การถอดรหัสข้อมูลเสียงตามมาตรฐานเอ็มเป็ก

MPEG DECODING

โดย

นางสาว ทิตยา เหลืองศิริธัญญะ เลขประจำตัว 40010275
นาย ธรรมนูญ ชัยปัญญากุล เลขประจำตัว40010310
นาย ธิติ อึ้งอารีย์วิทยา เลขประจำตัว 40010320

อาจารย์ที่ปรึกษา

ผศ.ดร.สมศักดิ์ ชุมช่วย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

การถอดรหัสข้อมูลเสียงตามมาตรฐานเอ็มเป็ก

MPEG DECODING

โดย

นางสาว ทิตยา เหลืองศิริธัญญะ เลขประจำตัว 40010275

นาย ธรรมนูญ ชัยปัญญากุล เลขประจำตัว40010310

นาย ธิติ อึ้งอารีย์วิทยา เลขประจำตัว 40010320

อาจารย์ที่ปรึกษา

ผศ.ดร.สมศักดิ์ ชุมช่วย

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

รายงาน ภาคเรียนที่ 2 ปีการศึกษา 2543

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การถอดรหัสข้อมูลเสียงตามมาตรฐานเอ็มเป็ก

ผู้จัดทำ

นางสาวทิตยา เหลืองศิริธัญญะ 40010275

นายธรรมนูญ ชัยปัญญากุล 40010310

นายธิติ อึ้งอารีย์วิทยา 40010320

..... อาจารย์ที่ปรึกษา

(ผศ.ดร.สมศักดิ์ ชุมช่วย)

การถอดรหัสข้อมูลเสียงตามมาตรฐานเอ็มเป็ก

MPEG DECODING

นางสาวทิตยา เหลืองศิริธัญญะ 40010275

นายธรรมนูญ ชัยปัญญากุล 40010310

นายธิติ อึ้งอารีย์วิทยา 40010320

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

..... อาจารย์ที่ปรึกษา

(ผศ.ดร. สมศักดิ์ ชุมช่วย)

การถอดรหัสข้อมูลเสียงตามมาตรฐานเอ็มเป็ก

นางสาวทิตยา เหลืองศิริธัญญะ
นายธรรมนูญ ชัยปัญญากุล
นายธิติ อึ้งอารีย์วิทยา
ผศ.ดร. สมศักดิ์ ชุมช่วย อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ปริญญานิพนธ์เรื่องการถอดรหัสข้อมูลเสียงตามมาตรฐานเอ็มเป็ก เป็นการศึกษาขั้นตอนในการถอดรหัสข้อมูลเสียงตามมาตรฐานเอ็มเป็ก 1 เลเยอร์ 3 โดยเขียนเอ็มไฟล์ของโปรแกรมแมทแลบ ซึ่งเป็นโปรแกรมที่ช่วยในการประมวลผลสัญญาณและการคำนวณทางคณิตศาสตร์ เพื่อทำความเข้าใจหลักการและวิธีของการถอดรหัส และประเมินความเป็นไปได้ที่จะสร้างวงจรรวมถอดรหัส โดยออกแบบวงจรรวมที่สามารถรับขบวนบิตของข้อมูลเสียงเอ็มเป็ก 1 เลเยอร์ 3 และจัดเก็บค่าต่างๆ ไว้เพื่อใช้ประมวลผลในขั้นตอนถัดไปด้วยภาษาวีเอชดีแอล ซึ่งเป็นภาษาที่ใช้อธิบายพฤติกรรมวงจรในระดับวงจรเกท โดยใช้โปรแกรมโมเดลซิมในการจำลองการทำงาน แล้วนำผลที่ได้มาวิเคราะห์ถึงความเป็นไปได้ที่จะออกแบบวงจรรวมถอดรหัสข้อมูลเสียงเอ็มเป็ก 1 เลเยอร์ 3 ทุกขั้นตอน เพื่อนำไปใช้งานจริงต่อไป

MPEG DECODING

Tittaya Laungsirithanya
Tammanoon Chaipanyakul
Thiti Ungarreevitaya
Dr. Somsak Chumchuay advisor
2000

ABSTRACT

This Thesis is to studying MPEG Decoding Algorithm by building m-file program in MATLAB which helps for digital signal processing and mathematics, then designs the integrated circuit that recieves MPEG 1 LAYER3 bitstreams and manages this data to collect in memory for using in next process.The operation of this circuit is specified via a High-Level Description Language(HDL) and verified to the predefined,proper signals by using HDL-Simulator.And assess the probability for implementation all process in MPEG Decoder Chip

สารบัญ

	หน้าที่
บทคัดย่อ	i
ABSTRACT	ii
สารบัญ	iii
สารบัญภาพ	vii
สารบัญตาราง	ix
บทที่ 1 บทนำ	1
1.1 กล่าวนำ	1
1.2 วัตถุประสงค์ของปริญญานิพนธ์	1
1.3 ขอบเขตของปริญญานิพนธ์	1
1.4 แผนการดำเนินงาน	2
1.5 ประโยชน์และผลที่ได้รับจากการดำเนินงาน	2
บทที่ 2 การบีบอัดข้อมูล	4
2.1 ทำไมจึงต้องบีบอัดข้อมูล	4
2.2 การประยุกต์การบีบอัดข้อมูล	4
2.3 รูปแบบในการบีบอัดข้อมูล	5
2.3.1 แบบไม่มีการสูญเสีย	5
2.3.2 แบบมีการสูญเสีย	6
บทที่ 3 การบีบอัดข้อมูลเสียงแบบเอ็มเป็ก	7
3.1 บทนำ	7
3.2 หลักการพื้นฐานและการใช้งาน	7
3.3 การเข้ารหัสแบบเอ็มเป็กโดยทั่วไป	8
3.3.1 โพลีเฟส ฟิลเตอร์แบงค์	9
3.3.2 ไซโครอะคูสติก โมเดล	14
3.3.3 การแบ่งแยกบิต	16
3.3.4 การเข้ารูปแบบ	17
3.4 การเข้ารหัสข้อมูลแบบเอ็มเป็กเลเยอร์ 3	18
บทที่ 4 การถอดรหัสข้อมูลเอ็มเป็ก 1 เลเยอร์ 3	20

	หน้าที่
4.1 การดึงข้อมูลและค้นหาส่วนหัวของเฟรม	21
4.2 การตรวจสอบความผิดพลาด	21
4.3 การถอดรหัสข้อมูลข้างเคียง	22
4.4 การถอดรหัสสเกลแฟคเตอร์	23
4.5 การถอดรหัสข้อมูลฮัฟแมน	27
4.6 การรีควอนไตซ์	30
4.7 การลดค่าปลอม	37
4.8 IMDCT(Inverse Modified Discrete Cosine Transform)	38
4.9 การทำวินโดว์	39
4.10 การทำโอเวอร์แลป	40
4.11 การรวมสัญญาณจากแต่ละช่องตัวกรอง	41
4.11.1 การส่งค่าอินพุทเข้าไปยังโพลีเฟส ฟิลเตอร์แบงค์	42
4.11.2 IDCT (Inverse Discrete Cosine Transform)	42
4.11.3 การสร้างค่า 512 ค่า จากค่า x_i	42
4.11.4 การคูณค่าสัมประสิทธิ์ของการสังเคราะห์วินโดว์	42
4.11.5 การคำนวณค่าของข้อมูลสุ่มตัวอย่าง PCM	42
บทที่ 5 โปรแกรมแมทแลบ	44
5.1 บทนำ	44
5.2 คำสั่งพื้นฐานทั่วไป	44
5.3 การใช้งานโปรแกรมแมทแลบ	46
บทที่ 6 ภาษาวีเอชดีแอล	48
6.1 บทนำ	48
6.2 ประวัติความเป็นมาของภาษาวีเอชดีแอล	48
6.3 ความสามารถของภาษาวีเอชดีแอล	49
6.4 กระบวนวิธีในการออกแบบโดยภาษาวีเอชดีแอล	51
6.5 หลักการสร้างโมเดลโดยใช้ภาษาวีเอชดีแอล	52
6.6 ระดับของการอธิบายระบบ	54
6.6.1 Behavioral Description	55
6.6.2 Dataflow Description	55

	หน้าที่
6.6.3 Structural Description	55
6.7 สรุป	56
บทที่ 7 การทดลองและผลการทดลอง	57
7.1 การทดลองในส่วนโปรแกรมแมบแลบ	57
7.2 ผลการทดลองในส่วนโปรแกรมแมทแลบ	58
7.3 การทดลองในส่วน VHDL	64
7.4 ผลการทดลองในส่วน VHDL	70
บทที่ 8 สรุปผลการทดลองและการดำเนินงาน	73
8.1 สรุปผลการทดลองในส่วนโปรแกรมแมทแลบ	73
8.2 สรุปผลการทดลองในส่วน VHDL	74
8.3 การประเมิณความเป็นไปได้ในการสร้างวงจรรวมถอดรหัส	75
8.4 ความก้าวหน้าของการดำเนินงาน	75
8.5 ปัญหาจากการดำเนินงาน	76
8.6 สรุปผลการดำเนินงานทั้งหมด	78
ภาคผนวก	80
ภาคผนวก ก บล็อกไดอะแกรมและโฟล์วชาร์ตประกอบบทที่ 4	81
ภาคผนวก ข รูปแบบข้อมูลเสียงเอ็มเป็กเลเยอร์ 3	101
ข.1 ไวยากรณ์ของข้อมูลเอ็มเป็กเลเยอร์ 3	101
ข.1.1 ลำดับข้อมูลเสียง	101
ข.1.2 เฟรมข้อมูลเสียง	101
ข.1.3 ส่วนหัวข้อมูล	101
ข.1.4 ส่วนตรวจสอบความผิดพลาด	102
ข.1.5 ส่วนข้อมูลเสียง	102
ข.1.6 ส่วนข้อมูลหลัก	103
ข.1.7 ส่วนการเข้ารหัสแบบฮัฟแมน	105
ข.1.8 ข้อมูลช่วย	106
ข.2 ความหมายของคำในไวยากรณ์ข้อมูลเอ็มเป็กเลเยอร์ 3	106
ข.2.1 ลำดับสัญญาณเสียงทั่วๆ ไป	106

	หน้าที่
ข.2.2 เฟรมของสัญญาณ	106
ข.2.3 ส่วนหัวของข้อมูล	107
ข.2.4 ส่วนตรวจสอบความผิดพลาด	108
ข.2.5 ส่วนข้อมูลเสียง	109
ข.2.6 ข้อมูลช่วย	114
ภาคผนวก ค โปรแกรมถอดรหัสด้วยโปรแกรมแมทแลบ	115
ภาคผนวก ง โมเดลการถอดรหัสด้วยภาษา VHDL	155
กิตติกรรมประกาศ	205
บรรณานุกรม	206

สารบัญภาพ

หน้าที่

รูปที่ 2.1 แสดงการไหลของข้อมูลจากอินพุทไปที่เอาท์พุท 5
รูปที่ 2.2 แสดงข้อมูลก่อนการบีบอัดและข้อมูลหลังการขยาย 5
รูปที่ 2.3 แสดงการบีบอัดแบบมีการสูญเสีย(Lossy) 6
รูปที่ 3.1 กราฟเปรียบเทียบอัตราส่วนการบีบอัดเสียงของการเข้ารหัส
เลเยอร์ต่างๆเมื่อเทียบกับคุณภาพเสียงต้นแบบจากคอมแพ็คดิสก์ 8
รูปที่ 3.2 แสดงบล็อกไดอะแกรมของการเข้ารหัสและการถอดรหัสแบบเอ็มเป็ก 9
รูปที่ 3.3 แสดงโฟล์วชาร์ต (Flowchart) อธิบายขั้นตอนการทำโพลีเฟส ฟิลเตอร์แบงค์ 10
รูปที่ 3.4 แสดงการเปรียบเทียบสัมประสิทธ์ C[n] และ h[n] 12
รูปที่ 3.5 แสดงการเหลื่อมซึ่งกันและกันระหว่างย่านความถี่ย่อยที่ติดกัน 13
รูปที่ 3.6 แสดงการเกิดเอาท์พุทของโพลีเฟส ฟิลเตอร์แบงค์เมื่อสัญญาณเข้าเป็น
สัญญาณไซน์หนึ่งความถี่ 13
รูปที่ 3.7 แสดงระดับความดังเสียงที่เริ่มได้ยินที่ความถี่ต่าง ๆ 14
รูปที่ 3.8 แสดงผลของการปิดกั้นเสียง 15
รูปที่ 3.9 แสดงการปิดกั้นเสียงที่ไม่ได้ยินในย่านความถี่ย่อยหนึ่ง ๆ
ของการเข้ารหัสเอ็มเป็ก 15
รูปที่ 3.10 แสดงรูปแบบข้อมูลเอ็มเป็กเลเยอร์ 3 17
รูปที่ 3.11 แสดงบิตต่าง ๆ ในส่วนหัวข้อมูล 17
รูปที่ 3.12 แสดงบล็อคไดอะแกรมของการทำ MDCT 18
รูปที่ 4.1 แสดงโฟล์วชาร์ต (Flow Chart) ของการถอดรหัส ไฟล์
“เอ็มเป็ก – 1 เลเยอร์ – 3” 20
รูปที่ 4.2 แสดงโฟล์วชาร์ตการทำงานในส่วนนี้ 21
รูปที่4.4 แสดงการซ้อนทับ (Overlapping) ของสัญญาณ 40
รูปที่ 4.5 แสดงโฟลว์ชาร์ตแสดงกระบวนการรวมสัญญาณจากแต่ละย่านความถี่ย่อย 41
รูปที่ 6.1 กรรมวิธีในการออกแบบโดยใช้ Hardware Description Language และ
ระเบียบวิธีในการออกแบบโดยใช้ Hardware Description Language 51
รูปที่ 7.1 บล็อคไดอะแกรมแสดงการเชื่อมต่อกันของฮาร์ดแวร์ 64

หน้าที่

รูปที่ 7.2 แสดงหน้าต่างโปรแกรมโมเดลซิมที่ใช้ในการทดลองวงจรรวม ที่ออกแบบโดยภาษา VHDL 65
รูปที่ 7.3 แสดงรูปสัญญาณการตรวจพบซิงค์เวิร์ด 66
รูปที่ 7.4 แสดงค่าของข้อมูลที่เก็บได้ 69
รูปที่ ก.1 บล๊อกไดอะแกรมการเข้ารหัสเอ็มเป็กเลเยอร์3 81
รูปที่ ก.2 บล๊อกไดอะแกรมการถอดรหัสเอ็มเป็กเลเยอร์3 82
รูปที่ ก.3 แสดงการถอดรหัสเอ็มเป็กเลเยอร์3 83
รูปที่ ก.4 รูปแบบของข้อมูลบิตสตรีม 84
รูปที่ ก.5 แสดง อินพุท และ เอาท์พุท ของการถอดรหัสสเกลแฟคเตอร์ 85
รูปที่ ก.6 โฟลว์ชาร์ตหลักของการถอดรหัสสเกลแฟคเตอร์ 87
รูปที่ ก.7 การคำนวณหาจำนวนบิตของข้อมูลที่เป็นสเกลแฟคเตอร์ของแกรนนูล 0 88
รูปที่ ก.8 การคำนวณหาจำนวนบิตของข้อมูลที่เป็นสเกลแฟคเตอร์ของแกรนนูล 1 89
รูปที่ ก.9 แสดงการหาขอบเขตของแต่ละส่วนภายใน big_value และการเลือกใช้ตาราง 90
รูปที่ ก.10 โฟล์วชาร์ต แสดงขั้นตอน และการตัดสินใจ ในการถอดรหัสฮัฟแมน 91
รูปที่ ก.11 แสดงการถอดรหัสฮัฟแมนโค้ด 92
รูปที่ ก.12 แสดงการเข้าหัสแบบสเตอริโอ 93
รูปที่ ก.13 แสดงการเข้ารหัส สเตอริโอ 94
รูปที่ ก.14 แแสดงการถอดรหัสจากบิตสตรีมเป็นสัญญาณเชิงความถี่แล้วแปลง เป็นสัญญาณในเชิงเวลา 95
รูปที่ ก.15 แสดงการทำ aliasing reduction และการทำ IMDCT 96
รูปที่ ก.16 แสดงการทำโอเวอร์แลป 97
รูปที่ ก.17 แสดงการทำโพลีเฟส ฟิลเตอร์ แบงก์1 98
รูปที่ ก.18 แสดงการทำโพลีเฟส ฟิสเตอร์ แบงก์2 99
รูปที่ ก.19 วงจรแสดงการคำนวณหาค่า CRC โดยใช้ Linear Feedback Shift Register 100

สารบัญตาราง

หน้าที่

ตารางที่ 3.1 แสดงความกว้างของย่านความถี่วิกฤติย่านต่าง ๆ 16
ตารางที่ 4.1 แสดงค่า pretab[cb] ที่ scalefactor band ต่างๆ 31
ตารางที่ 4.2 แสดงค่าสัมประสิทธ์ของการลดค่าปลอม(Coefficients for alias reduction)38
ตารางที่ ก.1 แสดงข้อมูลส่วนหัวของเฟรม 86
ตารางที่ ข.1 ความหมายของรหัสข้อมูลในเลเยอร์ 107
ตารางที่ ข.2 ความหมายของรหัสข้อมูลใน Mode 108
ตารางที่ ข.3 ความหมายของรหัสข้อมูลใน Mode_extention 108
ตารางที่ ข.4 ความหมายของรหัสข้อมูลใน Emphasis 108
ตารางที่ ข.5 ความหมายของรหัสข้อมูลใน Emphasis 109
ตารางที่ ข.6 ความหมายของรหัสข้อมูลใน scfsi_band 109
ตารางที่ ข.7 ความหมายของรหัสข้อมูลใน scalefac_compress[gr][ch] 110
ตารางที่ ข.8 ความหมายของรหัสข้อมูลใน block_type[gr] 112
ตารางที่ ข.9 ความหมายของรหัสข้อมูลใน scalefac_scale[gr] 113
ตารางที่ ข.10 ความหมายของรหัสข้อมูลใน count1table_select[gr][ch] 113

บทที่ 1
บทนำ

1.1 กล่าวนำ

ในปัจจุบัน การส่งสัญญาณวิทยุ และโทรทัศน์ ได้มีการใช้เทคโนโลยีทางดิจิตอลเข้ามาแทนที่เทคโนโลยีทางอนาลอก ได้มีการคาดหมายไว้ว่า การส่งสัญญาณเสียงดิจิตอลจะแพร่หลายแทนที่การส่งสัญญาณเสียงวิทยุเอฟเอ็มภายใน 20 ปีนี้ ในขณะที่เทคโนโลยีดิจิตอลเอ็ชดีทีวี (Digital HDTV Alliance) ได้ถูกใช้เป็นมาตรฐานในประเทศญี่ปุ่นและสหรัฐอเมริกาแล้ว

เหตุผลในการเปลี่ยนเป็นการส่งสัญญาณทางดิจิตอลนี้ คือ สัญญาณรบกวน (Noise) จะมีน้อยมากจนไม่มีผล พลังงานของสัญญาณที่ส่งจะดีกว่าที่การส่งในช่วงส่งผ่านสัญญาณความถี่ต่ำ และมีการลดทอนน้อยกว่า ถึงแม้ว่าเทคโนโลยีทางดิจิตอลนี้จะให้ประโยชน์มากมาย แต่ก็มีปัญหาสำคัญเกิดขึ้น คือ สัญญาณดิจิตอล วีดีโอและสัญญาณเสียงดิจิตอลที่มีความถี่สูงจะต้องทำการสุ่มสัญญาณที่มีความถี่การสุ่มสูง จะได้ข้อมูลที่มีขนาดใหญ่มาก นั่นคือจะไม่เหมาะแก่การส่งผ่านสัญญาณที่มีระยะทางไกล ดังนั้นจึงต้องมีการบีบอัดขนาดข้อมูล

เทคโนโลยีเอ็มเป็กเลเยอร์3 เป็นเทคโนโลยีที่บีบอัดสัญญาณเสียงที่มีความแพร่หลาย และมีประสิทธิภาพในการบีบอัดสูง ดังนั้นการทำวงจรถอดรหัสเอ็มเป็กเลเยอร์3 จะช่วยให้เครื่องเล่นเพลงจากไฟล์เอ็มเป็กเลเยอร์3 มีประสิทธิ์ภาพที่สูงขึ้น

1.2 วัตถุประสงค์ของปริญญานิพนธ์

1. ศึกษาหลักการเข้ารหัสและการถอดรหัสข้อมูลแบบเอ็มเป็กเลเยอร์3
2. เขียนโปรแกรมถอดรหัสข้อมูลเอ็มเป็กเลเยอร์3 ด้วยโปรแกรมแมทแลบได้
3. ออกแบบวงจรรวมถอดรหัสข้อมูลเอ็มเป็กเลเยอร์3 ด้วยภาษาวีเอชดีแอลได้ในบางส่วน โดยแสดงการทำงาน ของวงจรด้วยโปรแกรมโมเดลซิมหรือโปรแกรมวีซิสเต็ม
4. ประเมินผลและหาความเป็นไปได้ที่จะสังเคราะห์วงจรนี้เป็นชิปวงจรรวมถอดรหัสเพื่อใช้งานจริง

1.3 ขอบเขตของปริญญานิพนธ์

เป็นการศึกษาการถอดรหัสไฟล์เอ็มเป็กเลเยอร์ 3 โดยการออกแบบด้วยโปรแกรมแมทแลบเพื่อทำความเข้าใจในการทำงานการถอดรหัสเอ็มเป็กเลเยอร์ 3 และสามารถเขียนภาษาวีเอชดีแอลเพื่อใช้ในการสังเคราะห์การทำงานของวงจรได้ในบางส่วน เพื่อที่จะสามารถใช้โปรแกรม

โมเดลซิมหรือโปรแกรมวีซิสเต็มช่วยในการจำลองการทำงานของวงจร และประเมินความเป็นไปได้ที่จะสามารถนำวงจรนี้ไปสังเคราะห์เป็นชิปวงจรรวมถอดรหัสเอ็มเป็กเลเยอร์ 3 เพื่อนำไปใช้งานจริง

1.4 แผนการดำเนินงาน

อธิบายแผนการดำเนินงานเป็นขั้นตอนดังต่อไปนี้

1. หาข้อมูลจากแหล่งข้อมูลต่างๆ เช่น อินเตอร์เน็ต , สำนักงานมาตรฐานผลิตภัณฑ์อุตสาหกรรมแห่งชาติ , ห้องสมุด , ปริญญานิพนธ์และหนังสือต่างๆ เป็นต้น
2. ศึกษาการทำงานของการเข้ารหัส , การถอดรหัสของเอ็มเป็กเลเยอร์ 3 และการประมวลผลสัญญาณ
3. ศึกษาการใช้งานโปรแกรมแมทแลป การเขียน เอ็ม-ไฟล์ การประมวลผลโปรแกรม
4. เขียนโปรแกรมถอดรหัสเอ็มเป็กเลเยอร์ 3 โดยใช้โปรแกรมแมทแลบ เพื่อทำความเข้าใจการทำงานของการถอดรหัส และเพื่อนำผลที่ได้จากการถอดรหัสนี้ไปเปรียบเทียบกับผลลัพธ์ที่ได้จากการถอดรหัสด้วยวงจรรวมที่ออกแบบด้วยภาษาวีเอชดีแอล ในการเขียนโปรแกรมนี้จะเริ่มเขียนเป็นการทำงานแต่ละส่วน แล้วนำทุกส่วนมาเชื่อมต่อกัน
5. ตรวจสอบผลลัพธ์ที่ได้จากโปรแกรมที่เขียนในแมทแลป แก้ไขและปรับปรุงจนโปรแกรมสามารถทำงานได้ถูกต้องตามต้องการ
6. ศึกษาการออกแบบวงจรรวมทางดิจิตอลด้วยภาษาวีเอชดีแอล และการใช้งานเบื้องต้นของโปรแกรมโมเดลซิม หรือโปรแกรมวีซิสเต็มที่อาจนำมาช่วยในการจำลองการทำงานของวงจรได้
7. ออกแบบวงจรรวมถอดรหัสเอ็มเป็กเลเยอร์ 3 โดยการเขียนภาษาวีเอชดีแอลได้ในบางส่วน โดยแบ่งเป็นวงจรย่อยส่วนต่างๆ โดยอาจจะทดสอบการทำงานโดยใช้โปรแกรมโมเดลซิม หรือโปรแกรมวีซิสเต็ม เพื่อตรวจสอบความถูกต้องในการทำงาน
8. สรุปและประเมินผลหาความเป็นไปได้ในการการสังเคราห์วงจรรวมถอดรหัสจากไฟล์เอ็มเป็กเลเยอร์ 3 เพื่อนำไปใช้งานจริง

1.5 ประโยชน์และผลที่ได้รับจากการดำเนินงาน

1. ทราบถึงหลักการเข้ารหัสข้อมูลเสียงแบบเอ็มเป็กเลเยอร์ 3
2. ทราบถึงหลักการและเทคนิคในการเขียนโปรแกรมด้วยแมทแลป และสามารถใช้โปรแกรมแมทแลบช่วยในการออกแบบได้

3. ทราบถึงหลักการออกแบบวงจรรวมด้วยการใช้ภาษาวีเอชดีแอล โดยใช้โปรแกรมโมเดลซิมหรือโปรแกรมวีซิสเต็ม ช่วยในการจำลองการทำงานของวงจร
4. สามารถนำผลการดำเนินการ ไปใช้ในการสังเคราะห์วงจรเพื่อเป็นชิปวงจรรวมถอดรหัสเอ็มเป็กเลเยอร์ 3 ได้ในอนาคต

บทที่ 2
การบีบอัดข้อมูล
(Data Compression)

2.1 ทำไมจึงต้องบีบอัดข้อมูล

ในสมัยอดีตนั้นหน่วยความจำมีราคาแพงมาก ทำให้ต้องประหยัดเนื้อที่หน่วยความจำเป็นอย่างมาก จึงเกิดกระบวนการที่ทำให้ข้อมูลมีขนาดเล็กลง กระบวนการนั้นเรียกว่า "การบีบอัด" (Compression) การบีบอัดจะเป็นการลดอัตราการส่งข้อมูล (Bit Rate) และลดขนาดของข้อมูลลง แต่สามารถจะขยายกลับเป็นขนาดเดิม และมีรายละเอียดของข้อมูลครบถ้วนเหมือนตอนก่อนที่จะบีบอัด

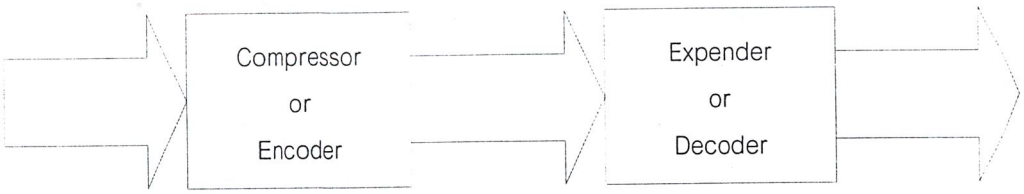
ในปัจจุบันหน่วยความจำมีราคาถูกลงมาก แต่การบีบอัดกลับมีความสำคัญมากขึ้นกว่าในสมัยอดีต เพราะขนาดข้อมูลก็มีขนาดใหญ่ขึ้นกว่าแต่ก่อนมากเช่นเดียวกัน การที่จะเคลื่อนย้ายข้อมูลจะทำได้ลำบากขึ้น เพราะต้องใช้หน่วยความจำขนาดใหญ่ตามไปด้วย และในงานบางประเภทหากไม่ใช้การบีบอัดเข้าช่วยก็จะไม่สามารถใช้งานได้

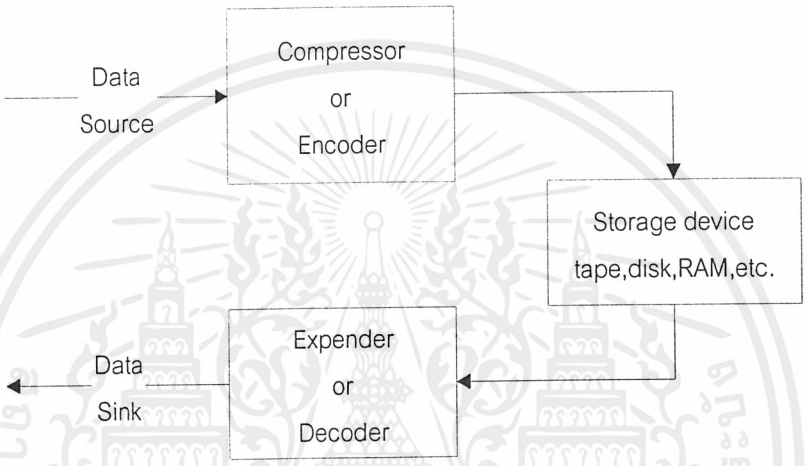
จะเห็นได้ว่า "การบีบอัด" (Compression) เป็นกระบวนการที่มีความสำคัญมาก ดังนั้นจึงสรุปได้ว่าเราใช้ "การบีบอัด" เพื่อ

- ลดพื้นที่ในการเก็บข้อมูล ช่วยให้ประหยัดพื้นที่ในการเก็บข้อมูล
- ช่วยให้บางกระบวนการที่เคยทำไม่ได้ให้ทำได้

2.2 การประยุกต์การบีบอัด

การบีบอัดแสดงไว้ดังรูปที่ 2.1 แสดงการไหลของข้อมูลจากอินพุทไปที่เอาท์พุท จากรูปที่ 2.1(a) ปริมาณของข้อมูลถูกลดลงโดยตัวบีบอัด (Compressor) หรือโดยตัวเข้ารหัส (Encoder)จากนั้นข้อมูลที่ถูกบีบอัดแล้วจะผ่านไปตางช่องส่ง (Transmission Channel) และข้อมูลที่ถูกบีบอัดจะถูกขยายออกเป็นข้อมูลเดิมก่อนถูกเข้ารหัสโดยตัวขยาย (Expender) หรือตัวถอดรหัส (Decoder) แต่กรณีนี้มีข้อเสียคือ ข้อมูลจะถูกบีบอัดและขยายเลย ถ้าต้องการใช้งานอีกก็ต้องบีบอัดและทำการขยายใหม่ ดังนั้นจึงใช้หน่วยความจำ เช่น เทป (Tape), ดิสก์ (Disk), แรม (RAM) เป็นต้น เก็บข้อมูลที่ถูกเข้ารหัสแล้วจากตัวเข้ารหัส ถ้าต้องการนำไปใช้งานก็จึงนำข้อมูลที่ถูกเข้ารหัสนั้นมาถอดรหัสแล้วจึงนำไปใช้งาน โดยที่ข้อมูลที่ถูกเข้ารหัสแล้วก็ยังคงถูกเก็บอยู่ในหน่วยความจำ ดังนั้นถ้าต้องการใช้งานอีกก็สามารถนำข้อมูลที่ถูกเข้ารหัสแล้วนี้ไปถอดรหัสได้เลย โดยไม่ต้องทำการเข้ารหัสอีกดังรูปที่ 2.1(b) แสดงตำแหน่งของหน่วยความจำ





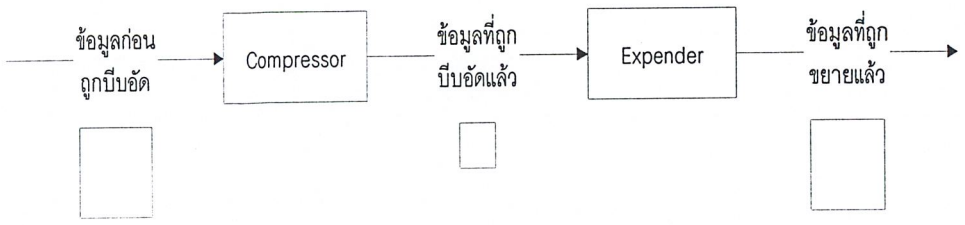
รูปที่ 2.1 แสดงการไหลของข้อมูลจากอินพุทไปที่เอาท์พุท

2.3 รูปแบบในการบีบอัด

การบีบอัดนั้นจะมีอยู่ด้วยกัน 2 ประเภทคือ

- แบบไม่มีการสูญเสีย (Lossless)
- แบบมีการสูญเสีย (Lossy)

2.3.1 แบบไม่มีการสูญเสีย (Lossless)



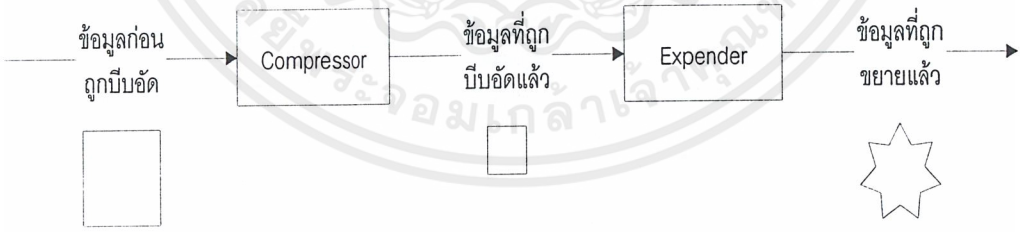
รูปที่ 2.2 แสดงข้อมูลก่อนการบีบอัดและข้อมูลหลังการขยาย

เป็นการบีบอัดที่ข้อมูลก่อนถูกบีบอัดกับข้อมูลเดียวกันที่ถูกขยายออกมา จะมีลักษณะเหมือนกันทุกประการ ไม่มีส่วนขาดส่วนเกิน ก็คือในกระบวนการบีบอัดจะไม่มีการตัดข้อมูลออกนั่นเอง ซึ่งแสดงได้ดังรูปที่ 2.2 แสดงข้อมูลก่อนการบีบอัดและข้อมูลหลังการขยาย

2.3.2 แบบมีการสูญเสีย (Lossy)

การบีบอัดแบบนี้ ข้อมูลจะถูกตัดออกบางส่วนในระหว่างการบีบอัด ซึ่งมีผลทำให้ข้อมูลหลังการขยายจะผิดเพี้ยนไปจากเดิม ดังรูปที่ 2.3 แสดงการบีบอัดแบบมีการสูญเสีย (Lossy) วิธีการนี้จะสามารถบีบอัดข้อมูลได้มากกว่าแบบแรก (แบบ Lossless) การบีบอัดแบบนี้มักจะใช้กับข้อมูลเสียงและข้อมูลภาพ

- สำหรับข้อมูลเสียงเราจะสนใจเพียงเสียงที่ได้ยินเท่านั้น โดยไม่สนใจว่าข้อมูลจะถูกตัดออกไปหรือไม่ ก็คือข้อมูลเสียงจะถูกตัดเสียงที่มนุษย์ไม่ได้ยินเมื่อเทียบกับเสียงอื่นออกและบีบอัดข้อมูล เมื่อทำการขยายออกมาข้อมูลที่ได้จะผิดเพี้ยนจากเดิม แต่คุณภาพเสียงเหมือนเดิม (แยกไม่ออกว่าเสียงไหนเป็นเสียงจริง เสียงไหนเป็นเสียงที่ถูกตัดบางส่วนออกแล้ว)
- สำหรับข้อมูลภาพเราจะสนใจเพียงภาพที่มองเห็นเท่านั้น โดยไม่สนใจว่าข้อมูลจะถูกตัดออกไปหรือไม่ ก็คือข้อมูลภาพจะตัดสีที่เหลี่ยมกันเล็กน้อยจนมนุษย์ไม่สามารถมองเห็นได้ออกจากข้อมูลเดิมและบีบอัดข้อมูล เมื่อขยายออกมาก็จะภาพคุณภาพไม่ต่างจากภาพจริงเลย (มนุษย์แยกไม่ออก)



รูปที่ 2.3 แสดงการบีบอัดแบบมีการสูญเสีย(Lossy)

บทที่ 3
การบีบอัดข้อมูลเสียงแบบเอ็มเป็ก
(MPEG Audio Compression)

3.1 บทนำ

การบีบอัดข้อมูลเสียงแบบเอ็มเป็ก (Mpeg Audio Compression) เป็นวิธีการบีบอัดข้อมูลเสียงแบบดิจิตอล (Digital Compression) ที่มีความเหมือนจริงสูง (High Fidelity : HiFi) การบีบอัดข้อมูลแบบเอ็มเป็ก สำเร็จลงในปี 1993 โดยคณะกรรมการสากลแห่งการเชี่ยวชาญการบีบอัดเสียงเหมือนจริง ซึ่งรู้จักกันดีในชื่อว่า Motion Picture Expert Group (MPEG) และได้รับมาตรฐาน ISO/IEC 11172-3

3.2 หลักการพื้นฐานและการใช้งาน

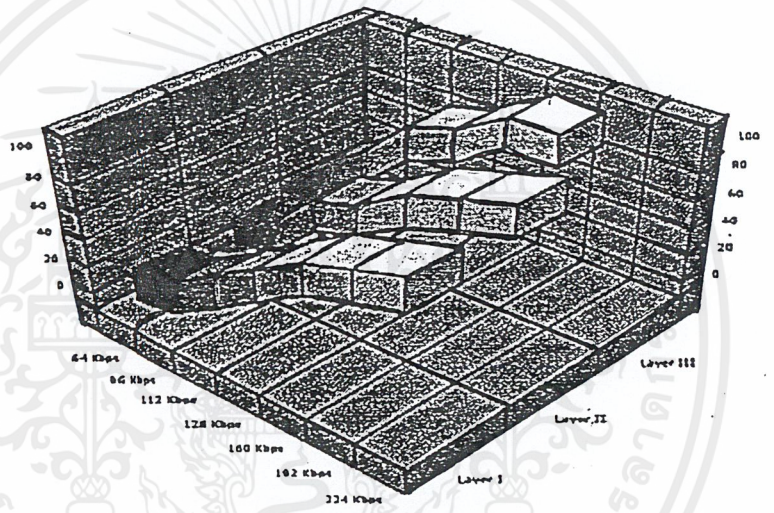
หลักการบีบอัดข้อมูลแบบเอ็มเป็ก จะใช้ประโยชน์จากขีดจำกัดในการได้ยินของมนุษย์โดยไม่จัดเก็บข้อมูลเสียงที่ที่มนุษย์ฟังไม่ได้ยิน เพื่อให้ข้อมูลมีขนาดเล็กลง ซึ่งใช้หลักการว่าถ้าเราได้ยินเสียง 2 เสียงที่มีความถี่ใกล้เคียงกัน โดยให้เสียงที่ 2 ดังกว่าเสียงที่ 1 มาก เมื่อเราฟัง 2 เสียงนี้พร้อมกันจะได้ยินเสียงที่ 2 (ดังกว่า) เพียงเสียงเดียว ดังนั้นในการจัดเก็บข้อมูลสามารถตัดข้อมูลเสียงที่ค่อยกว่าออกไปได้บางส่วน ซึ่งหลักการนี้จะถูกอธิบายในหัวข้อ "ไซโครอะคูสติก" (Phychoacoustic) อีกครั้ง การบีบอัดข้อมูลแบบเอ็มเป็กสามารถเลือกวิธีการบีบอัดเสียงได้หลาย ๆ แบบได้ดังนี้

- อัตราการสุ่มข้อมูล (Sampling rate) 32, 44.1 หรือ 48 กิโลเฮิร์ต (kHz)
- รองรับระบบเสียงได้ทั้ง 1 และ 2 ช่องเสียง ซึ่งมีอยู่ 4 แบบดังนี้
    1. แบบโมโนเดี่ยว (Monophonic mode)
    2. แบบโมโนคู่ (Dual Monophonic mode)
    3. แบบสเตอริโอ (Stereo mode)
    4. แบบจอย – สเตอริโอ (Joint Stereo mode)
- สามารถเลือกค่าอัตราการส่งข้อมูล (bit rate) ได้ตั้งแต่ 32 ถึง 224 Kbit/sec ต่อหนึ่งช่องเสียง ขึ้นอยู่กับอัตราการสุ่มตัวอย่าง (Sampling rate)
- รูปแบบของการเข้ารหัสไฟล์เอ็มเป็ก มี 3 เลเยอร์ (Layer) คือ เลเยอร์ 1 เลเยอร์ 2 และเลเยอร์ 3 แต่ละเลเยอร์มีลักษณะต่างกันดังนี้

- เลเยอร์ 1 มีความซับซ้อนน้อยที่สุด ต้องใช้อัตราการส่งข้อมูลสูงถึง 384 กิโลบิตต่อวินาที (Kbps) เพื่อที่จะให้ได้เสียงคุณภาพสูง

- เลเยอร์ 2 มีความซับซ้อนมากขึ้น คุณภาพเสียงสูงกว่าเลเยอร์ 1 ใช้อัตราการส่งข้อมูลที่ 160 กิโลบิตต่อวินาทีจะได้เสียงคุณภาพสูง และที่ 192 กิโลบิตต่อวินาทีจะได้คุณภาพเสียงไม่แตกต่างจากเสียงต้นแบบ
- เลเยอร์ 3 มีความซับซ้อนที่สุดแต่สามารถให้คุณภาพเสียงที่ดีที่สุด ใช้อัตราการส่งข้อมูลที่ 160 กิโลบิตต่อวินาที ให้คุณภาพเสียงไม่แตกต่างเสียงต้นแบบ

สามารถแสดงความสัมพันธ์ของอัตราการส่งข้อมูลเสียง (กิโลบิตต่อวินาที) สำหรับการเข้ารหัสเลเยอร์ต่าง ๆ กับคุณภาพเสียงเมื่อเทียบกับเสียงต้นแบบได้ดังรูปที่ 3.1



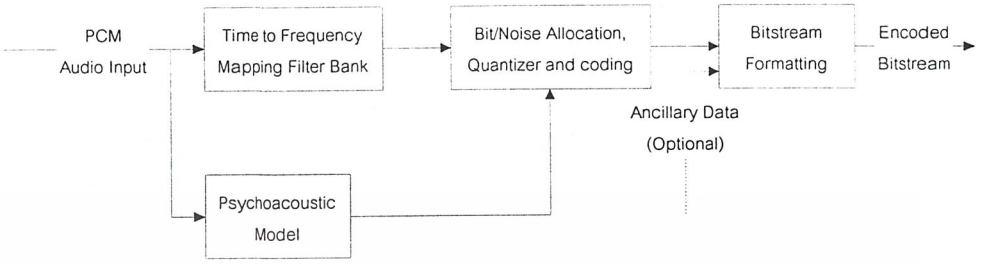
รูปที่ 3.1 กราฟเปรียบเทียบอัตราส่วนการบีบอัดเสียงของการเข้ารหัสเลเยอร์ต่างๆ เมื่อเทียบกับคุณภาพเสียงต้นแบบจากคอมแพ็คดิสก์ (CD)

3.3 การเข้ารหัสแบบเอ็มเป็กโดยทั่วไป

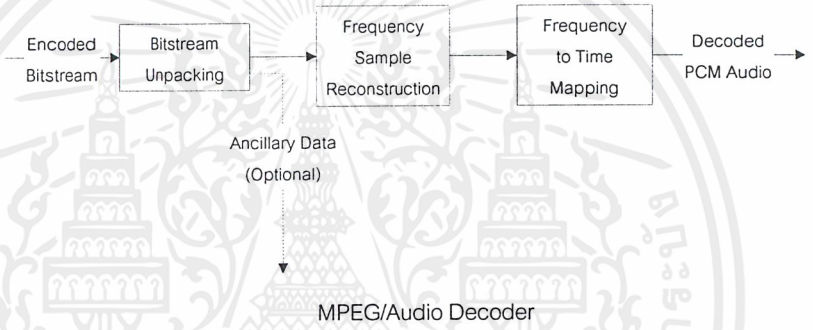
การเข้ารหัสแบบเอ็มเป็กโดยทั่วไป จะตัดข้อมูลเสียงที่จัดเก็บบางส่วนออกแต่สามารถคงรายละเอียดของเสียงที่ได้ยินไว้เท่าเดิม ดังรูปที่ 3.2 แสดงบล็อกไดอะแกรม (Block Diagram) ของการเข้ารหัสและถอดรหัสแบบเอ็มเป็ก กระบวนการเข้ารหัสมีขั้นตอนดังนี้

1. เริ่มจากข้อมูลเสียงจะถูกส่งไปยังโพลีเฟส ฟิลเตอร์แบงค์ (The Polyphase Filter bank) เพื่อแบ่งข้อมูลเสียงออกเป็นหลาย ๆ ช่วงความถี่ (Subband of Frequency)
2. ขณะเดียวกันข้อมูลเสียงก็จะผ่านไปยังไซโครอะคูสติก โมเดล (Psychoacoustic Model) ด้วยเพื่อหาสัดส่วนของพลังงานของสัญญาณต่อการกำหนดค่ามาสก์กิงเทรดโฮล (Masking threshold : SMR) ของแต่ละช่วงความถี่

3. ข้อมูลจากโพลีเฟส ฟิลเตอร์แบงค์ ถูกส่งไปยังส่วนแบ่งแยกบิต(Bit/Noise Allocation) และปรับข้อมูล (Quantizing)



MPEG/Audio Encoder



MPEG/Audio Decoder

รูปที่ 3.2 แสดงบล็อกไดอะแกรมของการเข้ารหัสและการถอดรหัสแบบเอ็มเป็ก

กระบวนการถอดรหัสบิตสตรีม (Bitstream) จะนำข้อมูลมาผ่านกระบวนการปรับข้อมูลย้อนกลับและทำการรวมข้อมูลแต่ละช่วงความถี่กลับเป็นข้อมูลเสียง

3.3.1 โพลีเฟส ฟิลเตอร์แบงค์ (The Polyphase Filter bank)

โพลีเฟส ฟิลเตอร์แบงค์ (The Polyphase Filter bank) เป็นหลักการทั่วไปในการเข้ารหัสแบบเอ็มเป็ก โพลีเฟส ฟิลเตอร์แบงค์ จะแบ่งสัญญาณเสียงออกเป็นช่วงความถี่ที่มีความกว้าง (bandwidth) ออกเป็น 32 ช่วงความถี่ย่อย (Subband) ที่เท่ากัน

คุณสมบัติของโพลีเฟส ฟิลเตอร์แบงค์

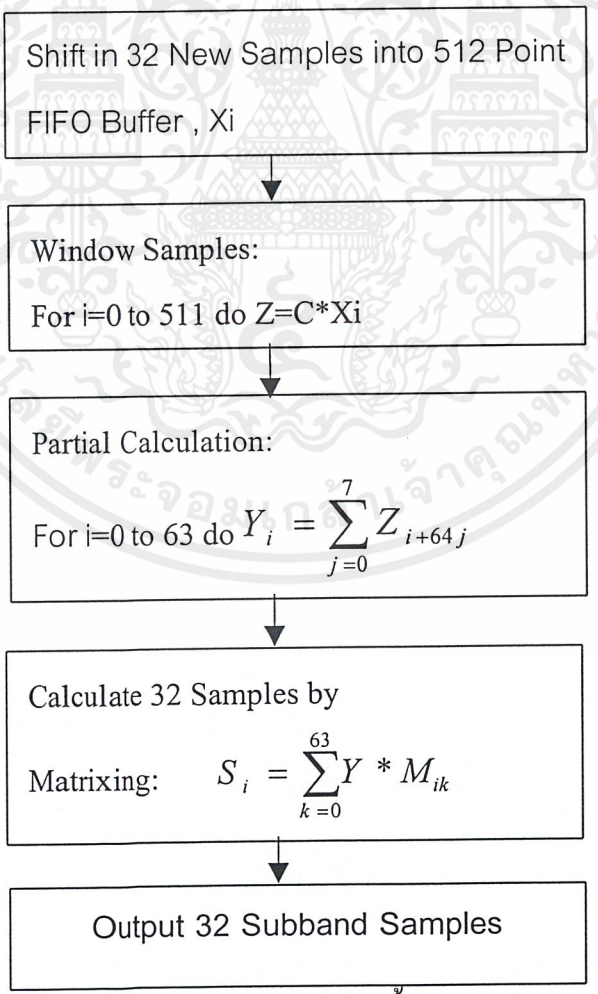
1. ความกว้างของช่วงความถี่ในแต่ละช่วงเท่ากัน
2. ฟิลเตอร์แบงค์ และการแปลงกลับเป็นการแปลงแบบมีการสูญเสีย (lossy)
3. ย่านความถี่แต่ละย่านจะมีการเหลื่อมซึ่งกันและกัน

ขั้นตอนการทำโพลีเฟส ฟิลเตอร์แบงค์

โพลีเฟส ฟิลเตอร์แบงค์ เป็นกระบวนการในการแยกสัญญาณที่ถูกสุ่มตัวอย่าง (Sampling) ด้วยความถี่สุ่มตัวอย่าง (F_s) ออกเป็น 32 ย่านความถี่ย่อย โดยแต่ละช่วงของย่านความถี่ย่อยเหล่านี้จะมีความถี่เท่ากันคือ $F_s/32$ ขั้นตอนการวิเคราะห์มีดังนี้

1. ข้อมูลเสียงเข้า 32 ข้อมูลสุ่มตัวอย่าง (Sample)
2. สร้างข้อมูลบัฟเฟอร์ (Buffer) ไว้สำหรับข้อมูลสุ่มตัวอย่าง 512 ข้อมูล โดยจะเลื่อนข้อมูลเข้า 32 ข้อมูลเข้ามาในบัฟเฟอร์ในตำแหน่งที่ 0-31
3. คูณค่าข้อมูลอินพุท 512 ค่ากับสัมประสิทธิ์ (c) 512 ค่า
4. คำนวณค่า ตามสมการที่ให้มา (64 ค่า)
5. คำนวณค่าสับแบนด์แซมเปิล (Subband Sample : S) 32 ค่าออกมา

อธิบายขั้นตอนการทำโพลีเฟส ฟิลเตอร์แบงค์ เป็นโฟว์ชารจ์ (Flowchart) ได้ดังนี้



รูปที่ 3.3 แสดงโฟล์วชาร์ต (Flowchart) อธิบายขั้นตอนการทำโพลีเฟส ฟิลเตอร์แบงค์

$$Si[i] = \sum_{k=0}^{63} \sum_{j=0}^{7} M[i][k] \times (C[k + 64j] \times x[k + 64j]) \qquad \text{สมการที่ 3.1}$$

โดย

i คือ ดรรชนีย่านความถี่ ตั้งแต่ 0 – 31

s_t คือ สัญญาณเอาท์พุทที่ถูกกรองความถี่แล้ว สำหรับย่านความถี่ i ที่เวลา t โดย t เป็นเลขจำนวนเต็ม

C[n] คือ สัมประสิทธิ์ของวินโดว์ ลำดับที่ n ถูกนิยามไว้ในมาตรฐาน

X[n] คือ สัญญาณเสียงอินพุทที่ถูกอ่านจากอินพุทบัฟเฟอร์ที่เก็บอินพุทไว้ 512 สัญญาณความถี่

$$M[i][k] = \cos \frac{(2 \times i + 1) \times (k - 16) \times \pi}{64} \qquad \text{สมการที่ 3.2}$$

สมการที่ 3.1 เป็นการคำนวณที่สามารถลดจำนวนครั้งของการคูณได้มากที่สุด เนื่องจากพจน์ที่อยู่ในวงเล็บไม่เป็นฟังก์ชั่นของ I และ M[I][k] ไม่ขึ้นกับ j ดังนั้นการคำนวณทุก ๆ 32 สัญญาณสุ่มตัวอย่างเอาท์พุท จะคูณเลขเพียง $512 + 32 \times 64 = 2560$ ครั้ง และการบวก $64 \times 7 + 32 \times 63 = 2464$ ครั้ง หรือประมาณ 80 ครั้งต่อสัญญาณเอาท์พุท 1 สัญญาณ ข้อสังเกตจากการทำฟิลเตอร์แบงค์ ทุก ๆ 32 สัญญาณอินพุทเราจะได้สัญญาณเอาท์พุท 32 สัญญาณ ด้วยเหตุนี้ทั้งในแต่ละย่านความถี่ย่อย (Subband) จึงถูกแบ่งให้มีเพียงเอาท์พุทเดียว ทั้ง 32 ย่านเมื่ออินพุทเข้ามา 32 สัญญาณ

จากสมการที่ 3.1 สามารถเขียนเป็นสมการพื้นฐานในรูปแบบผลบวกประสาน (Convolution) ได้คือ

$$Si[i] = \sum_{n=0}^{511} x[t - n] \times Hi[n] \qquad \text{สมการที่ 3.3}$$

$$Hi[n] = h[n] \times \cos\left[\frac{(2 \times i) \times (n - 16) \times \pi}{64}\right] \qquad \text{สมการที่ 3.4}$$

โดย x[τ] คือ สัญญาณเสียงสุ่มความถี่

h[n] = -c[n] เมื่อ n/64 เป็นเลขคี่

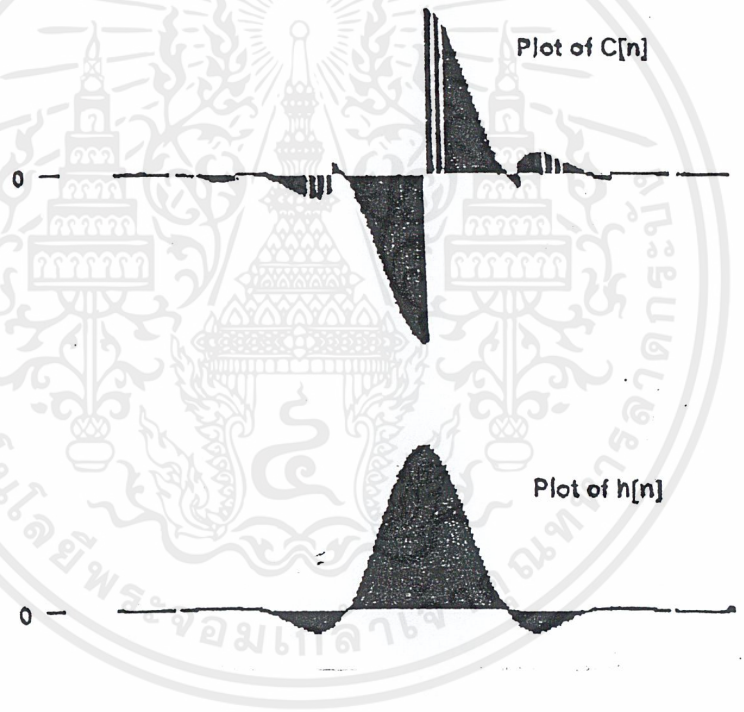
= c[n] กรณีอื่น

ในรูปแบบนี้แต่ละย่านความถี่ย่อยจะมีผลตอบสนองความถี่ของตนเองคือ Hi[n] แม้ว่ารูปแบบนี้จะง่ายต่อการวิเคราะห์ แต่จะมีจำนวนครั้งของการคูณและบวกในการคำนวณมากกว่า

สมการที่ 3.1 คือจะคูณเลข 512 ×32 = 16384 ครั้ง และการบวก 512 ×32 = 16384 ครั้ง เพื่อที่คำนวณสัญญาณเอาท์พุท 32 สัญญาณ

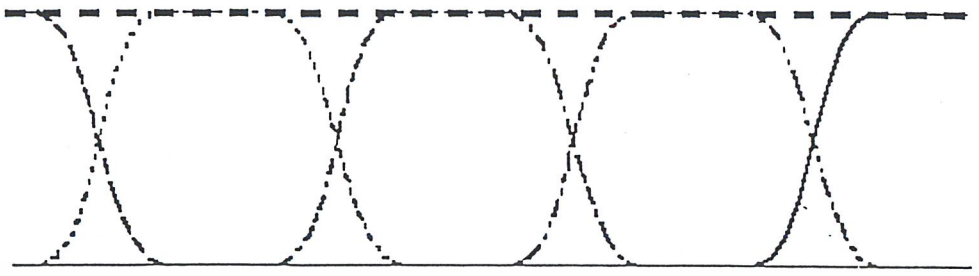
ค่าสัมประสิทธิ์ h[n] เป็นผลตอบสนองตัวกรองความถี่ต่ำสำหรับโพลีเฟส ฟิลเตอร์แบงค์ รูปที่ 3.4 แสดงสัมประสิทธิ์ C[n] และ h[n]

สมการสำหรับ H[n] แสดงให้เห็นว่าค่า H[n] ได้มาจาก h[n] คูณกับเทอมโคไซน์ (Cosine) เพื่อที่จะเลื่อนเฟสของผลตอบสนองตัวกรองความถี่ต่ำ (h[n] ให้เหมาะสมกับย่านความถี่ ดังนั้นจึงเรียกการกรองความถี่แบบนี้ว่า "โพลีเฟส" (Polyphase) การกรองความถี่แบบนี้จะมีความถี่กลางอยู่ที่ค่าเลขคี่คูณด้วย $\pi/(64T)$ ซึ่ง T เป็นคาบของการสุ่มตัวอย่าง (sampling period)



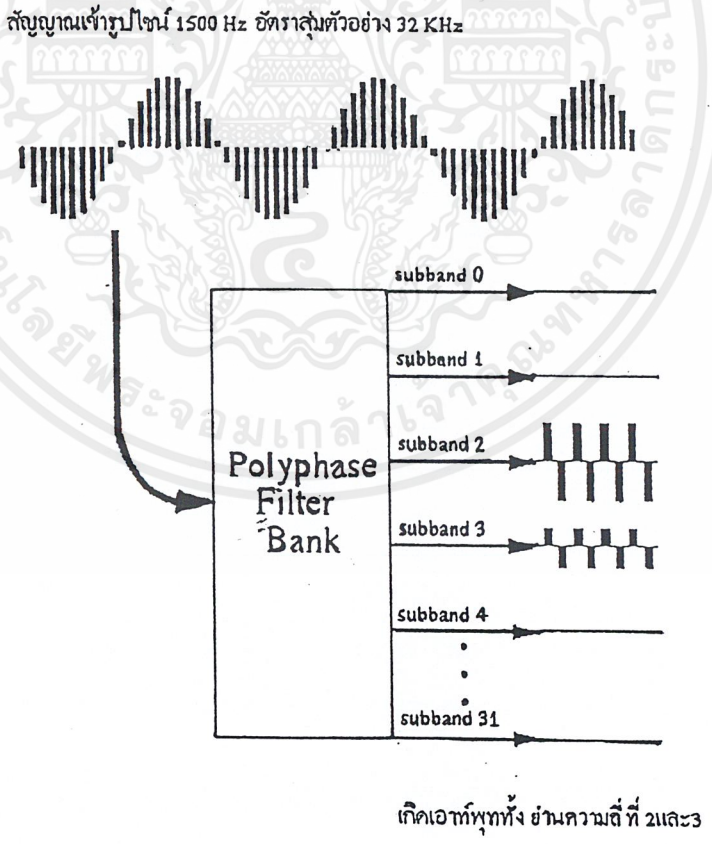
รูปที่ 3.4 แสดงการเปรียบเทียบสัมประสิทธิ์ C[n] และ h[n]

เนื่องจากผลตอบสนองตัวกรองความถี่ต่ำ ไม่มีความคมในการคัทออฟ (แถบนำกว้าง) ดังนั้นเมื่อแบ่งช่วงความถี่ที่ใช้งานทั้งหมดเป็น 32 ช่องจะเกิดการเหลื่อมซึ่งกันและกัน ดังแสดงตัวอย่างในรูปที่ 3.5



รูปที่ 3.5 แสดงการเหลื่อมซึ่งกันและกันระหว่างย่านความถี่ย่อยที่ติดกัน

ซึ่งผลของการเหลื่อมกันของกันของย่านความถี่ที่ติดกัน จะทำให้ประสิทธิภาพของการบีบอัดข้อมูลลดลงเนื่องจากพลังงานของสัญญาณที่มีความถี่ใกล้ ๆ กันของของย่านความถี่หนึ่ง ๆ จะปรากฏ เป็นเอาท์พุทของโพลีเฟส ฟิลเตอร์แบงค์ สองเอาท์พุทที่อยู่ติดกัน ดังรูปที่ 3.6 แสดงตัวอย่างของสัญญาณรูปไซน์หนึ่งความถี่ เมื่อผ่านกระบวนการโพลิเฟส ฟิลเตอร์แบงค์ แล้วปรากฏเอาท์พุทออกมาสองย่านความถี่



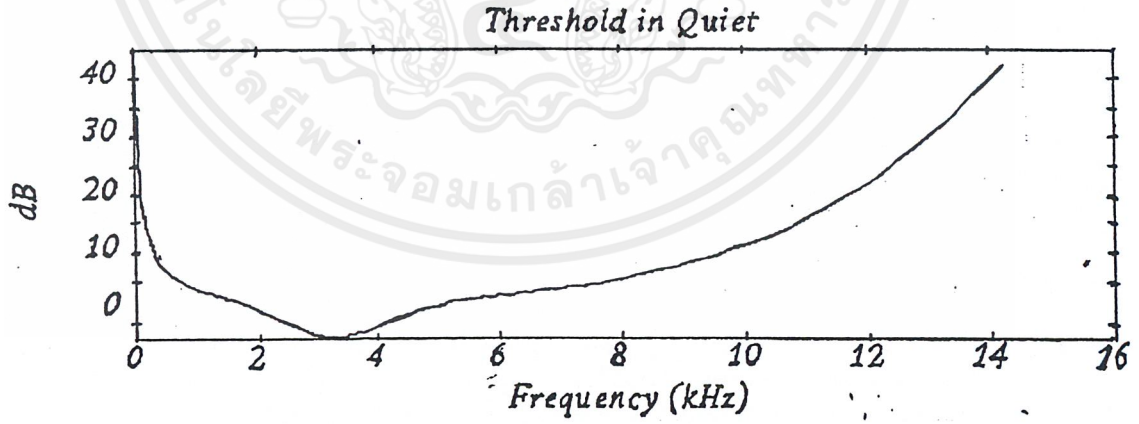
รูปที่ 3.6 แสดงการเกิดเอาท์พุทของโพลีเฟส ฟิลเตอร์แบงค์
เมื่อสัญญาณเข้าเป็นสัญญาณไซน์หนึ่งความถี่

3.3.2 ไซโครอะคูสติก โมเดล (Psychoacoustics Model)

การเข้ารหัสข้อมูลเสียงแบบเอ็มเป็ก ใช้หลักการตัดเสียงบางส่วนที่ฟังไม่ได้ยินออก เนื่องจากความได้เปรียบในการรับฟังเสียงของมนุษย์ที่ไม่สามารถได้ยินเสียงรบกวน (noise) ภายใต้เงื่อนไขการปิดกั้นการได้ยิน (Auditory masking) การปิดกั้นการได้ยินเป็นคุณสมบัติอย่างหนึ่งในการรับฟังเสียงของมนุษย์ เกิดขึ้นเมื่อเรารับฟังเสียงที่จากแหล่งกำเนิดสองแหล่ง จะไม่สามารถได้ยินเสียงจากแหล่งกำเนิดที่ค่อยกว่า ถ้าแหล่งกำเนินทั้งสองมีความถี่ใกล้เคียงกัน

ความสามารถในการได้ยินเสียงของมนุษย์

1. ช่วงความถี่ที่สามารถรับฟัง ประมาณ 20 – 20000 เฮิร์ต
2. ไดนามิคเรนจ์ (dynamic range) คือเสียงค่อยสุดถึงดังสุด ที่สามารถรับฟังได้ ประมาณ 96 เดซิเบล
3. ความถี่เสียงพูดปกติ 500 – 2000 เฮิร์ต
4. ความไวในการรับฟังเสียงของมนุษย์ จากการทดลองให้คน 1 คนเข้าไปนั่งในห้องที่เงียบสนิท แล้วเพิ่มความค่อย ๆ ดังเสียง จนกระทั่งคน ๆ นั้นเริ่มได้ยิน (Threshold level) วัดระดับความดังเสียง แปรความถี่แล้วนำค่ามาวาดกราฟ แสดงได้ดังรูปที่ 3.7 พบว่ามนุษย์สามารถรับเสียงช่วงความถี่ 2 – 4 กิโลเฮิร์ตได้ไวที่สุด

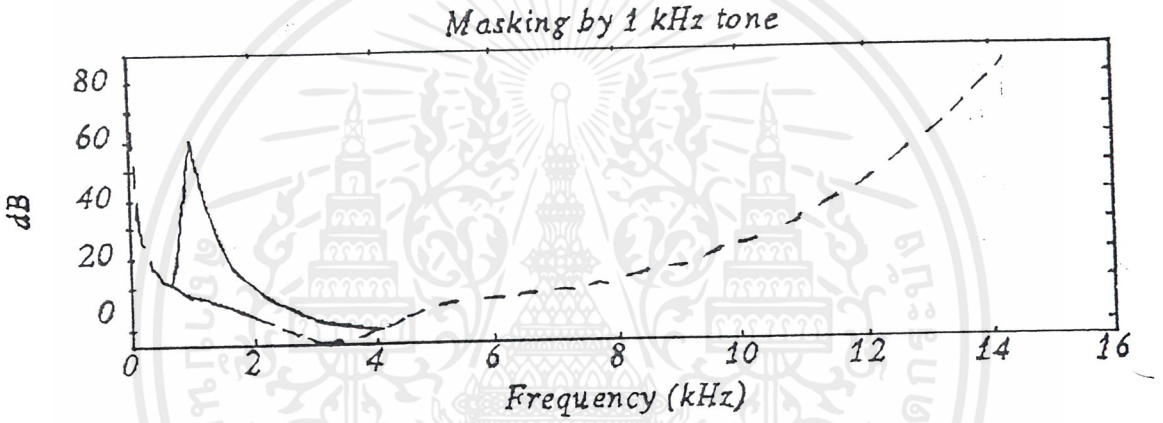


รูปที่ 3.7 แสดงระดับความดังเสียงที่เริ่มได้ยินที่ความถี่ต่าง ๆ

5. การปิดกั้นเสียงเมื่อมีสัญญาณเสียงหลาย ๆ สัญญาณที่มีความถี่ใกล้เคียงกันเข้ามา มนุษย์สามารถได้ยินสัญญาณที่มีความดัง (แรง) กว่าเท่านั้นหรือกล่าวได้ว่าสัญญาณที่แรงกว่าปิดกั้นสัญญาณที่อ่อนกว่า และเรียกสัญญาณที่แรงกว่านั้นว่า ตัวปิดกั้น (MASKER)

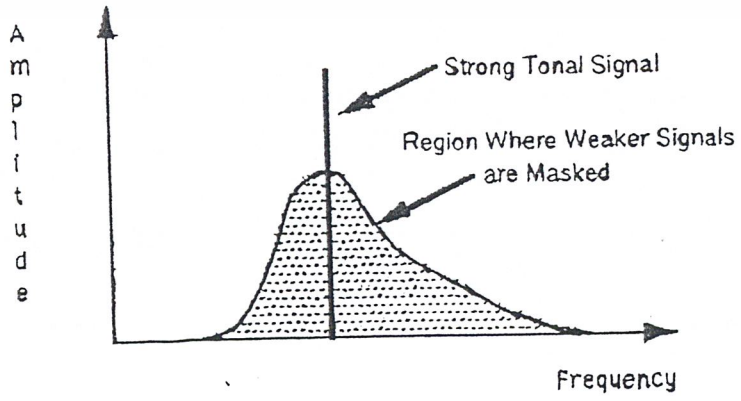
จากการทดลองเรื่องการปิดกั้นเสียง โดยใช้ให้คนฟังเสียงในห้องที่มีเสียงที่ความถี่ 1 กิโลเฮิร์ต ความดัง 60 เดซิเบล แล้วเล่นเสียงจากแหล่งกำเนิดอีกแหล่ง วัดระดับเสียงที่ได้ยินจากแหล่งกำเนิดที่สอง แปรความถี่ วาดกราฟ ได้ดังรูปที่ 3.8 ซึ่งแสดง ให้เห็นว่าที่ความถี่ใกล้กับความถี่ 1 กิโลเฮิร์ต ระดับความดังเสียงของแหล่งกำเนิดที่สองต้องมีค่าสูง เพื่อให้รับฟังเสียงได้

สามารถสรุปจากข้อมูลทั้งหมดได้ว่า มนุษย์สามารถรับฟังเสียงเป็นช่วงความถี่ และรับพลังงานเสียงได้ไม่เท่ากันในแต่ละย่านความถี่ ซึ่งเราเรียกว่าย่านความถี่วิกฤติ (critical band) และในแต่ละย่านความถี่มนุษย์จะแยกสัญญาณจากแหล่งกำเนิดต่าง ๆ ออกจากกันได้ยาก ตารางที่ 3.1 แสดงความกว้างของย่านความถี่ย่านต่าง ๆ



