลายๆตัวที่เกิดอย่างสุ่มจะมีแนวโน้มเข้าใกล้พีดีเอฟของเกาส์เซียนมากขึ้นทุกที เมื่อจำนวนตัวแปรเพิ่มมากขึ้น โดยทั่วไปวิธีเซนทรัลลิมิตจะใช้ตัวสะสมค่าซึ่งทำให้ความเร็วของเอาท์พุทช้าลงตามค่าของ N โดยที่ N คือจำนวนของตัวแปรตัวสะสมค่า แสดงได้ดังรูปที่ 3.10



รูปที่ 3.10 แสดงวิธีเซนทรัลลิมิต (N=4)

3.6 การใช้ภาษาวีเอชดีแอลออกแบบอิควอไลเซอร์

วงจรอิควอไลเซอร์มีหน้าที่ในการปรับแก้รูปสัญญาณให้มีคุณลักษณะที่ดีขึ้น เพื่อให้การดึงสัญญาณเดิมกลับคืนมามีความถูกต้องมากขึ้น ซึ่งโดยโครงสร้างในการทำงานของวงจรอิควอไลเซอร์จะประกอบไปด้วยวงจรคูณ วงจรบวกสัญญาณ และวงจรหน่วงเวลา โดยพิจารณาจากรูปที่ 3.11 ในที่นี้การออกแบบวงจรคูณเราจะใช้วิธีการคูณแบบบ็อก-วูเลย์ ซึ่งมีหลักการทำงานตามขั้นต้นที่ได้กล่าวมา และสำหรับวงจรหน่วงเวลานั้นจะอยู่ในตำแหน่งหลังจากการคูณค่าสัมประสิทธิ์แล้ว



รูปที่ 3.11 แสดงโครงสร้างของวงจรอิควอไลเซอร์

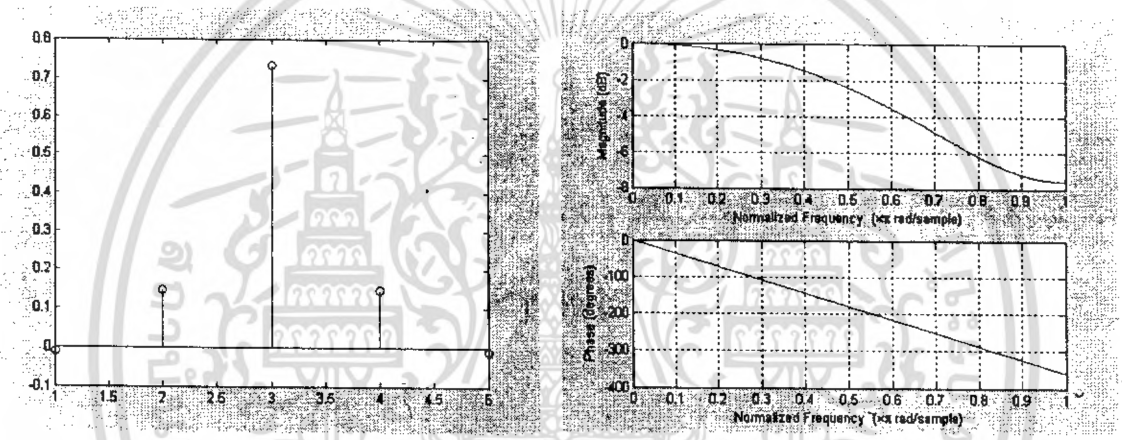
จากรูปที่ 3.11 โครงสร้างของวงจรอิควอไลเซอร์ที่เชื่อมต่อในลักษณะนี้มีชื่อเรียกว่า โครงสร้างแบบทรานสโพสแบบตรง โดยกรรมวิธีในการปรับค่าสัมประสิทธิ์จะขึ้นอยู่กับค่า e(n) ซึ่ง e(n) คือผลต่างที่ได้จากการเปรียบเทียบระหว่างค่า d(n) (ค่าสัญญาณที่ต้องการ) กับค่า Y(n) ฟังก์ชั่นที่ใช้ในการปรับค่าปรับค่าสัมประสิทธิ์นี้เราจะใช้อัลกอริทึมแบบแอลเอ็มเอส

บทที่ 4

การทดลองและผลการทดลอง

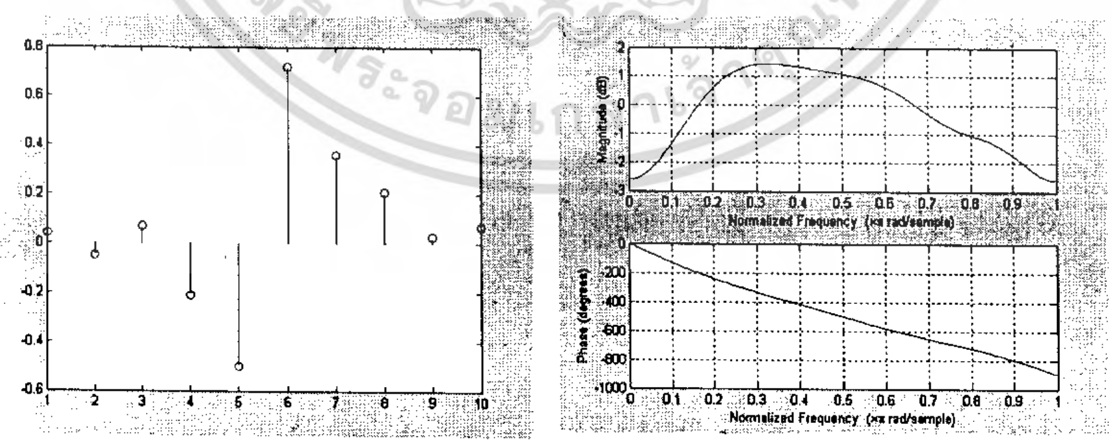
4.1 ผลการทดลองการจำลองการทำงานด้วยโปรแกรมแมทแลบ

ในการทดลองนี้ เป็นการจำลองการทำงานด้วยโปรแกรมแมทแลบ เพื่อพิจารณาการทำงานวงจรอิควอไลเซอร์ที่ออกแบบโดยใช้อัลกอริทึมค่าเฉลี่ยกำลังสองน้อยสุด มาทำการปรับแต่งช่องสัญญาณแบบสุ่มที่มีสัญญาณรบกวนแบบ AWGN ซึ่งใช้หลักการของวงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัดคำนวณหาค่าสัมประสิทธิ์ โดยในที่นี้จะใช้วงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัดอันดับ 4 และช่องสัญญาณแบบโพรอากิส-เอ [7 pp.631] ที่มีผลตอบสนองอิมพัลส์และผลตอบสนองทางความถี่ของช่องสัญญาณดังรูป



(ก) ผลตอบสนองอิมพัลส์ของช่องสัญญาณ (ข) ผลตอบสนองทางความถี่ของช่องสัญญาณ

รูปที่ 4.1 แสดงผลตอบสนองอิมพัลส์และผลตอบสนองทางความถี่ของช่องสัญญาณที่สุ่มขึ้น

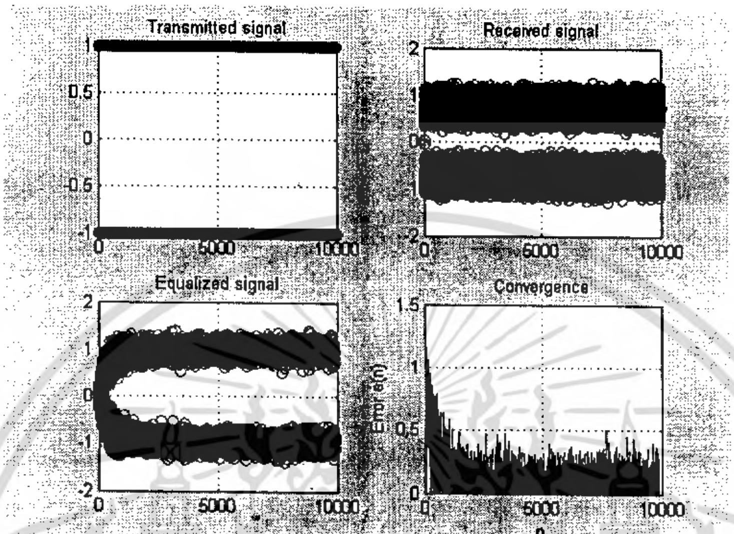


(ก) ผลตอบสนองอิมพัลส์ของช่องสัญญาณ (ข) ผลตอบสนองทางความถี่ของช่องสัญญาณ

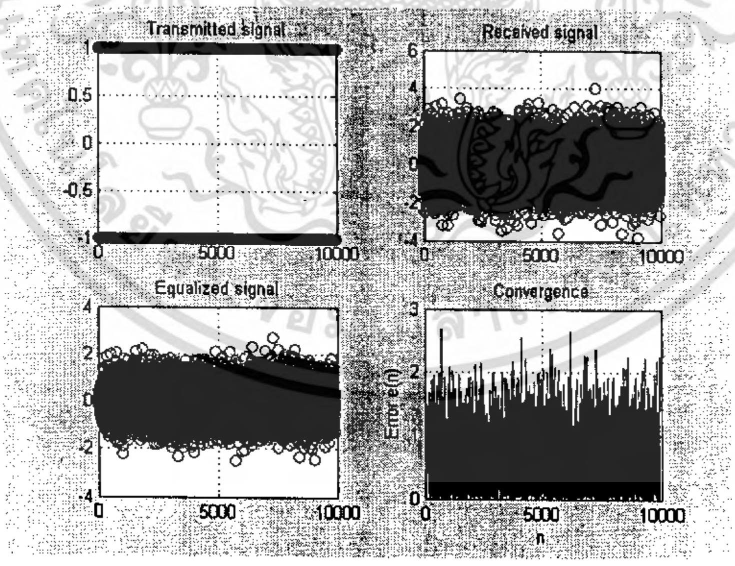
รูปที่ 4.2 แสดงผลตอบสนองอิมพัลส์และผลตอบสนองทางความถี่ของช่องสัญญาณแบบโพรอากิส-เอ

4.1.1 เมื่อสัญญาณอินพุทเป็นสัญญาณพีเอเอ็ม 2 ระดับ

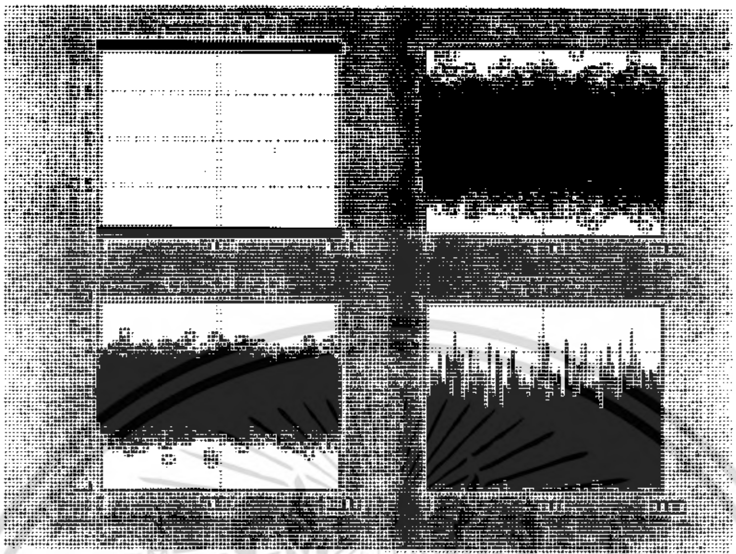
ป้อนสัญญาณพีเอเอ็ม 2 ระดับ ผ่านช่องสัญญาณที่มีค่า SNR แตกต่างกัน แล้วทำการอิควอไลเซอร์ ได้ผลการทดลองดังรูป



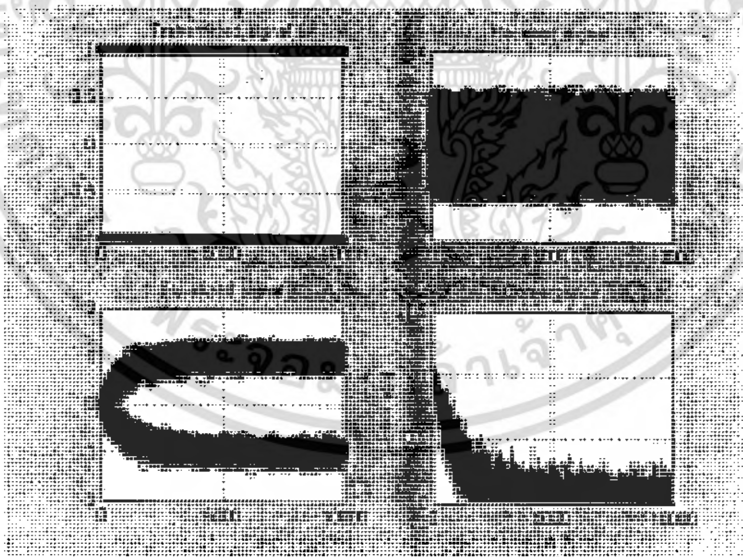
รูปที่ 4.3 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = 20



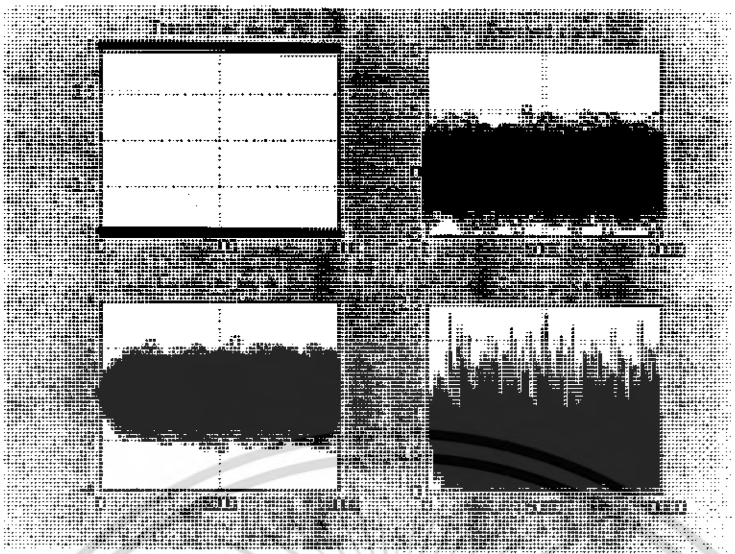
รูปที่ 4.4 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = 0



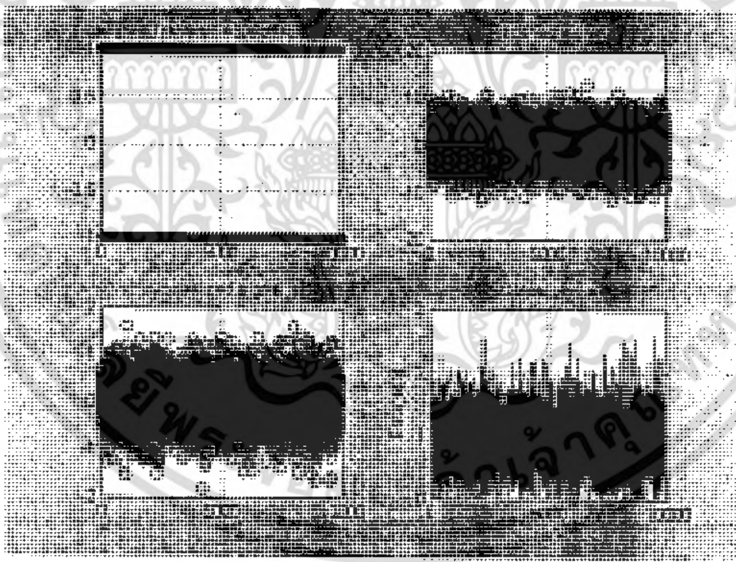
รูปที่ 4.5 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = -10



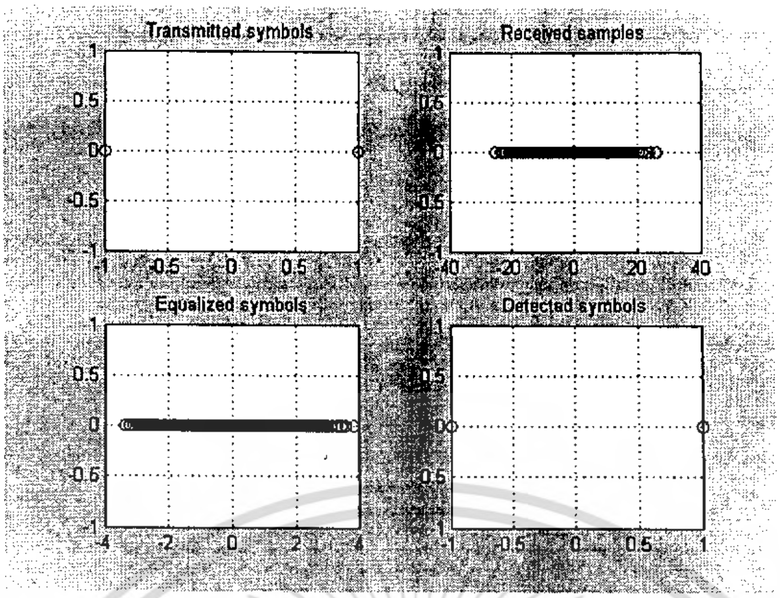
รูปที่ 4.6 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณโพรอากิส-เอที่มีค่า SNR = 20



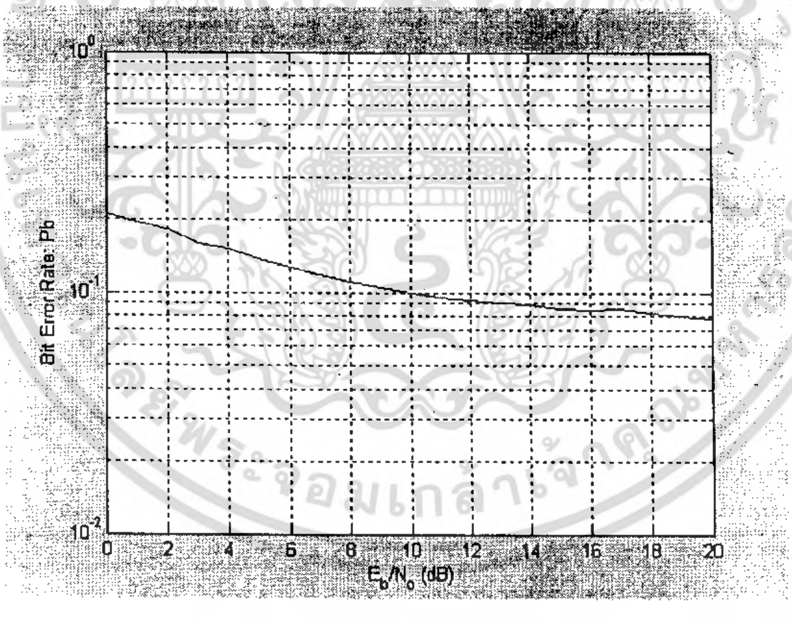
รูปที่ 4.7 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณโพรอากิส-เอที่มีค่า SNR = 0



รูปที่ 4.8 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 2 ระดับในช่องสัญญาณโพรอากิส-เอที่มีค่า SNR = -10



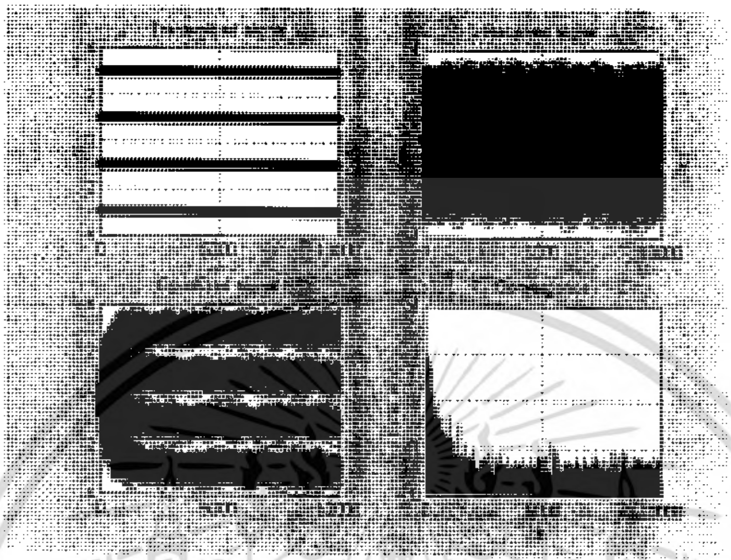
รูปที่ 4.9 แสดง Constellation ของสัญญาณพีเอเอ็ม 2 ระดับ



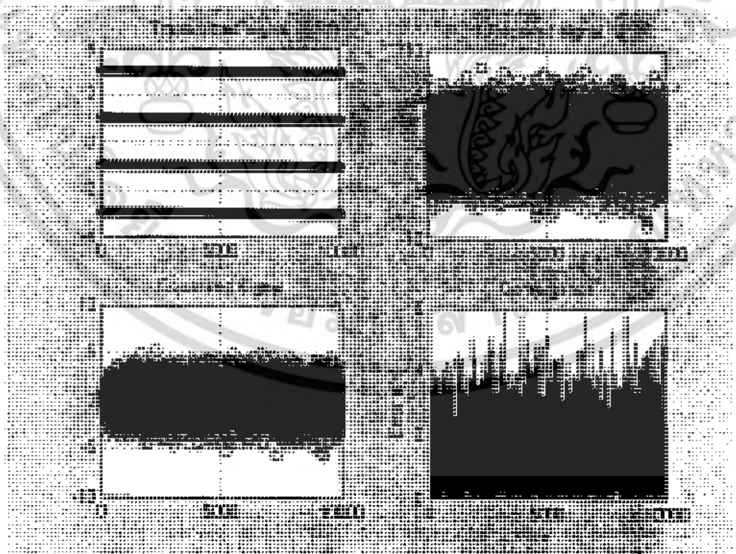
รูปที่ 4.10 แสดงค่า BER ของสัญญาณพีเอเอ็ม 2 ระดับ

4.1.2 เมื่อสัญญาณอินพุทเป็นสัญญาณพีเอเอ็ม 4 ระดับ

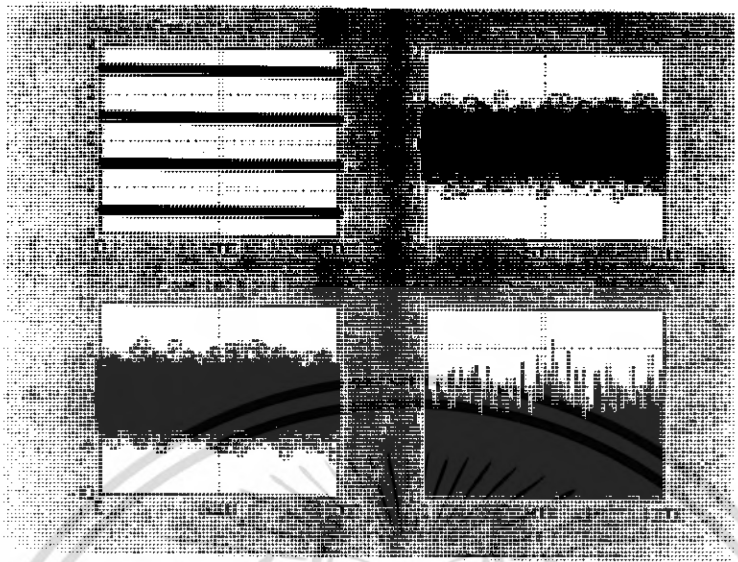
ป้อนสัญญาณพีเอเอ็ม 4 ระดับ ผ่านช่องสัญญาณที่มีค่า SNR แตกต่างกัน แล้วทำการอิควอไลเซอร์ ได้ผลการทดลองดังรูป



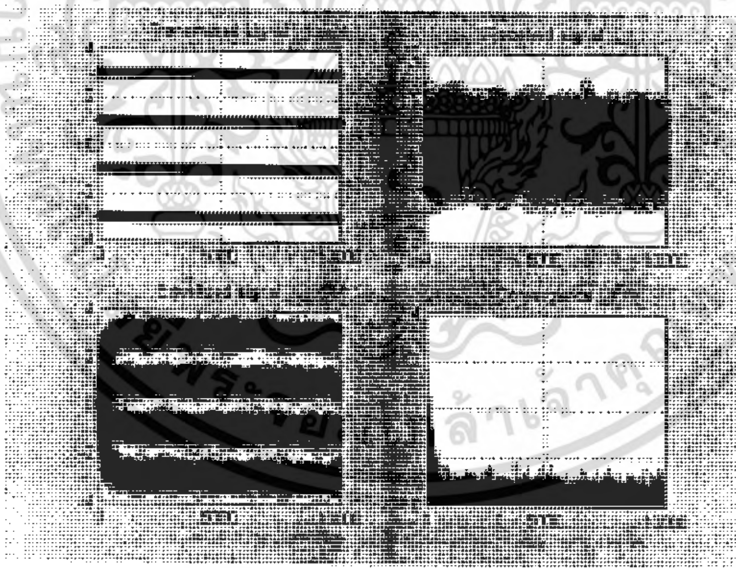
รูปที่ 4.11 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = 20



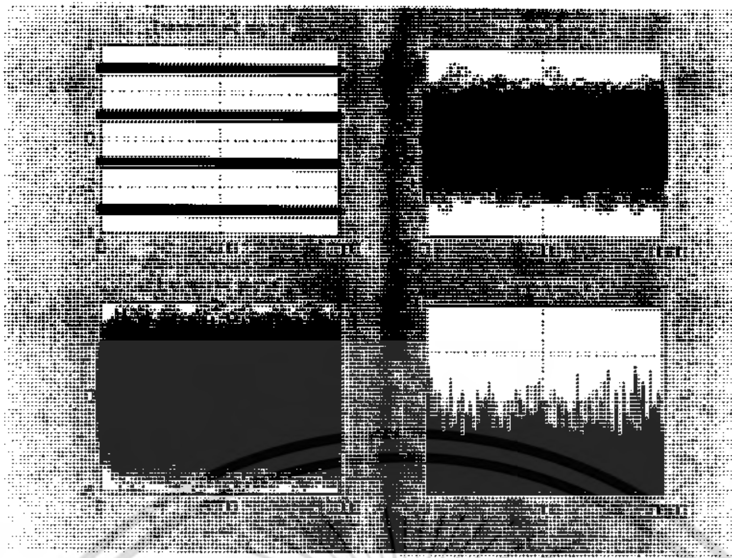
รูปที่ 4.12 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = 0



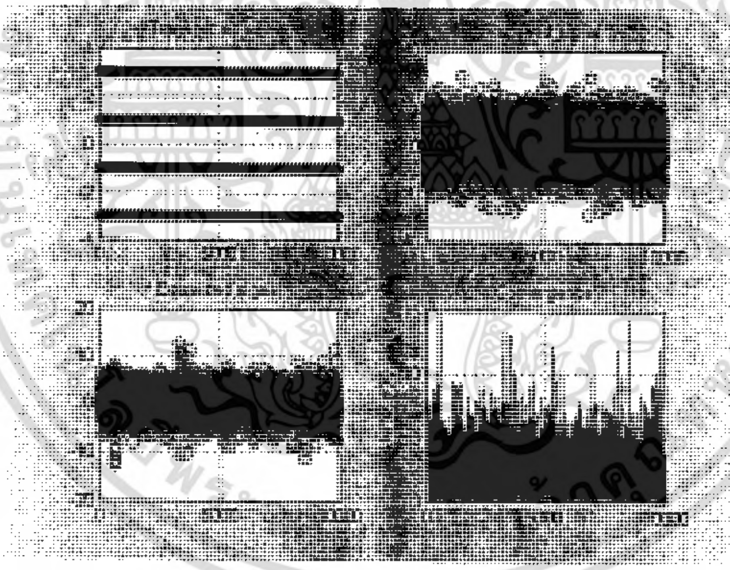
รูปที่ 4.13 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณสุ่มที่มีค่า SNR = -10



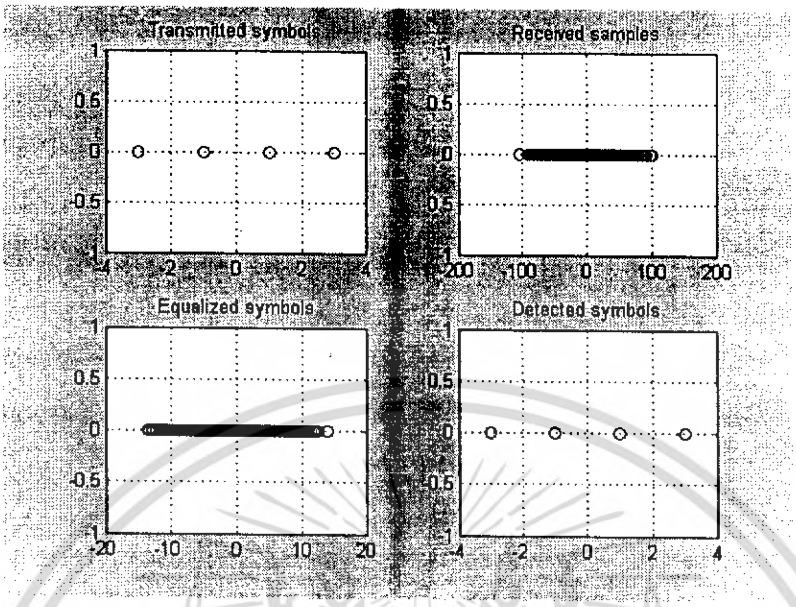
รูปที่ 4.14 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณโพรอากิส-เอที่มีค่า SNR = 20



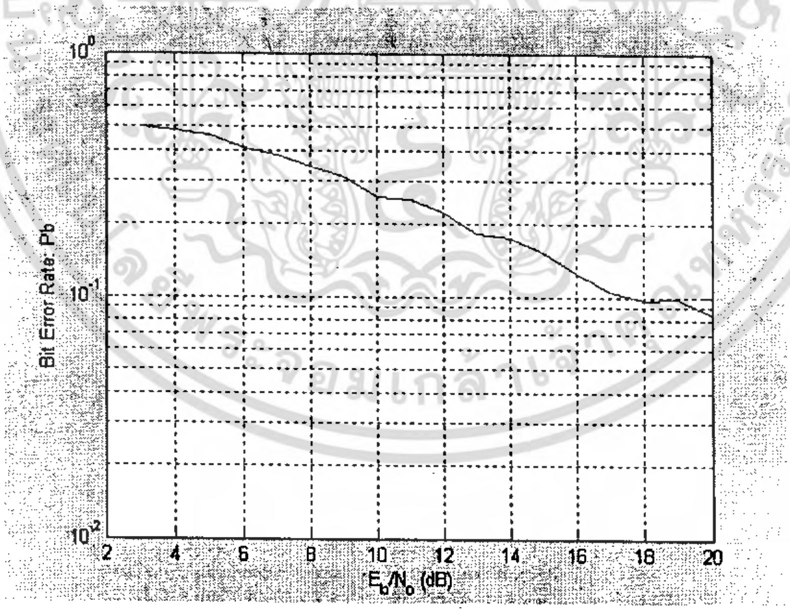
รูปที่ 4.15 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณโพรอากิส-เอที่มีค่า SNR = 0



รูปที่ 4.16 แสดงผลที่ได้เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับในช่องสัญญาณโพรอากิส-เอที่มีค่า SNR = -10



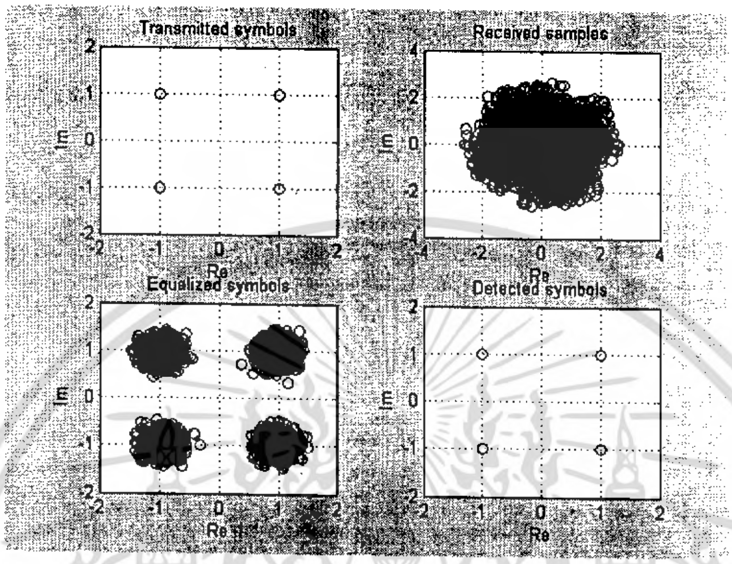
รูปที่ 4.17 แสดง Constellation ของสัญญาณพีเอเอ็ม 4 ระดับ