รูปที่ 3.8 แสดงผลของการปิดกั้นเสียง

เนื่องด้วยเหตุผลที่กล่าวมาทั้งหมดข้างต้น สามารถปิดกั้นเสียงที่ไม่ได้ยิน (Noise Masking) ภายในย่านความถี่หนึ่ง ๆ ได้ดังรูปที่ 3.9 หลักการนี้การเข้ารหัสแบบเอ็มเป็กยึดถือเป็นสำคัญ โดยแบ่งสัญญาณเสียงออกเป็นย่านความถี่ย่อยประมาณเท่ากับย่านความถี่วิกฤติแล้วจึงปรับข้อมูล (Quantize) ในแต่ละย่านความถี่ตามความสามารถในการได้ยินในแต่ละย่านความถี่



รูปที่ 3.9 แสดงการปิดกั้นเสียงที่ไม่ได้ยินในย่านความถี่ย่อยหนึ่ง ๆ ของการเข้ารหัสเอ็มเป็ก

ตารางที่ 3.1 แสดงความกว้างของย่านความถี่วิกฤติย่านต่าง ๆ

Band Number	Frequency (Hz)	Band Number	Frequency (Hz)
0	50	14	1970
1	95	15	2340
2	140	16	2720
3	235	17	3280
4	330	18	3840
5	420	19	4690
6	560	20	5440
7	660	21	6375
8	800	22	7690
9	940	23	9375
10	1125	24	11625
11	1265	25	15375
12	1500	26	20250
13	1735		

ไซโครอะคูรูสติก โมเดล (Phychoacoustic Model)

ไซโครอะคูรูสติก โมเดล (Phychoacoustic Model) ทำหน้าที่วิเคราะห์หาจุด (ความถี่) ที่ต้องปิดกั้นในย่านความถี่หนึ่ง ตัวเข้ารหัสจะใช้ข้อมูลนี้ในการตัดสินใจว่า ควรแทนสัญญาณอินพุทที่เข้ามาด้วยจำนวนบิตมากน้อยเพียงใด

3.3.3 การแบ่งแยกบิต (Bit or Noise allocation)

การแบ่งแยกบิตทำหน้าที่หาจำนวนของบิตเพื่อที่จะแยกในแต่ละย่านความถี่ย่อยขึ้นอยู่กับข้อมูลที่ได้มาจากไซโครอะคูรูสติก โมเดล

การแบ่งแยกบิตของการเข้ารหัสเอ็มเป็กเลเยอร์ 3 มีรายละเอียดดังนี้

1. ปรับข้อมูลให้เป็นค่า (quantize)
2. นับจำนวนบิตของการเข้ารหัสแบบฮัฟแมน (Huffman Coding)

การแบ่งแยกบิตของการเข้ารหัสเอ็มเป็กเลเยอร์ 3 มีรายละเอียดดังนี้

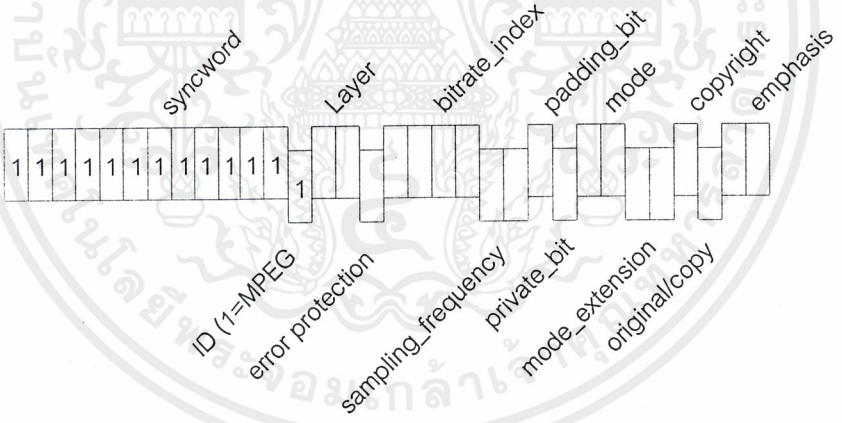
1. ปรับข้อมูลให้เป็นค่า (quantize)
2. นับจำนวนบิตของการเข้ารหัสแบบฮัฟแมน (Huffman Coding)
3. คำนวณผลของเสียงที่ตัดออกจริง ๆ (Actual calculates the resulting noise)
4. ทำขั้นตอนที่ 1 – 3 ซ้ำ เพื่อปรับค่าให้อยู่ในระดับที่ยอมรับได้ (Iterative Loop)

3.3.4 การเข้ารูปแบบ (Formatting)

เป็นขั้นตอนการจัดรูปแบบข้อมูลให้ตรงตามมาตรฐาน ซึ่งมาตรฐานการจัดเรียงข้อมูลแบบเอ็มเป็กของเลเยอร์ 3 แสดงได้ดังรูปที่ 3.10

หัวข้อมูล (Header)	ตรวจสอบความผิดพลาด (CRC)	ข้อมูลข้างเคียง (Side Information)	ข้อมูลหลัก (Main Data)

รูปที่ 3.10 แสดงรูปแบบข้อมูลเอ็มเป็กเลเยอร์ 3



รูปที่ 3.11 แสดงบิตต่าง ๆ ในส่วนหัวข้อมูล

1. ส่วนหัวข้อมูล (Header) เป็นข้อมูลขนาด 32 บิตแสดงลักษณะทั่วไปของไฟล์นั้น ๆ ข้อมูลในส่วนนี้ประกอบด้วยบิตต่าง ๆ ดังรูป 3.11
2. ส่วนตรวจสอบความผิดพลาด เป็นข้อมูลขนาด 16 บิต ตรวจสอบพาริตี้ ของข้อมูลที่ถูกเข้ารหัส
3. ส่วนข้อมูลข้างเคียงเป็นข้อมูลขนาด 17 หรือ 32 ไบต์ (17 ไบต์ สำหรับการเข้ารหัสแบบโมโนเดี่ยว (Monophonic mode) 32 ไบต์สำหรับแบบอื่น ๆ) ข้อมูลในส่วนนี้เก็บองค์ประกอบต่าง ๆ ที่ใช้ในการถอดรหัสโดยตรง

4. ข้อมูลหลัก จะไม่ถูกจำกัดความยาวข้อมูล ขึ้นอยู่กับความถี่สุ่มตัวอย่าง และอัตราการส่งข้อมูล ดังสมการ

$$N = 144 \times \frac{bitrate}{sampling\ \_frequency}$$ สมการที่ 3.5

เมื่อ N คือ ความยาวข้อมูลระหว่างสอง syncword ที่อยู่ติดกัน

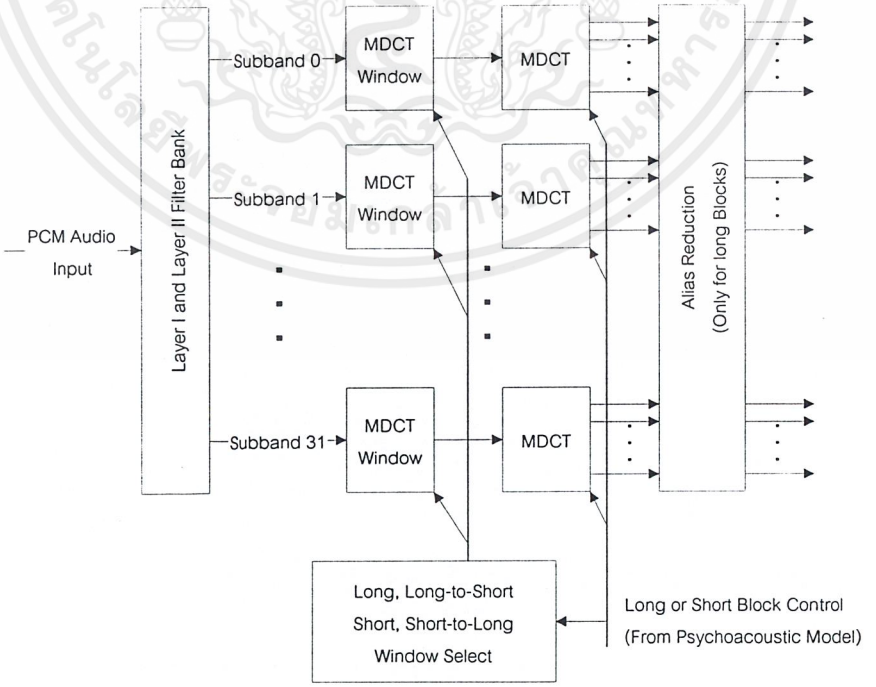
Birate คือ อัตราการส่งข้อมูล

Sampling_frequency คือ ความถี่สุ่มตัวอย่าง

3.4 การเข้ารหัสข้อมูลแบบเอ็มเป็กเลเยอร์ 3

การเข้ารหัสข้อมูลแบบเอ็มเป็กเลเยอร์ 3 มีหลักการเหมือนการเข้ารหัสแบบเอ็มเป็กโดยทั่วไปดังที่กล่าวมาในหัวข้อ 3.3 มีเพียงบางเทคนิคที่เพิ่มเติมเข้ามาเพื่อให้การบีบข้อมูลมีประสิทธิภาพสูงขึ้นดังนี้

1. ใช้การแปลงแบบ MDCT (Modified Cosine Tranfrom) เพื่อเพิ่มความละเอียดของสเปคตรัม หลังจากที่ผ่านขั้นตอนการทำ โพลีเฟสฟิลเตอร์แบงค์ แสดงบล็อคไดอะแกรมของการทำ MDCT ดังรูปที่ 3.12



รูปที่ 3.12 แสดงบล็อคไดอะแกรมของการทำ MDCT

มีรูปแบบการทำ MDCT อยู่สองลักษณะ โดยแบ่งตามความยาวของบล็อคคือ

- บล็อคยาวมี 18 สัญญาณสุ่มตัวอย่าง แต่มีการเหลื่อมของย่านความถี่อยู่ร้อยละ 50 ดังนั้นบล็อคยาวจึงมีความยาว 36 สัญญาณสุ่มตัวอย่าง
- บล็อคสั้นมี 6 สัญญาณสุ่มตัวอย่าง แต่มีการเหลื่อมของย่านความถี่อยู่ร้อยละ 50 ดังนั้นบล็อคสั้นจึงมีความยาว 12 สัญญาณสุ่มตัวอย่าง

ความแตกต่างของบล็อคยาวกับบล็อคสั้นคือ บล็อคยาวให้ความละเอียดในด้านความถี่มากส่วน บล็อคสั้นให้ความละเอียดด้านเวลาสำหรับสัญญาณทรานเซียนส์ (Transient) สังเกตว่าความยาวของบล็อคสั้นเท่ากับหนึ่งในสามของบล็อคยาว ดังนั้นถ้าใช้บล็อคสั้นต้องใช้สามบล็อคเพื่อให้ความยาวข้อมูลเท่ากับเมื่อใช้บล็อคยาว

ในการเข้ารหัสเฟรม (หนึ่งเฟรมคือ 1152 สัญญาณสุ่มตัวอย่างอินพุท) หนึ่ง ๆ สามารถใช้บล็อคชนิดเดียวกัน หรือบล็อคผสม (Mix block) ก็ได้ ถ้าเข้ารหัสแบบผสม 2 ย่านความถี่ต่ำสุดจะต้องใช้บล็อคยาวอีก 30 ย่านความถี่ที่เหลือเป็นบล็อคสั้น ซึ่งการเข้ารหัสแบบบล็อคผสมจะให้ความละเอียดด้านความถี่สูงที่ความถี่ต่ำ และความละเอียดด้านเวลาสูงที่ความที่ถี่สูง

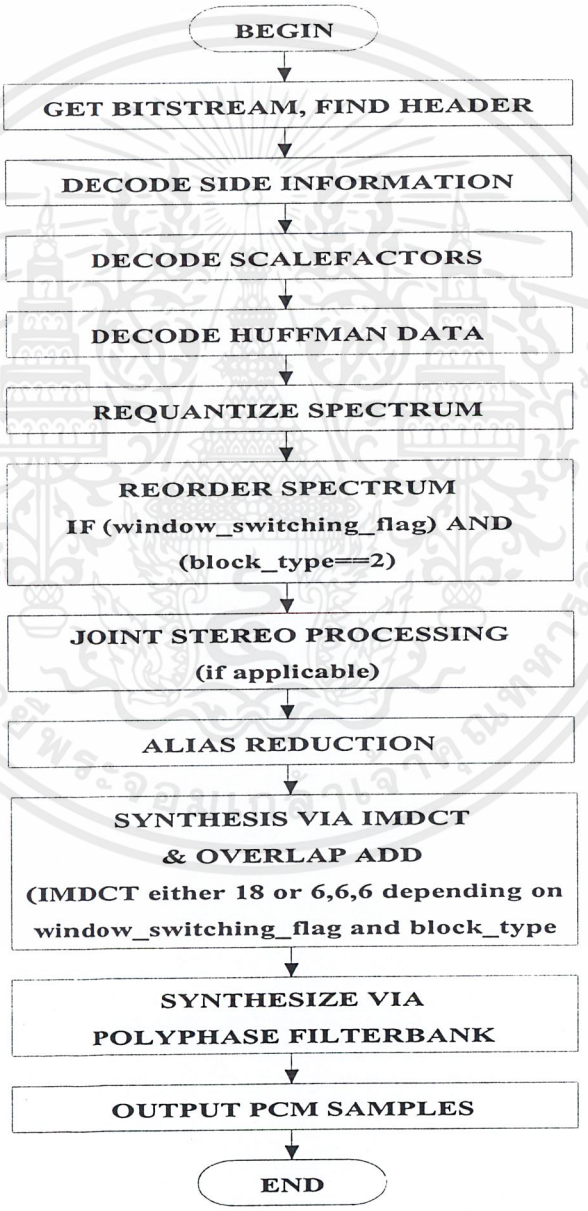
2. การลดผลจากการเหลื่อมของย่านความถี่ที่ติดกัน (Alias reduction)
3. การปรับค่าแบบไม่เป็นรูปแบบ (non – uniform quantization)
4. ย่านสเกลแฟคเตอร์ (scalefactor band) ใช้สเกลแฟคเตอร์ต่างกันเมื่อต่างย่านความถี่
5. การเข้ารหัสข้อมูลแบบเอ็นทรอปี้ เอ็มเป็กเลเยอร์ 3 ใช้การเข้ารหัสข้อมูลแบบฮัฟแมน (Huffman) มาช่วยในการบีบอัดข้อมูล

บทที่ 4

การถอดรหัสข้อมูล "เอ็มเป็ก – 1 เลเยอร์ – 3"

(MPEG – 1 Layer – 3 Decoder)

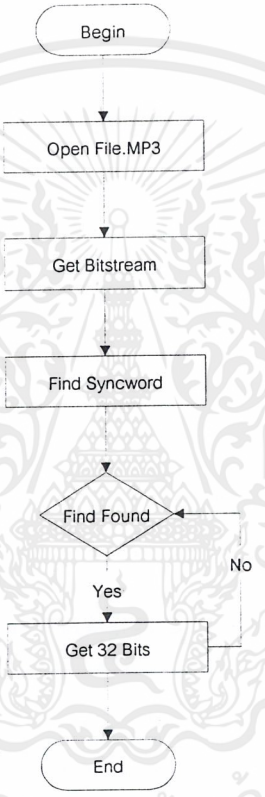
ในบทนี้จะกล่าวถึงการถอดรหัสข้อมูล "เอ็มเป็ก – 1 เลเยอร์ – 3" (MPEG – 1 Layer – 3 Decoder) การทำงานก็จะคล้ายการเข้ารหัส แต่จะทำตรงข้ามกัน จะมีขั้นตอนตามรูปที่ 4.1 แสดงโฟล์วชาร์ต (Flow Chart) ของการถอดรหัส ไฟล์ "เอ็มเป็ก – 1 เลเยอร์ – 3"



รูปที่ 4.1 แสดงโฟล์วชาร์ต (Flow Chart) ของการถอดรหัส ไฟล์ "เอ็มเป็ก – 1 เลเยอร์ – 3"

4.1 การดึงบิตข้อมูลและค้นหาส่วนหัวของเฟรม (Get Bitstream and Find Header)

กระบวนการนี้เริ่มต้นด้วยการเปิดไฟล์เอ็มเป็กเลเยอร์ 3 (MPEG – 1 Layer – 3 files) ดึงข้อมูลจากสายการไหลของบิต (Bitstream) จากนั้นทำการค้นหาส่วนหัวของเฟรมแรก (Header of First Frame) โดยค้นหา "ซิงค์เวิร์ด" (syncword) ซึ่งเป็นบิต "1" จำนวน 12 ตัวติดกัน เมื่อพบจึงทำการดึงข้อมูลจากบิตสตรีมมา 32 บิต โดยเริ่มจากบิตแรกของซิงค์เวิร์ด ส่วนรายละเอียดของแต่ละบิตได้กล่าวไปแล้วในบทที่ 3 แล้ว รูปที่ 4.2 แสดงโฟล์วชาร์ตการทำงานในส่วนนี้



รูปที่ 4.2 แสดงโฟล์วชาร์ตการทำงานในส่วนนี้

4.2 การตรวจสอบความผิดพลาด (CRC Error Checking)

การตรวจสอบความผิดพลาดที่เกิดขึ้นในการถอดรหัสข้อมูลเสียงตามมาตรฐาน MPEG1-layer3 ทำได้โดยการใช้การตรวจสอบค่า CRC (Cyclic Redundancy Code) 16 บิต การตรวจสอบความผิดพลาดจะเกิดขึ้นก็ต่อเมื่อ protection bit ในส่วนหัวของเฟรม (Header) มีค่าเท่ากับ 0 ซึ่งจะทำให้มี CRC-Check word จำนวน 16 บิต เพิ่มเข้ามาต่อจากส่วนหัวของเฟรม การตรวจสอบความผิดพลาดโดยวิธี CRC เราสามารถทำได้ดังนี้

ขั้นแรกเราต้องคำนวณหาค่า CRC จำนวน 16 บิต จากข้อมูลในบิตสตรีมเพื่อนำมาเปรียบเทียบกับ CRC-Check word ซึ่งข้อมูลในบิตสตรีมที่เรานำมาใช้ในการคำนวณค่า CRC นี้นำ

มาจากข้อมูลบางส่วนในส่วนหัวของข้อมูล (Header) และข้อมูลข้างเคียง(Side Information) ซึ่งเราจะเรียกข้อมูลส่วนนี้ว่า Protection field ในกรณีของ MPEG1-layer3 ข้อมูลในส่วน Protection field ประกอบด้วยข้อมูลต่างๆ ดังนี้

กรณี Single Channel จะมีจำนวณ 19 ไบต์ ได้แก่

- บิตที่ 16-31 ของส่วนหัวข้อมูล
- บิตที่ 0-135 ของข้อมูลข้างเคียง

กรณี Stereo/Double Channel จะมีจำนวณ 34 ไบต์ ได้แก่

- บิตที่ 16-31 ของส่วนหัวข้อมูล
- บิตที่ 0-255 ของข้อมูลข้างเคียง

โดยมี Generator Polynomial : $g(x) = x^{16}+x^{15}+x^{12}+1$

การคำนวณค่า CRC จาก Protection field ในทางคณิตศาสตร์สามารถหาได้จากเศษจากการหาร $x^{16}u(x)$ ด้วย $g(x)$ ซึ่ง $u(x)$ คือ ข้อมูลจาก Protection field ในบิตสตรีมนั่นเอง เมื่อหาค่า CRC ได้แล้วก็นำไปเปรียบเทียบกับ CRC-Check word ในบิตสตรีมถ้ามีค่าเท่ากันแสดงว่าข้อมูลในส่วน Protection field ของบิตสตรีมถูกต้อง

นอกจากนี้วิธีที่ง่ายที่สุดในการคำนวณค่า CRC สามารถทำได้โดยการใช้ Linear Feedback Shift Register (LFSR) ดังรูปที่ ก.19 ในภาคผนวก ก ซึ่งวงจรนี้จะทำหน้าที่เหมือนการหาร โดยการป้อน $u(x)$ เป็นอินพุทของวงจรทีละ 1 บิต และชิฟท์ข้อมูลในรีจิสเตอร์ทีละ 1บิตในแต่ละครั้งจนครบ 19 หรือ 34 ไบต์ แล้วเปรียบเทียบเอาท์พุทที่ได้ (b_0-b_{15}) กับ CRC-check word เช่นเดียวกัน

4.3 การถอดรหัสข้อมูลข้างเคียง (Decode Side Information)

ข้อมูลข้างเคียง (Side Information) จะเป็นส่วนต่อจากเฮดเดอร์ (ถ้าไม่มี CRC Check Error) ซึ่งมีความสำคัญเหมือนกันเพราะเป็นตัวบอกถึงค่าต่าง ๆ ที่จำเป็นในการถอดรหัสเช่น ตารางฮัฟแมน (Huffman Table) เป็นต้น ไฟล์ .MP3 ที่เป็นการเข้ารหัสโดยใช้มาตราฐาน ISO/IEC11172-3 และอยู่ในโหมด สเตอริโอ (Stereo mode) ข้อมูลข้างเคียงจะมีขนาด 32 ไบท์ ในส่วนนี้จะดึงข้อมูลจากไฟล์ MP3 ต่อจากเฮดเดอร์อีก 32 ไบท์ จากนั้นก็จะนำข้อมูลข้างเคียงไปถอดรหัสโดยนำไปเทียบกับตารางที่สร้างไว้แล้ว และดึงค่าในตารางมาเก็บไว้เพื่อนำไปใช้ในส่วนต่อไป

ต่อจากส่วนนี้ก็จะเริ่มเข้าสู่ข้อมูลเสียงหลัก (main audio data) ของเฟรม โดยจะมีขนาดเท่ากับขนาดของแต่ละเฟรม (N) ลบด้วยเฮดเดอร์ (Header(Bit)) และลบด้วยข้อมูลข้างเคียงดังสมการที่ (4.1)

$$Main\_Data(bit) = N - Header(bit) - Side\_Information(bit)$$ สมการที่ 4.1

สำหรับเลเยอร์ – 3 จะมีขนาดของแต่ละเฟรม (N) ดังสมการที่ 4.2

$$N = 144000 \times \frac{bitrate}{Sampling\ \_frequency}$$ สมการที่ 4.2

ข้อมูลหลักของเฟรมนั้น ๆ จะถูกดึงมาเก็บไว้ในบัฟเฟอร์ (Buffer) เพื่อนำไปใช้ในการถอดรหัสเอ็มเป็ก

4.4 การถอดรหัสสเกลแฟคเตอร์ (Decode ScaleFactors)

อธิบายการถอดรหัสสเกลแฟคเตอร์ได้ดังนี้

ส่วนแรกของข้อมูลหลักซึ่งเป็นอินพุทของการถอดรหัสสเกลแฟคเตอร์นี้ จะเป็นส่วนของข้อมูลที่แจ้งลักษณะข้อมูลที่เข้ารหัสมาดังแสดงในรูปที่ ก.5 ในภาคผนวก เอาท์พุทของส่วนนี้คือค่า sclefacter_s และ scalefacter_l ซึ่งจะถูกนำไปใช้ต่อในกระบวนการ รีออร์เดอร์ริง

การทำงานของส่วนถอดรหัสสเกลแฟคเตอร์ เป็นดังนี้คือ

ในรูปที่ ก.6 กระบวนการหลักของการถอดรหัสสเกลแฟคเตอร์ ค่า slen1 และ slen2 จะถูกนำมาใช้งานหลังจากคำนวณจำนวนบิตของข้อมูลที่ถูกเข้ารหัสสเกลแฟคเตอร์ (ค่า part2_length) ได้ โดยค่านี้จะถูกคำนวณจากในแต่ละแชนเนล ในแต่ละแกรนูล หลังจากนั้น สเกลแฟคเตอร์ของแต่ละ แชนเนล ของแต่ละ แกรนูล จะถูกถอดรหัสโดย 1 ใน 3 ของ อัลกอริทึม ของกระบวนการ โดยในกรณีที่เป็น วินโดว์แบบสั้น จะใช้ อัลกอริทึม decode_scalefac_s() ในกรณีที่เป็นวินโดว์แบบยาว จะใช้ อัลกอริทึม decode_scalefacter_l() และในกรณีที่เป็น วินโดว์แบบผสม ก็จะใช้ อัลกอริทึม decode_scalefacter_n&s()

จำนวนรหัสสเกลแฟคเตอร์ที่ถูกส่งมา จะถูกใช้ในแต่ละ แชนเนล ในแต่ละ แกรนูล จำนวนที่แท้จริงของรหัสสเกลแฟคเตอร์จะขึ้นอยู่กับชนิดของวินโดว์ ที่ใช้ในกระบวนการ MDCT ตอนเข้ารหัส โฟล์วชาร์ตหลักของการถอดรหัสสเกลแฟคเตอร์แสดงในรูปที่ ก.6 ในภาคผนวก ก

ข้อมูลในแต่ละ แชนเนล ในแต่ละ แกรนูล จะถูกแบ่งออกเป็น 12 หรือ 21 ช่วงความถี่สเกลแฟคเตอร์ สำหรับ วินโดว์แบบสั้น และ วินโดว์แบบยาว ซึ่งช่วงความถี่เหล่านี้ จะถูกนำไปจัดเรียงกันอีกครั้งโดยแบ่งออกเป็น 2 กลุ่ม ตามค่าของ ชนิดของบล็อก เช่น 0-10 กับ 11-20 เมื่อเป็น วินโดว์แบบยาว และ 0-5 กับ 6-11 เมื่อเป็น วินโดว์แบบสั้น โดยมีค่า slen1 และ slen2 แสดงจำนวนบิตที่ถูกเข้ารหัสมาของแต่ละกลุ่มของช่วงความถี่สเกลแฟคเตอร์

งานแรกของกระบวนการถอดรหัสสเกลแฟคเตอร์ คือ การหาค่า slen1 และ slen2 โดยการใช้ค่า scalefac_compress ซึ่งเป็นค่าตัวแปรขนาด 4 บิต ที่จะนำไปใช้เป็นดรรชนี เพื่อหาค่า

slen1 และ slen2 จากตารางตามมาตรฐาน ISO/IEC 11172-3 (MPEG 1 Layer 3 หรือ MP3) เพื่อนำไปใช้ควบคุม อัลกอริทึมถอดรหัส โดยจำนวนบิตที่ถูกเข้ารหัสสเกลแฟคเตอร์ ในแต่ละแชนเนลในแต่ละแกรนูล (ค่า part2_length) จะถูกคำนวณขึ้นมาโดยการคำนวณค่า part2_length สำหรับ แกรนูล แรก จะแสดงในรูปที่ ก.7 ในภาคผนวก ก

เมื่อจะคำนวณค่า part2_length สำหรับ แกรนูล ที่ 2 Side Information ของแต่ละแชนเนลในทั้ง 2 แกรนูล จะถูกนำมาพิจารณาด้วย โดยมีค่า scfsi ที่ระบุจำนวนบิต ที่ถูกใช้สำหรับสเกลแฟคเตอร์ ของ แกรนูล ที่ 2 ใน เฟรม นั้นๆ โดย เมื่อใช้ วินโดว์ อื่นๆ นอกจากวินโดว์แบบสั้น ค่า scfsi จะระบุจำนวนสเกลแฟคเตอร์ ที่จะส่งไปสำหรับ แกรนูล แรก และในแกรนูล ที่ 2 ก็เช่นเดียวกัน ดังเช่นที่แสดงการคำนวณค่า part2_length ของ แกรนูล ที่ 2 ในรูปที่ ก.8 ในภาคผนวก ก

อัลกอริทึม 4.1 แสดงการถอดรหัสสเกลแฟคเตอร์ เมื่อใช้ วินโดว์แบบยาว โดย อัลกอริทึมนี้ จะถูกแยกออกเป็นส่วนที่เป็นอิสระต่อกัน 4 ส่วน (Loop for 1 ครั้ง คือ 1 ส่วน) เมื่อใช้วินโดว์แบบยาว สเปกตรัมความถี่ จะถูกแบ่งออกเป็น 21 ช่วงความถี่สเกลแฟคเตอร์ ซึ่งจะถูกนำไปจัดเรียงเป็นกลุ่ม 4 กลุ่ม ตามช่วงความถี่สเกลแฟคเตอร์ 0-5, 6-10, 11-15 และ 16-20 โดย 2 กลุ่มแรกถูกเข้ารหัสจำนวน slen1 บิต ต่อ 1 ช่วงความถี่สเกลแฟคเตอร์ และ 2 กลุ่มหลังถูกเข้ารหัสจำนวน slen2 บิต ต่อ 1 ช่วงความถี่สเกลแฟคเตอร์

เมื่อถอดรหัสสเกลแฟคเตอร์ของ แกรนูล แรก ทั้ง 4 ส่วนจะถูกคำนวณ แต่ในกรณีที่เป็นแกรนูล ที่ 2 ค่า Side Information จะถูกนำมาพิจารณาด้วย

เมื่อใช้ วินโดว์แบบยาว ใน เลเยอร์ 3 จะอนุญาตให้นำค่าสเกลแฟคเตอร์ ของทั้ง 2 แกรนูล กลับมาใช้อีกได้ โดยในกรณีนี้ค่า scfsi จะมีตำแหน่งอยู่ใน Side Information ซึ่งค่าของ scfsi จะพิจารณาว่า สเกลแฟคเตอร์ที่ถูกส่งมาใช้สำหรับ แกรนูล แรก จะถูกนำมาใช้อีกครั้งสำหรับ แกรนูล ที่ 2 ซึ่งค่า scfsi แต่ละค่า จะช่วยทำให้การพิจารณานำค่าสเกลแฟคเตอร์ ของ 1 ใน 4 กลุ่ม ของช่วงความถี่สเกลแฟคเตอร์ กลับมาใช้ได้ เช่น ถ้า scfsi[0]=0 แสดงว่า สำหรับสเกลแฟคเตอร์ที่ถูกรวมอยู่ในกลุ่มที่ 1 ซึ่งจะถูกส่งไปใช้สำหรับ แกรนูล แรก แต่ถ้า scfsi[0]=1 ค่าสเกลแฟคเตอร์ที่ถูกใช้ใน แกรนูล แรก จะถูกนำไปใช้ใน แกรนูลที่ 2 ด้วย และจะไม่มีการส่งค่าสเกลแฟคเตอร์สำหรับ แกรนูล ที่ 2 มา สำหรับสเกลแฟคเตอร์กลุ่มแรก

อัลกอริทึม 4.1 decode_scalefac_l(). อัลกอริทึมสำหรับถอดรหัสสเกลแฟคเตอร์สำหรับวินโดว์แบบยาว

ค่า *coded_sf* แทนค่าสเกลแฟคเตอร์ของทั้งแชนเนลซ้ายและขวา

1: *index := 0* //Initialization of poiter used to index the coded scalefactors *coded_sf*

2: *for sfb = 0 to 20 do* //For all scalefactor bands

3: if (*sfb* < 6) then //For first group of scalefactor bands

4: if (*gr* = 0) or (scfsi[*ch*][0] = 0) then //First granule or *scalefactors* not reused

```
5:          scalefac_l[gr][ch][sfb] := coded_sf[index : (index + slen1 - 1)]    //Get slen1 bits
6:          index := index + slen1          //Increase index to point to the succeeding slen1 bits
7:        else
8:          scalefac_l[gr][ch][sfb] := coded_sf[gr – 1][ch][sfb]    //Copy scalefactors from granule 0
9:        end if
10:       else if (sfb < 11) then  //For second group of scalefactor bands
11:         if (gr = 0) or (scfsi[ch][1] = 0) then          //First granule or scalefactors not reused
12:                 scalefac_l[gr][ch][sfb] := coded_sf[index : (index + slen1 - 1)]  //Get slen1 bits
13:                 index := index + slen1  //Increase index to point to the succeeding slen1 bits
14:         else
15:                 scalefac_l[gr][ch][sfb]:=coded_sf[gr - 1][ch][sfb] //Copy scalefactors from granule 0
16:         end if
17:       else if (sfb < 16) then  //For second group of scalefactor bands
18:         if (gr = 0) or (scfsi[ch][2] = 0) then          //First granule or scalefactors not reused
19:                 scalefac_l[gr][ch][sfb] := coded_sf[index : (index + slen2 - 1)]  //Get slen2 bits
20:                 index := index + slen2  //Increase index to point to the succeeding slen2 bits
21:         else    //For fourth group of scalefactor bands
22:                 scalefac_l[gr][ch][sfb]:=coded_sf[gr - 1][ch][sfb] //Copy scalefactors from granule 0
23:         end if
24:       else
25:         if (gr = 0) or (scfsi[ch][2] = 0) then          //First granule or scalefactors not reused
26:                 scalefac_l[gr][ch][sfb] := coded_sf[index : (index + slen2 - 1)]  //Get slen2 bits
27:                 index := index + slen2  //Increase index to point to the succeeding slen2 bits
28:         else    //For fourth group of scalefactor bands
29:                 scalefac_l[gr][ch][sfb]:=coded_sf[gr - 1][ch][sfb] //Copy scalefactors from granule 0
30:         end if
31:       end if
32: end for
```

อัลกอริทึม 4.2 แสดง อัลกอริทึม ที่ใช้ในกรณีที่เป็น วินโดว์แบบสั้น โดย 36 สเกลแฟคเตอร์ จะถูกส่งไป ซึ่งสำหรับ วินโดว์แบบสั้น นี้ จำนวนช่วงความถี่สเกลแฟคเตอร์ จะมีจำนวนเท่ากับ 12 โดยจะถูกแบ่งออกเป็น 2 กลุ่ม คือ 0-5 และ 6-11 ในแต่ละช่วงความถี่สเกลแฟคเตอร์ 3 วินโดว์แบบสั้น จะถูกนำมาใช้ และ 1 สเกลแฟคเตอร์ ต่อ 1 วินโดว์ ต่อ 1 ช่วงความถี่สเกลแฟคเตอร์ ก็จะถูกส่งออกไป

สเกลแฟคเตอร์ของกลุ่มแรกถูกเข้ารหัสด้วยจำนวน slen1 บิต ต่อ 1 วินโดว์ และสเกลแฟคเตอร์ในกลุ่มที่ 2 ถูกเข้ารหัสด้วยจำนวน slen2 บิต ต่อ 1 วินโดว์

อัลกอริทึม 4.2 decode_scalefac_s().อัลกอริทึมสำหรับถอดรหัสสเกลแฟคเตอร์สำหรับวินโดว์แบบสั้น

ค่า *coded_sf* แทนค่าสเกลแฟคเตอร์ของทั้งแชนเนลซ้ายและขวา

```
1: index := 0              //Initialization of pointer used to index the coded scalefactos coded_sf
2: for sfb = 0 to 36 do //For all scalefactor bands
3:    if (sfb < 18) then //For first group of scalefactor bands
4:          scalefac_s[gr][ch][sfb] := coded_sf[index : (index + slen1 −1)]
5:          index := index + slen1 //Increase index to point to the succeeding slen1 bits
6:    else   //For second group of scalefactor bands
7:          scalefac_s[gr][ch][sfb] := coded_sf[index : (index + slen2 −1)]
8:          index := index + slen2 //Increase index to point to the succeeding slen2 bits
9:    end if
10: end for
```

อัลกอริทึม 4.3 เป็น อัลกอริทึม สำหรับถอดรหัสสเกลแฟคเตอร์ที่ใช้ วินโดว์แบบผสม ในกรณีนี้ 12 ช่วงความถี่สเกลแฟคเตอร์ จะถูกนำมาใช้ โดย 2 ชนิด ของช่วงความถี่สเกลแฟคเตอร์คือ ช่วงความถี่สเกลแฟคเตอร์แบบสั้น และ แบบยาว ซึ่งแต่ละชนิด จะถูกใช้ตามความเหมาะสมกับชนิดของ วินโดว์ ที่ใช้มาในกระบวนการ MDCT ตอนเข้ารหัส ดังนั้น เมื่อเปลี่ยนชนิด วินโดว์ ชนิดของช่วงความถี่สเกลแฟคเตอร์ ก็จะเปลี่ยนตามไปด้วย จึงเป็นสิ่งที่ควรพิจารณาเมื่อจะทำการถอดรหัสสเกลแฟคเตอร์ เพราะช่วงความถี่ของสเกลแฟคเตอร์ที่แตกต่างกันไป ใน วินโดว์แบบยาว และวินโดว์แบบสั้น ขึ้นอยู่กับความถี่สุ่ม ตามตารางของมาตรฐาน MP3

ช่วงความถี่สเกลแฟคเตอร์ที่ 0-7 (ช่วงความถี่สเกลแฟคเตอร์ ที่ใช้วินโดว์แบบยาว) และช่วงความถี่สเกลแฟคเตอร์ที่ 3-5 (ช่วงความถี่สเกลแฟคเตอร์ ที่ใช้วินโดว์แบบสั้น) จะถูกเข้ารหัสด้วยจำนวน slen1 บิต และช่วงความถี่สเกลแฟคเตอร์ที่ 6-11 (ช่วงความถี่สเกลแฟคเตอร์ ที่ใช้วินโดว์แบบสั้น) จะถูกเข้ารหัสด้วยจำนวน slen2 บิต

อัลกอริทึม 4.3 decode_scalefac_n&s().อัลกอริทึมสำหรับถอดรหัสสเกลแฟคเตอร์สำหรับวินโดว์แบบผสม

ค่า *coded_sf* แทนค่าสเกลแฟคเตอร์ของทั้งแชนเนลซ้ายและขวา

```
1: index := 0              //initialization of pointer used to index the coded scalefactors coded_sf
2: for sfb = 0 to 35 do //For all scalefactor bands
3:    if (sfb < 8) then   //For 8 long scalefactor bands
4:          scalefac_l[gr][ch][sfb] := coded_sf[index : (index + slen1 −1)]
5:          index := index + slen1 //Increase index to point to the succeeding slen1 bits
6:    else if (sfb < 18) then          //For first group of short scalefactor bands
7:          scalefac_s[gr][ch][sfb] := coded_sf[index : (index + slen1 −1)]
8:          index := index + slen1 //Increase index to point to the succeeding slen1 bits
```

```
9:    else
10:         scalefac_s[gr][ch][sfb] := coded_sf[index : (index + slen2 –1)]
11:         index := index + slen2 //Increase index to point to the succeeding slen2 bits
12:   end if
13: end for
```

4.5 การถอดรหัสข้อมูลฮัฟแมน (Decode Huffman Data)

ในการบีบอัดข้อมูลแบบ เอ็มเป็ก/ออดิโอ เลเยอร์ 3 การเข้ารหัสฮัฟแมนจะใช้กับเส้นความถี่ควอนไตซ์ (Quantized frequency lines) ซึ่งใช้พื้นฐานในการสมมติให้ช่วงความถี่สูงจะมีค่าขนาดที่น้อยกว่าช่วงความถี่ต่ำ

การเข้ารหัสจะมีประสิทธิภาพมากที่สุด เมื่อข้อมูลในการเข้ารหัสข้อมูลเส้นความถี่ควอนไตซ์ทั้งหมดจะแบ่งเป็นช่วงน้อยๆ ดังนั้นในแต่ละแกรนูลจะถูกแบ่งออกเป็น 3 ช่วง โดยจะเริ่มจากด้านขวา คือ ความถี่ที่สูงกว่าคู่เส้นความถี่ควอนไตซ์ที่มีค่าเข้าใกล้ 0 จะถูกนับจำนวนที่นับได้ เรียกว่า rzero ถัดไปจะเป็นเส้นความถี่ควอนไตซ์กลุ่มละ 4 เส้นที่มีค่าขนาดน้อยกว่าหรือเท่ากับ 1 จะถูกนับ และจำนวนที่นับได้ เรียกว่า count1 เส้นความถี่ควอนไตซ์ที่เหลืออยู่เป็นคู่ จำนวนคู่ที่เหลือนี้จะเรียกว่า big_values

สำหรับคู่ของเส้นความถี่ควอนไตซ์ที่อยู่ในช่วง rzero นี้จะไม่ส่งผ่านไป เพราะค่าภายในช่วงนี้ได้ถูกกำหนดไว้แล้ว สำหรับในช่วง count1 จะเป็นกลุ่มเส้นความถี่ควอนไตซ์ละ 4 เส้น จะถูกเข้ารหัสด้วยบิตรหัสฮัฟแมน โดยใช้ตารางฮัฟแมน A หรือ B สำหรับในช่วง big_values คู่ของเส้นความถี่ โดยใช้ตารางฮัฟแมน 1 ใน 30 ตาราง และในช่วงนี้จะถูกแบ่งออกเป็น 2 หรือ 3 ภาค และค่าเหล่านี้จะถูกเข้ารหัสโดยใช้ 2 หรือ 3 ตารางฮัฟแมนแยกออกจากกัน ขอบเขตของการแบ่งช่วง big_values และตารางฮัฟแมนที่ใช้จะถูกบอกไว้ที่ side_information

```
xxxxxxxxxxxxxxxxxx----------------------------------0000000000000000000
|                 |                               |                 |
1              big_values*2          big_values*2+count1*4          576
```

รูปที่ 4.3 การแบ่งส่วนต่างๆของแกรนูลเป็น 3 ส่วน
และแต่ละส่วนจะใช้ตารางในการเข้ารหัสแตกต่างกัน

ก่อนเข้ารหัสฮัฟแมน การจัดรูปแบบเส้นความถี่ควอนไตซ์จะต้องพิจารณาดังนี้ เริ่มแรกค่าขนาดของแต่ละเส้นความถี่จะถูกคำนวณ ต่อถ้ามีขนาดเกิน 15 จะถูกเข้ารหัสเป็นค่า 15 และเศษที่เหลือ เรียกว่า ค่า ESCAPE ซึ่งเป็นเลขฐานสอง จำนวนบิตนี้จะเรียกว่า linbits ดังนี้

ESCAPE value = Quantized frequency line – 15 ,

for Quantized frequency line ≥ 15

Linbits = Word length (ESCAPE value)

สำหรับค่าที่ไม่เท่ากับ 0 จะถูกพิจารณาโดยให้มีบิตเครื่องหมายกำหนดด้วย ดังนี้ กลุ่มละ 4 เส้นความถี่ควอนไตซ์นี้จะไม่มีค่า ESCAPE ส่งผ่าน แต่จะมีบิตเครื่องหมาย ส่งผ่านไปด้วย

การถอดรหัสข้อมูลลักษณะฮัฟแมน

ข้อมูลลักษณะฮัฟแมน (Huffman Info Decoding) เป็นขั้นตอนที่ให้ข้อมูลที่จำเป็นแก่การถอดรหัสฮัฟแมนที่เข้ามา ซึ่งจะเป็นการถอดรหัสเอาส่วนที่กำหนดรูปแบบฮัฟแมนออกจากส่วน side information และวิธีการเลือกตารางฮัฟแมน และภาคต่างๆในช่วง big_value

ใน side information ค่า part2_3length คือ ค่าความยาวของส่วนข้อมูลหลัก(main data) นั่นคือ ส่วนสเกลแฟคเตอร์และฮัฟแมนบิต ค่า part2_length คือ ค่าความยาวของส่วนข้อมูลที่เป็นสเกลแฟคเตอร์ ดังนั้นจะได้ค่าความยาวบิตของส่วนข้อมูลฮัฟแมน คือค่า part3_length = part2_3length – part2_length

ส่วนค่า big_value คือ จำนวนคู่ของเส้นความถี่ที่อยู่ในช่วง big_value ของแกรนนูล ดังนั้นในช่วง big_valuesนี้จะมีจำนวนเส้นความถี่เท่ากับ big_values = big_value*2 การที่รู้ค่านี้และ part3_length จะทำให้สามารถรู้ค่า count1 ได้ นั่นคือ ถ้าเราถอดรหัสในช่วง big_values เสร็จจำนวนบิตที่เหลือภายใน part3_length คือ ช่วง count1 ถ้าจำนวนบิตหมดแล้วแต่เส้นความถี่ยังไม่ครบ 576 นั่นเส้นความถี่ที่เหลือจะอยู่ในช่วง rzero

ในช่วง big_values นี้จะถูกแบ่งออกเป็น 2 หรือ 3 ภาค โดยขนาดของแต่ละภาคจะถูกกำหนดด้วยค่า region0_count และ region1_count ซึ่งค่านี้จะเป็นตัวชี้ที่เทอมของสเกลแฟคเตอร์แบนด์ในตารางในมาตรฐานการเข้ารหัสเอ็มเป็ก ISO/IEC 11142-3 ค่าในตารางนี้จะเป็นจำนวนเส้นความถี่ภายในช่วงนั้นๆหรืออาจกล่าวได้ว่าเป็นค่าที่น้อยกว่าจำนวนสเกลแฟคเตอร์แบนด์อยู่ 1

- ในกรณีที่ตัวแปร window_switching_flag เป็น 1 ตัดสินได้ก่อนว่า ขอบเขตของภาคจะขึ้นกับค่า block_type และ mixed_block_flag ถ้า block_type = 2 และ mixed_block_flag =1 แล้ว region0_count มีค่าเท่ากับ 8 และ region1_count จะมีค่าเท่ากับ 36 นอกจาก block_type และ mixed_block_flag กรณีนี้แล้ว region0_count มีค่าเท่ากับ 7 และ region1_count จะมีค่าเท่ากับ 36

หรือจะสรุปได้ว่าถ้า window_switching_flag = 1 จำนวนเส้นความถี่ในภาคที่ 1 เท่ากับ 36 เส้น และภาคที่ 2 คือจำนวนเส้นที่เหลืออยู่ภายในช่วง big_values

- ในกรณีที่ตัวแปร window_switching_flag เป็น 0 ค่า region0_count , region1_count จะถูกกำหนดไว้ใน side information

กระบวนการหาจำนวนเส้นความถี่ในแต่ละภาค จะแสดงอยู่ในรูปที่ ก.9 และ ก.11 ในภาคผนวก ก

เมื่อได้ข้อมูลลักษณะฮัฟแมนแล้วเราจะเริ่มขบวนการถอดรหัสข้อมูลฮัฟแมนตามที่ได้กล่าวไว้ข้างต้นดังนี้

1. ถอดรหัสในภาคแรก และใช้ตารางฮัฟแมน และค่า linbits ของภาคแรก
2. นำหัวของขบวนบิต มาเปรียบเทียบกับค่า hcod ในตารางฮัฟแมน
3. ถ้าตรงกับค่า hcod ตัวไหนก็จะนำค่า x และ y ของ hcod ตัวนั้นมา
4. ถ้าค่า x = 15 ต้องดึงข้อมูลจากขบวนของบิตซึ่งมีจำนวนบิตเท่ากับค่าของ linbit มาบวกเข้ากับ x
5. ถ้า x ไม่เท่ากับ 0 ก็ต้องดึงข้อมูลมาอีก 1 บิตเพื่อกำหนดเครื่องหมาย โดยที่

   ถ้าบิตที่ดึงมามีค่าเป็น 0 x จะมีค่าบวก

   ถ้าบิตที่ดึงมามีค่าเป็น 1 x จะมีค่าลบ
6. สำหรับค่า y ทำเช่นเดียวกับค่า x
7. เก็บค่า x และ y ลงใน “is”
8. เมื่อจำนวนตัวรวมของ x และ y เท่ากับค่า r[0] แล้ว ก็จะเริ่มถอดรหัสฮัฟแมนในภาคที่ 2 และภาคที่ 3 ต่อไป โดยการถอดรหัสจะเช่นเดียวกับภาคแรก แต่จะใช้ตารางฮัฟแมน และค่า linbits ของภาคที่ 2 และภาคที่ 3 ตามลำดับ
9. ในภาคที่ 4 จะใช้ค่าจาก count1table_select โดยค่านี้จะใช้ชี้ตารางฮัฟแมน (Huffman Table) A และ B เท่านั้น (2 ตารางนี้จะให้ค่า 4 ค่าคือ v,w,x,y จากค่า hcod 1 ค่า) ส่วนการถอดรหัสจะเหมือนกับภาคที่ 1 ต่างกันเพียงแต่จะได้ค่าจาก 4 ค่า จุดสิ้นสุดของภาคที่ 4 จะถูกกำหนดโดยค่า part2_3_length ใน side information ลบด้วยค่า part2_length (จากหัวข้อที่ 4.3) แต่ถ้าค่าของ part2_3_length – part2_length มีค่ามากกว่า 576 จุดสิ้นสุดของภาคที่ 4 จะอยู่ที่ 576

10. ในกรณีที่ค่าของ part2_3_length – part2_length มีค่าน้อยกว่า 576 ค่าที่เหลือจากตำแหน่งที่ค่าของ part2_3_length – part2_length ชี้อยู่จนถึง 576 จะให้มีค่าเป็น "0"

โฟล์วชาร์ตแสดงขั้นตอนและการตัดสินใจการถอดรหัสฮัฟแมน ดังรูปที่ ก.10 ในภาคผนวก ก

4.6 การรีควอนไตซ์ (Requantize Spectrum)

เป็นการนำเอาค่าจาก "is" มาทำการรีควอนไตซ์จะได้ผลลัพท์เป็นสัญญาณที่ถูกสุ่มตัวอย่าง (Sampling Signal) ในแกนความถี่ (Frequency Domain) จะกำหนดให้สัญญาณนี้เป็น "xr" โดยแบ่งเป็น 2 ขั้นตอน คือ การดีสเกลลิงและการรีออร์เดอร์ริงตามลำดับ

4.6.1 การดีสเกลลิง(Descaling)

จุดประสงค์ของการทำดีสเกลลิง คือ เพื่อที่จะสร้าง perceptually identical copy ของเส้นความถี่ที่ได้มาจากการทำ MDCT ในการเข้ารหัส การทำดีสเกลลิงจะขึ้นอยู่กับค่าเส้นความถี่ที่ผ่านการควอนไตซ์และสเกลแล้ว (is_i) ซึ่งถูกสร้างขึ้นมาใหม่โดยการถอดรหัสฮัฟแมนและสเกลแฟคเตอร์ ที่สร้างมาจากการถอดรหัสสเกลแฟคเตอร์ การคำนวณค่าเส้นความถี่ของ 1 แชลเนล ภายใน 1 แกรนูล สามารถทำได้ดังสมการ

-กรณีบล็อกยาว

$$xr_i = sign(is_i) \times |is_i|^{\frac{4}{3}} \times 2^{\frac{1}{4}(global\_fain[gr]-210)} \times 2^{-(scalefac\_multiplier \times (scalefac\_s[gr][ch][sfg][window]))}$$

สมการที่ 4.3

-กรณีบล็อกสั้น

$$xr_i = sign(is_i) \times |is_i|^{\frac{4}{3}} \times 2^{\frac{1}{4}(global\_fain[gr]-210-(8\times subblock\_gain[window][gr]))} \times 2^{-(scalefac\_multiplier \times (scalefac\_s[gr][ch][sfg][window]))}$$

สมการที่ 4.4

ทั้งสมการ 4.3 และ 4.4 ค่า is_i จะถูกยกกำลังด้วย 4/3 ซึ่งเป็นค่าตัวยกกำลังส่วนกลับที่ใช้ในการปรับค่าไม่เป็นรูปแบบ (Non-uniform quentizer) ตอนเข้ารหัส หลังจากนั้นจึงมีการนำเครื่องหมายของ is_i มาใช้

ซึ่งจะเห็นว่าค่าของ xr_i จะขึ้นอยู่กับค่าต่อไปนี้

- ค่า is_i ซึ่งเป็นค่าเอาท์พุทจากฮัฟแมน

- ตัวแปร grobal_gain ที่ใช้ในสมการ 4.3 และ 4.4 หมายถึง ขนาดขั้นตอนของการควอนไตซ์ที่ใช้ต่อ 1 แซลเนลใน 1 แกรนูล
- ค่า 210 คือ ค่าคงที่ของระบบซึ่งแน่ใจได้ว่ามีการสเกลของเอาท์พุทที่เหมาะสมตาม [ISO/IEC 11172-3,1993]
- ค่า scalefactor ซึ่งถูกควอนไตซ์แบบล็อกการิทึมขณะเข้ารหัสด้วยค่า 2 และ $\sqrt{2}$ ซึ่งสามารถพิสูจน์ได้จากค่าของ scalefac_scale flag โดย

  if scalefac_scale = 0 then scalefac_multiplier = 0.5

  else scalefac_multiplier = 1
- ส่วนตัวแปร pretab และ preflag จะถูกใช้ในกรณีของบล็อกยาวใน MDCT เท่านั้น ตาราง ใน [ISO/IEC 11172-3,1993] จะระบุค่าที่แน่นอนของตัวแปร pretab ที่ใช้ในแต่ละย่านความถี่สเกลแฟคเตอร์ ดังตารางที่ 4.1

ตารางที่ 4.1 แสดงค่า pretab[cb] ที่ scalefactor band ต่างๆ

Scalefactor band	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Pretab[cb]	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	2	2	3	3	3	2

ตัวแปร prefag จะถูกรวมเข้ากับสเกลแฟคเตอร์ scalefac_l เพื่อเพิ่มการขยายสัญญาณที่ความถี่สูง

- ค่า scalefac_l และ scalefac_s คือ ค่าสเกลแฟคเตอร์ของบล็อกยาวและบล็อกสั้นที่ได้จากการถอดรหัสสเกลแฟคเตอร์ตามลำดับ

เมื่อมีการใช้บล็อกสั้นในสมการ 4.4 เกนแฟคเตอร์ สำหรับแต่ละบล็อกสั้นจะถูกกำหนดโดยค่า subblock_gain ใน side information ซึ่งค่า subblocck_gain จะเป็นค่าเฉพาะสำหรับแต่ละแซลเนลในแต่ละแกรนูล

ฉะนั้นในหัวข้อนี้จึงเป็นเพียงแค่การแทนค่าตามสูตร แต่ในการแทนค่าต้องอยู่ในเงื่อนไขต่อไปนี้

ค่าของ scalefac_l และ scalefac_s ใช้ค่าเดียวกันในแต่ละย่านความถี่ย่อย

การใช้สูตร

1. ถ้า window_switching_flag = 1 และ block_type[gr][ch] == 2 และ
   1.1 mixed_block_flag = 0 xr_i ทุกตัวเป็นบล็อกสั้นทั้งหมด
   1.2 mixed_block_flag = 1 xr_i 36 ตัวแรกจะเป็นบล็อกยาว ที่เหลือจะเป็นบล็อกสั้น ทั้งหมด
2. จาก 1 ถ้าไม่ใช่ xr_i ทุกตัวจะเป็นบล็อกยาวทั้งหมด

4.6.2 การรีออร์เดอร์ริง (Reordering)

เส้นความถี่ที่ได้จากส่วนดีสเกลลิงมักจะไม่ถูกเรียงไปในทิศทางเดียวกัน โดยที่ในส่วนของการทำ MDCT การใช้บล็อกยาวขึ้นอยู่กับการทรานส์ฟอร์ม (Transformation) เป็นหลัก ซึ่งจะสร้างลำดับของเส้นความถี่ เริ่มจากย่านความถี่ย่อยเป็นอันดับแรก ต่อมาคือวินโดว์ (Window) และความถี่ตามลำดับ ทำให้เป็นการช่วยเพิ่มประสิทธิภาพของการเข้ารหัสฮัฟแมน แต่ในกรณีของบล็อกสั้นเส้นความถี่จะถูกเรียงลำดับจากย่านความถี่ย่อย ต่อมาคือ ความถี่และวินโดว์เป็นลำดับสุดท้าย

จุดประสงค์ของการทำรีออร์เดอร์ริง คือ การตรวจสอบว่าบล็อกสั้นถูกใช้ในย่านความถี่ย่อยใดๆ ใน 36 ย่านความถี่ย่อยหรือไม่ เพราะเฉพาะกรณีของบล็อกสั้นเท่านั้น ที่จะต้องมีการรีออร์เดอร์ริงค่าของเส้นความถี่ อย่างไรก็ตามการทำรีออร์เดอร์ริง จำเป็นที่จะทดสอบเมื่อมีพิจารณาค่า window_switching_flag และ block_type ในส่วนของ sideinformation เท่านั้น ซึ่งอัลกอริทึม (Algorithm) สำหรับการทำรีออร์เดอร์ริงของย่านความถี่ย่อยหนึ่งๆ แสดงในอัลกอริทึม 2.1

อัลกอริทึม 2.1 อัลกอริทึม 2.1 สำหรับการรีออร์เดอร์ริงของเส้นความถี่ในย่านความถี่ย่อยหนึ่งๆ n แทนย่านความถี่ย่อยและอยู่ในช่วง {0,1,2,...,31} Fd แทนเส้นความถี่ที่ยังไม่ได้รีออร์เดอร์และ Fo แทนเส้นความถี่ที่ผ่านการรีออร์เดอร์แล้ว

```
1: Reorder(n)
2: for window = 0 to 2 do
3:    for index = 0 to 5 do
4:       Fo(n*18+6*window+index) = Fd(n*18+window+3*index)
5:    end for
6: end for
```

การทำรีออร์เดอร์ริงขึ้นอยู่กับการดีสเกลลิงที่มีการเคลื่อนย้ายข้อมูลจากอัลกอริทึม รวมทั้งข้อมูลในส่วน DSP ตามปกติในกระบวนการทำดีสเกลลิงขึ้นอยู่กับชนิดของวินโดว์ที่ใช้และย่านสเกลแฟคเตอร์ การที่ขึ้นอยู่กับชนิดของวินโดว์จะแสดงออกมาเพราะว่าชนิดของวินโดว์จะถูกพิจารณาเป็นส่วนสำคัญในการทำดีสเกลลิง และเนื่องจากความจำเป็นนี้แทนที่เราจะเลือกสมการ 2.1 หรือ 2.2 มาใช้ ซึ่งในการเลือกเอาสมการใดมาใช้นั้นขึ้นอยู่กับย่านสเกลแฟคเตอร์และค่าอื่นๆ ของเส้นความถี่ ภายในแต่ละย่านสเกลแฟคเตอร์ ตัวชี้ความถี่(Frequency index) ของแต่ละย่านสเกลแฟคเตอร์ จะถูกกำหนดไว้ในตารางใน [ISO/IEC 11172-3,1993] เมื่อจำนวนของเส้นความถี่ในแต่ละย่านสเกลแฟคเตอร์ถูกดีสเกลแล้วจำนวน freq line ภายในย่านสเกลแฟคเตอร์ที่หาได้จะถูก

ค้นหา ซึ่งตัวชี้ย่านสเกลแฟคเตอร์(Scalefactor band index) ครั้งนี้จะขึ้นอยู่กับชนิดของวินโดว์และความถี่สุ่ม(Sampling frequency)

การรีออร์เดอร์ริงของเส้นความถี่ที่ถูกควอนไตซ์

เอาท์พุทจากการถอดรหัสฮัฟแมนซึ่งเป็นเส้นความถี่ที่ถูกควอนไตซ์แล้ว มักจะไม่ได้เรียงลำดับตามความถี่ที่เพิ่มขึ้น ดังนั้นเมื่อมีการใช้บล็อกยาว เส้นความถี่ที่ถูกควอนไตซ์แล้วจะถูกจัดเรียงใหม่จากย่านความถี่ย่อย และความถี่ที่เพิ่มขึ้นภายในแต่ละย่านความถี่ย่อย ถ้ามีการใช้บล็อกสั้น การเรียงลำดับเส้นความถี่ที่ถูกควอนไตซ์แล้วจะถูกจัดเรียงโดยเส้นความถี่ที่ถูกควอนไตซ์แล้วที่แตกต่างกัน โดยในกรณีนี้เส้นความถี่ที่ถูกควอนไตซ์แล้วจะถูกจัดเรียงใหม่โดยความถี่และวินโดว์ตามลำดับ จุดประสงค์ของอัลกอริทึมที่ใช้ในการรีออร์เดอร์ริง คือ สำหรับกรณีของบล็อกสั้นเท่านั้นที่จะมีการเรียงลำดับของเส้นความถี่ที่ถูกควอนไตซ์ใหม่ โดยเริ่มจากย่านความถี่ย่อยและความถี่แทน การรีออร์เดอร์ริง เป็นกระบวนการที่ง่ายกว่าการประกอบขึ้นใหม่ของการรีออร์เดอร์ริงของเส้นความถี่ภายในแต่ละย่านความถี่ย่อย การรีออร์เดอร์ริงของเส้นความถี่ซึ่งสอดคล้องกับย่านความถี่ย่อยเดียวแสดงในรูปที่ ก.17 ในภาคผนวก ก

การรีออร์เดอร์ริงของตัวแปรที่ขึ้นอยู่กับย่านสเกลแฟคเตอร์

การเรียงลำดับของเส้นความถี่ที่ถูกควอนไตซ์แล้ว ต้องการเรียงลำดับของสเกลแฟคเตอร์และค่าตัวแปรอื่นๆ ที่ขึ้นอยู่กับย่านสเกลแฟคเตอร์ เช่น subblock_gain และ pretab ส่วนที่สำคัญของการทำรีออร์เดอร์ริง คือ ค่าตัวแปรเหล่านี้ในแต่ละเส้นความถี่จะต้องถูกนำไปพิจารณาแทนที่จะพิจารณาตัวแปรที่ขึ้นอยู่กับย่านสเกลแฟคเตอร์ของแต่ละเส้นความถี่ ตัวชี้ย่านสเกลแฟคเตอร์ที่แตกต่างกันจะถูกนำขึ้นมาพิจารณา โดยที่ตัวชี้ย่านสเกลแฟคเตอร์จะขึ้นอยู่กับความถี่สุ่ม (fs) และชนิดของวินโดว์ที่ใช้งานว่าเป็นบล็อกสั้นหรือบล็อกยาว รูปแสดงการแจกแจงของสเกลแฟคเตอร์, วินโดว์และย่านสเกลแฟคเตอร์ ที่ fs = 32 kHz ,sfb_index แสดงตัวชี้เริ่มต้นของแต่ละย่านสเกลแฟคเตอร์แสดงในรูปที่ ก.17 ในภาคผนวก ก

การแจกแจงของสเกลแฟคเตอร์ที่ใช้กับสเปคตรัมความถี่(Frequency spectrum) สามารถอธิบายได้ดังนี้

- กรณีบล็อกยาว สเกลแฟคเตอร์ที่ใช้จะมีทั้งหมด 21 ย่านสเกลแฟคเตอร์
- กรณีบล็อกสั้น สเกลแฟคเตอร์เดี่ยวที่ใช้กับทุกๆ 3 บล็อก จะเป็นย่านสเกลแฟคเตอร์เดียวกัน

- กรณีบล็อกรวม สเกลแฟคเตอร์เดี่ยวจะใช้กับทุกๆ 8 ย่านสเกลแฟคเตอร์ยาวแรกเท่านั้น ส่วนย่านสเกลแฟคเตอร์ที่เหลือจะมีการใช้สลับไปมา แล้วแต่ว่าบล็อกที่เหลือเป็นบล็อกสั้นหรือบล็อกยาว โดยจะใช้สเกลแฟคเตอร์เดี่ยวกับทุกๆ 3 บล็อกในย่านสเกลแฟคเตอร์เดียวกันก็ต่อเมื่อมีการย่านสเกลแฟคเตอร์สั้นเท่านั้น

ต่อไปนี้เป็นอักษรย่อที่ใช้ในกระบวนการเบื้องต้น(Preprocessing) ซึ่งได้แก่ scalefactors ,subblock_gain และ pretab คือ sf[i],sbg[i] และ pt[i] ตามลำดับ โดยที่ i แทนตัวชี้ความถี่มีค่าตั้งแต่ 0-575 ในการพิจารณาเลือกใช้ค่า sf[i],sbg[i] และ pt[i] สำหรับแต่ละย่านสเกลแฟคเตอร์และวินโดว์(ถ้ามีการใช้) ค่าเหล่านี้จะต้องถูกคัดลอกเป็นความถี่ทั้งหมดภายในย่านสเกลแฟคเตอร์เฉพาะ จำนวนของเส้นความถี่ภายในย่านสเกลแฟคเตอร์ขึ้นอยู่กับความถี่สุ่มและชนิดของวินโดว์ที่ใช้ โดยที่จำนวนของตัวชี้สเกลแฟคเตอร์และตัวชี้จุดสิ้นสุดของแต่ละย่าสเกลแฟคเตอร์ จะระบุอยู่ใน [ISO/IEC 11172-3,1993] สำหรับทั้งกรณีบล็อกสั้นและบล็อกยาวและค่าความถี่สุ่มที่เป็นไปได้ทั้ง 3 ความถี่ คือ 32,44.1และ 48 kHz รูปที่ 4.13 แสดงการแจกแจงสเกลแฟคเตอร์ที่ความถี่ fs = 32 kHz

การเลือกใช้ sf[i],sbg[i] และ pt[i] สามารถกระทำได้โดยใช้อัลกอริทึม 4.4 ซึ่งจะพบว่า ย่านความถี่สเกลแฟคเตอร์ที่เพิ่มเข้ามาสำหรับวินโดว์ทั้ง 3 ประเภท สามารถใช้ได้โดยอาศัยสมการสมการเดียว ย่านสเกลแฟคเตอร์นี้จะครอบคลุมความถี่ตั้งแต่จุดสิ้นสุดของย่านความถี่สุดท้ายซึ่งไม่เกิน 575 ส่วนค่า sf[i],sbg[i] และ pt[i] จะเป็น 0 ที่ตำแหน่งความถี่เหล่านี้ โดยที่เราสามารถทำการรีออร์เดอร์ริงก่อนหรือทำการดีสเกลลิงก่อนก็ได้ และเราสามารถรวมสมการ 4.3 และ4.4 เป็นสมการ 4.5

$$xr[i] = sign(is_{ro}[i]) \bullet abs(is_{ro}[i])^{\frac{4}{3}} \bullet \frac{2^{\frac{1}{4}(gloval\_gain - 210 - 8 \bullet sbg_{ro}[i]}}{2^{(scalefac\_multiplier \bullet sf_{ro}[i] + preflag \bullet pt_{ro}[i])}}$$ สมการที่ 4.5

จากสมการ 4.5 จะพบว่าจะเหลือค่าตัวแปรเพียงค่าเดียวเท่านั้น คือ ตัวชี้ความถี่ (i) เมื่อเปรียบเทียบกับสมการ 4.3 และ 4.4 การดีสเกลลิงตอนนี้สามารถกระทำได้โดยใช้สมการเพียงสมการเดียวและไม่ขึ้นกับชนิดของวินโดว์หรือย่านสเกลแฟคเตอร์ สมการ 4.5 เป็นการกระทำใน 1 แซลเนลในแต่ละแกรนูล

อัลกอริทึม 4.4 อัลกอริทึมสำหรับกระบวนการเบื้องต้นของ scalefactors, subblock_gain และ pretab จากข้อมูลใน 1 แซลเนล โดย *sf[i], sbg[i]* และ *pt[i]* แทน scalefactors, subblock_gain และ pretab ตามลำดับสำหรับเส้นความถี่ที่ *i* ซึ่ง $i \in \{0,1,2,\ldots,575\}$ *sfb_index_l[fs][sfb]* และ *sfb_index_s[fs][sfb]* สอดคล้องกับตัวชี้เริ่มต้นของย่านสเกลแฟคเตอร์ยาวและสั้นตามลำดับ โดยใช้ ความถี่สุ่ม *fs* และย่านสเกลแฟคเตอร์ *sfb*

1: if window_switching_flag = 1 and block_type = 2 then

2: if mixed_block_flag = 1 then // For short and long windows

3: for *sfb* = 0 to 7 do // For the first 8 long scalefactor bands

```
4:        for i = sfb_index_l[fs][sfb] to sfb_index_l[fs][sfb+1]-sfb_indes_l[fs][sfb] do
5:          sf[i] = scalefac_l[sfb]              // Copy scalefactors
6:          sbg[i] = 0                           // Gain for subblocks not used for long windows
7:          pt[i] = pretab[sfb]                  // Copy pretab values
8:        end for
9:      end for
10:     for sfb = 3 to 11 do                     // For the last 9 short scalefactor bands
11:       for window = 0 to 2 do                 // For each windows within the scalefactor bands
12:         for i = sfb_index_s[fs][sfb] to sfb_index_s[fs][sfb+1]-sfb_index_s[fs][sfb] do
13:           sf[i] = scalefac_s[sfb+window]   // Copy scalefactors
14:           sbg[i] = subblock_gain[window]   // Copy gain for subblocks
15:           pt[i] = 0                        // Pretab values not used for short windows
16:         end for
17:       end for
18:     end for
19:   else                                       // For only short windows
20:     for sfb = 0 to 11 do                     // For all 12 short scalefactor bands
21:       for window = 0 to 2 do                 // For each windows within the scalefactor bands
22:         for i = sfb_index_s[fs][sfb] to sfb_index_s[fs][sfb+1]-sfb_index_s[fs][sfb] do
23:           sfb[i] = scalefac_s[sfb+window]  // Copy scalefactors
24:           sbg[i] = subblock_gain[window]   // Copy gain for subblocks
25:           pt[i] = 0                        // Pretab values not used for short windows
26:         end for
27:       end for
28:     end for
29:   end if
30: else                                         // For only long windows
31:   for sfb = 0 to 20 do                       // For the all 20 long scalefactor bands
32:     for i = sfb_index_l[fs][sfb] to sfb_index_l[fs][sfb+1]-sfb_index_l[fs][sfb] do
33:       sf[i] = scalefac_l[sfb]                // Copy scalefactors
34:       sfb[i] = 0                             // Gain for subblock not used for long windows
35:       pt[i] = pretab[sfb]                    // Copy pretab values
36:     end for
37:   end for
38: end if
```

```
39: for i' = i to 575 do                    // Zeropad for the rest of the spectrum
40:   sf[i'] = 0                            // Zeropad scalefactors
41:   sfb[i'] = 0                           // Zeropad subblock gains
42:   pt[i'] = 0                            // Zeropad pretab values
43: end for
```

อัลกอริทึม 4.5 อัลกอริทึมสำหรับการรีออร์เดอร์ริง 576 แซมเปิล ซางสอดคล้องกับข้อมูลใน 1 แชลเนล แซมเปิล (samples)จะถูกเรียงลำดับใหม่เป็น 32 ย่านความถี่ย่อย แต่ละย่านความถี่ย่อยประกอบด้วย 18 สับแบนแซมเปิล (subbands samples) $X \in \{is, sf, sbg, pt\}$

```
1: if window_switching_flag ang block_type then
2:   if mixed_block_flag = 1 then              // For short and long windows
3:     for sb = 0 to 1 do                      // For the first 2 subbands
4:       for ss = 0 to 17 do                   // For all 18 subbands samples
5:         Xro[18*sb+ss] = Xdis[18*sb+ss]      // No ordering
6:       end for
7:     end for
8:     for sb = 2 to 31 do                     // For the last 30 subbands
9:       for window = 0 to 2 do                // For each of the 3 windows
10:        for ws = 0 to 5 do                  // For each samples within a windows
11:          Xro[18*sb+ ss] = Xdis[18*sb+window+3*ws]   // Reorder
12:        end for
13:      end for
14:    end for
15:  else                                      // For only short windows
16:    for sb = 0 to 31 do                     // For all 32 subbands
17:      for window = 0 to 2 do                // For each of the 3 windows
18:        for ws = 0 to 5 do                  // Foreach samples within a windows
19:          Xro[18*sb+ss] = Xdis[18*sb+window+3*ws]   // Reorder
20:        end for
21:      end for
22:    end for
23:  end if
24:  else                                      // For only long windows
25:    for sb = 0 to 31 do                     // For all 32 subbands
```

```
26:     for ss = 0 to 17 do                    // For all 18 subbands samples
27:       Xro[18*sb+ss] = Xdis[18*sb+ss]       // No reordering
28:     end for
29:   end for
30: end if
```

รูปแบบการคำนวณที่ใช้ในการดีออร์เดอร์ริง

การคำนวณในส่วนของการดีออร์เดอร์ริง สามารถแสดงได้เป็น 2 อัลกอริทึม คือ อัลกอริทึม 4.4 และ 4.5 อัลกอริทึม 4.4 เป็นกระบวนการเตรียมค่าต่างๆ ที่จำเป็นต้องใช้ของเส้นความถี่ ภายในแต่ละวินโดว์ของแต่ละย่านสเกลแฟคเตอร์ โดยจะมีเส้นความถี่ทั้งหมด 576 เส้นความถี่ในแต่ละค่าตัวแปรที่ต้องมีการเตรียมไว้ก่อน ซึ่งมีทั้งหมด 3 ตัวแปร คือ sf[i], sbg[i] และ pt[i] ซึ่งถูกเก็บไว้ในหน่วยความจำ ซึ่งกระบวนการนี้จะทำซ้ำในแต่ละแชลเนลของแต่ละแกรนูล ส่วนอัลกอริทึม 4.5 เป็นการทำการทำรีออร์เดอร์ริง โดยนำค่าอินพุทจากหน่วยความจำมาเก็บใน Reordered address ดังนั้นการกระทำพื้นฐานที่ต้องการในอัลกอริทึมนี้ คือ การเคลื่อนย้ายหน่วยความจำ เช่น การโหลดและเก็บโอเปอร์เรชั่น(Operations), การรวมกันของโอเปอร์เรชั่นทางคณิตศาสตร์ ซึ่งคำสั่ง for loop และ if sentence จะใช้ในการควบคุมโอเปอร์เรชั่นเหล่านี้ เช่น การกระโดดแทนการตัดสินใจ อัลกอริทึม 4.5 จะถูกกระทำโดย MEM operation ดังนั้นกระบวนการรีออร์เดอร์ริงจริงๆ แล้วประกอบด้วย การเคลื่อนย้ายค่าจากตำแหน่งหนึ่งในหน่วยความจำไปยังตำแหน่งอื่นๆ เมื่อมีการใช้เฉพาะบล็อกสั้นเท่านั้น เส้นความถี่ทั้งหมด 576 เส้นความถี่จะถูกเรียงลำดับใหม่ ดังนั้นกระบวนการรีออร์เดอร์ริง เสมือนกับการเตรียมค่าตัวแปรต่างๆ ทั้ง 3 ตัวแปร คือ sf[i], sbg[i] และ pt[i] บรรจุไว้ในเวิร์ดเดียวกัน ซึ่งวิธีการนี้ก็เพื่อที่จะดำเนินการเพียงครั้งเดียวเท่านั้นแทนที่จะใช้ 3 MEM operation ต่อ 1 เส้นความถี่ นอกจากนี้จำนวนของข้อมูลที่จำเป็นต้องถูกส่งไปยังส่วนของ DSP ก็จะลดลงครึ่งหนึ่ง

4.7 การลดค่าปลอม (Alias Reduction)

ในกรณีที่ค่า xr_i เป็นบล็อกยาว (block_type ไม่เท่ากับ 2) จะเกิดการเหลื่อมกันของสเปกตรัมจนทำให้เกิดค่าปลอมของ xr_i ขึ้น 1 ตัว (เกิด alias) จึงต้องทำการลดค่าปลอม (Alias Reduction) ลงโดยใช้โปรแกรมดังนี้

```
For (sb = 1 ; sb<32 ; sb++)
    For (I = 0 ; I<8 ; I++){
        Xar[18*sb-1-i] = xr[18*sb-1-i]  Cs[i] – xr[18*sb+i]Ca[i]
```

Xar[18*sb+i] = xr[18*sb+i]Cs[i] + xr[18*sb-1-i]Ca[i]

}

โดยค่า Ca[i] และ Cs[i] หาได้จากสูตรดังนี้

$$Cs[i] = \frac{1}{\sqrt{1+Ci^2}}$$ สมการที่ 4.6

$$Ca[i] = \frac{Ci}{\sqrt{1+Ci^2}}$$ สมการที่ 4.7

ซึ่งค่า Ci เป็นค่าสัมประสิทธ์ของการลดค่าปลอม(Coefficients for alias reduction) มีค่าดังตารางที่ 4.2

ตารางที่ 4.2 แสดงค่าสัมประสิทธ์ของการลดค่าปลอม(Coefficients for alias reduction)

(i)	C_I
0	-0.6000
1	-0.5350
2	-0.3300
3	-0.1850
4	-0.0950
5	-0.0410
6	-0.0142
7	-0.0037

4.8 IMDCT (Inverse Modified Discrete Cosine Transform)

เป็นการแปลงสัญญาณในแกนความถี่ (Frequency Domain) เป็นสัญญาณเชิงเวลา (Time Domain) โดยใช้สูตรดังสมการที่ 4.7 แสดงสูตรการแปลง IMDCT

$$x_i = \sum_{k=0}^{\frac{n}{2}-1} X_k \cos\left(\frac{\pi}{2}\left(2i+1+\frac{n}{2}\right)(2k+1)\right)$$ for i=0 to n-1 สมการที่ 4.8

โดยที่ x_i เป็นสัญญาณในแกนเวลา (Time Domain)

X_k เป็นสัญญาณในแกนความถี่ (Frequency domain)

n เป็นจำนวนของตัวอย่างวินโดว์ (บล็อกสั้นใช้ n =12, บล็อกยาว n = 36)

4.9 การทำวินโดว์ (Windowing)

ขึ้นกับค่า Block_type ดังนี้

4.9.1 Block_type = 0 (normal window)

ใช้สูตรดังสมการที่ 4.9

$$z_i = x_i \sin\left(\frac{\pi}{36}\left(i+\frac{1}{2}\right)\right) \qquad \text{for i=0 to 35} \qquad \text{สมการที่ 4.9}$$

4.9.2 Block_type = 1 (start block)

ใช้สูตรดังสมการที่ 4.10

$$z_i = x_i \sin\left(\frac{\pi}{36}\left(i+\frac{1}{2}\right)\right) \qquad \text{for i=0 to 17}$$

$$z_i = x_i \qquad \text{for i=18 to 23}$$

$$z_i = x_i \sin\left(\frac{\pi}{12}\left(i-18+\frac{1}{2}\right)\right) \qquad \text{for i=24 to 29}$$

$$z_i = 0 \qquad \text{for i=30 to 35} \qquad \text{สมการที่ 4.10}$$

4.9.3 Block_type = 3 (stop block)

ใช้สูตรดังสมการที่ 4.11

$$z_i = 0 \qquad \text{for i=0 to 5}$$

$$z_i = x_i \sin\left(\frac{\pi}{12}\left(i-6+\frac{1}{2}\right)\right) \qquad \text{for i=6 to 11}$$

$$z_i = x_i \qquad \text{for i=12 to 17}$$

$$z_i = x_i \sin\left(\frac{\pi}{36}\left(i+\frac{1}{2}\right)\right) \qquad \text{for i=18 to 35} \qquad \text{สมการที่4.11}$$

4.9.4 Block_type = 2 (short block)

มี 3 window แยกกัน window ละ 12 ตัว โดยที่

$$y_i^{(j)} = x_i^{(j)} \sin\left(\frac{\pi}{12}\left(i+\frac{1}{2}\right)\right) \qquad \text{for i=0 to 11 , j =0 to 2} \qquad \text{สมการที่ 4.12}$$

โดยที่ i เป็นลำดับสัญญาณ

j เป็นลำดับวินโดว์

ใช้สูตรดังสมการที่ 4.13

$$z_i = 0 \qquad \text{for i=0 to 5}$$

$$z_i = y_{i-6}^{(1)} \qquad \text{for i=6 to 11}$$

$$z_i = y_{i-6}^{(1)} + y_{i-12}^{(2)} \qquad \text{for i=12 to 17}$$

$$z_i = y_{i-12}^{(2)} + y_{i-18}^{(3)} \qquad \text{for i=18 to 23}$$

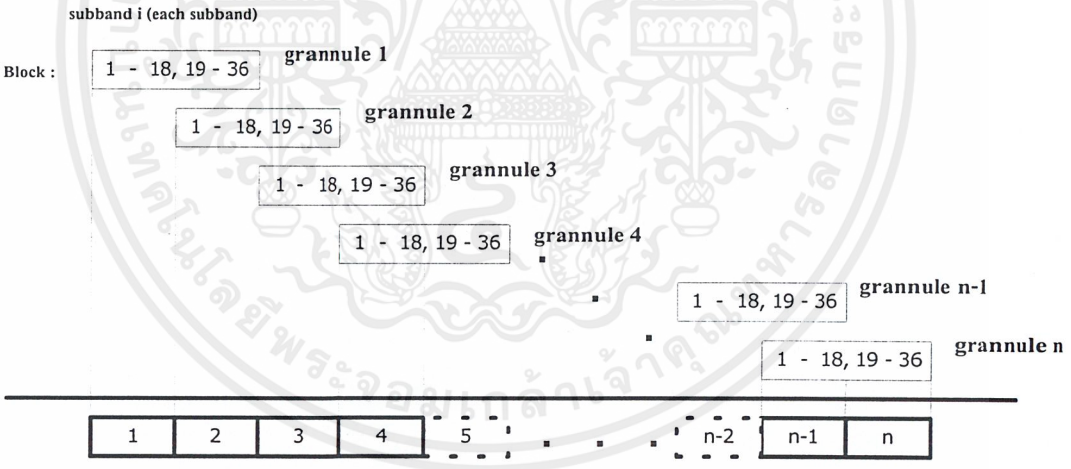
$$z_i = y_{i-18}^{(3)} \qquad \text{for i=24 to 29}$$

$$z_i = 0 \qquad \text{for i=30 to 35}$$

สมการที่ 4.13

4.10 การทำโอเวอร์แลบ (Overlapping)

ครึ่งแรกของบล็อกของค่า 36 ค่า จะทับกับครึ่งหลังของบล็อกก่อนหน้านี้ (Previous block) ดังรูปที่ 4.5



Overlap add of the resulf of the IMDCT

For grannule = 1 to n

{ ในแต่ละบล็อก ของ subband มี 36 samples ซึ่ง ครึ่งบล็อกแรกนี้จะโอเวอร์แลป กับครึ่งบล็อกหลัง ของแกรนนูลก่อนหน้า และครึ่งบล็อกหลังนี้จะเก็บไว้เพื่อใช้บวกกับครึ่งบล็อกแรกของแกรนนูลถัดไป

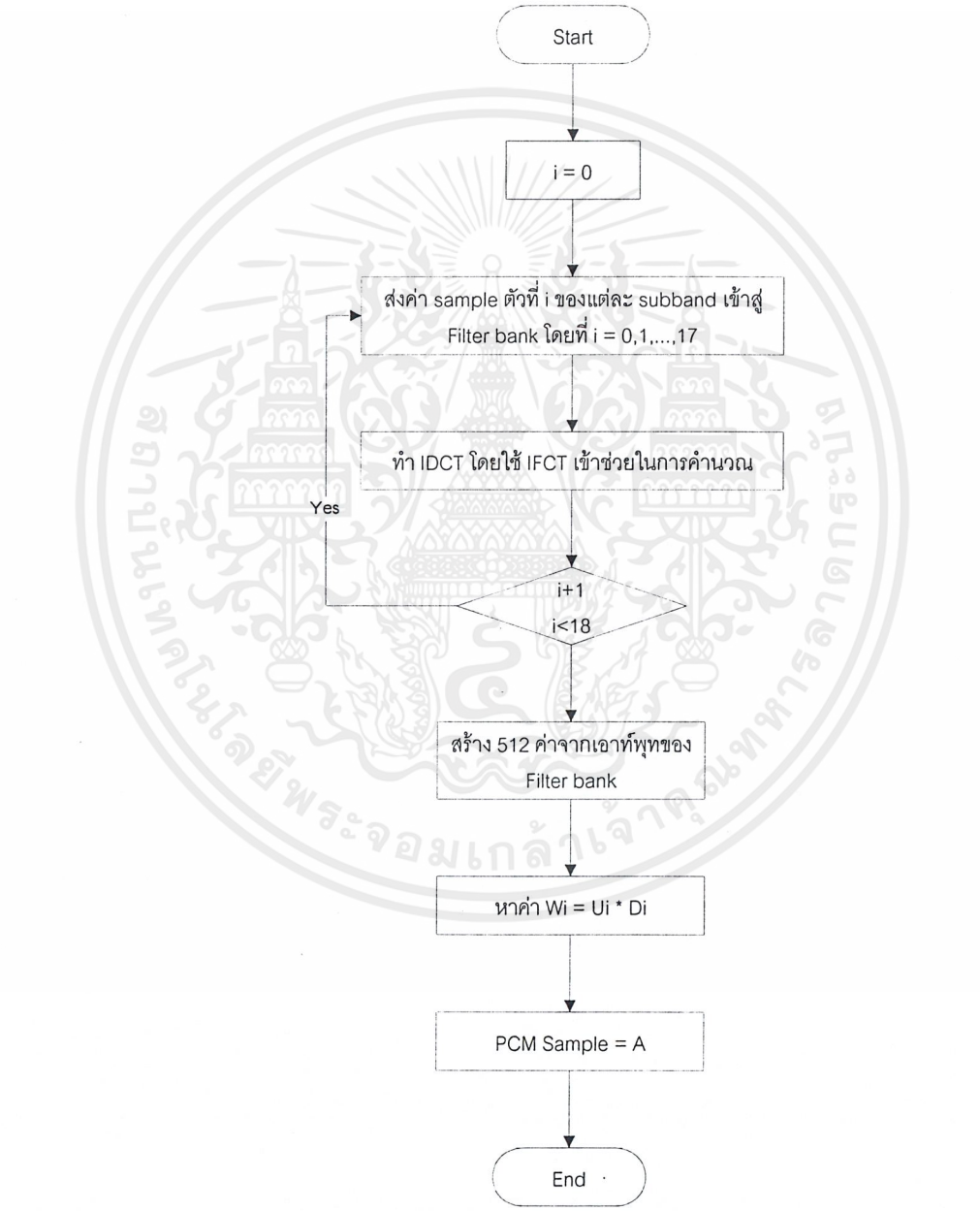
result(i) = Z(i) + S(i) for i= 0 to 17

S(i) = Z(i+18) for i= 0 to 17

รูปที่ 4.4 แสดงการซ้อนทับ (Overlapping) ของสัญญาณ

4.11 การรวมสัญญาณจากแต่ละช่องตัวกรอง (Synthesis Filter Banks)

หัวข้อนี้เป็นขั้นตอนสุดท้ายของกระบวนการถอดรหัส ได้ผลลัพท์เป็นสัญญาณแบบพีซีเอ็ม (PCM Sample) สามารถอธิบายกระบวนการโดยโฟล์วชาร์ตดังรูปที่ 4.6 แสดงโฟล์วชาร์ตแสดงกระบวนการรวมสัญญาณจากแต่ละย่านความถี่ย่อย



รูปที่ 4.5 แสดงโฟลว์ชาร์ตแสดงกระบวนการรวมสัญญาณจากแต่ละย่านความถี่ย่อย

*หมายเหตุ $A = Sj = \sum_{i=0}^{15} W_{j+32i}$

4.11.1 การส่งค่าอินพุทเข้าไปยังโพลีเฟส ฟิลเตอร์แบงค์ (Polyphase Filter banks)

ค่าที่ส่งให้กับโพลีเฟส ฟิลเตอร์แบงค์ คือค่าของข้อมูลสุ่มตัวอย่างตัวที่ i ของทุก ๆ ย่านความถี่ย่อย โดยโพลีเฟส ฟิลเตอร์แบงค์รับค่าทีละ 1 ค่าจากแต่ละย่านความถี่ย่อย (ที่ 32 ย่านความถี่ย่อย) ก็คือ โพลีเฟส ฟิลเตอร์แบงค์รับค่าทีละ 1 ค่าและประมวลผลทีละ 32 ค่า

ในแต่ละย่านความถี่ย่อยมีข้อมูลสุ่มตัวอย่างอยู่ 18 ค่า เพราะฉะนั้น โพลีเฟส ฟิลเตอร์แบงค์จะประมวลผล 18 ครั้งต่อแชลแนลต่อเฟรม

4.11.2 IDCT (Inverse Discrete Transform)

นำ 32 ค่าอินพุทเข้ากระบวนการทำIDCTโดยนำ 32 ค่าอินพุทแทนลงในสมการที่ 4.15

$$x_i = \sum_{k=0}^{31} X_k \cos\left((16+i)(2k+1)\frac{\pi}{64}\right) \quad ; \quad \text{for i = 0 to 63} \qquad \text{สมการที่ 4.15}$$

โดยที่ x_i เป็นเอาท์พุท

X_k เป็นอินพุทจากย่านความถี่ย่อย

4.11.3 การสร้างค่า 512 ค่าจากค่า x_i

นำค่าเอาท์พุท x_i จากหัวข้อ 4.10.2 มาสร้างค่า 512 และกำหนดให้ 512 ค่านี้แทนด้วย U_i โดยที่ i = 0,1,2,.....,511 ดังโปรแกรมด้านล่าง

```
for i = 0 to 7 do
    for j = 0 to 31 do
        U[i*64+j] = x[i*128+j]
        U[i*64+32+j] = x[i*128+96+j]
```

4.11.4 การคูณค่าสัมประสิทธ์ของการสังเคราะห์วินโดว์ (Coefficients D_i of the synthesis window)

นำ U_i จากหัวข้อ 4.10.3 มาคูณด้วยค่า D_i ซึ่งค่า D_i นี้ถูกกำหนดอยู่ในมาตรฐานของเอ็มเป็กดังสมการที่ 4.16 จะได้ค่า W_i ออกมา

$$W_i = U_i D_i \qquad \text{สมการที่ 4.16}$$

4.11.5 การคำนวณค่าของข้อมูลสุ่มตัวอย่างPCM

การคำนวณค่าของข้อมูลสุ่มตัวอย่าง PCM 32 ค่านี้ สามารถหาได้จากสมการที่ 4.17

$$Sj = \sum_{i=0}^{15} W_{j+32i} \qquad \text{สมการที่4.17}$$

ค่า Si ที่ได้ เป็นค่าของการสุ่มตัวอย่างแบบ PCM (PCM Sample) เป็นข้อมูลเสียงแบบเชิงเลข (Digital) จากนั้นค่า 32 ค่านี้จะถูกส่งไปยังอุปกรณ์เสียง (Audio Device) ของตัวถอดรหัส ค่าของการสุ่มตัวอย่างแบบ PCM จะถูกส่งเข้าสู่อุปกรณ์อย่างต่อเนื่อง และเกิดเสียงต่อเนื่องขึ้นที่อุปกรณ์เสียง

บทที่ 5
โปรแกรมแมทแลบ (MATLAB)

5.1 บทนำ

ในการคำนวณเพื่อแก้ไขปัญหาทางด้านวิศวกรรมศาสตร์และวิทยาศาสตร์นั้น ความง่ายและความรวดเร็วเป็นสิ่งจำเป็นอย่างยิ่ง มีภาษาทางคอมพิวเตอร์หลายตัวถูกสร้างขึ้นมา เพื่อเป็นเครื่องช่วยในการแก้ไขปัญหาต่างๆเหล่านั้น เช่น ภาษาฟอร์แทรน(FORTRAN) หรือภาษาซี(C) แต่ภาษาเหล่านี้ จัดเป็นภาษาคอมพิวเตอร์ชั้นสูง (High-Level Language) ค่อนข้างจะยุ่งยากและซับซ้อน เนื่องจากต้องใช้คำสั่งมากมายและต้องมีรูปแบบของคำสั่งที่แน่นอน

สำหรับโปรแกรม MATLAB ย่อมาจาก Matrix Laboratory ซึ่งในอดีตเป็นโปรแกรมที่ถูกสร้างขึ้นมาเพื่อช่วยคำนวณเฉพาะในส่วนที่เกี่ยวข้องกับเมตริกซ์เท่านั้น

แต่ในปัจจุบันโปรแกรม แมทแลบ ได้ถูกพัฒนาขึ้นโดยต่างจากโปรแกรม แมทแลบ ในตอนเริ่มต้นอย่างสิ้นเชิง ซอฟต์แวร์ของ แมทแลบ ได้ถูกพัฒนาขึ้นเพื่อจุดมุ่งหมายสำหรับใช้ในการแก้ปัญหาทางวิศวกรรมศาสตร์และวิทยาศาสตร์ทั่วๆไป ซึ่งทำให้การเขียนโปรแกรมคอมพิวเตอร์สำหรับแก้ปัญหาต่างๆง่ายและไม่ซับซ้อนเหมือนกับการเขียนด้วยภาษาชั้นสูงของคอมพิวเตอร์ทั่วๆไป

5.2 คำสั่งพื้นฐานทั่วไป

โปรแกรม แมทแลบ ต้องการใช้ 2 หน้าต่าง

1. หน้าต่างคำสั่ง (Commander Window)
2. หน้าต่างกราฟ (Graphics Window)

หน้าต่างคำสั่งเป็นหน้าต่างแรกที่พบ เมื่อเปิดโปรแกรม แมทแลบ การป้อนคำสั่งต่าง ๆ จะป้อนลงในหน้าต่างนี้ เพื่อทำการคำนวณและประมวลผลต่างๆ และเป็นหน้าต่างที่ใช้แสดงผลที่ได้จากการประมวลผลต่างๆ ยกเว้นรูปกราฟซึ่งจะแยกไปแสดงผลในหน้าต่างกราฟ

ในการป้อนคำสั่งสำหรับการประมวลผลของโปรแกรม แมทแลบ สามารถป้อนได้ 2 วิธีด้วยกัน คือ หนึ่งเป็นการป้อนคำสั่ง หรือค่าตัวแปรต่างๆ บนหน้าต่างคำสั่งทีละคำสั่งตามขั้นตอน

ของการแก้ปัญหา หรือสองเป็นการกำหนดค่าตัวแปรต่างๆ และชุดคำสั่งต่างๆ ที่ใช้ในการประมวลผลตามขั้นตอนของการคำนวณเพื่อแก้ปัญหาลงในไฟล์ ซึ่งจะต้องเก็บไว้ด้วยนามสกุล .m โดยที่จะเรียกชุดตัวแปรและชุดคำสั่งทั้งหมดที่เขียนในลักษณะนี้ว่า โปรแกรม เอ็ม-ไฟล์ (M-File) หรือเรียกว่า สคริปต์-ไฟล์ (Script-File) ซึ่งหมายถึงไฟล์ที่ถูกเก็บในรูปของรหัส ASCII Code โดย เอ็ม-ไฟล์ หรือ สคริปต์-ไฟล์ นี้ สามารถสร้างมาจากโปรแกรมอีดิเตอร์ (editor) ใดๆ หรือโปรแกรมเวิร์ดโปรเซสเซอร์(Word Processor) ใดๆ ก็ได้ ในการเรียก เอ็ม-ไฟล์ นี้สามารถทำได้อย่างง่าย โดยเพียงแต่พิมพ์ชื่อของ เอ็ม-ไฟล์ โดยไม่ต้องตามด้วยนามสกุล (.m) ลงในหน้าต่างคำสั่ง โปรแกรม แมทแลบ ก็จะไปทำการอ่านคำสั่งใน เอ็ม-ไฟล์ ชื่อนั้นๆ และทำการประมวลผลไปทีละบรรทัดจนจบโปรแกรม

เมื่อเริ่มต้นเข้าสู่โปรแกรม แมทแลบ เราจะพบเครื่องหมาย แมทแลบ Prompt (>>) ซึ่งหมายความว่าโปรแกรม แมทแลบ พร้อมที่จะรอรับคำสั่งต่างๆ สำหรับคำสั่งที่จะออกจากโปรแกรม แมทแลบ จะใช้คำสั่ง quit หรือ exit และก่อนที่จะออกจากโปรแกรม ถ้าต้องการเก็บตัวแปรต่างๆ ที่ได้กำหนดค่าไปแล้ว เราสามารถใช้คำสั่ง save ซึ่งการใช้คำสั่งนี้จะเป็นการเก็บตัวแปรต่างๆ ที่ได้กำหนดไว้แล้วในหน้าต่างคำสั่งไว้ในไฟล์ที่ชื่อ matlab.dat และเมื่อเรากลับมาใช้โปรแกรมอีกครั้ง เราสามารถเรียกตัวแปรต่างๆ ที่เก็บไว้มาทำการประมวลผลต่อได้ด้วยคำสั่ง load

สำหรับผู้ที่ใช้โปรแกรมคำสั่ง แมทแลบ ครั้งแรก โปรแกรม แมทแลบ มีคำสั่งซึ่งช่วยดูลักษณะการใช้งาน และความสามารถในการทำงานของโปรแกรม แมทแลบ และคำสั่งคอมพิวเตอร์ซึ่งจะแสดงชนิดของเครื่อง คอมพิวเตอร์ ที่เรากำลังใช้งานอยู่ในขณะนั้น นอกจากนี้ ยังมีคำสั่ง help ซึ่งจะอธิบายการใช้คำสั่งต่างๆ รวมถึงรูปแบบคำสั่งที่มีใช้ในโปรแกรม แมทแลบ

นอกจากนี้ เอ็ม-ไฟล์ ยังสามารถใช้ในการสร้างฟังก์ชันของ แมทแลบ ที่เหมาะสมเฉพาะงานขึ้นมาใช้ได้อีกต่างหาก

คำสั่งต่อไปจะเป็นคำสั่งสำหรับตรวจสอบเกี่ยวกับ เอ็ม-ไฟล์ โดยถ้าเราต้องการตรวจสอบว่าใน Directory ที่เรากำลังทำงานอยู่นั้นมีโปรแกรม เอ็ม-ไฟล์ ใดๆ อยู่บ้าง ก็สามารถตรวจสอบได้โดยใช้คำสั่ง what ส่วนถ้าต้องการดูรายละเอียดของคำสั่งภายในไฟล์ที่เราต้องการดู ก็สามารถทำได้โดยคำสั่ง type ซึ่งคำสั่ง type นี้จะต้องถูกพิมพ์ตามด้วยชื่อไฟล์ที่เราต้องการดูรายละเอียด ซึ่งในกรณี

นี้ถ้าเรากำหนดชื่อไฟล์ โดยไม่มีนามสกุล โปรแกรม แมทแลบ จะตั้งให้ไฟล์ที่ตามหลังคำสั่ง type นี้เป็นนามสกุล .m โดยอัตโนมัติ

5.3 การใช้งานโปรแกรมแมทแลบ

เมื่อเริ่มต้นใช้โปรแกรม แมทแลบ เพื่อทำการประมวลผลหน้าต่างคำสั่ง สำหรับโปรแกรม แมทแลบ มีค่าตัวแปรต่างๆ และค่าของการประมวลผลต่างๆ แสดงอยู่ในส่วนหน้าต่างกราฟ อาจจะมีรูปกราฟแสดงอยู่ในกรณีที่เราได้มีการใช้คำสั่งวาดกราฟ ถ้าต้องการลบหน้าต่างคำสั่งหรือหน้าต่างกราฟ ให้กลับไปยังจุดเริ่มต้น เมื่อเปิดโปรแกรมใหม่จะมีคำสั่งที่เกี่ยวข้อง 3 คำสั่ง คือ clc (Clear Command Window) เป็นคำสั่งสำหรับลบค่าต่างๆ ที่แสดงในหน้าต่างต่างคำสั่งและให้ แมทแลบ Prompt (>>) ไปอยู่ยังตำแหน่งบนสุดของหน้าต่างคำสั่ง แต่คำสั่งนี้ค่าตัวแปรที่ได้กำหนดไว้ จะต้องใช้คำสั่ง clear ส่วนคำสั่งที่ใช้สำหรับลบภาพในหน้าต่างกราฟคือ clg (Clear Graphic Window)

โปรแกรม แมทแลบ เป็นโปรแกรมที่พัฒนามาจากโปรแกรมที่เกี่ยวข้องกับ เมตริกซ์ ซึ่งในส่วนที่เกี่ยวข้องกับ เมตริกซ์ นั้น ส่วนหนึ่งคือการใช้อักษรตัวใหญ่หรือตัวเล็ก เช่น การใช้ตัวอักษรใหญ่จะหมายถึงเมตริกซ์ ส่วนตัวอักษรเล็กมักจะหมายถึงค่าในเมตริกซ์ โปรแกรม แมทแลบ ได้มีการนำความแตกต่างของตัวอักษรนี้มาใช้ในการกำหนดค่าตัวแปรต่างๆ นั่นคือ ตัวแปรที่เป็นอักษรตัวพิมพ์กับตัวอักษรตัวเขียนนั้น โปรแกรม แมทแลบ จะถือว่าไม่ได้เป็นค่าตัวแปรเดียวกัน กรณีที่เราต้องการให้โปรแกรมเข้าใจว่าตัวแปรที่กำหนดด้วยตัวอักษรพิมพ์ กับตัวแปรที่กำหนดด้วยตัวอักษรเขียนเป็นตัวอักษรเดียวกัน เราต้องใช้คำสั่ง casesen off (Case sensitive off) และถ้าต้องการให้เครื่องกลับไปแยกความแตกต่างของตัวอักษรอีก ให้ใช้คำสั่ง casesen (Case sensitive) ซึ่งโดยปกติค่าที่ตั้งเมื่อเริ่มเปิดโปรแกรมจะเป็นคำสั่ง casesen

ในบางครั้ง เมื่อมีการกำหนดค่าตัวแปรต่างๆ หรือเมตริกซ์ อาจจำเป็นต้องมีกรตรวจสอบค่าหรือขนาดของตัวแปร และเมตริกซ์ใดๆ โปรแกรม แมทแลบ ได้สร้างคำสั่งเพื่อสะดวกในการตรวจสอบตัวแปรและเมตริกซ์ต่างๆ เหล่านี้ไว้ในหลายคำสั่ง โดยคำสั่ง who จะเป็นการสั่งให้โปรแกรมแสดงค่าตัวแปรและเมตริกซ์ต่างๆ ที่ได้กำหนดไว้แล้ว คำสั่ง whos จะแสดงค่าตัวแปร ans ด้วยในกรณีที่มีการใช้ตัวแปรตัวนี้ ตัวแปรที่ชื่อ ans นี้จะเป็นตัวแปรชั่วคราวที่เครื่องกำหนดให้ ในกรณีที่เราทำการประมวลผลและไม่ได้กำหนดให้ผลลัพธ์เป็นค่าตัวแปรใดๆ

ยังมีอีกหนึ่งคำสั่งสำหรับการตรวจสอบขนาดของตัวแปรใดๆ คือคำสั่ง size ซึ่งมีรูปแบบของคำสั่งคือ size(A) โดย A เป็นค่าตัวแปรใดๆ ที่เป็นขนาดของ A

สัญลักษณ์หนึ่งตัวที่ใช้บ่อยในโปรแกรม แมทแลบ คือ % สัญลักษณ์ % เมื่อเป็นตัวแรกของคำสั่งหรือข้อความใดๆจะเป็น เครื่องหมายที่บอกให้เครื่องทราบว่าไม่ต้องทำการประมวลผลใดๆ ต่อคำสั่ง หรือข้อความที่อยู่หลังเครื่องหมายนี้ วัตถุประสงค์ของการใช้เครื่องหมายนี้ก็เพื่อเป็นการเขียนคำอธิบายโปรแกรมในคำสั่งของบรรทัดต่อๆไปที่เราได้เขียนโปรแกรมไว้ ทั้งนี้เป็นการเตือนความจำของเราเองถึงขั้นตอนการทำงานของโปรแกรม หรือผู้ที่จะนำโปรแกรมไปใช้ให้เข้าใจถึงโปรแกรมที่เราเขียนขึ้น

บทที่ 6
ภาษา VHDL

บทนี้จะแนะนำประวัติความเป็นมาอย่างย่อๆ ของภาษา VHDL (VHSIC Hardware Description Language)

6.1 ภาษา VHDL

VHDL ย่อมาจากคำว่า VHSIC Hardware Description Language (VHSIC ย่อมาจาก Very High Speed Integrated Circuit) เป็นภาษาคอมพิวเตอร์ระดับสูง(High Level Language) ซึ่งใช้อธิบายการทำงานของระบบเชิงตัวเลข สามารถอธิบายฟังก์ชันการทำงานได้หลายๆ ระดับ ตั้งแต่ระดับผลงานจนกระทั่งจนถึงระดับวงจรเกท ความซับซ้อนของระบบสามารถจะเขียนได้ตั้งแต่ระดับวงจรเกทประกอบกันจนเป็นระบบที่สมบูรณ์ รูปแบบของภาษา VHDL นั้นจะประกอบไปด้วย 2 ส่วนใหญ่ๆ ได้แก่ ส่วนของ Sequential Language และ Concurrent Language การโปรแกรมด้วยภาษา VHDL สามารถจะเขียนได้ทั้ง 2 รูปแบบรวมกัน เพราะในการทำงานของระบบใดๆ ย่อมจะมีการทำงานในแบบ Sequential และ Concurrent อยู่รวมกัน นอกจากนี้ตัวภาษา VHDL ยังสามารถอธิบายถึงการเชื่อมต่อระหว่างระบบย่อยๆ เข้าด้วยกันเพื่อให้เป็นระบบใหญ่ได้ ตัวภาษา VHDL นอกจากจะกำหนดรูปแบบไวยากรณ์ (Syntax) ของตัวภาษาแล้ว ยังมีการตรวจสอบความหมายของตัวภาษาว่าจะทำการจำลองการทำงานได้หรือไม่ เพราะว่าโปรแกรมที่เขียนโดย VHDL ต้องผ่านการจำลองการทำงานเพื่อตรวจสอบดูการทำงาน ฉะนั้นในการ Compile จะมีการตรวจสอบทั้ง Syntax และ Simulation Semantics อย่างไรก็ตามตัวภาษานั้นถึงจะมีความซับซ้อนมากมายในรูปแบบและกฎเกณฑ์ของภาษาแต่ในการเรียนรู้เพียงบางส่วนของตัวภาษาก็สามารถนำมาใช้งานได้โดยไม่จำเป็นจะต้องศึกษารายละเอียดทั้งหมด เนื่องจากตัวภาษา VHDL ออกแบบมาให้ใช้ออกแบบได้ตั้งแต่วงจรที่มีขนาดเล็กจนถึงวงจรที่มีขนาดใหญ่ซับซ้อน

6.2 ประวัติความเป็นมาของภาษา VHDL

ความต้องการภาษานี้เริ่มจากโครงการ VHSIC ของ Department of Defense (DOD) ของสหรัฐอเมริกาเนื่องจากมีบริษัทที่สร้าง VHSIC Chip หลายบริษัทได้ร่วมโครงการที่จะพัฒนาในขณะนั้นๆ หลายๆ บริษัทใช้ภาษา VHDL ซึ่งแตกต่างกัน ในการอธิบายการทำงาน Chip ของตนด้วยเหตุนี้ทำให้เกิดความแตกต่างแต่ละบริษัทไม่สามารถแลกเปลี่ยนเทคโนโลยีให้กันและกันได้ ทำให้ DOD เกิดปัญหาในการที่จะพัฒนาและซ่อมบำรุงในภายหลัง จึงเกิดความต้องการภาษา VHDL ซึ่ง

เป็นมาตรฐานในการที่จะอธิบายถึงตัวที่ทำการออกแบบนั้นๆ ดังนั้น DOD จึงมอบให้บริษัท IBM, Texas Instrument และ Intermetics 3 บริษัทแรกร่วมกันพัฒนาและกำหนดมาตรฐานของ VHDL ขึ้นมาในปี 1983 หลังจากนั้น VHDL Version 7.2 ได้ทำการพัฒนาและออกเผยแพร่ต่อสาธารณชนในปี 1983 ได้รับความสนใจอย่างมากในอุตสาหกรรมโดยเฉพาะอย่างยิ่งบริษัทที่ทำ VHSIC Chip จากผลสำเร็จนี้ทำให้เกิดมาตรฐาน IEEE ของ VHDL ในปี 1986 ภายหลังจากนั้นก็มีการพัฒนาขยายขีดความสามารถของตัวภาษา VHDL เพิ่มขึ้นจากในวงการอุตสาหกรรม, มหาวิทยาลัย และDOD ก็มีการปรับปรุงและจดมาตรฐานใหม่ IEEE ปี 1987 อีกครั้งหนึ่งซึ่งเป็นที่รู้จักกันในชื่อของ IEEE STD 1076-1987 หลังจากกันยายน 1988 บริษัทใดๆ ที่ทำการพัฒนา ASIC CHIP ใน DOD ของอเมริกาต้องส่งตัว VHDL Model พร้อมกับ TEST BENCH ตามมาตรฐานที่ได้กำหนดเอาไว้

6.3 ความสามารถของภาษา VHDL (CAPABILITY)

หัวข้อดังต่อไปนี้คือความสามารถหลักๆ ของตัวภาษา VHDL

6.3.1 ตัวภาษา VHDL สามารถใช้เป็นสื่อกลางในการแลกเปลี่ยนสื่อสารระหว่างผู้ผลิต Chip กับผู้ออกแบบ (CAD Tools)

6.3.2 ใช้เป็นสื่อกลางในการแลกเปลี่ยนสื่อสารระหว่าง CAE และ CAD Tools เช่น Source Code ของภาษา VHDL สามารถใช้ Compile โดยใช้ Compile&Simulator ได้หลายตัวแตกต่างกัน

6.3.3 ภาษา VHDL สนับสนุนการออกแบบ Top-Down Design และ Bottom-Up Design หรือผสมกันทั้ง 2 แบบ

6.3.4 ตัวภาษา VHDL เป็น Generic คือ ไม่อิงเทคโนโลยีอันใดอันหนึ่ง (แต่สามารถอิงเทคโนโลยีใดก็ได้และในขณะเดียวกัน ก็สามารถสนับสนุนหลายๆ เทคโนโลยี)

6.3.5 สนับสนุนการออกแบบทั้งระบบ Synchronous และ Asynchronous

6.3.6 Algorithmic หรือ Boolean Equation

6.3.7 ตัวภาษาVHDL สามารถอ่านและทำความเข้าใจได้โดยมนุษย์ (Human readable)

6.3.8 ภาษา VHDL เป็นมาตรฐานรับรองโดย IEEE และ ANSI ทำให้ Model ที่ออกแบบโดย VHDL สามารถเคลื่อนย้าย (Portable) ไปยังระบบใดๆ ก็ได้และสามารถนำกลับมาใช้ใหม่ได้ (Reuse)

6.3.9 ภาษา VHDL สนับสนุนรูปแบบการเขียนถึง 3 รูปแบบ ได้แก่ Behavioral Style, Structural Style, Dataflow Style หรือสามารถเขียนรวมกันทั้ง 3 รูปแบบ (Mix Style)

6.3.10 สนับสนุนการออกแบบขนาดใหญ่โดยใช้ความสามารถของ Component, Function Procedure และ Package

6.3.11 ไม่จำเป็นต้องศึกษา Software Simulator เพราะ Simulation Model สามารถเขียนได้ โดยใช้ภาษา VHDL เช่นกัน

6.3.12 สามารถเขียน Model ได้ขนาดไม่จำกัด ไม่มีข้อจำกัดในตัวภาษาเรื่องขนาดModel (ขึ้นอยู่กับ Software Tools)

6.3.13 สามารถอธิบาย Parameter ที่เกี่ยวกับฟังก์ชันทางด้านเวลา เช่น Propagation Delay, Min-Max Delay, Setup, Holding Time, Spike Detection สามารถอธิบายได้ภายในตัวภาษา

6.3.14 GENERICS ช่วยให้เราสามารถสร้าง Parameter ของตัวที่ทำการออกแบบ

6.3.15 Model ที่สร้างด้วยภาษา VHDL นั้น ไม่เพียงแต่จะอธิบายฟังก์ชันการทำงานเท่านั้น แต่ยังสามารถอธิบายถึงรายละเอียดของตัว Model เช่น Total Area และ Speed ของ Model

6.3.16 ภาษา VHDL เป็นมาตรฐานใช้โดยบริษัทและผู้ออกแบบหลายๆ แห่ง ฉะนั้นจึงเป็นการง่ายที่จะทำความเข้าใจ ถึงแม้ว่าจะมาจากแหล่งต่างๆ

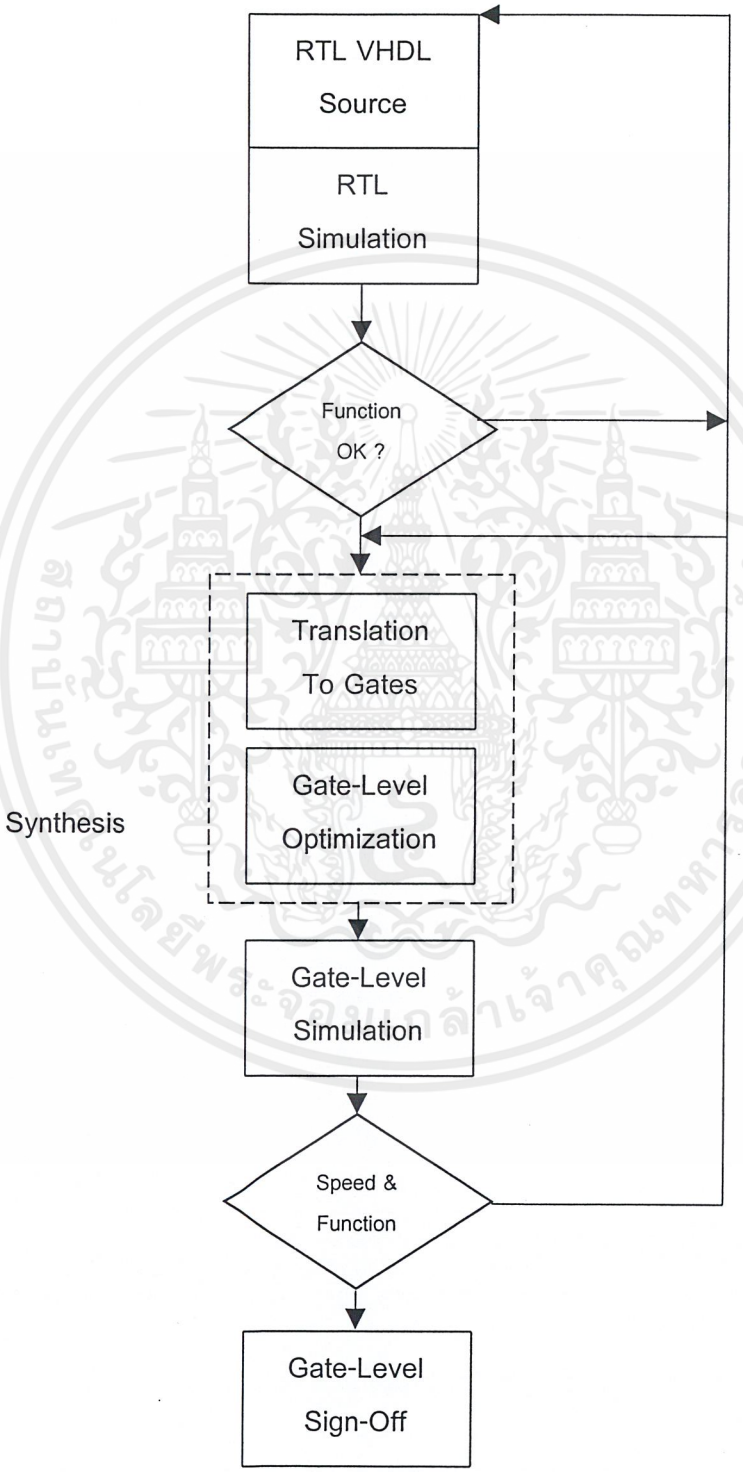
6.3.17 Model ที่สร้างขึ้นสามารถจำลองการทำงานได้ เพราะว่าตัวแปลภาษาได้ตรวจสอบไวยากรณ์ทางด้าน Simulation Semantics ไว้ด้วย

6.3.18 การอธิบาย Model ด้วย Behavioral Style สามารถสังเคราะห์ไปเป็นระดับวงจรเกทได้ ถ้าทำตามกฎของ Synthesis Guideline

6.3.19 มีความสามารถที่ให้เราออกแบบข้อมูลชนิดใหม่ๆ ได้ ทำให้ VHDL Modelเป็นการออกแบบในระดับสูงที่ไม่ต้องคำนึงถึงว่าจะสร้างตัว Model นั้นขึ้นมาได้อย่างไร

6.4 กระบวนวิธีในการออกแบบโดยภาษา VHDL

ขั้นตอนในการออกแบบนั้นสามารถแสดงได้ดังโฟลว์ชาร์ตดังรูปที่ 3.1



รูปที่ 6.1 กรรมวิธีในการออกแบบโดยใช้ Hardware Description Language

ระเบียบวิธีในการออกแบบโดยใช้ Hardware Description Language

6.5 หลักการสร้างโมเดลโดยใช้ภาษา VHDL (General VHDL Modeling Principles)

VHDL เป็นภาษาที่ใช้สำหรับอธิบายการทำงานของฮาร์ดแวร์ในรูปแบบฟอร์มที่อ่านเข้าใจได้ (Human Readable) ซึ่งช่วยในการสร้างและออกแบบวงจรของระบบเชิงตัวเลข (Digital Hardware System, Circuit Board) และอุปกรณ์ต่างๆ อาจใช้อธิบายระบบทั้งระบบหรือ อธิบายเพียงบางส่วน ซึ่งอยู่ในรูปของแผนผังอุปกรณ์ (Component Block) จากนั้นก็ทำการจำลองการทำงานโดยที่ตัวที่ทำการออกแบบนั้นยังไม่ได้สร้างขึ้นจริงหรือเพียงแต่อยู่ในรูปของคำอธิบายเท่านั้น (Textual Format) หลังจากจำลองการทำงานจนได้ตามที่ต้องการจึงนำไปทำการสังเคราะห์เพื่อให้ได้วงจรเกทต่อไป

ประโยชน์จริงๆ ของการใช้ VHDL เป็น Tool แทนการสร้างระบบต้นแบบขึ้นมาจริงแบบเมื่อก่อน ก็คือ เราสามารถอธิบาย Product Idea, Product Proposal, Product Specification เป็นลักษณะในรูปของแฟ้ม Text จากนั้นก็นำไป Compile เพื่อดูผังเวลาการทำงาน จากนั้นก็ทำการ Refine แก้ไขจนกว่าจะได้ Specification ตามต้องการ เมื่อผลิตภัณฑ์ได้ผลตามที่ต้องการแล้วจึงนำไปสู่การสังเคราะห์เพื่อให้ได้ผังวงจรในระดับวงจรเกท แล้วนำไปสร้างเป็นระบบต้นแบบจริงต่อไป ซึ่งระบบต้นแบบที่สร้างนั้นทำงานได้จริงเพราะได้ทำการจำลองการทำงานมาเรียบร้อยแล้ว เป็นการลดเวลาและค่าใช้จ่ายในการสร้างระบบต้นแบบได้มาก

เนื่องจากว่าภาษา VHDL เป็นภาษาที่มีประสิทธิภาพสูง เราจึงใช้ VHDL อธิบายฮาร์ดแวร์เพราะว่ามีข้อดี 2 ประการ คือ

6.5.1 เข้าใจได้ง่าย (Easy To Understand)

6.5.2 สามารถแก้ไขได้ง่าย (Modificable)

การเข้าใจได้ง่ายมีประโยชน์ต่อใครก็ได้ซึ่งมีความจำเป็นที่จะต้องอ่าน Code ที่ได้ออกแบบมาแล้วโดยไม่จำเป็นต้องให้ผู้ออกแบบมาอธิบายให้ฟัง ตัวภาษา VHDL อธิบายการทำงานภายในตัวอยู่แล้ว ส่วนอีกประการหนึ่งก็คือความต้องการในการเปลี่ยนแปลง Hardware ที่ได้ออกแบบแล้วก็คือว่าหลังจากทดสอบแล้วพบข้อที่ผิดพลาดซึ่งต้องแก้ไขหรือว่าระหว่างพัฒนามีการเปลี่ยนแปลงความต้องการของระบบหรือต้องการเพิ่มการทำงานบางส่วน

กรณีอื่นๆ ที่ VHDL มีประโยชน์ก็คือ ตัวภาษานั้นสนับสนุนหลักการต่างๆ ให้เขียน แก้ไข และบำรุงรักษาระบบเชิงตัวเลขที่มีความซับซ้อนเป็นไปได้อย่างรวดเร็ว และมีประสิทธิภาพ และหลักการดังที่กล่าวมาดังนี้

- Top Down Design
- Modularity
- Abstraction

- Information Hiding
- Uniformity

ซึ่งจะอธิบายประโยชน์ของหลักการต่างๆ ในหัวข้อต่อไป และแสดงให้เห็นว่า VHDL นั้นช่วยในการออกแบบระบบเชิงตัวเลขขนาดใหญ่และซับซ้อนนั้นให้อ่านเข้าใจได้ง่ายและแก้ไขได้ง่ายอย่างไร

Top Down Design

ในการพัฒนาระบบเชิงตัวเลขขนาดใหญ่ที่มีความซับซ้อน วิศวกร หรือผู้ออกแบบมักจะมองระบบที่จะออกแบบให้อยู่ในรูปของแผนผังงานเสียก่อน ก่อนที่จะย่อยระบบที่จะออกแบบให้ลึกถึงรายละเอียดต่อไป ซึ่ง VHDL นั้นอนุญาตให้

1 อธิบายการทำงานของแต่ละส่วน

2 วิเคราะห์การทำงาน

3 จัดการแก้ไขและปรับปรุงการทำงานจากผลที่วิเคราะห์เพื่อให้ได้การทำงานตามที่ต้องการ ก่อนที่จะทำการออกแบบให้ละเอียดลึกลงไปในขั้นตอนต่อไป การแก้ไขในขั้นตอนนี้จะทำให้ลดค่าใช้จ่ายกว่าการไปแก้ไขในช่วงของการพัฒนาในระดับสร้าง Silicon Chip

Modularity

Modularity คือหลักการในการแยกส่วน (Partitioning) ฮาร์ดแวร์ออกเป็นส่วนย่อยเล็กๆ ลงไปซึ่งปกติการทำงานของฮาร์ดแวร์ส่วนใหญ่ต้องประกอบด้วยฮาร์ดแวร์ส่วนย่อยๆ ลงไป ซึ่งวงจรระดับเกททั้งหมดจะอยู่ในรูปๆ เดียว (Flatten Design) หลังจากนั้น จะตัดออกเป็นส่วนย่อยๆ เล็กลงมา เมื่อเราออกแบบโดยใช้ VHDL หน้าที่การทำงานของแต่ละส่วนสามารถอธิบายได้โดย Module ของรหัส (คล้าย Function หรือ Procedure) ซึ่งแสดงการทำงานของส่วนย่อยนั้นอย่างชัดเจน ซึ่งการแยกระบบที่จะออกแบบใหญ่ๆ ออกเป็นส่วนย่อยๆ นี้ทำให้ง่ายต่อการจัดการและง่ายต่อการทำความเข้าใจ

ตัวภาษา VHDL ประกอบขึ้นมาด้วย Language Building Block ซึ่งประกอบไปด้วย 75 Reserved Word และมากกว่า 200 Combination Words

Abstraction

คำนิยามของระบบที่จะออกแบบ จะอธิบายการทำงานของตัวระบบมากกว่าที่จะอธิบายถึงว่าจะพัฒนาตัวระบบนั้นอย่างไร หลักการนี้จะมีความสัมพันธ์อย่างใกล้ชิดกับหลักการ Modularity

อีกวิธีการหนึ่งซึ่งแสดงถึงการอธิบายการทำงานของระบบโดยใช้ VHDL ในหลายๆ ระดับของการนิยาม ROM (Read Only Memory) อธิบายโดยใช้ภาษาระดับสูงแสดงถึง

Address ต่างๆ ซึ่งเก็บ DATA ไว้ใน Address นั้นๆ ที่ระดับนี้ไม่ต้องสนใจถึง Address Lime, Data Line และ Control Line เราสามารถเพ่งจุดสนใจไปที่ขนาดของ DATAโดยไม่ต้องคิดถึงสัญญาณควบคุมต่างๆ มากมายภายใน เพราะว่าส่วนนั้นจะถูกจัดการเองในระดับต่ำลงมา ในระดับล่างลงมาเราสามารถอธิบายการทำงานของสัญญาณแต่ละเส้นภายใน ROM ในการจัดการสัญญาณภายในทุกเส้นภายในการที่จะอ่านข้อมูลหรือโปรแกรมข้อมูลใน ROM ถ้าต้องการเปลี่ยนค่าข้อมูลภายใน ROM เราควรขึ้นมาแก้ไขในระดับมีความเหมาะสมแตกต่างกันไป และตรงจุดนี้เองทำให้ระบบที่เราออกแบบง่ายต่อการแก้ไขโดยการใช้ประโยชน์ของ Abstraction

Information Hiding

เมื่อเราทำการเขียนรหัสต้นแบบของภาษา VHDL ขึ้นมาเพื่ออธิบายการทำงานของฮาร์ดแวร์ตัวหนึ่งบางครั้งเราอาจต้องการที่ซ่อนรายละเอียดการพัฒนา Module นั้นๆ โดยไม่ต้องการให้ส่วน Module อื่นๆ รู้การทำงานภายใน Information Hiding มีประโยชน์ก็คือทำให้ระบบที่ออกแบบนั้น สามารถจัดการได้ง่ายและสามารถอ่านและทำความเข้าใจได้ง่ายกว่า หลักการนี้จะใช้สนับสนุนการ Abstraction คือ จะสนใจรายละเอียดในการใช้งานมากกว่าจะสนใจว่าระบบนั้นจะถูกสร้างขึ้นมาอย่างไรบ้างเป็นต้น การซ่อนรายละเอียดภายใน Module ทำให้ความสนใจของผู้ออกแบบสนใจไปในส่วนที่สำคัญมากกว่า ในส่วนที่ไม่น่าสนใจจะซ่อนไว้และเข้าถึงไม่ได้ เช่น คำอธิบายของ Nand Gate นั้นจะถูกปิดบังเอาไว้จากคนที่เขียนอธิบาย Flip-Flop คนที่เขียนอธิบายการทำงานของ Flip-Flop ไม่ต้องสนใจเลยว่า Nand Gate จะทำงานอย่างไรจะต่อกันภายในอย่างไร โดย Nand Gate สามารถเขียนขึ้นมาแล้วคอมไพล์เก็บไว้ใน Library ผู้ที่ออกแบบ Flip-Flop ระดับสูงขึ้นมาเพียงแต่ต้องรู้ว่าจะเชื่อมต่อ Input/Output ของ Nand Gate มาใช้งานได้อย่างไร และประโยชน์อีกอย่างของ Information Hiding ก็คือป้องกันข้อมูลภายในในกรณีที่แจกจ่าย VHDL Module ไปยังที่อื่นๆ เช่นส่งไปให้บริษัทใช้พัฒนาร่มกับ VHDL อื่นๆ โดยเป็นการแจกจ่าย อาจส่งไปแค่ VHDL ที่คอมไพล์แล้วไม่ต้องส่งตัวรหัสต้นแบบไป ทำให้เราป้องกันทรัพย์สินทางปัญญาได้ในอีกระดับหนึ่ง

Uniformility

Uniformility เป็นหลักการอีกอย่างที่ช่วยในการอธิบายระบบด้วยภาษา VHDL นั้นอ่านได้ง่ายขึ้น Uniformility หมายถึงการสร้าง Module ของรหัสในลักษณะคล้ายกัน โดยใช้ตัวภาษา VHDL Building Block ทำให้เกิดการเขียนรหัสต้นแบบที่ดีอย่างเช่น มีการใช้ย่อหน้ามีการใช้ Comment อธิบายเป็นต้นทำให้การพัฒนา Module อ่านและทำการเข้าใจง่าย

6.6 ระดับของการอธิบายระบบ (Level of Abstraction)

การออกแบบโดยใช้ภาษา VHDL Description ก็มีหลายระดับแต่ละระดับก็เหมาะสมและมีข้อเสียต่างกันไปซึ่งรูปแบบของภาษา VHDL มี 3 ระดับดังนี้

-Behavioral Description

-Dataflow Description

-Structural Description

6.6.1 Behavioral Description

เป็นระดับที่เป็นนามธรรมที่สุด โดยภาษา VHDL นั้นมีรูปแบบนี้เป็นการอธิบายการทำงานของระบบคล้ายๆ กับโปรแกรมคอมพิวเตอร์ คือ มี Procedural Form และ Function Form ซึ่งไม่ต้องกล่าวถึงรายละเอียดและการสร้างตัวระบบเลย การใช้ภาษา VHDL รูปแบบนี้เหมาะสำหรับอธิบายระบบที่มีความซับซ้อนมากๆ โดยนักออกแบบไม่ต้องทราบเลยว่าระบบจะประกอบด้วยอุปกรณ์อะไรบ้าง ทำให้ VHDL รูปแบบนี้เหมาะกับผู้ที่ไม่ใช่วิศวกรและผู้ใช้งานทั่วๆ ไปและเป็นการอธิบายตัวระบบที่ดีไปในตัว (Good Documentation)

6.6.2 Dataflow Description

เป็นระดับที่ต่ำกว่า Behavioral คือแสดงการไหลของข้อมูลภายในหน่วยย่อยๆ ของระบบโดยตัวภาษาจะอธิบายการสื่อสารข้อมูลระหว่างหน่วยย่อยๆ นั้น ผู้ออกแบบจะต้องรู้การทำงานของหน่วยย่อยๆ แต่ละหน่วยและรูปแบบของข้อมูลที่จะส่งไปมา การอธิบายในระดับนี้ไม่เหมาะกับผู้ใช้ทั่วๆ ไป แต่จะเหมาะกับระดับช่างเทคนิค หรือวิศวกรเท่านั้น การเขียนอธิบายแบบนี้จะเสียเวลามากกว่าแบบ Behavioral ก็จริงแต่ข้อดีก็คือสามารถที่จะสังเคราะห์ออกมาได้ดีกว่าเพราะแสดงรายละเอียดของระบบออกมาในระดับผังงานซึ่งมากกว่าแบบ Behavioral ซึ่งไม่แสดงอะไรออกมาเลย แต่อย่างไรก็ตามเราจะใช้รูปแบบ Dataflow ในการอธิบายการทำงานของระบบโดยการไหลข้อมูลระหว่าง Register และ Bus

6.6.3 Structural Description

เป็นการอธิบายฮาร์ดแวร์ในระดับต่ำสุดโดยอธิบายถึงการเชื่อมต่ออุปกรณ์แต่ละตัวเพื่อที่จะให้ได้เป็นระบบขึ้นมา ระดับนี้เป็นระดับที่สังเคราะห์ได้ง่ายที่สุดเพราะแสดงรายละเอียดของระบบได้มากที่สุด อย่างไรก็ตามภาษา VHDL ในระดับนี้อธิบายฮาร์ดแวร์ได้ชัดเจนที่สุด บางครั้งเรียกว่า Hardware Description at Gate Level ระดับนี้เข้าใจได้ยากที่สุด แต่เหมาะสำหรับการทำงานที่ต้องการผังเวลาที่ละเอียดที่สุด

6.7 สรุป

6.7.1 ภาษา VHDL คือ ภาษาที่ออกแบบมาเฉพาะเพื่อที่ใช้อธิบายการทำงานของฮาร์ดแวร์ให้อยู่ในรูปที่สามารถอ่านมำความเข้าใจได้ (Human Readable) สามารถอธิบายได้ถึงการจัดระบบและการทำงานของวงจรระดับเกท วงจรระดับ Board และอุปกรณ์ต่างๆ

6.7.2 เหตุผลที่ทำให้ภาษา VHDL ใช้ในการออกแบบและจำลองการทำงานของผลิตภัณฑ์ตัวหนึ่งซึ่งยังไม่ได้สร้างจริงๆ เพื่อดูการทำงานก่อนลงมือสร้าง หรืออาจใช้เป็นตัวแทนแนวคิดในผลิตภัณฑ์นั้นๆ มีดังนี้

1 ภาษา VHDL อนุญาตให้เราออกแบบ, จำลองการทำงาน และทดสอบระบบ โดยใช้รูปแบบของภาษาระดับสูงจนถึงระดับวงจรเกท

2 ภาษา VHDL ถ้าเราเขียนตามรูปแบบของ VHDL นั้นไปทำการสร้างวงจรได้โดยใช้ VHDL Synthesis Guide จะทำให้เราสามารถใช้รหัสภาษา VHDL นั้นไปทำการสร้างวงจรได้โดยใช้ VHDL Synthesis Tools

3 เพราะว่าภาษา VHDL เป็นภาษาที่กำหนดเป็นมาตรฐาน IEEE 1076-1987 วิศวกร หรือผู้ออกแบบทั่วๆ ไป สามารถใช้ภาษานี้ในการพัฒนาได้เหมือนกัน ลดปัญหาความเข้ากันไม่ได้ลงไป

4 VHDL มีคุณสมบัติที่ดีที่ทำให้เราสามารถเขียนและแก้ไขระบบเชิงตัวเลขที่มีขนาดใหญ่และซับซ้อนได้อย่างสะดวกรวดเร็วและมีประสิทธิภาพ ดังนี้คือ

6.7.3 Top Down Design วิธีการนี้ ทำให้เราสามารถอธิบายการทำงานของระบบได้ในลักษณะของ Block ใหญ่ๆ จากนั้นทำการวิเคราะห์จำลองการทำงานและแก้ไขให้ได้คุณสมบัติตามที่เราต้องการ ณ. ระดับ Block ก่อนที่จะลงลึกในระดับต่ำต่อไป

6.7.4 Modularity วิธีการที่แยกส่วน (หรือการประกอบส่วนย่อยๆ ขึ้นมา) ระบบที่เราออกแบบเป็นส่วนย่อยๆ เล็กๆ ออกมา

6.7.5 Abstraction เป็นรายละเอียดใน Module ซึ่งอธิบายการทำงานของ Module มากกว่าที่จะอธิบายถึงการพัฒนาและการสร้าง Module นั้น

บทที่ 7
การทดลองและผลการทดลอง

7.1 การทดลองการถอดรหัสข้อมูลเสียงตามมาตรฐานเอ็มเป็ก1 เลเยอร์ 3 โดยใช้โปรแกรมแมทแลบ

ขั้นตอนการทดลอง

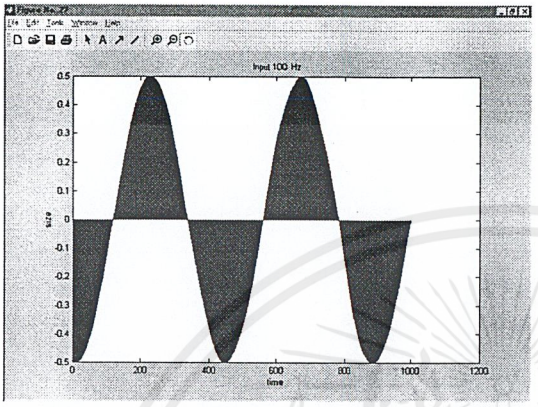
1. สร้างไฟล์เสียง(*.wav) โดยสร้างไฟล์เสียงที่มีความถี่ 100 Hz , 11 kHz และ 20 kHz ให้มีความถี่ในการสุ่มเท่ากับ 44.1 kHz
2. นำไฟล์เสียงที่ได้ไปเข้ารหัสเอ็มเป็ก(mp3) ด้วยโปรแกรม Shuffler Music Converter เลือก 8 bit mono 44.1kHz frequency sampling 56kbit/s
3. นำไฟล์เอ็มเป็กที่ได้(*.mp3) มาถอดรหัสได้ดังนี้
   3.1 แปลงไฟล์ *.mp3 ให้อยู่ในรูปของขบวนบิตด้วยโปรแกรมแมทแลบ bitstreams
   3.2 เมื่อได้ขบวนบิตดังกล่าว ให้ save workspace ไว้ แล้วรันโปรแกรม set1 เป็นการเซ็ทค่าเริ่มต้นให้กับการถอดรหัส
   3.3 เริ่มทำการถอดรหัส โดยรันโปรแกรม inform โปรแกรมจะแสดงผลในแต่ละขั้นตอนจนกระทั่งได้สัญญาณ PCM ออกมา
4. บันทึกผลการทดลองที่ได้จากขั้นตอนการถอดรหัสในรูปของกราฟที่ความถี่ต่างๆ เพื่อนำกราฟที่ได้มาเปรียบเทียบกับกราฟของสัญญาณเสียงก่อนที่จะนำมาเข้ารหัสดังผลการทดลอง

   กราฟที่ทำการบันทึกมีทั้งหมด 10 กราฟ ต่อ สัญญาณเสียง 1 ความถี่ ได้แก่

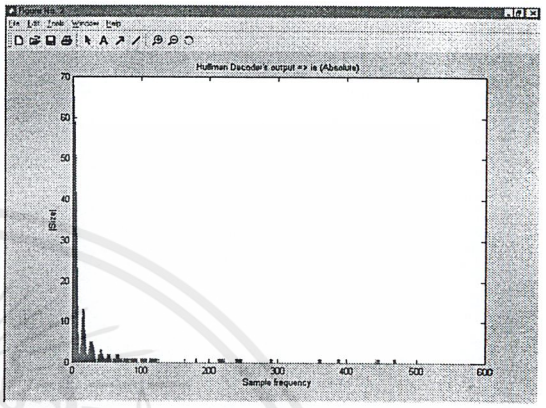
   1. กราฟสัญญาณเสียงก่อนที่จะนำมาเข้ารหัสที่ความถี่ 100 Hz, 11 kHz, 20 kHz
   2. กราฟเอาท์พุทที่ได้จากการถอดรหัสฮัฟแมน abs(is)
   3. กราฟเอาท์พุทที่ได้จากการรีควอนไตซ์ abs(xr)
   4. กราฟเอาท์พุทที่ได้จากการลดค่าปลอม
   5. กราฟเอาท์พุทที่ได้จากการทำ IMDCT
   6. กราฟเอาท์พุทที่ได้จากการทำ IMDCT พิจารณาเฉพาะขนาด และย่านความถี่ย่อย
   7. กราฟเอาท์พุทที่ได้จากการทำวินโดว์
   8. กราฟเอาท์พุทที่ได้จากการทำโอเวอร์แลบ
   9. กราฟเอาท์พุทที่ได้จากการทำ Polyphase filterbank
   10. กราฟเอาท์พุทของสัญญาณเสียงที่ได้หลังจากผ่านการถอดรหัสเรียบร้อยแล้ว (เฉพาะเฟรมแรกเท่านั้น)