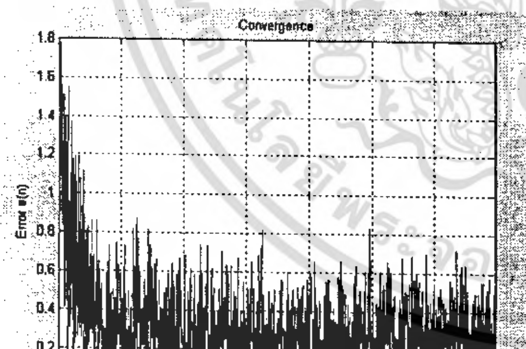
รูปที่ 4.18 แสดงBER ของสัญญาณพีเอเอ็ม 4 ระดับ

4.1.3 เมื่อสัญญาณอินพุทเป็นสัญญาณคิวพีเอสเค

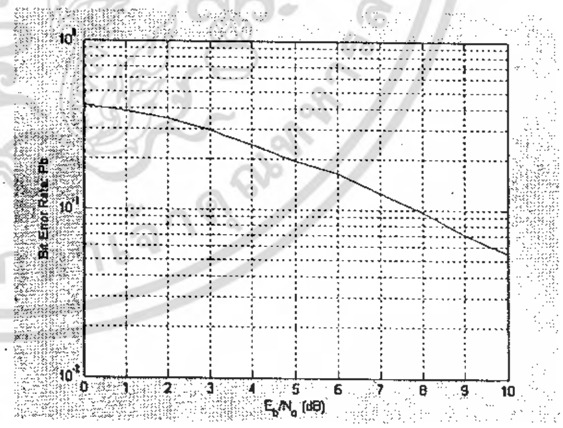
ป้อนสัญญาณคิวพีเอสเคผ่านช่องสัญญาณที่มีค่า SNR แตกต่างกัน แล้วทำการอิควอไลเซอร์ได้ผลการทดลองดังรูป



รูปที่ 4.19 แสดง Constellation ของสัญญาณคิวพีเอสเค



(ก) แสดงค่าความผิดพลาดที่เกิดขึ้น



(ข) แสดงBER ของสัญญาณคิวพีเอสเค

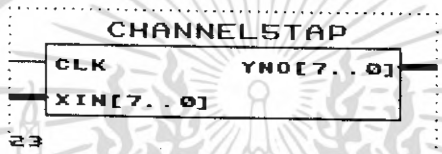
รูปที่ 4.20 แสดงผลที่ได้เมื่อป้อนสัญญาณอินพุทแบบคิวพีเอสเค ในช่องสัญญาณทดลอง

4.2 ผลการทดลองการออกแบบวงจรโดยใช้ภาษาวีเอชดีแอล

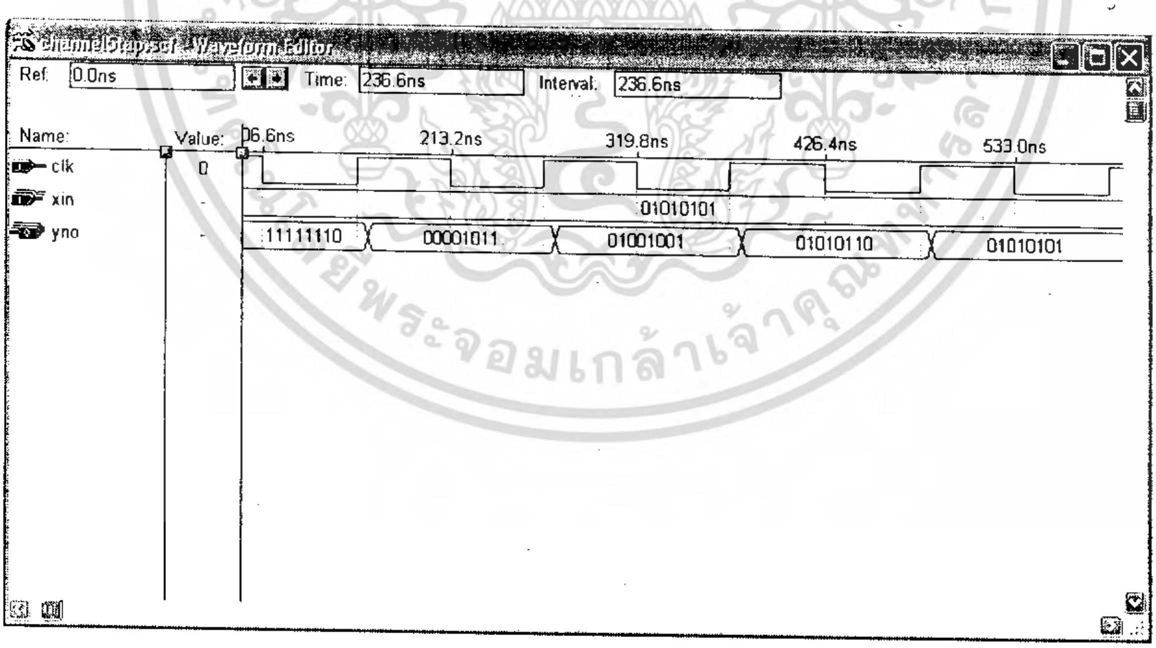
ในส่วนการทดลองนี้จะมีการออกแบบวงจร 2 ส่วน คือในส่วนของการสร้างแบบจำลองช่องสัญญาณ AWGN (AWGN channel emulator) และส่วนของการสร้างวงจรอิควอไลเซอร์

4.2.1 วงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัด

นำค่าสัมประสิทธิ์ที่ได้มาทำการออกแบบวงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัดด้วยภาษาวีเอชดีแอล โดยใช้วิธีการคูณแบบ Baugh-Wooley



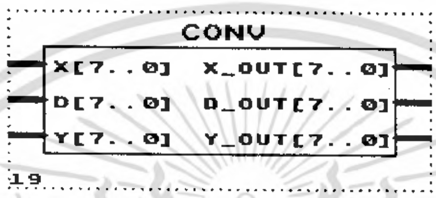
รูปที่ 4.21 แสดงสัญลักษณ์ของวงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัด



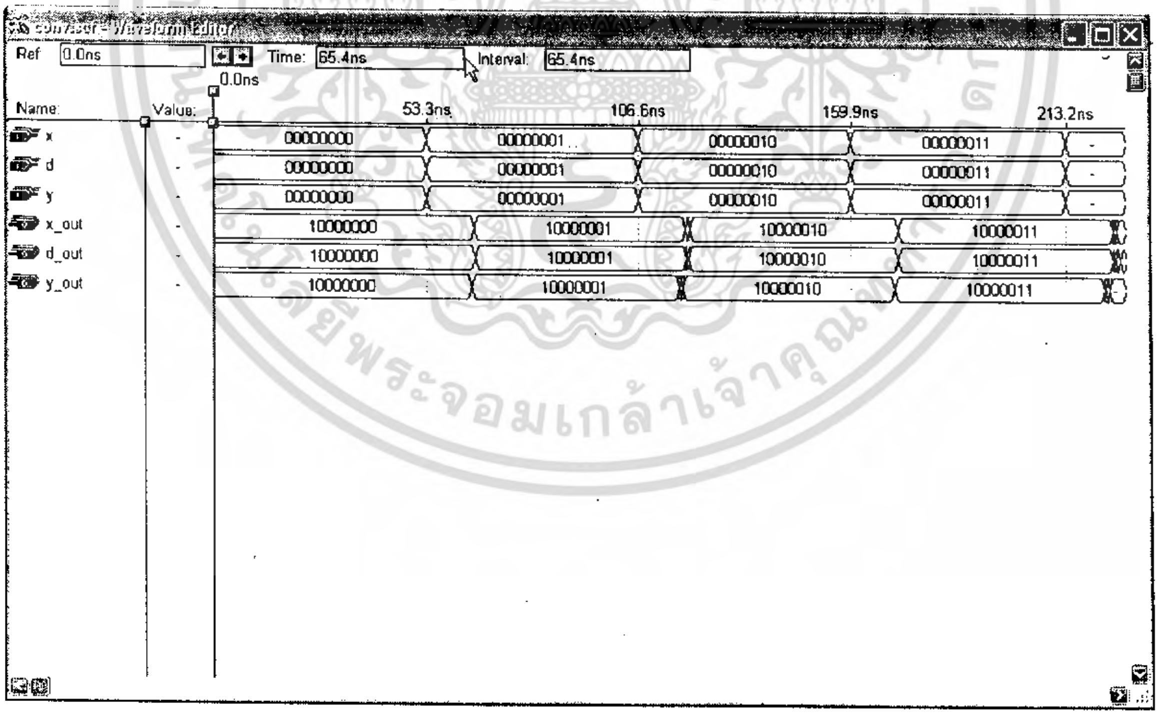
รูปที่ 4.22 แสดงผลการจำลองการทำงานของวงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัด

4.2.2 วงจรกลับบิทเครื่องหมาย

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



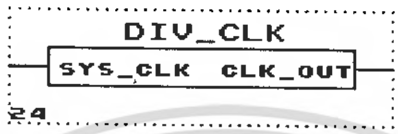
รูปที่ 4.23 แสดงสัญลักษณ์ของวงจรกลับบิทเครื่องหมาย



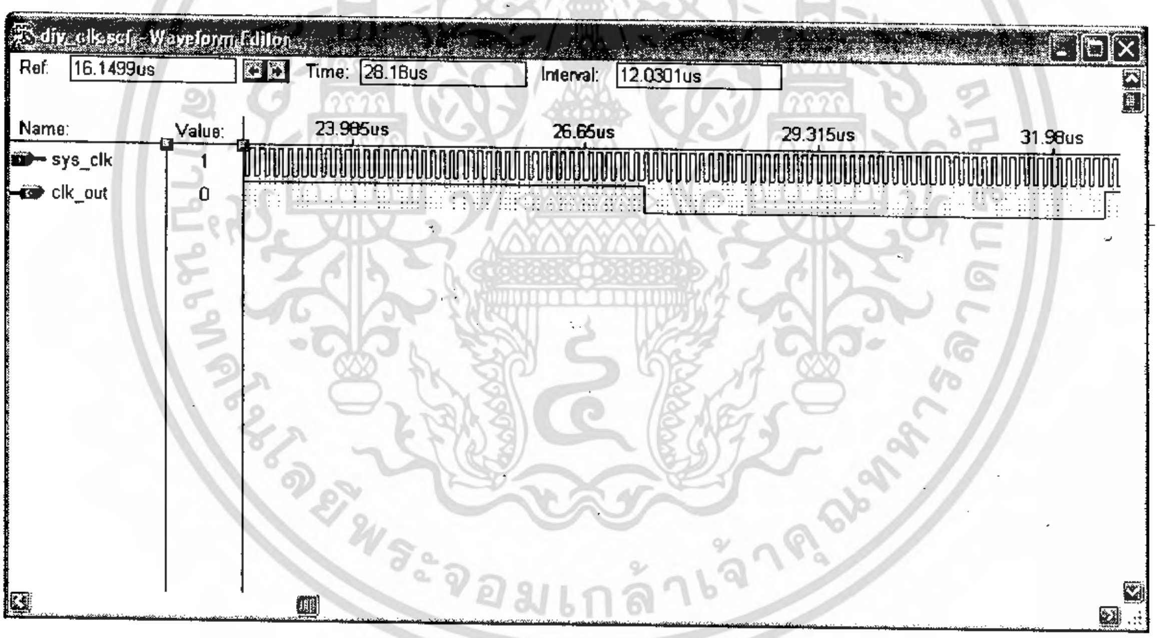
รูปที่ 4.24 แสดงผลการจำลองการทำงานของวงจรกลับบิทเครื่องหมาย

4.2.3 วงจรหารความถี่

เป็นวงจรที่ใช้หารความถี่ของสัญญาณนาฬิกาที่สร้างจากออสซิลเลเตอร์ (Oscillator) ให้มีความถี่ที่เหมาะสมเพื่อใช้ในการกำหนดค่าความถี่แซมปลิ้ง (Sampling Frequency)



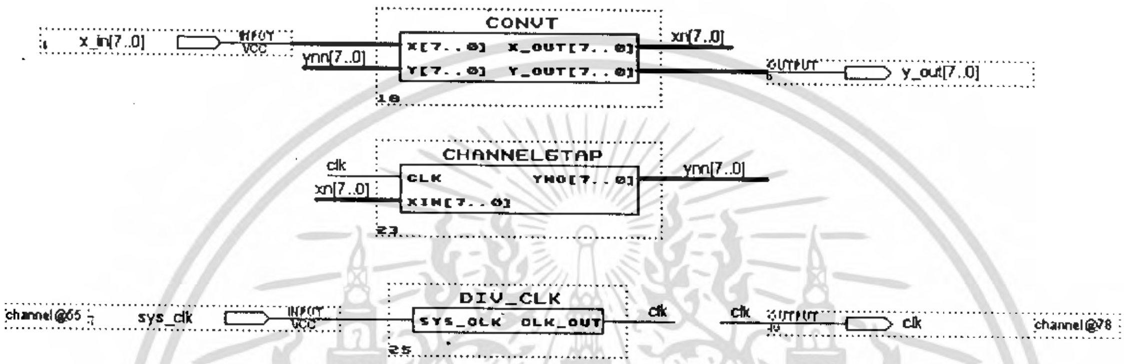
รูปที่ 4.25 แสดงสัญลักษณ์ของวงจรหารความถี่



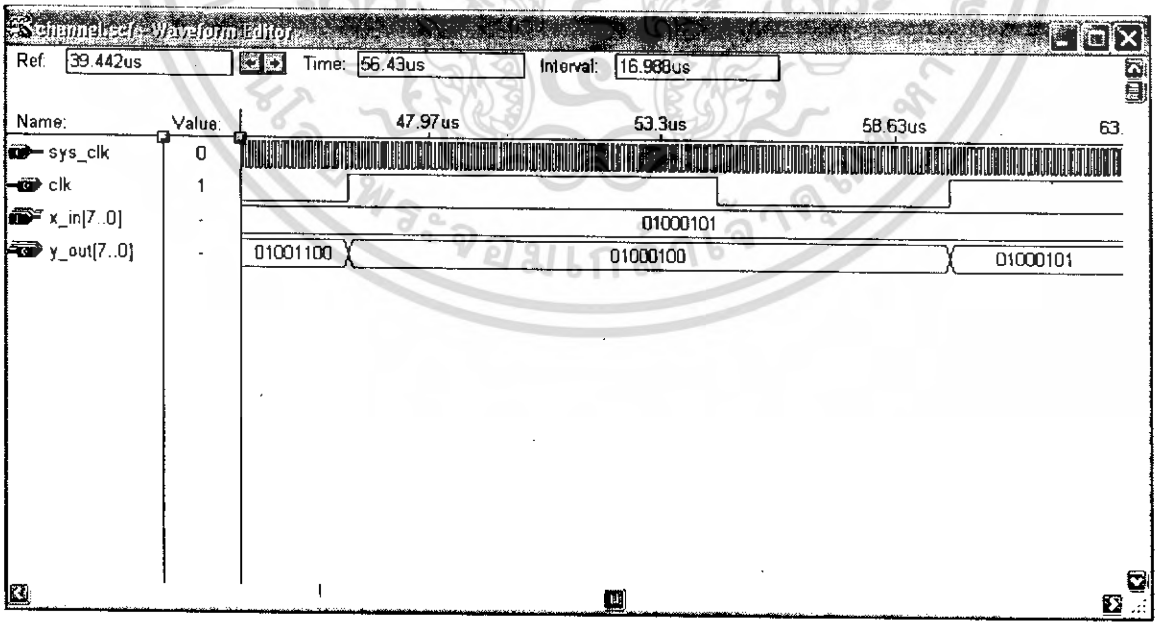
รูปที่ 4.26 แสดงผลการจำลองการทำงานของวงจรหารความถี่

4.2.4 ส่วนการสร้างแบบจำลองช่องสัญญาณ

นำวงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัดที่ได้มาทำการรวมวงจรให้เป็นแบบจำลองช่องสัญญาณโดยใช้สัญญาณอินพุทจากวงจรกลับบิทเครื่องหมาย และใช้สัญญาณนาฬิกาจากวงจรหารความถี่



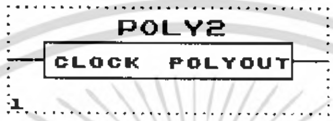
รูปที่ 4.27 แสดงสัญลักษณ์ของวงจรแบบจำลองช่องสัญญาณ



รูปที่ 4.28 แสดงผลการจำลองการทำงานของวงจรแบบจำลองช่องสัญญาณ

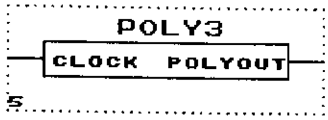
4.2.5 วงจร LFSR

เป็นวงจรสร้างสัญญาณไบนารี่แบบสุ่ม สำหรับในการสร้างสัญญาณรบกวนแบบ AWGN โดยการกำหนดโพลิโนเมียล m แตกต่างกัน ในที่นี้จะกำหนดโพลิโนเมียล m ต่างกัน 8 ค่า คือ m = 2, m = 3, m = 5, m = 6, m = 7, m = 8, m = 9 และ m = 11

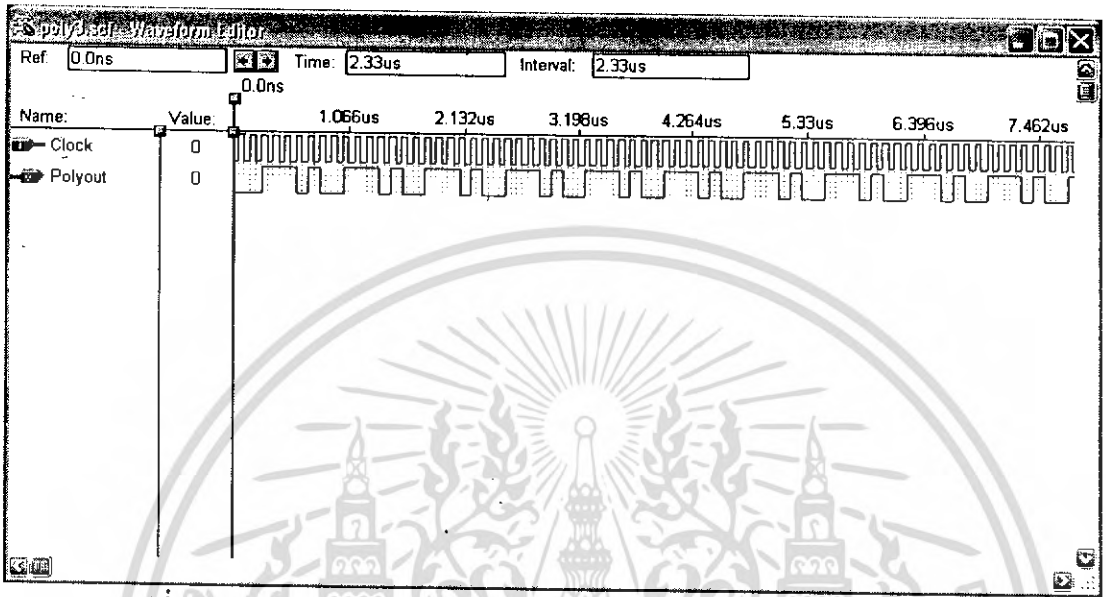


รูปที่ 4.29 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 2

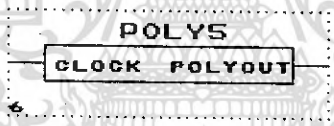
รูปที่ 4.30 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 2



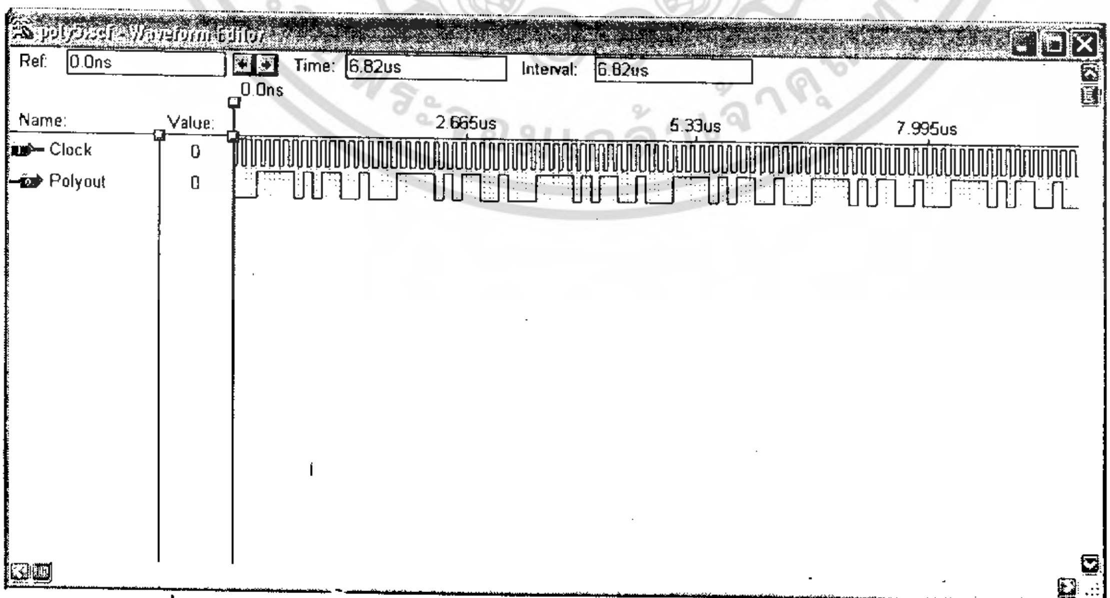
รูปที่ 4.31 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 3



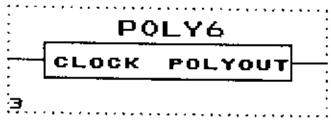
รูปที่ 4.32 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 3



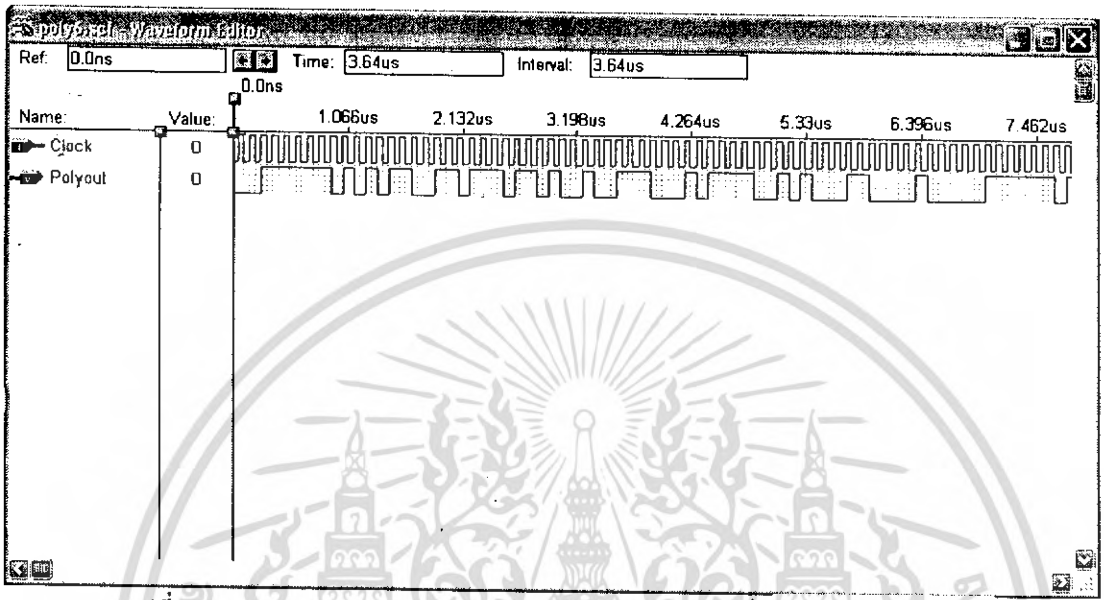
รูปที่ 4.33 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 5



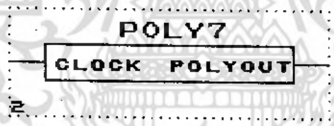
รูปที่ 4.34 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 5



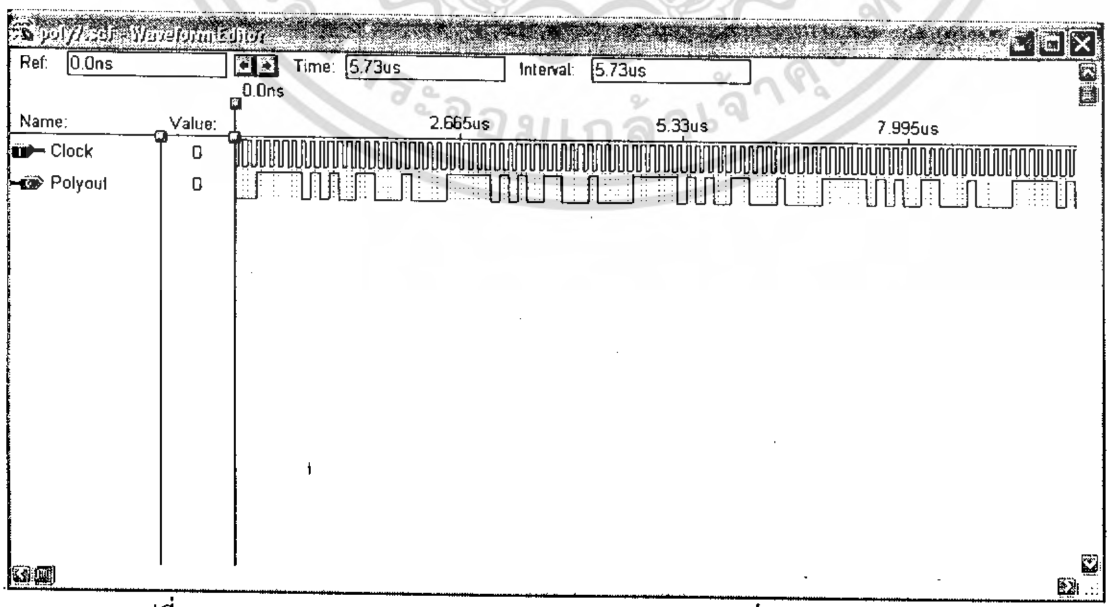
รูปที่ 4.35 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 6



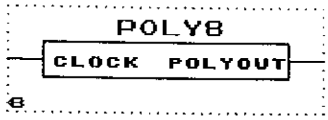
รูปที่ 4.36 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 6



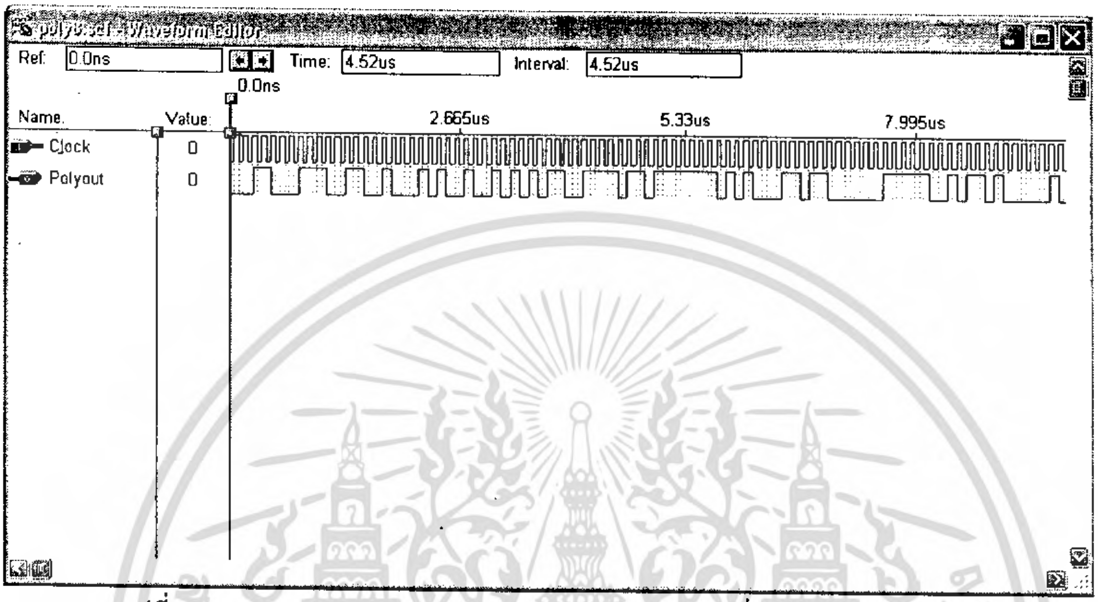
รูปที่ 4.37 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 7



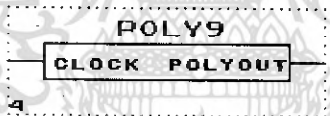
รูปที่ 4.38 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 7



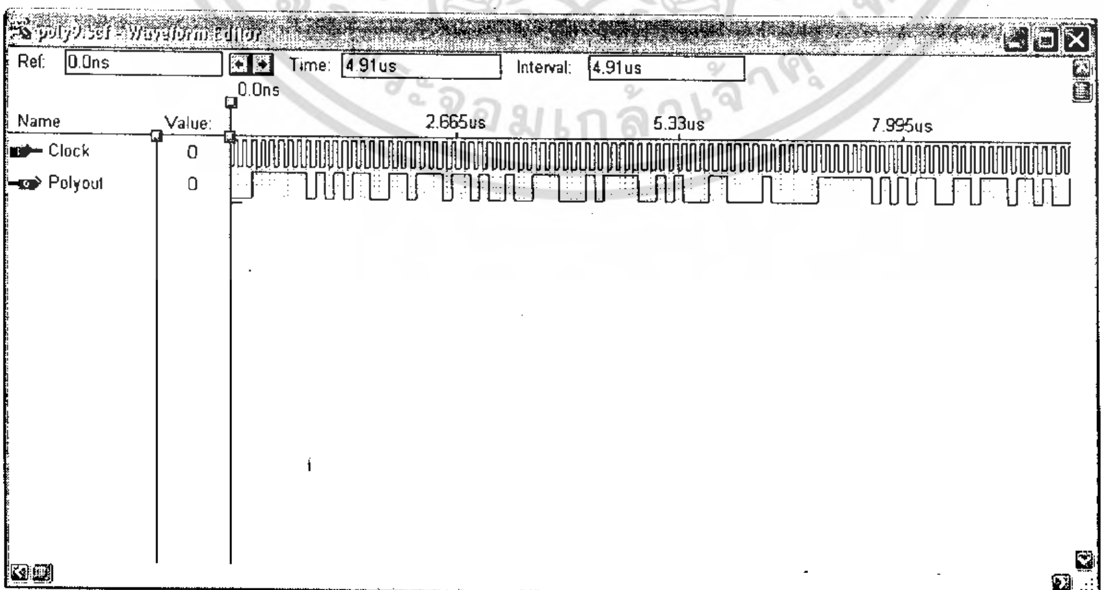
รูปที่ 4.39 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 8



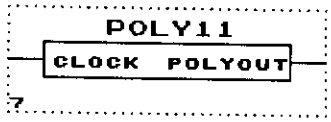
รูปที่ 4.40 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 8



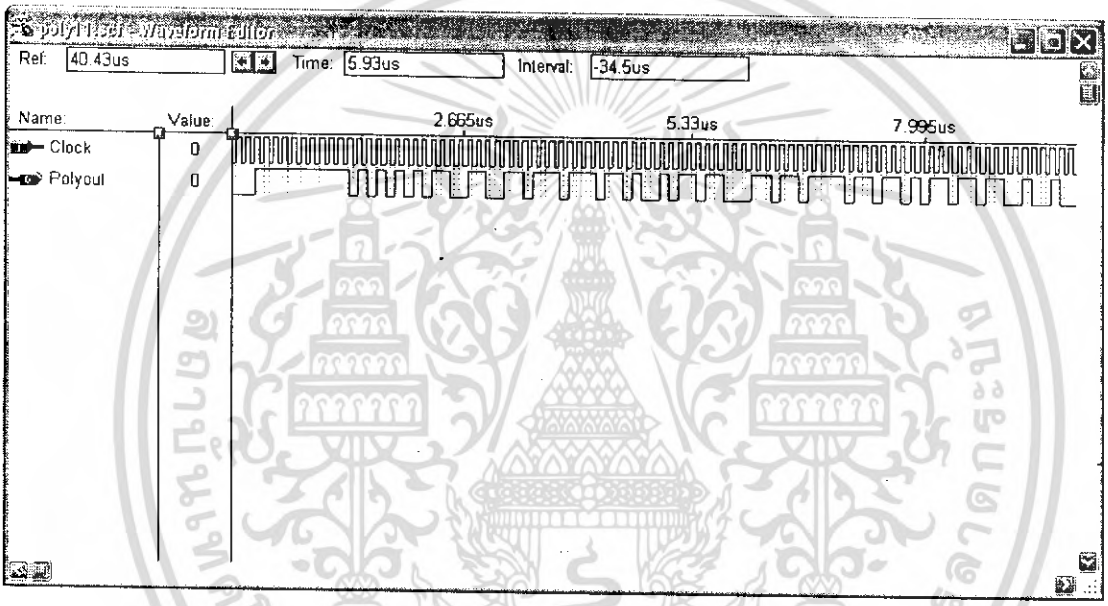
รูปที่ 4.41 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 9



รูปที่ 4.42 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 9



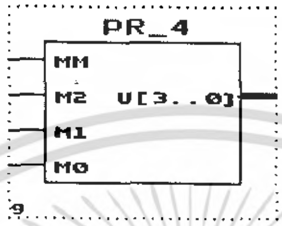
รูปที่ 4.43 แสดงสัญลักษณ์ของวงจร LFSR ที่มีโพลิโนเมียล m = 11



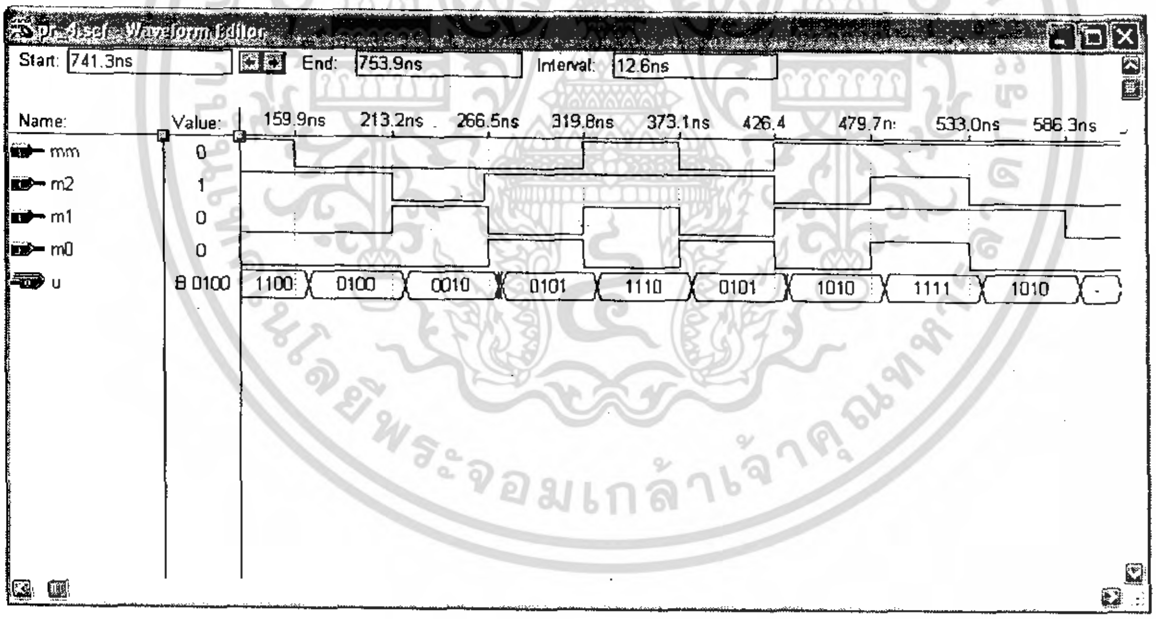
รูปที่ 4.44 แสดงผลการจำลองการทำงานของวงจร LFSR ที่มีโพลิโนเมียล m = 11

4.2.6 วงจรรวมบิท

เป็นวงจรที่นำบิทของแต่ละอินพุทมาทำการรวมบิทกันเป็นเอาท์พุท อินพุทเข้า 1 บิท เอาท์พุทออก 4 บิท.