7.2 ผลการทดลองในส่วนแมทแลบ

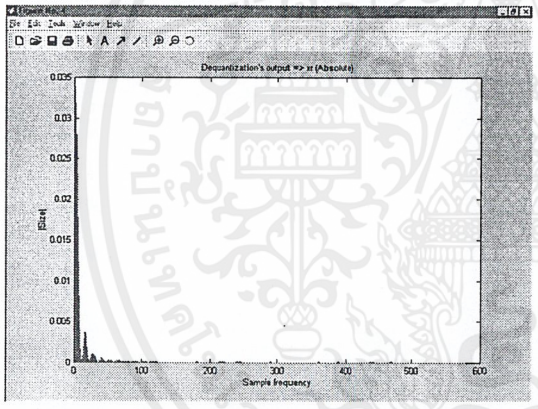
7.2.1 ทำการทดลองด้วยไฟล์เสียงที่มีความถี่ 100 Hz



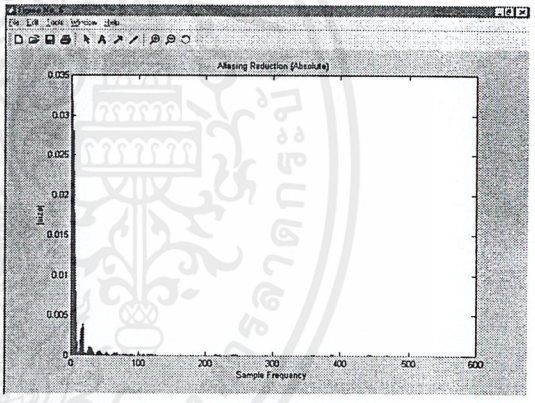
Input 100 Hz



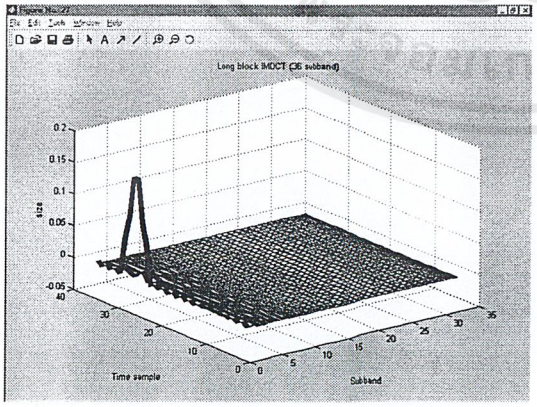
Huffman Decode



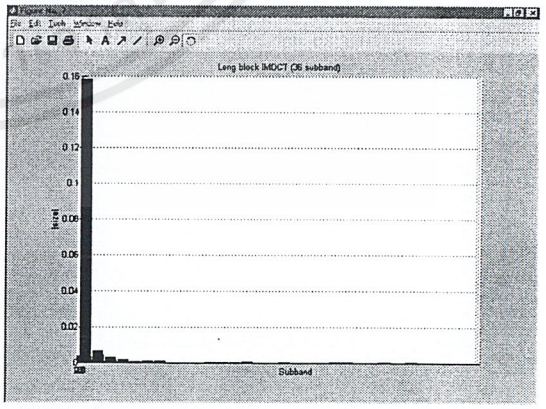
Descaling



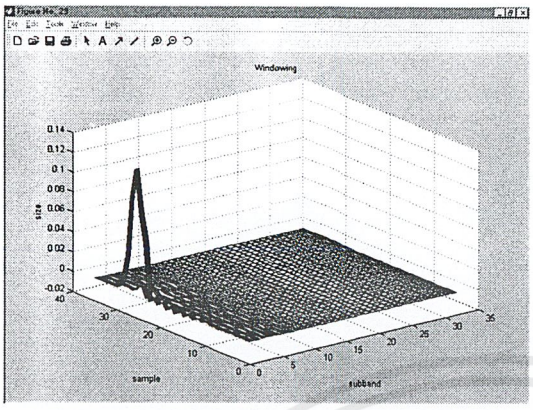
Aliasing Reduction



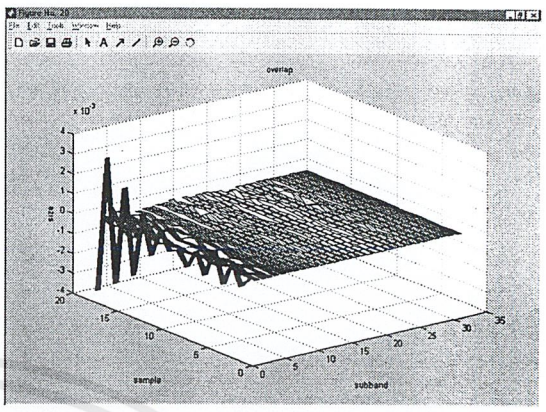
IMDCT



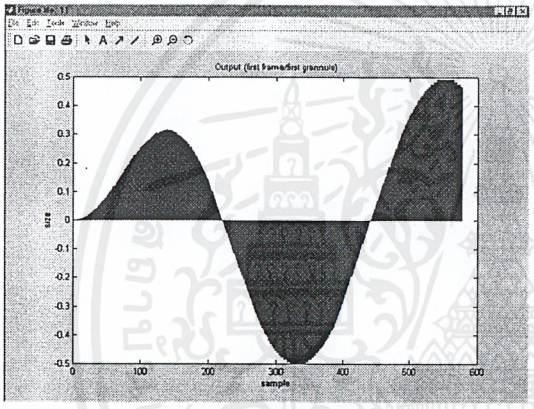
IMDCT (พิจารณาเฉพาะขนาดและsubband)



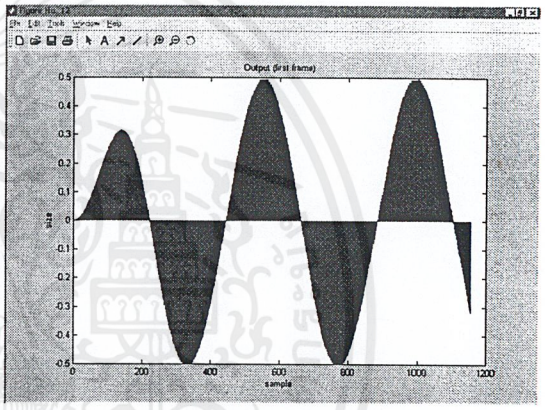
Windowing



Overlap

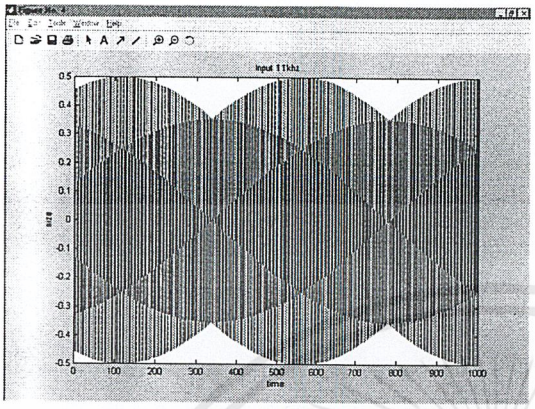


Polyphase Filterbank

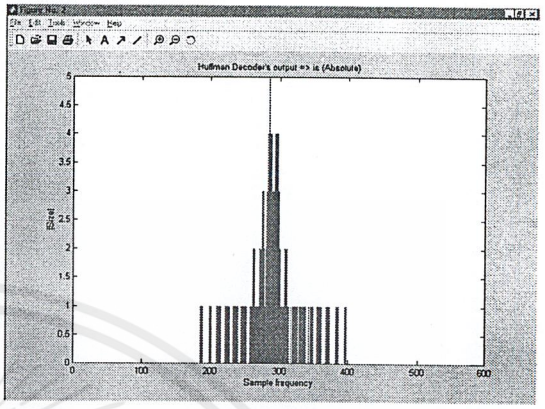


Output (เฟรมแรก)

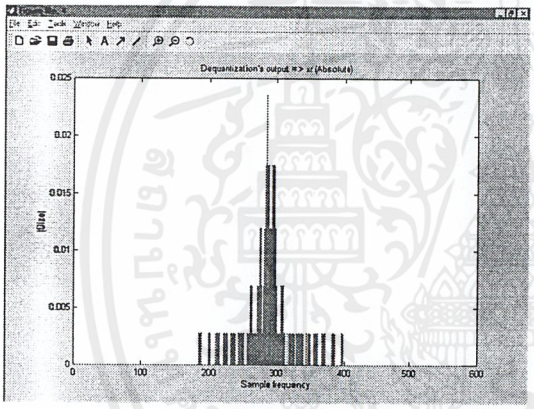
7.2.2 ทำการทดลองด้วยไฟล์เสียงที่มีความถี่ 11 kHz



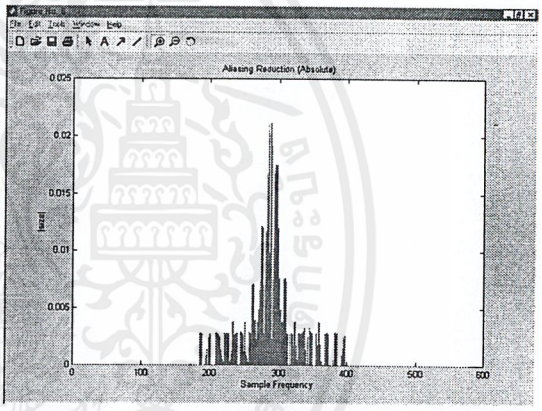
Input 11 kHz



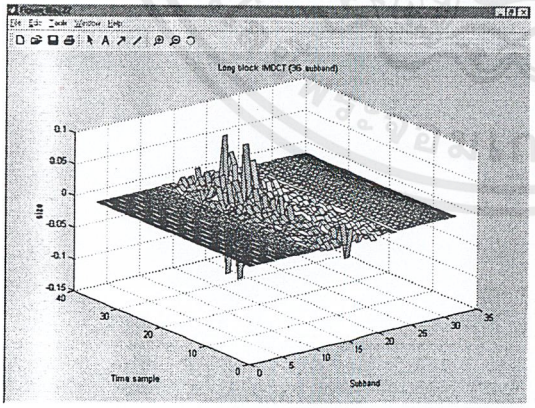
Huffman Decode



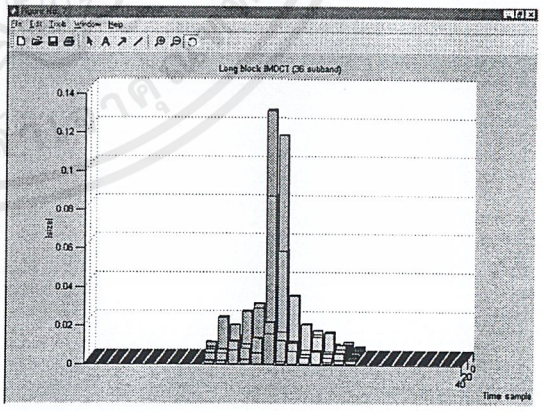
Descaling



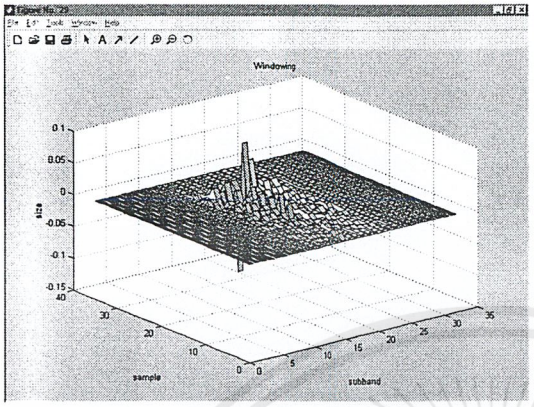
Aliasing Reduction



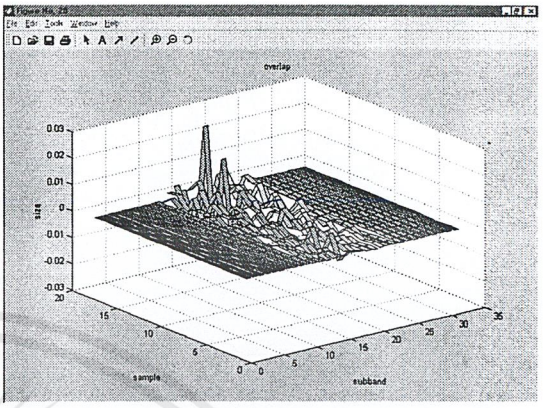
IMDCT



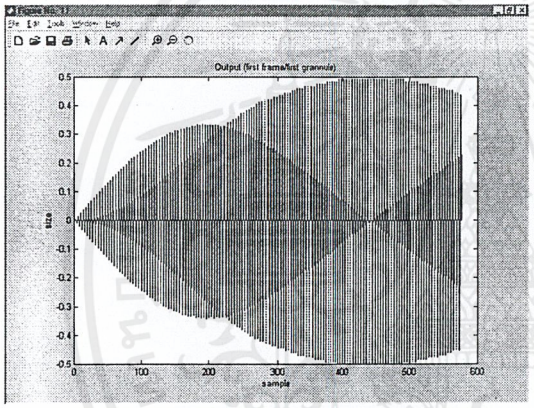
IMDCT (พิจารณาเฉพาะขนาดและsubband)



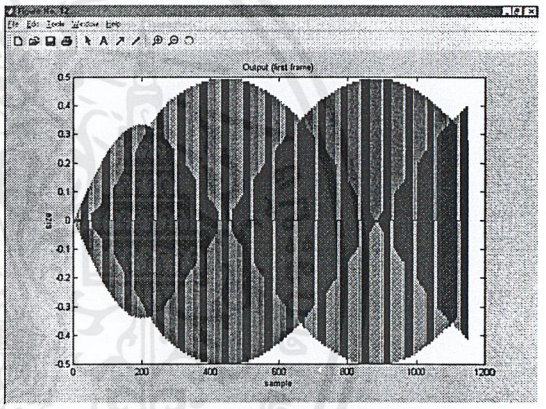
Windowing



Overlap

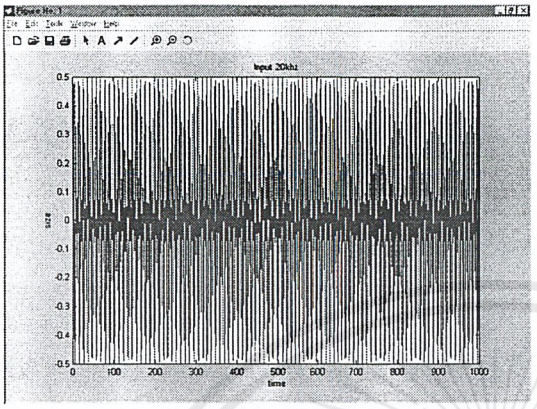


Polyphase Filterbank

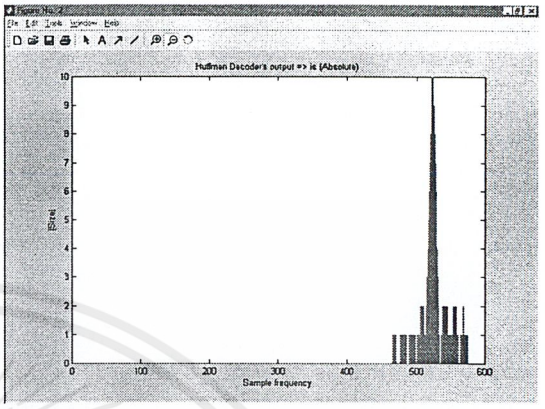


Output (เฟรมแรก)

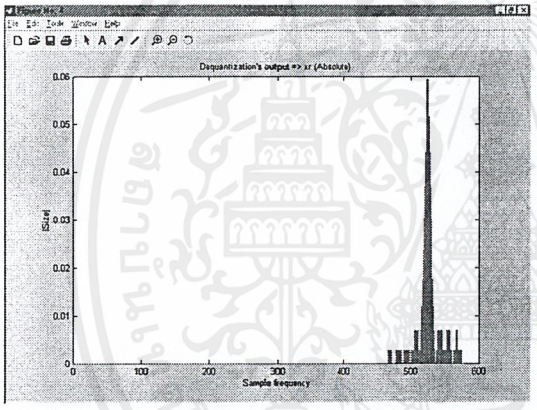
7.2.3 ทำการทดลองด้วยไฟล์เสียงที่มีความถี่ 20 kHz



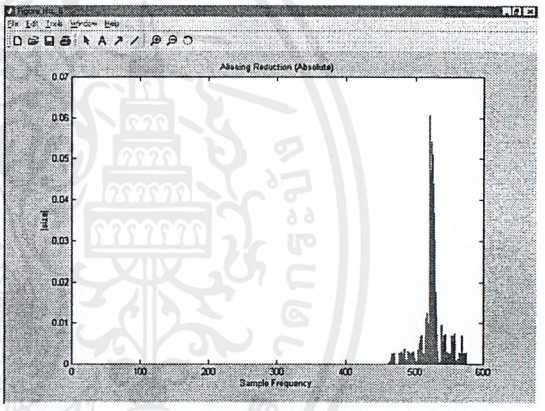
Input 20 kHz



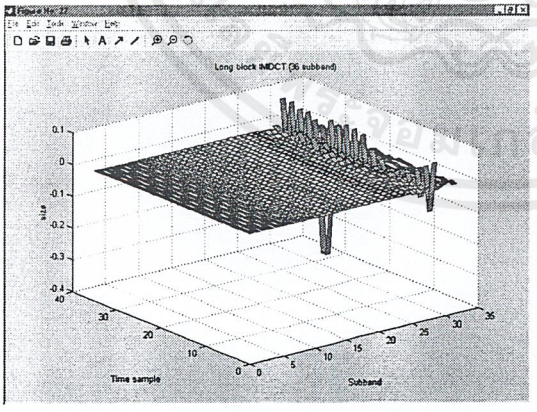
Huffman Decode



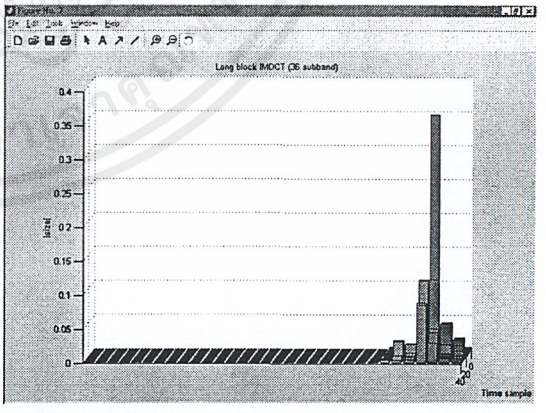
Descaling



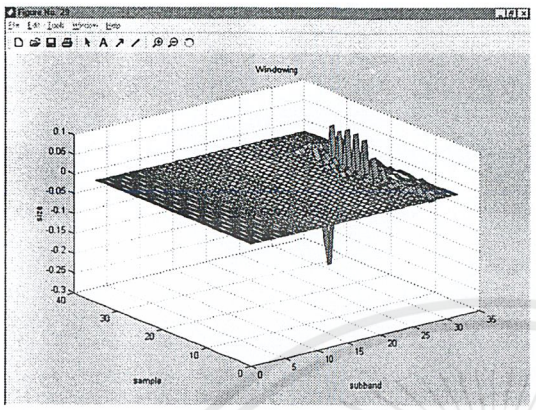
Aliasing Reduction



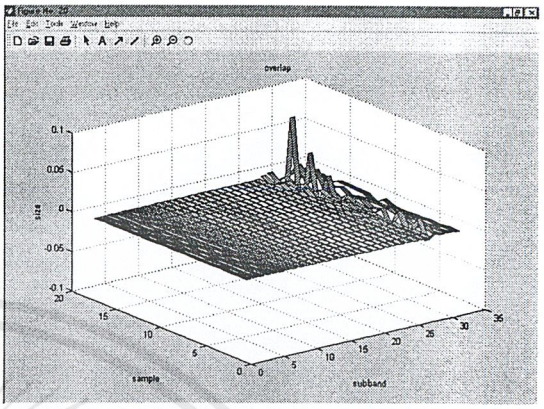
IMDCT



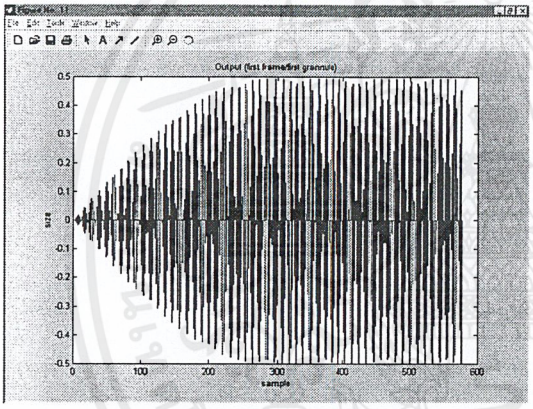
IMDCT (พิจารณาเฉพาะขนาดและsubband)



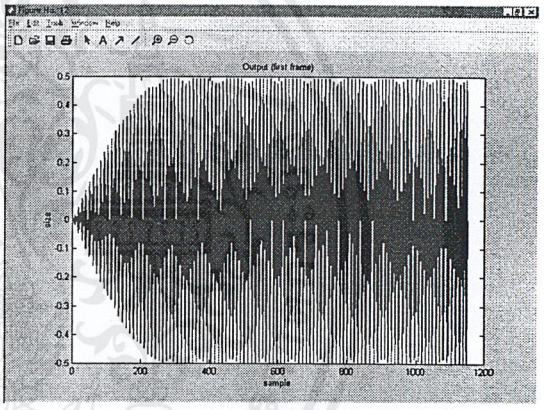
Windowing



Overlap



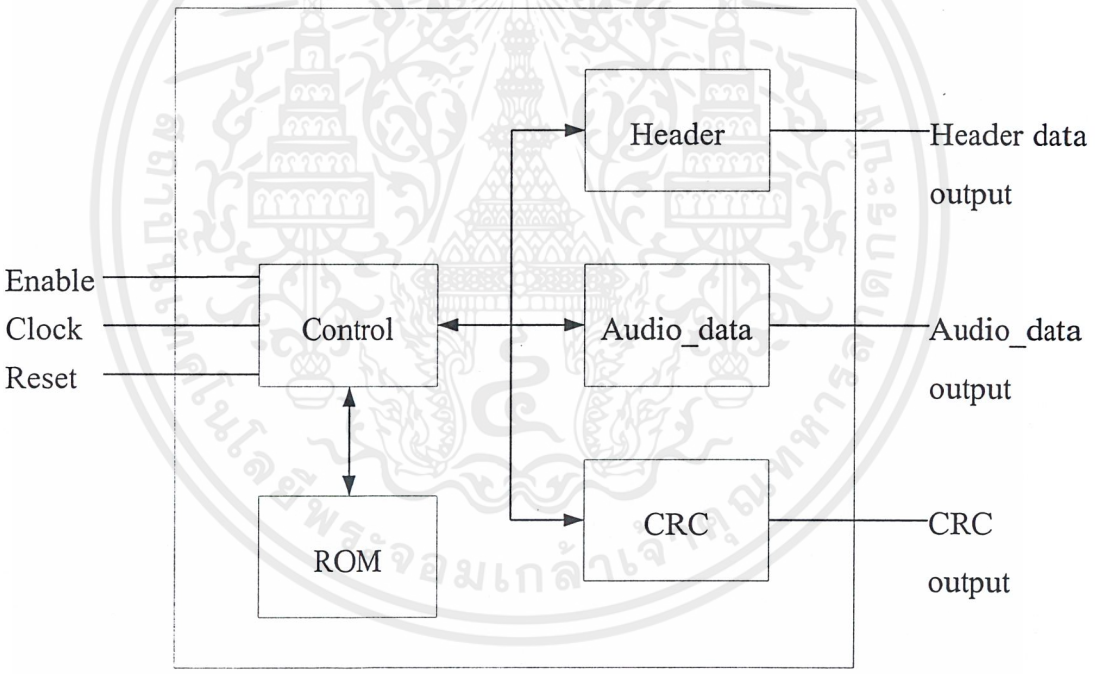
Polyphase Filterbank



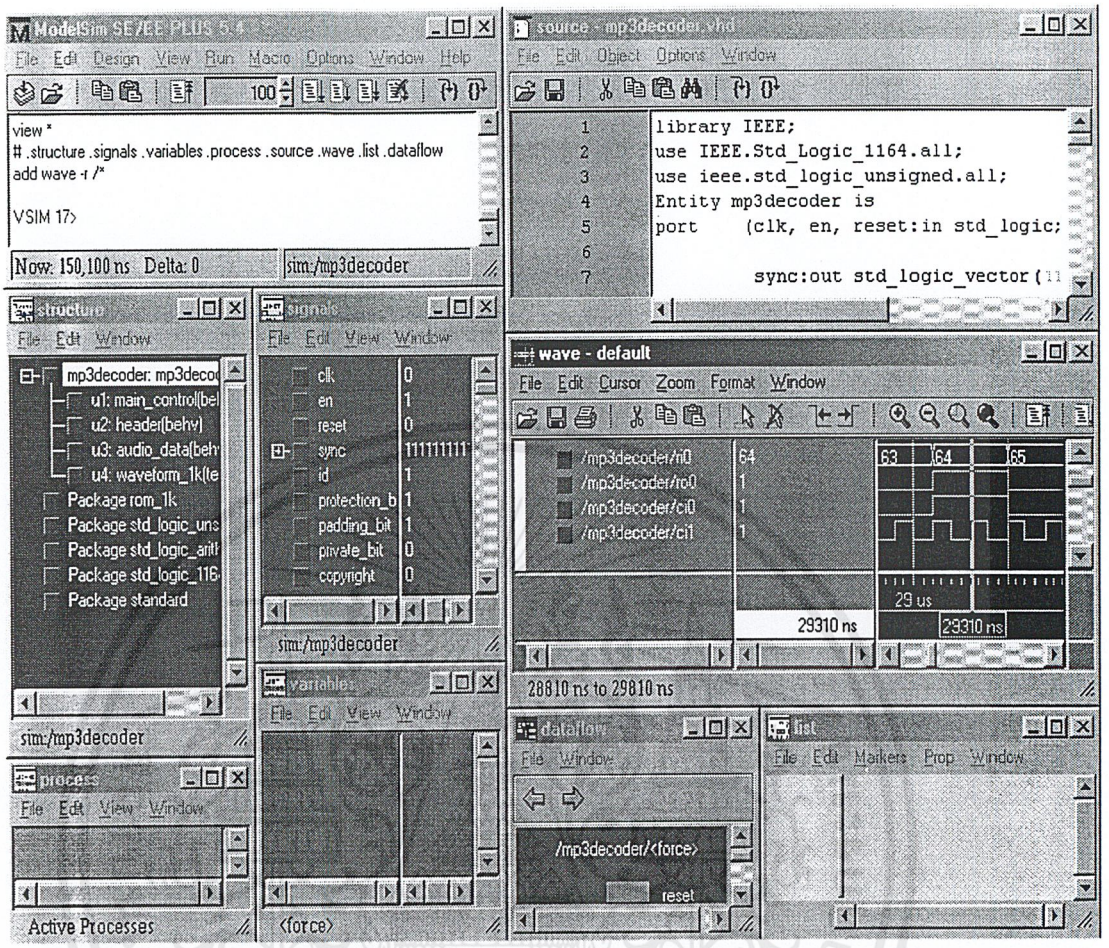
Output (เฟรมแรก)

7.3 การทดลองวงจรรวมที่ออกแบบด้วยภาษา VHDL

เราจะทำการทดลองหาผลลัพธ์ที่เกิดขึ้นจากวงจรรวมที่ออกแบบด้วยภาษา VHDL โดยใช้โปรแกรมโมเดลซิม ซึ่งจะยกตัวอย่างแสดงในส่วนของการเก็บค่าส่วนหัวข้อมูลและส่วนของการถอดรหัสสเกลแฟคเตอร์ซึ่งประกอบไปด้วยกระบวนการตรวจจับค่าซิงค์เวิร์ด เพื่อบอกการเริ่มต้นเก็บข้อมูล,กระบวนการเก็บข้อมูลพื้นฐานและการถอดรหัสสเกลแฟคเตอร์รวมทั้งการตรวจสอบความผิดพลาดด้วย CRC Check โดยกระบวนการทั้งหมดที่กล่าวมานี้จะประกอบไปด้วยส่วนประกอบหลักๆคือ ส่วนการควบคุม,ส่วนตรวจจับค่า Syncword และ ค่า Header, ส่วนถอดรหัสสเกลแฟคเตอร์, ส่วนตรวจสอบค่า CRC และ ROM ที่เราสมมติขึ้นมาเพื่อใส่ค่าข้อมูลส่วนหัวของข้อมูล MP3 ไว้ ซึ่งจะนำมาต่อรวมกันได้ดังรูป



รูปที่ 7.1 บล็อคไดอะแกรมแสดงการเชื่อมต่อกันของฮาร์ดแวร์



รูปที่ 7.2 แสดงหน้าต่างโปรแกรมโมเดลซิมที่ใช้ในการทดลองวงจรรวม
ที่ออกแบบโดยภาษา VHDL

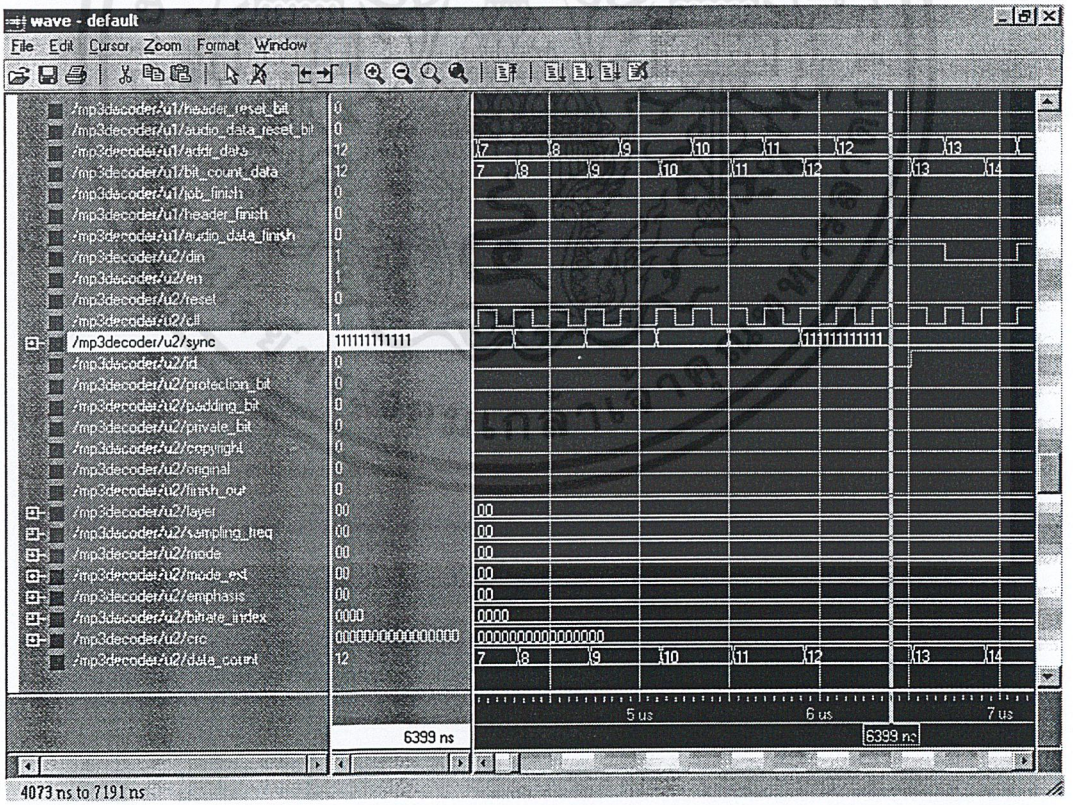
การทำงานของอุปกรณ์ต่างๆ

เริ่มต้นเมื่อได้รับสัญญาณให้ทำงานได้ (Device enable) ระบบก็จะเริ่มต้นทำงานโดย ส่วนควบคุม(Control) จะส่งสัญญาณให้ส่วนของ Header ทำงานก่อน Header ก็จะทำการตรวจจับ Syncword คือ 1 ทั้งหมด 12 บิตเรียงกันของข้อมูลบิตสตรีมที่ไหลเข้ามา เมื่อพบก็จะเริ่มต้นทำการเก็บข้อมูล Header ทั้งหมดดังต่อไปนี้

1. sync word 12 บิต
2. id 1 บิต
3. layer 2 บิต
4. protection_bit 1 บิต
5. bitrate_index 4 บิต

6. sampling_frequency 2 บิต
7. pading_bit 1 บิต
8. private_bit 1 บิต
9. mode 2 บิต
10. mode_extension 2 บิต
11. copyright 1 บิต
12. original/copy 1 บิต
13. emphasis 2 บิต

หลังจากนั้นก็จะทำการตรวจสอบค่า protection_bit เพื่อดูว่า ข้อมูลที่ได้มามีการใส่รหัสตรวจสอบความผิดพลาดมาด้วยหรือไม่ ถ้าเป็น 0 ก็แสดงว่ามีการใส่รหัสมาด้วย ก็จะทำการเก็บค่า CRC ทั้งหมด 16 บิต แต่ถ้าไม่มีการใส่รหัสมาก็จะข้ามขั้นตอนนี้ไป แล้วจะทำการส่งสัญญาณแจ้งส่วนควบคุมว่าทำงานเสร็จสิ้นแล้ว



รูปที่ 7.3 แสดงรูปสัญญาณการตรวจพบซิงค์เวิร์ด

เมื่อส่วนควบคุมได้รับสัญญาณจาก Header แล้ว ก็จะทำการส่งสัญญาณให้ส่วนถอดรหัสสเกลแฟคเตอร์ (audio_data) ทำงานต่อ โดยการทำงานของ audio_data จะทำการตรวจสอบข้อมูล id และ layer ที่ได้จากหัวข้อมูลว่าข้อมูลที่ได้มาเป็นข้อมูลที่เข้ารหัสมาตามมาตรฐาน เอ็มเป็ก 1 เลเยอร์ 3 หรือไม่ ถ้าไม่ก็จะไม่ทำการเก็บข้อมูล แต่ถ้าใช่ก็จะทำการเก็บข้อมูลสเกลแฟคเตอร์ทั้งหมดดังนี้

1. เริ่มจากเก็บส่วนที่เป็นข้อมูลรวมของทั้ง 2 แกรนูล

main_data_begin	9	บิต
private_bits	5	บิต
scfsi_band0	1	บิต
scfsi_band1	1	บิต
scfsi_band2	1	บิต
scfsi_band3	1	บิต

2. ต่อจากนั้นก็จะทำการเก็บข้อมูลสเกลแฟคเตอร์ของแกรนูล 0 ดังนี้

part2_3_length_gr0	12	บิต
big_values_gr0	9	บิต
global_gain_gr0	8	บิต
scalefac_compress_gr0	4	บิต
window_switching_flag_gr0	1	บิต

3. แล้วทำการตรวจสอบ window_switching_flag ดูว่ามีการเปลี่ยนชนิดของวินโดว์ที่เข้ารหัสมาหรือไม่ ถ้าเป็น 1 แสดงว่าเปลี่ยน ก็จะทำการเก็บข้อมูลดังต่อไปนี้

block_type_gr0	2	บิต
mixed_block_flag_gr0	1	บิต
table_select_gr0_region0	5	บิต
table_select_gr0_region2	5	บิต
table_select_gr0_region2	5	บิต
subblock_gain_gr0_window0	3	บิต
subblock_gain_gr0_window1	3	บิต
subblock_gain_gr0_window2	3	บิต

แต่ถ้าเป็น 0 ก็จะทำการเก็บข้อมูลต่อไปนี้

table_select_gr0_region0	5	บิต

table_select_gr0_region2 5 บิต

table_select_gr0_region2 5 บิต

region0_count_gr0 4 บิต

region1_count_gr0 3 บิต

แล้วทำการเก็บข้อมูลต่อไปนี้

preflag_gr0 1 บิต

scalefac_scale_gr0 1 บิต

count1table_select_gr0 1 บิต

4.ต่อไปก็จะทำการเก็บข้อมูลสเกลแฟคเตอร์ของแกรนูล 1 โดยมีขั้นตอนเช่นเดียวกันกับของแกรนูล 0 โดยเริ่มต้นจากข้อมูลหลัก

part2_3_length_gr1 12 บิต

big_values_gr1 9 บิต

global_gain_gr1 8 บิต

scalefac_compress_gr1 4 บิต

window_switching_flag_gr1 1 บิต

- ถ้า window_switching_flag เป็น 1

block_type_gr1 2 บิต

mixed_block_flag_gr1 1 บิต

table_select_gr1_region0 5 บิต

table_select_gr1_region1 5 บิต

table_select_gr1_region2 5 บิต

subblock_gain_gr1_window0 3 บิต

subblock_gain_gr1_window1 3 บิต

subblock_gain_gr1_window2 3 บิต

- ถ้าเป็น 0

table_select_gr1_region0 5 บิต

table_select_gr1_region1 5 บิต

table_select_gr1_region2 5 บิต

region0_count_gr1 4 บิต

region1_count_gr1 3 บิต

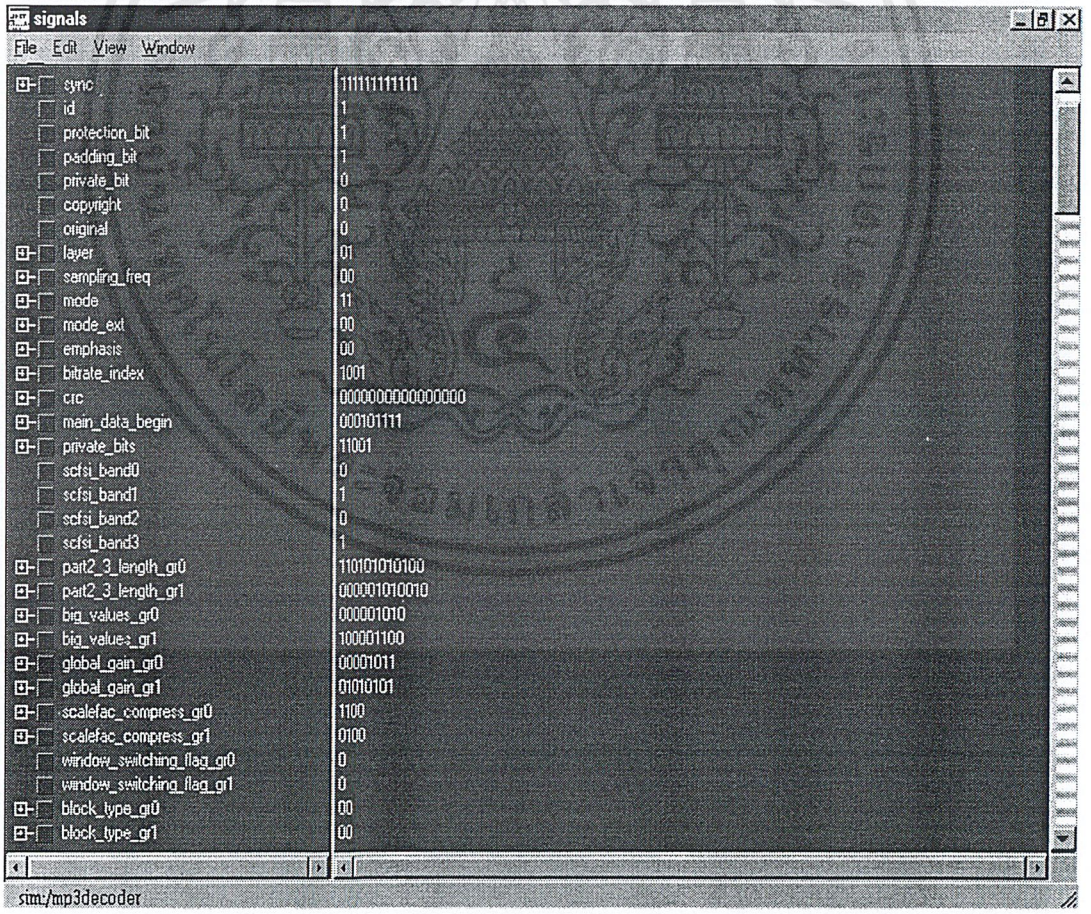
5. จากนั้นเก็บข้อมูลต่อไปนี้

preflag_gr1 1 บิต

scalefac_scale_gr1 1 บิต

count1table_select_gr1 1 บิต

หลังจากนั้น ก็จะทำการส่งสัญญาณ job-finish ไปบอกส่วนควบคุมว่าทำงานเสร็จสิ้นแล้วส่วนควบคุมก็จะทำการส่งสัญญาณแจ้งออกมาว่างานของส่วนการจับ header และถอดรหัสสเกลแฟคเตอร์ได้เสร็จสิ้นแล้ว หลังจากนั้นก็จะส่งข้อมูลทั้งหมดไปทำการถอดรหัส MP3 ในกระบวนการต่อๆไป



รูปที่ 7.4 แสดงค่าของข้อมูลที่เก็บได้

7.4 ผลการทดลองในส่วนของ VHDL

เราจะทำการทดลองโดยจะใช้ค่าที่เก็บเอาไว้ใน ROM เป็นอินพุท (input bitstream) ให้แก่ระบบ ซึ่งจะทำการทดลองกับข้อมูล MP3 2 แบบที่แตกต่างกันคือเป็นเสียงที่มีความถี่ขนาด 1KHz และ 10KHz ที่ถูกสุ่มด้วยความถี่ 44.1 KHz มีข้อมูลขนาด 8 บิต และมี 1 ช่องเสียง (MONO) เพื่อลดจำนวนข้อมูลที่จะใช้แสดงและระยะเวลาที่ใช้ในการทดลอง

รูปแบบบิตสตรีมของข้อมูลที่ป้อนให้เป็นอินพุท

- บิตสตรีมของเสียงที่มีความถี่ 1 KHz

  1111111111111011100100101100000000000000000000000010111010010001011111100111100000101011010110000000000000000010110100110100100000101001000000110010101011010000000000010011001010100101010011001011100101110011001011100100111000110100000011010111010100011000010000010110001101001100110100001101111101000011111010

- บิตสตรีมของเสียงที่มีความถี่ 10 KHz

  1111111111111011100100101100000000000000000000000010111111001010111010101010000000101000001011110000000000010110010001111001000001010010100001100101010101000000000001111100100000000000000000000000011100100000000000000000000000000111010010100101001010011111001101111101111011110100110101111011101111101111

ผลการทดลองถอดเสียงที่มีความถี่ 1 KHz และ 10 KHz ได้ผลลัพธ์ดังต่อไปนี้

ข้อมูลสเกลแฟคเตอร์	เสียงความถี่ 1 KHz	เสียงความถี่ 10 KHz
Syncword	111111111111	111111111111
id	1	1
layer	01	01
protection_bit	1	1
bitrate_index	1001	1001
sampling_frequency	00	00
pading_bit	1	1
private_bit	1	0
mode	11	11
mode_extension	00	00
copyright	0	0
original/copy	0	0
emphasis	00	00
CRC	0000000000000000	0000000000000000
main_data_begin	000101110	000101111
private_bits	10010	11001
scfsi_band0	0	0
scfsi_band1	0	1
scfsi_band2	1	0
scfsi_band3	0	1
part2_3_length_gr0	111111001111	110101010100
big_values_gr0	000001010	000001010
global_gain_gr0	11010110	00001011
scalefac_compress_gr0	0000	1100
window_swithching_flag_gr0	0	0
block_typr_gr0	00	00
mixed_block_flag_gr0	0	0
table_select_gr0_region0	00000	00000

ข้อมูลสเกลแฟคเตอร์	เสียงความถี่ 1 KHz	เสียงความถี่ 10 KHz
table_select_gr0_region1	00000	00010
table_select_gr0_region2	10110	11001
subblock_gain_gr0_window0	000	000
subblock_gain_gr0_window1	000	000
subblock_gain_gr0_window2	000	000
region0_count_gr0	1001	0001
region1_count_gr0	101	111
preflag_gr0	0	0
scalefac_scale_gr0	0	0
count1table_select_gr0	1	1
part2_3_length_gr1	000001010010	000001010010
big_values_gr1	000001100	100001100
global_gain_gr1	10101011	01010101
scalefac_compress_gr1	0100	0100
window_swithching_flag_gr1	0	0
block_typr_gr1	00	00
mixed_block_flag_gr1	0	0
table_select_gr1_region0	00000	00000
table_select_gr1_region1	00010	00011
table_select_gr1_region2	01100	11100
subblock_gain_gr1_window0	000	000
subblock_gain_gr1_window1	000	000
subblock_gain_gr1_window2	000	000
region0_count_gr1	1010	1000
region1_count_gr1	100	000
preflag_gr1	1	0
scalefac_scale_gr1	0	0
count1table_select_gr1	1	0

บทที่ 8
สรุปผลการทดลองและการดำเนินงาน

8.1 สรุปผลการทดลองในส่วนของแมทแลบ

จากผลการทดลองการถอดรหัสข้อมูลเสียงตามมาตรฐานของ MPEG-1 layer-3 ของสัญญาณเสียงที่ความถี่ 100 Hz, 11 kHz และ 20 kHzตามลำดับซึ่งแสดงในรูปของกราฟสเปคตรัมความถี่ของเอาท์พุทที่ได้จากแต่ละขั้นตอนโดยใช้โปรแกรม แมทแลบ เพื่อนำผลการทดลองที่ได้มาเปรียบเทียบกับสัญญาณเสียงอินพุทก่อนที่จะถูกนำมาเข้ารหัส โดยสัญญาณเอาท์พุทของเสียงที่ได้ภายหลังจากการถอดรหัสแล้วควรจะมีลักษณะ รูปร่าง และคุณสมบัติใกล้เคียงกับสัญญาณเสียงอินพุทในตอนแรก

ดังนั้น เมื่อพิจารณาผลการทดลองที่ได้ในบทที่ 7 แล้วสามารถสรุปได้ว่า

1.เอาท์พุทที่ได้จากการถอดรหัสข้อมูลฮัฟแมนจะเป็นเส้นความถี่ที่ถูกควอนไตซ์ (quantized frequency lines (is)) จำนวน 576 เส้นความถี่ต่อการถอดรหัสฮัฟแมนของข้อมูล 1 แชนแนล โดยในกระบวนการนี้จะเป็นการถอดรหัสฮัฟแมนโค้ดบิต (Huffman Code bits) ในบิตสตรีม เป็นเส้นความถี่ที่ถูกควอนไตซ์ จากกราฟที่ได้จะพบว่า ขนาดของเอาท์พุทที่ได้จะมีขนาดสูงสุดที่เส้นความถี่ที่แตกต่างกันขึ้นอยู่กับความถี่ของสัญญาณเสียงอินพุทด้วย

2. เอาท์พุทที่ได้จากการรีควอนไตซ์เส้นความถี่ที่ถูกควอนไตซ์ (is) จะมีจำนวนของเส้นความถี่เท่ากับ 576 เส้นความถี่เช่นเดียวกัน และมีลักษณะรูปแบบของสเปคตรัมความถี่เหมือนเดิม แต่จะพบว่าขนาดของเส้นความถี่ที่ผ่านการรีควอนไตซ์แล้ว (xr) จะมีขนาดลดลงอยู่ในช่วง [-1,+1] เท่านั้น เนื่องจากจะมีการดีสเกลลิงค่า is (เป็นผลมาจากการควอนไตซ์แบบนอนยูนิฟอร์ม (Non-uniform Quantizer) ในส่วนของการเข้ารหัส สาเหตุที่ต้องใช้การควอนไตซ์แบบนอนยูนิฟอร์ม เพราะปกติสัญญาณเสียงมักจะมีขนาดต่ำ ดังนั้นจึงจำเป็นต้องมีการลดขนาดของระดับ (step size) ลงที่ค่าสัญญาณอินพุทต่ำๆ และเพิ่มขนาดของระดับที่ค่าสัญญาณอินพุทที่สูงขึ้น) นอกจากนี้เอาท์พุทที่ได้ยังมีเครื่องหมายด้วย (แต่ในการแสดงผลในบทที่ 7 จะแสดงให้เห็นเป็นค่าสัมบูรณ์)

ในกระบวนการรีควอนไตซ์นี้นอกจากจะมีการดีสเกลลิงแล้วยังมีการทำรีออร์เดอร์ริงด้วยแต่การทำรีออร์เดอร์ริงจะกระทำก็ต่อเมื่อข้อมูลเป็นแบบวินโดว์สั้นเท่านั้น

3.การลดค่าปลอมของเส้นความถี่จะกระทำก็ต่อเมื่อข้อมูลเป็นแบบวินโดว์ยาวเท่านั้น โดยจะมีการลดค่าปลอมที่เกิดขึ้น 1 ค่า เนื่องจากการเหลื่อมกันของสเปคตรัม ซึ่งจะพบว่าลักษณะรูปแบบของเส้นความถี่ไม่แตกต่างไปจากเดิมมากนัก

4. เอาท์พุทที่ได้จากการทำ IMDCT จะอยู่ในแกนเวลา (Time domain) และมีจำนวนของเอาท์พุทเท่ากับ 1152 ไทม์แซมเปิล (time samples) เท่ากับจำนวนเส้นความถี่อินพุท เพราะการทำ IMDCT จะนำเส้นความถี่มาทีละ 18 เส้นความถี่ซึ่งอยู่ในแกนความถี่ (Frequency domain) มาแปลงให้อยู่ในแกนเวลาเป็นสัญญาณสุ่มในแกนเวลาจำนวน 36 ไทม์แซมเปิลต่อย่านความถี่ย่อยจนครบ 32 ย่านความถี่ย่อย ดังนั้นเอาท์พุทที่ได้จะมีสเปคตรัมในย่านความถี่ย่อยทั้งหมดยังคงมีลักษณะเหมือนเดิม และข้อมูลไทม์แซมเปิลจะอยู่ในย่านความถี่ย่อยเดิม

5. หลังจากการทำวินโดว์แล้ว จะพบว่าเอาท์พุทที่ได้จะมีความเป็นระเบียบมากขึ้น โดยจะสังเกตุเห็นว่าในไทม์แซมเปิลและย่านความถี่ย่อย ที่มีขนาดของข้อมูลน้อยจะถูกลดความสำคัญลงไปโดยจะให้ความสำคัญกับข้อมูลที่มีขนาดมาก

6. เอาท์พุทที่ได้จากการทำโอเวอร์แลบจะเป็นการนำไทม์แซมเปิล 18 ค่าแรกของทุกๆ ย่านความถี่ย่อยของแต่ละเฟรมไปโอเวอร์แลบกับ 18 ไทม์แซมเปิลของเฟรมถัดไป ในผลการทดลองนี้เป็นเฟรมแรกของข้อมูล ดังนั้นผลของการทำโอเวอร์แลบของเฟรมแรกนี้จึงเหลือ 18 ไทม์แซมเปิล แรกของแต่ละย่านความถี่ย่อยเท่านั้นและจะมีการเก็บไทม์แซมเปิล ที่เหลือไว้โอเวอร์แลบกับแกรนูลถัดไป เพื่อให้ได้ไทม์แซมเปิล 576 ค่า

7.เอาท์พุทที่ได้จากการทำโพลีเฟส ฟิลเตอร์แบงค์แต่ละครั้ง (1 แชนแนลใน 1 แกรนูล) จะได้เอาท์พุทออกมาจำนวน 576 ไทม์แซมเปิล และอยู่ในแกนเวลาโดยสมบูรณ์ ไม่มีการแบ่งข้อมูลเป็นย่านความถี่ย่อยอีกต่อไป โดยในการประมวลผลทำโดยดึงค่าข้อมูลสุ่มตัวอย่างที่ i เดียวกันของแต่ละย่านความถี่ย่อย(32 ย่านความถี่ย่อย) มาเรียงต่อกันจนครบ 18 ค่า

แต่การถอดรหัสข้อมูลเสียงในแต่ละเฟรมจะเสร็จสมบูรณ์ ก็ต่อเมื่อทำการถอดรหัสข้อมูลเสียงครบทั้ง 2 แชนแนล และ 2 แกรนูล เท่านั้น ในกรณีของการทดลองนี้เป็นแบบ Single channel ดังนั้นจึงต้องประมวลผลทั้งหมด 2 รอบ คือ สำหรับกรณี แกรนูล 0 และ 1 ดังแสดงในกราฟรูปสุดท้ายของผลการทดลองที่แต่ละความถี่

8.2 สรุปผลการทดลองในส่วน VHDL

จากการทดลองพบว่า ข้อมูลที่ได้จากการเก็บข้อมูลต่างๆจากบิตสตรีมมีค่าถูกต้องตามมาตรฐาน ซึ่งค่าที่ได้จาก Header บอกถึงข้อมูลว่าถูกเข้ารหัสมาตามมาตรฐาน เอ็มเป็ก 1 เลเยอร์ 3

โดยมีความถี่สุ่มเป็น 44.1 KHz มีบิตเรทขนาด 8 บิต และข้อมูลไม่ได้เข้ารหัสป้องกันความผิดพลาดมาด้วย ทั้งข้อมูลเสียงที่มีความถี่ 1 KHz และ 10 KHz

เมื่อพิจารณาลำดับการทำงานของอุปกรณ์ส่วนต่างๆพบว่า ทำงานได้ถูกต้องตามลำดับการทำงานที่ออกแบบไว้ โดยเราจะต่ออุปกรณ์โดยใช้สัญญาณนาฬิกาเป็นตัวให้จังหวะในการทำงานและรับ-ส่งข้อมูล แก่อุปกรณ์ต่างๆเพียงสัญญาณเดียว

สำหรับข้อมูลที่ไม่ได้อยู่ในกระบวนการตรวจจับในบางกรณี เช่น ในการเก็บค่าข้อมูลข้างเคียง ถ้า window_switching_flag เป็น '0' กระบวนการนี้จะไม่ทำการเก็บค่าของ block_type, mixed_block_flag และ subblock_gain ซึ่งในทางตรงกันข้าม ถ้า window_switchin_flag เป็น '1' กระบวนการนี้ก็จะไม่ทำการเก็บค่าของ region0count และ region1_count ซึ่งในกรณีนี้เราจะให้ค่าดังที่กล่าวมาเป็น 0 ทั้งหมด

8.3 การประเมินความเป็นไปได้ในการสร้างวงจรรวมทุกขั้นตอนของการถอดรหัส MPEG

ในขั้นตอนตั้งแต่รับบิตสตรีม การหาซิงค์เวิร์ด, การเก็บส่วนหัวข้อมูล, ข้อมูลข้างเคียง การถอดรหัสสเกลแฟคเตอร์ และการถอดรหัสฮัฟแมน สามารถที่จะออกแบบได้ง่าย เพราะข้อมูลที่ใช้ในขั้นตอนต่างๆสามารถเก็บอยู่ในรูปเลขฐานสองหรือบิตได้ แต่ในขั้นถอดถัดจากนี้ จะมีความสลับซับซ้อนมากกว่า เพราะข้อมูลที่คำนวณได้จะอยู่ในรูปของจำนวนจริง เป็น floating point จะมีทั้งการยกกำลัง ฟังก์ชันของโคไซน์ (cosine) การคูณด้วยวินโดว์ซึ่งเป็นเลขทศนิยม ซึ่งอาจจะทำในลักษณะของการเปิดตาราง (Look up Table) ได้ แต่อย่างไรก็ตาม การใช้ DSP เข้ามาช่วยจะทำให้ง่ายมากกว่า เนื่องจาก DSP สนับสนุนในการประมวลผลข้อมูลที่เป็นจำวนจริง ดังนั้นขั้นตอนตั้งแต่การรับบิตสตรีม การหาซิงค์เวิร์ดจนถึงการถอดรหัสฮัฟแมนสามารถออกแบบวงจรรวมได้ด้วยภาษาวีเอชดีแอล และข้อมูลที่ได้จะส่งต่อไปให้กับ DSP คำนวณในขั้นตอนตั้งแต่รีควอนไตซ์ถึงโพลีเฟสฟิลเตอร์ จะได้สัญญาณ PCM ออกมาแล้วส่งผ่านไปที่ลำโพงเป็นเสียงให้ได้ยิน

8.4 ความก้าวหน้าของการดำเนินงาน

จากการปฏิบัติตามแผนการดำเนินงานที่กล่าวไว้ในบทที่ 1 สามารถรายงานความก้าวหน้าของการดำเนินงานได้ดังนี้

ลำดับงาน	การดำเนินงาน	สถานะงาน
1	การหาข้อมูลจากแหล่งต่างๆ	✓
2	ศึกษาการทำงานของเอ็มเป็ก 1 เลเยอร์ 3	✓
3	ศึกษาการใช้งาน,การเขียนเอ็มไฟล์และการประมวลผลโปรแกรมแมทแลบ	✓
4	เขียนโปรแกรมถอดรหัสเอ็มเป็ก 1 เลเยอร์ 3 โดยใช้โปรแกรมแมทแลบ	✓
5	ศึกษาการออกแบบวงจรรวมถอดรหัสด้วยภาษา VHDL	✓
6	ออกแบบวงจรรวมถอดรหัสเอ็มเป็ก โดยใช้ภาษา VHDL ได้ในบางส่วน	✓
7	สรุปการดำเนินงานและวิเคราะห์ปัญหา	✓

8.5 ปัญหาในการดำเนินงาน

การดำเนินงานที่ผ่านมาประสบปัญหาต่างๆ ทั้งทางด้านการดำเนินงานในส่วนของการเขียนโปรแกรมถอดรหัสข้อมูลเอ็มเป็ก 1 เลเยอร์ 3 โดยใช้โปรแกรมแมทแลบ และในส่วนของการออกแบบวงจรรวมถอดรหัสข้อมูลโดยใช้ภาษา VHDL ดังต่อไปนี้

8.5.1 ปัญหาในการดำเนินงานในส่วนของการเขียนโปรแกรมถอดรหัสโดยใช้โปรแกรมแมทแลบ

1. ขาดแหล่งข้อมูลที่เพียงพอในการดำเนินงานในช่วงแรก เนื่องจากไม่มีข้อมูลเบื้องต้นที่ละเอียดเพียงพอและง่ายต่อการทำความเข้าใจ รวมทั้งแหล่งข้อมูลบางแหล่งให้ข้อมูลที่เชื่อถือไม่ได้ และข้อมูลจากแหล่งข้อมูลแต่ละแหล่งมักให้ข้อมูลแตกต่างกัน ทำให้ต้องใช้เวลาในการศึกษาและทำเข้าใจเพื่อหาข้อสรุปที่ถูกต้อง ในช่วงแรกเป็นเวลานาน
2. ขาดความรู้ความเข้าใจในการเขียนโปรแกรมแมทแลบอย่างลึกซึ้ง เนื่องจากผู้ดำเนินการยังไม่ค่อยมีประสบการณ์ในการใช้โปรแกรมแมทแลบมากเท่าที่ควร ต้องใช้เวลาในการฝึกฝนและศึกษาหาความรู้เพิ่มเติม เพื่อให้สามารถเขียนโปรแกรมแมทแลบได้อย่างถูกต้องและมีประสิทธิภาพ
3. ขาดความรู้ความเข้าใจการทำงานของการเข้ารหัส การถอดรหัสและการประมวลผลข้อมูลเอ็มเป็ก 1 เลเยอร์ 3 เนื่องจากจะต้องศึกษารายละเอียดและหลักการทั้งหมดตั้งแต่เริ่มแรก ดังนั้นจึงประสบปัญหาเรื่องการมีความรู้ไม่เพียงพอที่จะทำให้เข้าใจหลัก

การทำงานทั้งหมดอย่างละเอียด รวมทั้งเกิดความสับสนในบางขั้นตอนของการทำงาน ส่งผลให้เมื่อเกิดปัญหาขึ้นในการดำเนินงานไม่สามารถสรุปหาสาเหตุของปัญหาที่แน่ชัดได้

4. ในการทดสอบการทำงานของโปรแกรมถอดรหัสข้อมูลเอ็มเป็กที่เขียนขึ้นโดยโปรแกรมแมทแลบ ไม่สามารถหาไฟล์ข้อมูลมาตรฐานที่จะนำมาใช้ในการทดสอบได้ จึงต้องอาศัยการสร้างไฟล์เสียง(.wav) ขึ้นมาเองเพื่อตรวจสอบการทำงาน ซึ่งไฟล์ที่สร้างขึ้นมาเองนี้ไม่สามารถทดสอบคุณสมบัติต่างๆ ของโปรแกรมได้อย่างทั่วถึง ทำได้เพียงแต่ทดสอบว่าโปรแกรมที่เขียนขึ้นทำงานได้ถูกต้องหรือไม่เท่านั้น
5. จากผลของข้อ 4 ทำให้ต้องหาโปรแกรมที่จะนำมาใช้เข้ารหัสไฟล์เสียงตามมาตรฐานเอ็มเป็ก 1 เลเยอร์ 3 ที่เหมาะสมได้ยาก และบางโปรแกรมมีข้อจำกัดต่างๆ เช่น จำนวนบิตเรท ประเภทของช่องความถี่ในการเข้ารหัส มีการเข้ารหัสหลายเลเยอร์ในไฟล์เพียงไฟล์เดียว เป็นต้น ส่วนในการทดลองนี้เราใช้โปรแกรม Shuffler Music Converter

8.5.2 ปัญหาในส่วนของการออกแบบวงจรรวมถอดรหัสโดยใช้ภาษา VHDL

1. ขาดความรู้ความเข้าใจในหลักการและการใช้ภาษา VHDL ในการออกแบบวงจรรวมถอดรหัส เนื่องจากผู้ดำเนินการไม่มีพื้นฐานในเรื่องนี้มาก่อน ดังนั้นจึงเสียเวลาในการศึกษาและทำความเข้าใจเพื่อที่จะสามารถออกแบบวงจรรวมถอดรหัสให้สามารถทำงานได้อย่างถูกต้องเป็นเวลานาน นอกจากนี้ยังขาดทักษะในการใช้ภาษา VHDL ที่ดีพอ โปรแกรมที่เขียนขึ้นจึงยังมีประสิทธิภาพเท่าที่ควร
2. ในการออกแบบวงจรรวมถอดรหัส เราใช้วิธีออกแบบเป็นบล็อกการทำงาน ซึ่งแต่ละบล็อกต้องใช้เวลาในการออกแบบค่อนข้างนานและโปรแกรมมีความซับซ้อน เพราะไม่สามารถใช้การวนรอบการทำงานได้ ทำให้ไม่สามารถดำเนินงานได้เสร็จทันเวลาและบรรลุเป้าหมายที่วางไว้ในตอนแรก
3. หาโปรแกรมโมเดลซิมเพื่อใช้จำลองการทำงานของวงจรรวมได้อย่างสมบุรณ์ค่อนข้างยาก และประสบปัญหาในการนำอุปกรณ์ที่เขียนขึ้นมาแต่ละส่วนมาเชื่อมต่อกันทำได้ยาก เนื่องจากจะต้องให้การส่งและรับข้อมูลระหว่างอุปกรณ์แต่ละตัวมีจังหวะการทำงานที่สอดคล้องกันทั้งหมด (Synchronization)

8.6 สรุปผลการดำเนินงานทั้งหมด

เนื่องด้วยการดำเนินงานที่ผ่านมาในภาคเรียนที่ 1 และภาคการเรียนที่ 2 ประสบปัญหาต่างๆ ดังที่กล่าวมาแล้ว ทำให้การดำเนินงานในบางขั้นตอนค่อนข้างติดขัดและเกิดความล่าช้าไม่เป็นไปตามเป้าหมายที่กำหนดไว้ในตอนแรก แต่ผู้ดำเนินการได้พยายามอย่างเต็มที่ที่จะทำให้การดำเนินงานสำเร็จลุล่วงไปให้ได้มากที่สุดเท่าที่จะเป็นไปได้ ดังนั้นจึงขอสรุปผลการดำเนินงานในแต่ละขั้นตอนเพื่อให้เกิดความเข้าใจในลำดับการดำเนินงาน และได้เสนอแนะแนวทางการแก้ไขปัญหาที่เกิดขึ้น เพื่อการพัฒนาประสิทธิภาพในการถอดรหัสข้อมูลเอ็มเป็กในอนาคตต่อไป

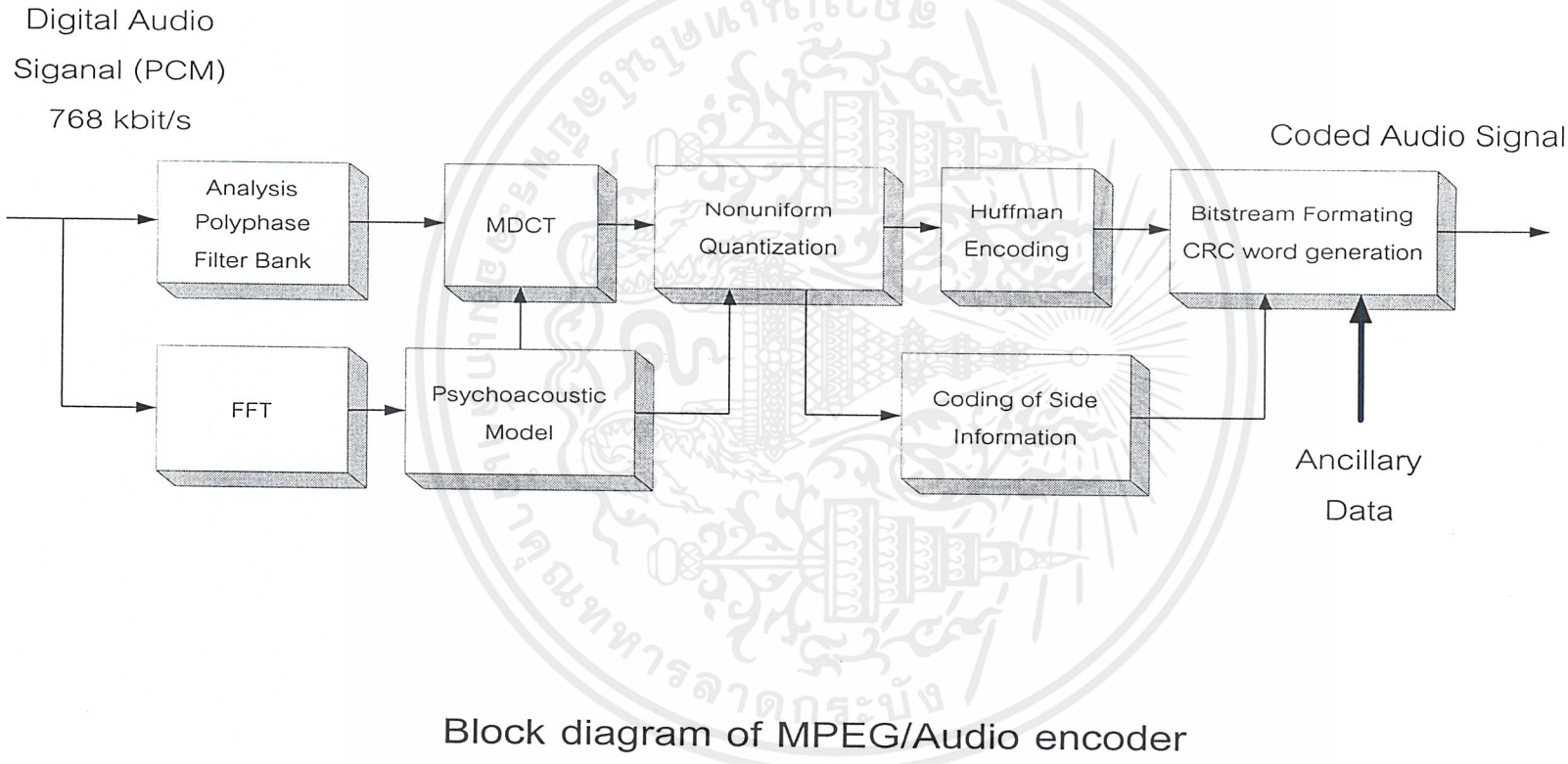
1. การหาข้อมูลจากแหล่งต่างๆ และการศึกษาการทำงานของเอ็มเป็ก 1 เลเยอร์ 3 ผู้จัดทำได้ค้นคว้าหาข้อมูลเกี่ยวกับรูปแบบการเข้ารหัส การถอดรหัส และการประมวลผลมาพอสมควร โดยแหล่งข้อมูลส่วนใหญ่จะอยู่ในมาตรฐานอุตสาหกรรมของเอ็มเป็ก และปริญญานิพนธ์ของมหาวิทยาลัยในต่างประเทศ แต่ข้อมูลที่ได้มาจะอยู่ในรูปของหลักการและวิธีการแบบกว้างๆ มากกว่าที่จะชี้เฉพาะเจาะจงลงไปถึงในส่วนของรายละเอียด ดังนั้นจึงส่งผลให้ยากต่อการทำความเข้าใจให้ถ่องแท้และการแก้ไขปัญหาที่เกิดขึ้นก็เป็นไปอย่างยากลำบาก เพราะไม่สามารถระบุสาเหตุที่แน่ชัดของปัญหานั้นๆ ได้
2. การศึกษาการเขียนโปรแกรมการถอดรหัสข้อมูลแบบเอ็มเป็ก 1 เลเยอร์ 3 โดยใช้โปรแกรมแมทแลบ และปรับปรุงแก้ไขโปรแกรมที่เขียนขึ้นให้ทำงานได้อย่างถูกต้องและมีประสิทธิภาพ เหมือนข้อมูลเสียงที่ถูกถอดรหัสจากโปรแกรมถอดรหัสที่ได้รับการยอมรับในปัจจุบัน (โปรแกรม Winamp) เป็นไปไม่ได้ เพราะโปรแกรมแมทแลบไม่ได้มีการประมวลผลแบบเวลาจริง (Real Time) แต่โปรแกรมแมทแลบจะช่วยในการทำความเข้าใจการทำงานของการถอดรหัสและสะดวกในการแก้ไขพัฒนาโปรแกรมให้ทำงานได้ถูกต้องและมีประสิทธิภาพ ก่อนที่จะนำไปใช้ในการออกแบบวงจรรวมถอดรหัสด้วยภาษา VHDL ซึ่งสามารถประมวลผลแบบเวลาจริงได้ต่อไป

   ในส่วนโปรแกรมการถอดรหัสที่เขียนนี้ ยังพบว่าข้อมูลในบางส่วนยังมีความผิดพลาดอยู่ ซึ่งยังไม่สามารถสรุปหาสาเหตุของความผิดพลาดที่เกิดขึ้นได้อย่างแน่นอน
3. การศึกษาการออกแบบวงจรรวมถอดรหัสด้วยภาษา VHDL ในส่วนนี้สามารถออกแบบวงจรรวมถอดรหัสได้เพียงบางส่วน ได้แก่ การดึงข้อมูลและค้นหาส่วนหัวของเฟรม, การตรวจสอบความผิดพลาด, การถอดรหัสข้อมูลข้างเคียง และการถอดรหัส

ข้อมูลสเกลแฟคเตอร์ เท่านั้น เนื่องจากในแต่ละขั้นตอนใช้เวลาในการออกแบบค่อนข้างมาก จึงทำให้ไม่สามารถออกแบบได้ครบหมดทุกขั้นตอน ดังนั้นจึงเป็นเพียงการศึกษาและทดลองออกแบบในบางส่วนเท่านั้น

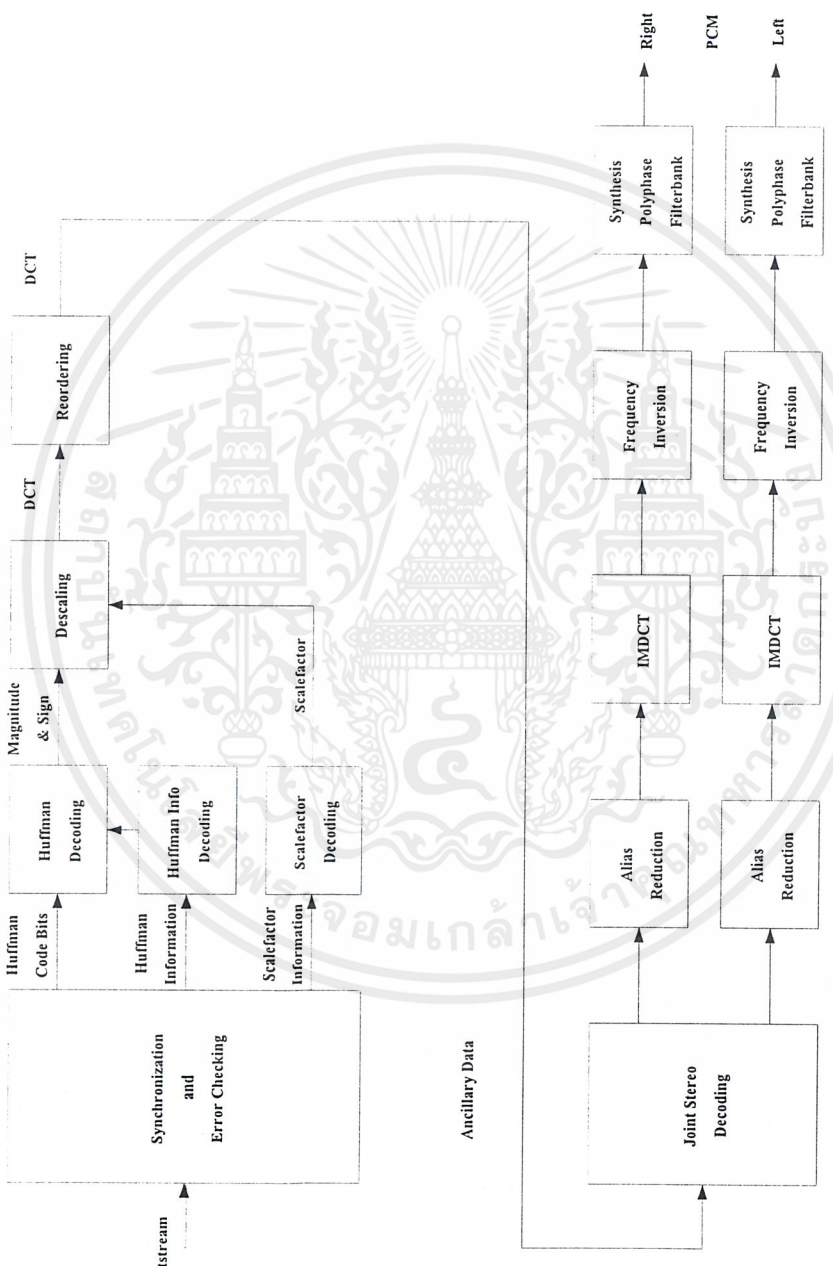
4. ในการทดสอบการทำงานของโปรแกรมการถอดรหัสข้อมูลแบบเอ็มเป็ก 1 เลเยอร์ 3 ในการทดลองนี้ได้ใช้ไฟล์เสียงที่สร้างขึ้นเองมาทำการตรวจสอบว่าโปรแกรมที่เขียนขึ้นมานี้สามารถทำงานได้ถูกต้องหรือไม่ แต่ถ้าต้องการตรวจสอบคุณสมบัติต่างๆ ให้ได้อย่างครบถ้วนควรใช้ไฟล์เสียงที่เป็นมาตรฐานมาทำการทดสอบการทำงานซึ่งจะสามารถตรวจสอบการทำงานของโปรแกรมได้ครบทุกเงื่อนไข ที่บางครั้งไฟล์เสียงที่เราสร้างขึ้นเองไม่สามารถทำได้

ภาคผนวก

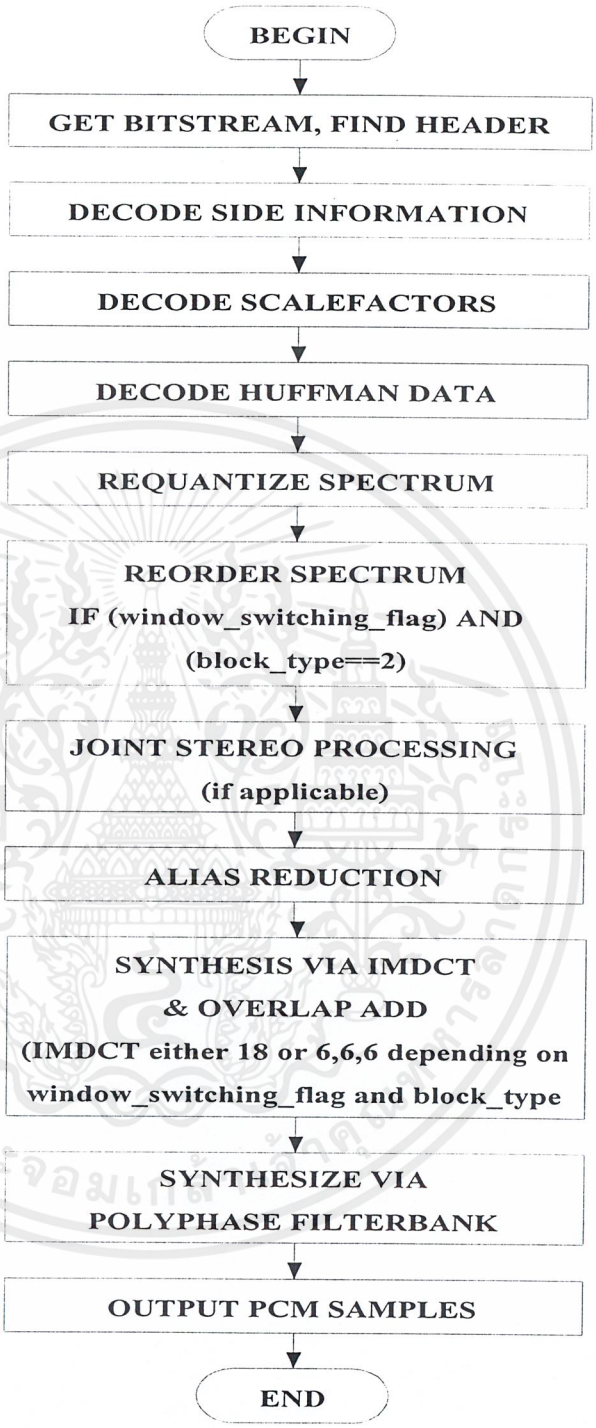


Block diagram of MPEG/Audio encoder

รูปที่ ก.1 บล๊อกไดอะแกรมการเข้ารหัสเอ็มเป็กเลเยอร์3

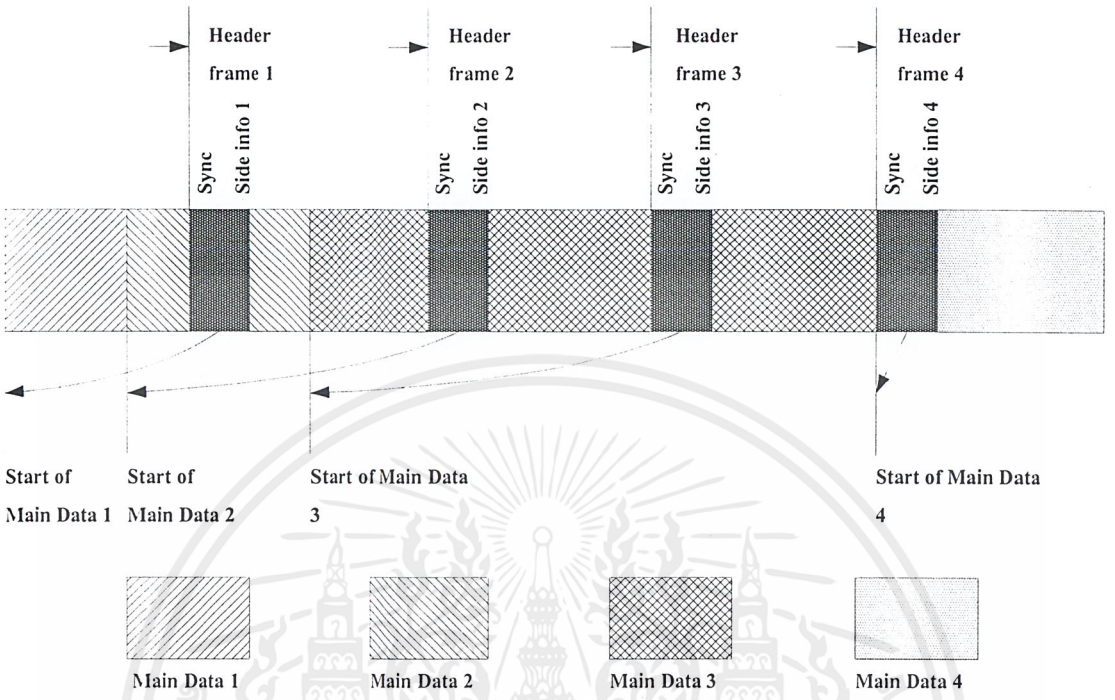


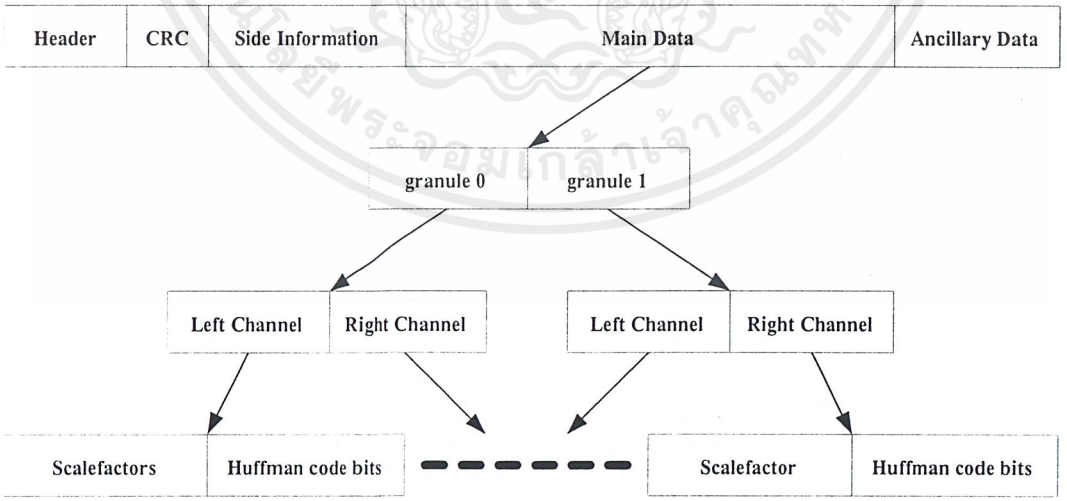
รูปที่ ก.2 บล็อกไดอะแกรมการถอดรหัสเอ็มพีกเลเยอร์3



Layer III decoder flow chart

รูปที่ ก.3 แสดงการถอดรหัสเอ็มเป็กเลเยอร์3



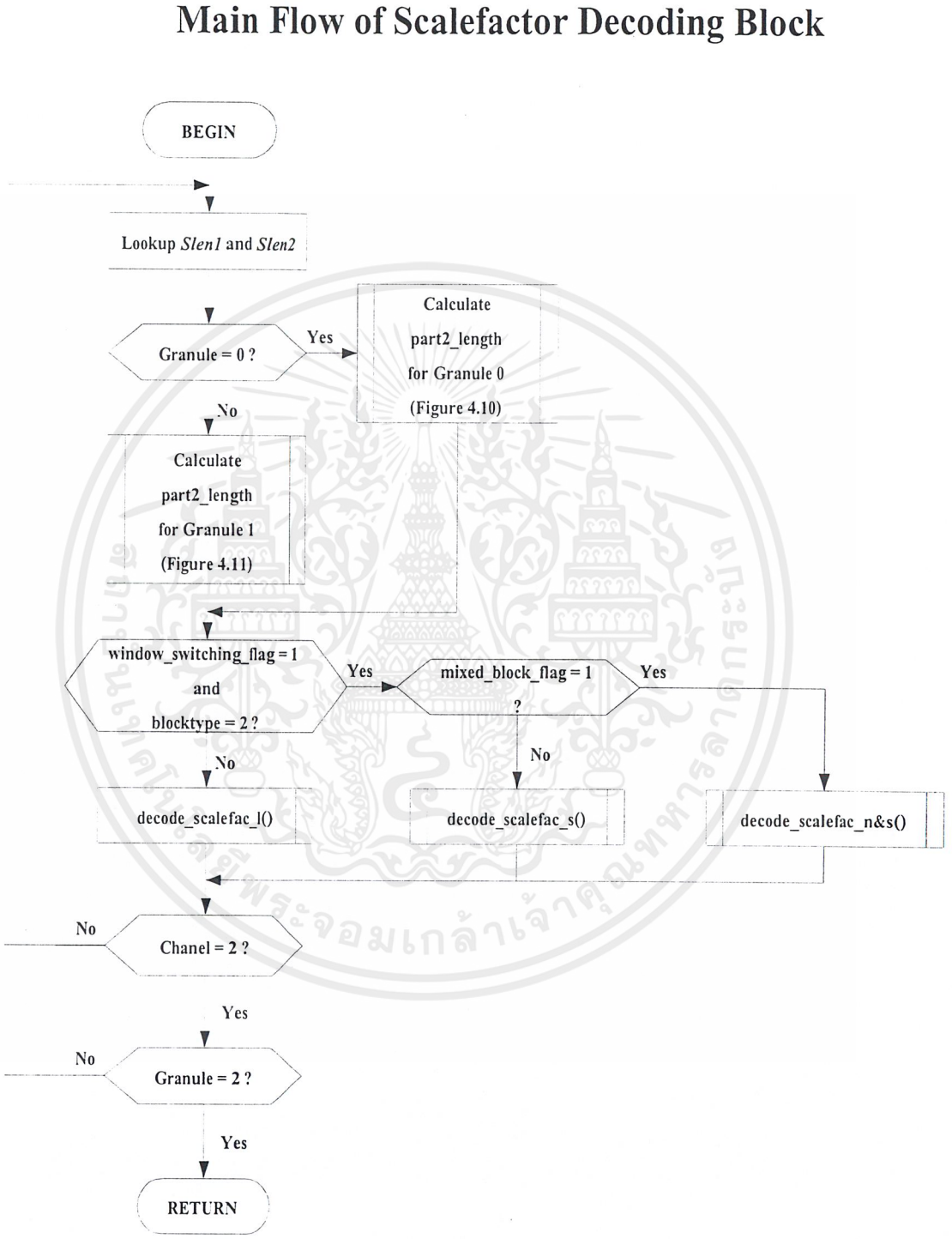


รูปที่ ก.4 รูปแบบของข้อมูลบิตสตรีม

รูปที่ ก.5 แสดง อินพุท และ เอาท์พุท ของการถอดรหัสสเกลแฟคเตอร์

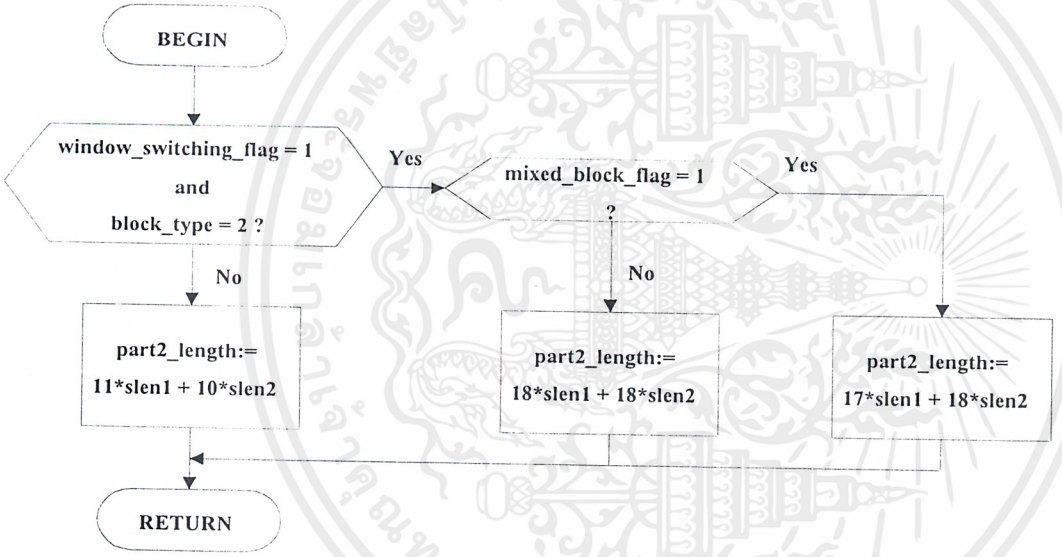
ตารางที่ ก.1 แสดงข้อมูลส่วน Header

Syntax	No. of bits	Mnemonic
Header ()		
{		
syncword	12	blslbf
ID	1	blslbf
Layer	2	blslbf
Protection_bit	1	blslbf
bitrate_index	4	blslbf
sampling_frequency	2	blslbf
padding_bit	1	blslbf
private_bit	1	blslbf
mode	2	blslbf
mode_extension	2	blslbf
copyright	1	blslbf
original/copy	1	blslbf
emphasis	2	blslbf
}		

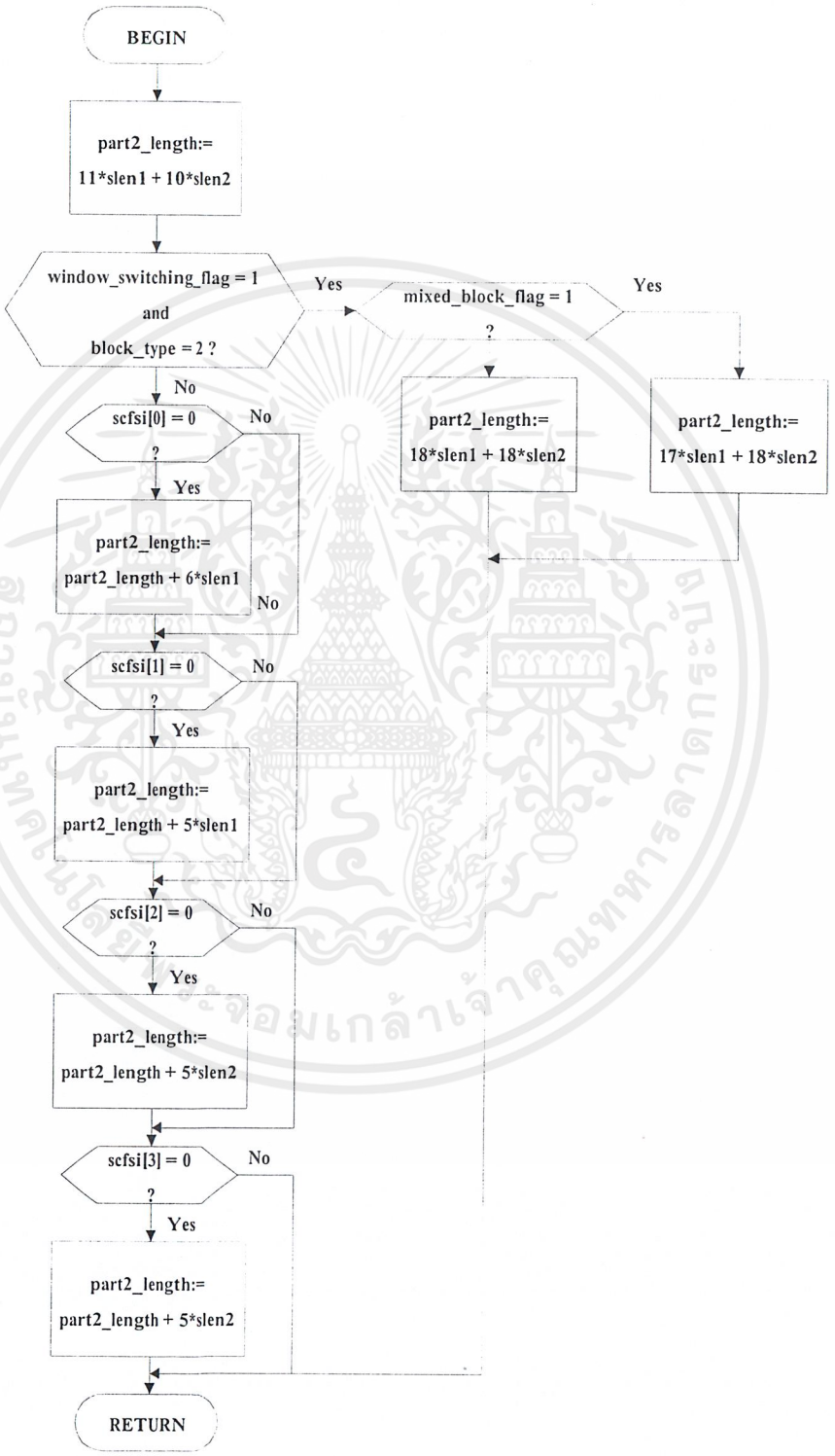


รูปที่ ก.6 โฟลว์ชาร์ตหลักของการถอดรหัสสเกลแฟคเตอร์

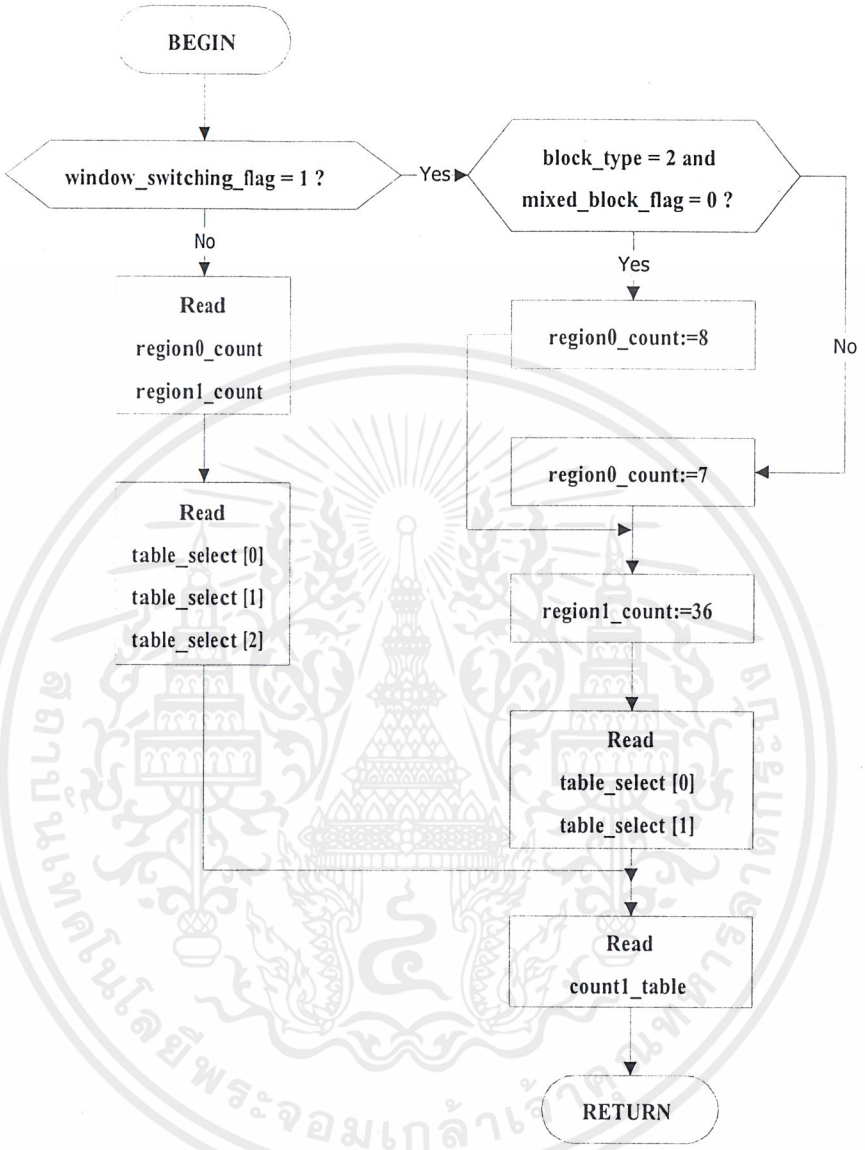
Calculatioin of part2_length for first granule (granule = 0)



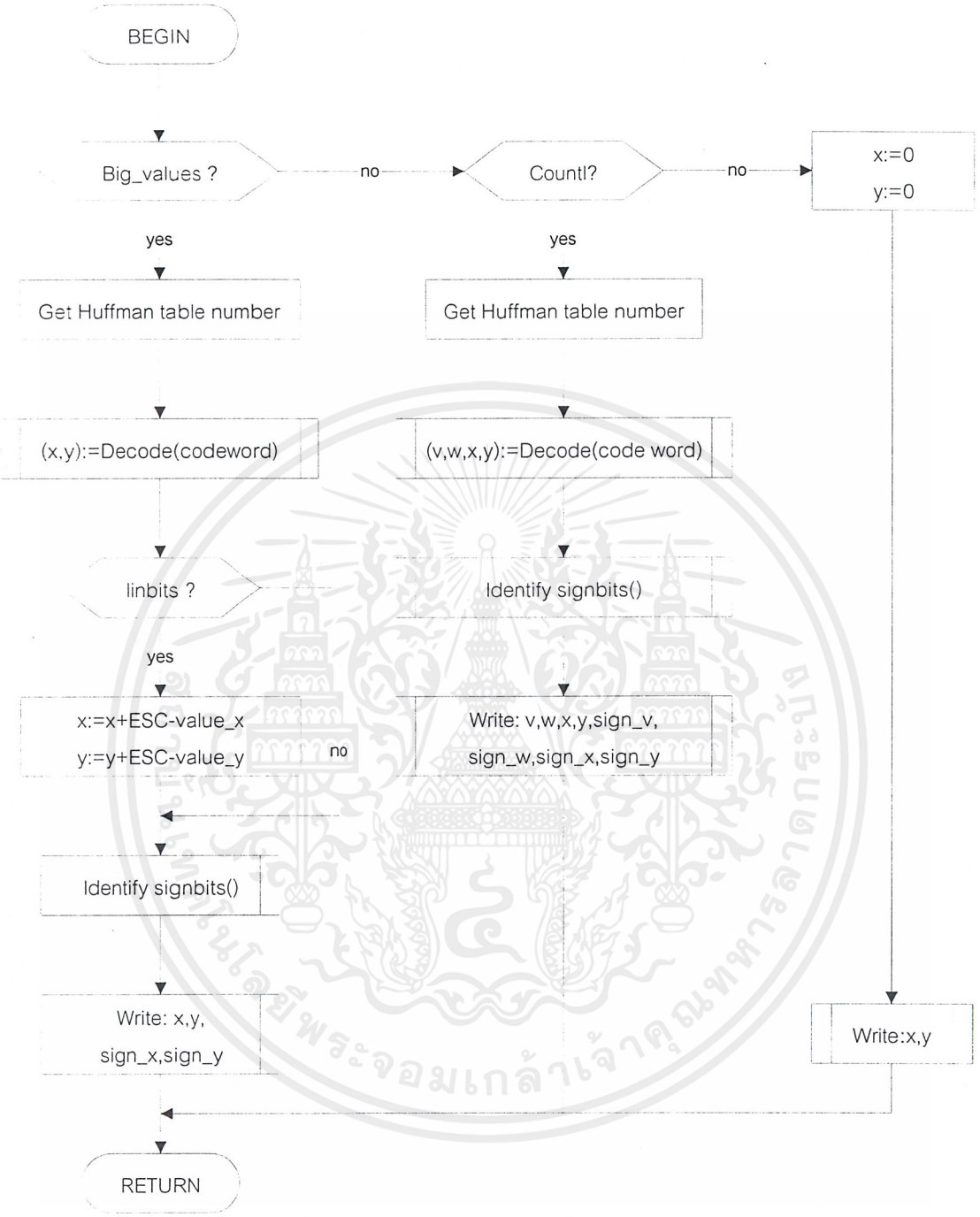
รูปที่ ก.7 การคำนวณหาจำนวนบิตของข้อมูลที่เป็นสเกลแฟคเตอร์ของแกรนนูล 0



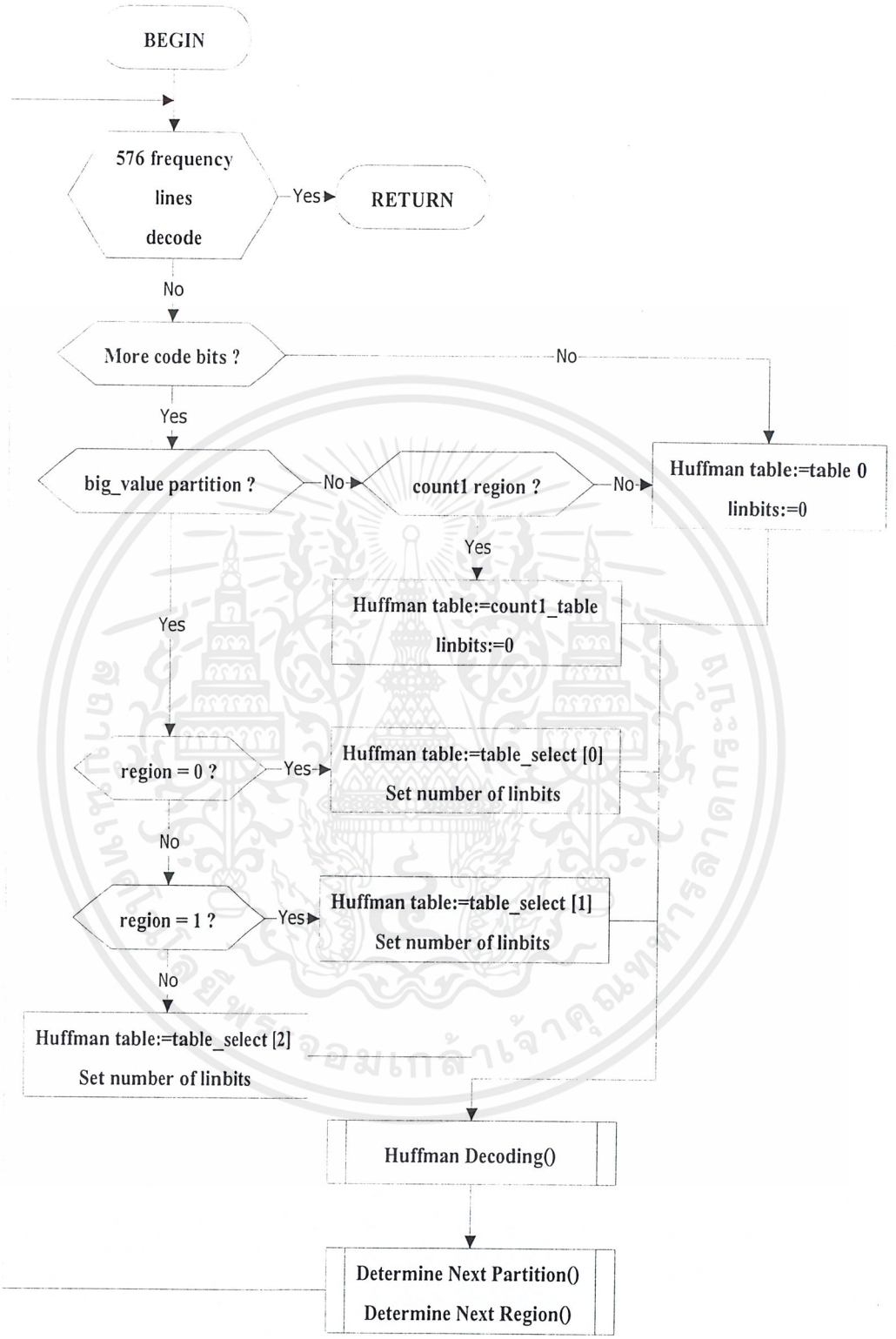
รูปที่ ก.8 การคำนวณหาจำนวนบิตของข้อมูลที่เป็นสเกลแฟคเตอร์ของแกรนนูล 1



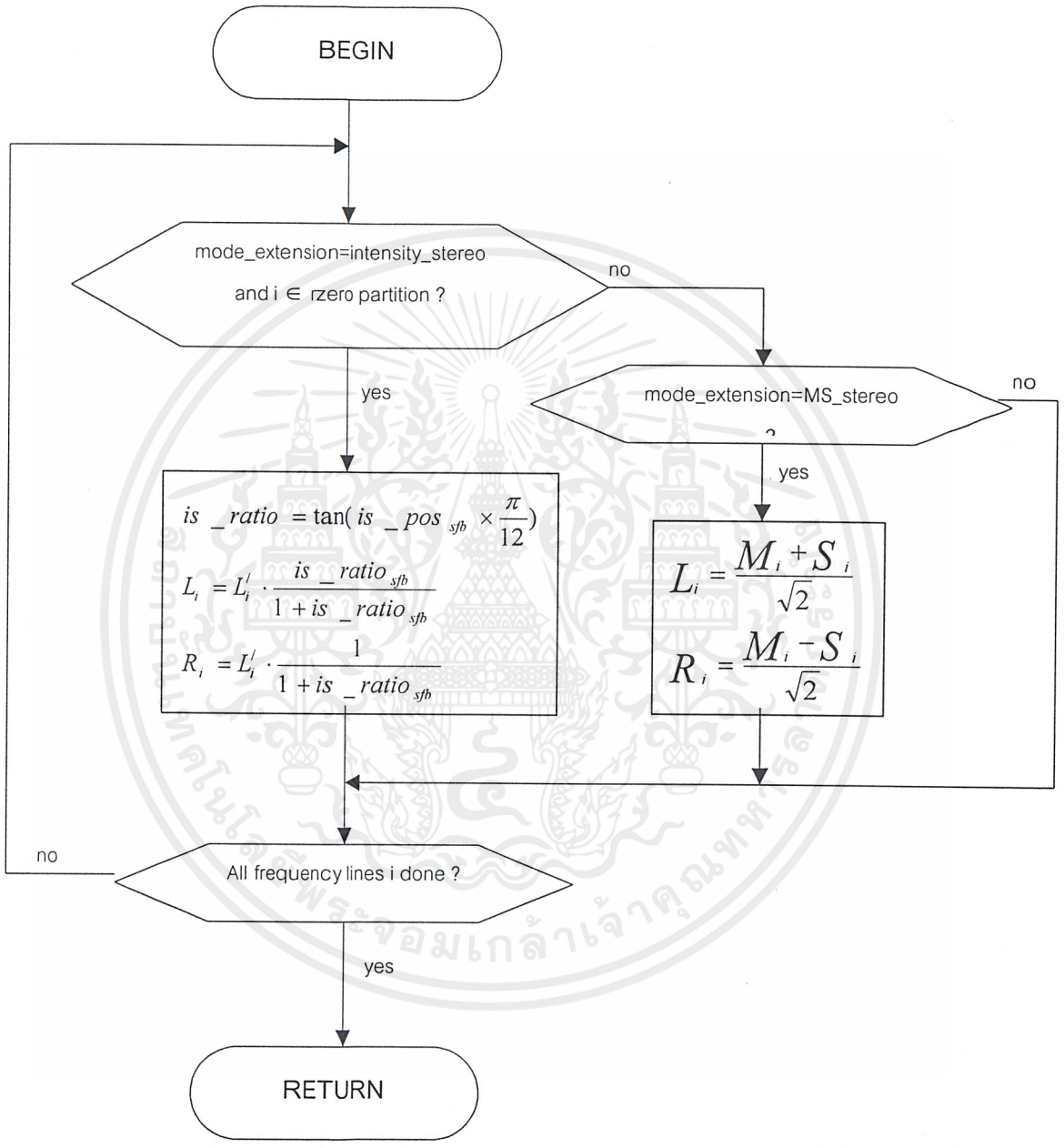
รูปที่ ก.9 แสดงการหาขอบเขตของแต่ละส่วนภายใน big_value และการเลือกใช้ตาราง



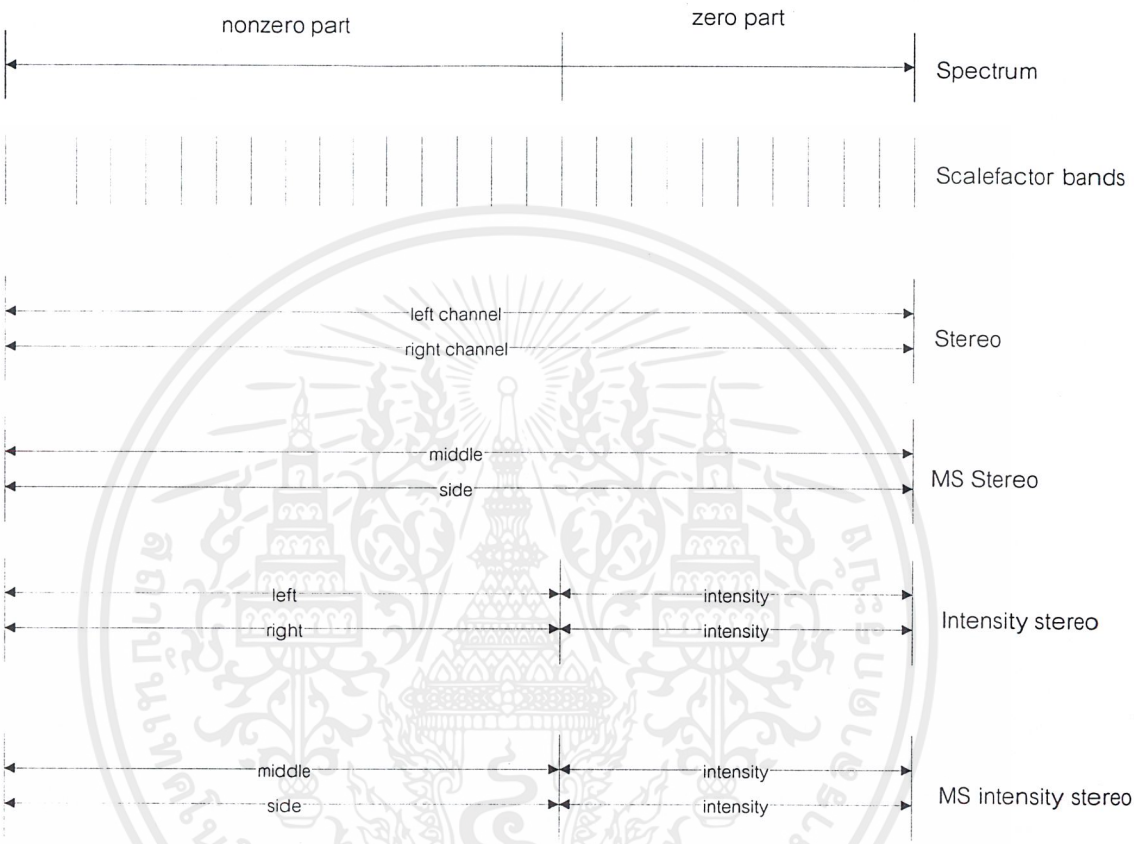
รูปที่ ก.10 โฟล์วชาร์ต แสดงขั้นตอน และการตัดสินใจ ในการถอดรหัสฮัฟแมน



รูปที่ ก.11 แสดงการถอดรหัสฮัฟแมนโค้ด

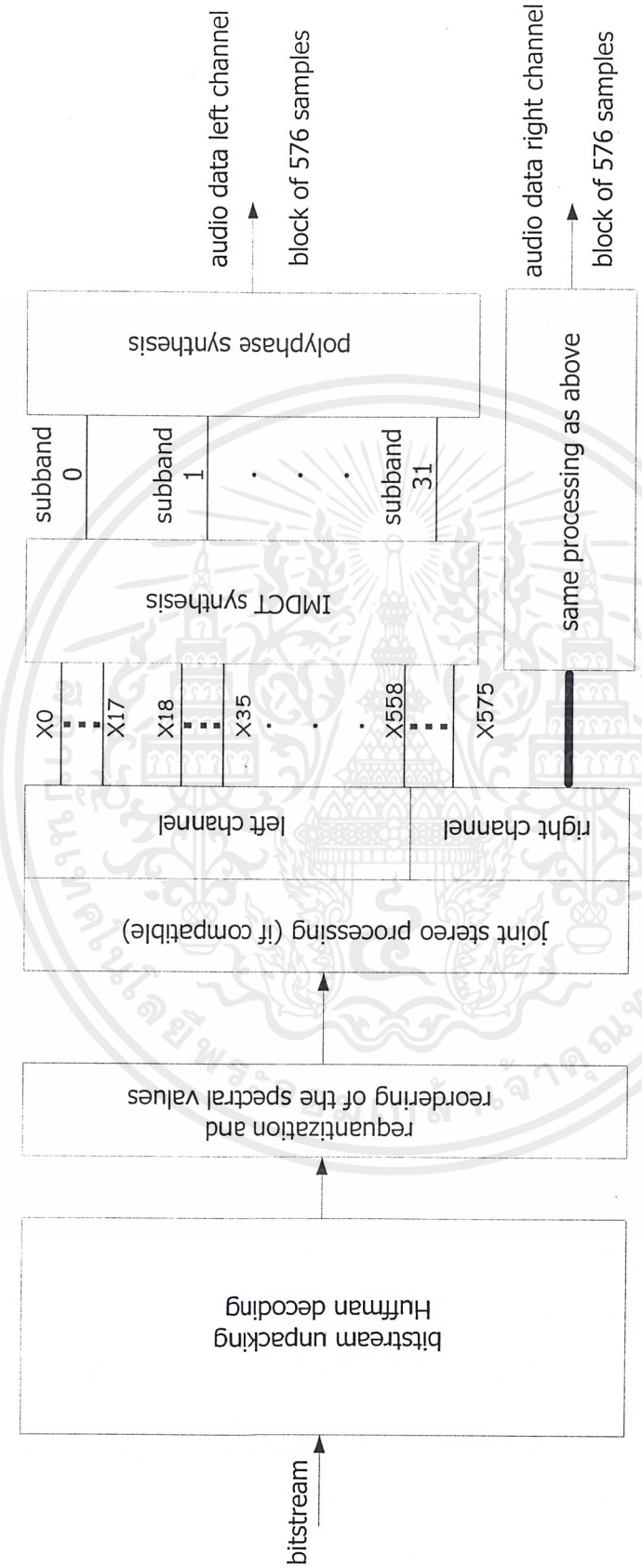


รูปที่ ก.12 แสดงการเข้าหัสแบบสเตอริโอ

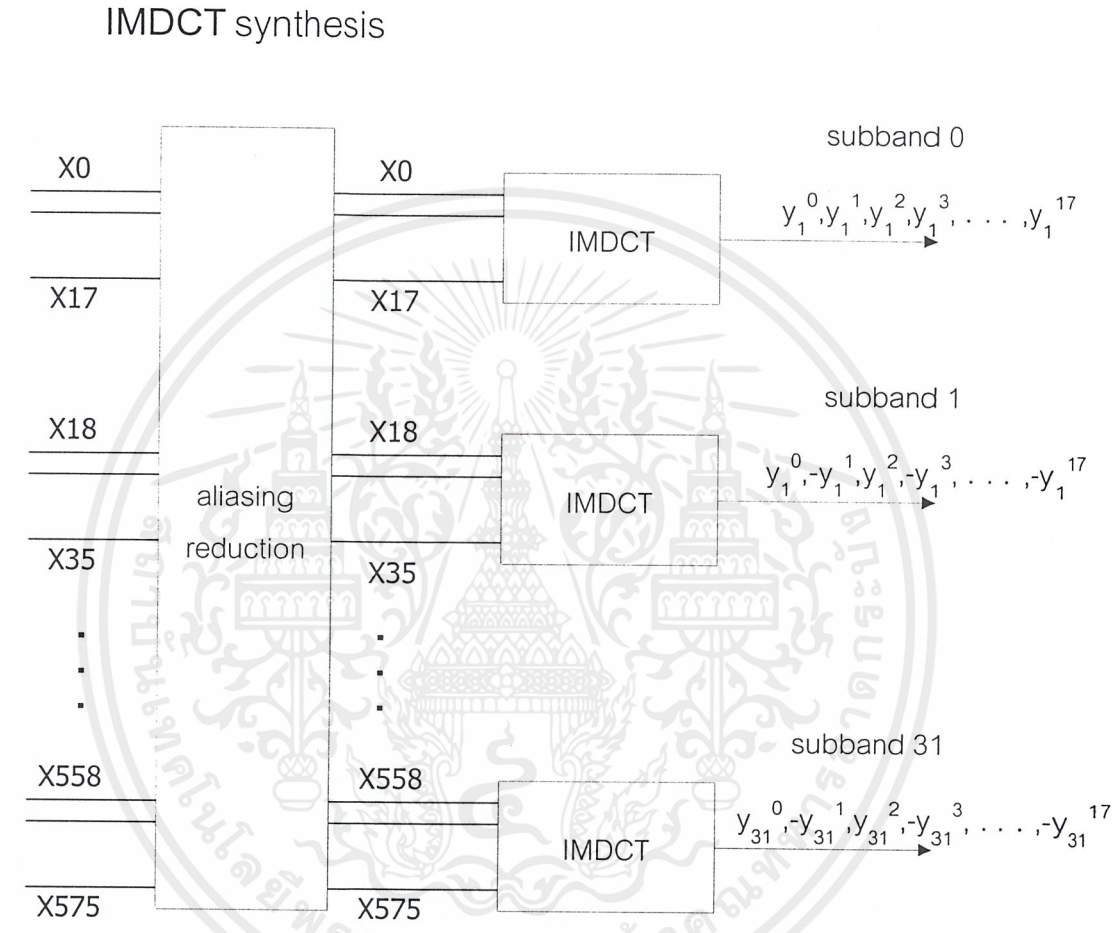


Different modes for encoding stereo signals . These modes are controlled by mode and mode_extension .

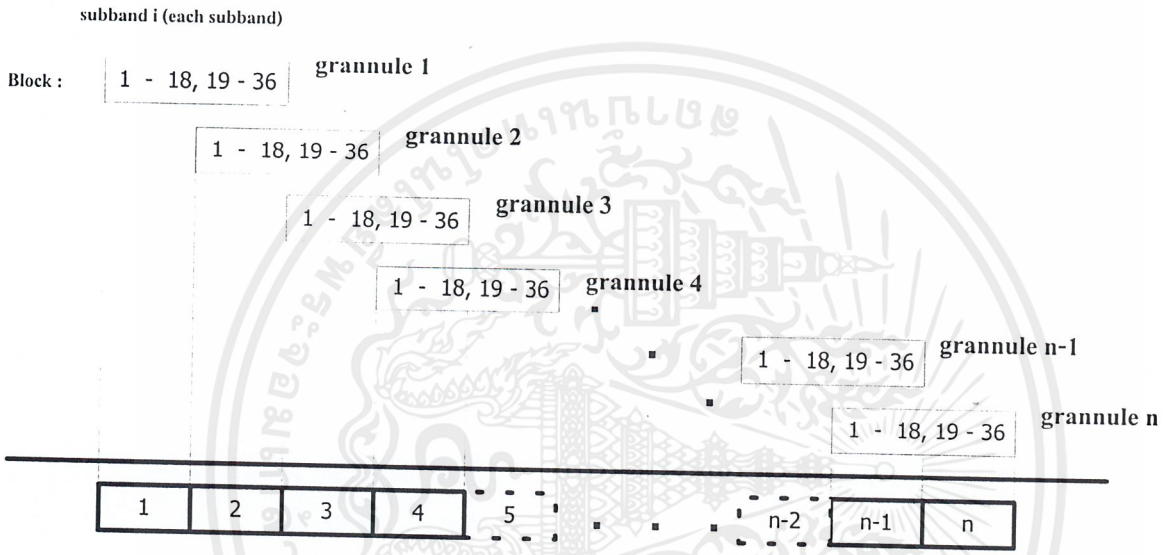
รูปที่ ก.13 แสดงการเข้ารหัส สเตอริโอ



รูปที่ ก.14 แสดงการถอดรหัสจากบิตสตรีมเป็นสัญญาณเสียงในโดเมนเวลา



รูปที่ ก.15 แสดงการทำ aliasing reduction และการทำ IMDCT



Overlap add of the resulf of the IMDCT

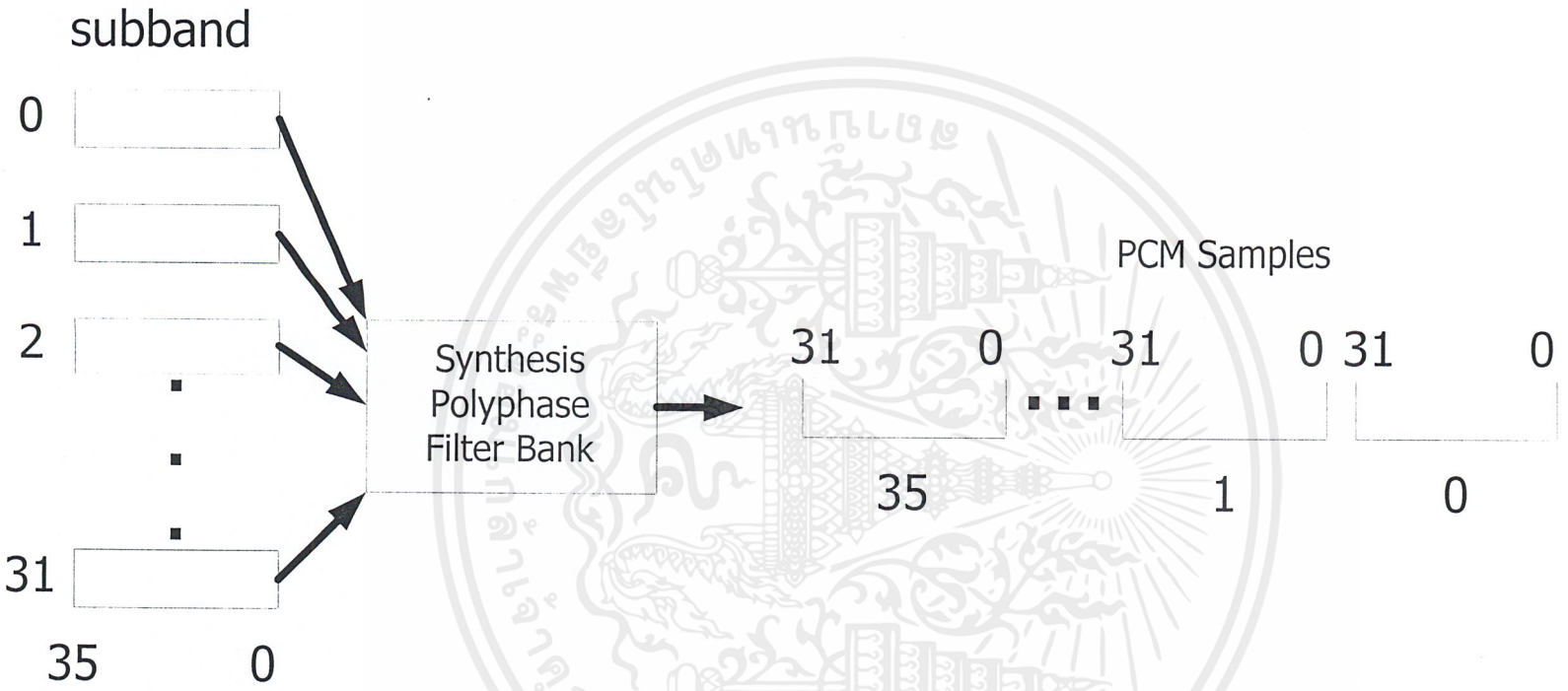
For grannule = 1 to n

{ ในแต่ละบล็อก ของ subband มี 36 samples ซึ่ง ครึ่งบล็อกแรกนี้จะโอเวอร์แลป กับครึ่งบล็อกหลัง ของแกรนนูลก่อนหน้า และครึ่งบล็อกหลังนี้จะเก็บไว้เพื่อใช้บวกกับครึ่งบล็อกแรกของแกรนนูลถัดไป

result(i) = Z(i) + S(i) for i= 0 to 17

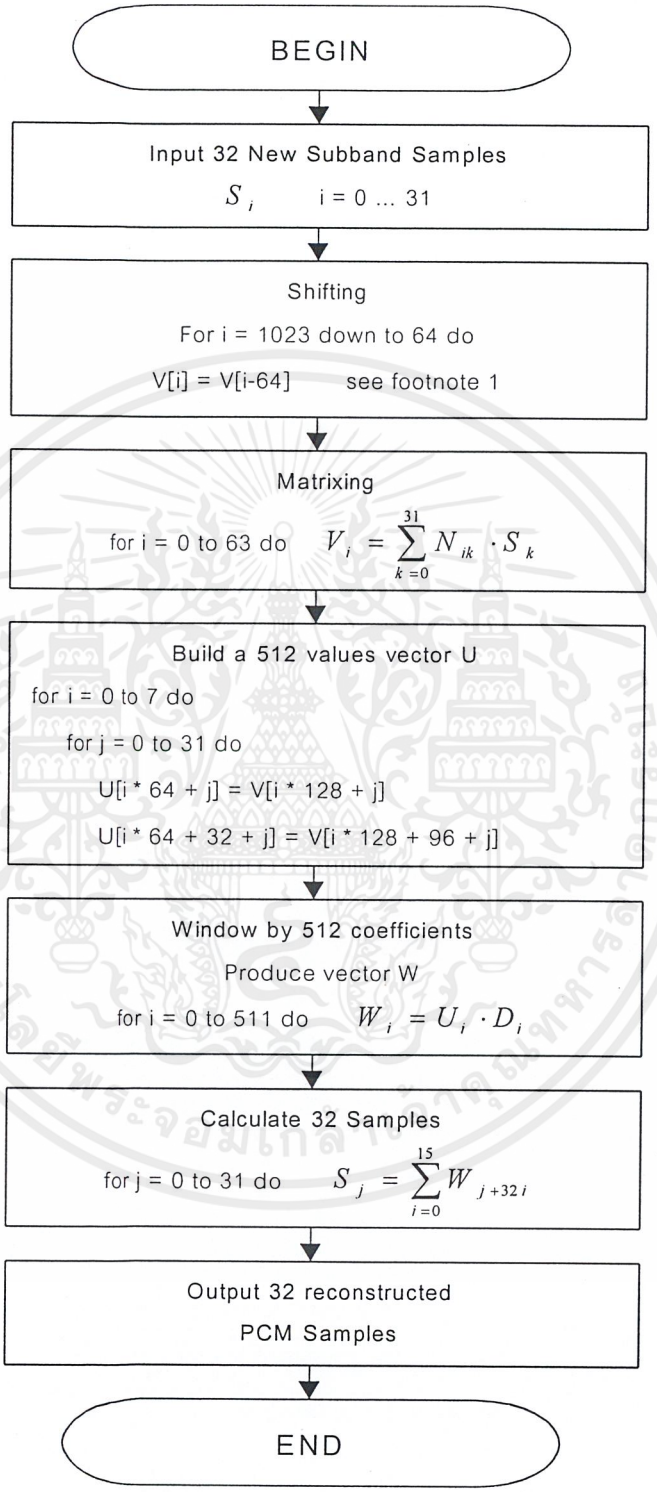
S(i) = Z(i+18) for i= 0 to 17

รูปที่ ก.16 แสดงการทำโอเวอร์แลป

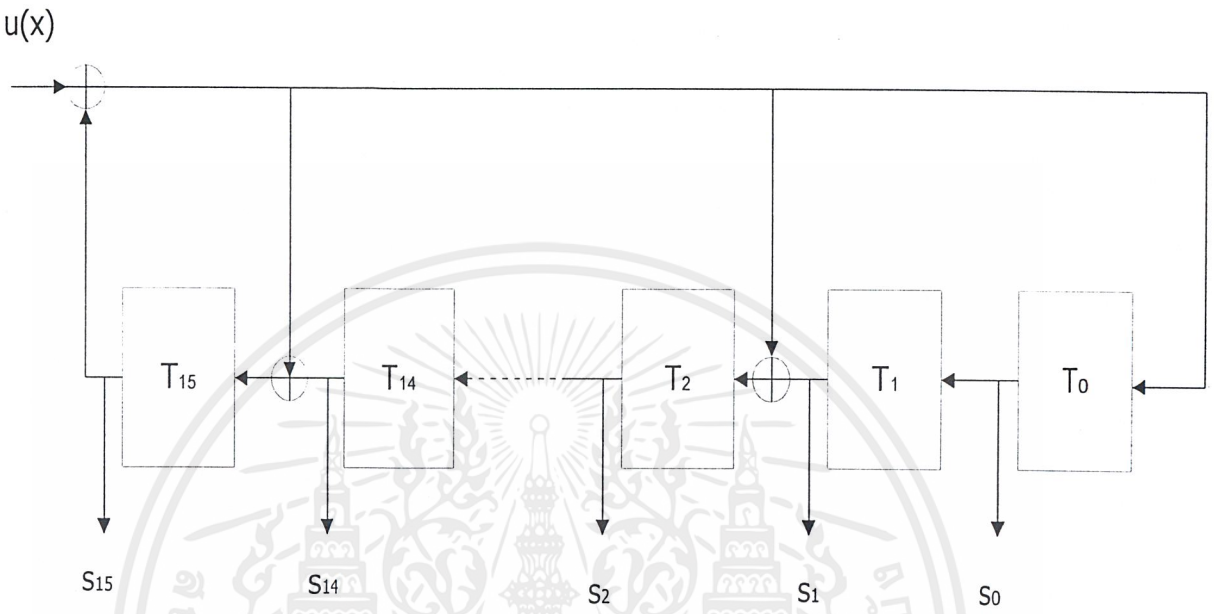


The synthesis filter transform a sample from each subband into 32 consecutive PCM samples

รูปที่ ก.17 แสดงการทำโพลีเฟส ฟิลเตอร์ แบงก์1



รูปที่ ก.18 แสดงการทำโพลีเฟส ฟิสเตอร์ แบงก์2



รูปที่ ก.19 วงจรแสดงการคำนวณหาค่า CRC โดยใช้ Linear Feedback Shift Register (LFSR)

ภาคผนวก ข

รูปแบบข้อมูลเสียงเอ็มเป็ก

(MPEG LAYER 3 FORMAT)

รูปแบบแฟ้มข้อมูลเอ็มเป็ก 3 (mpeg layer3 file format) อธิบายโดยใช้ไวยากรณ์ (Syntex) ของภาษาซีเพื่อให้เห็นภาพพจน์ โดยใช้ตัวอักษรเข็มแสดงชื่อของข้อมูลในแต่ละส่วน เลขด้านท้ายของแต่ละบรรทัดแสดงความยาวบิตของข้อมูลนั้น ส่วนคำอธิบายละเอียดจะถูกยกไปอธิบายในหัวข้อ ข.2

ข.1 ไวยากรณ์ของข้อมูลเอ็มเป็ก (Codec audio bitstream syntax)

ข.1.1 ลำดับข้อมูลเสียง (Audio sequence)

```
Audio sequence()
{
while (nextbits()==syncword){
frame();
}
}
```

ข.1.2 เฟรมข้อมูลเสียง (Audio frame)

```
Frame()
{
    header();
    error_check();
    audio_data();
    ancillary_data();
}
```

ข.1.3 ส่วนหัวข้อมูล (Header)

```
Header(){
    synword        12
    ID             1
    layer          2
```

protection_bit	1
bitrate_index	4
sampling_frequency	2
padding_bit	1
private_bit	1
mode	2
mode_extention	2
copyright	1
original/copy	1
emphasis	2}

ข.1.4 ส่วนตรวจสอบความผิดพลาด (Error check)

```
Error_check()
{
        if(protection_bit==0)
            crc_check                                    16
}
```

ข.1.5 ส่วนข้อมูลเสียง (Audio data)

```
Audio_data()
{
        main_data_begin                                  9
        if(mode==single_channel)
            private_bits                                 5
        else
            private_bits                                 3
        for(ch=0;ch<nch;ch++)
            for(scfsi_band=0;scfk_band<4;scfi_band++)
                scfsi[scfsi_band]                        1
```

```
for(gr=0;gr<2;gr++)
    for(ch=0;ch<nch;ch++){
        part2_3_length[gr] [ch]                    2
        big_values[gr] [ch]                        9
        global_gain[gr] [ch]                       8
        scalefac_compress[gr] [ch]                 4
        window_switching_flag[gr] {ch}             1
        if(window_switching_flag[gr] [ch]){
            block_type[gr] [ch]                    2
            mixed_block_flag[gr] [ch]              1
            for (region=0;region<2;region++)
                table_select{gr] [ch] [window]     3}
            else{
            for(region=0;region,3;region++)
                table_select [gr] [ch] [region]    5
                region0_count [gr] [ch]            4
                region1_count [gr] [ch]            3
            }
        preflag[gr] [ch]                           1
        scalefac_scale [gr] [ch]                   1
        count1table_select[gr] [ch]                1
        }
    main_data()
}
```

ข.1.6 ส่วนข้อมูลหลัก (main data)

```
Maindata()
{
for(gr];gr<2;gr++) {
```

```
for )ch=0;ch<nch++){
if ((window_switching_flag[gr][ch] ==1)&&(block_type[gr][ch]==2)){
if(mixed_block_flag [gr][ch]){
        for (sfb =0; sfb<8 ;sfb++)
                scalefac_l[gr][ch][sfb]                         0..4
for (sfb=3;sfb<12;sfb++)
        for (window =0;windows<3;windows++)
                scalefac_s[gr][ch][sfb][window]                 0..4}
else{
for(sfb=0;sfb<12;sfb++)
        for(window=0;window<3;window++)
                scalefac_s[gr][ch][sfb][window]                 0..4
        }}
else{
if((scfi[ch][0]==0 ll(gr==0))
        for(sfb=0;sfb<6;sfb++)
                scalefac_l[gr][ch][sfb]                         0..4
if((scfi[ch][1]==0 ll(gr==0))
        for(sfb=6;sfb<11;sfb++)
                scalefac_l[gr][ch][sfb]                         0..4
if((scfi[ch][2]==0 ll(gr==0))
        for(sfb=11;sfb<16;sfb++)
                scalefac_l[gr][ch][sfb]                         0..3
if((scfi[ch][3]==0 ll(gr==0))
        for(sfb=16;sfb<21;sfb++)
                scalefac_l[gr][ch][sfb]                         0..3
        }
Huffmancodebit()
        }
```

}

for(b=0;b<no_of_ancillarybits;b++)

ancillary_bit

}

ข.1.7 ส่วนการเข้ารหัสแบบฮัฟแมน

```
Huffmancodebits() {
        For(I=0; I<big_value*2;I+2) {
            hcod[|x|][|y|]                          0..19
        if (|x|==15&&linbits>0)
            linbitsx                                1..13
        if (x!=0)
            signx                                   1
        if (|y|==15&&linbits>0)
            linbitsy                                1..13
        if (y!=0)
            signy                                   1
        is[I] =x
            is[I+I] = y
        }
        for(;I<big_values*2+count*4;I+=4){
            hcod[|v|][|w|][|x|][|y|]                1..6
        if(v!=0)
            signv                                   1
        if(w!=0
            signw                                   1
        if(x!=0)
            signx                                   1
        if(y!=0)
```

```
                signy                                                    1
                is[I] =v
                is[I+1] =w
                is[I+2] =x
                is[I+3] =y
}
        for(;I<576;I++)
                is[I] = 0
}
```

ข.1.8 ข้อมูลช่วย (Ancillary data)

```
ancillary_data(){
        If ((layer==1)||(layer==2))
                        For(b=0;b<no_of_ancillary_bits;b++)
                ancillary_bit                                            1
}
```

ข.2 ความหมายของคำในไวยากรณ์ของข้อมูลเอ็มเป็กเลเยอร์ 3

ข.2.1 ลำดับสัญญาณเสียงทั่ว ๆ ไป

frame ส่วนของลำดับข้อมูลที่สามารถถอดรหัสได้ บรรจุข้อมูล 1152 สัญญาณสุ่มความถี่

ข.2.2 เฟรมของสัญญาณ (Audio Frame)

header ส่วนของลำดับข้อมูล (bitstream) ที่บรรจุข้อมูลเกี่ยวกับ การให้ความสัมพันธ์ (synchronization) และข้อมูลทั่วไป

error_check ส่วนของลำดับข้อมูลที่บรรจุข้อมูลเกี่ยวกับการหาข้อผิดพลาด

audio_data ส่วนของลำดับข้อมูลที่บรรจุข้อมูลเกี่ยวกับข้อมูลเสียงที่สุ่มตัวอย่าง

ancillary_data ส่วนของลำดับข้อมูลที่บรรจุข้อมูลเพิ่มเติม

ข.2.3 ส่วนหัวของข้อมูล (Header)

32 บิทแรกของลำดับข้อมูล บรรจุข้อมูลที่บอกถึงลักษณะทั่ว ๆ ไป ทั้งหมดของการเข้ารหัส

syncword ส่วนส่วนหัวของลำดับข้อมูลเอ็มเป็ก เท่ากับ "1111 1111 1111"