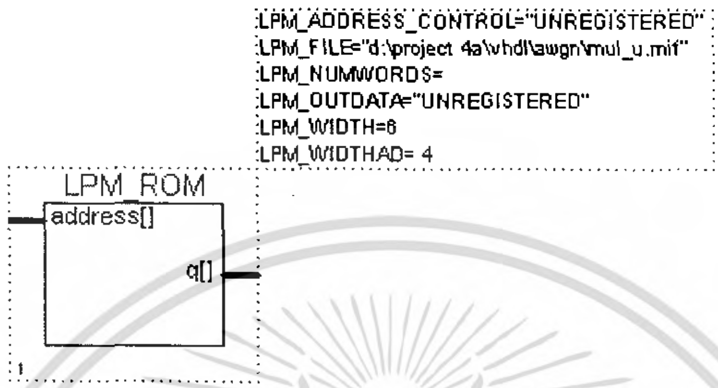
รูปที่ 4.45 แสดงสัญลักษณ์ของวงจรรวมบิท 4 บิท



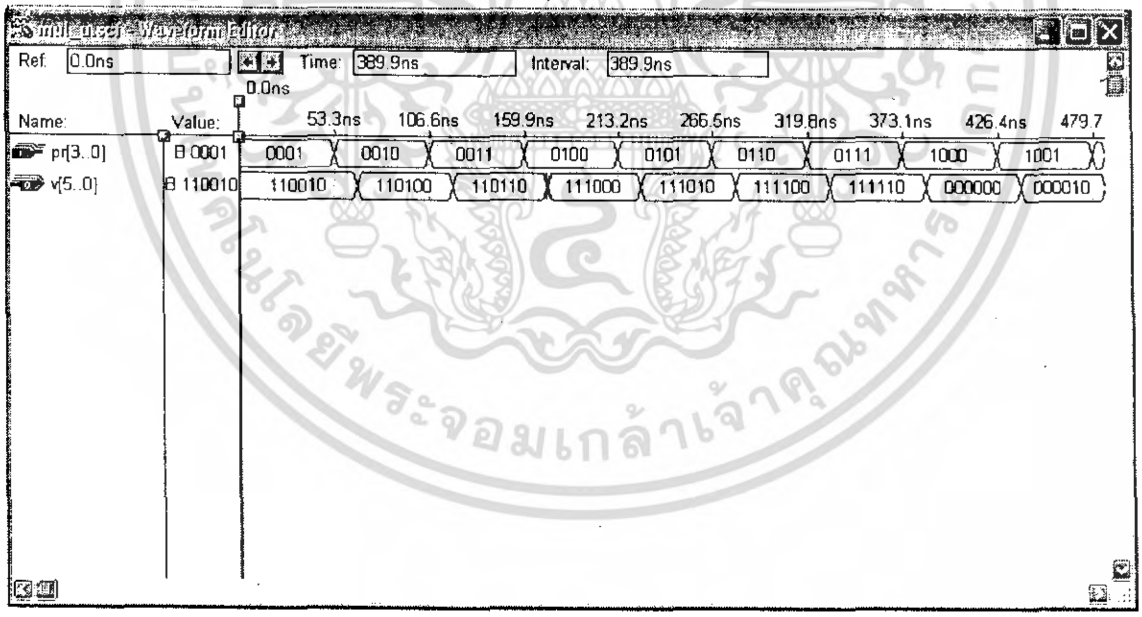
รูปที่ 4.46 แสดงผลการจำลองการทำงานของวงจรรวมบิท 4 บิท

4.2.7 วงจรเก็บค่าและเรียกค่าข้อมูล

เป็นวงจรที่ทำหน้าที่เก็บค่าผลลัพธ์ที่ได้จากการคำนวณของการสร้างสัญญาณ AWGN โดยสัญญาณอินพุทจะไปชี้ค่าที่แอดเดรสเพื่อเรียกค่าในแต่ละแอดเดรสออกมาเป็นเอาท์พุท



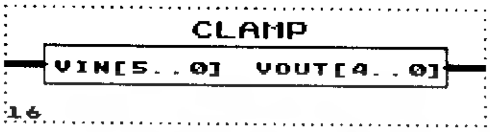
รูปที่ 4.47 แสดงสัญลักษณ์ของวงจรเก็บค่าและเรียกค่าข้อมูล



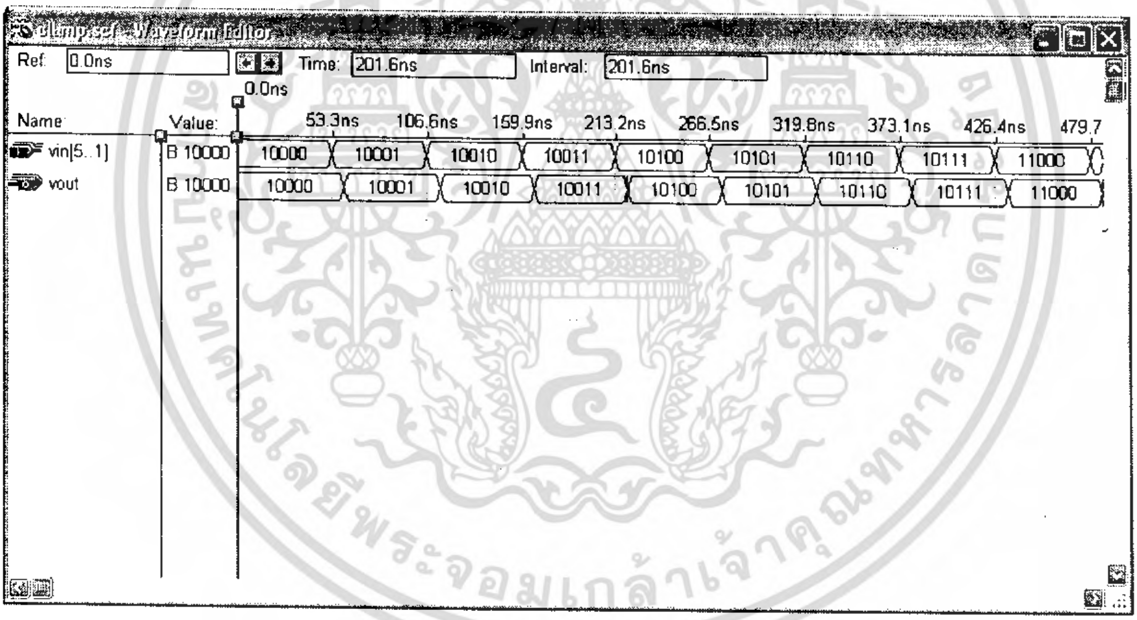
รูปที่ 4.48 แสดงผลการจำลองการทำงานของวงจรเก็บค่าและเรียกค่าข้อมูล

4.2.8 วงจรตัดบิท

เป็นวงจรที่ใช้ในการตัดบิท LSB (Least Significant Bit) ออก เพื่อนำสัญญาณเอาท์พุทไปเป็นสัญญาณอินพุทของวงจรที่ใช้จำนวนบิทน้อยกว่า



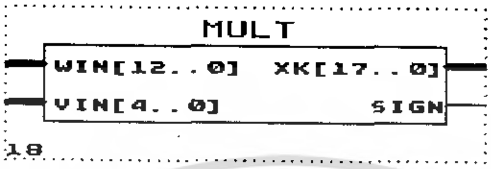
รูปที่ 4.49 แสดงสัญลักษณ์ของวงจรตัดบิท



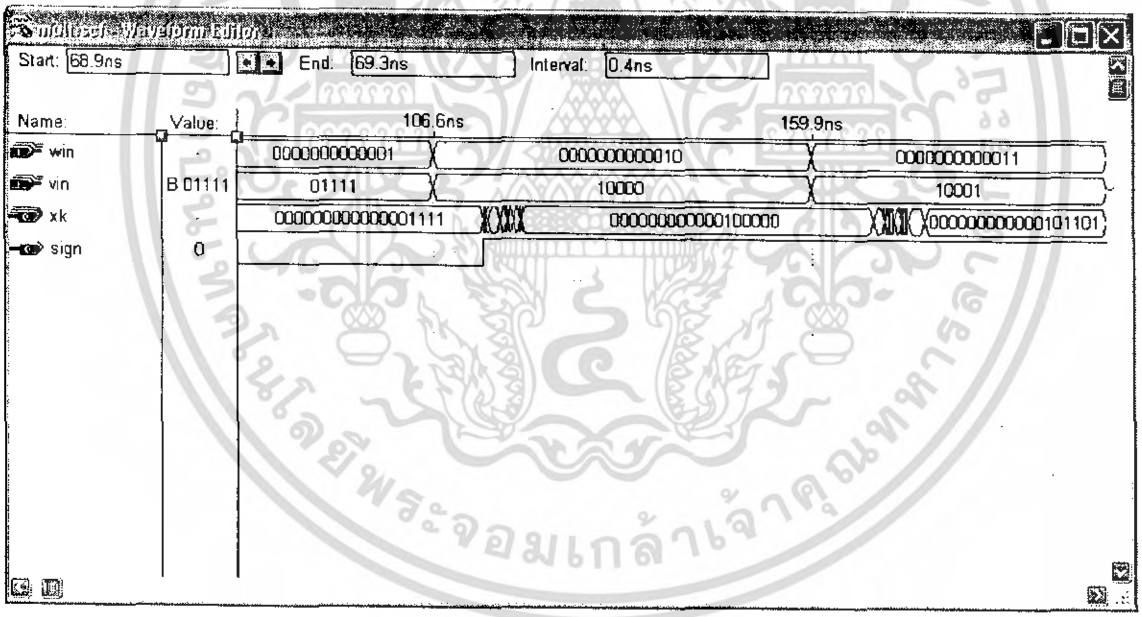
รูปที่ 4.50 แสดงผลการจำลองการทำงานของวงจรตัดบิท

4.2.9 วงจรคูณ

เป็นวงจรที่ทำหน้าที่หาค่าผลคูณระหว่างค่าที่วงจรเก็บค่าและเรียกค่าข้อมูล 2 วงจร เพื่อที่จะทำให้ได้ค่าที่สุ่มจากวงจร LFSR ทั้ง 8 วงจร เอาท์พุทที่ได้จะมีการนำค่า MSB มาเข้าวงจรทูคอมพลีเมนท์ก่อนจะเข้าส่วนของเซนทรัลลิมิต



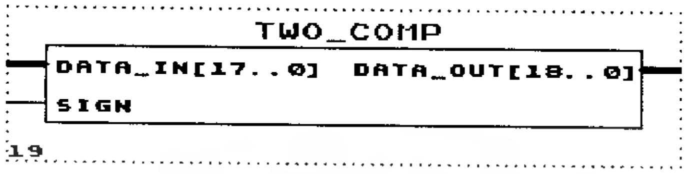
รูปที่ 4.51 แสดงสัญลักษณ์ของวงจรคูณ



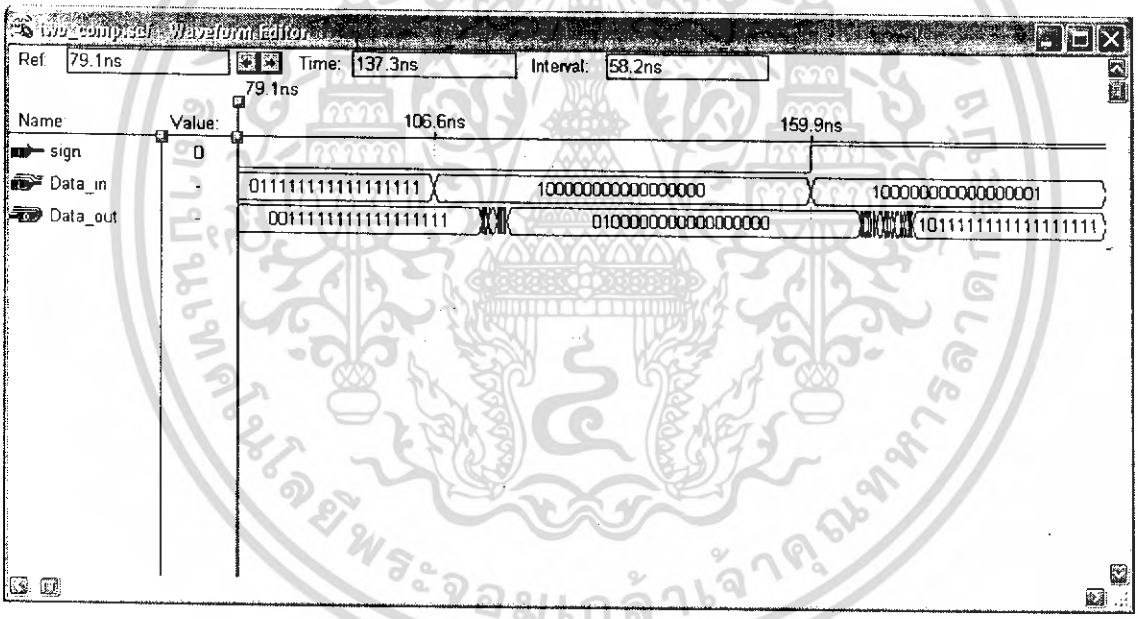
รูปที่ 4.52 แสดงผลการจำลองการทำงานของวงจรคูณ

4.2.10 วงจรทูคอมพลีเมนท์

เป็นวงจรที่ทำทูคอมพลีเมนท์ โดยถ้า Sign เป็น 1 จะทำการทูคอมพลีเมนท์ แต่ถ้า Sign เป็น 0 จะไม่ทำการทูคอมพลีเมนท์ เอาท์พุทที่ได้จะมีการเพิ่มอีก 1 บิทเพื่อป้องการโอเวอร์โฟลว์ (overflow)



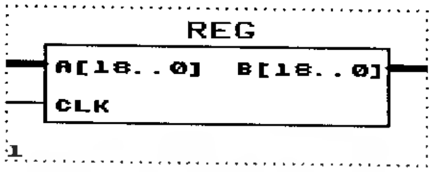
รูปที่ 4.53 แสดงสัญลักษณ์ของวงจรทูคอมพลีเมนท์



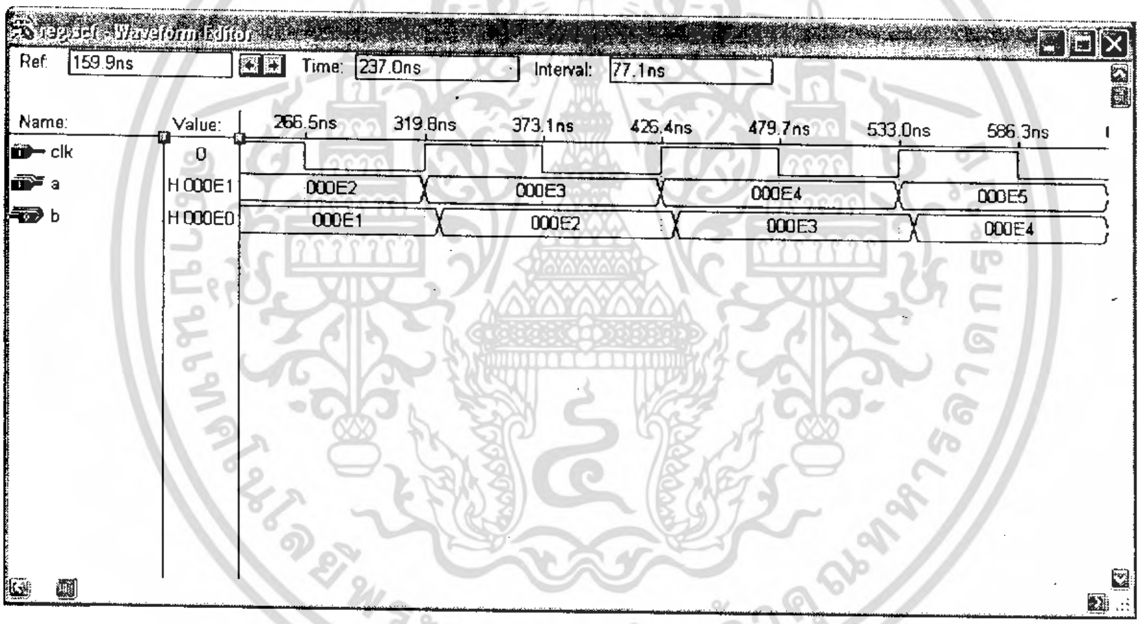
รูปที่ 4.54 แสดงผลการจำลองการทำงานของวงจรทูคอมพลีเมนท์

4.2.11 วงจรบัฟเฟอร์

เป็นวงจรที่ทำหน้าที่ใช้เป็นตัวกันชนระหว่างวงจรภายในและใช้ในการโหลดค่าออกเอาท์พุท โดยใช้ขาสัญญาณ clk ในการโหลดค่าออกเอาท์พุท



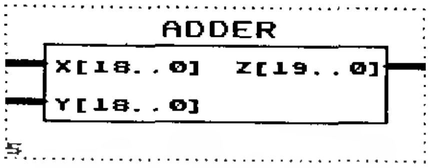
รูปที่ 4.55 แสดงสัญลักษณ์ของวงจรบัฟเฟอร์



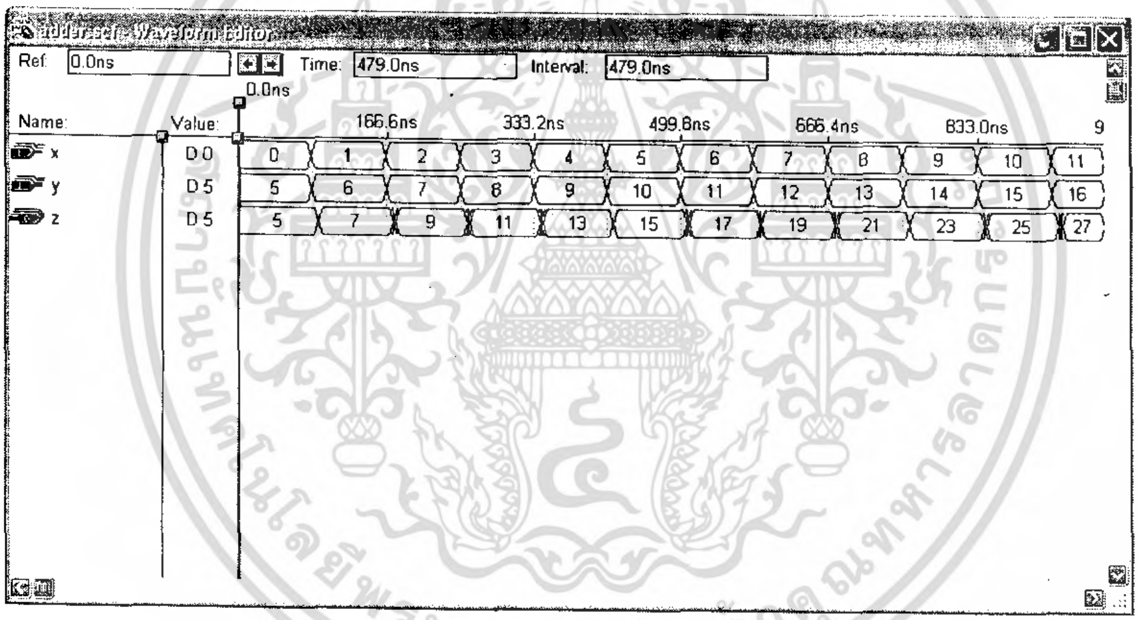
รูปที่ 4.56 แสดงผลการจำลองการทำงานของวงจรบัฟเฟอร์

4.2.12 วงจรบวก

เป็นวงจรที่ทำหน้าที่หาค่าผลรวมระหว่างค่าอินพุทของวงจรบัฟเฟอร์กับค่าเอาท์พุทของวงจรบัฟเฟอร์ขนาด 19 บิท



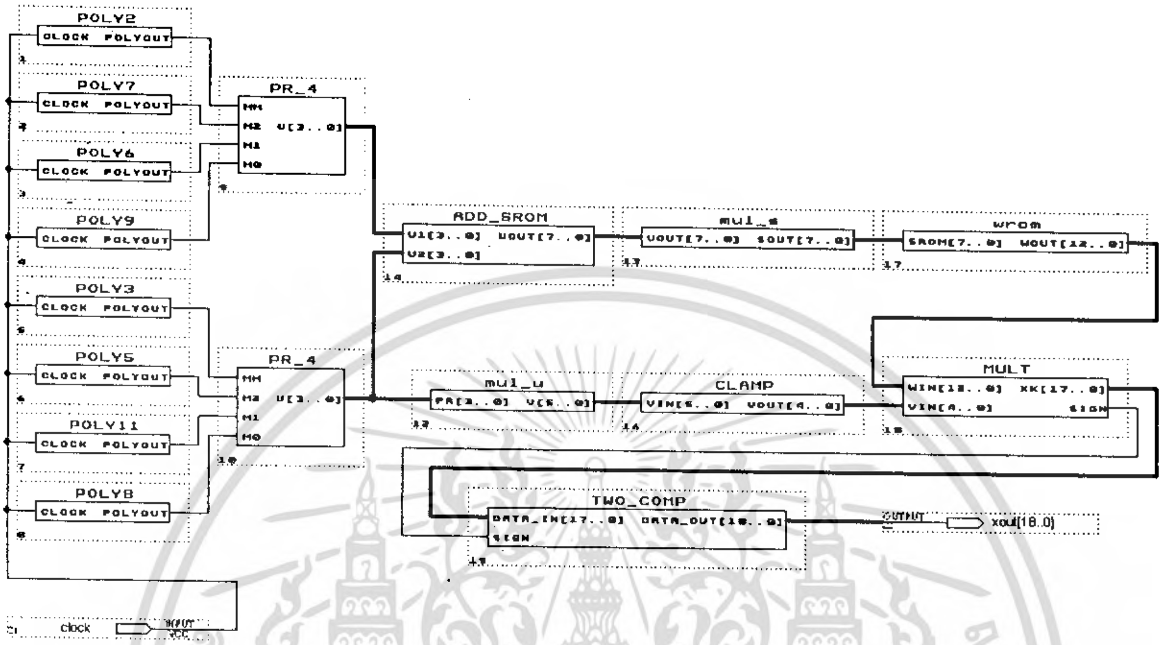
รูปที่ 4.57 แสดงสัญลักษณ์ของวงจรบวก



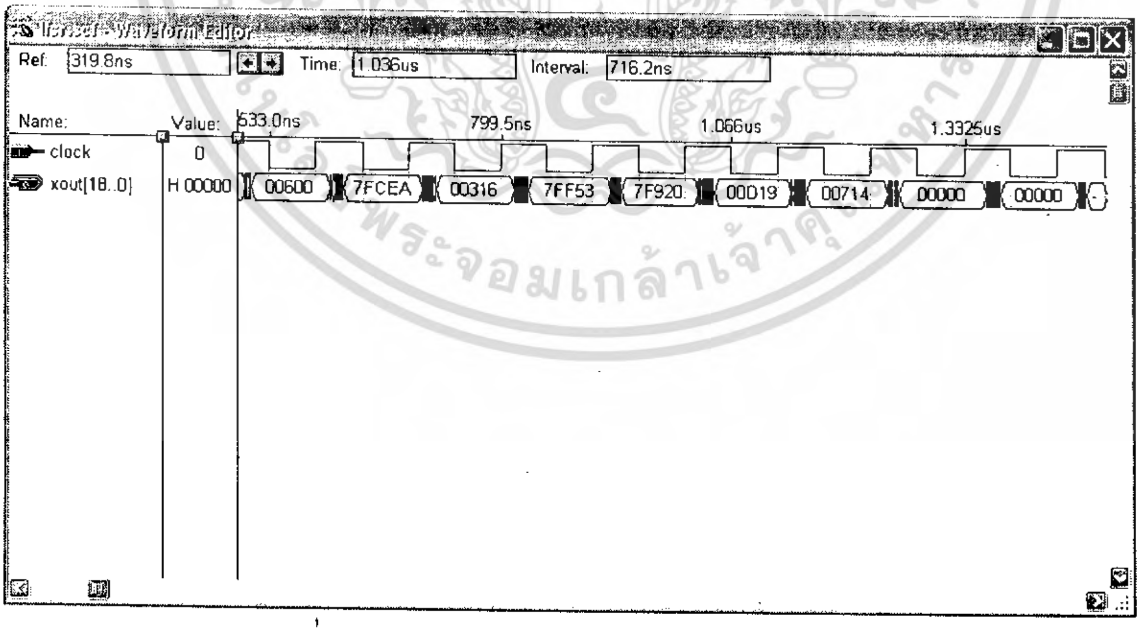
รูปที่ 4.58 แสดงผลการจำลองการทำงานของวงจรบวก

4.2.13 ส่วนการสร้างสัญญาณไบนารี่แบบสุ่มด้วยวิธีโพลาร์

เป็นการนำวงจร LFSR ที่มีโพลิโนเมียลต่างกัน 8 ตัวมาทำการรวมบิทเพื่อชี้แอดเดรสที่เก็บค่าที่คำนวณไว้ออกมาในแต่ละขั้นตอน และทำการคูณกันจะได้สัญญาณเอาท์พุทออกมา



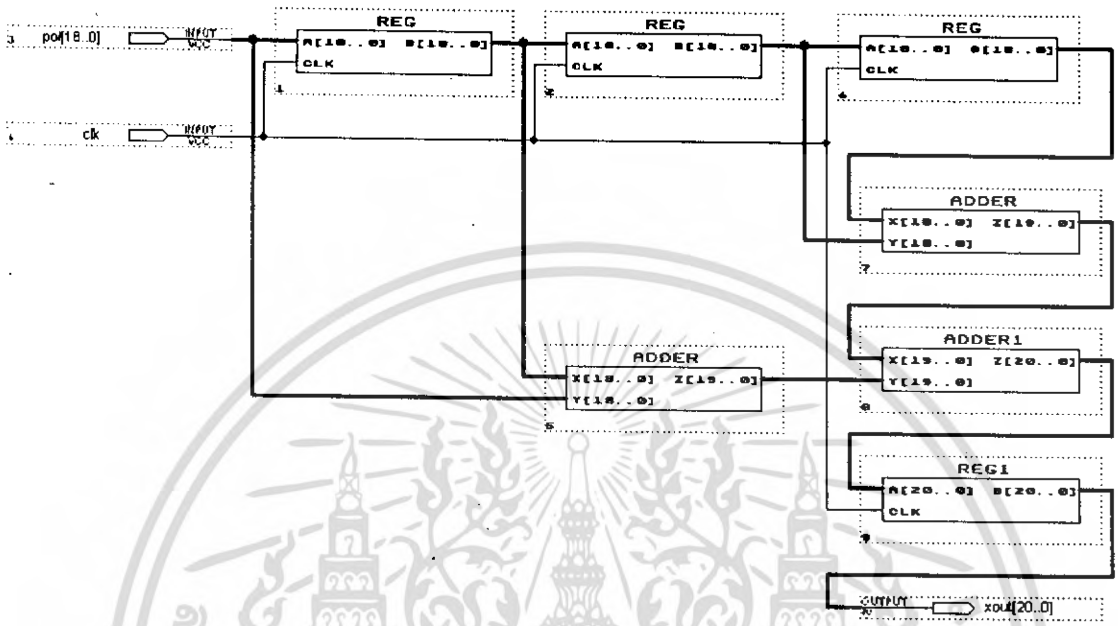
รูปที่ 4.59 แสดงสัญลักษณ์ของวงจรสัญญาณไบนารี่แบบสุ่มด้วยวิธีโพลาร์



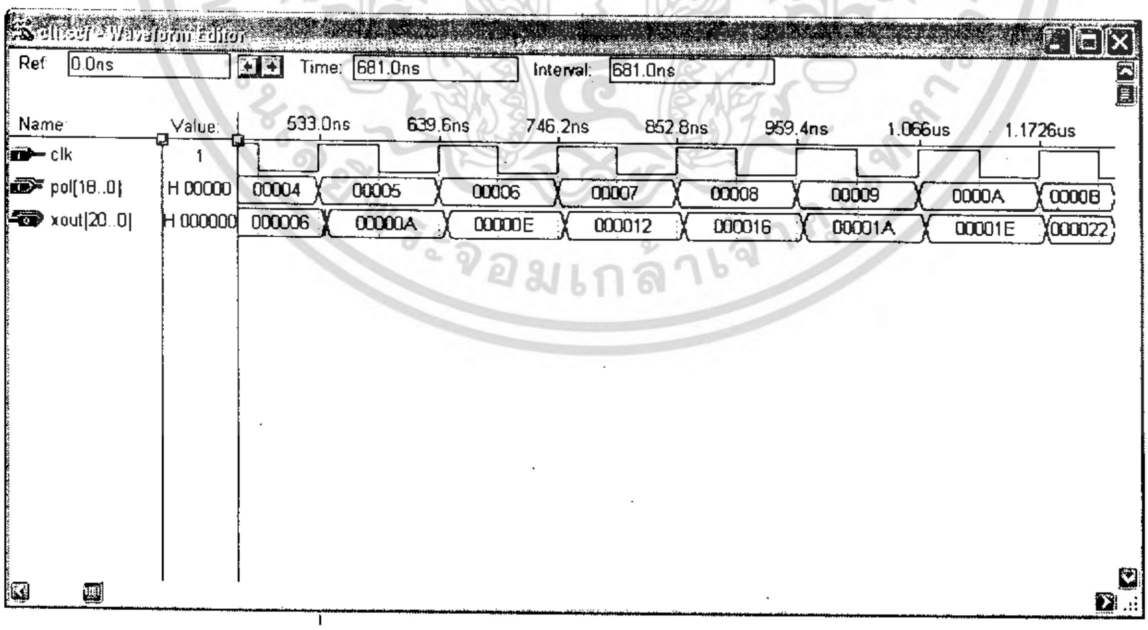
รูปที่ 4.60 แสดงผลการจำลองการทำงานของวงจรสัญญาณไบนารี่แบบสุ่มด้วยวิธีโพลาร์