ID บิทแสดงถึงมาตรฐานการเข้ารหัส "1" หมายถึงเข้ารหัสตามมาตรฐาน ISO 11172-3 "0" ไม่ใช่มาตรฐาน ISO

Layer แสดงเลเยอร์ ที่กำลังถอดรหัสอยู่ ดังตารางที่ ข.1

ตารางที่ ข.1 ความหมายของรหัสข้อมูลในเลเยอร์

เลเยอร์	ความหมาย
"11"	เลเยอร์ 1
"10"	เลเยอร์ 2
"01"	เลเยอร์ 3
"00"	ไม่ใช้

protect_bit บอกให้ทราบว่า ข้อมูลเสริมเกี่ยวกับการตรวจสอบความผิดพลาดได้บันทึกมาด้วยหรือไม่โดย

"1" ไม่มีข้อมูลเสริม

"0" มีข้อมูลเสริม

bitrate_index บอกอัตราการบีบอัดข้อมูล (bitrate) ที่ใช้

sampling_frequency บอกความถี่ในการสุ่มตัวอย่าง (sampling)

padding_bit เท่ากับ 1 แสดงว่าเฟรม ของข้อมูลนั้น ๆ บรรจุสล็อด (slot) .ใน layer 3 =1 byte เพิ่มเติมมา ถ้าเท่ากับ "0" แสดงว่าไม่ได้บรรจุ การบรรจุสล็อด เพื่อปรับอัตราส่วนระหว่างอัตราส่วนการบีบอัด กับความถี่ในการสุ่มตัวอย่าง sampling frequency ให้ลงตัว ในกรณีความถี่สุมตัวอย่าง 44.1 กิโลเฮิร์ต ไม่มี padding คือเท่ากับ 0 เสมอ

private ไม่ใช้ในการเข้ารหัสตามมาตรฐาน ISO/IEC 11172-3

copy right "1" ป้องกันการ copy "0" ไม่ป้องกันการ copy

original/copy "0" ข้อมูลนั้นถูก copy มา, "1" ข้อมูลนั้น เป็นต้นฉบับ

mode บอกเกี่ยวกับรูปแบบของข้อมูลว่าเป็นหนึ่งช่องเสียง (single channel), สองช่องเสียง(double channel), สเตอริโอ(stereo),จอย-สเตอริโอ(joint to stereo) ดังตารางที่ ข.2

ตารางที่ ข.2 ความหมายของรหัสข้อมูลใน Mode

Mode	ความหมาย
“00”	สเตอริโอ
“01”	จอย-สเตอริโอ
“10”	สองช่องเสียง
“11”	หนึ่งช่องเสียง

mode_extention ใช้บอกชนิดของจอย-สเตอริโอ (joint to stereo mode) ว่ามี ms-stereo, intensity-stereo หรือไม่อย่างไร ดังตารางที่ ข.3

ตารางที่ ข.3 ความหมายของรหัสข้อมูลใน Mode_extention

Mode_extention	Intensity	Ms_sterio
“00”	Off	Off
“01”	On	Off
“10”	Off	On
“11”	On	On

emphasis แสดงถึงชนิดของเอ็มพาไซส์ (emphasis) ที่ใช้งานแสดงในตารางที่ ข.4

ตารางที่ ข.4 ความหมายของรหัสข้อมูลใน Emphasis

Emphasis	ความหมาย
“00”	ไม่มีใช้
“01”	50/15 ไมโครวินาที
“10”	สงวน
“11”	CCITT J.17

ข.2.4 ส่วนตรวจสอบความผิดพลาด

crc check ข้อมูล 16 บิต เพื่อตรวจสอบ parity ของข้อมูล ว่าถูกต้องหรือไม่

ข.2.5 ส่วนข้อมูลเสียง

Main_data_begin แสดงตำแหน่งแรกของข้อมูลหลักของกรอบนั้น ๆ โดยเป็นค่าออฟเซ็ท (offset byte) เป็นค่าตำแหน่งที่ห่างออกมาจากไบต์แรกของซิงค์เวิร์ด (syncword) โดยไม่รวมจำนวนไบต์ของส่วนหัว (Header) และ ไซด์ อินฟรอเมชั่น (side information)

Private_ bits จะไม่ใช้มาตรฐาน ISO/IEC จำนวนของไปรเวต์บิท (private bit) ขึ้นอยู่กับจำนวนช่องเสียง, จำนวนของไปรเวต์บิท ถูกใช้พิจารณาเพื่อเทียบกับจำนวนของ bit ทั้งหมดในไซด์ อินฟรอเมชั่น (side information)

scfsi[ch][scfsi_band] ในเข้ารหัสแบบเอ็มเป็กเลเยอร์ 3 scfsi (scalefactor selection information) ให้ข้อมูลเกี่ยวกับค่า สเกลเฟคเตอร์ (scale factor) ของแต่ละย่านความถี่ (subband) และแต่ละช่องเสียง (ch) ค่า scfsi_band ถูกใช้เพื่อเลือกกลุ่มของสเกลเฟคเตอร์ การใช้งานของสเกลเฟคเตอร์ ในแต่ละแกรนนูล (granule) จะถูกควบคุมโดย scfsi ตารางที่ ข.5 แสดงความหมายของ scfsi[scfsi_band]

ตารางที่ ข.5 ความหมายของรหัสข้อมูลใน scfsi[scfsi_band]

scfsi[scfsi_band]	
"0"	เลือกใช้แต่ละ granule แยกกัน
"1"	ทั้ง 2 เลือกใช้ scale factor เดียวกัน

scfsi_band ควบคุมการใช้ scfsi สำหรับกลุ่มของสเกลเฟคเตอร์ แสดงความหมายใน ตาราง ก.6

ตารางที่ ข.6 ความหมายของรหัสข้อมูลใน scfsi_band

scfsi_band	**ย่านความถี่ที่ถูกใช้งาน**
0	0..5
1	6..10
2	11..15
3	16..20

part2_3_length[gr][ch] บอกจำนวนของ bit ของข้อมูลในส่วนที่เข้ารหัสแบบฮัฟแมน (Huffman code) และสเกลเฟคเตอร์เพื่อใช้หาตำแหน่งเริ่มต้นของ main information สำหรับแกรนูล (granule) ถัดไป

big_values[gr][ch] ค่าที่ถูกเข้ารหัสโดยใช้ รหัสฮัฟแมน (Huffman code) โดยอ้างค่าจากตารางฮัฟแมน (Huffman code table)

global_gain{gr][ch] เป็นตัวแปรที่ใช้ในขั้นตอนการรีควอนไตซ์ จะกล่าวถึงอีกครั้งในเรื่องการรีควอนไตซ์ในบทที่ 5

scalefac_compress[gr][ch] เลือกจำนวนของบิทที่ถูกใช้สำหรับ ส่งค่า สเกลเฟคเตอร์ (scalefactor) ดังตารางที่ ข.7

ตารางที่ ข.7 ความหมายของรหัสข้อมูลใน scalefac_compress[gr][ch]

scalefac_compress[gr][ch]	slen1	slen
0	0	0
1	0	1
2	0	2
3	0	3
4	3	0
5	1	1
6	1	2
7	1	3
8	2	1
9	2	2
10	2	3
11	3	1
12	3	2
13	3	3
14	4	2
15	4	3

block_type 0,1,3

slen1 : เป็นความยาวของ scale factor ย่าน 0 ถึง 10

slen2 : เป็นความยาวของ scale factor ย่าน 11 ถึง 20

block_type 2 และ mixed_block_flag เป็น 0

slen1 : เป็นความยาวของ scale factor ย่าน 0 ถึง 5

slen2 : เป็นความยาวของ scale factor ย่าน 6 ถึง 11

block_type 2 และ mixed_block_flag เป็น 1

slen1 : เป็นความยาวของ scale factor ย่าน 0 ถึง 7 (ย่าน long window)

: เป็นความยาวของ scale factor ย่าน 3 ถึง 5 (ย่าน short window)

slen2 : เป็นความยาวของ scale factor ย่าน 6 ถึง 11

window_switching_flag[gr][ch] แสดงว่าข้อมูลไม่ได้ใช้วินโดว์ (window) ปกติ (type=0) ในกรณีที่ window_switching_flag = 1 จะมีผลดังนี้

region0_count = 7 กรณี **block_type ==1** หรือ **block_type ==3**

หรือ **block_type ==2** และ mixed_block_flag

region0_count = 8 **block_type ==2** และ not mixed_block_flag

region0_count = 36 ค่าทั้งหมดของ **big_value** อยู่ใน **region 1**

ถ้า window_switching_flag = 0 แล้วค่า block_type = 0

block_type[gr][ch] ชี้ชนิดของ window ที่ใช้ของ granule นั้นแสดงความหมายในตารางที่ ข.8

block_type และ **mixed_block_flag** ให้ข้อมูลเกี่ยกับค่าที่อยู่ในบล็อค เกี่ยวกับความยาว, และการนับสำหรับการแปลง ถ้า window_switching_flag == 1 mixed_block_flag จะเป็นตัวชี้ว่า lower frequency poly phase ย่านใดถูกถอดรหัส โดยใช้ type ปกติ

ในกรณีบล็อคยาว (block_type ไม่ใช่2, หรือ ย่านต่ำที่ block type เป็น 2 เมื่อ mix block flag = 1) IMDCT จะให้เอาท์พุท 36 ค่าทุก ๆ อินพุท 18 ค่า เอาท์พุทจะขึ้นอยู่กับ block_type

ในกรณีบล็อคสั้น (ย่านที่เหนือกว่าบล็อคยาวของ block_type2 เมื่อ mix block flag = 1 หรือทุกย่านความถี่ของ block_type2 เมื่อ mixed_ block flag = 0) IMDCT ให้เอาท์พุท 12 ค่า

mixed_block_flag[gr][ch] เป็นตัวกำหนดการแปลงที่ความถี่ต่ำว่าใช้ block_type แบบใด

- กรณี **mixed_block_flag** เป็น 0 : ทุก block ถูกแปลงค่าโดยชี้จาก block_type [gr][ch]
- กรณี **mixed_block_flag** เป็น 1 : ย่านความถี่ต่ำสุด 2 ย่านถูกกำหนดให้แปลงด้วย window ปกติ 30 ย่านความถี่ที่เหลือ ถูกแปลงค่าโดยชี้จาก block_type[gr] [ch]

table_select[gr][ch][region] ใช้เลือกตารางฮัฟแมน (Huffman) จาก 32 ตาราง

subblock_gain[gr][ch][window] บอกค่าอัตราการขยายที่เพิ่มขึ้น/ลดลงจาก grobal_gain ของแต่ละบล็อคย่อย (subblock)

ตารางที่ ข.8 ความหมายของรหัสข้อมูลใน block_type[gr]

block_type[gr]	
0	ไม่ใช้
1	เริ่มต้นบล็อค
2	วินโดว์บล็อคสั้นสามบล็อค
3	ท้ายบล็อค

region0_count[gr][ch] ส่วนของสเป็คตรัม (Spectrum) ที่ถูกแบ่งเพื่อใช้เพิ่มความสามารถของตัวเข้ารหัสให้มีความถูกต้องยิ่งขึ้น โดยแต่ละส่วนที่ถูกแบ่งจะถูกเรียกว่า region 0,1,2 แต่ละ region จะใช้ตารางถอดรหัสฮัฟแมน (Huffman Table) ต่างกันขึ้นอยู่กับ maximun quntized value และ siganl statistic ค่าของ region0_count และ region1_count ถูกใช้ชี้ขอบเขตของ region นั้น ๆ

region1_count[gr][ch] นับจำนวนของสเกลแฟคเตอร์ใน region1 ลบออก 1

preflag[gr][ch] เป็นค่าเสริมเพื่อขยายค่าที่ความถี่สูง ถ้าค่า preflag ถูกตั้งจะนำค่าในตาราง preglag ไปคูณกับค่าของ scale factor อีกครั้ง ในกรณี block type เท่ากับ 2 preflag ไม่ถูกใช้

scalefac_scale[gr][ch] เป็นสเกลแฟคเตอร์ (scale factor) ที่ถูกปรับค่า (quantize) ด้วยสเกลล็อกการิทึม โดยคูณค่าของแต่ละลำดับด้วย 2 หรือ $\sqrt{2}$ ขึ้นอยู่กับ scalefac_scale ดังแสดงความหมายในตารางที่ ข.9

ตารางที่ ข.9 ความหมายของรหัสข้อมูลใน Scalfac_scale[gr]

scalfac_scale[gr]	scalefac_multipier
0	0,5
1	1

count1table_select[gr][ch] ใช้เลือกค่าจากตารางฮัฟแมน (Huffman) B7.A หรือ B7.B เมื่อค่าที่ปรับระดับ (quantize) ใน region คูณด้วย 4 แล้วไม่เกิน 1 ดังแสดงความหมายของตารางที่ ข.10

ตารางที่ ข.10 ความหมายของรหัสข้อมูลใน count1table_select[gr][ch]

count1table_select[gr][ch]	
0	ตาราง B7.A
1	ตาราง B7.B

scalefac_l[gr][ch], scalefac_s[gr][ch][sfb][window], is_pos[sfb] ถูกใช้ในการปรับระดับค่าคืน (requantization)

huffmancodedbits() ข้อมูลที่ถูกเข้ารหัสฮัฟแมน (Huffman)
รูปแบบของ Huffmancodecbit() แสดงวิธีการเข้ารหัส ข้อมูลลงใน big_value โดยข้อมูลที่ได้จะเป็นคู่ (x,y) ให้ค่าแต่ละค่ามีจำนวนบิทน้อยกว่า 15 บิต การเข้ารหัสจะถูกเลือกจากตาราง Huffman 0 ถึง 31 ถ้าข้อมูลที่เข้ามามีขนาดเกิน 15 บิต จะแยกเข้ารหัสต่างหาก ถ้าในคู่ของข้อมูลที่เข้ารหัสไม่เป็น 0 จะปรากฏเครื่องหมาย ให้ค่า + หรือ –

ในตาราง Huffman จะบรรจุ องค์ประกอบ 3 ส่วนคือ

- **hcod [x] [y]** รหัสข้อมูล huffman
- **hlen [x] [y]** ความยาวข้อมูล huffman
- **linbits** ความยาวของ linbitsX หรือ linbitsY เมื่อถูกเข้ารหัส

รูปแบบของส่วนประกอบ ของ huffmancodebits ประกอบด้วยกลุ่มของข้อมูลประกอบดังนี้

- **sig nv** เครื่องหมายของค่า v (0 เป็นบวก 1 เป็นลบ)

- **sig nw** เครื่องหมายของค่า w (0 เป็นบวก 1 เป็นลบ)
- **sig nx** เครื่องหมายของค่า x (0 เป็นบวก 1 เป็นลบ)
- **sig ny** เครื่องหมายของค่า y (0 เป็นบวก 1 เป็นลบ)
- **litbitsX** ใช้เมื่อเข้ารหัสถ้าขนาดของ x เกิน 15
- **litbitsY** ใช้เมื่อเข้ารหัสถ้าขนาดของ x เกิน 15
- **is [i]** คือ ค่าที่ถูก quatized สำหรับ frequency line ที่ 1

ข.2.6 Ancillary data

ancillary_bit ผู้ใช้สามารถนิยามเอง

ภาคผนวก ค
โปรแกรมถอดรหัสเอ็มเป็กเลเยอร์3 ด้วยโปรแกรมแมทแลบ

โปรแกรมที่ใช้หา Syncword และเก็บ Header ถ้ามี protection bit จะทำการถอดรหัส CRC เพื่อตรวจสอบว่าบิทที่ได้รับมีความถูกต้องหรือไม่ นอกจากนี้ยังแสดงผลว่า เฟรมแต่ละเฟรมมีคุณสมบัติตาม ISO และเป็น Layer 3 หรือไม่ ซึ่งถ้ามีความถูกต้องเราจะทำการถอดรหัส

ในการใช้โปรแกรม MATLAB กับ ไฟล์ mp3 นั้น เราจะทำการแปลงไฟล์ให้อยู่ในรูปบิทสตรีม ด้วยโปรแกรม bitstreams จากนั้นจะเรียกใช้โปรแกรม set1 เพื่อเรียกบิทสตรีมนี้เพื่อใช้ต่อไป และเซ็ทค่าเริ่มต้นต่าง จากนั้นเรียกโปรแกรม inform โดยโปรแกรมนี้จะประกอบด้วยโปรแกรมย่อย ได้แก่ โปรแกรม information เป็นโปรแกรมที่เก็บ Header โปรแกรม status เป็นโปรแกรมที่ตรวจสอบ Layer, ID และโปรแกรม crc_1 เป็นโปรแกรมในการถอดรหัส CRC (นอกจากนี้ยังมีฟังก์ชั่นพิเศษที่เพิ่มเติม คือ function arlint เป็นฟังก์ชั่นที่เขียนขึ้นแปลง bitstream ให้อยู่ในรูปจำนวนเต็ม)

ในการเก็บค่า side information นั้น จะเรียกใช้โปรแกรม audio จากนั้นเป็นการถอดรหัสเอาส่วน scalefactor และถอดรหัสส่วนที่เป็น Huffman Code bit ด้วยโปรแกรมย่อย huffcode3 ซึ่งในส่วนที่เป็นข้อมูลเหล่านี้จุดเริ่มของข้อมูลจะถูกกำหนดได้ด้วย main_data_begin จาก side information ดังนั้นในการเซ็ทจุดเริ่มจะมีการเรียกโปรแกรมย่อย offset และมีโปรแกรมย่อย inc_bit นอกจากนี้ยังมีฟังก์ชั่นพิเศษเพิ่มเติม คือ ฟังก์ชั่น getnbit (นำค่าจากบิตสตรีม) และ addnbit ทั้งหมดนี้ช่วยในกรณีที่ค่า main_data_begin ไม่เท่ากับ 0 โปรแกรมนี้จะช่วยจัดค่าตัวชี้ข้อมูลทำงานได้ถูกต้อง

ในการถอดรหัส Huffman จะใช้โปรแกรม Huffcode3 ซึ่งจะประกอบด้วยโปรแกรมย่อยในการถอดรหัส และการเลือกตาราง

จากนั้นเป็นการรีควอนไตซ์ด้วยโปรแกรม descale แล้วทำลดค่าปลอม ด้วยโปรแกรม alias

ต่อไปทำ IMDCT ด้วยโปรแกรม IMDCT จากนั้นนำผลลัพธ์ไปวินโดว์ ด้วยโปรแกรม windowing2 จากนั้นทำ Inverse Frequency และโฟลีเฟสฟิลเตอร์แบงก์ด้วยโปรแกรม p_f จะได้ PCM ออกมา

โปรแกรมแมทแลบ

1. โปรแกรม bitstreams ใช้สำหรับแปลงไฟล์เสียง (*.wav) ให้เป็นบิตสตรีมเพื่อให้โปรแกรมแมทแลบนี้นำไปใช้ต่อไป

```
clear all; cd c:\matlabr11\song3; fil=input('What"s File name with ? ... ','s'); [m,n]=size(fil);
fil2=fil; fil(n+1:n+4) = '.mp3'; fid = fopen(fil,'r'); F = fread(fid); [m,n] = size(F); length = 0;
for i = 1:m
  for j = 8:-1:1
    length = length+1 ;
    bitstream(length) = bitget(F(i),j);
  end
end
fclose all; save(fil2); cd c:\matlabr11\work2;
```

2. โปรแกรม set1 ใช้สำหรับให้ค่าเริ่มต้นของการถอดรหัสต่างและโหลดข้อมูลต่างๆขึ้นมา

```
clear all; cd c:\matlabr11\song3; fil=input('What"s File name ? ... ','s'); load(fil);
cd c:\matlabr11\work2; [a,length] = size(bitstream); nbit = 1; s_ovl=0; init=0; nt=0;
```

3. โปรแกรม inform เป็นโปรแกรมถอดรหัสเอ็มเป็ก 1 เลเยอร์ 3 ซึ่งต้องให้โปรแกรม set1 ทำงานก่อน

```
clc
% i,j,k is variable that not fixed  % length is the number of bit .....file mp3
% bitstream is array of bit .... file mp3  % nch is considered from MODE
% nbit is pointer , the value is the position that is considered
%load bit2.mat %start
find_sync = 0;  endb=0;  sw=1;
while endb==0
  clc
  disp('------------------------------------------------------')
  disp('******** Find Syncword and Header ********')
  for i = nbit:length
    if i+11 <= length
     if bitstream(i:i+11) == [1 1 1 1 1 1 1 1 1 1 1 1]
        find_sync = 1; % Get syncword
        break
     end
    else
      disp('End of Bits'); endb=1;
    end
  end
  if find_sync == 1
    find_sync = 0;
```

```
else
  break
end
nbit = i;  disp('position of syncword');  disp(nbit);
disp('-----------------------------------------------------------')
disp(' '); strt = nbit; nbit = nbit+12;
information   % Get information (Header)
error1 = 0;
status %check for layer,id
  if protection_bit == 0   % Error check
  disp('CRC CHECK STATUS');
  k = 0;
  for j = nbit:nbit+15
    k = k+1;
    crc_check(k) = [ bitstream(j) ];
  end
  nbit = nbit + 16;
  crc_1; %crc operate
  if crc_check2 ~= crc_check
    disp('CRC CHECK ERROR')
        error1 = 1;
  else disp('CRC CHECK RIGHT')
  end
end
disp('-----------------------------------------------------------')
if error1 == 1
  disp('THERE IS NO OPERATION WITH THIS FRAME')
else
  if init ~= 0
    if sam_f ~= sampling_frequency | mode ~= modes | bitr ~= arlint(bitrate_index)
      disp('THIS FRAME HASN''T THE CORRECT FORMAT');
      error1 =1;
    else
      disp('START DECODE THIS FRAME')
    end
  else
    disp('START DECODE THIS FRAME')
  end
end
if error1 == 0
```

```
    pause ; audio ; main_data ;
  end
  pause ; sw=0;
end
```

ภายในโปรแกรม inform นี้จะมีโปรแกรมย่อย ดังนี้

- โปรแกรม information เป็นการเก็บ Header

```
%subprogram from inform.m % Get Header
ID = [ bitstream(nbit) ]; layer = [ bitstream(nbit+1) bitstream(nbit+2)]; protection_bit = [ bitstream(nbit+3) ];
k = 0 ;
for j = nbit+4 : nbit+7
  k = k+1 ;
  bitrate_index(k) = [ bitstream(j) ];
end
sampling_frequency = [ bitstream(nbit+8) bitstream(nbit+9)]; padding_bit = [ bitstream(nbit+10) ];
private_bit = [ bitstream(nbit+11) ]; mode = [ bitstream(nbit+12) bitstream(nbit+13) ];
mode_extension = [ bitstream(nbit+14) bitstream(nbit+15) ]; copyright = [ bitstream(nbit+16) ];
original_copy = [ bitstream(nbit+17)]; emphasis = [ bitstream(nbit+18) bitstream(nbit+19) ];
nbit = nbit+20;
if init==0
  sam_f = sampling_frequency; modes = mode; bitr = arlint(bitrate_index);
end
```

- โปรแกรม status เป็นการตรวจสอบคุณสมบัติของเฟรมว่ามีความถูกต้องที่จะใช้ในการถอดรหัสหรือไม่

```
if ID == [1]
  disp('ISO')
else
  disp('Not ISO')
  error1 = 1;
end
if layer == [0 1]
  disp('Layer III')
else
  disp('Not Layer III')
  error1 = 1;
  if layer == [1 0]
    disp('Layer II')
  elseif layer == [1 1]
    disp('Layer I')
```

```
    else
      disp('reserved')
    end
  end
  if protection_bit == [0]
    disp('protection bit')
  else
    disp('No protection bit')
  end
  if layer == [0 1]
    if arlint(bitrate_index) == 0
      disp('bitrate => free');
    elseif arlint(bitrate_index) == 1
      disp('bitrate => 32 kbit/s');
    elseif arlint(bitrate_index) == 2
      disp('bitrate => 40 kbit/s');
    elseif arlint(bitrate_index) == 3
      disp('bitrate => 48 kbit/s');
    elseif arlint(bitrate_index) == 4
      disp('bitrate => 56 kbit/s');
    elseif arlint(bitrate_index) == 5
      disp('bitrate => 64 kbit/s');
    elseif arlint(bitrate_index) == 6
      disp('bitrate => 80 kbit/s');
    elseif arlint(bitrate_index) == 7
      disp('bitrate => 96 kbit/s');
    elseif arlint(bitrate_index) == 8
      disp('bitrate => 112 kbit/s');
    elseif arlint(bitrate_index) == 9
      disp('bitrate => 128 kbit/s');
    elseif arlint(bitrate_index) == 10
      disp('bitrate => 160 kbit/s');
    elseif arlint(bitrate_index) == 11
      disp('bitrate => 192 kbit/s');
    elseif arlint(bitrate_index) == 12
      disp('bitrate => 224 kbit/s');
    elseif arlint(bitrate_index) == 13
      disp('bitrate => 256 kbit/s');
    elseif arlint(bitrate_index) == 14
      disp('bitrate => 320 kbit/s');
```

```
    elseif arlint(bitrate_index) == 15
      disp('bitrate => forbidden');
    end
  end
  if sampling_frequency == [0 0]
    disp('sampling_frequency = 44.1 kHz')
  elseif sampling_frequency == [0 1]
    disp('sampling_frequency = 44.1 kHz')
  elseif sampling_frequency == [1 0]
    disp('sampling_frequency = 44.1 kHz')
  else
    disp('sampling fresquency = 1 1 is reserved')
  end
  if arlint(mode)==0
    disp('Stereo Mode')
  elseif arlint(mode)==1
    disp('Joint Stereo and/or Ms_Stereo')
  elseif arlint(mode)==2
    disp('Dual Channel')
  else
    disp('Single Channel')
  end
```

จากนั้น โปรแกรม inform จะเชื่อมโยงไปยังโปรแกรมอื่นๆเอง และจะแสดงผลที่ได้ตามลำดับ ซึ่งโปรแกรมอื่นได้แก่

3.1 โปรแกรม crc_1 เป็นโปรแกรมตรวจสอบ crc

```
i=0;
if mode == [1 1]
  for n = nbit+184:-1:nbit+49
    i = i + 1; bs(i) = bitstream(n);
  end
  for n = nbit+32:-1:nbit+17
    i = i + 1; bs(i) = bitstream(n);
  end
else
  for n = nbit+304:-1:nbit+49
    i = i + 1; bs(i)= bitstream(n);
  end
  for n = nbit+32:-1:nbit+17
    i = i + 1; bs(i)= bitstream(n);
```

```
  end
end
g_x = [1 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1];
x_16 =[1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
c = conv(x_16,bs);
[m2,n2] = size(g_x);
for k = 1:n2
  if g_x(k) == 1
    deg2 = n2-k+1;
    break
  end
end
[m1,n1] = size(c);
for j = 1:n1
  if c(j) == 1
    deg1 = n1-j+1;
    break
  end
end
while deg1 >= deg2
  g_x2 = g_x;
  delta = deg1 - deg2;
  for ii = 1:delta
    g_x2(n2+ii) = 0; % shift bit for seeking multi(divider)
  end
  [m3,n3]=size(g_x2);
  if n1 ~= n3
    g_x3 = zeros(1,n1); % n1 ~= n2 cann't xor then build vetor for g_x
    i = n1;
    for k = n2+delta:-1:1
      g_x3(i) = g_x2(k); % complete build g_x
      i=i-1;
    end
    g_x2 = g_x3;
  end
  c = xor(c,g_x2); % can xor
  [m1,n1] = size(c);
  for j = 1:n1
    if c(j) == 1
      deg1 = n1-j+1; % find degree of data for end  condition that degree 1 < degree 2
```

```
      break
    end
  end
end
%----------complete and then find 16 bit last to compare with crc check---------
for j = 1:n1
  if c(j) == 1
    k=j;% find degree of data
    break
  end
end
i=0;
for j=k:n1
  i=i+1; c2(i) = c(j); % c2 is data that not equal 0 first bit
end
% ---------------- upper ... find the first data that 1 first--------
% ---------------- lower ... cut out 16 bit last ---------------------
i=0;
for j = n1-15:n1
  i=i+1; crc_check2(i)=c(j);
end
```

3.2 โปรแกรม audio ใช้สำหรับเก็บ side information

```
% scfsi[ch][scfsi_band] => scfsi(ch+1,scfsi_band+1) % part2_3length[gr][ch] => part2_3length{gr+1,ch+1}
% Considered channel by Mode
clc ; disp('******** Get Side_information ********');
if mode == [1 1]
  nch = 1;
else
  nch = 2;
end
for i=1:9
  main_data_begin(i) = bitstream(nbit); nbit = nbit + 1;
end
disp('Main_data_begin'); disp(arlint(main_data_begin));
disp('-------------------------------------------------------')
if mode == [1 1]
  for i=1:5
    private_bits(i) = bitstream(nbit); nbit = nbit + 1;
  end
```

```
else
  for i=1:3;
    private_bits(i) = bitstream(nbit); nbit = nbit + 1;
  end
end
for ch=0:nch-1
  for scfsi_band=0:3
    scfsi(ch+1,scfsi_band+1) = bitstream(nbit); nbit=nbit+1;
  end
end
for gr=0:1
  for ch=0:nch-1
    disp('grannule'); disp(gr); disp('channel'); disp(ch);
      for k=1:12
      part2_3length{gr+1,ch+1}(k) = bitstream(nbit); nbit = nbit + 1;
      end
      disp('part2_3length'); disp(arlint(part2_3length{gr+1,ch+1}));
      for k=1:9
        big_values{gr+1,ch+1}(k) = bitstream(nbit); nbit = nbit + 1;
      end
      for k=1:8
        global_gain{gr+1,ch+1}(k) = bitstream(nbit); nbit = nbit + 1;
      end
      for k=1:4
        scalefac_compress{gr+1,ch+1}(k) = bitstream(nbit); nbit = nbit + 1;
      end
      window_switching_flag(gr+1,ch+1) = bitstream(nbit);
      if window_switching_flag(gr+1,ch+1) == 1
        disp('window_switching_flag = 1')
      else
        disp('window_switching_flag = 0')
      end
      nbit = nbit + 1;
      if window_switching_flag(gr+1,ch+1)
        for k=1:2
          block_type{gr+1,ch+1}(k) = bitstream(nbit); nbit = nbit + 1;
        end
        disp('block_type'); disp(arlint(block_type{gr+1,ch+1}));
        mixed_block_flag(gr+1,ch+1) = bitstream(nbit);
        if mixed_block_flag(gr+1,ch+1)== 1
```

```
        disp('mixed_block_flag = 1')
      else
        disp('mixed_block_flag = 0')
      end
      nbit = nbit + 1;
      for region=0:1
        for k=1:5
          table_select{gr+1,ch+1,region+1}(k) = bitstream(nbit); nbit = nbit + 1;
        end
      end
      disp('There are 2 region in part big_values in Huffman')
      for window = 0:2
        for k=1:3
          subblock_gain{gr+1,ch+1,window+1}(k) = bitstream(nbit);
          nbit = nbit + 1;
        end
      end
    else
  disp('no block_type (block_type =0) / no mixed_block_flag (window_switching_flag =0)');
  disp('There are 3 region in part bigvalues in Huffman')
      for region=0:2
        for k=1:5
          table_select{gr+1,ch+1,region+1}(k) = bitstream(nbit); nbit = nbit + 1;
        end
      end
      for k=1:4
        region0_count{gr+1,ch+1}(k) = bitstream(nbit); nbit = nbit + 1;
      end
      for k=1:3
        region1_count{gr+1,ch+1}(k) = bitstream(nbit); nbit = nbit + 1;
      end
    end
    preflag(gr+1,ch+1) = bitstream(nbit); scalefac_scale(gr+1,ch+1) = bitstream(nbit+1);
    count1table_select(gr+1,ch+1) = bitstream(nbit+2); nbit = nbit + 3;
  end
  disp('------------------------------------------------------'); pause;
end
```

3.3 โปรแกรม main_data เป็นโปรแกรมที่ใช้เก็บค่า scalefactor

```
offset; %scalefac  --- main data
```

```
slen1 = [0 0 0 0 3 1 1 1 2 2 2 3 3 3 4 4]; slen2 = [0 1 2 3 0 1 2 3 1 2 3 1 2 3 2 3];
part2_length = zeros(2,nch);
for gr=0:1
  for ch=0:nch-1
    clc
    disp('******** Decode Scale_Factor ********');
    disp('Grannule'); disp(gr); disp('Channel'); disp(ch); disp('slen1');
    disp(slen1(arlint(scalefac_compress{gr+1,ch+1})+1)); disp('slen2');
    disp(slen2(arlint(scalefac_compress{gr+1,ch+1})+1));
    if  window_switching_flag(gr+1,ch+1) == 1 & block_type{gr+1,ch+1} == [1 0]
      if mixed_block_flag(gr+1,ch+1)
        for sfb=0:7
          if slen1(arlint(scalefac_compress{gr+1,ch+1})+1)== 0
            scalefac_l{gr+1,ch+1,sfb+1} = 0;
          end
          for k=1:slen1(arlint(scalefac_compress{gr+1,ch+1})+1)
            bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
            scalefac_l{gr+1,ch+1,sfb+1}(k) = bs; inc_nbit;
          end
        end
        disp('Get scalefac_l for 8 subbands ')
        for sfb=3:5
          for window=0:2
            if slen1(arlint(scalefac_compress{gr+1,ch+1})+1)== 0
              scalefac_s{gr+1,ch+1,sfb+1,window+1} = 0;
            end
            for k=1:slen1(arlint(scalefac_compress{gr+1,ch+1})+1)
              bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
              scalefac_s{gr+1,ch+1,sfb+1,window+1}(k) = bs; inc_nbit;
            end
          end
        end
        for sfb=6:11
          for window=0:2
            if slen2(arlint(scalefac_compress{gr+1,ch+1})+1)== 0
              scalefac_s{gr+1,ch+1,sfb+1,window+1} = 0;
            end
            for k=1:slen2(arlint(scalefac_compress{gr+1,ch+1})+1)
              bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
              scalefac_s{gr+1,ch+1,sfb+1,window+1}(k) = bs; inc_nbit;
```

```
            end
          end
        end
        disp('Get scalefac_s for 9 subbands / each subband 3 windows')
        % decode scalefactor for block_type=2(short blocks) and mixed_block_flag=1
 part2_length(gr+1,ch+1) = 17*slen1(arlint(scalefac_compress{gr+1,ch+1})+1)+18*slen2(arlint(scalefac_compress
{gr+1,ch+1})+1);
      else
        for sfb=0:5
          for window=0:2
            if slen1(arlint(scalefac_compress{gr+1,ch+1})+1)== 0
              scalefac_s{gr+1,ch+1,sfb+1,window+1} = 0;
            end
            for k=1:slen1(arlint(scalefac_compress{gr+1,ch+1})+1)
              bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
              scalefac_s{gr+1,ch+1,sfb+1,window+1}(k) = bs; inc_nbit;
            end
          end
        end
        for sfb=6:11
          for window=0:2
            if slen2(arlint(scalefac_compress{gr+1,ch+1})+1)== 0
              scalefac_s{gr+1,ch+1,sfb+1,window+1} = 0;
            end
            for k=1:slen2(arlint(scalefac_compress{gr+1,ch+1})+1)
              bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
              scalefac_s{gr+1,ch+1,sfb+1,window+1}(k) = bs; inc_nbit;
            end
          end
        end
        disp('Get scalefac_s for 12 subbands / each subband 3 windows')
      end
      % decode scalefactor for block_type=2(short blocks) and mixed_block_flag=0
      part2_length(gr+1,ch+1) = 18*slen1(arlint(scalefac_compress{gr+1,ch+1})+1)+18*slen2(arlint(scalefac_compress
{gr+1,ch+1})+1);
    else
      if scfsi(ch+1,0+1)==[0] | gr==0
        for sfb=0:5
          if slen1(arlint(scalefac_compress{gr+1,ch+1})+1)== 0
            scalefac_l{gr+1,ch+1,sfb+1} = 0;
```

```
      end
      for k=1:slen1(arlint(scalefac_compress{gr+1,ch+1})+1)
        bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
        scalefac_l{gr+1,ch+1,sfb+1}(k) = bs; inc_nbit;
      end
    end
    if gr == 1
        % decode scalefactor for block_type=0,1,3(long blocks)
        part2_length(gr+1,ch+1) = part2_length(gr+1,ch+1) + 6*slen1(arlint(scalefac_compress{gr+1,ch+1})+1);
    end
  end
  if scfsi(ch+1,1+1)==[0] | gr==0
    for sfb=6:10
      if slen1(arlint(scalefac_compress{gr+1,ch+1})+1)== 0
        scalefac_l{gr+1,ch+1,sfb+1} = 0;
      end
      for k=1:slen1(arlint(scalefac_compress{gr+1,ch+1})+1)
        bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
        scalefac_l{gr+1,ch+1,sfb+1}(k) = bs; inc_nbit;
      end
    end
    if gr== 1
          % decode scalefactor for block_type=0,1,3(long blocks)
        part2_length(gr+1,ch+1) = part2_length(gr+1,ch+1) + 5*slen1(arlint(scalefac_compress{gr+1,ch+1})+1);
    end
  end
  if scfsi(ch+1,2+1)==[0] | gr==0
    for sfb=11:15
      if slen2(arlint(scalefac_compress{gr+1,ch+1})+1)== 0
        scalefac_l{gr+1,ch+1,sfb+1} = 0;
      end
      for k=1:slen2(arlint(scalefac_compress{gr+1,ch+1})+1)
        bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
        scalefac_l{gr+1,ch+1,sfb+1}(k) = bs; inc_nbit;
      end
    end
    if gr == 1
        % decode scalefactor for block_type=0,1,3(long blocks)
        part2_length(gr+1,ch+1) = part2_length(gr+1,ch+1) + 5*slen2(arlint(scalefac_compress{gr+1,ch+1})+1);
    end
```

```
    end
    if scfsi(ch+1,3+1)==[0] | gr==0
      for sfb=16:20
        if slen2(arlint(scalefac_compress{gr+1,ch+1})+1)== 0
          scalefac_1{gr+1,ch+1,sfb+1} = 0;
        end
        for k=1:slen2(arlint(scalefac_compress{gr+1,ch+1})+1)
          bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
          scalefac_1{gr+1,ch+1,sfb+1}(k) = bs; inc_nbit;
        end
      end
      if gr == 1
        % decode scalefactor for block_type=0,1,3(long blocks)
        part2_length(gr+1,ch+1) = part2_length(gr+1,ch+1) + 5*slen2(arlint(scalefac_compress{gr+1,ch+1})+1);
      end
    end
    if gr == 0
      % decode scalefactor for block_type=0,1,3(long blocks)
      part2_length(gr+1,ch+1) = 11*slen1(arlint(scalefac_compress{gr+1,ch+1})+1) + 10*slen2(arlint(scalefac_compress
{gr+1,ch+1})+1);
    end
    disp('window_switching_flag = 0'); disp('scfsi'); disp(scfsi(ch+1,:));
    disp('Get scalefac_1 in grannule = 1 or scfsi[ch][x]=0 ');
  end
  if window_switching_flag(gr+1,ch+1) == 0
    block_type{gr+1,ch+1} = [0 0];
  end
  disp('part2_3length'); disp(arlint(part2_3length{gr+1,ch+1})); disp(' ');
  disp('part2_length'); disp(arlint(part2_length(gr+1,ch+1)));
  disp('------------------------------------------------------------')
  pause;
  huffcode3;
  descale;
  alias;
  imdct;
  windowing2;
  overlap;
  p_f;
  %plot(Pcm)
  pause
```

```
    %close all
  end
end
```

3.3.1 โปรแกรม offset เป็นโปรแกรมที่ใช้ในการกำหนดส่วนข้อมูล

```
% begin of main_data
% now = nbit
% pointer after header & info = nbit_2
nbit_2 = nbit;
nbit_b = nbit - arlint(main_data_begin); % back to start main_data
nbit_f = strt;% forward to data after header & info nbit = nbit_2
nbit = nbit_b;
```

3.3.2 โปรแกรม huffcode3 เป็นโปรแกรมที่ใช้ในการถอดรหัสฮัฟแมน

```
% Consider sampling frequency
clc; disp('******** Huffman Decoding ********'); disp(' ');
if sampling_frequency == [0 0]
  disp('sampling_frequency = 44.1 kHz'); fs = 2 ; % 44,1 kHz;
elseif sampling_frequency == [0 1]
  disp('sampling_frequency = 48 kHz');  fs = 3 ; % 48 kHz;
elseif sampling_frequency == [1 0]
  disp('sampling_frequency = 32 kHz');  fs = 1 ; % 32 kHz;
else
  disp('sampling fresquency = 1 1 is reserved')
end
% Scalefactor bands
% ---------- 32,44.1,48 kHz sampling rate ----------
% long block
lsfw_l = [ 4 8 12 16 20 24 30 36 44 54 66 82 102 126 156 194 240 296 364 448 550 576;
       4 8 12 16 20 24 30 36 44 52 62 74 90 110 134 162 196 238 288 342 418 576;
       4 8 12 16 20 24 30 36 42 50 60 72 88 106 128 156 190 230 276 330 384 576];
% short block
lsfw_s = [ 4 8 12 16 20 24 28 32 36 40 44 48 64 60 66 74 82 90 102 114 126 142 158 174 194 214 234 260 286 312 346 380
414 456 498 540 576;
          4 8 12 16 20 24 28 32 36 40 44 48 54 60 66 74 82 90 100 110 120 132 144 156 170 184 198 216 234 252 274 296
318 348 378 408 576;
         4 8 12 16 20 24 28 32 36 40 44 48 54 60 66 72 78 84 94 104 114 126 138 150 164 178 192 208 224 240 260 280 300
326 352 378 576];
% mixed block flag 1
lsfw_m = [ 4 8 12 16 20 24 30 36 40 44 48 64 60 66 74 82 90 102 114 126 142 158 174 194 214 234 260 286 312 346 380 414
456 498 540 576;
```

```
        4 8 12 16 20 24 30 36 40 44 48 54 60 66 74 82 90 100 110 120 132 144 156 170 184 198 216 234 252 274 296 318
348 378 408 576;
        4 8 12 16 20 24 30 36 40 44 48 54 60 66 72 78 84 94 104 114 126 138 150 164 178 192 208 224 240 260 280 300
326 352 378 576];
sign = zeros(1,576); part3_length = arlint(part2_3length{gr+1,ch+1}) - part2_length(gr+1,ch+1);
disp('part3_length'); disp(part3_length); big_value = 2*arlint(big_values{gr+1,ch+1});
if window_switching_flag(gr+1,ch+1) == 1
    r(1) = 36; r(2) = big_value;
else
  reg1 = arlint(region0_count{gr+1,ch+1})+1;
  reg2 = arlint(region1_count{gr+1,ch+1})+1;
 if block_type{gr+1,ch+1} == [1 0]
   if mixed_block_flag(gr+1,ch+1) == 0
     r(1) = lsfw_s(fs,reg1); r(2) = lsfw_s(fs,reg2+reg1);
   else
     r(1) = lsfw_m(fs,reg1); r(2) = lsfw_m(fs,reg2+reg1);
   end
  else
    r(1) = lsfw_l(fs,reg1); r(2) = lsfw_l(fs,reg2+reg1);
  end
end
if big_value > 576
  r(3) = 576;
else r(3) = big_value;
end
if r(1) > r(3)
    r(1) = r(3); r(2) = r(3);
elseif r(2) > r(3)
    r(2) = r(3);
end
r %disp(r)
 huff_tab; %call table
 h = 0; huff_end = 0; huff_bit = 0;
 r1=1; r2=1; r3=1; r4=1;
 while h < r(3)
   if h < r(1)
     region = 0;
     if r1==1
     disp('region 1 use huffman table'); disp(arlint(table_select{gr+1,ch+1,region+1})); r1=0;
     end
```

```
    sel_tab; huff_dec1;
  elseif h < r(2)
    region = 1;
    if r2==1
    disp('region 2 use huffman table'); disp(arlint(table_select{gr+1,ch+1,region+1})); r2=0;
    end
    sel_tab; huff_dec1;
  elseif h < r(3)
    if r3==1
    region = 2; disp('region 3 use huffman table'); disp(arlint(table_select{gr+1,ch+1,region+1})); r3=0;
    end
    sel_tab; huff_dec1;
  end
end
while h+4 <= 576 & huff_bit < part3_length      % r(4)
    if r4==1
      disp('region 4')
      if count1table_select(gr+1,ch+1) == 0
        disp('use table A')
      else
        disp('use table B')
      end
      r4=0;
    end
    huff_tabab;%call table
    if count1table_select(gr+1,ch+1) == 0
        table = tablea;
    else
        table = tableb;
    end
     [o,p,q,z] = size(table); huff_dec_r41;
  end
  disp('nbit'); disp(nbit); disp('huff_bit - part3_length'); disp(huff_bit - part3_length);
if huff_bit > part3_length
  disp('part3_length is too short'); used=-1*(huff_bit - part3_length); nbit = addnbit(nbit,used,nbit_f,nbit_2);
elseif huff_bit < part3_length
  disp('part3_length is too far'); used=(part3_length-huff_bit); nbit = addnbit(nbit,used,nbit_f,nbit_2);
else disp('huff_bit = part3_length');
end
disp('new nbit'); disp(nbit); rzero = h;
```

```
while h < 576
  h = h+1; is(ch+1,h) = 0; sign(h) = 0;
end
disp('rzero => the start region of frequency line = 0  is'); disp(rzero);
for i=1:576
 if sign(i) == 1
   sgn = -1;
 else
   sgn = 1;
 end
 is2(i)=is(i)*sgn;
end
disp('the last bit'); disp(nbit); disp('-------------------------------------------------'); disp(' ')
if sw==1
figure(2)
bar(is)
colormap([0.6 0.2 0.8])
title('Huffman Decoder"s output => is (Absolute)')
xlabel('Sample frequency');
ylabel('|Size|');
end
pause
```

3.3.2.1โปรแกรม Huff_tab คือ ตารางฮัฟแมน 0-31

```
% Linbits Table
linbits = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 3 4 6 8 10 13 4 5 6 7 8 9 11 13];
% Huffman Table
table1 = {[1] [0 0 1];
  [0 1] [0 0 0]};
table2 = {[1] [0 1 0] [0 0 0 0 0 1];
  [0 1 1] [0 0 1] [0 0 0 0 1];
  [0 0 0 1 1] [0 0 0 1 0] [0 0 0 0 0 0]};
table3 = {[1 1] [1 0] [0 0 0 0 0 1];
  [0 0 1] [0 1] [0 0 0 0 1];
  [0 0 0 1 1] [0 0 0 1 0] [0 0 0 0 0 0]};
table5 = {[1] [0 1 0] [0 0 0 1 1 0] [0 0 0 0 1 0 1];
  [0 1 1] [0 0 1] [0 0 0 1 0 0] [0 0 0 0 1 0 0];
  [0 0 0 1 1 1] [0 0 0 1 0 1] [0 0 0 0 1 1 1] [0 0 0 0 0 0 0 1];
  [0 0 0 0 1 1 0] [0 0 0 0 0 1] [0 0 0 0 0 0 1] [0 0 0 0 0 0 0 0]};
table6 = {[1 1 1] [0 1 1] [0 0 1 0 1] [0 0 0 0 0 0 1];
  [1 1 0] [1 0] [0 0 1 1] [0 0 0 1 0];
  [0 1 0 1] [0 1 0 0] [0 0 1 0 0] [0 0 0 0 0 1];
  [0 0 0 0 1 1] [0 0 0 1 1] [0 0 0 0 1 0] [0 0 0 0 0 0 0]};
table7 = {[1] [0 1 0] [0 0 1 0 1 0] [0 0 0 1 0 0 1 1] [0 0 0 1 0 0 0 0] [0 0 0 0 0 1 0 1 0];
```

[0 1 1] [0 0 1 1] [0 0 0 1 1 1] [0 0 0 1 0 1 0] [0 0 0 0 1 0 1] [0 0 0 0 0 0 1 1];
[0 0 1 0 1 1] [0 0 1 0 0] [0 0 0 1 1 0 1] [0 0 0 1 0 0 0 1] [0 0 0 0 1 0 0 0] [0 0 0 0 0 0 1 0 0];
[0 0 0 1 1 0 0] [0 0 0 1 0 1 1] [0 0 0 1 0 0 1 0] [0 0 0 0 0 1 1 1 1] [0 0 0 0 0 1 0 1 1] [0 0 0 0 0 0 0 1 0];
[0 0 0 0 1 1 1] [0 0 0 0 1 1 0] [0 0 0 0 1 0 0 1] [0 0 0 0 0 1 1 1 0] [0 0 0 0 0 0 0 1 1] [0 0 0 0 0 0 0 0 0 1];
[0 0 0 0 0 1 1 0] [0 0 0 0 0 1 0 0] [0 0 0 0 0 0 1 0 1] [0 0 0 0 0 0 0 0 0 1 0] [0 0 0 0 0 0 0 0 1 0] [0 0 0 0 0 0 0 0 0 0 0]};
table8 = {[1 1] [1 0 0] [0 0 0 1 1 0] [0 0 0 1 0 0 1 0] [0 0 0 0 1 1 0 0] [0 0 0 0 0 0 1 0 1];
[1 0 1] [0 1] [0 0 1 0] [0 0 0 1 0 0 0 0] [0 0 0 0 1 0 0 1] [0 0 0 0 0 0 1 1];
[0 0 0 1 1 1] [0 0 1 1] [0 0 0 1 0 1] [0 0 0 0 1 1 1 0] [0 0 0 0 0 1 1 1] [0 0 0 0 0 0 0 1 1];
[0 0 0 1 0 0 1 1] [0 0 0 1 0 0 0 1] [0 0 0 0 1 1 1 1] [0 0 0 0 0 1 1 0 1] [0 0 0 0 0 1 0 1 0] [0 0 0 0 0 0 0 1 0 0];
[0 0 0 0 1 1 0 1] [0 0 0 0 1 0 1] [0 0 0 0 1 0 0 0] [0 0 0 0 0 1 0 1 1] [0 0 0 0 0 0 0 1 0 1] [0 0 0 0 0 0 0 0 0 1];
[0 0 0 0 0 1 1 0 0] [0 0 0 0 0 1 0 0] [0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 1] [0 0 0 0 0 0 0 0 0 0 1] [0 0 0 0 0 0 0 0 0 0 0]};
table9 = {[1 1 1] [1 0 1] [0 1 0 0 1] [0 0 1 1 1 0] [0 0 0 0 1 1 1 1] [0 0 0 0 0 0 1 1 1];
[1 1 0] [1 0 0] [0 1 0 1] [0 0 1 0 1] [0 0 0 1 1 0] [0 0 0 0 0 1 1 1];
[0 1 1 1] [0 1 1 0] [0 1 0 0 0] [0 0 1 0 0 0] [0 0 0 1 0 0 0] [0 0 0 0 0 1 0 1];
[0 0 1 1 1 1] [0 0 1 1 0] [0 0 1 0 0 1] [0 0 0 1 0 1 0] [0 0 0 0 1 0 1 0] [0 0 0 0 0 0 1 0 1];
[0 0 0 1 0 1 1] [0 0 0 1 1 1] [0 0 0 1 0 0 1] [0 0 0 0 1 1 0] [0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 1];
[0 0 0 0 1 1 1 0] [0 0 0 0 1 0 0] [0 0 0 0 0 1 1 0] [0 0 0 0 0 0 1 0] [0 0 0 0 0 0 1 1 0] [0 0 0 0 0 0 0 0 0];
table10 = {[1] [0 1 0] [0 0 1 0 1 0] [0 0 0 1 0 1 1 1] [0 0 0 1 0 0 0 1 1] [0 0 0 0 1 1 1 1 0] [0 0 0 0 0 1 1 0 0] [0 0 0 0 1 0 0 0 1];
[0 1 1] [0 0 1 1] [0 0 1 0 0 0] [0 0 0 1 1 0 0] [0 0 0 1 0 0 1 0] [0 0 0 0 1 0 1 0 1] [0 0 0 0 1 1 0 0] [0 0 0 0 0 1 1 1];
[0 0 1 0 1 1] [0 0 1 0 0 1] [0 0 0 1 1 1 1] [0 0 0 1 0 1 0 1] [0 0 0 1 0 0 0 0 0] [0 0 0 0 1 0 1 0 0 0] [0 0 0 0 1 0 0 1 1] [0 0 0 0 0 0 1 1 0];
[0 0 0 1 1 1 0] [0 0 0 1 1 0 1] [0 0 0 1 0 1 1 0] [0 0 0 1 0 0 0 1 0] [0 0 0 0 1 0 1 1 1 0] [0 0 0 0 0 1 0 1 1 1] [0 0 0 0 1 0 0 1 0] [0 0 0 0 0 0 0 1 1 1];
[0 0 0 1 0 1 0 0] [0 0 0 1 0 0 1 1] [0 0 0 1 0 0 0 0 1] [0 0 0 0 1 0 0 0 0 0] [0 0 0 0 0 1 1 0 1 1] [0 0 0 0 0 1 0 1 1 0] [0 0 0 0 0 0 1 0 0 1] [0 0 0 0 0 0 0 0 1
1];
[0 0 0 0 1 1 1 1 1] [0 0 0 0 1 0 1 1 0] [0 0 0 0 1 0 1 0 0 1] [0 0 0 0 0 1 1 0 1 0] [0 0 0 0 0 0 1 0 1 0 1] [0 0 0 0 0 0 1 0 1 0 0] [0 0 0 0 0 0 0 0 1 0 1] [0 0 0 0
0 0 0 0 0 1 1];
[0 0 0 0 1 1 1 0] [0 0 0 0 1 1 0 1] [0 0 0 0 0 1 0 1 0] [0 0 0 0 0 0 1 0 1 1] [0 0 0 0 0 1 0 0 0 0] [0 0 0 0 0 0 0 1 1 0] [0 0 0 0 0 0 0 0 1 0 1] [0 0 0 0 0 0 0
0 0 1];
[0 0 0 0 0 1 0 0 1] [0 0 0 0 1 0 0 0] [0 0 0 0 0 0 1 1 1] [0 0 0 0 0 0 1 0 0 0] [0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 1 0] [0 0 0 0 0 0
0 0 0 0 0];
table11 = {[1 1] [1 0 0] [0 1 0 1 0] [0 0 1 1 0 0] [0 0 1 0 0 0 1 0] [0 0 0 1 0 0 0 0 1] [0 0 0 1 0 1 0 1] [0 0 0 0 1 1 1 1];
[1 0 1] [0 1 1] [0 1 0 0] [0 0 1 0 1 0] [0 0 1 0 0 0 0 0] [0 0 0 1 0 0 0 1] [0 0 0 1 0 1 1] [0 0 0 0 1 0 1 0];
[0 1 0 1 1] [0 0 1 1 1] [0 0 1 1 0 1] [0 0 1 0 0 1 0] [0 0 0 1 1 1 1 0] [0 0 0 0 1 1 1 1 1] [0 0 0 1 0 1 0 0] [0 0 0 0 0 1 0 1];
[0 0 1 1 0 0 1] [0 0 1 0 1 1] [0 0 1 0 0 1 1] [0 0 0 1 1 1 0 1 1] [0 0 0 1 1 0 1 1] [0 0 0 0 0 1 0 0 1 0] [0 0 0 0 1 1 0 0] [0 0 0 0 0 1 0 1];
[0 0 1 0 0 0 1 1] [0 0 1 0 0 0 0 1] [0 0 0 1 1 1 1 1] [0 0 0 1 1 1 0 1 0] [0 0 0 0 1 1 1 1 0] [0 0 0 0 0 1 0 0 0 0] [0 0 0 0 0 0 1 1 1] [0 0 0 0 0 0 0 1 0 1];
[0 0 0 1 1 1 0 0] [0 0 0 1 1 0 1 0] [0 0 0 1 0 0 0 0 0] [0 0 0 0 0 1 0 0 1 1] [0 0 0 0 0 1 0 0 0 1] [0 0 0 0 0 0 0 1 1 1 1] [0 0 0 0 0 0 1 0 0 0] [0 0 0 0 0 0 0 1
1 1 0];
[0 0 0 0 1 1 1 0] [0 0 0 1 1 0 0] [0 0 0 1 0 0 1] [0 0 0 0 1 1 0 1] [0 0 0 0 0 1 1 1 0] [0 0 0 0 0 0 1 0 0 1] [0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 1];
[0 0 0 0 1 0 1 1] [0 0 0 0 1 0 0] [0 0 0 0 0 1 1 0] [0 0 0 0 0 0 1 1 0] [0 0 0 0 0 0 0 1 1 0] [0 0 0 0 0 0 0 1 1 1] [0 0 0 0 0 0 0 1 0 1] [0 0 0 0 0 0 0 0 0 0 0]};
table12 = {[1 0 0 1] [1 1 0] [1 0 0 0 0] [0 1 0 0 0 0 1] [0 0 1 0 1 0 0 1] [0 0 0 1 0 0 1 1 1] [0 0 0 1 0 0 1 1 0] [0 0 0 0 1 1 0 1 0];
[1 1 1] [1 0 1] [0 1 1 0] [0 1 0 0 1] [0 0 1 0 1 1 1] [0 0 1 0 0 0 0] [0 0 0 1 1 0 1 0] [0 0 0 0 1 0 1 1];
[1 0 0 0 1] [0 1 1 1] [0 1 0 1 1] [0 0 1 1 1 0] [0 0 1 0 1 0 1] [0 0 0 1 1 1 1 0] [0 0 0 1 0 1 0] [0 0 0 0 0 1 1 1];
[0 1 0 0 0 1] [0 1 0 1 0] [0 0 1 1 1 1] [0 0 1 1 0 0] [0 0 1 0 0 1 0] [0 0 0 1 1 1 0 0] [0 0 0 0 1 1 1 0] [0 0 0 0 0 1 0 1];
[0 1 0 0 0 0 0] [0 0 1 1 0 1] [0 0 1 0 1 1 0] [0 0 1 0 0 1 1] [0 0 0 1 0 0 1 0] [0 0 0 1 0 0 0 0] [0 0 0 0 1 0 0 1] [0 0 0 0 0 0 1 0 1];
[0 0 1 0 1 0 0 0] [0 0 1 0 0 0 1] [0 0 0 1 1 1 1 1] [0 0 0 1 1 1 0 1] [0 0 0 1 0 0 0 1] [0 0 0 0 0 1 1 0 1] [0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 1 0];
[0 0 0 1 1 0 1 1] [0 0 0 1 1 0 0] [0 0 0 1 0 1 1] [0 0 0 0 1 1 1 1] [0 0 0 0 1 0 1 0] [0 0 0 0 0 0 1 1 1] [0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 1];
[0 0 0 0 1 1 0 1 1] [0 0 0 0 1 1 0 0] [0 0 0 0 1 0 0 0] [0 0 0 0 0 1 1 0 0] [0 0 0 0 0 0 1 1 0] [0 0 0 0 0 0 0 1 1] [0 0 0 0 0 0 0 0 1] [0 0 0 0 0 0 0 0 0];

table13 = {[1] [0 1 0 1] [0 0 1 1 1 0] [0 0 1 0 1 0 1] [0 0 1 0 0 0 1 0] [0 0 0 1 1 0 0 1 1] [0 0 0 1 0 1 1 1 0] [0 0 0 1 0 0 0 1 1 1] [0 0 0 1 0 1 0 1 0] [0 0 0 0 1 1 0 1 0 0] [0 0 0 0 1 0 0 0 1 0 0] [0 0 0 0 0 1 1 0 1 0 0] [0 0 0 0 0 1 0 0 0 0 1 1] [0 0 0 0 0 0 1 0 1 1 0 0] [0 0 0 0 0 0 0 1 0 1 0 1 1] [0 0 0 0 0 0 0 0 1 0 0 1 1];

[0 1 1] [0 1 0 0] [0 0 1 1 0 0] [0 0 1 0 0 1 1] [0 0 0 1 1 1 1 1] [0 0 0 1 1 0 1 0] [0 0 0 1 0 1 1 0 0] [0 0 0 1 0 0 0 0 1] [0 0 0 0 1 1 1 1 1] [0 0 0 0 1 1 0 0 0] [0 0 0 0 1 0 0 0 0 0] [0 0 0 0 0 1 1 0 0 0] [0 0 0 0 0 0 1 1 1 1 1] [0 0 0 0 0 0 1 0 0 0 1 1] [0 0 0 0 0 0 0 1 0 1 1 0] [0 0 0 0 0 0 0 0 1 1 1 0];

[0 0 1 1 1 1] [0 0 1 1 0 1] [0 0 1 0 1 1 1] [0 0 1 0 0 1 0 0] [0 0 0 1 1 1 0 1 1] [0 0 0 1 1 0 0 0 1] [0 0 0 1 0 0 1 1 0 1] [0 0 0 1 0 0 0 0 0 1] [0 0 0 0 1 1 1 0 1] [0 0 0 0 1 0 1 0 0 0] [0 0 0 0 0 1 1 1 1 0] [0 0 0 0 0 1 0 1 0 0 0] [0 0 0 0 0 0 1 1 0 1 1] [0 0 0 0 0 0 1 0 0 0 0 1] [0 0 0 0 0 0 0 1 0 1 0 1 0] [0 0 0 0 0 0 0 0 1 0 0 0 0];

[0 0 1 0 1 1 0] [0 0 1 0 1 0 0] [0 0 1 0 0 1 0 1] [0 0 0 1 1 1 1 0 1] [0 0 0 1 1 1 0 0 0] [0 0 0 1 0 0 1 1 1 1] [0 0 0 1 0 0 1 0 0 1] [0 0 0 1 0 0 0 0 0 0] [0 0 0 0 1 0 1 0 1 1] [0 0 0 0 1 0 0 1 1 0 0] [0 0 0 0 0 1 1 1 0 0 0] [0 0 0 0 0 1 0 0 1 0 1] [0 0 0 0 0 0 1 1 0 1 0] [0 0 0 0 0 0 0 1 1 1 1 1] [0 0 0 0 0 0 0 0 1 1 0 0 1] [0 0 0 0 0 0 0 0 1 1 1 0];

[0 0 1 0 0 0 1 1] [0 0 1 0 0 0 0] [0 0 0 1 1 1 1 0 0] [0 0 0 1 1 1 0 0 1] [0 0 0 1 1 0 0 0 0 1] [0 0 0 1 0 0 1 0 1 1] [0 0 0 0 1 1 1 0 0 1 0] [0 0 0 0 1 0 1 1 0 1 1] [0 0 0 0 1 1 0 1 1 0] [0 0 0 0 1 0 0 1 0 0 1] [0 0 0 0 0 1 1 0 1 1 1] [0 0 0 0 0 0 1 0 1 0 0 1] [0 0 0 0 0 0 0 1 1 0 0 0 0] [0 0 0 0 0 0 0 1 1 0 1 0 1] [0 0 0 0 0 0 0 1 0 1 1 1] [0 0 0 0 0 0 0 0 0 1 1 0 0 0];

[0 0 0 1 1 1 0 1 0] [0 0 0 1 1 0 1 1] [0 0 0 1 1 0 0 1 0] [0 0 0 1 1 0 0 0 0 0] [0 0 0 1 0 0 1 1 0 0] [0 0 0 1 0 0 0 1 1 0] [0 0 0 0 1 0 1 1 1 0 1] [0 0 0 0 1 0 1 0 1 0 0] [0 0 0 0 1 0 0 1 1 0 1] [0 0 0 0 0 1 1 1 0 1 0] [0 0 0 0 0 1 0 0 1 1 1 1] [0 0 0 0 0 0 1 1 1 0 1] [0 0 0 0 0 0 1 0 0 1 0 1 0] [0 0 0 0 0 0 0 1 1 0 0 0 1] [0 0 0 0 0 0 0 1 0 1 0 0 1] [0 0 0 0 0 0 0 0 0 0 1 0 0 0 1];

[0 0 0 1 0 1 1 1 1] [0 0 0 1 0 1 1 0 1] [0 0 0 1 0 0 1 1 1 0] [0 0 0 1 0 0 1 0 1 0] [0 0 0 0 1 1 1 0 0 1 1] [0 0 0 0 1 0 1 1 1 1 0] [0 0 0 0 1 0 1 1 0 1 0] [0 0 0 0 1 0 0 1 1 1 1] [0 0 0 0 1 0 0 0 1 0 1] [0 0 0 0 0 1 0 1 0 0 1 1] [0 0 0 0 0 1 0 0 0 1 1 1] [0 0 0 0 0 0 1 1 0 0 1 0] [0 0 0 0 0 0 0 1 1 1 0 1 1] [0 0 0 0 0 0 0 1 0 0 1 1 0] [0 0 0 0 0 0 0 0 0 1 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 1 1 1 1];

[0 0 0 1 0 0 1 0 0 0] [0 0 0 1 0 0 0 1 0] [0 0 0 0 1 1 1 0 0 0] [0 0 0 0 1 0 1 1 1 1 1] [0 0 0 0 1 0 1 1 1 0 0] [0 0 0 0 1 0 1 0 1 0 1] [0 0 0 0 0 1 0 1 1 0 1 1] [0 0 0 0 0 1 0 1 1 0 1 0] [0 0 0 0 0 1 0 1 0 1 1 0] [0 0 0 0 0 1 0 0 1 0 0 1] [0 0 0 0 0 0 1 0 0 1 1 0 1] [0 0 0 0 0 0 1 0 0 0 0 0 1] [0 0 0 0 0 0 0 1 1 0 0 1 1] [0 0 0 0 0 0 0 0 1 0 1 1 0 0] [0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 1] [0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0];

[0 0 0 1 0 1 0 1 1] [0 0 0 1 0 1 0 0] [0 0 0 0 1 1 1 1 0] [0 0 0 0 1 0 1 1 0 0] [0 0 0 0 1 1 0 1 1 1] [0 0 0 0 1 0 0 1 1 1 0] [0 0 0 0 1 0 0 1 0 0 0] [0 0 0 0 0 1 0 1 0 1 1 1] [0 0 0 0 0 1 0 0 1 1 1 0] [0 0 0 0 0 0 1 1 1 1 0 1] [0 0 0 0 0 0 1 0 1 1 1 0] [0 0 0 0 0 0 0 1 1 0 1 1 0] [0 0 0 0 0 0 0 1 0 0 1 0 1] [0 0 0 0 0 0 0 0 0 1 1 1 1 0] [0 0 0 0 0 0 0 0 0 0 1 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0];

[0 0 0 0 1 1 0 1 0 1] [0 0 0 0 1 1 0 0 1] [0 0 0 0 1 0 1 0 0 1] [0 0 0 0 1 0 0 1 0 1] [0 0 0 0 0 1 0 1 1 0 0] [0 0 0 0 0 1 1 1 0 1 1] [0 0 0 0 0 1 1 0 1 1 0] [0 0 0 0 0 0 1 0 1 0 0 0 1] [0 0 0 0 0 1 0 0 0 0 1 0] [0 0 0 0 0 0 1 0 0 1 1 0 0] [0 0 0 0 0 0 0 1 1 1 0 0 1] [0 0 0 0 0 0 0 0 1 1 0 1 1 0] [0 0 0 0 0 0 0 0 1 0 0 1 0 1] [0 0 0 0 0 0 0 0 0 1 0 0 1 0] [0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 1] [0 0 0 0 0 0 0 0 0 0 0 1 0 1 1];

[0 0 0 0 1 0 0 0 1 1] [0 0 0 0 1 0 0 0 0 1] [0 0 0 0 0 1 1 1 1 1] [0 0 0 0 0 1 1 1 0 0 1] [0 0 0 0 0 1 0 1 0 1 0] [0 0 0 0 0 1 0 1 0 0 1 0] [0 0 0 0 0 1 0 0 1 0 0 0] [0 0 0 0 0 0 1 0 1 0 0 0 0] [0 0 0 0 0 0 1 0 1 1 1 1] [0 0 0 0 0 0 0 1 1 1 0 1 0] [0 0 0 0 0 0 0 0 1 1 0 1 1 1] [0 0 0 0 0 0 0 0 0 1 0 1 0 1] [0 0 0 0 0 0 0 0 0 1 0 1 1 0] [0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0] [0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0];

[0 0 0 0 0 1 1 0 1 0 1] [0 0 0 0 0 1 1 0 0 1] [0 0 0 0 0 1 0 1 1 1] [0 0 0 0 0 1 0 0 1 1 0] [0 0 0 0 0 1 0 0 0 1 1 0] [0 0 0 0 0 0 1 1 1 1 0 0] [0 0 0 0 0 0 1 1 0 0 1 1] [0 0 0 0 0 0 1 0 0 1 0 0] [0 0 0 0 0 0 0 1 1 0 1 1 1] [0 0 0 0 0 0 0 0 0 1 1 0 1 0] [0 0 0 0 0 0 0 1 0 0 0 1 0] [0 0 0 0 0 0 0 0 0 1 0 1 1 1] [0 0 0 0 0 0 0 0 0 0 1 1 0 1 1] [0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1];