4.2.14 ส่วนการสร้างเซนทรัลลิมิต

เป็นวงจรที่มีวงจรบัฟเฟอร์และวงจรบวกประกอบเข้ากัน ซึ่งที่เอาท์พุทจะมีการเพิ่ม 2 บิทเพื่อป้องกันการโอเวอร์โฟลว์



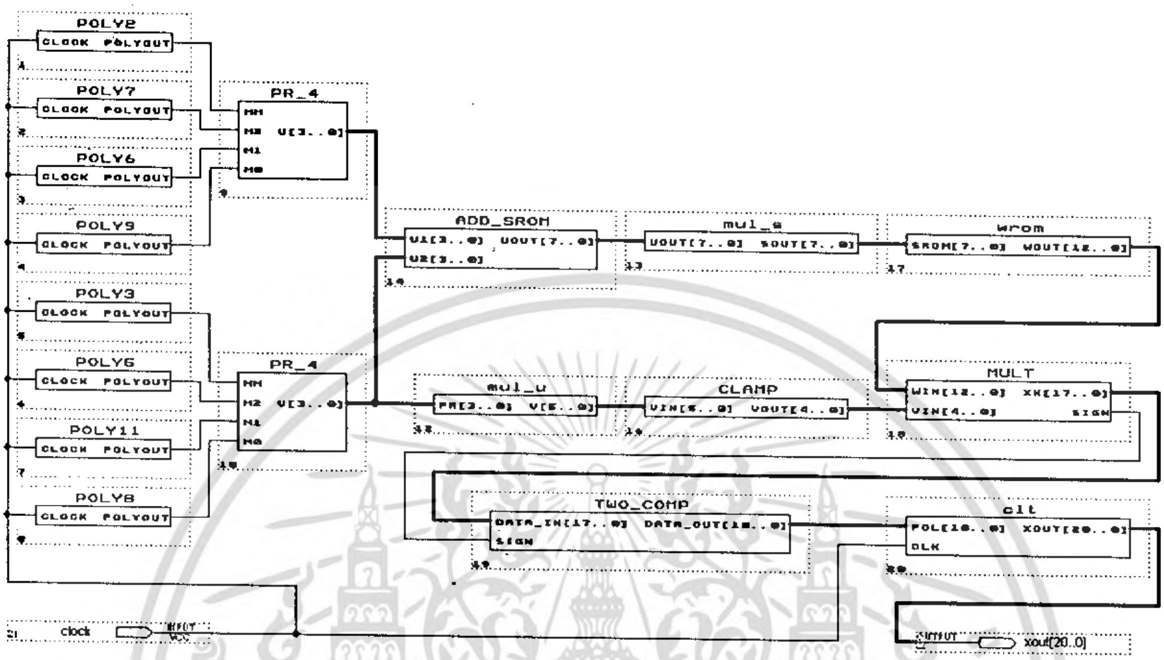
รูปที่ 4.61 แสดงสัญลักษณ์ของวงจรเซนทรัลลิมิต



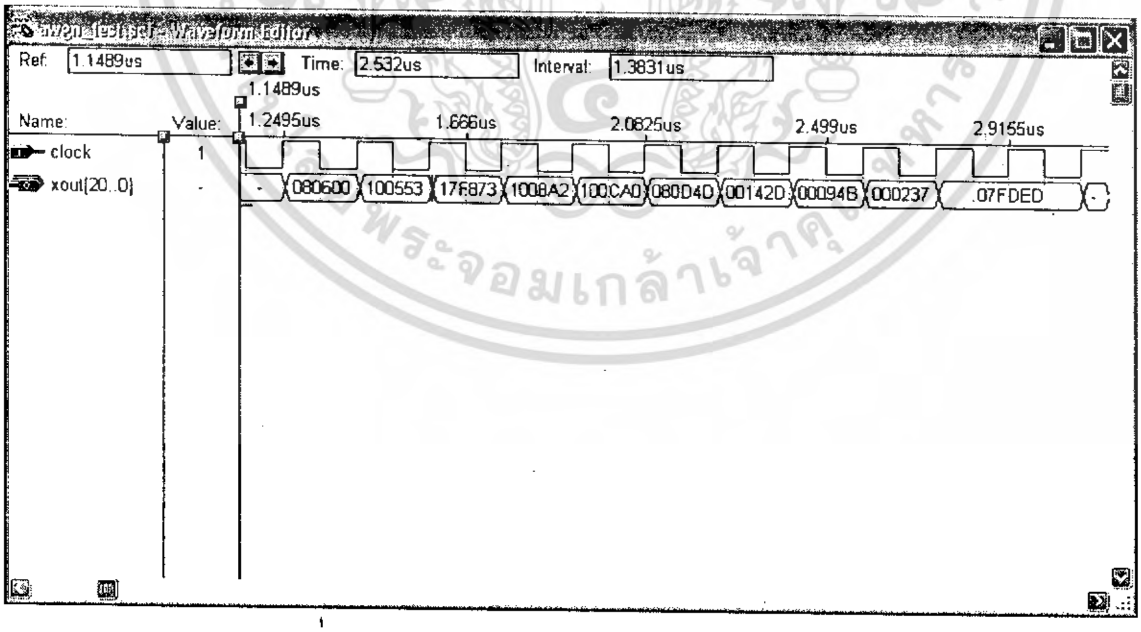
รูปที่ 4.62 แสดงผลการจำลองการทำงานของวงจรเซนทรัลลิมิต

4.2.15 ส่วนการสร้างสัญญาณ AWGN

เป็นการนำส่วนการสร้างสัญญาณไบนารี่แบบสุ่มด้วยวิธีโพลาร์และส่วนการสร้างเซนทรัลลิมิต ประกอบเข้ากัน เพื่อให้ได้สัญญาณ AWGN



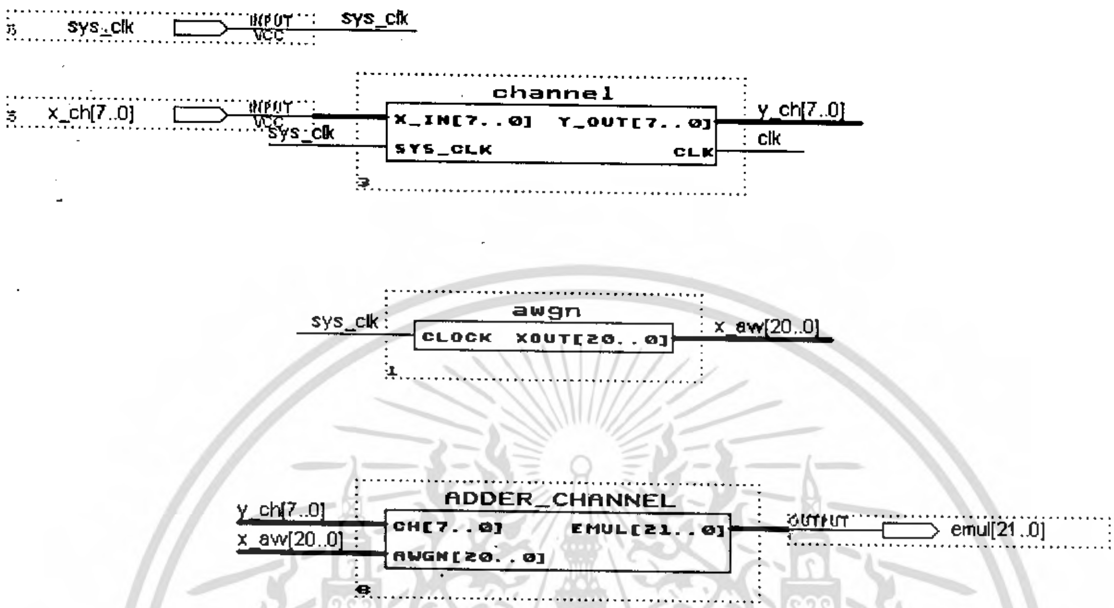
รูปที่ 4.63 แสดงสัญลักษณ์ของวงจรสร้างสัญญาณ AWGN



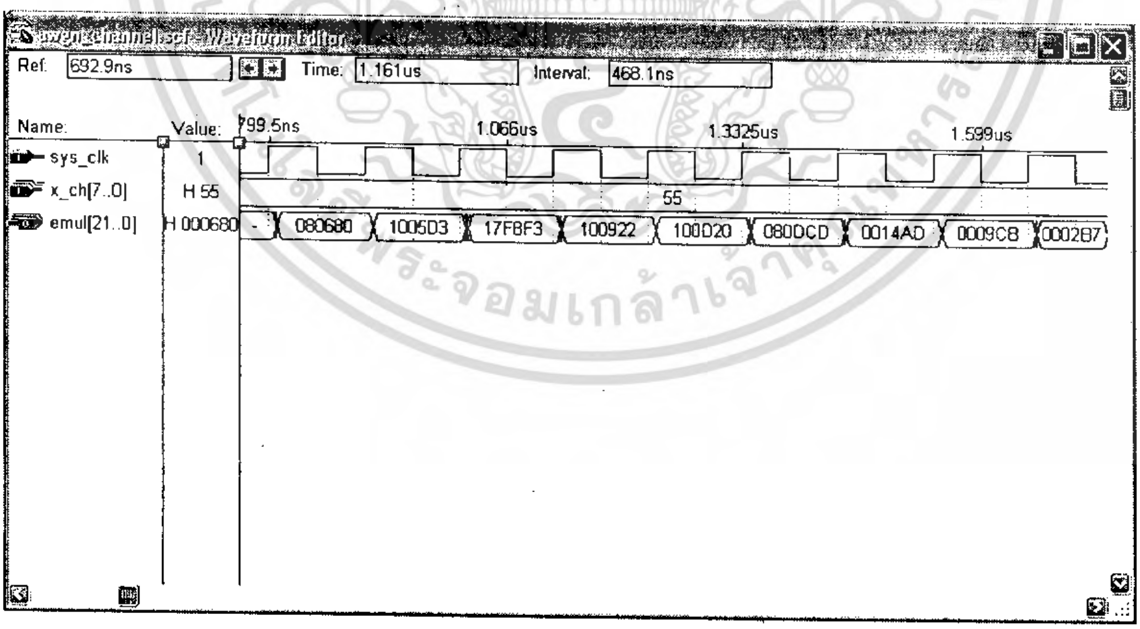
รูปที่ 4.64 แสดงผลการจำลองการทำงานของวงจรสร้างสัญญาณ AWGN

4.2.16 ส่วนการสร้างแบบจำลองช่องสัญญาณ AWGN

เป็นการนำส่วนการสร้างแบบจำลองช่องสัญญาณและส่วนการสร้างสัญญาณ AWGN ประกอบเข้ากัน เพื่อให้ได้แบบจำลองช่องสัญญาณ AWGN



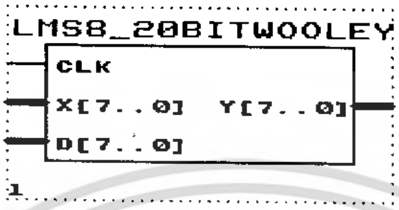
รูปที่ 4.65 แสดงสัญลักษณ์ของวงจรแบบจำลองช่องสัญญาณ AWGN



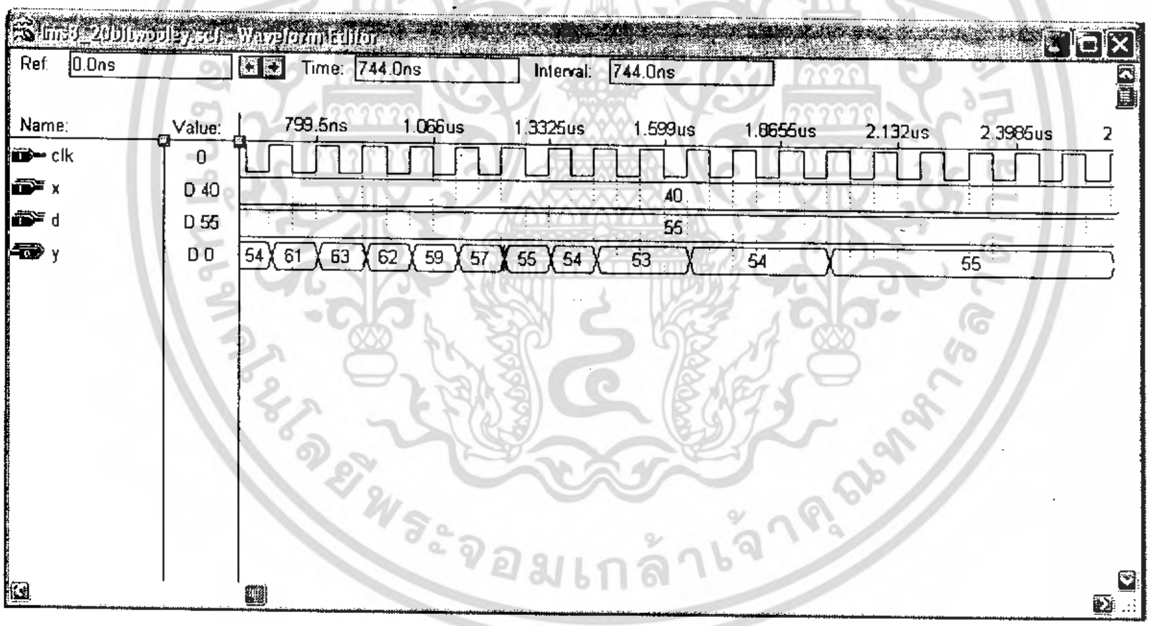
รูปที่ 4.66 แสดงผลการจำลองการทำงานของวงจรแบบจำลองช่องสัญญาณ AWGN

4.2.17 วงจรกรองความถี่ผ่านแบบปรับตัวได้

ในการสร้างอิควอไลเซอร์นั้นจะใช้หลักการของวงจรกรองความถี่ผ่านแบบปรับตัวได้ โดยในที่นี้จะใช้วงจรกรองผ่านแบบปรับตัวได้อันดับ 8 นำมาออกแบบด้วยภาษาวีเอชดีแอล โดยใช้วิธีการคูณแบบ Baugh-Wooley



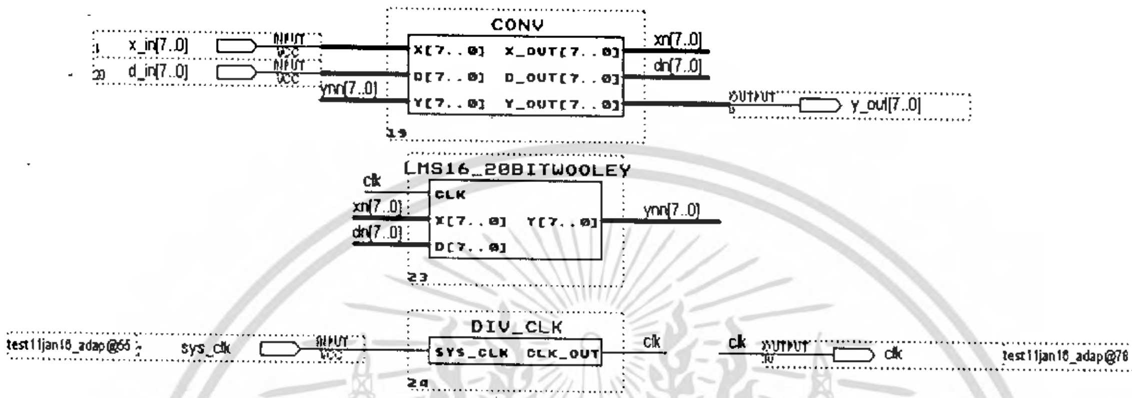
รูปที่ 4.67 แสดงสัญลักษณ์ของวงจรกรองความถี่ผ่านแบบปรับตัวได้



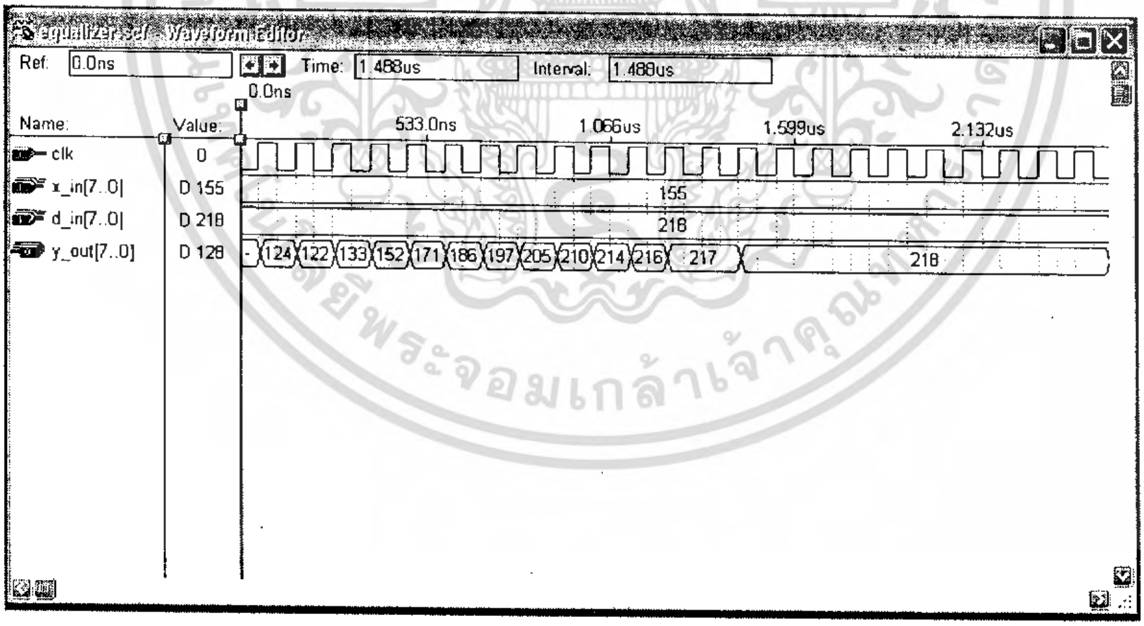
รูปที่ 4.68 แสดงผลการจำลองการทำงานของวงจรกรองความถี่ผ่านแบบปรับตัวได้

4.2.18 ส่วนการสร้างวงจรอิควอไลเซอร์

ในส่วนของการสร้างวงจรอิควอไลเซอร์จะใช้หลักการของวงจรกรองความถี่ผ่านแบบปรับตัวได้ ป้อนสัญญาณอินพุทและสัญญาณที่ต้องการจากวงจรกลับบิทเครื่องหมาย และสัญญาณนาฬิกาจากวงจรหารความถี่



รูปที่ 4.69 แสดงส่วนการสร้างวงจรอิควอไลเซอร์



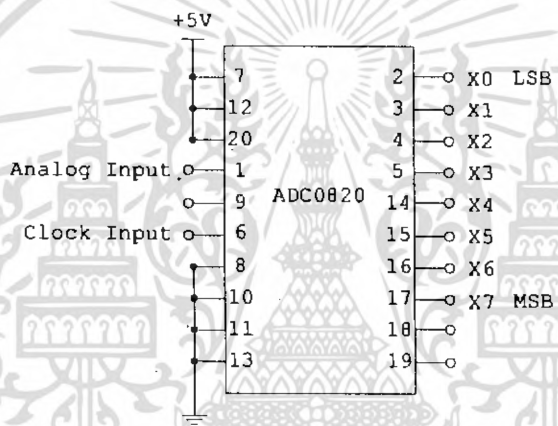
รูปที่ 4.70 แสดงผลการจำลองการทำงานของวงจรอิควอไลเซอร์

4.3 การนำโปรแกรมที่ออกแบบโดยภาษาวีเอชดีแอลไปสร้างเป็นวงจรอิควอไลเซอร์

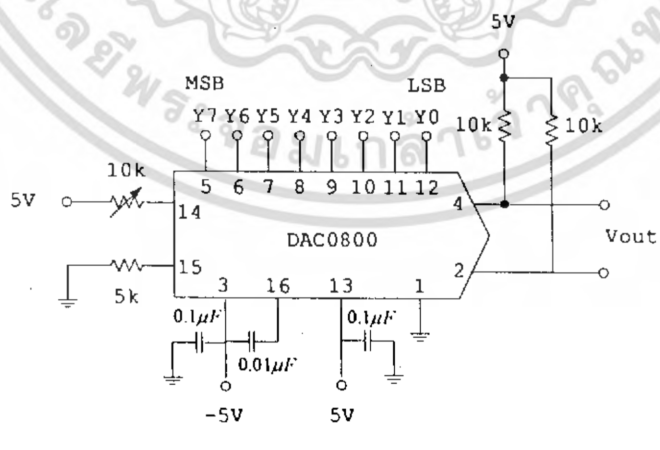
ในการทดลองนี้จะเป็นการทดลอง เพื่อทดสอบวงจรอิควอไลเซอร์ที่ออกแบบขึ้นว่าเป็นไปตามที่ออกแบบไว้หรือไม่

การทดลองนี้จะมีอุปกรณ์ที่ใช้อยู่ 3 ส่วนดังนี้

1. วงจรแปลงสัญญาณเชิงอนาล็อก (Analog) เป็นสัญญาณดิจิตอล (Digital) โดยจะใช้อุปกรณ์ที่เรียกว่า Analog To Digital Converter (ADC)
2. วงจรแปลงสัญญาณดิจิตอล (Digital) เป็นสัญญาณเชิงอนาล็อก (Analog) โดยจะใช้อุปกรณ์ที่เรียกว่า Digital To Analog Converter (DAC)
3. อุปกรณ์ FPGA ของบริษัท ALTERA ตระกูล ACEX1K โดยในส่วนของช่องสัญญาณใช้เบอร์ EP1K30TC144-3 และส่วนของอิควอไลเซอร์ใช้เบอร์ EP1K50TC144-3



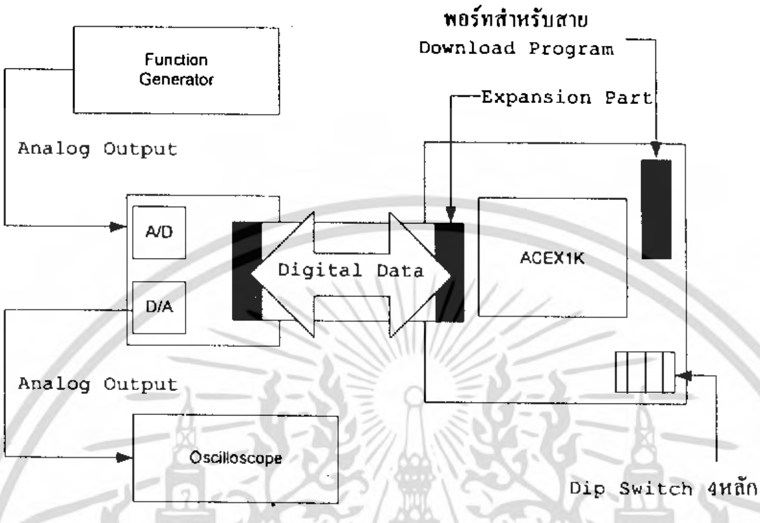
รูปที่ 4.71 วงจรแปลงสัญญาณเชิงอนาล็อกเป็นสัญญาณดิจิตอล



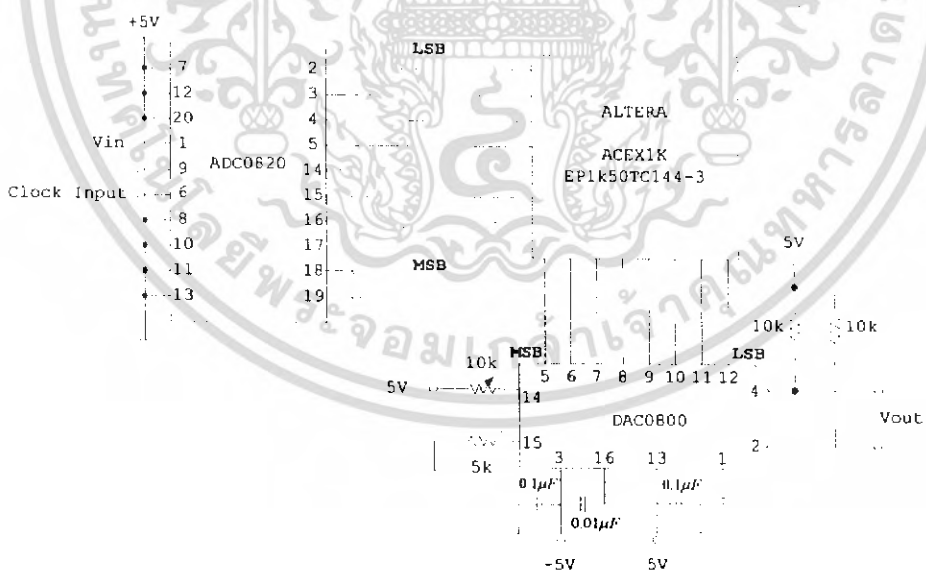
รูปที่ 4.72 วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณเชิงอนาล็อก

4.3.1 การออกแบบการทดลองสำหรับทดสอบวงจรอิควอไลเซอร์ที่ออกแบบ

ออกแบบวงจรรวมโดยใช้อุปกรณ์ทั้ง 3 ส่วนข้างต้นมาต่อรวมกันดังรูปที่ 4.71 เพื่อนำไปทดสอบวงจรอิควอไลเซอร์ที่ออกแบบไว้



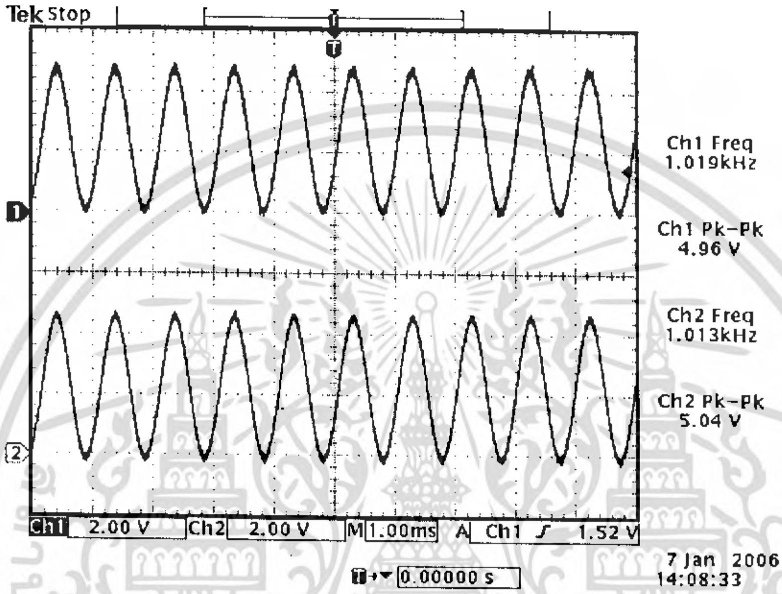
รูปที่ 4.73 การเตรียมอุปกรณ์สำหรับการทดลองเพื่อทดสอบวงจรอิควอไลเซอร์ที่ออกแบบขึ้น



รูปที่ 4.74 วงจรรวมสำหรับการทดลองเพื่อทดสอบวงจรอิควอไลเซอร์ที่ออกแบบขึ้น

4.3.2 ผลการทดลองวงจรแปลงสัญญาณเชิงอนาล็อกเป็นสัญญาณดิจิตอลและวงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณเชิงอนาล็อก

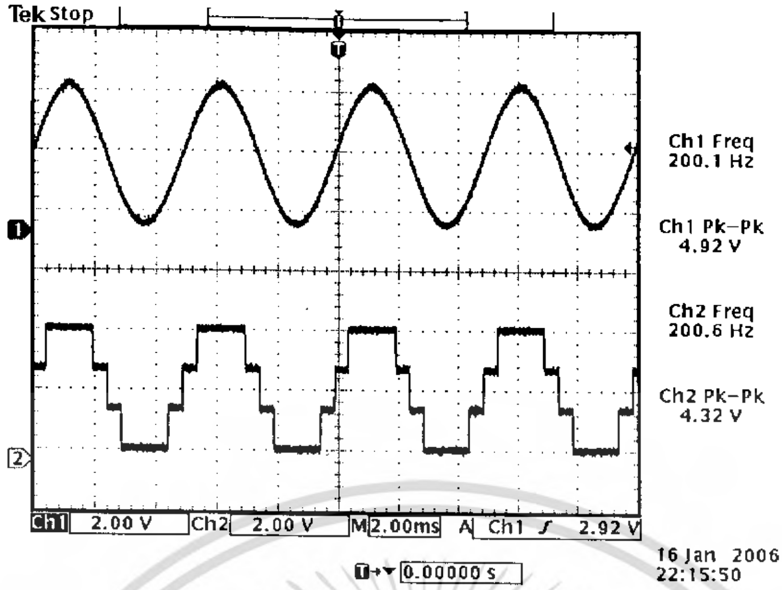
ในการทดลองนี้ ทำการป้อนสัญญาณ Sinusoidal ขนาด 5 V ความถี่ 1 kHz เข้าที่วงจรแปลงสัญญาณเชิงอนาล็อกเป็นสัญญาณดิจิตอลและทำการวัดเอาท์พุทที่วงจรแปลงสัญญาณดิจิตอลเป็นสัญญาณเชิงอนาล็อกเป็นการทดลองเพื่อดูสัญญาณเอาท์พุทว่าสามารถทำการแปลงสัญญาณได้



รูปที่ 4.75 แสดงสัญญาณเอาท์พุทจากการแปลงสัญญาณเชิงอนาล็อกเป็นสัญญาณดิจิตอลและสัญญาณดิจิตอลเป็นสัญญาณเชิงอนาล็อก

4.3.3 ผลการทดลองวงจรสร้างสัญญาณพีเอเอ็ม 4 ระดับ

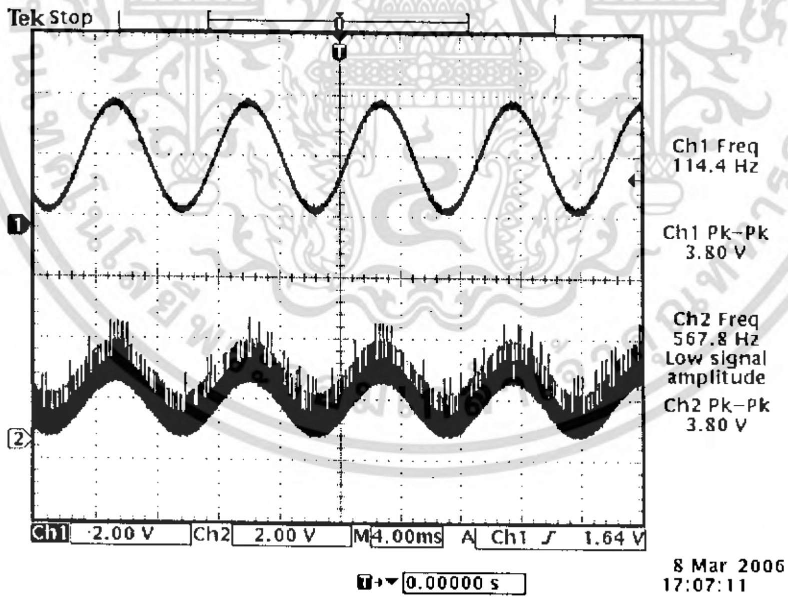
ในการทดลองนี้เราจะนำสัญญาณ Sinusoidal เข้าที่วงจรเก็บค่าและเรียกค่าข้อมูลเพื่อแปลงเป็นสัญญาณพีเอเอ็ม 4 ระดับ



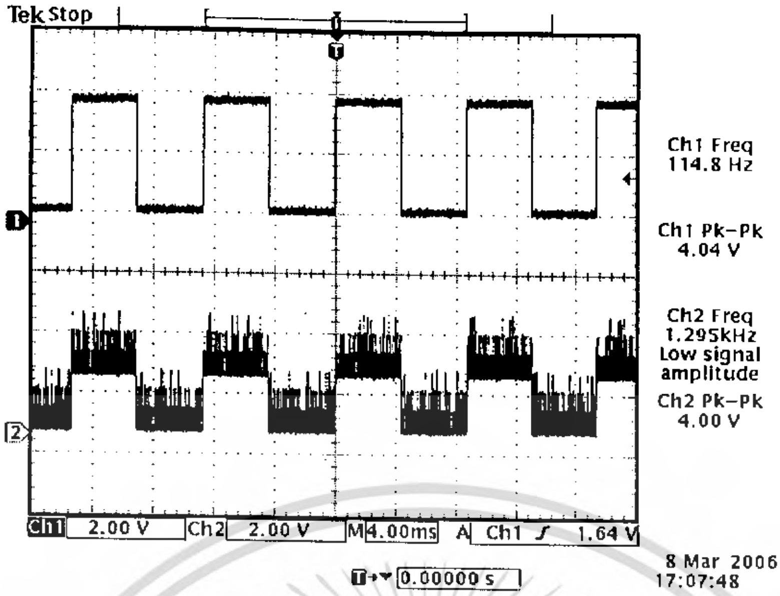
รูปที่ 4.76 แสดงสัญญาณพีเอเอ็ม 4 ระดับจากวงจรเก็บค่าและเรียกค่าข้อมูล

4.3.4 ผลการทดลองแบบจำลองช่องสัญญาณแบบ AWGN

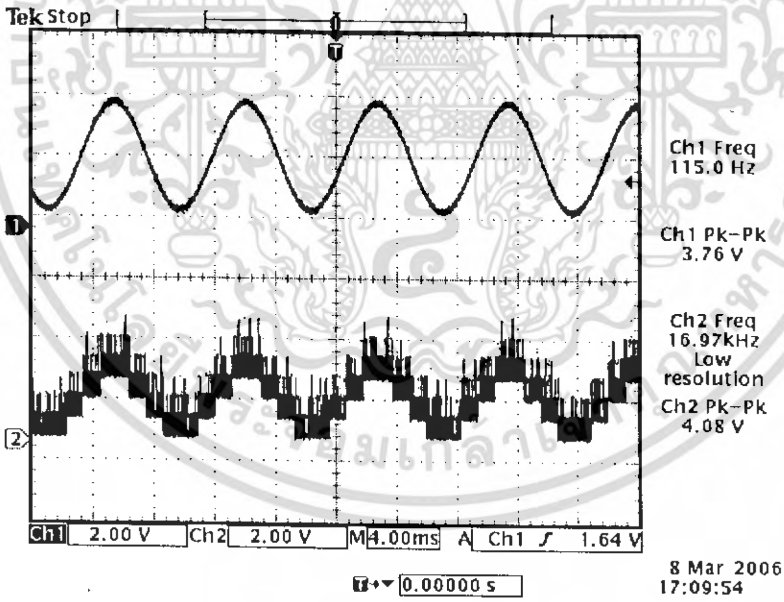
ในการทดลองนี้ จะนำสัญญาณอินพุทเป็นสัญญาณ Sinusoidal สัญญาณรูปสี่เหลี่ยมและสัญญาณพีเอเอ็ม 4 ระดับมาเข้าที่แบบจำลองช่องสัญญาณแบบ AWGN เพื่อทดสอบ



รูปที่ 4.77 แสดงสัญญาณเอาท์พุทจากแบบจำลองช่องสัญญาณแบบ AWGN
เมื่อป้อนสัญญาณ Sinusoidal



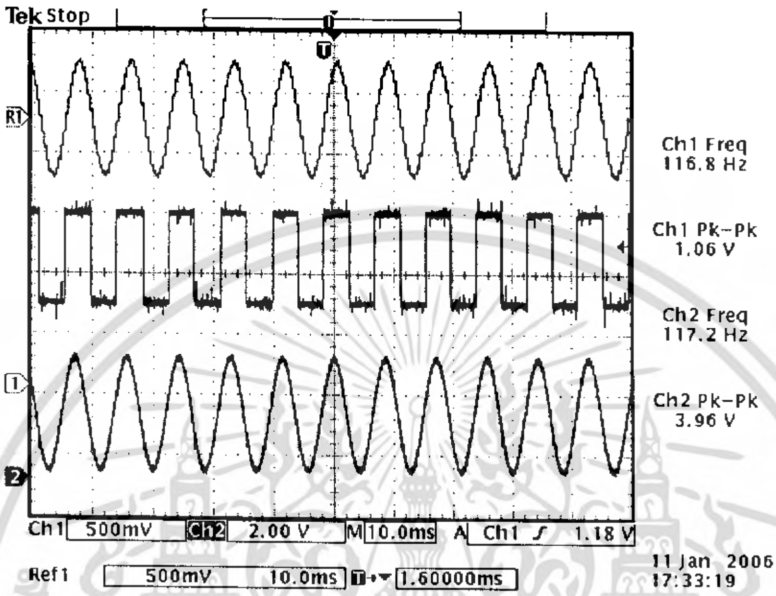
รูปที่ 4.78 แสดงสัญญาณเอาท์พุทจากแบบจำลองช่องสัญญาณแบบ AWGN เมื่อป้อนสัญญาณรูปสี่เหลี่ยม



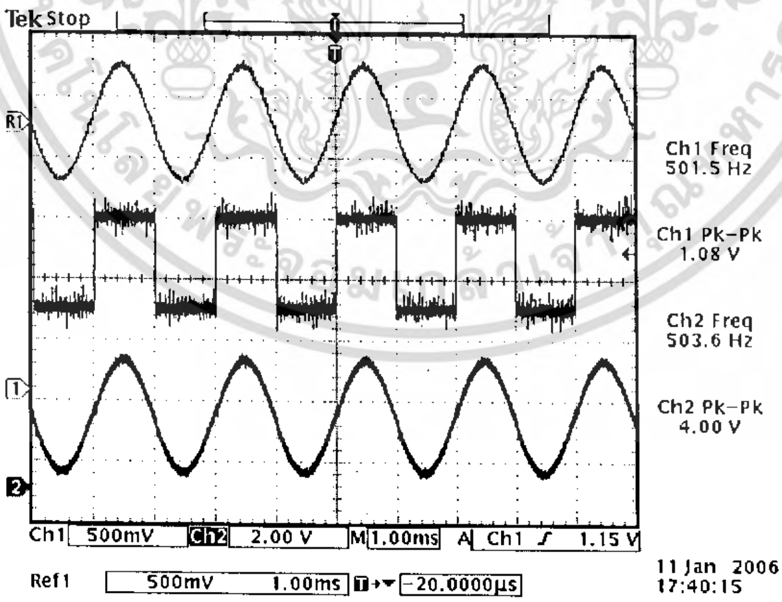
รูปที่ 4.79 แสดงสัญญาณเอาท์พุทจากแบบจำลองช่องสัญญาณแบบ AWGN เมื่อป้อนสัญญาณพีเอเอ็ม 4 ระดับ

4.3.5 ผลการทดลองวงจรกรองความถี่ผ่านแบบปรับตัวได้อันดับที่ 8

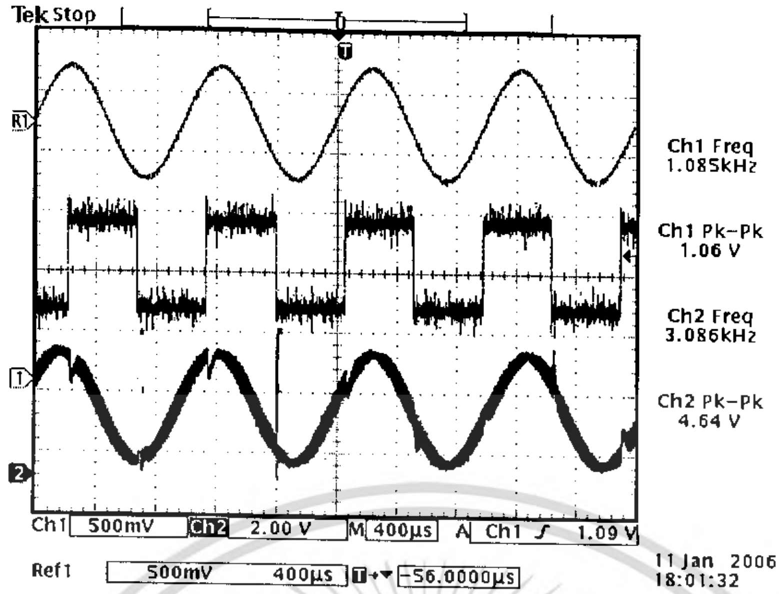
ในการทดลองนี้ จะทำการป้อนสัญญาณอินพุทเป็นสัญญาณรูปสี่เหลี่ยม ขนาด 1 V ความถี่ 100 Hz, 500 Hz, 1 kHz และ 2 kHz และกำหนดให้สัญญาณที่ต้องการเป็นรูปสัญญาณ Sinusoidal ที่มีความถี่เท่ากับสัญญาณอินพุท



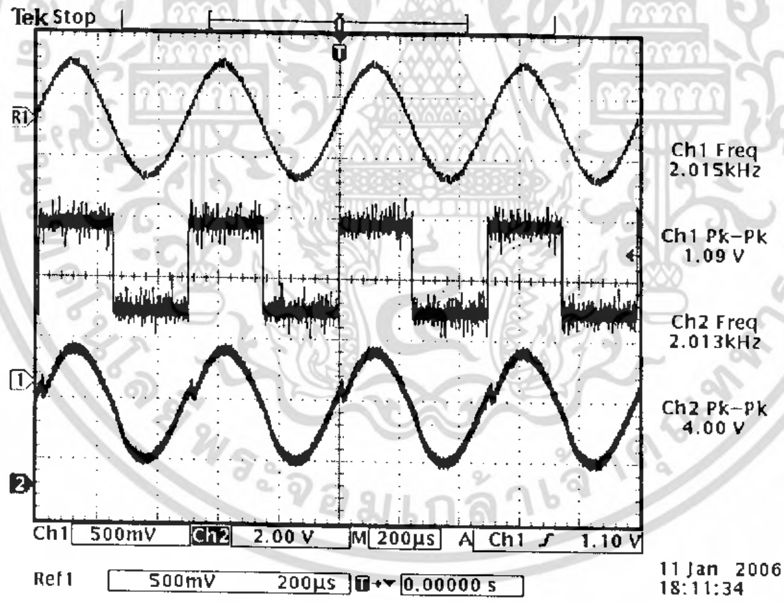
รูปที่ 4.80 สัญญาณเอาท์พุทที่ผ่านตัวกรองความถี่แบบปรับตัวได้อันดับที่ 8 โดยสัญญาณอินพุทเป็นสัญญาณรูปสี่เหลี่ยมความถี่ 100 Hz



รูปที่ 4.81 สัญญาณเอาท์พุทที่ผ่านตัวกรองความถี่แบบปรับตัวได้อันดับที่ 8 โดยสัญญาณอินพุทเป็นสัญญาณรูปสี่เหลี่ยมความถี่ 500 Hz



รูปที่ 4.82 สัญญาณเอาท์พุทที่ผ่านตัวกรองความถี่แบบปรับตัวได้อันดับที่ 8 โดยสัญญาณอินพุทเป็นสัญญาณรูปสี่เหลี่ยมความถี่ 1 kHz



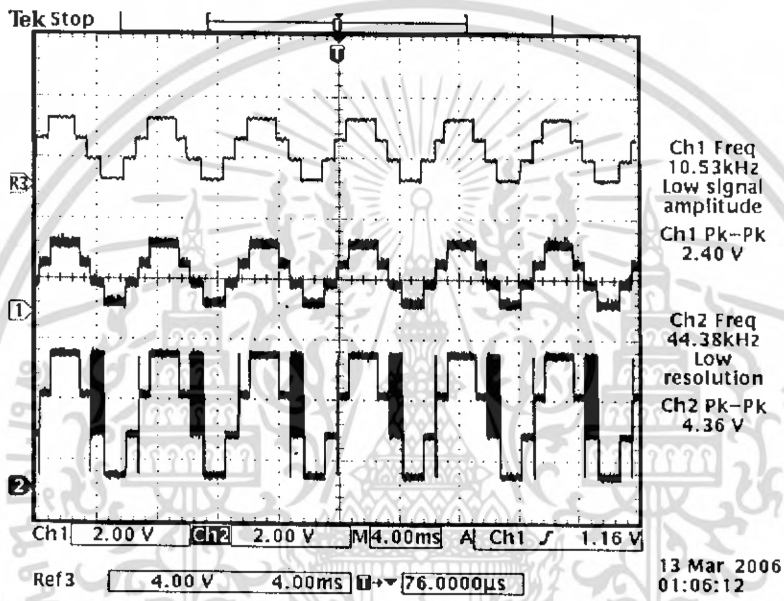
รูปที่ 4.83 สัญญาณเอาท์พุทที่ผ่านตัวกรองความถี่แบบปรับตัวได้อันดับที่ 8 โดยสัญญาณอินพุทเป็นสัญญาณรูปสี่เหลี่ยมความถี่ 2 kHz

4.3.6 ผลการทดลองวงจรอิควอไลเซอร์

ในการทดลองนี้ จะทำการป้อนสัญญาณอินพุทเป็นสัญญาณ Sinusoidal เพื่อแปลงไปเป็นสัญญาณพีเอเอ็ม 4 ระดับ แล้วนำไปเข้าช่องสัญญาณที่มีค่า SNR แตกต่างกัน จากนั้นเข้าสู่วงจรอิควอไลเซอร์ โดยมีช่องสัญญาณ 2 แบบคือ

4.3.6.1 เมื่อนำสัญญาณผ่านช่องสัญญาณแบบที่สุ่มขึ้น

ป้อนสัญญาณพีเอเอ็ม 4 ระดับ ผ่านช่องสัญญาณแบบที่สุ่มขึ้นที่มีค่า SNR แตกต่างกัน แล้วทำการอิควอไลเซอร์ได้ผลการทดลองดังรูป

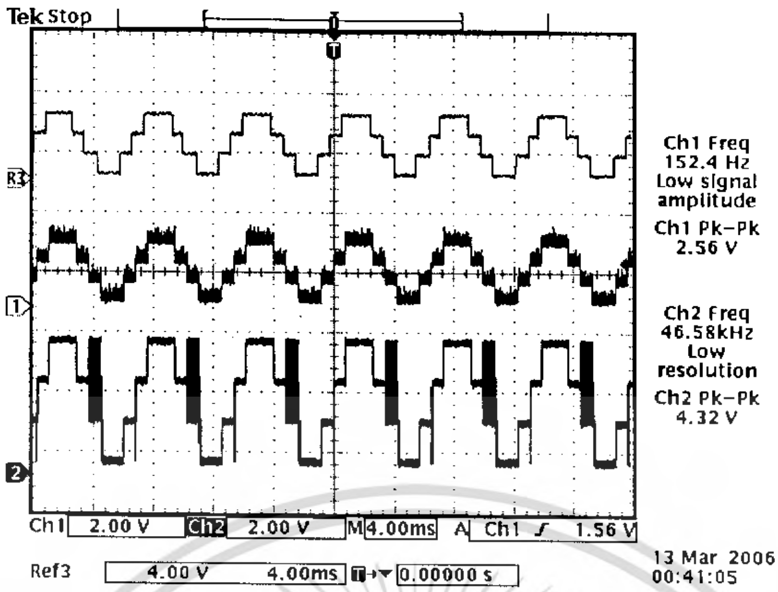


รูปที่ 4.84 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=23

ช่องสัญญาณอ้างอิง แสดงสัญญาณพีเอเอ็ม 4 ระดับก่อนเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 1 แสดงสัญญาณสัญญาณพีเอเอ็ม 4 ระดับหลังเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 2 แสดงสัญญาณพีเอเอ็ม 4 ระดับจากวงจรอิควอไลเซอร์

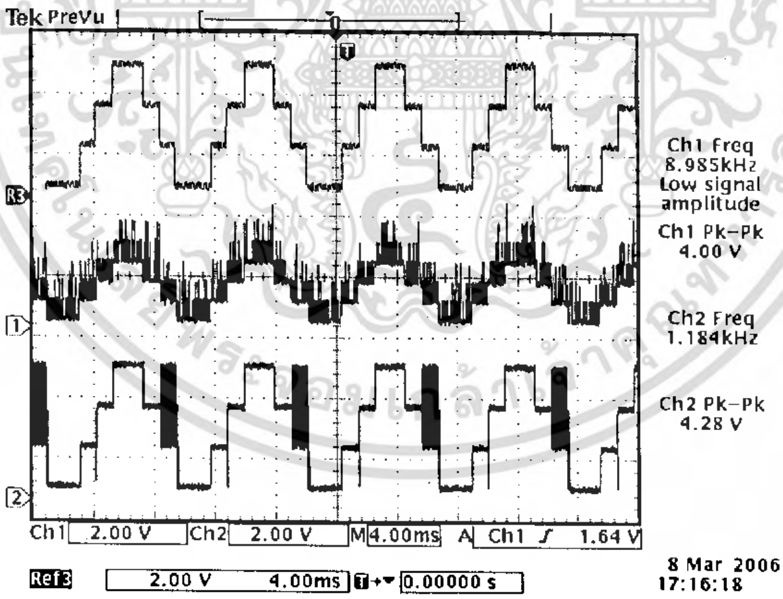


รูปที่ 4.85 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=17

ช่องสัญญาณอ้างอิง แสดงสัญญาณพีเอเอ็ม 4 ระดับก่อนเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 1 แสดงสัญญาณสัญญาณพีเอเอ็ม 4 ระดับหลังเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 2 แสดงสัญญาณพีเอเอ็ม 4 ระดับจากวงจรอิควอไลเซอร์



รูปที่ 4.86 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=11

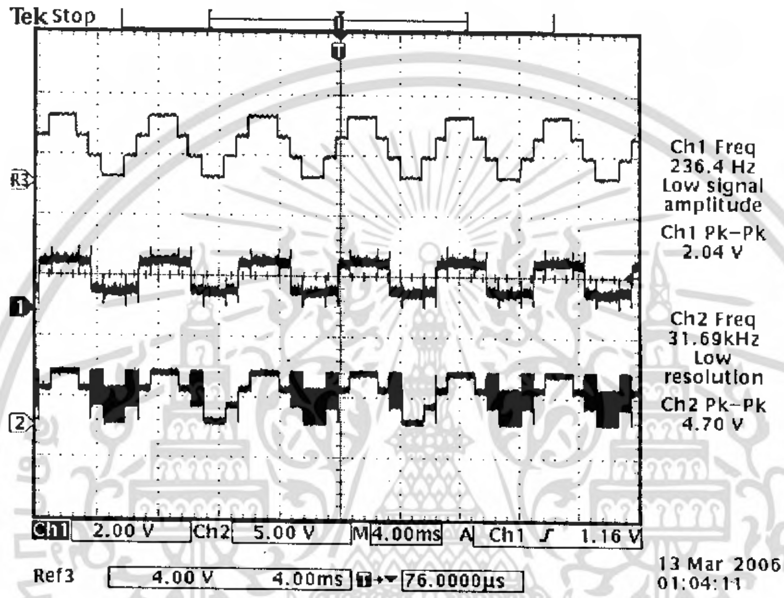
ช่องสัญญาณอ้างอิง แสดงสัญญาณพีเอเอ็ม 4 ระดับก่อนเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 1 แสดงสัญญาณสัญญาณพีเอเอ็ม 4 ระดับหลังเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 2 แสดงสัญญาณพีเอเอ็ม 4 ระดับจากวงจรอิควอไลเซอร์

4.3.6.2 เมื่อนำสัญญาณผ่านช่องสัญญาณแบบโพรอากิส-เอ

ป้อนสัญญาณพีเอเอ็ม 4 ระดับ ผ่านช่องสัญญาณแบบโพรอากิส-เอที่มีค่า SNR แตกต่างกัน แล้วทำการอิควอไลเซอร์ได้ผลการทดลองดังรูป

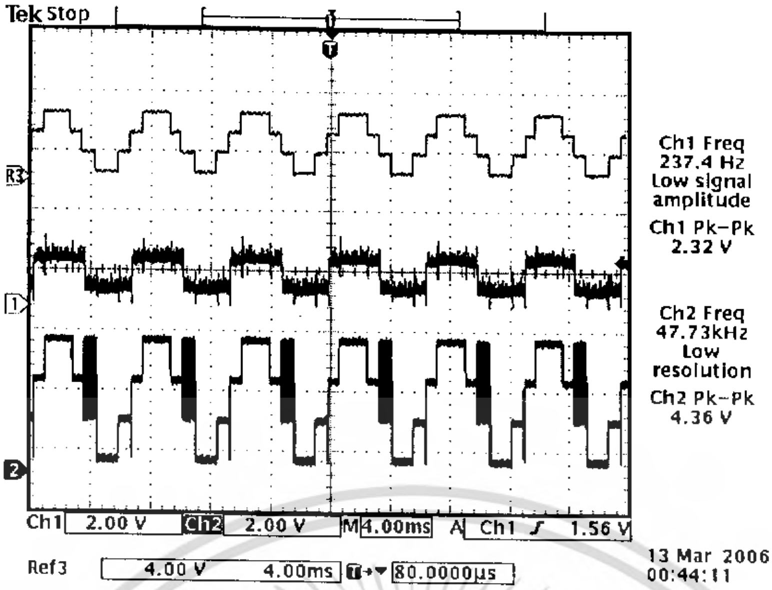


รูปที่ 4.87 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=23

ช่องสัญญาณอ้างอิง แสดงสัญญาณพีเอเอ็ม 4 ระดับก่อนเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 1 แสดงสัญญาณสัญญาณพีเอเอ็ม 4 ระดับหลังเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 2 แสดงสัญญาณพีเอเอ็ม 4 ระดับจากวงจรอิควอไลเซอร์

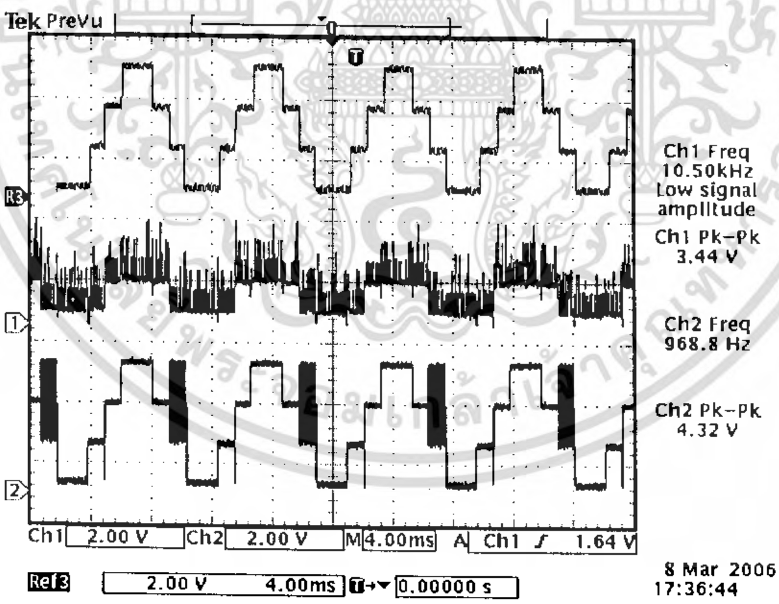


รูปที่ 4.88 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=17

ช่องสัญญาณอ้างอิง แสดงสัญญาณพีเอเอ็ม 4 ระดับก่อนเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 1 แสดงสัญญาณสัญญาณพีเอเอ็ม 4 ระดับหลังเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 2 แสดงสัญญาณพีเอเอ็ม 4 ระดับจากวงจรอิควอไลเซอร์



รูปที่ 4.89 แสดงสัญญาณเอาท์พุทที่ได้จากวงจรอิควอไลเซอร์ในช่องสัญญาณที่มี SNR=11

ช่องสัญญาณอ้างอิง แสดงสัญญาณพีเอเอ็ม 4 ระดับก่อนเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 1 แสดงสัญญาณสัญญาณพีเอเอ็ม 4 ระดับหลังเข้าช่องสัญญาณแบบ AWGN

ช่องสัญญาณที่ 2 แสดงสัญญาณพีเอเอ็ม 4 ระดับจากวงจรอิควอไลเซอร์

บทที่ 5
บทวิจารณ์และบทสรุป

5.1 สรุปผลการทำงานของวงจรอิควอไลเซอร์แบบปรับตัวได้ อันดับที่ 8

จากการออกแบบและทำการจำลองผลการทำงานของวงจรอิควอไลเซอร์แบบปรับตัวได้ โดยการซิมูเลชั่นด้วยโปรแกรมแมทแลบ ได้ทดลองการทำงานการปรับค่าสัมประสิทธิ์โดยจะใช้อัลกอริทึมแบบค่าเฉลี่ยกำลังสองน้อยสุดซึ่งเป็นระบบที่สามารถออกแบบการทำงานได้อย่างไม่ยุ่งยาก และพบว่าสามารถจะทำงานได้อย่างเหมาะสม โดยเมื่อทำการเปลี่ยนสัญญาณอินพุทในรูปแบบต่างๆกัน 3 รูปแบบ และในส่วนของช่องสัญญาณได้ทำการทดลองใน 2 ลักษณะ พบว่าผลลัพธ์ที่ได้จากวงจรอิควอไลเซอร์แบบปรับตัวได้ สามารถทำงานได้กับอินพุททุกๆรูปแบบ

ในส่วนของการสร้างเป็นฮาร์ดแวร์ เราจะแยกเป็น 3 ส่วน โดยในส่วนแรกคือส่วนวงจรสร้างสัญญาณอินพุท โดยในการทดลองเราใช้เป็นสัญญาณ PAM 4 ระดับ ในส่วนที่ 2 คือส่วนของช่องสัญญาณ AWGN ซึ่งมีส่วนของการสร้างสัญญาณรบกวน เพื่อเป็นตัวกำหนดค่า SNR ในระดับที่แตกต่างกัน และส่วนของช่องสัญญาณ ซึ่งเราได้ทำการทดลองใน 2 ลักษณะคือ แบบที่สุ่มขึ้นมา 5 tap และ แบบโพรอากิส-เอ จะพบว่าเมื่อนำสัญญาณอินพุทจากส่วนแรกเข้ามาในช่องสัญญาณแบบโพรอากิส-เอจะได้เอาท์พุทมีรูปร่างสัญญาณที่ผิดเพี้ยนไปมากกว่าเอาท์พุทที่ได้จากช่องสัญญาณแบบที่สุ่ม และในส่วนที่ 3 คือส่วนของอิควอไลเซอร์แบบปรับตัวได้ ซึ่งเป็นการประยุกต์การใช้ตัวกรองแบบปรับค่าได้ โดยได้ใช้การปรับค่าสัมประสิทธิ์แบบค่าเฉลี่ยกำลังสองน้อยสุด และใช้การคูณที่มีรูปแบบอัลกอริทึมแบบบ็อก-วูเลย์ ซึ่งเป็นอัลกอริทึมที่สามารถออกแบบทำงานได้ง่ายกว่าวงจรคูณในรูปแบบอื่นๆ คือ ในการคูณจะเป็นการคูณแบบทูคอมพลีเมนท์ที่มีการเลื่อนบิทได้โดยที่ต้องสนใจบิทเครื่องหมาย และรูปแบบของโครงสร้างที่เราเลือกใช้งานนั้นเป็นโครงสร้างแบบทรานสโพสแบบตรง ซึ่งมีข้อดีคือจะมีการทำงานที่รวดเร็วกว่าโครงสร้างรูปแบบอื่นๆ โดยในการทดลองได้เปลี่ยนค่าความถี่ในระดับต่างๆ จะพบว่าตัวกรองแบบปรับตัวได้สามารถทำงานได้ดีในช่วงตามที่ทำการทดลอง และในการทดลองวงจรอิควอไลเซอร์ เมื่อทำการทดลองที่ค่า SNR ในระดับที่แตกต่างกัน พบว่าที่วงจรอิควอไลเซอร์สามารถทำงานได้ดี ตามช่วงในการทดลอง

5.2 ปัญหาที่พบระหว่างการทำโครงงาน

การสร้างวงจรอิควอไลเซอร์แบบปรับตัวได้ ที่อันดับสูงขึ้นไป จะมีข้อจำกัดในเรื่องของจำนวนลอจิคเกทของอุปกรณ์บอร์ดเอฟพีจีเอ เนื่องจากจำนวนลอจิคเกทของอุปกรณ์บอร์ดเอฟพีจีเอ ของบริษัท ALTERA ตระกูล ACEX1K เบอร์ EP1K50TC144-3 ที่ใช้สำหรับโครงงานนี้นั้น มีจำนวนลอจิคเกทเพียง 50,000 ดังนั้นเมื่อต้องการออกแบบวงจรอิควอไลเซอร์แบบปรับตัวได้ ที่อันดับสูงขึ้นไป ก็จะต้องใช้อุปกรณ์บอร์ดเอฟพีจีเอ ที่มีจำนวนลอจิคเกทมากๆ ซึ่งจะทำให้ราคาของอุปกรณ์เอฟพีจีเอก็จะสูงตามไปด้วย ดังนั้นการจำลองกระบวนการทำงานนั้น จึงจะต้องแยกการทำงานออกเป็น 3 ส่วนตามที่กล่าวข้างต้น

ลงแยกในอุปกรณ์บอร์ดเอฟพีจีเอแต่ละส่วน และเนื่องจากวงจรอิควอไลเซอร์แบบปรับตัวได้ที่ออกแบบนี้เป็นแบบเชิงเส้น จึงสามารถใช้งานได้ดีเฉพาะกับช่องสัญญาณที่เป็นแบบเชิงเส้นเท่านั้น

5.3 การประยุกต์ใช้งาน

โดยภาพรวมในการออกแบบวงจรอิควอไลเซอร์นั้นได้มีการประยุกต์การใช้งานมาจากตัวกรองปรับตัวได้ ซึ่งจะทำการปรับค่าสัมประสิทธิ์ โดยจะใช้อัลกอริทึมแบบค่าเฉลี่ยกำลังสองน้อยสุด และวิธีที่ใช้ในการทดลองนี้จะใช้วิธีแบบเทรนซึ่งจะต้องทราบถึงสัญญาณที่ต้องการก่อน แล้วจึงจะสามารถทำการเปรียบเทียบสัญญาณเอาท์พุทกับผลตอบสนองของสัญญาณที่ต้องการได้ เพื่อนำไปปรับปรุงสมรรถนะของระบบต่อไปได้ และในการออกแบบช่องสัญญาณนั้นจะต้องให้ช่องสัญญาณอยู่ในลักษณะที่เป็นเชิงเส้นจึงจะสามารถทำให้การทำงานของวงจรอิควอไลเซอร์เป็นไปในลักษณะที่ดีได้ หากเมื่อสัญญาณอยู่ในช่องสัญญาณที่มีลักษณะที่ไม่เป็นเส้นแล้ว จะสามารถทำการปรับปรุงได้โดยการใช้วงจรอิควอไลเซอร์แบบที่ไม่เป็นเชิงเส้น คือจะมีส่วนของวงจรป้อนกลับด้วย

หนังสืออ้างอิง

[1] ชำนาญ ปัญญาใส, วัชรากร หนูทอง, "ภาษา VHDL สำหรับการออกแบบวงจรดิจิตอล", ซีเอ็ดยูเคชั่น: กรุงเทพฯ, 2547

[2] พรชัย ภววงษ์ศักดิ์, "การประมวลผลสัญญาณดิจิตอลเบื้องต้น", มหาวิทยาลัยเทคโนโลยีมหานคร http://www.ce.mut.ac.th/home/pornchai, 2543

[3] มนัส สังวรศิลป์, วรรัตน์ ภัทรอมรกุล, "คู่มือการใช้งาน MATLAB ฉบับสมบูรณ์". กรุงเทพฯ : สำนักพิมพ์อินโฟเพรส, 2543.

[4] วิวัฒน์ กิรานนท์, "วิศวกรรมการสื่อสาร", พิมพ์ครั้งที่ 3. กรุงเทพฯ : อักษรสยามการพิมพ์, 2544.

[5] C.R. Baugh and B.A. Wooley, "A two's complement parallel array multiplication algorithm," IEEE Trans. Comput., vol. C-22, pp. 1045-1047, Dec. 1973.

[6] H. Choo, K. Muhammad and K. Roy, "Two's complement computation sharing multiplier and its applications to high performance DFE," IEEE Trans. Signal Process., vol. 51, No. 2, pp. 458-469, Feb. 2003

[7] John G. Proakis, "Digital Communication". 4th edition, Mc Graw Hill Inc., 2001

[8] J. Park, H. Choo, K. Muhammad, S Choi, Y. Im and K. Roy, "Non-adaptive and adaptive filter implementation based on sharing multiplication," IEEE Acoustics, Speech and Signal Process., 2000. ICASSP'00., Vol.1, pp. 460-463, June 2000

[9] Reto Zimmermann, "Computer Arithmetic: Principles, Architectures, and VLSI Design", Integrated Systems Laboratory ETH Zurich, 1998.