[0 0 0 0 0 1 0 0 0 1 0] [0 0 0 0 0 1 0 0 0 0 0] [0 0 0 0 0 0 1 1 1 0 0] [0 0 0 0 0 0 1 0 0 1 1 1] [0 0 0 0 0 0 1 1 0 0 0 1] [0 0 0 0 0 0 1 0 0 1 0 1 1] [0 0 0 0 0 0 1 1 1 1 0] [0 0 0 0 0 0 0 1 1 0 1 0 0] [0 0 0 0 0 0 0 0 1 1 0 0 0 0] [0 0 0 0 0 0 0 0 1 0 1 0 0 0] [0 0 0 0 0 0 0 0 0 1 1 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 1 1 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1];

[0 0 0 0 0 0 1 0 1 1 0 1] [0 0 0 0 0 0 1 0 1 0 1] [0 0 0 0 0 0 1 0 0 0 1 0] [0 0 0 0 0 0 1 0 0 0 0 0 0] [0 0 0 0 0 0 0 0 1 1 1 0 0 0] [0 0 0 0 0 0 0 0 1 1 0 0 1 0] [0 0 0 0 0 0 0 0 1 1 0 0 0 1] [0 0 0 0 0 0 0 0 0 1 0 1 1 0 1] [0 0 0 0 0 0 0 0 0 0 1 1 1 1 1] [0 0 0 0 0 0 0 0 0 0 1 0 0 1 1] [0 0 0 0 0 0 0 0 0 0 0 1 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 1 1 1 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1];

[0 0 0 0 0 0 0 1 1 0 0 0 0] [0 0 0 0 0 0 0 0 1 0 1 1 1] [0 0 0 0 0 0 0 1 0 1 0 0] [0 0 0 0 0 0 0 1 0 0 1 1 1] [0 0 0 0 0 0 0 1 0 0 1 0 0] [0 0 0 0 0 0 0 1 0 0 0 1 1] [0 0 0 0 0 0 0 0 0 1 1 0 1 0 1] [0 0 0 0 0 0 0 0 0 1 0 1 0 1] [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0];

[0 0 0 0 0 0 0 0 1 0 0 0 0] [0 0 0 0 0 0 0 0 0 1 1 1 1] [0 0 0 0 0 0 0 0 0 1 0 0 0 1] [0 0 0 0 0 0 0 0 0 0 1 1 0 1 1] [0 0 0 0 0 0 0 0 0 0 1 1 0 0 1] [0 0 0 0 0 0 0 0 0 0 1 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1] [0 0 0 0 0 0 0 0 0 0 0 1 0 1 1] [0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

0] [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]};

table15 = {[1 1 1] [1 1 0 0] [1 0 0 1 0] [0 1 1 0 1 0 1] [0 1 0 1 1 1 1] [0 1 0 0 1 1 0 0] [0 0 1 1 1 1 1 1 0 0] [0 0 1 1 0 1 1 0 0] [0 0 1 0 1 1 0 0 1] [0 0 0 1 1 1 0 1 1 1] [0 0 0 1 1 0 1 1 0 0] [0 0 0 0 1 1 1 0 1 1 1] [0 0 0 0 1 1 0 1 0 1 1] [0 0 0 0 1 0 1 0 0 0 1] [0 0 0 0 0 1 1 1 1 0 1 0] [0 0 0 0 0 0 0 1 1 1 1 1 1];

[1 1 0 1] [1 0 1] [1 0 0 0 0] [0 1 1 0 1 1] [0 1 0 1 1 1 0] [0 1 0 0 1 0 0] [0 0 1 1 1 1 0 1] [0 0 1 1 0 0 1 1] [0 0 1 0 1 0 1 0] [0 0 1 0 0 0 1 1 0] [0 0 0 1 1 0 1 0 0] [0 0 0 1 0 1 0 0 1 1] [0 0 0 1 0 0 0 0 0 1] [0 0 0 0 1 0 1 0 0 1] [0 0 0 0 0 1 1 1 0 1 1] [0 0 0 0 0 1 0 0 1 0 0];

[1 0 0 1 1] [1 0 0 0 1] [0 1 1 1 1] [0 1 1 0 0 0] [0 1 0 1 0 0 1] [0 1 0 0 0 1 0] [0 0 1 1 1 0 1 1] [0 0 1 1 0 0 0 0] [0 0 1 0 1 0 0 0] [0 0 1 0 0 0 0 0 0] [0 0 0 1 1 0 0 1 1 1 0] [0 0 0 1 0 0 1 1 1 0] [0 0 0 0 1 1 1 1 1 0] [0 0 0 0 1 0 1 0 0 0 0] [0 0 0 0 0 1 1 0 0 0] [0 0 0 0 0 1 1 1 0 0 0] [0 0 0 0 1 0 0 0 0 1];[0 1 1 1 0 1] [0 1 1 1 0 0] [0 1 1 0 0 1] [0 1 0 1 0 1 1] [0 1 0 0 1 1 1] [0 0 1 1 1 1 1 1] [0 0 1 1 0 1 1 1] [0 0 1 0 1 1 1 0 1] [0 0 1 0 0 1 1 0 0] [0 0 0 1 1 1 0 1 1] [0 0 0 1 0 1 1 1 0 1] [0 0 0 1 0 0 1 0 0 0] [0 0 0 1 1 0 1 1 0] [0 0 0 1 0 0 1 0 1 1] [0 0 0 0 0 1 1 0 0 1 0] [0 0 0 0 0 0 1 1 1 0 1];[0 1 1 0 1 0 0] [0 1 0 1 1 0] [0 1 0 1 0 1 0] [0 1 0 1 0 0 0] [0 1 0 0 0 0 1 1] [0 0 1 1 1 0 0 1] [0 0 1 0 1 1 1 1 1] [0 0 1 0 0 1 1 1 1 1] [0 0 1 0 0 1 0 0 0] [0 0 0 1 1 1 0 0 1] [0 0 0 1 0 1 1 0 0 1] [0 0 0 1 0 0 0 1 0 1] [0 0 0 0 1 1 0 0 0 1] [0 0 0 0 1 0 0 0 1 0] [0 0 0 0 0 1 0 1 1 1 0];[0 0 1 1 1 1 1 0 1] [0 1 0 0 0 0 0] [0 0 1 1 1 0 0] [0 0 1 1 1 0 0 0] [0 0 1 1 0 0 1 0] [0 0 1 0 1 1 0 1 0] [0 0 1 0 0 1 1 1 0] [0 0 1 0 0 0 1 1 0] [0 0 0 1 1 1 0 1 0] [0 0 0 1 0 1 1 0 1] [0 0 0 1 0 0 1 0 0] [0 0 0 1 0 0 0 0 1] [0 0 0 0 1 1 1 1 1 0] [0 0 0 0 1 0 1 0 1 0] [0 0 0 0 1 0 0 0 0 0 0] [0 0 0 0 1 0 1 1 0 0] [0 0 0 0 1 0 1 0 0 1] [0 0 0 0 0 1 1 0 1 1 0] [0 0 0 0 0 1 0 0 1 1 0] [0 0 0 0 0 0 1 1 1 0 1];[0 0 1 1 0 1 1 0 1] [0 0 1 1 0 1 0 1] [0 0 1 1 0 0 0 1] [0 0 1 0 1 1 1 1 0] [0 0 1 0 1 1 0 0 1] [0 0 1 0 0 1 1 0 1] [0 0 1 0 0 1 0 0 1] [0 0 0 0 1 1 1 0 0 0] [0 0 0 0 1 0 1 0 1 0] [0 0 0 0 1 0 0 0 0 0 0] [0 0 0 0 1 0 1 1 0 1] [0 0 0 0 1 0 0 1 0 0] [0 0 0 0 0 1 1 0 1 1 1] [0 0 0 0 0 1 0 0 1 0 0] [0 0 0 0 0 0 1 1 1 1 1];

[0 0 0 1 1 0 1 1 0 1] [0 0 0 1 1 0 1 0 1] [0 0 0 1 1 0 0 1 1] [0 0 0 1 0 1 1 1 1] [0 0 0 1 0 1 1 0 1 0] [0 0 0 1 0 1 0 0 1 0] [0 0 0 0 1 1 1 0 1 0] [0 0 0 0 1 1 1 0 0 1] [0 0 0 0 1 1 0 0 0 0] [0 0 0 0 1 0 0 1 0 0 0] [0 0 0 0 0 1 1 1 0 0 1] [0 0 0 0 0 1 0 1 0 0 1] [0 0 0 0 0 0 1 1 0 1 1] [0 0 0 0 0 0 0 1 1 1 1 1 0] [0 0 0 0 0 0 0 1 0 0 1];

[0 0 0 1 0 1 0 1 1 0] [0 0 0 1 0 1 0 1 0] [0 0 0 1 0 1 0 0 0] [0 0 0 1 0 0 1 0 1] [0 0 0 1 0 0 0 1 1 0] [0 0 0 1 0 0 0 0 0 0] [0 0 0 0 1 1 0 1 0 0] [0 0 0 0 1 0 1 0 1 1] [0 0 0 0 1 0 0 0 1 1 0] [0 0 0 0 0 1 1 0 1 1 1] [0 0 0 0 0 1 0 1 0 1 0] [0 0 0 0 0 0 1 1 0 0 1] [0 0 0 0 0 0 0 1 1 1 0 1] [0 0 0 0 0 0 0 1 0 0 1 0] [0 0 0 0 0 0 0 0 1 0 1 1];

[0 0 0 0 1 1 1 0 1 1 0] [0 0 0 1 0 0 0 1 0 0] [0 0 0 0 1 1 1 1 0] [0 0 0 0 1 1 0 1 1 1] [0 0 0 0 1 1 0 0 1 0] [0 0 0 0 1 0 1 1 1 0] [0 0 0 0 1 0 0 1 0 1 0] [0 0 0 0 1 0 0 0 0 0 1] [0 0 0 0 0 1 1 0 0 0 1] [0 0 0 0 0 1 0 0 1 1 1] [0 0 0 0 0 0 1 1 0 0 0] [0 0 0 0 0 0 1 0 0 0 0] [0 0 0 0 0 0 0 1 0 1 1 0] [0 0 0 0 0 0 0 0 1 1 0 1] [0 0 0 0 0 0 0 0 0 1 1 1 0] [0 0 0 0 0 0 0 0 0 0 1 1 1];

[0 0 0 0 1 0 1 1 0 1 1] [0 0 0 0 1 0 1 1 0 0] [0 0 0 0 1 0 0 1 1 1] [0 0 0 0 1 0 0 1 1 0] [0 0 0 0 1 0 0 0 1 0] [0 0 0 0 0 1 1 1 1 1 1] [0 0 0 0 0 1 1 0 1 0 0] [0 0 0 0 1 0 1 1 0 1] [0 0 0 0 0 0 1 1 1 1] [0 0 0 0 0 1 1 0 1 0 0] [0 0 0 0 0 0 1 1 1 0 0] [0 0 0 0 0 0 1 0 0 1 1] [0 0 0 0 0 0 0 1 1 1 0] [0 0 0 0 0 0 0 0 1 0 0 0] [0 0 0 0 0 0 0 0 0 0 1 0 0] [0 0 0 0 0 0 0 0 0 0 0 0 1 1];

[0 0 0 0 0 1 1 1 0 1 1] [0 0 0 0 0 1 1 1 1 0 0] [0 0 0 0 0 1 1 1 0 1 0] [0 0 0 0 0 1 1 0 1 0 1] [0 0 0 0 0 1 0 1 1 1 1] [0 0 0 0 0 1 0 1 0 1 1] [0 0 0 0 0 1 0 0 0 0 0] [0 0 0 0 0 0 1 0 1 1 0] [0 0 0 0 0 0 1 0 0 1 1 1] [0 0 0 0 0 0 1 0 0 0 1 1] [0 0 0 0 0 0 0 1 1 1 1 1] [0 0 0 0 0 0 0 1 1 0 1 0] [0 0 0 0 0 0 0 0 1 1 1 0 0] [0 0 0 0 0 0 0 0 1 0 0 1 1] [0 0 0 0 0 0 0 0 1 1 1 0] [0 0 0 0 0 0 0 0 1 0 1 0 0] [0 0 0 0 0 0 0 0 0 1 0 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 1 1];

[0 0 0 0 0 1 0 0 0 1 1 1] [0 0 0 0 0 1 0 0 1 0 1] [0 0 0 0 0 1 0 0 0 1 0] [0 0 0 0 0 0 1 1 1 1 0] [0 0 0 0 0 0 1 1 1 0 0] [0 0 0 0 0 0 1 0 1 0 0] [0 0 0 0 0 0 0 1 0 0 1 1] [0 0 0 0 0 0 0 0 1 1 0] [0 0 0 0 0 0 0 0 0 1 0 0 0] [0 0 0 0 0 0 0 0 0 1 1 1 0] [0 0 0 0 0 0 0 0 0 0 1 0 1 0] [0 0 0 0 0 0 0 0 0 0 0 0 0 1];

table16 = {[1] [0 1 0 1] [0 0 1 1 0] [0 0 1 0 1 1 0 0] [0 0 1 0 0 1 0 1 0] [0 0 0 1 1 1 1 1] [0 0 0 1 1 0 1 1 1 0] [0 0 0 1 0 1 1 1 0 1] [0 0 0 1 0 1 0 1 1 0 0] [0 0 0 1 0 0 1 0 1 0 1] [0 0 0 1 0 0 0 1 0 1 0] [0 0 0 0 1 1 1 0 0 1 0] [0 0 0 0 1 1 1 0 0 0 0 1] [0 0 0 0 1 1 0 0 0 0 1 1] [0 0 0 0 1 0 1 1 1 1 0 0 0] [0 0 0 0 1 0 0 0 1];

[0 1 1] [0 1 0 0] [0 0 1 1 0 0] [0 0 1 0 1 0 0] [0 0 1 0 0 0 1 1] [0 0 0 1 1 1 1 1 0] [0 0 0 1 1 0 1 0 1] [0 0 0 1 0 1 1 1 1] [0 0 0 1 0 1 0 0 1 1] [0 0 0 1 0 0 0 1 0 1 1] [0 0 0 1 0 0 0 1 0 0] [0 0 0 0 1 1 1 0 1 1 1] [0 0 0 0 1 1 0 0 1 0 0 1] [0 0 0 0 1 1 0 1 0 1 1] [0 0 0 0 1 1 0 0 1 1 1 1] [0 0 0 0 1 0 0 1];

[0 0 1 1 1 1] [0 0 1 1 0 1] [0 0 1 0 1 1 1] [0 0 1 0 0 1 1 0] [0 0 1 0 0 0 0 1 1] [0 0 0 1 1 1 0 1 0] [0 0 0 1 1 0 0 1 1 1] [0 0 0 1 0 1 1 0 1 0] [0 0 0 1 0 1 0 0 0 0 1] [0 0 0 1 0 0 1 0 0 0] [0 0 0 0 1 1 1 1 1 1 1] [0 0 0 0 1 1 1 0 1 0 1] [0 0 0 0 1 1 0 1 1 1 0] [0 0 0 0 1 1 0 1 0 0 0 0 1] [0 0 0 0 1 1 0 0 1 1 1 0] [0 0 0 0 1 0 0 0 0];

[0 0 1 0 1 1 0 1] [0 0 1 0 1 0 1] [0 0 1 0 0 1 1 1] [0 0 1 0 0 0 1 0 1] [0 0 1 0 0 0 0 0 0] [0 0 0 1 1 1 0 0 1 0] [0 0 0 1 1 0 0 0 1 1] [0 0 0 1 0 1 0 1 1 1] [0 0 0 1 0 0 1 1 1 1 0] [0 0 0 1 0 0 0 1 1 0 0] [0 0 0 0 1 1 1 1 1 1 0 0] [0 0 0 0 1 1 0 1 0 1 0 0] [0 0 0 0 1 1 0 0 0 1 1 1] [0 0 0 0 1 1 0 0 0 0 0 1 1] [0 0 0 0 1 0 1 1 0 1 1 0 1] [0 0 0 0 0 1 1 0 1 0];

[0 0 1 0 0 1 0 1 1] [0 0 1 0 0 1 0 0] [0 0 1 0 0 0 0 1 0 0] [0 0 1 0 0 0 0 0 0 1] [0 0 0 1 1 1 0 0 1 1] [0 0 0 1 1 0 0 1 0 1] [0 0 0 1 0 1 1 0 0 1 1] [0 0 0 1 0 1 0 0 1 0 0] [0 0 0 1 0 0 1 1 0 1 1] [0 0 0 1 0 0 0 0 1 0 0 0] [0 0 0 0 1 1 1 1 0 1 1 0] [0 0 0 0 1 1 1 0 0 0 1 0] [0 0 0 0 1 1 0 0 0 1 0 1 1] [0 0 0 0 1 0 1 1 1 1 1 1 0] [0 0 0 0 1 0 1 1 0 1 0 1 0] [0 0 0 0 0 1 0 0 1];

[0 0 1 0 0 0 0 1 0] [0 0 0 1 1 1 1 1 0] [0 0 0 1 1 1 0 1 1] [0 0 0 1 1 1 0 0 0] [0 0 0 1 1 0 0 1 1 0] [0 0 0 1 0 1 1 1 0 0 1] [0 0 0 1 0 1 0 1 1 0 1] [0 0 0 1 0 0 0 0 1 0 0 1] [0 0 0 1 0 0 0 1 1 1 0] [0 0 0 0 1 1 1 1 1 0 1] [0 0 0 0 1 1 1 0 1 0 0 0] [0 0 0 0 1 1 0 0 1 0 0 0 0] [0 0 0 0 1 1 0 0 0 0 1 0 0] [0 0 0 0 1 0 1 1 1 1 0 1 0] [0 0 0 0 0 1 1 0 1 1 1 1 1 0 1] [0 0 0 0 0 1 0 0 0 0];

[0 0 0 1 1 0 1 1 1 1] [0 0 0 1 1 0 1 1 0] [0 0 0 1 1 0 1 0 0] [0 0 0 1 1 0 0 1 0 0] [0 0 0 1 0 1 1 1 0 0 0] [0 0 0 1 0 1 1 0 0 1 0] [0 0 0 1 0 1 0 0 0 0 0] [0 0 0 1 0 0 0 0 0 1 0 1] [0 0 0 1 0 0 0 0 0 0 0 1] [0 0 0 0 1 1 1 1 0 1 0 0] [0 0 0 0 1 1 1 0 0 1 0 0] [0 0 0 0 1 1 0 1 1 0 0 1] [0 0 0 0 1 1 0 0 0 0 0 0 1] [0 0 0 0 1 0 1 1 0 1 1 1 0] [0 0 0 0 1 0 1 1 0 1 0 1 1] [0 0 0 0 0 0 1 0 1 0];

[0 0 0 1 1 0 0 0 1 0] [0 0 0 1 1 0 0 0 0] [0 0 0 1 0 1 1 0 1 1] [0 0 0 1 0 1 1 0 0 0] [0 0 0 1 0 1 0 0 1 0 1] [0 0 0 1 0 0 1 1 1 0 1] [0 0 0 1 0 0 1 0 1 0 0] [0 0 1 0 0 0 0 0 1 0 1] [0 0 0 0 1 1 1 1 0 0 0] [0 0 0 0 1 1 0 0 1 0 1 1] [0 0 0 0 1 1 0 0 0 1 1 0 1] [0 0 0 0 1 0 1 1 1 0 1 0 0] [0 0 0 0 1 0 1 1 1 1 1 0 0] [0 0 0 0 1 1 0 1 1 1 0 0 0 1] [0 0 0 0 1 1 0 1 1 0 1 0 0] [0 0 0 0 0 0 1 0 1 1];

[0 0 0 1 0 1 0 1 0 1] [0 0 0 1 0 1 0 1 0 0] [0 0 0 1 0 1 0 0 0 1] [0 0 0 1 0 0 1 1 1 1 1] [0 0 0 1 0 0 1 1 1 0 0] [0 0 0 1 0 0 0 1 1 1 1] [0 0 0 1 0 0 0 0 0 1 0 0] [0 0 0 0 1 1 1 1 1 0 0 1] [0 0 0 0 1 1 0 1 0 1 0 1 1] [0 0 0 0 1 1 0 0 1 0 0 0 1] [0 0 0 0 1 1 0 0 0 1 0 0 0] [0 0 0 0 1 0 1 1 1 1 1 1 1] [0 0 0 0 1 0 1 1 0 1 0 1 1 1] [0 0 0 0 1 0 1 1 0 0 1 0 0 1] [0 0 0 0 1 0 1 1 0 0 0 1 0 0] [0 0 0 0 0 0 0 1 1 1];

[0 0 0 1 0 0 1 1 0 1 0] [0 0 0 1 0 0 1 1 0 0] [0 0 0 1 0 0 1 0 0 1] [0 0 0 1 0 0 0 1 1 0 1] [0 0 0 1 0 0 0 0 0 1 1] [0 0 0 1 0 0 0 0 0 0 0 0] [0 0 0 0 1 1 1 1 0 1 0 1 0 1] [0 0 0 0 1 1 0 1 0 1 0 1 0] [0 0 0 0 1 1 0 0 1 0 1 1 0] [0 0 0 0 1 1 0 0 0 1 0 1 0] [0 0 0 0 1 1 0 0 0 0 0 0 0] [0 0 0 0 1 0 1 1 0 1 1 1 1 1] [0 0 0 0 1 0 1 1 0 0 1 1 1 1] [0 0 0 0 1 0 1 1 0 0 0 1 1] [0 0 0 0 1 0 1 1 0 0 0 0 0] [0 0 0 0 0 0 0 1 0 1 1];

[0 0 0 1 0 0 0 1 0 1 1] [0 0 0 1 0 0 0 0 0 0 1] [0 0 0 1 0 0 0 0 1 1] [0 0 0 0 1 1 1 1 1 0 1] [0 0 0 0 1 1 1 0 1 1 1] [0 0 0 0 1 1 1 0 1 0 0 1] [0 0 0 0 1 1 0 1 1 1 1 0] [0 0 0 0 1 1 0 1 0 1 0 1] [0 0 0 0 1 1 0 0 1 1 0 1 0] [0 0 0 0 1 1 0 0 0 1 0 1 0] [0 0 0 0 1 1 0 0 0 0 0 0 0] [0 0 0 0 1 0 1 1 0 1 1 1 1 1] [0 0 0 0 1 0 1 1 0 0 1 1 1 1] [0 0 0 0 1 0 1 1 0 0 0 1 1] [0 0 0 0 1 0 1 1 0 0 0 0 0] [0 0 0 0 0 0 0 1 0 1 1];

[0 0 0 0 1 1 1 1 0 0 1 1] [0 0 0 0 1 1 1 1 0 0 0] [0 0 0 0 1 1 1 0 1 1 0] [0 0 0 0 1 1 0 0 1 1] [0 0 0 0 1 1 1 0 0 0 1] [0 0 0 0 1 1 0 1 1 1 1 1] [0 0 0 0 1 0 0 0 1 1 0 0] [0 0 0 0 1 0 1 1 1 0 1 0 1 0] [0 0 0 0 1 0 1 1 1 0 0 1 1 0] [0 0 0 0 1 0 1 1 1 0 0 0 0 0] [0 0 0 0 1 0 1 1 0 1 0 0 0 1] [0 0 0 0 1 0 1 1 0 0 1 0 0 0 0 1 0] [0 0 0 0 1 0 1 1 0 0 0 0 1 0] [0 0 0 0 1 1 0 1 1 1 1 1] [0 0 0 0 1 1 0 1 1 0 1 0 0] [0 0 0 0 0 0 0 0 1 1 0];

[0 0 0 0 1 1 0 0 1 0 1 0] [0 0 0 0 1 1 0 0 0 0 0] [0 0 0 0 1 1 0 1 1 1 1 0] [0 0 0 0 1 1 0 1 1 0 1 0] [0 0 0 0 1 1 0 1 1 0 0 0] [0 0 0 0 1 1 0 0 0 0 1 0 1] [0 0 0 0 1 1 0 0 0 0 0 1 1 0] [0 0 0 0 1 1 0 1 1 1 1 1 1] [0 0 0 0 1 1 0 0 0 0 1 1 0 0] [0 0 0 0 1 0 1 1 1 0 1 0 1 0] [0 0 0 0 1 0 1 1 1 0 0 1 1 0] [0 0 0 0 1 0 1 1 1 0 0 0 0 0] [0 0 0 0 1 0 1 1 0 1 0 0 0 1] [0 0 0 0 1 0 1 1 0 0 1 0 0 0 0] [0 0 0 0 1 0 1 1 0 0 0 0 1 0] [0 0 0 0 1 1 0 1 1 1 1 1] [0 0 0 0 1 1 0 1 1 0 1 0 0] [0 0 0 0 0 0 0 0 1 0 0];

[0 0 0 0 1 0 1 1 1 0 1 0 1] [0 0 0 0 1 1 0 1 0 0 1] [0 0 0 0 1 1 0 1 0 0 1 0] [0 0 0 0 1 1 0 1 0 0 0 0] [0 0 0 0 1 0 1 1 1 0 0 1 0] [0 0 0 0 1 0 1 1 1 1 0 1 1 1] [0 0 0 0 1 0 1 1 0 1 1 1 0] [0 0 0 0 1 0 1 1 0 0 1 0 1 0] [0 0 0 0 1 1 0 1 1 0 0 0 1 1] [0 0 0 0 1 1 0 1 1 0 0 1 1] [0 0 0 0 1 1 0 1 1 0 1 1 0 1] [0 0 0 0 0 1 1 0 1 1 0 1 1 0 0] [0 0 0 0 1 1 0 1 1 0 0 0 0 0 1 1] [0 0 0 0 1 1 0 1 1 0 0 0 0 1] [0 0 0 0 0 0 0 0 1 0];

[0 0 0 0 1 0 1 1 1 1 0 0 1] [0 0 0 0 1 0 1 1 1 0 0 0 1] [0 0 0 0 1 1 0 0 1 1 0] [0 0 0 0 1 0 1 1 1 0 1 1] [0 0 0 0 1 0 1 1 0 1 0 1 1 0] [0 0 0 0 1 0 1 1 0 1 0 0 1 0] [0 0 0 0 1 0 1 1 0 0 1 1 0] [0 0 0 0 1 0 1 1 1 0 0 0 1 1] [0 0 0 0 1 1 0 1 1 0 0 0 1 0 1] [0 0 0 0 1 1 0 1 1 0 0 0 1 0] [0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0] [0 0 0 0 1 1 0 1 1 0 0 1 1 1] [0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 1 0] [0 0 0 0 0 1 1 0 1 1 0 0 1 1 0] [0 0 0 0 1 1 0 1 1 0 0 1 0] [0 0 0 0 0 0 0 0 0 0 0 0];

[0 0 0 0 0 1 1 0 0] [0 0 0 0 1 0 1 0] [0 0 0 0 0 1 1 1] [0 0 0 0 0 1 0 1 1] [0 0 0 0 0 1 0 1 0] [0 0 0 0 1 0 0 0 1] [0 0 0 0 0 1 0 1 1] [0 0 0 0 0 0 1 0 0 1] [0 0 0 0 0 0 0 1 1 0 1] [0 0 0 0 0 0 0 1 1 0 0] [0 0 0 0 0 0 0 1 0 1 0] [0 0 0 0 0 0 0 0 1 1 1] [0 0 0 0 0 0 0 1 0 1] [0 0 0 0 0 0 0 0 1 1 0 1] [0 0 0 0 0 0 0 0 0 1 1 1]};

table24 = {[1 1 1] [1 1 0 1] [1 0 1 1 1 0] [1 0 1 0 0 0 0] [1 0 0 1 0 0 1 0] [1 0 0 0 0 0 1 1 0] [0 1 1 1 1 1 0 0 0] [0 1 1 0 1 1 0 0 1 0] [0 1 1 0 1 0 1 0 1 0] [0 1 0 1 0 0 1 1 1 0 1] [0 1 0 1 0 0 0 1 1 0 1] [0 1 0 1 0 0 0 1 0 0 1] [0 1 0 0 1 1 0 1 1 0 1] [0 1 0 0 0 0 0 0 1 0 1] [0 1 0 0 0 0 0 0 1 0 0 0] [0 0 1 0 1 1 0 0 0];

[1 1 1 0] [1 1 0 0] [1 0 1 0 1] [1 0 0 1 1 0] [1 0 0 0 1 1 1] [1 0 0 0 0 0 1 0] [0 1 1 1 1 0 1 0] [0 1 1 0 1 1 0 0 0] [0 1 1 0 1 0 0 0 1] [0 1 1 0 0 0 1 1 0] [0 1 0 1 0 0 0 1 1 1] [0 1 0 1 0 1 1 0 0 1] [0 1 0 0 1 1 1 1 1 1] [0 1 0 0 1 0 1 0 0 1] [0 1 0 0 0 1 0 1 1 1] [0 0 1 0 1 0 1 0];

[1 0 1 1 1 1] [1 0 1 1 0] [1 0 1 0 0 1] [1 0 0 1 0 1 0] [1 0 0 0 1 0 0] [1 0 0 0 0 0 0 0] [0 1 1 1 1 0 0 0] [0 1 1 0 1 1 1 0 1] [0 1 1 0 0 1 1 1 1] [0 1 1 0 0 0 0 1 0] [0 1 0 1 1 0 1 1 0] [0 1 0 1 0 1 0 1 0 0] [0 1 0 0 1 1 1 0 1 1] [0 1 0 0 1 0 0 1 1 1] [0 1 0 0 0 0 1 1 1 0 1] [0 0 1 0 0 1 0];

[1 0 1 0 0 0 1] [1 0 0 1 1 1] [1 0 0 1 0 1 1] [1 0 0 0 1 1 0] [1 0 0 0 0 1 1 0] [0 1 1 1 1 1 0 1] [0 1 1 1 0 1 0 0] [0 1 1 0 1 1 1 0 0] [0 1 1 0 0 1 1 0 0] [0 1 0 1 1 1 1 1 0] [0 1 0 1 1 0 0 1 0] [0 1 0 1 0 0 0 1 0 1] [0 1 0 0 1 1 0 1 1 1] [0 1 0 0 1 0 0 1 0 1] [0 1 0 0 0 0 1 1 1 1] [0 0 1 0 0 0 0 0];

[1 0 0 1 0 0 1 1] [1 0 0 1 0 0 0] [1 0 0 0 1 0 1] [1 0 0 0 0 1 1 1] [0 1 1 1 1 1 1 1] [0 1 1 1 0 1 1 0] [0 1 1 1 0 0 0 0] [0 1 1 0 1 0 0 1 0] [0 1 1 0 0 1 0 0 0] [0 1 0 1 1 1 1 0 0] [0 1 0 1 1 0 0 0 0 0] [0 1 0 1 0 0 0 0 1 1] [0 1 0 0 1 1 0 0 1 0] [0 1 0 0 0 1 1 1 0 1] [0 1 0 0 0 0 1 1 1 0 0] [0 0 0 1 1 1 0];

[1 0 0 0 0 0 1 1 1] [1 0 0 0 0 1 0] [1 0 0 0 0 0 0 1] [0 1 1 1 1 1 1 0] [0 1 1 1 0 1 1 1] [0 1 1 1 0 0 1 0] [0 1 1 0 1 0 1 1 0] [0 1 1 0 0 1 0 1 0] [0 1 1 0 0 0 0 0 0] [0 1 0 1 1 0 1 0 0] [0 1 0 1 0 1 0 1 0 1] [0 1 0 0 1 1 1 1 0 1] [0 1 0 0 1 0 1 1 0 1] [0 1 0 0 0 1 1 0 0 1] [0 1 0 0 0 0 0 1 1 0] [0 0 0 1 1 0 0];

[0 1 1 1 1 1 0 0 1] [0 1 1 1 1 0 1 1] [0 1 1 1 1 0 0 1] [0 1 1 1 0 1 0 1] [0 1 1 1 0 0 0 1] [0 1 1 0 1 0 1 1 1] [0 1 1 0 0 1 1 1 0] [0 1 1 0 0 0 0 1 1] [0 1 0 1 1 1 0 0 1] [0 1 0 1 0 1 1 0 1 1] [0 1 0 1 0 0 1 0 1 0] [0 1 0 0 1 1 0 1 0 0] [0 1 0 0 1 0 0 0 1 1] [0 1 0 0 0 1 0 0 0 0] [0 1 0 0 0 0 0 1 0 0 0] [0 0 0 1 0 1 0];

[0 1 1 0 1 1 0 0 1 1] [0 1 1 1 0 0 1 1] [0 1 1 0 1 1 1 1] [0 1 1 0 1 1 0 1] [0 1 1 0 1 0 0 1 1] [0 1 1 0 0 1 0 1 1] [0 1 1 0 0 0 1 0 0] [0 1 0 1 1 1 0 1 1] [0 1 0 1 1 0 0 0 0 1] [0 1 0 1 0 0 1 1 0 0] [0 1 0 0 1 1 1 0 0 1] [0 1 0 0 1 0 1 0 1 0] [0 1 0 0 0 1 1 0 1 1] [0 1 0 0 0 0 1 0 0 1 1] [0 0 1 0 1 1 1 1 1 1 0 1] [0 0 0 1 0 0 0 1];

[0 1 1 0 1 0 1 0 1 1] [0 1 1 0 1 0 1 0 0] [0 1 1 0 1 0 0 0 0] [0 1 1 0 0 1 1 0 1] [0 1 1 0 0 1 0 0 1] [0 1 1 0 0 0 0 0 1] [0 1 0 1 1 1 0 1 0] [0 1 0 1 1 0 0 0 1] [0 1 0 1 0 1 0 0 1] [0 1 0 1 0 0 0 0 0 0] [0 1 0 0 1 0 1 1 1 1] [0 1 0 0 1 0 1 1 0 1] [0 1 0 0 0 1 1 0 1 1] [0 1 0 0 0 0 1 0 0 1 1] [0 0 1 0 1 1 1 1 1 0 1] [0 0 0 1 0 0 0 1];

[0 1 0 1 0 0 1 1 1 1] [0 1 1 0 0 0 1 1 1] [0 1 1 0 0 0 1 0 1] [0 1 0 1 1 1 1 1 1] [0 1 0 1 1 1 1 0 1] [0 1 0 1 1 1 0 1 0 1] [0 1 0 1 0 1 1 1 0] [0 1 0 1 0 0 1 1 0 1] [0 1 0 1 0 0 0 0 0 1] [0 1 0 0 1 1 0 0 0 1] [0 1 0 0 1 0 0 0 0 1] [0 1 0 0 0 1 0 0 1 1] [0 1 0 0 0 0 0 1 0 0 1] [0 0 1 0 1 1 1 1 0 1 1] [0 0 1 0 1 1 1 0 0 1 1] [0 0 0 1 0 1 1];

[0 1 0 1 0 0 1 1 1 0] [0 1 0 1 1 1 0 0 0] [0 1 0 1 1 0 1 1 1] [0 1 0 1 1 0 0 1 1] [0 1 0 1 0 1 1 1 1] [0 1 0 1 0 1 1 0 0 0] [0 1 0 1 0 0 1 0 1 1] [0 1 0 0 1 1 1 0 1 0] [0 1 0 0 1 1 0 0 0 0] [0 1 0 0 1 0 0 0 1 0] [0 1 0 0 0 1 0 1 0 1] [0 1 0 0 0 0 1 0 0 1 0] [0 0 1 0 1 1 1 1 1 1 1] [0 0 1 0 1 1 1 0 1 0 1] [0 0 1 0 1 1 0 1 1 1 0] [0 0 0 0 1 0 1 0];

[0 1 0 1 0 0 0 1 1 0 0] [0 1 0 1 0 1 1 0 1 0] [0 1 0 1 0 1 0 1 1] [0 1 0 1 0 1 0 0 0] [0 1 0 1 0 0 1 0 0] [0 1 0 0 1 1 1 1 1 1 0] [0 1 0 0 1 1 0 1 0 1 0] [0 1 0 0 1 0 1 0 1 1] [0 1 0 0 0 1 1 1 1 1] [0 1 0 0 0 1 0 1 0 0] [0 1 0 0 0 0 0 0 1 1 1] [0 1 0 0 0 0 0 0 0 0 0 1] [0 0 1 0 1 1 1 0 1 1 1] [0 0 1 0 1 1 1 0 0 0 0] [0 0 1 0 1 1 0 1 0 1 0] [0 0 0 0 0 1 1 0];

[0 1 0 1 0 0 0 1 0 0 0] [0 1 0 1 0 0 0 0 1 0] [0 1 0 0 1 1 1 1 0 0] [0 1 0 0 1 1 1 0 0 0] [0 1 0 0 1 1 0 0 1 1] [0 1 0 0 1 0 1 1 1 0] [0 1 0 0 1 0 0 1 0 0] [0 1 0 0 0 1 1 1 0 0] [0 1 0 0 0 0 1 1 0 1] [0 0 1 0 0 0 0 0 1 0 1] [0 1 0 0 0 0 0 0 0 0 0] [0 0 1 0 1 1 1 1 1 0 0 0] [0 0 1 0 1 1 1 0 0 1 0] [0 0 1 0 1 1 0 1 1 0 0] [0 0 1 0 1 1 0 0 1 1 1] [0 0 0 0 0 1 0 0];

[0 1 0 0 1 1 0 1 1 0 0] [0 1 0 0 1 0 1 1 0 0] [0 1 0 0 1 0 1 0 0 0] [0 1 0 0 1 0 0 1 1 0] [0 1 0 0 1 0 0 0 0 0] [0 1 0 0 0 1 1 0 1 0] [0 1 0 0 0 1 0 0 0 1] [0 1 0 0 0 0 0 0 1 0 1 0] [0 1 0 0 0 0 0 0 0 1 1] [0 0 1 0 1 1 1 1 1 0 0] [0 0 1 0 1 1 1 0 1 1 0] [0 0 1 0 1 1 1 0 0 0 1] [0 0 1 0 1 1 0 1 1 0 1] [0 0 1 0 1 1 0 1 0 0 1] [0 0 1 0 1 1 0 0 1 0 1] [0 0 1 0 1 1 0 0 1 0 0] [0 0 0 0 0 0 0 0];

[0 1 0 0 0 0 0 0 1 0 0 1] [0 1 0 0 0 1 1 0 0 0] [0 1 0 0 0 1 0 1 1 0] [0 1 0 0 0 1 0 0 1 0] [0 1 0 0 0 0 1 0 1 1] [0 1 0 0 0 0 1 0 0 0] [0 1 0 0 0 0 0 0 1 1] [0 0 1 0 1 1 1 1 1 1 1 0] [0 0 1 0 1 1 1 1 0 1 0] [0 0 1 0 1 1 1 0 1 0 0] [0 0 1 0 1 1 1 0 0 1 1] [0 0 1 0 1 1 1 0 0 1 0 0] [0 0 0 0 0 0 0 0];

[0 0 1 0 1 0 1 1] [0 0 1 0 1 0 0] [0 0 1 0 0 1 1] [0 0 1 0 0 0 1] [0 0 0 1 1 1 1] [0 0 0 1 1 0 1] [0 0 0 1 0 1 1] [0 0 0 1 0 0 1] [0 0 0 0 1 1 1] [0 0 0 0 1 1 0] [0 0 0 0 1 0 0] [0 0 0 0 0 1 1 1] [0 0 0 0 0 1 0 1] [0 0 0 0 0 1 1] [0 0 0 0 0 0 0 1] [0 0 1 1]};

3.3.2.2 โปรแกรม Huff_tabab คือ ตารางฮัฟแมน ตาราง A,B

tablea{1,1,1,1} = [1]; tablea{1,1,1,2} = [0 1 0 1]; tablea{1,1,2,1} = [0 1 0 0]; tablea{1,1,2,2} = [0 0 1 0 1]; tablea{1,2,1,1} = [0 1 1 0];
tablea{1,2,1,2} = [0 0 0 1 0 1]; tablea{1,2,2,1} = [0 0 1 0 0]; tablea{1,2,2,2} = [0 0 0 1 0 0]; tablea{2,1,1,1} = [0 1 1 1]; tablea{2,1,1,2} = [0 0 0 1 1];
tablea{2,1,2,1} = [0 0 1 1 0]; tablea{2,1,2,2} = [0 0 0 0 0 0]; tablea{2,2,1,1} = [0 0 1 1 1]; tablea{2,2,1,2} = [0 0 0 0 1 0];
tablea{2,2,2,1} = [0 0 0 0 1 1]; tablea{2,2,2,2} = [0 0 0 0 0 1]; tableb{1,1,1,1} = [1 1 1 1]; tableb{1,1,1,2} = [1 1 1 0]; tableb{1,1,2,1} = [1 1 0 1];
tableb{1,1,2,2} = [1 1 0 0]; tableb{1,2,1,1} = [1 0 1 1]; tableb{1,2,1,2} = [1 0 1 0]; tableb{1,2,2,1} = [1 0 0 1]; tableb{1,2,2,2} = [1 0 0 0];
tableb{2,1,1,1} = [0 1 1 1]; tableb{2,1,1,2} = [0 1 1 0]; tableb{2,1,2,1} = [0 1 0 1]; tableb{2,1,2,2} = [0 1 0 0]; tableb{2,2,1,1} = [0 0 1 1];
tableb{2,2,1,2} = [0 0 1 0]; tableb{2,2,2,1} = [0 0 0 1]; tableb{2,2,2,2} = [0 0 0 0];

3.3.2.3 โปรแกรม Huff_dec1 เป็นโปรแกรมที่ใช้ถอดรหัสฮัฟแมนในส่วนภาคที่ 1,2,3

%Huffman Decoding

```
if tabs == 0
  h = h+1; is(ch+1,h) = 0; sign(h) = 0;
elseif tabs == 4 & tabs == 14
  for i =1:1000
    disp('Huffman Table not used')
  end
else
  [m,n] = size(table);
  for x = 1:m
    for y = 1:n
      [f,g] = size(table{x,y}); %bs = [bitstream(nbit:nbit+g-1)];
      bs = getnbit(nbit,nbit,nbit+g-1,nbit_f,nbit_2,bitstream);
      if table{x,y} == bs
        huff_bit = huff_bit + g;
          %nbit = nbit + g  ;
          nbit = addnbit(nbit,g,nbit_f,nbit_2);
        break % out of loop 1
      end
    end
    if table{x,y} == bs % out of loop 2
      break
    end
  end
  if table{x,y} ~= bs
      disp('wrong => data not equal data in huffman table')
      % check for data not equal data in huffman table
  end
    h = h+1;
    if x == 16
      bs = getnbit(nbit,nbit,nbit+linbit-1,nbit_f,nbit_2,bitstream);
      %is(ch+1,h) = arlint(bitstream(nbit:linbit+nbit-1))+x-1;
      is(ch+1,h) = arlint(bs)+x-1;
      huff_bit=huff_bit+linbit; %nbit = nbit + linbit;
      nbit = addnbit(nbit,linbit,nbit_f,nbit_2);
    else
      is(ch+1,h) = x-1;
    end
    if is(ch+1,h) ~= 0
      bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream); sign(h) = bs;
      huff_bit=huff_bit+1; inc_nbit;
```

```
    end
    h = h + 1;
    if y == 16
      bs = getnbit(nbit,nbit,nbit+linbit-1,nbit_f,nbit_2,bitstream);
      is(ch+1,h) = arlint(bs)+y-1;
      %nbit = nbit + linbit;
      huff_bit=huff_bit+linbit; nbit = addnbit(nbit,linbit,nbit_f,nbit_2);
    else
      is(ch+1,h) = y-1;
    end
    if is(ch+1,h) ~= 0
      bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
      sign(h) = bs;
      huff_bit=huff_bit+1;
      inc_nbit;
    end
end
```

3.3.2.4 โปรแกรม Huff_decr41 เป็นโปรแกรมที่ใช้ถอดรหัสฮัฟแมนในส่วนภาคที่ 4

```
%decode region(4)
for v = 1:o
  for w = 1:p
    for x = 1:q
      for y = 1:z
        [f,g] = size(table{v,w,x,y});          %bs = [bitstream(nbit:nbit+g-1)];
        bs = getnbit(nbit,nbit,nbit+g-1,nbit_f,nbit_2,bitstream);
        if table{v,w,x,y} == bs
          huff_bit = huff_bit + g; nbit = addnbit(nbit,g,nbit_f,nbit_2); %nbit = nbit + g;
          break % out loop 1
        end
      end
      if table{v,w,x,y} == bs
        break % out loop 2
      end
    end
    if table{v,w,x,y} == bs
      break % out loop 3
    end
  end
  if table{v,w,x,y} == bs
```

```
    break % out loop 4
  end
end
if table{v,w,x,y} ~= bs
  for i=1:1000
    disp('wrong => data not equal data in huffman table') ;% check for data not equal data in huffman table
  end
end
  h = h+1; is(ch+1,h) = x - 1;
  if is(ch+1,h) ~= 0
    bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream); sign(h) = bs; huff_bit=huff_bit+1; inc_nbit ;
  end
  h = h + 1; is(ch+1,h) = y - 1; if is(ch+1,h) ~= 0 ;bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
    sign(h) = bs; huff_bit=huff_bit+1; inc_nbit;
  end
  h = h + 1; is(ch+1,h) = v - 1; if is(ch+1,h) ~= 0
   bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream); sign(h) = bs; huff_bit=huff_bit+1; inc_nbit;
  end
  h = h + 1; is(ch+1,h) = w - 1;
  if is(ch+1,h) ~= 0
    bs = getnbit(nbit,nbit,nbit,nbit_f,nbit_2,bitstream);
    sign(h) = bs; huff_bit=huff_bit+1;          inc_nbit;
  end
```

3.3.3 โปรแกรม descale เป็นโปรแกรมที่ใช้ในการ requantization and reordering

```
clc
disp('----------------------------------------------------------') ;disp('******** Descale ********')
% Scalefactor bands
% ---------------------- 32,44.1,48 kHz sampling rate ----------------------
sfw_l = [4 4 4 4 4 4 6 6 8 10 12 16 20 24 30 38 46 56 68 84 102;
      4 4 4 4 4 4 6 6 8 8 10 12 16 20 24 28 34 42 50 54 76;
      4 4 4 4 4 4 6 6 6 8  10 12 16 18 22 28 34 40 46 54 54]; % long block
sfw_s = [4 4 4 4 6 8 12 16 20 26 34 42;
            4 4 4 4 6 8 10 12 14 18 22 30;
      4 4 4 4 6 6 10 12 14 16 20 26]; % short block
pretab = [0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 2 2 3 3 3 2];
scale_multi = [0.5 1];
% Consider sampling frequency
if sampling_frequency == [0 0]
  fs = 2 ; % 44,1 kHz
```

```
elseif sampling_frequency == [0 1]
  fs = 3 ; % 48 kHz
elseif sampling_frequency == [1 0]
  fs = 1 ; % 32 kHz
else
  disp('sampling fresquency = 1 1 is reserved')
end
i = 0;
if window_switching_flag(gr+1,ch+1) == 1 & block_type{gr+1,ch+1} == [1 0]
  if mixed_block_flag(gr+1,ch+1) == 1
    for sfb = 0:7
      for j = 1:sfw_l(fs,sfb+1)
        i = i + 1; sf(i) = arlint(scalefac_l{gr+1,ch+1,sfb+1}); sbg(i) = 0; pt(i) = pretab(sfb+1);
      end
    end
    for sfb = 3:11
      for window = 0:2
        for j = 1:sfw_s(fs,sfb+1)
          i = i + 1; sf(i) = arlint(scalefac_s{gr+1,ch+1,sfb+1,window+1});
            sbg(i) = arlint(subblock_gain{gr+1,ch+1,window+1}); pt(i) = 0;
        end
      end
    end
  else
    for sfb = 0:11
      for window = 0:2
        for j = 1:sfw_s(fs,sfb+1)
          i = i + 1; sf(i) = arlint(scalefac_s{gr+1,ch+1,sfb+1,window+1});
          sbg(i) = arlint(subblock_gain{gr+1,ch+1,window+1}); pt(i) = 0;
        end
      end
    end
  end
else
  for sfb = 0:20
    for j = 1:sfw_l(fs,sfb+1)
      i = i + 1; sf(i) = arlint(scalefac_l{gr+1,ch+1,sfb+1});
      sbg(i) = 0; pt(i) = pretab(sfb+1);
    end
  end
```

```
end
% Zeropad for the rest of the spectrum
j = i+1;
for i = j:576
  sf(i) = 0; sbg(i) = 0; pt(i) = 0;
end
% ----------------------Reordering-----------------------------
if window_switching_flag(gr+1,ch+1) == 1 & block_type{gr+1,ch+1} == [1 0]
  if mixed_block_flag(gr+1,ch+1) == 1
    for sb = 2:31 %for mixed block and reoder only short block
      for window = 0:2
        for ws=0:5
          is(ch+1,18*sb+6*window+ws+1) = is(ch+1,18*sb+window+3*ws+1);
        end
      end
    end
  else
    for sb = 0:31 %for short block
      for window =0:2
        for ws=0:5
          is(ch+1,18*sb+ws+1) = is(ch+1,18*sb+window+3*ws+1);
        end
      end
    end
  end
end
for i=1:576
  if sign(i) == 1
    signs = -1;
  else
    signs = 1;
  end
  power(ch+1,i) =((1/4)*(arlint(global_gain{gr+1,ch+1})-210-8*sbg(i)))-(scale_multi(scalefac_scale(gr+1,ch+1)+1)*sf
(i)+preflag(gr+1,ch+1)*pt(i));
  xr(ch+1,i) = signs*((is(ch+1,i))^(4/3))*(2^(power(ch+1,i)));
end
disp('-------------------------------------------------------')
if sw==1
figure(4); bar(abs(xr(ch+1,:))); title('Dequantization"s output => xr (Absolute)');
xlabel('Sample frequency'); ylabel('|Size|'); colormap([0.6 0.2 0.8]);
```

```
end
pause
```

3.3.4 โปรแกรม alias เป็นการทำ aliasing reduction

```
disp('******** Alias Reduction ********')
  C = [-0.6 -0.535 -0.33 -0.185 -0.095 -0.041 -0.0142 -0.0037];
  for i=0:7
    Cs(i+1) = 1/sqrt(1 + C(i+1)^2); Ca(i+1) = C(i+1)/sqrt(1 + C(i+1)^2);
  end
  al = zeros(1,576);
  if arlint(block_type{gr+1,ch+1}) ~= 2
    for sb=1:31
      for i=0:7
        xal(ch+1,18*sb-1-i) = xr(ch+1,18*sb-1-i)*Cs(i+1)-xr(ch+1,18*sb+i)*Ca(i+1);
        al(18*sb-1-i) = 1; al(18*sb+i) = 1;
        xal(ch+1,18*sb+i) = xr(ch+1,18*sb+i)*Cs(i+1)-xr(ch+1,18*sb-1-i)*Ca(i+1);
      end
    end
  end
  for i=1:576
    if al(i) ~= 1
      xal(ch+1,i) = xr(ch+1,i);
    end
  end
if arlint(block_type{gr+1,ch+1})~= 2
  disp('Alias Reduction is applied for long block_type grannule')
  disp('-------------------------------------------------------')
  if sw==1
  figure(4)
  bar(abs(xal(ch+1,:)))
  title('Aliasing Reduction (Absolute)')
  xlabel('Sample Frequency')
  ylabel('|size|')
  colormap([0.7 0.2 0.4])
  end
else
  disp('Alias Reduction is not applied for short block_type grannule')
  xal = xr; disp('-------------------------------------------------------')
end
pause
```

3.3.5 โปรแกรม IMDCT เป็นโปรแกรมทำ IMDCT

```
disp('******** Start imdct ********');
for i=0:11
  for k=0:5
    xshort(i+1,k+1)=cos((pi/24)*((2*i)+7)*((2*k)+1));
  end
end
for i=0:35
  for k=0:17
    xlong(i+1,k+1)=cos((pi/72)*((2*i)+19)*((2*k)+1));
  end
end
l=0;
if block_type{gr+1,ch+1}==[1 0]
  for sb=0:31
    for j=0:2
      for i=0:11
        xx = 0;
        for k=0:5
          xx=xx+xal(ch+1,(sb*18)+(j*6)+k+1)*cos((pi/24)*((2*i)+7)*((2*k)+1));
          xk(k+1)=xal(ch+1,(sb*18)+(j*6)+k+1);
        end
        l=l+1; Xi(l)=xx; X_i(i+1)=xx; Xi_short{sb+1,j+1,i+1}=xx;
      end
    end
  end
else
  disp('Long block');
  for sb=0:31
    for i=0:35
      xx = 0;
      for k=0:17
        xx=xx+xal(ch+1,(sb*18)+k+1)*cos((pi/72)*((2*i)+19)*((2*k)+1));
            xk(k+1)=xal(ch+1,(sb*18)+k+1);
      end
      l=l+1; Xi(l)=xx; X_i(i+1)=xx; Xi_long{sb+1,i+1}=xx;
    end
  end
end
```

```
l=0;
for sb=0:31
  for i=0:35
    l=l+1;
    Xi_sb(sb+1,i+1)=Xi(l);
  end
end
for i=0:35
  for j=0:31
    X_sb(i+1,j+1)=Xi_sb(j+1,i+1);
  end
end
ii=0;
for i=0:31
  ii = ii+1; jj=0;
  for j=0:35
    jj =jj+1; nbx(sb+1,i+1)=ii; nby(sb+1,i+1)=jj;
  end
end
disp('-------------------------------------------------------')
if sw==1
figure(8); clf; Xsb=abs(X_sb); bar3(Xsb); xlabel('Subband');
ylabel('Time sample'); zlabel('|size|');
if block_type{gr+1,ch+1}==[1 0]
  title('Short block IMDCT  (12 subband / 3 windows in each subband) ')
else
  title('Long block IMDCT (36 subband)')
end
pause; figure(9); clf; ribbon(X_sb); xlabel('Subband'); ylabel('Time sample'); zlabel('size');
if block_type{gr+1,ch+1}==[1 0]
  title('Short block IMDCT  (12 subband / 3 windows in each subband) ')
else
  title('Long block IMDCT (36 subband)')
end
end
pause
```

3.3.6 โปรแกรม Windowing2 เป็นโปรแกรมการทำ Windowing

```
disp('******** Windowing ********')
if mixed_block_flag(gr+1,ch+1)~=1
```

```
disp('Mixed block flag = 0');
if block_type{gr+1,ch+1}== [0 0]
  for sb=0:31
    for i=0:35
      Wi_long{sb+1,i+1}=sin((pi/36)*(i+(1/2))); Zi{sb+1,i+1}=Xi_long{sb+1,i+1}*Wi_long{sb+1,i+1};
    end
  end
elseif block_type{gr+1,ch+1}==[0 1]
  for sb=0:31
    for i=0:17
      Wi_long{sb+1,i+1}=sin((pi/36)*(i+(1/2))); Zi{sb+1,i+1}=Xi_long{sb+1,i+1}*Wi_long{sb+1,i+1};
    end
    for i=18:23
      Zi{sb+1,i+1}=Xi_long{sb+1,i+1};
    end
    for i=24:29
      Wi_long{sb+1,i+1}=sin((pi/12)*(i-18+(1/2))); Zi{sb+1,i+1}=Xi_long{sb+1,i+1}*Wi_long{sb+1,i+1};
    end
    for i=30:35
      Zi{sb+1,i+1}=0;
    end
  end
elseif block_type{gr+1,ch+1}==[1 1]
  for sb=0:31
    for i=0:5
      Zi{sb+1,i+1}=0;
    end
    for i=6:11
      Wi_long{sb+1,i+1}=sin((pi/12)*(i-6+(1/2))); Zi{sb+1,i+1}=Xi_long{sb+1,i+1}*Wi_long{sb+1,i+1};
    end
    for i=12:17
      Zi{sb+1,i+1}=Xi_long{sb+1,i+1};
    end
      for i=18:35
      Wi_long{sb+1,i+1}=sin((pi/36)*(i+(1/2))); Zi{sb+1,i+1}=Xi_long{sb+1,i+1}*Wi_long{sb+1,i+1};
    end
  end
elseif block_type{gr+1,ch+1}==[1 0]
  for sb=0:31
    for j=0:2
```

```
        for i=0:11
                Wi_short{sb+1,j+1,i+1}=sin((pi/12)*(i+(1/2)));
                Yi{sb+1,j+1,i+1}=Xi_short{sb+1,j+1,i+1}*Wi_short{sb+1,j+1,i+1};
        end
      end
    end
    for sb=0:31
      for i=0:5
        Zi{sb+1,i+1}=0;
      end
      for i=6:11
        Zi{sb+1,i+1}=Yi{sb+1,1,i-6+1};
      end
      for i=12:17
        Zi{sb+1,i+1}=Yi{sb+1,1,i-6}+Yi{sb+1,2,i-12+1};
      end
      for i=18:23
        Zi{sb+1,i+1}=Yi{sb+1,2,i-12}+Yi{sb+1,3,i-18+1};
      end
      for i=24:29
        Zi{sb+1,i+1}=Yi{sb+1,3,i-18+1};
      end
      for i=30:35
        Zi{sb+1,i+1}=0;
      end
    end
  end
  else
    disp('Mixed block flag = 1')
    disp('Two lowest subband use normal window type')
    for sb=0:1
      for i=0:35
        Wi_long{sb+1,i+1}=sin((pi/36)*(i+(1/2))); Zi{sb+1,i+1}=Xi_long{sb+1,i+1}*Wi_long{sb+1,i+1};
      end
    end
  if block_type{gr+1,ch+1}== [0 0]
    for sb=2:31
      for i=0:35
        Wi_long{sb+1,i+1}=sin((pi/36)*(i+(1/2))); Zi{sb+1,i+1}=Xi_long{sb+1,i+1}*Wi_long{sb+1,i+1};
      end
```

```
  end
elseif block_type{gr+1,ch+1}==[0 1]
  for sb=2:31
    for i=0:17
      Wi_long{sb+1,i+1}=sin((pi/36)*(i+(1/2))); Zi{sb+1,i+1}=Xi_long{sb+1,i+1}*Wi_long{sb+1,i+1};
    end
    for i=18:23
      Zi{sb+1,i+1}=Xi_long{sb+1,i+1};
    end
    for i=24:29
      Wi_long{sb+1,i+1}=sin((pi/12)*(i-18+(1/2))); Zi{sb+1,i+1}=Xi_long{sb+1,i+1}*Wi_long{sb+1,i+1};
    end
    for i=30:35
      Zi{sb+1,i+1}=0;
    end
  end
elseif block_type{gr+1,ch+1}==[1 1]
  for sb=2:31
    for i=0:5
      Zi{sb+1,i+1}=0;
    end
    for i=6:11
      Wi_long{sb+1,i+1}=sin((pi/12)*(i-6+(1/2))); Zi{sb+1,i+1}=Xi_long{sb+1,i+1}*Wi_long{sb+1,i+1};
    end
    for i=12:17
      Zi{sb+1,i+1}=Xi_long{sb+1,i+1};
    end
      for i=18:35
      Wi_long{sb+1,i+1}=sin((pi/36)*(i+(1/2))); Zi{sb+1,i+1}=Xi_long{sb+1,i+1}*Wi_long{sb+1,i+1};
    end
  end
elseif block_type{gr+1,ch+1}==[1 0]
  ss=i;
  %Window=j;
  for sb=2:31
    for j=0:2
      for i=0:11
        Wi_short{sb+1,j+1,i+1}=sin((pi/12)*(i+(1/2)));
        Yi{sb+1,j+1,i+1}=Xi_short{sb+1,j+1,i+1}*Wi_short{sb+1,j+1,i+1};
      end
```

```
    end
  end
  for sb=2:31
    for i=0:5
      Zi{sb+1,i+1}=0;
    end
    for i=6:11
      Zi{sb+1,i+1}=Yi{sb+1,1,i-6+1};
    end
    for i=12:17
      Zi{sb+1,i+1}=Yi{sb+1,1,i-6}+Yi{sb+1,2,i-12+1};
    end
    for i=18:23
      Zi{sb+1,i+1}=Yi{sb+1,2,i-12}+Yi{sb+1,3,i-18+1};
    end
    for i=24:29
      Zi{sb+1,i+1}=Yi{sb+1,3,i-18+1};
    end
    for i=30:35
      Zi{sb+1,i+1}=0;
    end
  end
end
end
l=0;
for i=1:32
  for j=1:36
    wind(i,j)=Zi{i,j}; l=l+1; wd(l)=Zi{i,j};
  end
end
if sw==1
for i=1:36
  for j=1:32
    ww(i,j)=wind(j,i);
  end
end
figure(13);ribbon(ww);xlabel('subband');ylabel('sample');zlabel('time');zlabel('size');title('Windowing');
end
pause
```

3.3.7 โปรแกรม Overlap เป็นการทำโอเวอร์แลป

```
if s_ovl == 0
  Si=zeros(32,18); s_ovl =1;
end
disp('******** Overlap ********')
  for sb = 0:31
    for i=0:17
      Oi(sb+1,i+1) = wind(sb+1,i+1) + Si(sb+1,i+1); Si(sb+1,i+1) = wind(sb+1,i+18+1);
    end
  end
  if sw==1
  l=0;
  for i=0:31
    for j=0:17
      l=l+1; Oii(l)=Oi(i+1,j+1); Ooi(j+1,i+1)=Oi(i+1,j+1);
    end
  end
  figure(15); ribbon(Ooi); xlabel('subband'); ylabel('sample'); zlabel('size'); title('overlap'); l=0;
  for i=0:31
    for j=0:17
      l=l+1; Soi(j+1,i+1)=Si(i+1,j+1);
    end
  end
  end
  pause
```

3.3.8 โปรแกรม p_f เป็น โปรแกรมทำโพลีเฟสฟิลเตอร์แบงค์

```
 disp('******** Polyphase filter ********')
Os=Oi; %invf
for sb=1:2:31
  for i=1:2:17
    Os(sb+1,i+1) = Os(sb+1,i+1)*(-1);
  end
end
for i=1:32
  for j=1:18
    Osi(j,i)=Os(i,j);
  end
end
  if init==0
```

```
for i=0:63
  for k=0:31
    N(i+1,k+1)= cos((16+i)*(2*k+1)*pi/64);
  end
end
V = zeros(1,1024); Di_tab; pm = 0; init=1;
end
for ts=0:17
for i=1023:-1:64
        V(i+1) = V(i+1-64);
end
for i=0:63
    tmp=0;
    for k=0:31
      tmp = tmp + N(i+1,k+1)*Os(k+1,ts+1);
    end
    V(i+1) = tmp;
  end
  for i=0:7
    for j=0:31
      U(64*i+j+1)=V(128*i+j+1); U(64*i+32+j+1)=V(128*i+96+j+1);
    end
  end
  for i=0:511
    Wi(i+1)=U(i+1)*Di(i+1);
  end
  for k=0:31
    tmp=0;
    for i=0:15
      tmp = tmp + Wi(k+32*i+1);
    end
    pm = pm+1; Pcm(pm)=tmp;
  end
end
initial=1;
```

3.3.8.1 โปรแกรม Di_tab เป็นโปรแกรม windowing fuction ของโพลีเฟสฟิลเตอร์แบงค์

```
Di=[ 0.000000000;-0.000015159;-0.000015259;-0.000015259;-0.000015259;-0.000015259;-0.000015259;-0.000030518;-0.000030518;-0.000030518;-
0.000030518;-0.000045776;-0.000045776;-0.000061035;-0.000061035;-0.000076294;-0.000076294;-0.000091553;-0.000106812;-0.000106812;
  -0.000122070;-0.000137329;-0.000152588;-0.000167847;-0.000198364;-0.000213623;-0.000244141;-0.000259399;-0.000289917;-0.000320435;-
  0.000366211;-0.000396729;-0.000442505;-0.000473022;-0.000534058;-0.000579834;-0.000625610;-0.000686646;-0.000747681;-0.000808716;
```

-0.000885010;-0.000961304;-0.001037598;-0.001113892;-0.001205444;-0.001296997;-0.001388550;-0.001480103;-0.001586914;-0.001693726;-0.001785278;-0.001907349;-0.002014160;-0.002120972;-0.002243042;-0.002349854;-0.002456665;-0.002578435;-0.002685547;-0.002792358;-0.002899170;-0.002990723;-0.003082275;-0.003173828; 0.003250122; 0.003326416; 0.003387451; 0.003433228; 0.003463745; 0.003479004; 0.003479004; 0.003463745; 0.003417969; 0.003372192; 0.003280640; 0.003173828; 0.003051758; 0.002883911; 0.002700806; 0.002487183; 0.002227793; 0.001937866; 0.001617432; 0.001266479; 0.000869751; 0.000442505; 0.000030518; 0.000549316;-0.001098633;-0.001693726;-0.002334595;-0.003005981;-0.003723145;-0.004486084;-0.005294800;-0.006118774;-0.007003784;-0.007919312;-0.008865356;-0.009841919;-0.010848999;-0.011886597;-0.012939453;-0.014022827;-0.015121460;-0.016235352;-0.017349243;-0.018463135;-0.019577026;-0.020690918;-0.021789551;-0.022857666;-0.023910522;-0.024932861;-0.025909424;-0.026840210;-0.027725220;-0.028533936;-0.029281616;-0.029937744;-0.030532837;-0.031005859;-0.031387329;-0.031661987;-0.031814575;-0.031845093;-0.031738281;-0.031478882; 0.031082153; 0.030517578; 0.029785156; 0.028884888; 0.027801514; 0.026535034; 0.025085449; 0.023422241; 0.021575928; 0.019531250; 0.017257690; 0.014801025; 0.012115479; 0.009231567; 0.006134033; 0.002822876;-0.000686646;-0.004394531;-0.008346040;-0.012420654;-0.016708374;-0.021179199;-0.025817871;-0.030609131;-0.035552979;-0.040634155;-0.045837402;-0.051132202;-0.056533813;-0.061996460;-0.067520142;-0.073059082;-0.078628540;-0.084182739;-0.089706421;-0.095169067;-0.100540161;-0.105819702;-0.110946655;-0.115921021;-0.120697021;-0.125259399;-0.129562378;-0.133590698;-0.137298584;-0.140670776;-0.143676758;-0.146255493;-0.148422241;-0.150115967;-0.151306152;-0.151962280;-0.152069092;-0.151596069;-0.150497437;-0.148773193;-0.146362305;-0.143264771;-0.139450073;-0.134887695;-0.129577637;-0.123474121;-0.116577148;-0.105586201; 0.100311279; 0.090927124; 0.080688477; 0.069595337; 0.057617187; 0.044784546; 0.031082153; 0.016510010; 0.001068115;-0.015228271;-0.032379150;-0.050354004;-0.069168091;-0.088775635;-0.109161377;-0.130310059;-0.152206421;-0.174789429;-0.198059082;-0.221984863;-0.246805737;-0.271591187;-0.297210693;-0.323318481;-0.349868774;-0.376800537;-0.404083252;-0.431655884;-0.459472656;-0.487472534;-0.515609741;-0.543823242;-0.572036743;-0.600219727;-0.628295898;-0.656219482;-0.683914185;-0.711218970;-0.738372803;-0.765029907;-0.791213989;-0.816864014;-0.841949463;-0.866363525;-0.890090942;-0.913055420;-0.935195923;-0.956481934;-0.976852417;-0.996246338;-1.014617920;-1.031936646;-1.048156738;-1.063217163;-1.077117920;-1.089782715;-1.101211548;-1.111373901;-1.120223999;-1.127746582;-1.133926392;-1.138763428;-1.142211914;-1.144287109; 1.144989014; 1.144287109; 1.142211914; 1.138763428; 1.133926392; 1.127746582; 1.120223999; 1.111373901; 1.101211548; 1.089782715; 1.077117920; 1.063217163; 1.048156738; 1.031936646; 1.014617920; 0.996246338; 0.976852417; 0.956481934; 0.935195923; 0.913055420; 0.890090942; 0.866363525; 0.841949463; 0.816864014; 0.791213989; 0.765029907; 0.738372803; 0.711318970; 0.683914185; 0.656219482; 0.628295898; 0.600219727; 0.572036743; 0.543823242; 0.515609741; 0.487472534; 0.459472656; 0.431655884; 0.404083252; 0.376800537; 0.349868774; 0.323318481; 0.297210693; 0.271591187; 0.246505737; 0.221984863; 0.198059082; 0.174789429; 0.152206421; 0.130310059; 0.109161377; 0.088775635; 0.069168091; 0.050354004; 0.032379150; 0.015228271;-0.001068115;-0.016510010;-0.031082153;-0.044784546;-0.057617187;-0.069595337;-0.080688477;-0.090927124; 0.100311279; 0.108856201; 0.116577148; 0.123474121; 0.129577637; 0.134887695; 0.139450073; 0.143264771; 0.146362305; 0.148773193; 0.150497437; 0.151596069; 0.152069092; 0.151962280; 0.151306152; 0.150115967; 0.148422241; 0.146255493; 0.143676758; 0.140670776; 0.137298584; 0.133590698; 0.129562378; 0.125259399; 0.120697021; 0.115921021; 0.110946655; 0.105819702; 0.100540161; 0.095169067; 0.089706421; 0.084182739; 0.078628540; 0.073059082; 0.067520142; 0.061996460; 0.056533813; 0.051132202; 0.045837402; 0.040634155; 0.035552979; 0.030609131; 0.025817871; 0.021179199; 0.016708374; 0.012420654; 0.008316040; 0.004394531; 0.000686646;-0.002822876;-0.006134033;-0.009231567;-0.012115479;-0.014801025;-0.017257690;-0.019531250;-0.021575928;-0.023422241;-0.025085449;-0.026535034;-0.027801514;-0.028884888;-0.029785156;-0.030517578; 0.031082153; 0.031478882; 0.031738281; 0.031845093; 0.031814575; 0.031661987; 0.031387239; 0.031005859; 0.030532837; 0.029937744; 0.029281616; 0.028533936; 0.027725220; 0.026840210; 0.025909424; 0.024932861; 0.023910522; 0.022857666; 0.021789551; 0.020690918; 0.019577026; 0.018463135; 0.017349243; 0.016235352; 0.015121460; 0.014022827; 0.012939453; 0.011886597; 0.010848999; 0.009841919; 0.008865356; 0.007919312; 0.007003784; 0.006118774; 0.005294800; 0.004486084; 0.003723145; 0.003005981; 0.002334595; 0.001693726; 0.001098633; 0.000549316; 0.000030518;-0.000442505;-0.000869751;-0.001266479;-0.001617432;-0.001937866;-0.002227783;-0.002487183;-0.002700806;-0.002883911;-0.003051758;-0.003173828;-0.003280640;-0.003372192;-0.003417969;-0.003463745;-0.003479004;-0.003479004;-0.003463745;-0.003433228;-0.003387451;-0.003326416; 0.003250122; 0.003173828; 0.003082275; 0.002990723; 0.002899170; 0.002792358; 0.002685547; 0.002578735; 0.002456665; 0.002349854; 0.002243042; 0.002120972; 0.002014160; 0.001907349; 0.001785278; 0.001693726; 0.001586914; 0.001480103; 0.001388550; 0.001296997; 0.001205444; 0.001113892; 0.001037598; 0.000961304; 0.000885010; 0.000808716; 0.000747681; 0.000686646; 0.000625610; 0.000579834; 0.000534058; 0.000473022; 0.000442505; 0.000396729; 0.000366211; 0.000320435; 0.000289917; 0.000259399; 0.000244141; 0.000213623; 0.000198364; 0.000167847; 0.000152588; 0.000137329; 0.000122070; 0.000106812; 0.000106812; 0.000091553; 0.000076294; 0.000076294; 0.000061035; 0.000061035; 0.000045776; 0.000045776; 0.000030518; 0.000030518; 0.000030518; 0.000030518; 0.000015259; 0.000015259; 0.000015259; 0.000015259; 0.000015259];