[10] Y. Fan and Z. Zilic, "A novel scheme of implementing high speed AWGN communication channel emulators in FPGAS," IEEE Circuits and Systems, 2004. ISCAS'04., Vol.2, pp. II 877-II 880, May 2004

ภาคผนวก

ภาคผนวก ก.
ภาษา VHDL และส่วนประกอบต่างๆของภาษา

ปัจจุบันในการออกแบบระบบดิจิตอลที่เรารู้จักกันจะเป็นการออกแบบโดยใช้การวาดวงจร (Capture Schematic) โดยใช้โปรแกรมช่วยในการวาด (Schematic entry tools) ซึ่งผู้ออกแบบจะต้องมีทักษะสูงในการออกแบบ และต้องใช้เวลามากในการออกแบบระบบจำลองการทำงาน (Simulation) ตลอดจนถึงการแก้ไขความถูกต้องของระบบ (Debugging) ซึ่งในการออกแบบจะต้องอ้างอิงเทคโนโลยีที่ใช้การออกแบบระบบดิจิตอล (Technology Dependent) ถ้าต้องการเปลี่ยนเทคโนโลยีของระบบที่ออกแบบค่อนข้างยากและใช้เวลามาก และเมื่อต้องการออกแบบระบบดิจิตอลที่มีความซับซ้อนสูง ยิ่งทำได้ยากหรือทำไม่ได้ โดยใช้การออกแบบเก่าๆ แต่ในการออกแบบระบบดิจิตอลในปัจจุบันได้มีกระบวนการออกแบบระบบรูปแบบใหม่ที่มีประสิทธิภาพสูง รวดเร็วและไม่ยึดติดกับเทคโนโลยีที่ใช้ออกแบบ (Technology Independent) กระบวนการดังกล่าว คือ Top-down design ซึ่งกระบวนการดังกล่าวจะใช้ภาษาบรรยายฮาร์ดแวร์ HDL (Hardware Description Language) ในการออกแบบจำลองการทำงานสังเคราะห์วงจร (Synthesis) ในรูปแบบของเทคโนโลยีที่เราต้องการ และสามารถทดสอบวงจรที่ออกแบบได้บนฮาร์ดแวร์จำพวกชิป FPGA (Field Programmable Gate Arrays) or ASICs (Application Specific Integrated Circuits)
ดังนั้นการออกแบบสามารถทำได้ง่ายและมีความสะดวกมากยิ่งขึ้น

ก.1 VHDL

VHDL ย่อมาจาก VHSIC Hardware Description Language เป็นภาษาบรรยายฮาร์ดแวร์ประเภทหนึ่ง โดยภาษาเกิดขึ้นจากโครงการที่มีชื่อ VHSIC (Very High Speed Integrated Circuits) ที่ถูกพัฒนาขึ้นโดยกระทรวงกลาโหมของสหรัฐอเมริกา โดยเป้าหมายของโครงการนี้ก็เพื่อพัฒนาขีดความสามารถในการออกแบบวงจรรวมให้สูงขึ้นและสามารถทำได้ง่ายมากยิ่งขึ้น โดยมีเป้าหมายหลัก 2 ประการคือ

- ต้องการภาษาที่สามารถรองรับการออกแบบวงจรที่มีความซับซ้อนได้
- ต้องการภาษาที่เป็นมาตรฐานหรือเป็นภาษากลางที่ทำให้สามารถเผยแพร่ผลงานการออกแบบกันในกลุ่มนักออกแบบด้วยกันได้

ก.2 ประโยชน์ของภาษา VHDL

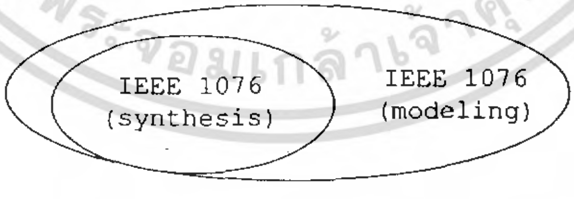
- เป็นภาษามาตรฐานสากลโดยรับรองจากสถาบัน IEEE ทำให้มีโปรแกรมและเครื่องมือต่างๆ และบริษัทที่สนับสนุนการทำงานมากมาย นอกจากนี้วงจรที่ออกแบบโดยภาษา VHDL ก็จะใช้งานได้นานเนื่องจากมีความเข้ากันได้ของภาษากับวงจรที่ได้รับการออกแบบใหม่
- ได้รับการสนับสนุนจากรัฐบาล เนื่องจากภาษา VHDL ได้รับการพัฒนาโดยกระทรวงกลาโหมของสหรัฐอเมริกา
- เป็นภาษาที่ใช้งานจริงในอุตสาหกรรม เนื่องจากภาษา VHDL เป็นภาษาที่เป็นมาตรฐานจากสถาบัน IEEE จึงมีอุตสาหกรรมจำนวนมากที่นำภาษานี้ไปใช้ออกแบบ

- เป็นภาษาที่สามารถใช้ได้หลายระบบ การออกแบบโดยใช้ภาษา VHDL สามารถนำไปจำลองการทำงานหรือสังเคราะห์โดยซอฟต์แวร์ตัวใดก็ได้ที่รองรับภาษา VHDL จึงเป็นการออกแบบที่ไม่ยึดติดกับซอฟต์แวร์ที่ใช้ในการออกแบบ
- เป็นภาษาที่สามารถใช้จำลองรูปแบบการทำงานของวงจร ผู้ออกแบบวงจรสามารถออกแบบวงจรโดยใช้ภาษา VHDL ได้หลายระดับ ตั้งแต่ระดับมาโครบล็อก (Macro block) จนถึงระดับเกต (Gate) และสามารถออกแบบวงจรที่มีความซับซ้อนสูงและมีขนาดใหญ่ได้
- เป็นภาษาที่สามารถนำกลับมาใช้ใหม่ได้ วงจรที่ออกแบบโดยภาษา VHDL สารถนำกลับมาใช้ใหม่ได้ง่าย เนื่องจากสามารถเปลี่ยนแปลงแก้ไขวงจรได้ง่าย คล้ายคลึงกับโปรแกรมของซอฟต์แวร์
- เป็นภาษาที่สามารถนำไปใช้เป็นเอกสารประกอบได้ เป็นภาษาในรูปแบบพฤติกรรม ทำให้เราสามารถอธิบายการทำงานของวงจรภายในการออกแบบได้ทันที

ก.3 ลักษณะการใช้งานภาษา VHDL

การใช้งานภาษา VHDL อาจจำแนกออกเป็น 5 ประเภทดังนี้คือ

- Document Language : เป็นภาษาที่ใช้บรรยายรายละเอียดการทำงานของวงจรที่ออกแบบ
- Design Language : เป็นภาษาที่ใช้สำหรับออกแบบวงจรที่มีความซับซ้อนสูง
- Verification Language : เป็นภาษาที่ใช้ตรวจสอบความถูกต้องของวงจรที่ออกแบบ
- Test Language : เป็นภาษาที่ใช้สำหรับการทำงานของวงจรที่ออกแบบ
- Synthesis Language : เป็นภาษาที่ใช้สำหรับสังเคราะห์วงจร (Synthesis) จริงแต่รูปแบบดังกล่าวนี้ไม่ได้ครอบคลุมทุกรูปแบบทั้งหมดของการเขียนในภาษา VHDL แต่มีเพียงบางรูปแบบเท่านั้นที่สามารถเขียนสามารถนำไปสังเคราะห์เป็นวงจรได้ หรือเรียกว่าเป็น รูปแบบการเขียนสำหรับการสังเคราะห์ (VHDL for Synthesis)



รูปที่ ก.1 ขอบเขตของรูปแบบการเขียนสำหรับการสังเคราะห์วงจร

ก.4 ส่วนประกอบของภาษา

ส่วนประกอบที่สำคัญและเป็นพื้นฐานของการเขียนมี 4 หน่วยคือ

- หน่วยการออกแบบเอนทิตี (Entity Design unit)
- หน่วยการออกแบบสถาปัตยกรรม (Architecture Design unit)

- หน่วยการออกแบบแพกเกจ (Package Design unit)
- หน่วยการออกแบบโครงแบบ (Configuration Design unit)

ก.4.1 หน่วยการออกแบบเอนทิตี (Entity Design unit)

หน่วยการออกแบบนี้เป็นส่วนที่ใช้สำหรับติดต่อระหว่างภายนอกกับรูปแบบที่เขียนขึ้นโดยเป็นการกำหนดจุดเชื่อมต่อของรูปแบบ กำหนดทิศทางการไหลของสัญญาณ และประเภทของค่าที่สามารถกำหนดให้กับสัญญาณตามจุดต่างๆของข้อมูลที่ไหลผ่านจุดต่อเหล่านั้น รูปที่ 2 แสดงให้เห็นถึงโครงสร้างของหน่วยการออกแบบเอนทิตี

```
Entity component name
is
  Input and Output
ports
  Physical and other
parameter
End [component_name];
```

รูปที่ ก.2 โครงสร้างโดยทั่วไปของหน่วยการออกแบบเอนทิตี

ส่วนนี้จะขึ้นต้นด้วยคำว่า Entity ... is ระหว่างคำทั้งสองคำเป็นส่วนสำหรับชื่อของรูปแบบที่ต้องการจะเขียน (component name) หลังจากนั้นจะตามด้วยส่วนที่ใช้กำหนดช่องทางเข้าและข้อมูล (input-output) รวมทั้งพารามิเตอร์อื่นๆ และที่สำคัญคือ หน่วยการออกแบบเอนทิตีจะต้องปิดท้ายด้วยคำว่า End และเครื่องหมายอัฒภาคเสมอ (;)

ก.4.2 หน่วยการออกแบบสถาปัตยกรรม (Architecture Design unit)

หน่วยการออกแบบสถาปัตยกรรม คือส่วนที่ใช้เขียนบรรยายพฤติกรรมของรูปแบบในมุมมองของการจำลองการทำงาน พฤติกรรมต่างๆที่บรรยายในส่วนนี้ขึ้นอยู่กับข้อมูลที่ผ่านเข้าและออกตรงช่องทาง ตลอดจนพารามิเตอร์ต่างๆที่กำหน่วยการออกแบบเอนทิตี รูปที่ 3 แสดงให้เห็นถึงโครงสร้างของหน่วยการออกแบบสถาปัตยกรรม

```
Architecture identifier of
company_name is
[declaration]
Begin
Specification of the functionality
Of the component in term of its
input lines and as influenced by
physical and other parameter
End [identifier];
```

รูปที่ ก.3 โครงสร้างโดยทั่วไปของหน่วยการออกแบบสถาปัตยกรรม

ส่วนของหน่วยการออกแบบสถาปัตยกรรมเริ่มต้นด้วยคำว่า Architecture และตามด้วยชื่อ (identifier) สิ่งที่ต้องกำหนดลงไปได้แก่ สิ่งที่แสดงให้เห็นว่า Architecture นั้นใช้บรรยายการออกแบบเอนทิตีใดๆ (of <entity design unit> is) ส่วนที่อยู่ระหว่าง Architecture และ Begin เป็นพื้นที่ส่วนประกาศหน่วยของสถาปัตยกรรมกำหนด (Architecture declaration area) ที่เป็นส่วนเผื่อเลือก (Option) ในบริเวณนี้สามารถใช้เขียนประกาศกำหนดค่าต่างๆที่จะนำไปใช้ภายในสถาปัตยกรรมนั้นได้ ส่วนที่ใช้บรรยายความสัมพันธ์ระหว่างข้อมูลที่ไหลเข้าและไหลออกของรูปแบบ (สัญญาณที่กำหนดในชุดคำสั่ง port) นั้นจะถูกบรรยายในบริเวณของเนื้อที่ระหว่างคำว่า Begin กับ End ของหน่วยการออกแบบสถาปัตยกรรมและนอกจากนั้นชุดคำสั่งทุกคำสั่งที่อยู่ภายในบริเวณนี้จะเป็นชุดคำสั่งแบบแข่งขนาน (Concurrent statement) เท่านั้น คือทุกๆ statementจะทำงานพร้อมกัน ลำดับก่อนหลังจะไม่มีผลต่อการทำงานของรูปแบบ หน่วยการออกแบบสถาปัตยกรรมจะต้องปิดท้ายด้วยคำสั่ง End และชื่อของสถาปัตยกรรมนั้นๆ โดยทั่วไปการเขียนรูปแบบระบบเชิงเลขด้วยภาษา VHDL สามารถเขียนได้ในลักษณะต่างๆดังนี้

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

ถ้าผู้ออกแบบต้องการออกแบบวงจรดิจิตอลใดๆจะต้องออกแบบหน่วยการออกแบบพื้นฐานก็คือส่วนของ Entity และส่วนของ Architecture ที่มีความสัมพันธ์กัน ซึ่งมีด้วยกันหลายรูปแบบสำหรับการออกแบบวงจรเดียวกัน และความเข้าใจของผู้ออกแบบเอง

ก.4.3 หน่วยการออกแบบแพกเกจ

แพกเกจเป็นหน่วยการออกแบบที่ใช้เก็บข้อมูลต่างๆตลอดจนโปรแกรมย่อย (Subprogram) ที่เป็นประโยชน์ในการเขียนรูปแบบบรรยายวงจรดิจิตอล โดยข้อมูลใน Package สามารถถูกเรียกใช้ได้โดย Entity, Architecture หรือด้วย Package อื่นๆด้วยคำสั่ง USE Statement นอกจากนั้นสิ่งที่นิยมทำกันมากคือรูปแบบมาตรฐานต่างๆเช่น Standard components ต่างๆจะถูกเก็บไว้ใน Package ที่ทุกคนเรียกใช้งานได้ โดยสิ่งที่สามารถสร้างไว้ใน Package ได้แก่

- Subprogram
- Types
- Constants
- Signal
- Aliases
- Attributes
- Component
- Disconnection Specification

การเรียกใช้งาน Package จะใช้คำสั่ง USE Package โดยมีรูปแบบดังนี้

```
USE libary-name.package-name.item;
```

รูปที่ ก.4 โครงสร้างโดยทั่วไปของส่วน USE Package

แพกเกจจะแบ่งออกเป็น 2 ส่วน

1. Package Declaration

ส่วนที่มีความสำคัญที่สุดในแพกเกจ (ถ้ามองในแง่ของการนำไปใช้จากภายนอก) ได้แก่ส่วนการประกาศแพกเกจ เพราะจะเป็นส่วนที่กำหนดชื่อของสิ่งที่ประกาศอยู่ภายในแพกเกจสำหรับไปใช้ภายนอกของตัวแพกเกจเอง สิ่งใดๆที่ถูกประกาศไว้ในส่วนของบอดี้ แต่ไม่ได้ถูกประกาศไว้ในส่วนการประกาศแพกเกจจะมาสามารถถูกนำค่าและพฤติกรรมไปใช้ส่วนนอกบอดี้ ซึ่งสามารถเปรียบเทียบได้กับสิ่งที่ประกาศไว้ในส่วนของประกาศเอนทิตี คือจุดเชื่อมต่อ หรือพอร์ท ที่มีหน้าที่ติดต่อกับโลกภายนอก ฉะนั้นโดยทั่วไปแล้วแพกเกจสามารถสร้างขึ้นได้โดยไม่จำเป็นต้องมีส่วนบอดี้ และสามารถถูกนำไปใช้จากรูปแบบภายนอกได้ เช่น ใช้สำหรับประกาศชนิด (Type) หรือสัญญาณ เช่นเดียวกับส่วนบอดี้ แพกเกจที่ไม่จำเป็นต้องมีส่วนประกาศแพกเกจ แต่แพกเกจนั้นจะไม่สามารถนำไปจากรูปแบบภายนอกได้

```
PACKAGE package_name IS
      constant_declarations
      type_declarations
      signal_declarations
      component_declarations
END package_name;
```

รูปที่ ก.5 โครงสร้างโดยทั่วไปของส่วนประกาศแพกเกจ

2. Package Body

โครงสร้างที่ประกอบด้วยคำสั่งต่างๆในรูปของคำสั่งลำดับ (Sequence) ที่ใช้บรรยายฟังก์ชั่นการทำงานของโปรแกรมย่อย (Subprogram) ทั้งหลาย ที่ชื่อของโปรแกรมย่อยนั้นที่ถูกประกาศไปในส่วนของการประกาศแพกเกจแล้วจะถูกเก็บไว้ในส่วนบอดี้แพกเกจ ทั้งร่วมทั้งการประกาศค่าคงที่ต่างๆอันได้แก่ค่าคงที่ที่ถูกประกาศชื่อก่อนในส่วนของประกาศแพกเกจ แต่ถูกกำหนดค่าในส่วนของบอดี้แพกเกจ ฉะนั้นส่วนของบอดี้แพกเกจจึงไม่จำเป็นต้องมีถ้าในส่วนประกาศแพกเกจไม่มีการประกาศชื่อที่เป็นโปรแกรมย่อยหรือค่าคงที่ การเขียนบอดี้แพกเกจนั้นเป็นไปตามกฎเกณฑ์ที่แสดงในรูปที่ 6

```
PACKAGE package_name IS

constant_declarations
      type_declarations
      subprogram_body
END package_name;
```

รูปที่ ก.6 โครงสร้างโดยทั่วไปของบอดี้แพกเกจ

ก.4.4 หน่วยการออกแบบโครงแบบ

ดังที่ทราบกันแล้วว่ารูปแบบหนึ่งของระบบเชิงตัวเลขไม่ว่าจะเป็นอะไร จะมีหน่วยการออกแบบเอนทิตีได้เพียงหนึ่งเดียวเท่านั้น แต่หน่วยการออกแบบเอนทิตีหนึ่งหน่วยนี้อาจจะมีสถาปัตยกรรมที่เป็นหน่วยรองได้หลายหน่วย ดังนั้นจะต้องมีหน่วยการออกแบบโครงแบบมาเพื่อกำหนดการใช้โครงแบบ (Configuration) ประกอบเอนทิตีกับหน่วยการออกแบบสถาปัตยกรรมหน่วยไหนเข้าด้วยกัน

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

รูปที่ ก.7 โครงสร้างโดยทั่วไปของหน่วยการออกแบบโครงแบบ

ก.5 การเขียนแบบลำดับชั้น (Hierarchical Model)

ในการออกแบบวงจรดิจิตอลบางครั้งจำเป็นต้องมีการแบ่งวงจรเป็นบล็อกย่อยๆตามหน้าที่การทำงาน เนื่องจากระบบที่ต้องการออกแบบมีความซับซ้อนสูงไม่สามารถที่จะออกแบบได้เพียงบล็อกเดียว ทำให้จำเป็นต้องแบ่งวงจรออกเป็นบล็อกย่อยๆ ซึ่งภาษา VHDL ก็มีความสามารถในการเขียนแบบลำดับชั้น (Hierarchy) ได้ โดยเขียนบรรยายวงจรตั้งแต่ระดับบนสุด (Top model) ที่มีแต่ความสัมพันธ์ของการเชื่อมต่อในแต่ละบล็อกย่อย โดยยังไม่มีรายละเอียดของวงจรในแต่ละบล็อก และในระดับล่างลงไปถึงจะมีรายละเอียดของวงจร

ก.6 ไลบรารีในภาษา VHDL

เป็นส่วนที่เก็บข้อมูลต่างๆของ Design unit ที่ถูกทำการวิเคราะห์ตรวจสอบความถูกต้องตามการทำงานแล้ว (Compiled and simulated) เพื่อให้พร้อมที่จะนำไปใช้ได้กับ Design unit อื่นๆโดยการอ้างถึงชื่อของ Library นั้นๆที่อ้างถึงได้และข้อมูลที่เก็บไว้ใน Library ได้จะแบ่งเป็น

- Primary units
  - Entity declarations
  - Package declarations
  - Configuration specification
- Secondary units
  - Architecture bodies
  - Package bodies

ส่วน Secondary units เป็นส่วนที่ต้องอ้างถึง Primary units ดังนั้นจะต้องคอมไพล์ในส่วน Primary units ก่อนเสมอ เช่น คอมไพล์ Package declarations ก่อนคอมไพล์ Package bodies หรือคอมไพล์ Entity ก่อนคอมไพล์ Architecture โดยปกติแล้วจะมี Library อยู่ 2 ประเภทคือ

- Working Library หมายถึงไดเรกทอรี (Directory) ที่กำลังใช้งานอยู่ปกติจะถูกกำหนดให้เป็น Work เสมอ
- Resource Library หมายถึงไลบรารีเพิ่มเติมโดยสามารถตั้งชื่ออะไรก็ได้ ซึ่งในภาษา VHDL ได้สงวนชื่อของไลบรารีไว้ 2 ชื่อด้วยกัน STD และ WORK โดยใน STD ประกอบไปด้วย 2 package คือ
  - Package STANDARD กำหนดโดย IEEE 1076 โดยที่ภายในจะประกอบด้วยการประกาศ (Declaration) ต่างๆ เช่น VHDL type (integer, real, time, bit และ Boolean เป็นต้น)
  - Package TEXTIO ภายในประกอบด้วยโปรแกรมย่อยต่างๆที่สำหรับแก้ไขดัดแปลงข้อมูลจำพวก ASCII ไฟล์

การเรียกใช้งาน Library มีรูปแบบดังนี้

```
LIBRARY library_name, another_library_name;
```

รูปที่ ก.8 โครงสร้างโดยทั่วไปของไลบรารี

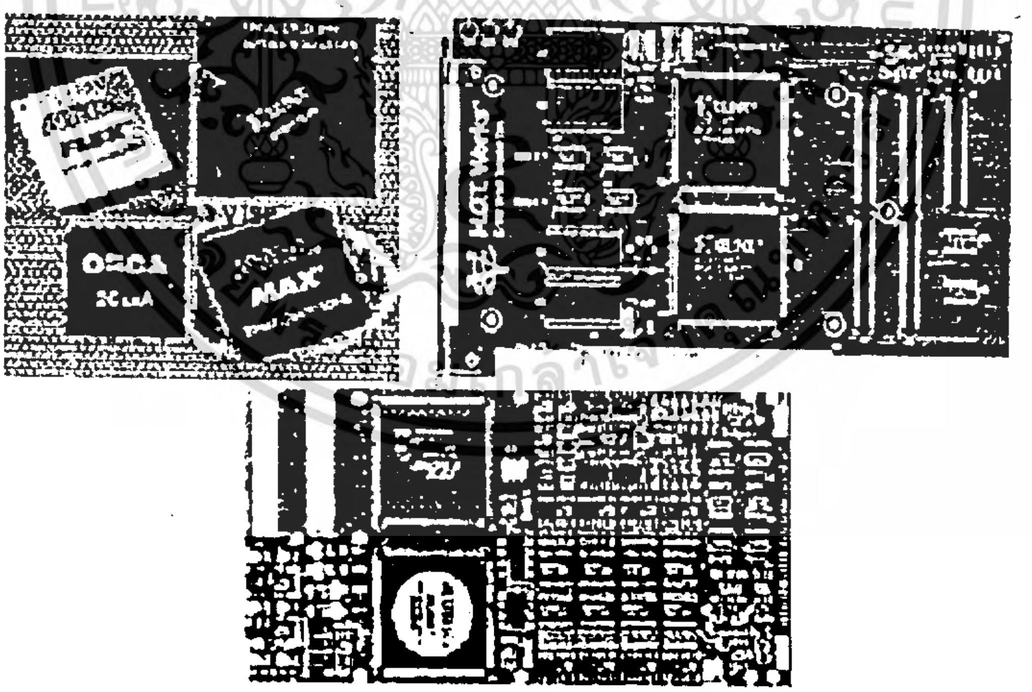
ภาคผนวก ข.

สถาปัตยกรรมและการออกแบบบน FPGA

ข.1 การออกแบบวงจรเชิงเลขด้วยอุปกรณ์ FPGA

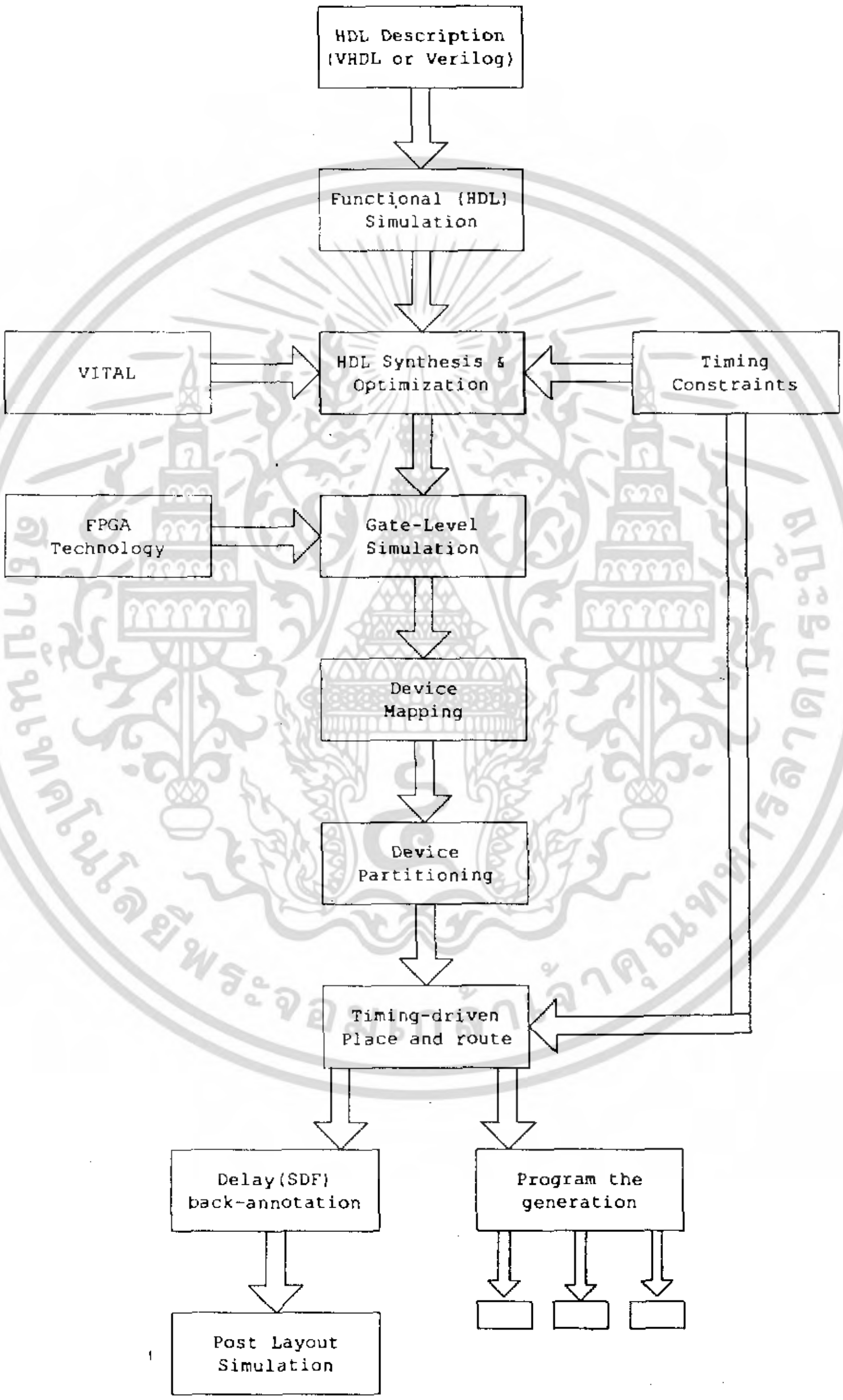
อุปกรณ์ FPGA (Field Programmable Gate Array) เป็นอุปกรณ์ที่ใช้ในการโปรแกรมวงจรที่ได้ออกแบบลงไปเพื่อให้อุปกรณ์ FPGA มีฟังก์ชั่นการทำงานตามที่ออกแบบไว้ ในการทำ FPGA ซึ่งเป็นวิธีการออกแบบ IC (Integrated Circuit) แบบ Semi-custom อีกวิธีหนึ่ง เมื่อเทียบกับการทำ ASICs (Application Specific Integrated Circuits) แล้วนั้นก็มีทั้งข้อดีและข้อเสีย คือการทำ FPGA จะมีข้อจำกัดในด้านขนาดของวงจรเพราะภายในอุปกรณ์ FPGA จะมีจำนวนเกท (gate) ให้ใช้จำนวนจำกัด และการทำ FPGA ก็เหมาะสำหรับการทำผลิตภัณฑ์ต้นแบบหรือเพื่อผลิตในปริมาณต่ำ ส่วนข้อดีของการทำ FPGA ก็คือระยะเวลาที่ใช้ในการทำตั้งแต่เขียนรหัส (code) อธิบายฮาร์ดแวร์จนกระทั่งดาวน์โหลดนั้นน้อยกว่าการทำ ASIC มากและการตรวจสอบหรือแก้ไขการออกแบบก็ทำได้สะดวก

การทำ FPGA ในปัจจุบันมีประสิทธิภาพและความสะดวกมากขึ้น ทั้งนี้ก็เนื่องจากทางบริษัทผู้ผลิต อุปกรณ์ FPGA ได้เพิ่มความสามารถของอุปกรณ์ FPGA โดยเพิ่มจำนวนองค์ประกอบภายในหรือปรับปรุงโครงสร้างสถาปัตยกรรมภายใน และยังได้เพิ่มประสิทธิภาพของซอฟแวร์ที่ใช้ทำ PPR (Partitioning Placement and Routing) สำหรับอุปกรณ์นั้นๆด้วย ลักษณะของตัว FPGA และการนำไปใช้งานแสดง ดังรูปที่ ข.1



รูปที่ ข.1 ลักษณะของตัว FPGA และการนำไปใช้งาน

สำหรับตัวอุปกรณ์ FPGA นั้นก็มีโครงสร้างพื้นฐาน เทคโนโลยีที่ใช้สร้าง ตลอดจนเทคนิควิธีการโปรแกรมที่แตกต่างกันสำหรับผู้ผลิตแต่ละราย นอกจากนั้นอุปกรณ์ FPGA ของแต่ละผู้ผลิตก็มีโครงสร้างและความสามารถที่แตกต่างกันบางส่วน ในการใช้งานนั้นอุปกรณ์ FPGA สามารถนำไปประยุกต์ใช้งานได้ เช่น การประมวลผลสัญญาณเชิงเลข (DSP: Digital Signal Processing) การออกแบบไมโครคอนโทรเลอร์ เป็นต้น โดยมีขั้นตอนในการออกแบบ ดังแสดงในรูปที่ ข.2



รูปที่ ข.2 ขั้นตอนการออกแบบโดยใช้อุปกรณ์ FPGA

ข.1.1 การออกแบบโดยใช้ภาษาอธิบายการทำงานของฮาร์ดแวร์

ในการออกแบบวงจรเชิงเลขนั้นทำได้โดยการวาดวงจร หรือใช้ภาษาฮาร์ดแวร์ ในขั้นตอนนี้เป็นขั้นตอนที่ไม่แตกต่างกันระหว่างการออกแบบด้วย FPGA และ ASIC ในกรณีที่ใช้ภาษาอธิบายฮาร์ดแวร์ แต่ในกรณีที่ออกแบบโดยวิธีการวาดวงจรจะแตกต่างกันโดยที่การทำวิธีนี้จะต้องคำนึงถึงเทคโนโลยีที่จะใช้ซึ่งแต่ละเทคโนโลยีก็มีความแตกต่างกันไปจะเห็นได้ว่าการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์ทำได้สะดวกว่า เพราะการทำด้วยวิธีนี้ไม่ต้องคำนึงถึงเทคโนโลยีที่จะใช้ และที่สำคัญการออกแบบด้วยวิธีนี้สามารถที่จะแก้ไขโมเดล หรือ เปลี่ยนแปลงเทคโนโลยีได้สะดวกกว่าเพราะไม่ต้องวาดวงจรใหม่ นั่นคือการออกแบบโดยใช้ภาษาอธิบายฮาร์ดแวร์จะทำให้โมเดลที่ได้ไม่ขึ้นกับเทคโนโลยี

ในการเขียนโค้ด สิ่งที่ต้องคำนึงถึงคือเขียนอย่างไรจึงจะสามารถสังเคราะห์เป็นวงจรได้และให้คุณสมบัติของวงจรตามที่กำหนด เพราะลักษณะการเขียนโค้ดจะมีผลโดยตรงกับวงจรที่ได้ เนื่องจากในการสังเคราะห์วงจรนั้นซอร์ฟแวร์สังเคราะห์วงจร (Synthesis Tools) จะทำการสังเคราะห์ตามโค้ดที่เขียน ถ้าอธิบายการทำงานของวงจรเดียวกัน แต่เขียนโค้ดในลักษณะที่ต่างกันเมื่อสังเคราะห์แล้วจะได้วงจรที่ต่างกัน และจากวงจรที่ต่างกัน เมื่อนำไปทำต้นแบบด้วย FPGA หรือการทำ ASIC แล้วจะได้ไอซีที่มีคุณสมบัติต่างกันทั้งในด้านของขนาดหรือความเร็ว (area and time) ส่วนการเขียนโค้ดลักษณะใดเพื่อให้ได้ผลลัพธ์ที่ดีที่สุดนั้นก็ขึ้นอยู่กับประสบการณ์ในการออกแบบ

ข.1.2 การจำลองการทำงานของวงจร (Simulation)

ขั้นตอนนี้เป็นขั้นตอนที่สำคัญเพราะเป็นขั้นตอนที่ใช้ตรวจสอบฟังก์ชั่น การทำงานของวงจรว่าถูก ต้องหรือไม่ มีข้อผิดพลาดตรงไหน เพื่อที่จะได้ทำการแก้ไขให้ถูกต้อง ในขั้นตอนนี้จะใช้ซอร์ฟแวร์สำหรับการทำการจำลองการทำงานของวงจร เช่น V-System และ MODELSIM ของบริษัท Model Technology

ข.1.3 การสังเคราะห์วงจร

ในขั้นตอนนี้จะใช้ซอร์ฟแวร์สังเคราะห์วงจร ทำการสังเคราะห์โค้ดเพื่อให้ได้เป็นวงจรขึ้นมา แต่ต้องตรวจสอบด้วยว่าซอร์ฟแวร์นั้นๆสนับสนุนเทคโนโลยี FPGA (FPGA Library) ที่ต้องการใช้หรือไม่ โดย FPGA ที่นิยมใช้งานเช่นของบริษัท XILINX ตระกูล XC4000 และของบริษัท ALTERA ตระกูล FLEX 10 K ซอร์ฟแวร์สังเคราะห์วงจรที่นิยมใช้เช่นโปรแกรม Leonardo Spectrum ของบริษัท Exemplar Logic ซึ่งในขั้นตอนนี้ซอร์ฟแวร์สังเคราะห์วงจรจะแปลงโค้ดและทำการออปติไมซ์ (Optimization) เพื่อให้ได้วงจรตามเทคโนโลยีที่เลือกใช้ นอกจากนี้ยังสามารถกำหนดข้อบังคับสำหรับวงจรได้เช่น ข้อบังคับในเรื่องของเวลา หรือข้อบังคับในเรื่องของพื้นที่ ซึ่งข้อบังคับเหล่านี้จะถูกนำไปใช้ในขั้นตอนการออปติไมซ์เพื่อให้วงจรที่ได้เป็นไปตามที่กำหนด ส่วนสำคัญในการการออปติไมซ์คือการเทียบ (mapping) วงจรให้เข้ากับเทคโนโลยีที่ใช้เพื่อให้ได้วงจรที่เหมาะสมกับโครงสร้างสถาปัตยกรรมภายในอุปกรณ์ FPGA ในกรณีของ XILINX ตระกูล XC4000 และบริษัท ALTERA ตระกูล FLEX 10 K จะเทียบโดยใช้วิธี LUT (Look Up Table) เมื่อทำการสังเคราะห์วงจรเสร็จแล้วซอร์ฟแวร์สังเคราะห์วงจร ก็จะมีการรายงานผลว่าวงจรที่ออกแบบไปนั้นเป็นอย่างไร เช่น มีความหน่วงเท่าไร ใช้ทรัพยากรต่างใน FPGA อะไรบ้างเป็นต้น

ข.1.4 การแบ่งวงจร (Partitioning)

ขั้นตอนนี้เป็นการแบ่งวงจรที่ได้จากการสังเคราะห์ให้เป็นส่วนย่อย สำหรับลงใน CLBs, IOBs หรือองค์ประกอบอื่นๆ ภายในอุปกรณ์ FPGA สำหรับเกณฑ์ที่ใช้ในการแบ่งคือ ให้แต่ละส่วนที่จะแยกออกจากกันมีจำนวนสัญญาณที่เชื่อมต่อระหว่างกันน้อยที่สุดเท่าที่จะทำได้ เพื่อช่วยลดความหนาแน่นในการทำเชื่อมต่อสัญญาณ (routing) ในขั้นตอนนี้จะใช้ซอร์ฟแวร์ทำ โดยซอร์ฟแวร์จะเทียบส่วนประกอบของวงจรเช่น เกท (gate) , ฟลิบฟลอบ (flip flop) ลงในทรัพยากรต่างๆ ที่มีอยู่ภายในอุปกรณ์ FPGA (CLBs, IOBs, BUFT และ edge decoder) หลังจากทำขั้นตอนนี้เสร็จแล้วสามารถที่จะทราบว่าวงจรใช้จำนวนทรัพยากรภายในอุปกรณ์ FPGA ที่ใช้งาน เช่น FPGA ของบริษัท XILINX จะใช้ XILINX Foundation Series 2.1i ซึ่งซอร์ฟแวร์นี้ตัวนี้จะรวมเอาซอร์ฟแวร์ย่อยอื่นๆ อีก เพื่อให้การทำ PPR (Partitioning Placement and Routing) เป็นไปอย่างต่อเนื่อง ส่วน FPGA ของบริษัท ALTERA จะใช้ ALTERA MAX + II

ข.1.5 การวางอุปกรณ์ (Placement)

ขั้นตอนนี้เป็นการเลือกทำเลที่ตั้งของแต่ละส่วนของวงจรที่ผ่านการแบ่งวงจร (Placement) มาแล้วว่าควรจะอยู่ในตำแหน่งใดในอุปกรณ์ FPGA เพื่อให้ได้ผลลัพธ์ที่ดีที่สุด เช่นวงจรส่วนไหนควรอยู่ใกล้กันเพื่อจะได้ค้นหาเส้นทางได้ง่าย หรือช่วยลดความหน่วง จะเห็นได้ว่าตำแหน่งภายในอุปกรณ์ FPGA นั้นมีความสำคัญเพราะถ้าจัดวางวงจรลงในตำแหน่งที่ไม่เหมาะสมแล้วจะทำให้ความหน่วงเพิ่มขึ้นหรือตัว Router ทำการค้นหาเส้นทางสัญญาณได้ไม่หมด

ข.1.6 การเชื่อมต่อสัญญาณ (Routing)

ขั้นตอนนี้เป็นการเชื่อมต่อสัญญาณระหว่างองค์ประกอบต่างๆ ภายในอุปกรณ์ FPGA เช่น ระหว่าง CLBs หรือระหว่าง CLBs กับ IOBs ขั้นตอนนี้จะทำต่อเนื่องจากการวางอุปกรณ์ ในกรณีที่ทำการวางอุปกรณ์ไว้ไม่ดีซอร์ฟแวร์ก็จะทำการเชื่อมต่อสัญญาณได้ไม่หมด หรือเกิดความหน่วงเกินค่าที่กำหนดในข้อบังคับ โดยสามารถทำขั้นตอนนี้ได้โดยใช้ซอร์ฟแวร์ เช่นกัน หรือจะทำการเชื่อมต่อสัญญาณด้วยตัวเอง (manual layout) ก็ได้ แต่ทางที่ดีควรใช้ซอร์ฟแวร์ทำดีกว่าโดยให้ทำการค้นหาเส้นทางหลายๆครั้งเพื่อหาครั้งที่ดีที่สุด นอกจากนั้นการกำหนดข้อบังคับทางเวลา (time constraints) จะช่วยให้ผลที่ได้จากการทำการเชื่อมต่อสัญญาณดีขึ้นได้

ข.1.7 การโปรแกรมอุปกรณ์ FPGA (Configuration)

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

จากที่อธิบายมาทั้งหมดจะเห็นได้ว่าการออกแบบเพื่อทำ FPGA นั้น ทำได้สะดวกกว่าการทำ ASIC มากเพราะใช้เวลาน้อยกว่ามาก ส่วนสำคัญที่ใช้ในการทำ FPGA คือซอร์ฟแวร์ที่ใช้ตั้งแต่การเขียน

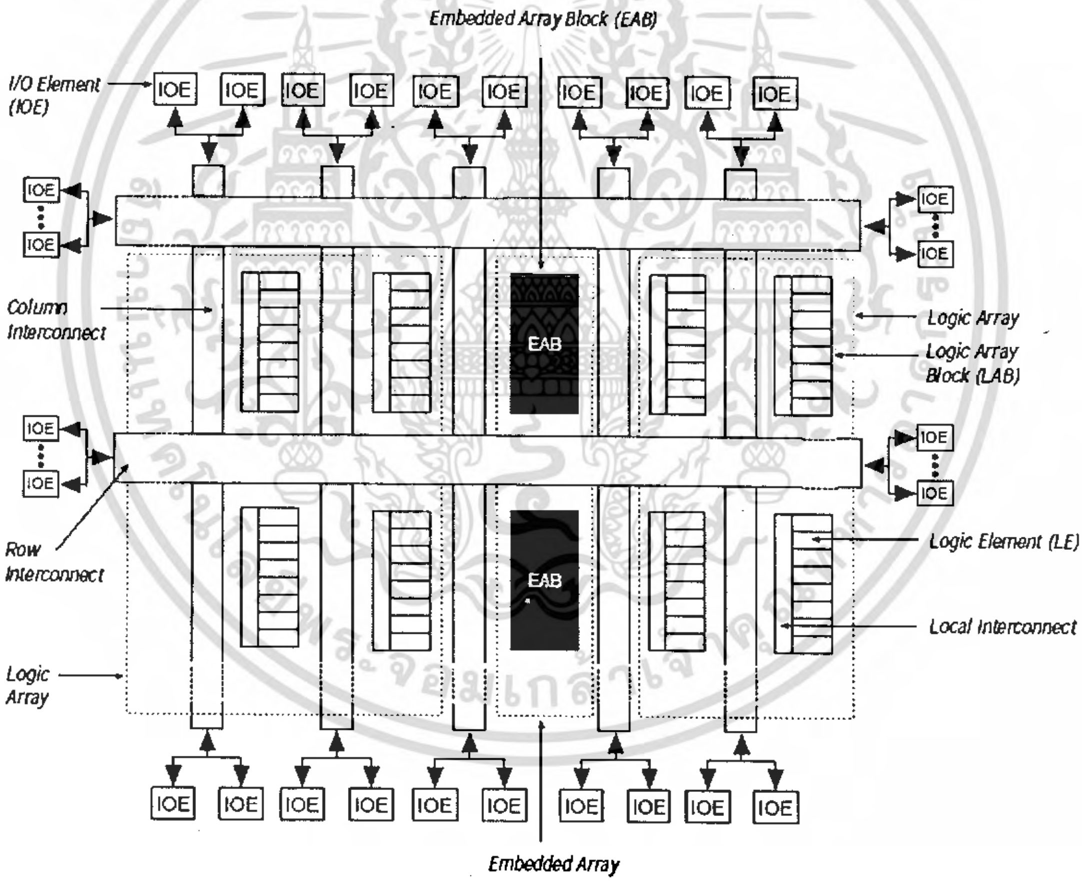
โค้ดอธิบายฮาร์ดแวร์ จนกระทั่งดาวน์โหลดลงในอุปกรณ์ FPGA ซึ่งซอร์ฟแวร์ที่ใช้ต้องเป็นซอร์ฟแวร์ที่ใช้ทำงานต่อเนื่องกัน

ข.2 สถาปัตยกรรมภายในของ FPGA

ในการทดลองนี้ได้ใช้ FPGA เบอร์ EP1k50TC144-3 ในตระกูล Power ACEX 1k-50

ข.2.1 รายละเอียดของ POWER ACEX1K SERIES

FPGA ตระกูล ACEX1K ของบริษัท ALTERA เป็น FPGA ที่มีโครงสร้างภายในเป็นแบบ SRAM-Based FPGA ใช้เทคโนโลยีในการโปรแกรมเหมือนกับหน่วยความจำแบบ SRAM (Static RAM) ทำให้การโปรแกรมสามารถทำซ้ำได้โดยไม่จำกัดจำนวนครั้ง และใช้เวลาในการโปรแกรมชิป FPGA น้อยมาก (ระดับ nsec) การโปรแกรมทำได้ง่ายเช่นเดียวกับการเขียน SRAM ทั่วไป แต่ข้อเสียคือ มันต้องการไฟเลี้ยงในการเก็บวงจรที่ได้ออกแบบไว้



รูปที่ ข.3 โครงสร้างภายในของ FPGA ตระกูล ACEX1K

Feature	EP1K10	EP1K30	EP1K50	EP1K100
Typical gates	10,000	30,000	50,000	100,000
Maximum system gates	56,000	119,000	199,000	257,000
Logic elements (LEs)	576	1,728	2,880	4,992
EABs	3	6	10	12
Total RAM bits	12,288	24,576	40,960	49,152
Maximum user I/O pins	136	171	249	333

ตารางที่ ข.1 คุณลักษณะต่างๆของชิป FPGA ที่ใช้กับบอร์ดในกลุ่ม POWER ACEX1K SERIES

ข.2.2 JTAG CONNECTOR

JTAG Connector เป็น Header ตัวผู้แบบ 5×2 ขา ใช้สำหรับต่อกับสาย Byte Blaster MV เพื่อดาวน์โหลด ข้อมูลทางลอจิกของวงจรที่ได้ออกแบบไว้จากคอมพิวเตอร์ลงสู่ชิปเอฟพีจีเอ

ในการโปรแกรมจะมี Jumper ให้เลือกว่าจะโปรแกรมลงในชิป FPGA หรือโปรแกรมลง Configuration Device

ในกรณีที่ต้องการโปรแกรมลงชิป FPGA ACEX 1K สามารถดูวิธีการ Complier และโปรแกรมลง Configuration Device ด้วยโปรแกรม MAX+PLUS II ให้ดูวิธีการใช้ MAX+PLUS II

ข.2.3 Configuration Device Socket

Socket แบบ PLCC 20 ขา สำหรับใส่ Configuration Device ซึ่งเป็นอุปกรณ์แบบ Flash Memory ใช้สำหรับเก็บข้อมูลวงจรที่เราได้ออกแบบเอาไว้ เมื่อเริ่มต้นจ่ายไฟให้แก่บอร์ด POWER ACEX 1K SERIES วงจรที่อยู่ภายใน Configuration Device จะถูกโหลดลงไปเก็บไว้ในชิป FPGA โดยอัตโนมัติ Configuration Device ที่ใช้ร่วมกับ FPGA แบบ SRAM-BASE FPGA สำหรับเบอร์ของ Configuration Device ที่ใช้กับบอร์ด POWER ACEX 1K SERIES จะเป็นเบอร์ EPC2LI20 หรือ EPC2LC20 มีขนาด 1,695,680 × 1 Bit สามารถใช้กับแรงดัน 5.0V หรือ 3.3V ก็ได้ ในกรณีที่ไม่ต้องการให้ FPGA โหลดวงจรจาก Configuration Device ให้เลือก JUMPER J10 มาที่ตำแหน่ง OFF ลักษณะและตำแหน่งขาของ EPC2LI20 และ EPC2LC20

ข.2.4 พอร์ตขยายช่องสัญญาณแบบ Header 13×2 ขา จำนวน 4 ตัว

บอร์ด POWER ACEX 1K SERIES แต่ละรุ่นจะมีพอร์ตขยายช่องสัญญาณ J6, J7, J8 และ J9 ให้ใช้งานได้อย่างอิสระ 4 พอร์ต แต่ละพอร์ตจะมีขนาด 13×2 ขา ต่อกับขา I/O ของ FPGA โดยตรง

J6 Pin Number	ACEX1K Pin Number	J6 Pin Number	ACEX1K Pin Number
1	7(I/O)	2	8(I/O)
3	9(I/O)	4	10(I/O)
5	11(I/O)	6	12(I/O)
7	13(I/O)	8	14(I/O)
9	17(I/O)	10	18(I/O)
11	19(I/O)	12	20(I/O)
13	21(I/O)	14	22(I/O)
15	23(I/O)	16	26(I/O)
17	27(I/O)	18	28(I/O)
19	29(I/O)	20	30(I/O)
21	31(I/O)	22	32(I/O)
23	33(I/O)	24	36(I/O)
25	37(I/O)	26	GND

ตารางที่ ข.2 ความสัมพันธ์ของ J6 กับตำแหน่ง I/O ของ ACEX1K FPGA

J7 Pin Number	ACEX1K Pin Number	J7 Pin Number	ACEX1K Pin Number
1	38(I/O)	2	39(I/O)
3	41(I/O)	4	42(I/O)
5	43(I/O)	6	44(I/O)
7	46(I/O)	8	47(I/O)
9	48(I/O)	10	49(I/O)
11	51(I/O)	12	54(Ded.Input)
13	56(Ded.Input)	14	59(I/O)
15	60(I/O)	16	62(I/O)
17	63(I/O)	18	64(I/O)
19	65(I/O)	20	67(I/O)
21	68(I/O)	22	69(I/O)
23	77(I/O)	24	72(I/O)
25	73(I/O)	26	GND

ตารางที่ ข.3 ความสัมพันธ์ของ J7 กับตำแหน่ง I/O ของ ACEX1K FPGA

J8 Pin Number	ACEX1K Pin Number	J8 Pin Number	ACEX1K Pin Number
1	78(I/O)	2	79(I/O)
3	80(I/O)	4	81(I/O)
5	82(I/O)	6	83(I/O)
7	86(I/O)	8	87(I/O)
9	88(I/O)	10	89(I/O)
11	90(I/O)	12	91(I/O)
13	92(I/O)	14	95(I/O)
15	96(I/O)	16	97(I/O)
17	98(I/O)	18	99(I/O)
19	100(I/O)	20	101(I/O)
21	102(I/O)	22	109(I/O)
23	110(I/O)	24	111(I/O)
25	112(I/O)	26	GND

ตารางที่ ข.4 ความสัมพันธ์ของ J8 กับตำแหน่ง I/O ของ ACEX1K FPGA

J9 Pin Number	ACEX1K Pin Number	J9 Pin Number	ACEX1K Pin Number
1	113(I/O)	2	114(I/O)
3	116(I/O)	4	117(I/O)
5	118(I/O)	6	119(I/O)
7	120(I/O)	8	121(I/O)
9	122(I/O)	10	124(Ded.Input)
11	126(Ded.Input)	12	128(I/O)
13	130(I/O)	14	131(I/O)
15	132(I/O)	16	133(I/O)
17	135(I/O)	18	136(I/O)
19	137(I/O)	20	138(I/O)
21	140(I/O)	22	141(I/O)
23	142(I/O)	24	143(I/O)
25	144(I/O)	26	GND

ตารางที่ ข.5 ความสัมพันธ์ของ J9 กับตำแหน่ง I/O ของ ACEX1K FPGA

ข.2.5 Module Oscillator

บอร์ด POWER ACEX 1K SERIES มีโมดูลออสซิลเลเตอร์ความถี่ 25.175 MHz (สามารถถอดและเปลี่ยนเป็นค่าความถี่ที่ต้องการได้) เป็นแหล่งกำเนิดความถี่สำหรับใช้ในการออกแบบวงจรที่ต้องการอ้างอิงกับสัญญาณนาฬิกา โดยการป้อนเข้าที่ขา 55 (GCLK1) ของ FPGA

Module Oscillator Pin Number	ACEX1K Pin Number
CLK	55(GCLK1)

ตารางที่ ข.6 ความสัมพันธ์ระหว่างสัญญาณ Oscillator กับตำแหน่งขาของ ACEX1K FPGA

ข.2.6 Terminal แบบ 2 ขั้ว สำหรับ GCLK1 และ GCLK2

Terminal ตัวผู้แบบ 2 ขั้ว ใช้สำหรับต่อกับสัญญาณนาฬิกาจากภายนอกเพื่อป้อนให้แก่ FPGA ทางขา Global Clock 1 (GCLK1) หรือขา Global Clock 2 (GCLK2)

T4 Pin Number	ACEX1K Pin Number	T5 Pin Number	ACEX1K Pin Number
1(+)	125(GCLK2)	1(+)	55(GCLK1)
2(-)	GND	2(-)	GND

ตารางที่ ข.7 ความสัมพันธ์ของ T4, T5 กับตำแหน่งขาของ ACEX1K FPGA

ข.2.7 Terminal แบบ 2 ขั้ว สำหรับแรงดัน 2.5 โวลท์, 3.3 โวลท์ และ 5 โวลท์

Terminal ตัวผู้แบบ 2 ขั้วสำหรับต่อกับแหล่งจ่ายแรงดันภายในบอร์ด POWER ACEX1K SERIES เพื่อเป็นแหล่งจ่ายแรงดันไฟตรงให้กับวงจรภายนอก โดยมีแรงดันให้เลือกใช้งาน 3 ระดับคือ 2.5 โวลท์, 3.3 โวลท์ และ 5.0 โวลท์

T1 Pin Number	Volt	T2 Pin Number	Volt	T3 Pin Number	Volt
1(+)	5 Volt	1(+)	3.3 Volt	1(+)	2.5 Volt
2 (-)	GND	2 (-)	GND	2 (-)	GND

ตารางที่ ข.8 ความสัมพันธ์ของ T1, T2 และ T3 กับแหล่งจ่ายแรงดัน
ภายในบอร์ด POWER ACEX1K SERIES

ข.2.8 คอนเนคเตอร์แหล่งจ่ายไฟ

บอร์ด POWER ACEX1K SERIES ถูกออกแบบมาให้ใช้กับไฟกระแสตรงแรงดัน 9 โวลท์ ถึง 12 โวลท์ โดยแกนในของคอนเนคเตอร์ J1 เป็นขั้วบวกและแกนนอกเป็นขั้วลบ นอกจากนี้ยังมีวงจรกลับขั้วแรงดัน สำหรับในกรณีที่มีการป้อนแรงดันผิดขั้วให้แก่บอร์ดอีกด้วย

ภาคผนวก ค.

โปรแกรมแมทแลบ

อัลกอริทึมค่าเฉลี่ยกำลังสองน้อยสุดเมื่อสัญญาณอินพุทเป็น 2- PAM

```
clc;
clear all;
close all;

dB=17;              % SNR in dB value
N=16;               % filter length = N+1
M=10000;            % total number of data

%%%%%%%channel%%%%%%%
% impulse of the proakis A channel
h=[0.04 -0.05 0.07 -0.21 -0.5 0.72 0.36 0.21 0.03 0.07];
% channel random 5 tap
%h=[-0.01269442675941   0.14578007919020   0.73382869513842
0.14578007919020  -0.01269442675941];
% impulse of the channel
%h=randn(1,5);
L=length(h);

%%%%% input pam_2 + channel %%%%%%
Xk=round(rand(1,M))*2-1; % 2-PAM sequence
Rk=filter(h,1,Xk);         % received signal

%%%%% noise %%%%%%
noise=randn(size(Rk));    % randn generates Gaussian distributed noise
noise1=norm(Rk)/(norm(noise)*10^(dB/20)); % new noise
Rkn=Rk+noise*noise1;                 % add noise into received signal

%%%% equalizer %%%%%
D=round((N+L)/2);  % equalization delay
step_size=0.005;   % step-size of the algorithm
e=zeros(1,M);      % used to save instant error
f=zeros(N+1,1);    % equalizer coefficients, initializations

for i=1:M-100
     rk=Rkn(i+N:-1:i)';      % equalizer input  signal
     yk(i)=f'*rk;            % equalizer output signal
     e(i)=Xk(i+N-D)-yk(i);   % error signal
     f=f+step_size*e(i)*rk;  % update equalizer estimation
end

%%%%% figure %%%%%
subplot(221);
plot(Xk,'o'); % Convergence behavior of the LMS-algorithm.
title('Transmitted signal');
grid;

subplot(222);
plot(Rkn,'o'); % Convergence behavior of the LMS-algorithm.
title('Received signal');
grid;

subplot(223);
plot(yk,'o'); % Convergence behavior of the LMS-algorithm.
title('Equalized signal');
```

```
grid;

subplot(224);
plot(abs(e));                    % show the convergence
title('Convergence');
xlabel ('n');ylabel('Error e(n)');
grid;
```

อัลกอริทึมค่าเฉลี่ยกำลังสองน้อยสุดเมื่อสัญญาณอินพุทเป็น 4- PAM

```
clc;
clear all;
close all;

dB=17;                  % SNR in dB value
N=16;                   % filter length = N+1
M=10000;                % total number of data

%%%%channel%%%%
%h=[0.04 -0.05 0.07 -0.21 -.5 0.72 0.36 0.21 0.03 0.07];              %
impulse of the channel
h=[-0.01269442675941   0.14578007919020   0.73382869513842
0.14578007919020  -0.01269442675941];
%h=randn(1,5);                % impulse of the channel
L=length(h);

%%%% input pam_4 + channel %%%%
a=[-3 -1 1 3];              % 4-PAM alphabet
Xk=a(randint(M,1,4)+1);     % 4-PAM sequence
Rk=filter(h,1,Xk);          % received signal

%%%% noise %%%%
noise=randn(size(Rk));             % randn generates Gaussian
distributed noise
noise1=norm(Rk)/(norm(noise)*10^(dB/20)); % new noise
Rkn=Rk+noise*noise1;               % add noise into received signal

%%%% Equalizer %%%%
D=round((N+L)/2);                  % equalization delay
step_size=0.001;                   % step-size of the algorithm
e=zeros(1,M);                      % used to save instant error
f=zeros(N+1,1);                    % equalizer coefficients,
initializations

for i=1:M-50
     rk=Rkn(i+N:-1:i)';            % equalizer input  signal
     yk(i)=f'*rk;                  % equalizer output signal
     e(i)=Xk(i+N-D)-yk(i);         % error signal
     f=f+step_size*e(i)*rk;        % update equalizer estimation
end

%%%% figure %%%%%
subplot(221);
plot(Xk,'o');              % Convergence behavior of the LMS-algorithm.
title('Transmitted signal');
grid;

subplot(222);
plot(Rkn,'o');             % Convergence behavior of the LMS-algorithm.
title('Received signal');
grid;
```

```
subplot(223);
plot(yk,'o');            % Convergence behavior of the LMS-algorithm.
title('Equalized signal');
grid;

subplot(224);
plot(abs(e));            % show the convergence
title('Convergence');
xlabel ('n');ylabel('Error e(n)');
grid;
```

อัลกอริทึมค่าเฉลี่ยกำลังสองน้อยสุดเมื่อสัญญาณอินพุทเป็น QPSK

```
clc;
close all;
clear all;

%%% Condition %%%
T=4000;             % Number of Data
Lh=5;               % Channel Tap = Lh+1
dB=0;               % noise in dB
M=3500;             % total number of training symbols
mu=0.005;           % parameter to adjust convergence and steady error
N=20;               % Equalizer Tap = N+1

%%% QPSK Signal %%%
qa=[1+1i;-1+1i;-1-1i;1-1i];               % QPSK or 4 QAM alphabet
q=round(rand(1,T))*2-1;
qpsk=q+sqrt(-1)*(round(rand(1,T))*2-1);  % QPSK or 4 QAM symbol
sequence

%%% Channel %%%
h=randn(1,Lh+1)+sqrt(-1)*randn(1,Lh+1);   % channel (complex)
Ch=h/norm(h);                             % normalize
x=filter(Ch,1,qpsk);                      % Passed channel Signal

%%% Adding Noise %%%
noise=randn(size(qpsk))+sqrt(-1)*randn(size(qpsk));  % AWGN noise
(complex)
noise1=norm(x)/norm(noise)*10^(dB/-20);   % adjust noise power with
SNR dB value
SNR=20*log10(norm(x)/norm(noise1*noise))  % Check SNR of the received
samples
xn=x+noise1*noise;                        % received signal

%%% Equalizer %%%
d=round(N/2);         % Equalizer Delay
D=round((Lh+N)/2);    % Total Equalization Delay
Lp=T-N;               % remove several first samples to avoid 0 or
negative subscript
f=zeros(N+1,Lp);      % sample vectors (each column is a sample vector)
for i=1:Lp
    f(:,i)=xn(i+N:-1:i).';
end

w=zeros(N+1,1);                   % initial condition
e=zeros(1,M-d);                   % used to save instant error
for i=1:M-d
    e(i)=qpsk(i+d+N-D)-w'*f(:,i+d);       % instant error
    w=w+mu*conj(e(i))*f(:,i+d);           % update equalizer estimation
```

```
    i_e=[i/10000 abs(e(i))];              % output information
end

y=w'*f;                      % estimate symbols (perform equalization)

%%% Signal Detection %%%
qp=y.';
s_b=repmat(qp,1,4).'; % 4 x Lp -matrix, each line contains the same
sn-vector
c_b=repmat(qa,1,Lp);  % 4 x Lp -matrix, each column contains the
constant
dist=abs(s_b-c_b);    % 4 x Lp -matrix, every column contains the
received symbol
              % distances to all possible symbol constellation points
[a,b]=min(dist);      % Returns the minimum distance a and the
corresponding
         % constellation index b. Both vectors have the size of 1 x T
qpsk_det=qa(b);       % Using vector b, we can determine the detected
symbol vector

for i = 1:Lp
    [a,c]=min(abs(qa-qp(i)));
    qpsk_det(i)=qa(c);
end

%%% calculate Symbol Error Rate (SER) %%%
sb1=y/norm(w);                                    % scale the output
sb1=sign(real(sb1))+sqrt(-1)*sign(imag(sb1));    % perform symbol
detection
start=7;             % carefully find the corresponding begining point
sb2=sb1-qpsk(start+1:start+length(sb1));     % find error symbols
SER=length(find(sb2~=0))/length(sb2)         % calculate SER

%%% figure %%%
if 1
    subplot(221);
    plot(qpsk,'o');          % Show the pattern of transmitted symbols
    title('Transmitted symbols');
    xlabel ('Re');ylabel('Im');
    axis([-2 2 -2 2]);
    grid;

    subplot(222);
    plot(xn,'o');            % Show the pattern of received samples
    title('Received samples');
    xlabel ('Re');ylabel('Im');
    axis([-4 4 -4 4]);
    grid;

    subplot(223);
    plot(y,'o');             % Show the pattern of the equalized symbols
    title('Equalized symbols');
    xlabel ('Re');ylabel('Im');
    axis([-2 2 -2 2]);
    grid;

    subplot(224);
    plot(qpsk_det,'o');     % Show the pattern of the detected symbols
    title('Detected symbols');
    xlabel ('Re');ylabel('Im');
    axis([-2 2 -2 2]);
```

```
    grid;
end

figure(2)
plot(abs(e));              % show the convergence
title('Convergence');
xlabel ('n');ylabel('Error e(n)');
grid;
```

ภาคผนวก ง.

โปรแกรมวีเอชดีแอล

วงจรกรองสัญญาณเชิงเลขชนิดผลตอบสนองอิมพัลส์จำกัด

```
Library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_Unsigned.ALL;

Entity channel5tap is
Port ( clk     : in   std_logic;
       xin     : in   std_logic_vector( 7 downto 0);
       yno     : out  std_logic_vector( 7 downto 0));
End ;

Architecture rtl of channel5tap is
Begin
 PROCESS(clk,xin)

 Type   Yarray is array(4 downto 0) of std_logic_vector(27 downto 0);
 Variable yv          :  Yarray;

 Type  Ynarray is array(3 downto 0) of std_logic_vector(27 downto 0);
 Variable ynd         :  Ynarray;

 Type  Harray  is array(4 downto 0) of std_logic_vector(19 downto 0);
 Variable h           :  Harray;

 Variable xlow        : std_logic_vector(18 downto 0);
 Variable xlowb       : std_logic_vector(20 downto 0);

 Variable ynv         : std_logic_vector(27 downto 0);

 Constant cont        : std_logic_vector(12 downto 0) :="1000000000001";

  BEGIN

    if clk'event and clk='1' then

       h(0):="11111110011000000000";
       h(1):="00010010101010001111";
       h(2):="01011101111011100010";
       h(3):="00010010101010001111";
       h(4):="11111110011000000000";

       yv(0)(27 downto 21):="0000000";
       yv(1)(27 downto 21):="0000000";
       yv(2)(27 downto 21):="0000000";
       yv(3)(27 downto 21):="0000000";
       yv(4)(27 downto 21):="0000000";

       -- 0
       xlow:=( xin(0)&xin(0)&xin(0) &xin(0)&xin(0)&xin(0)
&xin(0)&xin(0)&xin(0) &xin(0)&xin(0)&xin(0) &xin(0)&xin(0)&xin(0)
&xin(0)&xin(0)&xin(0)&xin(0));
       yv(0)(20 downto 0):=('0' & (h(0)(19) nand xin(0)) & (h(0)(18
downto 0) and xlow ));
       yv(1)(20 downto 0):=('0' & (h(1)(19) nand xin(0)) & (h(1)(18
downto 0) and xlow ));
```

```
        yv(2)(20 downto 0):=('0' & (h(2)(19) nand xin(0)) & (h(2)(18
downto 0) and xlow ));
        yv(3)(20 downto 0):=('0' & (h(3)(19) nand xin(0)) & (h(3)(18
downto 0) and xlow ));
        yv(4)(20 downto 0):=('0' & (h(4)(19) nand xin(0)) & (h(4)(18
downto 0) and xlow ));

        -- 1
        xlow:=(xin(1)&xin(1)&xin(1) &xin(1)&xin(1)&xin(1)
&xin(1)&xin(1)&xin(1) &xin(1)&xin(1)&xin(1) &xin(1)&xin(1)&xin(1)
&xin(1)&xin(1)&xin(1)&xin(1));
        yv(0)(21 downto 1):=yv(0)(20 downto 1)+((h(0)(19) nand xin(1))
& (h(0)(18 downto 0) and xlow ));
        yv(1)(21 downto 1):=yv(1)(20 downto 1)+((h(1)(19) nand xin(1))
& (h(1)(18 downto 0) and xlow ));
        yv(2)(21 downto 1):=yv(2)(20 downto 1)+((h(2)(19) nand xin(1))
& (h(2)(18 downto 0) and xlow ));
        yv(3)(21 downto 1):=yv(3)(20 downto 1)+((h(3)(19) nand xin(1))
& (h(3)(18 downto 0) and xlow ));
        yv(4)(21 downto 1):=yv(4)(20 downto 1)+((h(4)(19) nand xin(1))
& (h(4)(18 downto 0) and xlow ));

        -- 2
        xlow:=(xin(2)&xin(2)&xin(2) &xin(2)&xin(2)&xin(2)
&xin(2)&xin(2)&xin(2) &xin(2)&xin(2)&xin(2) &xin(2)&xin(2)&xin(2)
&xin(2)&xin(2)&xin(2) &xin(2));
        yv(0)(22 downto 2):=yv(0)(21 downto 2)+((h(0)(19) nand xin(2))
& (h(0)(18 downto 0) and xlow ));
        yv(1)(22 downto 2):=yv(1)(21 downto 2)+((h(1)(19) nand xin(2))
& (h(1)(18 downto 0) and xlow ));
        yv(2)(22 downto 2):=yv(2)(21 downto 2)+((h(2)(19) nand xin(2))
& (h(2)(18 downto 0) and xlow ));
        yv(3)(22 downto 2):=yv(3)(21 downto 2)+((h(3)(19) nand xin(2))
& (h(3)(18 downto 0) and xlow ));
        yv(4)(22 downto 2):=yv(4)(21 downto 2)+((h(4)(19) nand xin(2))
& (h(4)(18 downto 0) and xlow ));

        -- 3
        xlow:=(xin(3)&xin(3)&xin(3) &xin(3)&xin(3)&xin(3)
&xin(3)&xin(3)&xin(3) &xin(3)&xin(3)&xin(3) &xin(3)&xin(3)&xin(3)
&xin(3)&xin(3)&xin(3) &xin(3));
        yv(0)(23 downto 3):=yv(0)(22 downto 3)+((h(0)(19) nand xin(3))
& (h(0)(18 downto 0) and xlow ));
        yv(1)(23 downto 3):=yv(1)(22 downto 3)+((h(1)(19) nand xin(3))
& (h(1)(18 downto 0) and xlow ));
        yv(2)(23 downto 3):=yv(2)(22 downto 3)+((h(2)(19) nand xin(3))
& (h(2)(18 downto 0) and xlow ));
        yv(3)(23 downto 3):=yv(3)(22 downto 3)+((h(3)(19) nand xin(3))
& (h(3)(18 downto 0) and xlow ));
        yv(4)(23 downto 3):=yv(4)(22 downto 3)+((h(4)(19) nand xin(3))
& (h(4)(18 downto 0) and xlow ));

        -- 4
        xlow:=(xin(4)&xin(4)&xin(4) &xin(4)&xin(4)&xin(4)
&xin(4)&xin(4)&xin(4) &xin(4)&xin(4)&xin(4) &xin(4)&xin(4)&xin(4)
&xin(4)&xin(4)&xin(4) &xin(4));
        yv(0)(24 downto 4):=yv(0)(23 downto 4)+((h(0)(19) nand xin(4))
& (h(0)(18 downto 0) and xlow ));
        yv(1)(24 downto 4):=yv(1)(23 downto 4)+((h(1)(19) nand xin(4))
& (h(1)(18 downto 0) and xlow ));
```

```
        yv(2)(24 downto 4):=yv(2)(23 downto 4)+((h(2)(19) nand xin(4))
& (h(2)(18 downto 0) and xlow ));
        yv(3)(24 downto 4):=yv(3)(23 downto 4)+((h(3)(19) nand xin(4))
& (h(3)(18 downto 0) and xlow ));
        yv(4)(24 downto 4):=yv(4)(23 downto 4)+((h(4)(19) nand xin(4))
& (h(4)(18 downto 0) and xlow ));

        -- 5
        xlow:=(xin(5)&xin(5)&xin(5) &xin(5)&xin(5)&xin(5)
&xin(5)&xin(5)&xin(5) &xin(5)&xin(5)&xin(5) &xin(5)&xin(5)&xin(5)
&xin(5)&xin(5)&xin(5) &xin(5));
        yv(0)(25 downto 5):=yv(0)(24 downto 5)+((h(0)(19) nand xin(5))
& (h(0)(18 downto 0) and xlow ));
        yv(1)(25 downto 5):=yv(1)(24 downto 5)+((h(1)(19) nand xin(5))
& (h(1)(18 downto 0) and xlow ));
        yv(2)(25 downto 5):=yv(2)(24 downto 5)+((h(2)(19) nand xin(5))
& (h(2)(18 downto 0) and xlow ));
        yv(3)(25 downto 5):=yv(3)(24 downto 5)+((h(3)(19) nand xin(5))
& (h(3)(18 downto 0) and xlow ));
        yv(4)(25 downto 5):=yv(4)(24 downto 5)+((h(4)(19) nand xin(5))
& (h(4)(18 downto 0) and xlow ));

        -- 6
        xlow:=(xin(6)&xin(6)&xin(6) &xin(6)&xin(6)&xin(6)
&xin(6)&xin(6)&xin(6) &xin(6)&xin(6)&xin(6) &xin(6)&xin(6)&xin(6)
&xin(6)&xin(6)&xin(6) &xin(6));
        yv(0)(26 downto 6):=yv(0)(25 downto 6)+((h(0)(19) nand xin(6))
& (h(0)(18 downto 0) and xlow ));
        yv(1)(26 downto 6):=yv(1)(25 downto 6)+((h(1)(19) nand xin(6))
& (h(1)(18 downto 0) and xlow ));
        yv(2)(26 downto 6):=yv(2)(25 downto 6)+((h(2)(19) nand xin(6))
& (h(2)(18 downto 0) and xlow ));
        yv(3)(26 downto 6):=yv(3)(25 downto 6)+((h(3)(19) nand xin(6))
& (h(3)(18 downto 0) and xlow ));
        yv(4)(26 downto 6):=yv(4)(25 downto 6)+((h(4)(19) nand xin(6))
& (h(4)(18 downto 0) and xlow ));

        -- 7
        xlow:=(h(0)(18 downto 0) nand (xin(7)&xin(7)&xin(7)
&xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7)
&xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7) &xin(7)));
        xlowb:=('1' & (h(0)(19) and   xin(7)) & xlow );
        yv(0)(27 downto 7):=yv(0)(27 downto 7)+xlowb;

        yv(0)(27 downto 0):=("00000000" & cont(12 downto 0)&
"0000000")+yv(0)(27 downto 0);

        ynv:=ynd(3)+yv(0);
        yno<=ynv(26 downto 19);

        xlow:=(h(1)(18 downto 0) nand (xin(7)&xin(7)&xin(7)
&xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7)
&xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7) &xin(7)));
        xlowb:=('1' & (h(1)(19)  and   xin(7)) & xlow );
        yv(1)(27 downto 7):=yv(1)(27 downto 7)+xlowb;

        yv(1)(27 downto 0):=("00000000" & cont(12 downto 0)&
"0000000")+yv(1)(27 downto 0);

        ynd(3):=ynd(2)+yv(1);
```

```
        xlow:=(h(2)(18 downto 0) nand (xin(7)&xin(7)&xin(7)
&xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7)
&xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7) &xin(7)));
        xlowb:=('1' & (h(2)(19)  and   xin(7)) & xlow );
        yv(2)(27 downto 7):=yv(2)(27 downto 7)+xlowb;

        yv(2)(27 downto 0):=("00000000" & cont(12 downto 0)&
"0000000")+yv(2)(27 downto 0);

        ynd(2):=ynd(1)+yv(2);

        xlow:=(h(3)(18 downto 0) nand (xin(7)&xin(7)&xin(7)
&xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7)
&xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7) &xin(7)));
        xlowb:=('1' & (h(3)(19)  and   xin(7)) & xlow );
        yv(3)(27 downto 7):=yv(3)(27 downto 7)+xlowb;

        yv(3)(27 downto 0):=("00000000" & cont(12 downto 0)&
"0000000")+yv(3)(27 downto 0);

        ynd(1):=ynd(0)+yv(3);

        xlow:=(h(4)(18 downto 0) nand (xin(7)&xin(7)&xin(7)
&xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7)
&xin(7)&xin(7)&xin(7) &xin(7)&xin(7)&xin(7) &xin(7)));
        xlowb:=('1' & (h(4)(19)  and   xin(7)) & xlow );
        yv(4)(27 downto 7):=yv(4)(27 downto 7)+xlowb;

        yv(4)(27 downto 0):=("00000000" & cont(12 downto 0)&
"0000000")+yv(4)(27 downto 0);

        ynd(0):=yv(4);

      end if;

   END PROCESS ;
End;
```

วงจรกลับบิทเครื่องหมาย

```
Library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_unsigned.ALL;

Entity conv is
Port ( x,d          : in std_logic_vector(7 downto 0);
       y            : in std_logic_vector(7 downto 0);
       x_out,d_out: out std_logic_vector(7 downto 0);
       y_out        : out std_logic_vector(7 downto 0));
end;

Architecture rtl of conv is
Begin
      x_out(6 downto 0) <= x(6 downto 0);
      x_out(7)          <= not(x(7));
      d_out(6 downto 0) <= d(6 downto 0);
      d_out(7)          <= not(d(7));
      y_out(6 downto 0) <= y(6 downto 0);
      y_out(7)          <= not(y(7));

End;
```

วงจรหารความถี่

```
Library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_unsigned.ALL;

Entity div_clk is
Port (   sys_clk  : in  std_logic;
         clk_out  : out std_logic);
End ;

Architecture rtl  of div_clk is
  SIGNAL count  : integer range 0 to 99;
Begin
   PROCESS (sys_clk)
     BEGIN
       if sys_clk'event and sys_clk='1' then
               count<=count+1;
               if count<49 then
                  clk_out<='0';
               else
                  clk_out<='1';
               end if;
       end if;
   END PROCESS ;
End;
```

วงจร LFSR ที่มีโพลิโนเมียล m = 2

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity Poly2 is
Port(  Clock       : in std_logic;
       Polyout     : out std_logic);
end;

Architecture rtl of Poly2 is
Signal Poly :  std_logic_vector(1 downto 0);
Begin
  Process(Clock)
      Begin
         if(Clock'event and Clock='1') then
            if(Poly(1 downto 0) = ('0','0')) then
                        Poly(0)     <= '1';
                  else
                        Poly(0) <= Poly(1);
                        Poly(1) <= (Poly(1) xor Poly(0));
                        Polyout <= Poly(1);
                  end if;
            end if;
  end process;
end;
```

วงจร LFSR ที่มีโพลิโนเมียล m = 3

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity Poly3 is
Port( Clock         : in std_logic;
      Polyout       : out std_logic);
end;

Architecture rtl of Poly3 is
Signal Poly :  std_logic_vector(2 downto 0);
Begin
  Process(Clock)
      Begin
         if(Clock'event and Clock='1') then
            if(Poly(2 downto 0) = ('0','0','0')) then
                        Poly(0)     <= '1';
                  else
                        Poly(1 downto 0) <= Poly(2 downto 1);
                        Poly(2) <= (Poly(2) xor Poly(0));
                        Polyout <= Poly(2);
                  end if;
            end if;
  end process;
end;
```

วงจร LFSR ที่มีโพลิโนเมียล m = 5

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity Poly5 is
Port( Clock        : in std_logic;
      Polyout      : out std_logic);
end;

Architecture rtl of Poly5 is
Signal Poly :  std_logic_vector(4 downto 0);
Begin
   Process(Clock)
      Begin
         if(Clock'event and Clock='1') then
            if(Poly(4 downto 0) = ('0','0','0','0','0')) then
                  Poly(1)      <= '1';
            else
                  Poly(3 downto 0) <= Poly(4 downto 1);
                  Poly(4) <= (Poly(4) xor Poly(1));
                  Polyout <= Poly(4);
            end if;
         end if;
   end process;
end;
```

วงจร LFSR ที่มีโพลิโนเมียล m = 6

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity Poly6 is
Port( Clock        : in std_logic;
      Polyout      : out std_logic);
end;

Architecture rtl of Poly6 is
Signal Poly : std_logic_vector(5 downto 0);
Begin
  Process(Clock)
      Begin
        if(Clock'event and Clock='1') then
            if(Poly(5 downto 0) = ('0','0','0','0','0','0')) then
                  Poly(0)      <= '1';
            else
                  Poly(4 downto 0) <= Poly(5 downto 1);
                  Poly(5) <= (Poly(5) xor Poly(0));
                  Polyout <= Poly(5);
            end if;
        end if;
  end process;
end;
```

วงจร LFSR ที่มีโพลิโนเมียล m = 7

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity Poly7 is
Port( Clock       : in std_logic;
      Polyout     : out std_logic);
end;

Architecture rtl of Poly7 is
Signal Poly :  std_logic_vector(6 downto 0);
Begin
  Process(Clock)
      Begin
        if(Clock'event and Clock='1') then
            if(Poly(6 downto 0) = ('0','0','0','0','0','0','0')) then
                  Poly(2)      <= '1';
            else
                  Poly(5 downto 0) <= Poly(6 downto 1);
                  Poly(6) <= (Poly(6) xor Poly(2));
                  Polyout <= Poly(6);
            end if;
        end if;
  end process;
end;
```

วงจร LFSR ที่มีโพลิโนเมียล m = 8

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity Poly8 is
Port(Clock        : in std_logic;
     Polyout      : out std_logic);
end;

Architecture rtl of Poly8 is
Signal Poly :  std_logic_vector(7 downto 0);
Begin
  Process(Clock)
    Begin
      if(Clock'event and Clock='1') then
        if(Poly(7 downto 0) = ('0','0','0','0','0','0','0','0')) then
            Poly(3 downto 1)  <= ('1','1','1');
        else
            Poly(6 downto 0) <= Poly(7 downto 1);
            Poly(7) <= (Poly(7) xor Poly(3)) xor
                         (Poly(2) xor Poly(1));
            Polyout <= Poly(7);
        end if;
      end if;
  end process;
end;
```

วงจร LFSR ที่มีโพลิโนเมียล m = 9

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity Poly9 is
Port( Clock         : in std_logic;
      Polyout       : out std_logic);
end;

Architecture rt1 of Poly9 is
Signal Poly :      std_logic_vector(8 downto 0);
Begin
      Process(Clock)
      Begin
            if(Clock'event and Clock='1') then
                  if(Poly(8 downto 0) =
('0','0','0','0','0','0','0','0','0')) then
                        Poly(3)       <= '1';
                  else
                        Poly(7 downto 0) <= Poly(8 downto 1);
                        Poly(8) <= (Poly(8) xor Poly(3));
                        Polyout <= Poly(8);
                  end if;
            end if;
      end process;
end;
```

วงจร LFSR ที่มีโพลิโนเมียล m = 11

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity Poly11 is
Port(Clock         : in std_logic;
     Polyout       : out std_logic);
end;

Architecture rt1 of Poly11 is
Signal Poly : std_logic_vector(10 downto 0);
Begin
  Process(Clock)
    Begin
      if(Clock'event and Clock='1') then
        if(Poly(10 downto 0) = ("0000000000")) then
           Poly(1)       <= '1';
        else
            Poly(9 downto 0) <= Poly(10 downto 1);
            Poly(10) <= (Poly(10) xor Poly(1));
            Polyout <= Poly(10);
        end if;
      end if;
   end process;
end;
```

วงจรรวมบิท

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity pr_4 is
Port( mm,m2,m1,m0 : in   std_logic;
      u           : out  std_logic_vector(3 downto 0));
end;

Architecture rt1 of pr_4 is
Begin
  process(mm,m2,m1,m0)
      Begin
            u <= (mm & m2 & m1 & m0);
  end process;
end;
```

วงจรตัดบิท

```
Library ieee;
Use ieee.std_logic_1164.all;

Entity clamp is
Port( vin   : in   std_logic_vector(5 downto 0);
      vout  : out  std_logic_vector(4 downto 0));
end;

Architecture rt1 of clamp is
Begin
   process(vin)
      Begin
            vout <= vin(5 downto 1);
   end process;
end;
```

วงจรคูณ

```
Library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_Unsigned.ALL;

Entity mult is
Port (win     : in  std_logic_vector (12 downto 0);
      vin     : in  std_logic_vector (4 downto 0);
      xk      : out std_logic_vector (17 downto 0);
      sign    : out std_logic);
End;

Architecture rtl  of mult is
Begin
  Process(win,vin)
      variable com       : std_logic_vector(4 downto 0);
       Begin
           if      vin(4)='1' then
                   com    :=     (((vin xor "11111") + '1'));
                   xk           <=     win*com;
          elsif    vin(4)='0' then
                   xk     <=     win*vin;
          end if;
            sign    <=     vin(4);
  end process;
end;
```

วงจรทูคอมพลีเมนท์

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;

Entity two_comp is
Port( Data_in       : in   std_logic_vector(17 downto 0);
      Data_out      : out  std_logic_vector(18 downto 0);
      sign          : in   bit);
end;

Architecture rtl of two_comp is
Begin
  Process(Data_in,sign)
    variable one_comp    : std_logic_vector(18 downto 0);
      Begin
          if sign='1' then
             one_comp := (('0' & Data_in) xor "1111111111111111111") +
'1';
             Data_out <= one_comp ;
      elsif sign='0' then
             Data_out <= '0' & Data_in;
      end if;
  end process;
end;
```

วงจรบัฟเฟอร์

```
library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_unsigned.ALL;

Entity reg is
port (a     : in  std_logic_vector(18 downto 0);
      clk   : in  std_logic;
      b     : out std_logic_vector(18 downto 0));
end;

Architecture rtl of reg is
begin
  process(a,clk)
     begin
       if clk'event and clk = '1'  then
            b<=a;
       end if;
  end process;
end;
```

วงจรบวก

```
library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_unsigned.ALL;

Entity adder is
port (x,y    : in  std_logic_vector(18 downto 0);
      z      : out std_logic_vector(19 downto 0));
end;

Architecture rtl of adder is
begin
        z <= x+y;
end;
```

วงจรกรองความถี่ผ่านแบบปรับตัวได้

```
Library ieee;
Use ieee.std_logic_1164.ALL;
Use ieee.std_logic_Unsigned.ALL;

PACKAGE componentmul IS
  subtype bit_vector is std_logic_vector(19 downto 0);
   Function  and_xh(xbit  : in  std_logic;
                   hbits : in  bit_vector)   return
std_logic_vector;
   Function nand_xh(xbit  : in  std_logic;
                   hbits : in  bit_vector)   return
std_logic_vector;
   Function  twocomadd(a  : in  bit_vector;
                      b  : in  bit_vector;
                      as : in  std_logic)   return std_logic_vector;
END;

PACKAGE BODY componentmul IS
 Function  and_xh(xbit  : in  std_logic;
                 hbits : in  bit_vector) return std_logic_vector  IS
  Begin
      return ((hbits(19) nand xbit) &
             ( hbits(18 downto 0) and (xbit & xbit & xbit & xbit &
                                      xbit & xbit & xbit & xbit &
                                      xbit & xbit & xbit & xbit &
                                      xbit & xbit & xbit & xbit &
                                      xbit & xbit & xbit)) );
     end;

 Function nand_xh(xbit  : in  std_logic;
                 hbits : in  bit_vector) return std_logic_vector  IS
  Begin
     return ((hbits(19) and xbit) &
            ( hbits(18 downto 0) nand (xbit & xbit & xbit & xbit &
                                      xbit & xbit & xbit & xbit &
                                      xbit & xbit & xbit & xbit &
                                      xbit & xbit & xbit & xbit &
                                      xbit & xbit & xbit)) );
     end;

 Function  twocomadd(a   : in  bit_vector;
                    b   : in  bit_vector;
                    as  : in  std_logic) return std_logic_vector  IS
  Begin
     return ((as&as&as&as & as&as&as&as & as&as&as&as &
              as&as&as&as & as&as&as&as ) xor b)+a+as;
     end;
END;

LIBRARY ieee;   USE ieee.std_logic_1164.ALL;
                Use ieee.std_logic_Unsigned.ALL;
LIBRARY work;   USE work.componentmul.ALL;

Entity lms8_20bitwooley is
 GENERIC(N : INTEGER :=19);
Port ( clk   : in   std_logic;
       x     : in   std_logic_vector( 7 downto 0);
       d     : in   std_logic_vector( 7 downto 0);
       y     : out  std_logic_vector( 7 downto 0));
```

```
End ;

Architecture rtl of lms8_20bitwooley is

  Begin
   PROCESS(clk,x,d)

      Type      Yarray      is array(8 downto 0) of std_logic_vector(20
downto 0);
      Variable yv           :  Yarray;

      Type      Ynarray     is array(8 downto 0) of std_logic_vector(19
downto 0);
      Variable ynd          :  Ynarray;

      Type      Harray      is array(8 downto 0) of std_logic_vector(19
downto 0);
      Variable   h          :  Harray;

      Variable  count       :  std_logic;

      Constant const_add : std_logic_vector(12 downto 0) :=
"1000000000001";
      Variable err         : std_logic_vector(19 downto 0);
      Variable xsign       : std_logic_vector(8 downto 0);

    BEGIN
      if clk'event and clk='1' then

           err:=("00" & d & "0000000000")+not(ynd(8)+'1');

           xsign(8 downto 1):=xsign(7 downto 0);
           xsign(0):=x(7);

             yv(0):="000000000000000000000";
             yv(1):="000000000000000000000";
             yv(2):="000000000000000000000";
             yv(3):="000000000000000000000";
             yv(4):="000000000000000000000";
             yv(5):="000000000000000000000";
             yv(6):="000000000000000000000";
             yv(7):="000000000000000000000";
             yv(8):="000000000000000000000";

--         1
         yv(N-19):=('0' & and_xh(x(0),h(N-19)));
           yv(N-19):=yv(N-19)(N downto 7)+const_add;

         yv(N-18):=('0' & and_xh(x(0),h(N-18)));
           yv(N-18):=yv(N-18)(N downto 7)+const_add;

         yv(N-17):=('0' & and_xh(x(0),h(N-17)));
           yv(N-17):=yv(N-17)(N downto 7)+const_add;

         yv(N-16):=('0' & and_xh(x(0),h(N-16)));
           yv(N-16):=yv(N-16)(N downto 7)+const_add;

         yv(N-15):=('0' & and_xh(x(0),h(N-15)));
           yv(N-15):=yv(N-15)(N downto 7)+const_add;

         yv(N-14):=('0' & and_xh(x(0),h(N-14)));
```

```
      yv(N-14):=yv(N-14)(N downto 7)+const_add;

    yv(N-13):=('0' & and_xh(x(0),h(N-13)));
      yv(N-13):=yv(N-13)(N downto 7)+const_add;

    yv(N-12):=('0' & and_xh(x(0),h(N-12)));
      yv(N-12):=yv(N-12)(N downto 7)+const_add;

    yv(N-11):=('0' & and_xh(x(0),h(N-11)));
      yv(N-11):=yv(N-11)(N downto 7)+const_add;

--    2
      yv(N-19):=yv(N-19)(N+1 downto 1)+and_xh(x(1),h(N-19));
      yv(N-18):=yv(N-18)(N+1 downto 1)+and_xh(x(1),h(N-18));
      yv(N-17):=yv(N-17)(N+1 downto 1)+and_xh(x(1),h(N-17));
      yv(N-16):=yv(N-16)(N+1 downto 1)+and_xh(x(1),h(N-16));
      yv(N-15):=yv(N-15)(N+1 downto 1)+and_xh(x(1),h(N-15));
      yv(N-14):=yv(N-14)(N+1 downto 1)+and_xh(x(1),h(N-14));
      yv(N-13):=yv(N-13)(N+1 downto 1)+and_xh(x(1),h(N-13));
      yv(N-12):=yv(N-12)(N+1 downto 1)+and_xh(x(1),h(N-12));
      yv(N-11):=yv(N-11)(N+1 downto 1)+and_xh(x(1),h(N-11));

--    3
      yv(N-19):=yv(N-19)(N+1 downto 1)+and_xh(x(2),h(N-19));
      yv(N-18):=yv(N-18)(N+1 downto 1)+and_xh(x(2),h(N-18));
      yv(N-17):=yv(N-17)(N+1 downto 1)+and_xh(x(2),h(N-17));
      yv(N-16):=yv(N-16)(N+1 downto 1)+and_xh(x(2),h(N-16));
      yv(N-15):=yv(N-15)(N+1 downto 1)+and_xh(x(2),h(N-15));
      yv(N-14):=yv(N-14)(N+1 downto 1)+and_xh(x(2),h(N-14));
      yv(N-13):=yv(N-13)(N+1 downto 1)+and_xh(x(2),h(N-13));
      yv(N-12):=yv(N-12)(N+1 downto 1)+and_xh(x(2),h(N-12));
      yv(N-11):=yv(N-11)(N+1 downto 1)+and_xh(x(2),h(N-11));

--    4
      yv(N-19):=yv(N-19)(N+1 downto 1)+and_xh(x(3),h(N-19));
      yv(N-18):=yv(N-18)(N+1 downto 1)+and_xh(x(3),h(N-18));
      yv(N-17):=yv(N-17)(N+1 downto 1)+and_xh(x(3),h(N-17));
      yv(N-16):=yv(N-16)(N+1 downto 1)+and_xh(x(3),h(N-16));
      yv(N-15):=yv(N-15)(N+1 downto 1)+and_xh(x(3),h(N-15));
      yv(N-14):=yv(N-14)(N+1 downto 1)+and_xh(x(3),h(N-14));
      yv(N-13):=yv(N-13)(N+1 downto 1)+and_xh(x(3),h(N-13));
      yv(N-12):=yv(N-12)(N+1 downto 1)+and_xh(x(3),h(N-12));
      yv(N-11):=yv(N-11)(N+1 downto 1)+and_xh(x(3),h(N-11));

--    5
      yv(N-19):=yv(N-19)(N+1 downto 1)+and_xh(x(4),h(N-19));
      yv(N-18):=yv(N-18)(N+1 downto 1)+and_xh(x(4),h(N-18));
      yv(N-17):=yv(N-17)(N+1 downto 1)+and_xh(x(4),h(N-17));
      yv(N-16):=yv(N-16)(N+1 downto 1)+and_xh(x(4),h(N-16));
      yv(N-15):=yv(N-15)(N+1 downto 1)+and_xh(x(4),h(N-15));
      yv(N-14):=yv(N-14)(N+1 downto 1)+and_xh(x(4),h(N-14));
      yv(N-13):=yv(N-13)(N+1 downto 1)+and_xh(x(4),h(N-13));
      yv(N-12):=yv(N-12)(N+1 downto 1)+and_xh(x(4),h(N-12));
      yv(N-11):=yv(N-11)(N+1 downto 1)+and_xh(x(4),h(N-11));

--    6
      yv(N-19):=yv(N-19)(N+1 downto 1)+and_xh(x(5),h(N-19));
      yv(N-18):=yv(N-18)(N+1 downto 1)+and_xh(x(5),h(N-18));
      yv(N-17):=yv(N-17)(N+1 downto 1)+and_xh(x(5),h(N-17));
      yv(N-16):=yv(N-16)(N+1 downto 1)+and_xh(x(5),h(N-16));
      yv(N-15):=yv(N-15)(N+1 downto 1)+and_xh(x(5),h(N-15));
```

```
            yv(N-14):=yv(N-14)(N+1 downto 1)+and_xh(x(5),h(N-14));
            yv(N-13):=yv(N-13)(N+1 downto 1)+and_xh(x(5),h(N-13));
            yv(N-12):=yv(N-12)(N+1 downto 1)+and_xh(x(5),h(N-12));
            yv(N-11):=yv(N-11)(N+1 downto 1)+and_xh(x(5),h(N-11));

--          7
            yv(N-19):=yv(N-19)(N+1 downto 1)+and_xh(x(6),h(N-19));
            yv(N-18):=yv(N-18)(N+1 downto 1)+and_xh(x(6),h(N-18));
            yv(N-17):=yv(N-17)(N+1 downto 1)+and_xh(x(6),h(N-17));
            yv(N-16):=yv(N-16)(N+1 downto 1)+and_xh(x(6),h(N-16));
            yv(N-15):=yv(N-15)(N+1 downto 1)+and_xh(x(6),h(N-15));
            yv(N-14):=yv(N-14)(N+1 downto 1)+and_xh(x(6),h(N-14));
            yv(N-13):=yv(N-13)(N+1 downto 1)+and_xh(x(6),h(N-13));
            yv(N-12):=yv(N-12)(N+1 downto 1)+and_xh(x(6),h(N-12));
            yv(N-11):=yv(N-11)(N+1 downto 1)+and_xh(x(6),h(N-11));

--          8
            yv(N-19):=('0' & yv(N-19)(N+1 downto 1))+('1' &
nand_xh(x(7),h(N-19)));
            ynd(8):=yv(0)(N+1 downto 1)+ynd(7);
            y<=ynd(8)(N-2 downto N-9);

            yv(N-18):=('0' & yv(N-18)(N+1 downto 1))+('1' &
nand_xh(x(7),h(N-18)));
            ynd(7):=yv(1)(N+1 downto 1)+ynd(6);

            yv(N-17):=('0' & yv(N-17)(N+1 downto 1))+('1' &
nand_xh(x(7),h(N-17)));
            ynd(6):=yv(2)(N+1 downto 1)+ynd(5);

            yv(N-16):=('0' & yv(N-16)(N+1 downto 1))+('1' &
nand_xh(x(7),h(N-16)));
            ynd(5):=yv(3)(N+1 downto 1)+ynd(4);

            yv(N-15):=('0' & yv(N-15)(N+1 downto 1))+('1' &
nand_xh(x(7),h(N-15)));
            ynd(4):=yv(4)(N+1 downto 1)+ynd(3);

            yv(N-14):=('0' & yv(N-14)(N+1 downto 1))+('1' &
nand_xh(x(7),h(N-14)));
            ynd(3):=yv(5)(N+1 downto 1)+ynd(2);

            yv(N-13):=('0' & yv(N-13)(N+1 downto 1))+('1' &
nand_xh(x(7),h(N-13)));
            ynd(2):=yv(6)(N+1 downto 1)+ynd(1);

            yv(N-12):=('0' & yv(N-12)(N+1 downto 1))+('1' &
nand_xh(x(7),h(N-12)));
            ynd(1):=yv(7)(N+1 downto 1)+ynd(0);

            yv(N-11):=('0' & yv(N-11)(N+1 downto 1))+('1' &
nand_xh(x(7),h(N-11)));
            ynd(0):=yv(8)(N+1 downto 1);

            if count='0' then
               count:=not(count);
                     h(0):="00000100110011100000";
                     h(1):="00000100110011100000";
                     h(2):="00000100110011100000";
                     h(3):="00000100110011100000";
                     h(4):="00000100110011100000";
```

```
                    h(5):="000001001100111000000";
                    h(6):="000001001100111000000";
                    h(7):="000001001100111000000";
                    h(8):="000001001100111000000";

          else
                    h(0):=twocomadd(h(0),err,xsign(0));
                    h(1):=twocomadd(h(1),err,xsign(1));
                    h(2):=twocomadd(h(2),err,xsign(2));
                    h(3):=twocomadd(h(3),err,xsign(3));
                    h(4):=twocomadd(h(4),err,xsign(4));
                    h(5):=twocomadd(h(5),err,xsign(5));
                    h(6):=twocomadd(h(6),err,xsign(6));
                    h(7):=twocomadd(h(7),err,xsign(7));
                    h(8):=twocomadd(h(8),err,xsign(8));

          end if;
    end if;
 END PROCESS;
End;
```