4 โปรแกรมและฟังก์ชั่นอื่นๆ เป็นโปรแกรมที่ใช้ในการเซ็ทค่า,เพิ่มค่า,บวกค่าให้มีความถูกต้อง ได้แก่

โปรแกรม inc_bit เป็นโปรแกรมเพิ่มค่าให้กับตัวชี้บิตสตรีม

```
nbit = nbit+1;
if nbit == nbit_f
    nbit = nbit_2;
end
```

ฟังก์ชั่น addnbit เป็นฟังก์ชั่นที่ใช้ในการบวกค่าของตัวชี้บิตสตรีม

```
% example nbit = nbit + linbit  % use nbit = addnbit(nbit,linbit,nbit_f,nbit_2)
function[nbit] = addnbit(nbit,number,nbit_f,nbit_2)
if number > 0
    for n=1:number
        nbit = nbit+1;
        if nbit == nbit_f;
            nbit = nbit_2;
        end
    end
elseif number < 0
  for n=-1:-1:number
    nbit = nbit -1;
    if nbit == nbit_2-1
    nbit = nbitf-1;
    end
  end
end
```

ฟังก์ชั่น getnbit เป็นฟังก์ชั่นที่ใช้รับค่าจากบิตสตรีมมาเก็บไว้

```
% example bs = [bitstream(nbit:nbit+g-1)]
% format function[bs] = getnbit(nbit,start,stop,nbit_f,nbit_2,bitstream)
% use bs2 = getnbit(nbit,nbit,nbit+g-1,nbit_f,nbit_2,bitstream)
function[bs] = getnbit(nbit,start,stop,nbit_f,nbit_2,bitstream)
i = 0; [g,h]=size(bitstream);
if stop > h
  disp('Can not getnbit'); stop = h;
end
if start <= stop
  for n=start:stop
    i=i+1;
    if nbit == nbit_f
```

```
      nbit = nbit_2;
    end
    bs(i)=bitstream(nbit); nbit = nbit+1;
  end
else
  bs=0;
end
```

ฟังก์ชั่น arlint เป็นฟังก์ชั่นในการแปลงบิตสตรีมให้เป็นข้อมูลในรูปเลขฐาน 10

```
function[int] = arlint(a)
% Function => Array convert to integer.
% Array must be 2-D for 1XN matrix.
% Previous bit from bitstream is more significant.
% Elements of matrix is only 0 or 1 (per 1 element).
d = 0; d1 = 0; [m,n] = size(a);
for i=1:n
  d1 = bitshift(a(i),n-i); d = d + d1;
end
int = d;
```

ภาคผนวก ง

VHDL Source Code

```
library IEEE;
use IEEE.Std_Logic_1164.all;
use ieee.std_logic_unsigned.all;
Entity mp3decoder is
port        (clk, en, reset:in std_logic;
             sync:out std_logic_vector(11 downto 0):="000000000000";
             id, protection_bit, padding_bit, private_bit,
             copyright, original:out std_logic:='0';
             layer, sampling_freq, mode, mode_ext,
             emphasis:out std_logic_vector(1 downto 0):="00";
             bitrate_index:out std_logic_vector(3 downto 0):="0000";
             crc:out std_logic_vector(15 downto 0):="0000000000000000";
             main_data_begin:out std_logic_vector(8 downto 0):="000000000";
             private_bits:out std_logic_vector(4 downto 0):="00000";
             scfsi_band0, scfsi_band1, scfsi_band2, scfsi_band3:out std_logic:='0';
             part2_3_length_gr0, part2_3_length_gr1:out std_logic_vector(11 downto 0):="000000000000";
             big_values_gr0, big_values_gr1:out std_logic_vector(8 downto 0):="000000000";
             global_gain_gr0, global_gain_gr1:out std_logic_vector(7 downto 0):="00000000";
             scalefac_compress_gr0, scalefac_compress_gr1:out std_logic_vector(3 downto 0):="0000";
             window_switching_flag_gr0, window_switching_flag_gr1:out std_logic:='0';
             block_type_gr0, block_type_gr1:out std_logic_vector(1 downto 0):="00";
             mixed_block_flag_gr0, mixed_block_flag_gr1:out std_logic:='0';
             table_select_gr0_region0, table_select_gr0_region1, table_select_gr0_region2,
             table_select_gr1_region0, table_select_gr1_region1, table_select_gr1_region2
             :out std_logic_vector(4 downto 0):="00000";
             subblock_gain_gr0_window0, subblock_gain_gr0_window1, subblock_gain_gr0_window2,
             subblock_gain_gr1_window0, subblock_gain_gr1_window1, subblock_gain_gr1_window2
             :out std_logic_vector(2 downto 0):="000";
             region0_count_gr0, region0_count_gr1:out std_logic_vector(3 downto 0):="0000";
             region1_count_gr0, region1_count_gr1:out std_logic_vector(2 downto 0):="000";
             preflag_gr0, preflag_gr1:out std_logic:='0';
             scalefac_scale_gr0, scalefac_scale_gr1:out std_logic:='0';
             count1table_select_gr0, count1table_select_gr1:out std_logic:='0';
             bitcount:out integer:=0;
             finish:out std_logic:='0');
end;
Architecture behv of mp3decoder is
```

```
component main_control
        port (din, clk, en, reset, header_finish_in, audio_data_finish_in:in std_logic;
         header_count, audio_data_count:in integer;
         dout, header_en, audio_data_en, header_reset, audio_data_reset, finish:out std_logic:='0';
         addr_out, bit_count:out integer:=0);
end component;
component header
        port (din, en, reset, clk:in std_logic;
         sync:out std_logic_vector(11 downto 0):="000000000000";
         id, protection_bit, padding_bit, private_bit,
         copyright, original, finish_out:out std_logic:='0';
         layer, sampling_freq, mode, mode_ext,
         emphasis:out std_logic_vector(1 downto 0):="00";
         bitrate_index:out std_logic_vector(3 downto 0):="0000";
         crc:out std_logic_vector(15 downto 0):="0000000000000000";
         data_count:out integer:=0);
end component;
component audio_data
        port (din, en, reset, clk:in std_logic;
         mode_in, layer_in:in std_logic_vector(1 downto 0);
         count_in:in integer;
         main_data_begin:out std_logic_vector(8 downto 0):="000000000";
         private_bits:out std_logic_vector(4 downto 0):="00000";
         scfsi_band0, scfsi_band1, scfsi_band2, scfsi_band3:out std_logic:='0';
         part2_3_length_gr0, part2_3_length_gr1:out std_logic_vector(11 downto 0):="000000000000";
         big_values_gr0, big_values_gr1:out std_logic_vector(8 downto 0):="000000000";
         global_gain_gr0, global_gain_gr1:out std_logic_vector(7 downto 0):="00000000";
         scalefac_compress_gr0, scalefac_compress_gr1:out std_logic_vector(3 downto 0):="0000";
         window_switching_flag_gr0, window_switching_flag_gr1:out std_logic:='0';
         block_type_gr0, block_type_gr1:out std_logic_vector(1 downto 0):="00";
         mixed_block_flag_gr0, mixed_block_flag_gr1:out std_logic:='0';
         table_select_gr0_region0, table_select_gr0_region1, table_select_gr0_region2,
         table_select_gr1_region0, table_select_gr1_region1, table_select_gr1_region2
         :out std_logic_vector(4 downto 0):="00000";
         subblock_gain_gr0_window0, subblock_gain_gr0_window1, subblock_gain_gr0_window2,
         subblock_gain_gr1_window0, subblock_gain_gr1_window1, subblock_gain_gr1_window2
         :out std_logic_vector(2 downto 0):="000";
         region0_count_gr0, region0_count_gr1:out std_logic_vector(3 downto 0):="0000";
         region1_count_gr0, region1_count_gr1:out std_logic_vector(2 downto 0):="000";
         preflag_gr0, preflag_gr1:out std_logic:='0';
```

```
        scalefac_scale_gr0, scalefac_scale_gr1:out std_logic:='0';
        count1table_select_gr0, count1table_select_gr1:out std_logic:='0';
        data_count:out integer:=0;
        finish_out:out std_logic:='0');
  end component;
  component waveform_10K
        port (addr:in integer:=0;
            dout:out std_logic:='0');
  end component;
signal    ri0:integer:=0;
signal    ro0:std_logic:='0';
signal    ci0, ci1, ci2, ci3, ci4, ci5:std_logic:='0';
signal    ci6, ci7:integer:=0;
signal    co0, co1, co2, co3, co4, co5:std_logic:='0';
signal    co6, co7:integer:=0;
signal    hi0, hi1, hi2, hi3:std_logic:='0';
signal    ho0:std_logic_vector(11 downto 0):="000000000000";
signal    ho1, ho2, ho3, ho4, ho5, ho6, ho7:std_logic:='0';
signal    ho8, ho9, ho10, ho11, ho12:std_logic_vector(1 downto 0):="00";
signal    ho13:std_logic_vector(3 downto 0):="0000";
signal    ho14:std_logic_vector(15 downto 0):="0000000000000000";
signal    ho15:integer:=0;
signal    ai0, ai1, ai2, ai3:std_logic:='0';
signal    ai4, ai5:std_logic_vector(1 downto 0):="00";
signal    ai6:integer:=0;
signal    ao0:std_logic_vector(8 downto 0):="000000000";
signal    ao1:std_logic_vector(4 downto 0):="00000";
signal    ao2, ao3, ao4, ao5:std_logic:='0';
signal    ao6, ao7:std_logic_vector(11 downto 0):="000000000000";
signal    ao8, ao9:std_logic_vector(8 downto 0):="000000000";
signal    ao10, ao11:std_logic_vector(7 downto 0):="00000000";
signal    ao12, ao13:std_logic_vector(3 downto 0):="0000";
signal    ao14, ao15:std_logic:='0';
signal    ao16, ao17:std_logic_vector(1 downto 0):="00";
signal    ao18, ao19:std_logic:='0';
signal    ao20, ao21, ao22, ao23, ao24, ao25:std_logic_vector(4 downto 0):="00000";
signal    ao26, ao27, ao28, ao29, ao30, ao31:std_logic_vector(2 downto 0):="000";
signal    ao32, ao33:std_logic_vector(3 downto 0):="0000";
signal    ao34, ao35:std_logic_vector(2 downto 0):="000";
signal    ao36, ao37, ao38, ao39, ao40, ao41:std_logic:='0';
```

```
signal    ao42:integer:=0;
signal    ao43:std_logic:='0';
signal    mi0, mi1, mi2:std_logic:='0';
signal    mo0:std_logic_vector(11 downto 0):="000000000000";
signal    mo1, mo2, mo3, mo4, mo5, mo6:std_logic:='0';
signal    mo7, mo8, mo9, mo10, mo11:std_logic_vector(1 downto 0):="00";
signal    mo12:std_logic_vector(3 downto 0):="0000";
signal    mo13:std_logic_vector(15 downto 0):="0000000000000000";
signal    mo14:std_logic_vector(8 downto 0):="000000000";
signal    mo15:std_logic_vector(4 downto 0):="00000";
signal    mo16, mo17, mo18, mo19:std_logic:='0';
signal    mo20, mo21:std_logic_vector(11 downto 0):="000000000000";
signal    mo22, mo23:std_logic_vector(8 downto 0):="000000000";
signal    mo24, mo25:std_logic_vector(7 downto 0):="00000000";
signal    mo26, mo27:std_logic_vector(3 downto 0):="0000";
signal    mo28, mo29:std_logic:='0';
signal    mo30, mo31:std_logic_vector(1 downto 0):="00";
signal    mo32, mo33:std_logic:='0';
signal    mo34, mo35, mo36, mo37, mo38, mo39:std_logic_vector(4 downto 0):="00000";
signal    mo40, mo41, mo42, mo43, mo44, mo45:std_logic_vector(2 downto 0):="000";
signal    mo46, mo47:std_logic_vector(3 downto 0):="0000";
signal    mo48, mo49:std_logic_vector(2 downto 0):="000";
signal    mo50, mo51, mo52, mo53, mo54, mo55:std_logic:='0';
signal    mo56:integer:=0;
signal    mo57:std_logic:='0';
Begin

          -----component instantiation-----
          U1:main_control port map(ci0, ci1, ci2, ci3, ci4, ci5, ci6, ci7,
                                   co0, co1, co2, co3, co4, co5, co6, co7);
          U2:header port map(hi0, hi1, hi2, hi3,
                             ho0, ho1, ho2, ho3, ho4, ho5, ho6, ho7, ho8,
                             ho9, ho10, ho11, ho12, ho13, ho14, ho15);
          U3:audio_data port map(ai0, ai1, ai2, ai3, ai4, ai5, ai6,
                             ao0, ao1, ao2, ao3, ao4, ao5, ao6, ao7, ao8,
                             ao9, ao10, ao11, ao12, ao13, ao14, ao15, ao16,
                             ao17, ao18, ao19, ao20, ao21, ao22, ao23, ao24,
                             ao25, ao26, ao27, ao28, ao29, ao30, ao31, ao32,
                             ao33, ao34, ao35, ao36, ao37, ao38, ao39, ao40,
                             ao41, ao42, ao43);
          U4:waveform_10K port map(ri0, ro0);
```

```
-----data link from component to component-----
---control component input---
ci0 <= ro0;-----data from rom
ci1 <= mi0;          -----clock
ci2 <= mi1;          -----system enable
ci3 <= mi2;          -----system reset
ci4 <= ho7;          -----header finish
ci5 <= ao43;         -----audio data finish
ci6 <= ho15;         -----data count from header
ci7 <= ao42;         -----data count from audio data
---header component input---
hi0 <= co0;          -----data from control(from rom)
hi1 <= co1;          -----enable from control
hi2 <= co3;          -----reset from control
hi3 <= mi0;          -----clock
---audio data component input---
ai0 <= co0;          -----data from control(from rom)
ai1 <= co2;          -----enable from control
ai2 <= co4;          -----reset from control
ai3 <= mi0;          -----clock
ai4 <= ho10;         -----mode from header
ai5 <= ho8;          -----layer from header
ai6 <= co7;          -----bit count from control(start bit)
---rom component input---
ri0 <= co6;-----address from control
---system input---
mi0 <= clk;          -----system clock
mi1 <= en;-----system enable
mi2 <= reset;        -----system reset
---system output signal link---
mo0 <= ho0;          -----sync
mo1 <= ho1;          -----id
mo2 <= ho2;          -----protection bit
mo3 <= ho3;          -----padding bit
mo4 <= ho4;          -----private bit
mo5 <= ho5;          -----copyright
mo6 <= ho6;          -----original
mo7 <= ho8;          -----layer
mo8 <= ho9;          -----sampling frequency
mo9 <= ho10;         -----mode
```

```
mo10 <= ho11;      -----mode extension
mo11 <= ho12;      -----emphasis
mo12 <= ho13;      -----bitrate index
mo13 <= ho14;      -----crc
mo14 <= ao0;       -----main data begin
mo15 <= ao1;       -----private bits
mo16 <= ao2;       -----scfsi band0
mo17 <= ao3;       -----scfsi band1
mo18 <= ao4;       -----scfsi band2
mo19 <= ao5;       -----scfsi band3
mo20 <= ao6;       -----part2 3 length gr0
mo21 <= ao7;       -----part2 3 length gr1
mo22 <= ao8;       -----big values gr0
mo23 <= ao9;       -----big values gr1
mo24 <= ao10;      -----global gain gr0
mo25 <= ao11;      -----global gain gr1
mo26 <= ao12;      -----scalefac compress gr0
mo27 <= ao13;      -----scalefac compress gr1
mo28 <= ao14;      -----window switching flag gr0
mo29 <= ao15;      -----window switching flag gr1
mo30 <= ao16;      -----block type gr0
mo31 <= ao17;      -----block type gr1
mo32 <= ao18;      -----mixed block flag gr0
mo33 <= ao19;      -----mixed block flag gr1
mo34 <= ao20;      -----table select gr0 region0
mo35 <= ao21;      -----table select gr0 region1
mo36 <= ao22;      -----table select gr0 region2
mo37 <= ao23;      -----table select gr1 region0
mo38 <= ao24;      -----table select gr1 region1
mo39 <= ao25;      -----table select gr1 region2
mo40 <= ao26;      -----subblock gain gr0 window0
mo41 <= ao27;      -----subblock gain gr0 window1
mo42 <= ao28;      -----subblock gain gr0 window2
mo43 <= ao29;      -----subblock gain gr1 window0
mo44 <= ao30;      -----subblock gain gr1 window1
mo45 <= ao31;      -----subblock gain gr1 window2
mo46 <= ao32;      -----region0 count gr0
mo47 <= ao33;      -----region0 count gr1
mo48 <= ao34;      -----region1 count gr0
mo49 <= ao35;      -----region1 count gr1
```

```
mo50 <= ao36;          -----preflag gr0
mo51 <= ao37;          -----preflag gr1
mo52 <= ao38;          -----scalefac compress gr0
mo53 <= ao39;          -----scalefac compress gr1
mo54 <= ao40;          -----count1table select gr0
mo55 <= ao41;          -----count1table select gr1
mo56 <= co6;           -----bit count
mo57 <= co5;           -----finish
---system output port link---
sync <= mo0;
id <= mo1;
protection_bit <= mo2;
padding_bit <= mo3;
private_bit <= mo4;
copyright <= mo5;
original <= mo6;
layer <= mo7;
sampling_freq <= mo8;
mode <= mo9;
mode_ext <= mo10;
emphasis <= mo11;
bitrate_index <= mo12;
crc <= mo13;
main_data_begin <= mo14;
private_bits <= mo15;
scfsi_band0 <= mo16;
scfsi_band1 <= mo17;
scfsi_band2 <= mo18;
scfsi_band3 <= mo19;
part2_3_length_gr0 <= mo20;
part2_3_length_gr1 <= mo21;
big_values_gr0 <= mo22;
big_values_gr1 <= mo23;
global_gain_gr0 <= mo24;
global_gain_gr1 <= mo25;
scalefac_compress_gr0 <= mo26;
scalefac_compress_gr1 <= mo27;
window_switching_flag_gr0 <= mo28;
window_switching_flag_gr1 <= mo29;
block_type_gr0 <= mo30;
```

```
        block_type_gr1 <= mo31;
        mixed_block_flag_gr0 <= mo32;
        mixed_block_flag_gr1 <= mo33;
        table_select_gr0_region0 <= mo34;
        table_select_gr0_region1 <= mo35;
        table_select_gr0_region2 <= mo36;
        table_select_gr1_region0 <= mo37;
        table_select_gr1_region1 <= mo38;
        table_select_gr1_region2 <= mo39;
        subblock_gain_gr0_window0 <= mo40;
        subblock_gain_gr0_window1 <= mo41;
        subblock_gain_gr0_window2 <= mo42;
        subblock_gain_gr1_window0 <= mo43;
        subblock_gain_gr1_window1 <= mo44;
        subblock_gain_gr1_window2 <= mo45;
        region0_count_gr0 <= mo46;
        region0_count_gr1 <= mo47;
        region1_count_gr0 <= mo48;
        region1_count_gr1 <= mo49;
        preflag_gr0 <= mo50;
        preflag_gr1 <= mo51;
        scalefac_scale_gr0 <= mo52;
        scalefac_scale_gr1 <= mo53;
        count1table_select_gr0 <= mo54;
        count1table_select_gr1 <= mo55;
        bitcount <= mo56;
        finish <= mo57;
end;


library IEEE;
use IEEE.Std_Logic_1164.all;
use ieee.std_logic_unsigned.all;
Entity main_control is


port    (din, clk, en, reset, header_finish_in, audio_data_finish_in:in std_logic;
         header_count, audio_data_count:in integer;
         dout, header_en, audio_data_en, header_reset, audio_data_reset, finish:out std_logic:='0';
```

```
        addr_out, bit_count:out integer:=0);
end;
Architecture behv of main_control is
signal header_en_bit, audio_data_en_bit, header_reset_bit, audio_data_reset_bit:std_logic:='0';
signal addr_data, bit_count_data:integer:=0;
signal job_finish, header_finish, audio_data_finish:std_logic:='0';
Begin
Process
Begin
        wait until clk'event and clk = '1';
        if reset = '1' then
                header_en_bit <= '0';
                audio_data_en_bit <= '0';
                header_reset_bit <= '1',
                                '0' after 20 ns;
                audio_data_reset_bit <= '1',
                                        '0' after 20 ns;
                addr_data <= 0;
                job_finish <= '0';
        else
                if en = '1' and job_finish = '0' then
                        wait until clk'event and clk = '1';
                        -----reset system at start first-----
                        header_en_bit <= '0';
                        audio_data_en_bit <= '0';
                        header_reset_bit <= '1',
                                                '0' after 20 ns;
                        audio_data_reset_bit <= '1',
                                                        '0' after 20 ns;
                        addr_data <= 0;
                        job_finish <= '0';
                        wait until clk'event and clk = '1';
                        -----start working job by header-----
                        header_wait:loop
                                wait until clk'event and clk ='1';
                                if header_finish = '1' then
                                        exit header_wait;
                                else
                                        header_en_bit <= '1';
                                        audio_data_en_bit <= '0';
```

```
                                        addr_data <= header_count;
                                end if;
                        end loop;
                        wait until clk'event and clk = '1';
                        -----next working job by audio_data-----
                        audio_data_wait:loop
                                wait until clk'event and clk = '1';
                                if audio_data_finish = '1' then
                                        job_finish <= '1';
                                        exit audio_data_wait;
                                else
                                        header_en_bit <= '0';
                                        audio_data_en_bit <= '1';
                                        addr_data <= audio_data_count;
                                end if;
                        end loop;
                else
                        header_en_bit <= header_en_bit;
                        audio_data_en_bit <= audio_data_en_bit;
                        header_reset_bit <= header_reset_bit;
                        audio_data_reset_bit <= audio_data_reset_bit;
                        addr_data <= addr_data;
                        job_finish <= job_finish;
                end if;
        end if;
end process;
        header_en <= header_en_bit;
        audio_data_en <= audio_data_en_bit;
        header_reset <= header_reset_bit;
        audio_data_reset <= audio_data_reset_bit;
        addr_out <= addr_data;
        bit_count <= bit_count_data;
        bit_count_data <= header_count;
        header_finish <= header_finish_in;
        audio_data_finish <= audio_data_finish_in;
        dout <= din;
        finish <= job_finish;
end;

library IEEE;
```

```
use IEEE.Std_Logic_1164.all;
use ieee.std_logic_unsigned.all;
Entity header is
port        (din, en, reset, clk:in std_logic;
             sync:out std_logic_vector(11 downto 0):="000000000000";
             id, protection_bit, padding_bit, private_bit,
             copyright, original, finish_out:out std_logic:='0';
             layer, sampling_freq, mode, mode_ext,
             emphasis:out std_logic_vector(1 downto 0):="00";
             bitrate_index:out std_logic_vector(3 downto 0):="0000";
             crc:out std_logic_vector(15 downto 0):="0000000000000000";
             data_count:out integer:=0);
end;
Architecture behv of header is
Signal      sync_data:std_logic_vector(11 downto 0):="000000000000";
Signal      id_data, protection_bit_data, padding_bit_data, private_bit_data,
            copyright_data, original_data, finish:std_logic:='0';
Signal      layer_data, sampling_freq_data, mode_data, mode_ext_data,
            emphasis_data:std_logic_vector(1 downto 0):="00";
Signal      bitrate_index_data:std_logic_vector(3 downto 0):="0000";
Signal      crc_data:std_logic_vector(15 downto 0):="0000000000000000";
Signal      din_count:integer:=0;
Begin
Process
Begin
            wait until clk'event and clk='1';
            if reset='1' then
                        din_count <= 0;
                        sync_data <= "000000000000";
                        id_data <= '0' ;
                        layer_data <= "00";
                        protection_bit_data <= '0';
                        bitrate_index_data <= "0000";
                        sampling_freq_data <= "00";
                        padding_bit_data <= '0';
                        private_bit_data <= '0';
                        mode_data <= "00";
                        mode_ext_data <= "00";
                        copyright_data <= '0';
                        original_data <= '0';
```

```
                emphasis_data <= "00";
                crc_data <= "0000000000000000";
                finish <= '0';
        else if en='1' and finish = '0' then
                -----get syncword-----
                wait until clk'event and clk='1';
                get_sync:loop
                        wait until clk'event and clk='1';
                exit get_sync when sync_data="111111111111";
                        wait until clk'event and clk='1';
                        sync_data <= shl(sync_data,"1");
                        sync_data(0) <= din after 10 ns;
                        din_count <= din_count+1 after 20 ns;
                end loop;
                -----get id-----
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                id_data <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                -----get layer-----
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                layer_data <= shl(layer_data,"1");
                layer_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                layer_data <= shl(layer_data,"1");
                layer_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                wait until clk'event and clk = '1';
                wait until clk'event and clk='1';
                -----check layer-----
                if layer_data="01" then
                -----get protection bit-----
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                protection_bit_data <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                -----get bitrate index-----
```

```
wait until clk'event and clk='1';
wait until clk'event and clk='1';
bitrate_index_data <= shl(bitrate_index_data,"1");
bitrate_index_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
bitrate_index_data <= shl(bitrate_index_data,"1");
bitrate_index_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
bitrate_index_data <= shl(bitrate_index_data,"1");
bitrate_index_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
bitrate_index_data <= shl(bitrate_index_data,"1");
bitrate_index_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get sampling frequency-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
sampling_freq_data <= shl(sampling_freq_data,"1");
sampling_freq_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
sampling_freq_data <= shl(sampling_freq_data,"1");
sampling_freq_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get padding bit-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
padding_bit_data <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get private bit-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
private_bit_data <= din after 10 ns;
```

```
din_count <= din_count+1 after 20 ns;
-----get mode-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
mode_data <= shl(mode_data,"1");
mode_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
mode_data <= shl(mode_data,"1");
mode_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get mode extension-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
mode_ext_data <= shl(mode_ext_data,"1");
mode_ext_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
mode_ext_data <= shl(mode_ext_data,"1");
mode_ext_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get copyright-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
copyright_data <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get original/copy-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
original_data <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get emphasis-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
emphasis_data <= shl(emphasis_data,"1");
emphasis_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
```

```
        wait until clk'event and clk='1';
        emphasis_data <= shl(emphasis_data,"1");
        emphasis_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        -----check for protection bit to get crc-----
        wait until clk'event and clk='1';
        if protection_bit_data='1' then
                for i in 0 to 15 loop
                        wait until clk'event and clk='1';
                        wait until clk'event and clk='1';
                        crc_data <= shl(crc_data,"1");
                        crc_data(0) <= din after 10 ns;
                        din_count <= din_count+1 after 20 ns;
                end loop;
        end if;
        finish <= '1';
        else
        protection_bit_data <= '0';
        bitrate_index_data <= "0000";
        sampling_freq_data <= "00";
        padding_bit_data <= '0';
        private_bit_data <= '0';
        mode_data <= "00";
        mode_ext_data <= "00";
        copyright_data <= '0';
        original_data <= '0';
        emphasis_data <= "00";
        crc_data <= "0000000000000000";
        finish <= '1';
        end if;
else
  protection_bit_data <= protection_bit_data;
        bitrate_index_data <= bitrate_index_data;
        sampling_freq_data <= sampling_freq_data;
        padding_bit_data <= padding_bit_data;
        private_bit_data <= private_bit_data;
        mode_data <= mode_data;
        mode_ext_data <= mode_ext_data;
        copyright_data <= copyright_data;
        original_data <= original_data;
```

```
                    emphasis_data <= emphasis_data;
                    crc_data <= crc_data;
                    din_count <= din_count;
                    finish <= finish;
                end if;
            end if;
end process;
            sync <= sync_data;
            id <= id_data;
            layer <= layer_data;
            protection_bit <= protection_bit_data;
            bitrate_index <= bitrate_index_data;
            sampling_freq <= sampling_freq_data;
            padding_bit <= padding_bit_data;
            private_bit <= private_bit_data;
            mode <= mode_data;
            mode_ext <= mode_ext_data;
            copyright <= copyright_data;
            original <= original_data;
            emphasis <= emphasis_data;
            crc <= crc_data;
            data_count <= din_count;
            finish_out <= finish;
end;

library IEEE;
use IEEE.Std_Logic_1164.all;
use ieee.std_logic_unsigned.all;
Entity audio_data is
port        (din, en, reset, clk:in std_logic;
             mode_in, layer_in:in std_logic_vector(1 downto 0);
             count_in:in integer;
            main_data_begin:out std_logic_vector(8 downto 0):="000000000";
            private_bits:out std_logic_vector(4 downto 0):="00000";
            scfsi_band0, scfsi_band1, scfsi_band2, scfsi_band3:out std_logic:='0';
            part2_3_length_gr0,   part2_3_length_gr1:out std_logic_vector(11 downto 0):="000000000000";
            big_values_gr0, big_values_gr1:out std_logic_vector(8 downto 0):="000000000";
            global_gain_gr0, global_gain_gr1:out std_logic_vector(7 downto 0):="00000000";
            scalefac_compress_gr0, scalefac_compress_gr1:out std_logic_vector(3 downto 0):="0000";
            window_switching_flag_gr0, window_switching_flag_gr1:out std_logic:='0';
```

```
        block_type_gr0, block_type_gr1:out std_logic_vector(1 downto 0):="00";
        mixed_block_flag_gr0, mixed_block_flag_gr1:out std_logic:='0';
        table_select_gr0_region0, table_select_gr0_region1, table_select_gr0_region2,
        table_select_gr1_region0, table_select_gr1_region1, table_select_gr1_region2
        :out std_logic_vector(4 downto 0):="00000";
        subblock_gain_gr0_window0, subblock_gain_gr0_window1, subblock_gain_gr0_window2,
        subblock_gain_gr1_window0, subblock_gain_gr1_window1, subblock_gain_gr1_window2
        :out std_logic_vector(2 downto 0):="000";
        region0_count_gr0, region0_count_gr1:out std_logic_vector(3 downto 0):="0000";
        region1_count_gr0, region1_count_gr1:out std_logic_vector(2 downto 0):="000";
        preflag_gr0, preflag_gr1:out std_logic:='0';
        scalefac_scale_gr0, scalefac_scale_gr1:out std_logic:='0';
        count1table_select_gr0, count1table_select_gr1:out std_logic:='0';
        data_count:out integer:=0;
        finish_out:out std_logic:='0');
end;
Architecture behv of audio_data is
Signal  main_data_begin_data:std_logic_vector(8 downto 0):="000000000";
Signal  private_bits_data:std_logic_vector(4 downto 0):="00000";
Signal  scfsi_band0_data, scfsi_band1_data, scfsi_band2_data, scfsi_band3_data:std_logic:='0';
Signal  part2_3_length_gr0_data, part2_3_length_gr1_data:std_logic_vector(11 downto 0):="000000000000";
Signal  big_values_gr0_data, big_values_gr1_data:std_logic_vector(8 downto 0):="000000000";
Signal  global_gain_gr0_data, global_gain_gr1_data:std_logic_vector(7 downto 0):="00000000";
Signal  scalefac_compress_gr0_data, scalefac_compress_gr1_data:std_logic_vector(3 downto 0):="0000";
Signal  window_switching_flag_gr0_data, window_switching_flag_gr1_data:std_logic:='0';
Signal  block_type_gr0_data, block_type_gr1_data:std_logic_vector(1 downto 0):="00";
Signal  mixed_block_flag_gr0_data, mixed_block_flag_gr1_data:std_logic:='0';
Signal  table_select_gr0_region0_data, table_select_gr0_region1_data, table_select_gr0_region2_data,
        table_select_gr1_region0_data, table_select_gr1_region1_data, table_select_gr1_region2_data
        :std_logic_vector(4 downto 0):="00000";
Signal  subblock_gain_gr0_window0_data, subblock_gain_gr0_window1_data, subblock_gain_gr0_window2_data,
        subblock_gain_gr1_window0_data, subblock_gain_gr1_window1_data, subblock_gain_gr1_window2_data
        :std_logic_vector(2 downto 0):="000";
Signal  region0_count_gr0_data, region0_count_gr1_data:std_logic_vector(3 downto 0):="0000";
Signal  region1_count_gr0_data, region1_count_gr1_data:std_logic_vector(2 downto 0):="000";
Signal  preflag_gr0_data, preflag_gr1_data:std_logic:='0';
Signal  scalefac_scale_gr0_data, scalefac_scale_gr1_data:std_logic:='0';
Signal  count1table_select_gr0_data, count1table_select_gr1_data:std_logic:='0';
Signal  din_count, ch:integer:=0;
Signal  mode, layer:std_logic_vector(1 downto 0):="00";
```

```
Signal  finish:std_logic:='0';
Begin
Process
Begin
        wait until clk'event and clk='1';
        if reset='1' then
                main_data_begin_data <= "000000000";
                private_bits_data <= "00000";
                scfsi_band0_data <= '0';
                scfsi_band1_data <= '0';
                scfsi_band2_data <= '0';
                scfsi_band3_data <= '0';
                part2_3_length_gr0_data <= "000000000000";
                part2_3_length_gr1_data <= "000000000000";
                big_values_gr0_data <= "000000000";
                big_values_gr1_data <= "000000000";
                global_gain_gr0_data <= "00000000";
                global_gain_gr1_data <= "00000000";
                scalefac_compress_gr0_data <= "0000";
                scalefac_compress_gr1_data <= "0000";
                window_switching_flag_gr0_data <= '0';
                window_switching_flag_gr1_data <= '0';
                block_type_gr0_data <= "00";
                block_type_gr1_data <= "00";
                mixed_block_flag_gr0_data <= '0';
                mixed_block_flag_gr1_data <= '0';
                table_select_gr0_region0_data <= "00000";
                table_select_gr0_region1_data <= "00000";
                table_select_gr0_region2_data <= "00000";
                table_select_gr1_region0_data <= "00000";
                table_select_gr1_region1_data <= "00000";
                table_select_gr1_region2_data <= "00000";
                subblock_gain_gr0_window0_data <= "000";
                subblock_gain_gr0_window1_data <= "000";
                subblock_gain_gr0_window2_data <= "000";
                subblock_gain_gr1_window0_data <= "000";
                subblock_gain_gr1_window1_data <= "000";
                subblock_gain_gr1_window2_data <= "000";
                region0_count_gr0_data <= "0000";
                region0_count_gr1_data <= "0000";
```

```
                region1_count_gr0_data <= "000";
                region1_count_gr1_data <= "000";
                preflag_gr0_data <= '0';
                preflag_gr1_data <= '0';
                scalefac_scale_gr0_data <= '0';
                scalefac_scale_gr1_data <= '0';
                count1table_select_gr0_data <= '0';
                count1table_select_gr1_data <= '0';
                finish <= '0';
                din_count <= 0;
else if en='1' and finish = '0' then
    if layer = "01" then
                din_count <= count_in;
                -----get main data begin-----
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                main_data_begin_data <= shl(main_data_begin_data,"1");
                main_data_begin_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                main_data_begin_data <= shl(main_data_begin_data,"1");
                main_data_begin_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                main_data_begin_data <= shl(main_data_begin_data,"1");
                main_data_begin_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                main_data_begin_data <= shl(main_data_begin_data,"1");
                main_data_begin_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                main_data_begin_data <= shl(main_data_begin_data,"1");
                main_data_begin_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                wait until clk'event and clk='1';
```

```
wait until clk'event and clk='1';
main_data_begin_data <= shl(main_data_begin_data,"1");
main_data_begin_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
main_data_begin_data <= shl(main_data_begin_data,"1");
main_data_begin_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
main_data_begin_data <= shl(main_data_begin_data,"1");
main_data_begin_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
main_data_begin_data <= shl(main_data_begin_data,"1");
main_data_begin_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
-----check mode-----
case mode is
        when "11" => ch <= 0;
        when "10" => ch <= 1;
        when "00" => ch <= 1;
        when others => ch <= 3;
end case;
wait until clk'event and clk='1';
-----get private bits-----
if mode="11" then
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        private_bits_data <= shl(private_bits_data,"1");
        private_bits_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        private_bits_data <= shl(private_bits_data,"1");
        private_bits_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
```

```
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                private_bits_data <= shl(private_bits_data,"1");
                private_bits_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                private_bits_data <= shl(private_bits_data,"1");
                private_bits_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                private_bits_data <= shl(private_bits_data,"1");
                private_bits_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
        else
                wait until clk'event and clk='1';
                private_bits_data <= "00000";
        end if;
        wait until clk'event and clk='1';
        -----get scfsi-----
        case ch is
                when 0 =>
                        wait until clk'event and clk='1';
                        wait until clk'event and clk='1';
                        scfsi_band0_data <= din after 10 ns;
                        din_count <= din_count+1 after 20 ns;
                        wait until clk'event and clk='1';
                        wait until clk'event and clk='1';
                        scfsi_band1_data <= din after 10 ns;
                        din_count <= din_count+1 after 20 ns;
                        wait until clk'event and clk='1';
                        wait until clk'event and clk='1';
                        scfsi_band2_data <= din after 10 ns;
                        din_count <= din_count+1 after 20 ns;
                        wait until clk'event and clk='1';
                        wait until clk'event and clk='1';
                        scfsi_band3_data <= din after 10 ns;
                        din_count <= din_count+1 after 20 ns;
                when others =>
```

```
                    wait until clk'event and clk='1';
                    scfsi_band0_data <= '0';
                    scfsi_band1_data <= '0';
                    scfsi_band2_data <= '0';
                    scfsi_band3_data <= '0';
end case;
wait until clk'event and clk='1';
case ch is
when 0 =>
          -----get audio data for granule 0-----
          -----get part2 3 length-----
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
          part2_3_length_gr0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
          part2_3_length_gr0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
          part2_3_length_gr0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
          part2_3_length_gr0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
          part2_3_length_gr0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
          part2_3_length_gr0_data(0) <= din after 10 ns;
```

```
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
part2_3_length_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
part2_3_length_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
part2_3_length_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
part2_3_length_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
part2_3_length_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr0_data <= shl(part2_3_length_gr0_data,"1");
part2_3_length_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get big values-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr0_data <= shl(big_values_gr0_data,"1");
big_values_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr0_data <= shl(big_values_gr0_data,"1");
```

```
big_values_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr0_data <= shl(big_values_gr0_data,"1");
big_values_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr0_data <= shl(big_values_gr0_data,"1");
big_values_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr0_data <= shl(big_values_gr0_data,"1");
big_values_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr0_data <= shl(big_values_gr0_data,"1");
big_values_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr0_data <= shl(big_values_gr0_data,"1");
big_values_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr0_data <= shl(big_values_gr0_data,"1");
big_values_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get global gain-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
```

```
global_gain_gr0_data <= shl(global_gain_gr0_data,"1");
global_gain_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr0_data <= shl(global_gain_gr0_data,"1");
global_gain_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr0_data <= shl(global_gain_gr0_data,"1");
global_gain_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr0_data <= shl(global_gain_gr0_data,"1");
global_gain_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr0_data <= shl(global_gain_gr0_data,"1");
global_gain_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr0_data <= shl(global_gain_gr0_data,"1");
global_gain_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr0_data <= shl(global_gain_gr0_data,"1");
global_gain_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr0_data <= shl(global_gain_gr0_data,"1");
global_gain_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get scalefac compress-----
wait until clk'event and clk='1';
```

```
wait until clk'event and clk='1';
scalefac_compress_gr0_data <= shl(scalefac_compress_gr0_data,"1");
scalefac_compress_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
scalefac_compress_gr0_data <= shl(scalefac_compress_gr0_data,"1");
scalefac_compress_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
scalefac_compress_gr0_data <= shl(scalefac_compress_gr0_data,"1");
scalefac_compress_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
scalefac_compress_gr0_data <= shl(scalefac_compress_gr0_data,"1");
scalefac_compress_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get window switching flag-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
window_switching_flag_gr0_data <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----check window-----
wait until clk'event and clk='1';
if window_switching_flag_gr0_data='1' then
        -----get block type-----
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        block_type_gr0_data <= shl(block_type_gr0_data,"1");
        block_type_gr0_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        block_type_gr0_data <= shl(block_type_gr0_data,"1");
        block_type_gr0_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        -----get mixed block flag-----
        wait until clk'event and clk='1';
```

```
wait until clk'event and clk='1';
mixed_block_flag_gr0_data <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get table select-----
---***region0***---
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region0_data <= shl(table_select_gr0_region0_data,"1");
table_select_gr0_region0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region0_data <= shl(table_select_gr0_region0_data,"1");
table_select_gr0_region0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region0_data <= shl(table_select_gr0_region0_data,"1");
table_select_gr0_region0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region0_data <= shl(table_select_gr0_region0_data,"1");
table_select_gr0_region0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region0_data <= shl(table_select_gr0_region0_data,"1");
table_select_gr0_region0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
---***region1***---
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region1_data <= shl(table_select_gr0_region1_data,"1");
table_select_gr0_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region1_data <= shl(table_select_gr0_region1_data,"1");
table_select_gr0_region1_data(0) <= din after 10 ns;
```

```
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region1_data <= shl(table_select_gr0_region1_data,"1");
table_select_gr0_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region1_data <= shl(table_select_gr0_region1_data,"1");
table_select_gr0_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region1_data <= shl(table_select_gr0_region1_data,"1");
table_select_gr0_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
---***region2***---
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region2_data <= shl(table_select_gr0_region2_data,"1");
table_select_gr0_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region2_data <= shl(table_select_gr0_region2_data,"1");
table_select_gr0_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region2_data <= shl(table_select_gr0_region2_data,"1");
table_select_gr0_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region2_data <= shl(table_select_gr0_region2_data,"1");
table_select_gr0_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region2_data <= shl(table_select_gr0_region2_data,"1");
```

```
          table_select_gr0_region2_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          -----get subblock gain-----
          ---***window0***---
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr0_window0_data <= shl(subblock_gain_gr0_window0_data,"1");
          subblock_gain_gr0_window0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr0_window0_data <= shl(subblock_gain_gr0_window0_data,"1");
          subblock_gain_gr0_window0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr0_window0_data <= shl(subblock_gain_gr0_window0_data,"1");
          subblock_gain_gr0_window0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          ---***window1***---
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr0_window1_data <= shl(subblock_gain_gr0_window1_data,"1");
          subblock_gain_gr0_window1_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr0_window1_data <= shl(subblock_gain_gr0_window1_data,"1");
          subblock_gain_gr0_window1_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr0_window1_data <= shl(subblock_gain_gr0_window1_data,"1");
          subblock_gain_gr0_window1_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          ---***window2***---
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr0_window2_data <= shl(subblock_gain_gr0_window2_data,"1");
          subblock_gain_gr0_window2_data(0) <= din after 10 ns;
```

```
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr0_window2_data <= shl(subblock_gain_gr0_window2_data,"1");
          subblock_gain_gr0_window2_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr0_window2_data <= shl(subblock_gain_gr0_window2_data,"1");
          subblock_gain_gr0_window2_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
else
          -----get table select-----
          ---***region0***---
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          table_select_gr0_region0_data <= shl(table_select_gr0_region0_data,"1");
          table_select_gr0_region0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          table_select_gr0_region0_data <= shl(table_select_gr0_region0_data,"1");
          table_select_gr0_region0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          table_select_gr0_region0_data <= shl(table_select_gr0_region0_data,"1");
          table_select_gr0_region0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          table_select_gr0_region0_data <= shl(table_select_gr0_region0_data,"1");
          table_select_gr0_region0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          table_select_gr0_region0_data <= shl(table_select_gr0_region0_data,"1");
          table_select_gr0_region0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          ---***region1***---
```

```
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region1_data <= shl(table_select_gr0_region1_data,"1");
table_select_gr0_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region1_data <= shl(table_select_gr0_region1_data,"1");
table_select_gr0_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region1_data <= shl(table_select_gr0_region1_data,"1");
table_select_gr0_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region1_data <= shl(table_select_gr0_region1_data,"1");
table_select_gr0_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region1_data <= shl(table_select_gr0_region1_data,"1");
table_select_gr0_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
---***region2***---
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region2_data <= shl(table_select_gr0_region2_data,"1");
table_select_gr0_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region2_data <= shl(table_select_gr0_region2_data,"1");
table_select_gr0_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region2_data <= shl(table_select_gr0_region2_data,"1");
table_select_gr0_region2_data(0) <= din after 10 ns;
```

```
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region2_data <= shl(table_select_gr0_region2_data,"1");
table_select_gr0_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr0_region2_data <= shl(table_select_gr0_region2_data,"1");
table_select_gr0_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get region0 count-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
region0_count_gr0_data <= shl(region0_count_gr0_data,"1");
region0_count_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
region0_count_gr0_data <= shl(region0_count_gr0_data,"1");
region0_count_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
region0_count_gr0_data <= shl(region0_count_gr0_data,"1");
region0_count_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
region0_count_gr0_data <= shl(region0_count_gr0_data,"1");
region0_count_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get region1 count-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
region1_count_gr0_data <= shl(region1_count_gr0_data,"1");
region1_count_gr0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
```

```
                region1_count_gr0_data <= shl(region1_count_gr0_data,"1");
                region1_count_gr0_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
                wait until clk'event and clk='1';
                wait until clk'event and clk='1';
                region1_count_gr0_data <= shl(region1_count_gr0_data,"1");
                region1_count_gr0_data(0) <= din after 10 ns;
                din_count <= din_count+1 after 20 ns;
        end if;
        -----get preflag-----
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        preflag_gr0_data <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        -----get scalefac scale-----
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        scalefac_scale_gr0_data <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        -----get count1 table select-----
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        count1table_select_gr0_data <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        -----get data for granule 1-----
        -----get part2 3 length-----
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
        part2_3_length_gr1_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
        part2_3_length_gr1_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
        part2_3_length_gr1_data(0) <= din after 10 ns;
```

```
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
part2_3_length_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
part2_3_length_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
part2_3_length_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
part2_3_length_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
part2_3_length_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
part2_3_length_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
part2_3_length_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
part2_3_length_gr1_data(0) <= din after 10 ns;
```

```
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
part2_3_length_gr1_data <= shl(part2_3_length_gr1_data,"1");
part2_3_length_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get big values-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr1_data <= shl(big_values_gr1_data,"1");
big_values_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr1_data <= shl(big_values_gr1_data,"1");
big_values_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr1_data <= shl(big_values_gr1_data,"1");
big_values_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr1_data <= shl(big_values_gr1_data,"1");
big_values_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr1_data <= shl(big_values_gr1_data,"1");
big_values_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr1_data <= shl(big_values_gr1_data,"1");
big_values_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr1_data <= shl(big_values_gr1_data,"1");
```

```
big_values_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr1_data <= shl(big_values_gr1_data,"1");
big_values_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
big_values_gr1_data <= shl(big_values_gr1_data,"1");
big_values_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get global gain-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr1_data <= shl(global_gain_gr1_data,"1");
global_gain_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr1_data <= shl(global_gain_gr1_data,"1");
global_gain_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr1_data <= shl(global_gain_gr1_data,"1");
global_gain_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr1_data <= shl(global_gain_gr1_data,"1");
global_gain_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr1_data <= shl(global_gain_gr1_data,"1");
global_gain_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
```

```
global_gain_gr1_data <= shl(global_gain_gr1_data,"1");
global_gain_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr1_data <= shl(global_gain_gr1_data,"1");
global_gain_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
global_gain_gr1_data <= shl(global_gain_gr1_data,"1");
global_gain_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get scalefac compress-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
scalefac_compress_gr1_data <= shl(scalefac_compress_gr1_data,"1");
scalefac_compress_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
scalefac_compress_gr1_data <= shl(scalefac_compress_gr1_data,"1");
scalefac_compress_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
scalefac_compress_gr1_data <= shl(scalefac_compress_gr1_data,"1");
scalefac_compress_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
scalefac_compress_gr1_data <= shl(scalefac_compress_gr1_data,"1");
scalefac_compress_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get window switching flag-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
window_switching_flag_gr1_data <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----check window-----
```

```
wait until clk'event and clk='1';
if window_switching_flag_gr1_data='1' then
        -----get block type-----
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        block_type_gr1_data <= shl(block_type_gr1_data,"1");
        block_type_gr1_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        block_type_gr1_data <= shl(block_type_gr1_data,"1");
        block_type_gr1_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        -----get mixed block flag-----
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        mixed_block_flag_gr1_data <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        -----get table select-----
        ---***region0***---
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        table_select_gr1_region0_data <= shl(table_select_gr1_region0_data,"1");
        table_select_gr1_region0_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        table_select_gr1_region0_data <= shl(table_select_gr1_region0_data,"1");
        table_select_gr1_region0_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        table_select_gr1_region0_data <= shl(table_select_gr1_region0_data,"1");
        table_select_gr1_region0_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        table_select_gr1_region0_data <= shl(table_select_gr1_region0_data,"1");
        table_select_gr1_region0_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
```

```
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region0_data <= shl(table_select_gr1_region0_data,"1");
table_select_gr1_region0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
---***region1***---
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region1_data <= shl(table_select_gr1_region1_data,"1");
table_select_gr1_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region1_data <= shl(table_select_gr1_region1_data,"1");
table_select_gr1_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region1_data <= shl(table_select_gr1_region1_data,"1");
table_select_gr1_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region1_data <= shl(table_select_gr1_region1_data,"1");
table_select_gr1_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region1_data <= shl(table_select_gr1_region1_data,"1");
table_select_gr1_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
---***region2***---
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region2_data <= shl(table_select_gr1_region2_data,"1");
table_select_gr1_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region2_data <= shl(table_select_gr1_region2_data,"1");
```

```
          table_select_gr1_region2_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          table_select_gr1_region2_data <= shl(table_select_gr1_region2_data,"1");
          table_select_gr1_region2_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          table_select_gr1_region2_data <= shl(table_select_gr1_region2_data,"1");
          table_select_gr1_region2_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          table_select_gr1_region2_data <= shl(table_select_gr1_region2_data,"1");
          table_select_gr1_region2_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          -----get subblock gain-----
          ---***window0***---
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr1_window0_data <= shl(subblock_gain_gr1_window0_data,"1");
          subblock_gain_gr1_window0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr1_window0_data <= shl(subblock_gain_gr1_window0_data,"1");
          subblock_gain_gr1_window0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr1_window0_data <= shl(subblock_gain_gr1_window0_data,"1");
          subblock_gain_gr1_window0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          ---***window1***---
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr1_window1_data <= shl(subblock_gain_gr1_window1_data,"1");
          subblock_gain_gr1_window1_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
```

```
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr1_window1_data <= shl(subblock_gain_gr1_window1_data,"1");
          subblock_gain_gr1_window1_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr1_window1_data <= shl(subblock_gain_gr1_window1_data,"1");
          subblock_gain_gr1_window1_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          ---***window2***---
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr1_window2_data <= shl(subblock_gain_gr1_window2_data,"1");
          subblock_gain_gr1_window2_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr1_window2_data <= shl(subblock_gain_gr1_window2_data,"1");
          subblock_gain_gr1_window2_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
subblock_gain_gr1_window2_data <= shl(subblock_gain_gr1_window2_data,"1");
          subblock_gain_gr1_window2_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
else
          -----get table select-----
          ---***region0***---
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          table_select_gr1_region0_data <= shl(table_select_gr1_region0_data,"1");
          table_select_gr1_region0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
          wait until clk'event and clk='1';
          table_select_gr1_region0_data <= shl(table_select_gr1_region0_data,"1");
          table_select_gr1_region0_data(0) <= din after 10 ns;
          din_count <= din_count+1 after 20 ns;
          wait until clk'event and clk='1';
```

```
wait until clk'event and clk='1';
table_select_gr1_region0_data <= shl(table_select_gr1_region0_data,"1");
table_select_gr1_region0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region0_data <= shl(table_select_gr1_region0_data,"1");
table_select_gr1_region0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region0_data <= shl(table_select_gr1_region0_data,"1");
table_select_gr1_region0_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
---***region1***---
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region1_data <= shl(table_select_gr1_region1_data,"1");
table_select_gr1_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region1_data <= shl(table_select_gr1_region1_data,"1");
table_select_gr1_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region1_data <= shl(table_select_gr1_region1_data,"1");
table_select_gr1_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region1_data <= shl(table_select_gr1_region1_data,"1");
table_select_gr1_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region1_data <= shl(table_select_gr1_region1_data,"1");
table_select_gr1_region1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
```

```
---***region2***---
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region2_data <= shl(table_select_gr1_region2_data,"1");
table_select_gr1_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region2_data <= shl(table_select_gr1_region2_data,"1");
table_select_gr1_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region2_data <= shl(table_select_gr1_region2_data,"1");
table_select_gr1_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region2_data <= shl(table_select_gr1_region2_data,"1");
table_select_gr1_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
table_select_gr1_region2_data <= shl(table_select_gr1_region2_data,"1");
table_select_gr1_region2_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get region0 count-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
region0_count_gr1_data <= shl(region0_count_gr1_data,"1");
region0_count_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
region0_count_gr1_data <= shl(region0_count_gr1_data,"1");
region0_count_gr1_data(0) <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
wait until clk'event and clk='1';
wait until clk'event and clk='1';
region0_count_gr1_data <= shl(region0_count_gr1_data,"1");
```

```
        region0_count_gr1_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        region0_count_gr1_data <= shl(region0_count_gr1_data,"1");
        region0_count_gr1_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        -----get region1 count-----
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        region1_count_gr1_data <= shl(region1_count_gr1_data,"1");
        region1_count_gr1_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        region1_count_gr1_data <= shl(region1_count_gr1_data,"1");
        region1_count_gr1_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
        wait until clk'event and clk='1';
        wait until clk'event and clk='1';
        region1_count_gr1_data <= shl(region1_count_gr1_data,"1");
        region1_count_gr1_data(0) <= din after 10 ns;
        din_count <= din_count+1 after 20 ns;
end if;
-----get preflag-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
preflag_gr1_data <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get scalefac scale-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
scalefac_scale_gr1_data <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
-----get count1 table select-----
wait until clk'event and clk='1';
wait until clk'event and clk='1';
count1table_select_gr1_data <= din after 10 ns;
din_count <= din_count+1 after 20 ns;
finish <= '1' after 20 ns;
```

```
when others =>
        part2_3_length_gr0_data <= "000000000000";
        part2_3_length_gr1_data <= "000000000000";
        big_values_gr0_data <= "000000000";
        big_values_gr1_data <= "000000000";
        global_gain_gr0_data <= "00000000";
        global_gain_gr1_data <= "00000000";
        scalefac_compress_gr0_data <= "0000";
        scalefac_compress_gr1_data <= "0000";
        window_switching_flag_gr0_data <= '0';
        window_switching_flag_gr1_data <= '0';
        block_type_gr0_data <= "00";
        block_type_gr1_data <= "00";
        mixed_block_flag_gr0_data <= '0';
        mixed_block_flag_gr1_data <= '0';
        table_select_gr0_region0_data <= "00000";
        table_select_gr0_region1_data <= "00000";
        table_select_gr0_region2_data <= "00000";
        table_select_gr1_region0_data <= "00000";
        table_select_gr1_region1_data <= "00000";
        table_select_gr1_region2_data <= "00000";
        subblock_gain_gr0_window0_data <= "000";
        subblock_gain_gr0_window1_data <= "000";
        subblock_gain_gr0_window2_data <= "000";
        subblock_gain_gr1_window0_data <= "000";
        subblock_gain_gr1_window1_data <= "000";
        subblock_gain_gr1_window2_data <= "000";
        region0_count_gr0_data <= "0000";
        region0_count_gr1_data <= "0000";
        region1_count_gr0_data <= "000";
        region1_count_gr1_data <= "000";
        preflag_gr0_data <= '0';
        preflag_gr1_data <= '0';
        scalefac_scale_gr0_data <= '0';
        scalefac_scale_gr1_data <= '0';
        count1table_select_gr0_data <= '0';
        count1table_select_gr1_data <= '0';
        finish <= '1' after 20 ns;
end case;
else
```

```
main_data_begin_data <= "000000000";
private_bits_data <= "00000";
scfsi_band0_data <= '0';
scfsi_band1_data <= '0';
scfsi_band2_data <= '0';
scfsi_band3_data <= '0';
part2_3_length_gr0_data <= "000000000000";
part2_3_length_gr1_data <= "000000000000";
big_values_gr0_data <= "000000000";
big_values_gr1_data <= "000000000";
global_gain_gr0_data <= "00000000";
global_gain_gr1_data <= "00000000";
scalefac_compress_gr0_data <= "0000";
scalefac_compress_gr1_data <= "0000";
window_switching_flag_gr0_data <= '0';
window_switching_flag_gr1_data <= '0';
block_type_gr0_data <= "00";
block_type_gr1_data <= "00";
mixed_block_flag_gr0_data <= '0';
mixed_block_flag_gr1_data <= '0';
table_select_gr0_region0_data <= "00000";
table_select_gr0_region1_data <= "00000";
table_select_gr0_region2_data <= "00000";
table_select_gr1_region0_data <= "00000";
table_select_gr1_region1_data <= "00000";
table_select_gr1_region2_data <= "00000";
subblock_gain_gr0_window0_data <= "000";
subblock_gain_gr0_window1_data <= "000";
subblock_gain_gr0_window2_data <= "000";
subblock_gain_gr1_window0_data <= "000";
subblock_gain_gr1_window1_data <= "000";
subblock_gain_gr1_window2_data <= "000";
region0_count_gr0_data <= "0000";
region0_count_gr1_data <= "0000";
region1_count_gr0_data <= "000";
region1_count_gr1_data <= "000";
preflag_gr0_data <= '0';
preflag_gr1_data <= '0';
scalefac_scale_gr0_data <= '0';
scalefac_scale_gr1_data <= '0';
```

```
            count1table_select_gr0_data <= '0';
            count1table_select_gr1_data <= '0';
            finish <= '1' after 20 ns;
            din_count <= 0;
        end if;
        else
            main_data_begin_data <= main_data_begin_data;
            private_bits_data <= private_bits_data;
            scfsi_band0_data <= scfsi_band0_data;
            scfsi_band1_data <= scfsi_band1_data;
            scfsi_band2_data <= scfsi_band2_data;
            scfsi_band3_data <= scfsi_band3_data;
            part2_3_length_gr0_data <= part2_3_length_gr0_data;
            part2_3_length_gr1_data <= part2_3_length_gr1_data;
            big_values_gr0_data <= big_values_gr0_data;
            big_values_gr1_data <= big_values_gr1_data;
            global_gain_gr0_data <= global_gain_gr0_data;
            global_gain_gr1_data <= global_gain_gr1_data;
            scalefac_compress_gr0_data <= scalefac_compress_gr0_data;
            scalefac_compress_gr1_data <= scalefac_compress_gr1_data;
            window_switching_flag_gr0_data <= window_switching_flag_gr0_data;
            window_switching_flag_gr1_data <= window_switching_flag_gr1_data;
            block_type_gr0_data <= block_type_gr0_data;
            block_type_gr1_data <= block_type_gr1_data;
            mixed_block_flag_gr0_data <= mixed_block_flag_gr0_data;
            mixed_block_flag_gr1_data <= mixed_block_flag_gr1_data;
            table_select_gr0_region0_data <= table_select_gr0_region0_data;
            table_select_gr0_region1_data <= table_select_gr0_region1_data;
            table_select_gr0_region2_data <= table_select_gr0_region2_data;
            table_select_gr1_region0_data <= table_select_gr1_region0_data;
            table_select_gr1_region1_data <= table_select_gr1_region1_data;
            table_select_gr1_region2_data <= table_select_gr1_region2_data;
            subblock_gain_gr0_window0_data <= subblock_gain_gr0_window0_data;
            subblock_gain_gr0_window1_data <= subblock_gain_gr0_window1_data;
            subblock_gain_gr0_window2_data <= subblock_gain_gr0_window2_data;
            subblock_gain_gr1_window0_data <= subblock_gain_gr1_window0_data;
            subblock_gain_gr1_window1_data <= subblock_gain_gr1_window1_data;
            subblock_gain_gr1_window2_data <= subblock_gain_gr1_window2_data;
            region0_count_gr0_data <= region0_count_gr0_data;
            region0_count_gr1_data <= region0_count_gr1_data;
```

```
                region1_count_gr0_data <= region1_count_gr0_data;
                region1_count_gr1_data <= region1_count_gr1_data;
                preflag_gr0_data <= preflag_gr0_data;
                preflag_gr1_data <= preflag_gr1_data;
                scalefac_scale_gr0_data <= scalefac_scale_gr0_data;
                scalefac_scale_gr1_data <= scalefac_scale_gr1_data;
                count1table_select_gr0_data <= count1table_select_gr0_data;
                count1table_select_gr1_data <= count1table_select_gr1_data;
                din_count <= din_count;
                finish <= finish;
            end if;
        end if;
end process;
        main_data_begin <= main_data_begin_data;
        private_bits <= private_bits_data;
        scfsi_band0 <= scfsi_band0_data;
        scfsi_band1 <= scfsi_band1_data;
        scfsi_band2 <= scfsi_band2_data;
        scfsi_band3 <= scfsi_band3_data;
        part2_3_length_gr0 <= part2_3_length_gr0_data;
        part2_3_length_gr1 <= part2_3_length_gr1_data;
        big_values_gr0 <= big_values_gr0_data;
        big_values_gr1 <= big_values_gr1_data;
        global_gain_gr0 <= global_gain_gr0_data;
        global_gain_gr1 <= global_gain_gr1_data;
        scalefac_compress_gr0 <= scalefac_compress_gr0_data;
        scalefac_compress_gr1 <= scalefac_compress_gr1_data;
        window_switching_flag_gr0 <= window_switching_flag_gr0_data;
        window_switching_flag_gr1 <= window_switching_flag_gr1_data;
        block_type_gr0 <= block_type_gr0_data;
        block_type_gr1 <= block_type_gr1_data;
        mixed_block_flag_gr0 <= mixed_block_flag_gr0_data;
        mixed_block_flag_gr1 <= mixed_block_flag_gr1_data;
        table_select_gr0_region0 <= table_select_gr0_region0_data;
        table_select_gr0_region1 <= table_select_gr0_region1_data;
        table_select_gr0_region2 <= table_select_gr0_region2_data;
        table_select_gr1_region0 <= table_select_gr1_region0_data;
        table_select_gr1_region1 <= table_select_gr1_region1_data;
        table_select_gr1_region2 <= table_select_gr1_region2_data;
        subblock_gain_gr0_window0 <= subblock_gain_gr0_window0_data;
```

```
        subblock_gain_gr0_window1 <= subblock_gain_gr0_window1_data;
        subblock_gain_gr0_window2 <= subblock_gain_gr0_window2_data;
        subblock_gain_gr1_window0 <= subblock_gain_gr1_window0_data;
        subblock_gain_gr1_window1 <= subblock_gain_gr1_window1_data;
        subblock_gain_gr1_window2 <= subblock_gain_gr1_window2_data;
        region0_count_gr0 <= region0_count_gr0_data;
        region0_count_gr1 <= region0_count_gr1_data;
        region1_count_gr0 <= region1_count_gr0_data;
        region1_count_gr1 <= region1_count_gr1_data;
        preflag_gr0 <= preflag_gr0_data;
        preflag_gr1 <= preflag_gr1_data;
        scalefac_scale_gr0 <= scalefac_scale_gr0_data;
        scalefac_scale_gr1 <= scalefac_scale_gr1_data;
        count1table_select_gr0 <= count1table_select_gr0_data;
        count1table_select_gr1 <= count1table_select_gr1_data;
        data_count <= din_count;
        mode <= mode_in;
        layer <= layer_in;
        finish_out <= finish;
end ;
library IEEE;
use IEEE.Std_Logic_1164.all;
package rom_10K is
        constant rom_width:integer:=1;
        constant rom_length:integer:=300;
        subtype rom_word is std_logic;
        type rom_table is array (0 to rom_length-1) of rom_word;
        constant rom:rom_table:=rom_table'
        ('1','1','1','1','1','1','1','1','1','1','1','1','1','0','1','1','1','0','0','1','0','0','1','0','1','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','1','0','1','1','1','1','1','1','0','0','1','0','1','0','1','1','1','0','1','0','1','0','1','0','1','0','0','0','0','0','0','0','1','0','1','0','0
','0','0','0','1','0','1','1','1','1','0','0','0','0','0','0','0','0','0','0','0','1','0','1','1','0','0','1','0','0','0','1','1','1','1','0','0','1','0','0','0','0','0','1','0','1','0
','0','1','0','1','0','0','0','0','1','1','0','0','0','1','0','1','0','1','0','1','0','1','0','0','0','0','0','0','0','0','0','0','0','1','1','1','1','1','0','0','1','0','0','0','0','0
','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','1','1','1','0','0','1','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0
','1','1','1','0','1','0','0','1','0','1','0','0','1','0','1','0','0','1','0','1','0','0','1','1','1','1','1','0','0','1','1','0','1','1','1','1','0','1','1','1','1','0','1','1','1','1
','0','1','0','0','1','1','0','1','0','1','1','1','1','0','1','1','1','1','0','1','1','1','1','0','1','1','1','1');
end;
library IEEE;
use IEEE.Std_Logic_1164.all;
use ieee.std_logic_unsigned.all;
use work.rom_10K.all;
```

```
entity waveform_10K is
        port (addr:in integer:=0;
                dout:out rom_word:='0');
end;
architecture test_10K of waveform_10K is
begin
        dout <= rom(addr);
end;
```

กิตติกรรมประกาศ

ขอขอบคุณ

- ผศ.ดร. สมศักดิ์ ชุมช่วย ที่ให้คำปรึกษาแนะนำ
- ผู้แต่งหนังสือทุกๆเล่มที่ช่วยให้เข้าใจในขั้นตอนการทำงานของระบบเอ็มเป็ก ,โปรแกรมแมทแลบ และภาษาวีเอชดีแอล
- ผู้จัดทำเว็บไซต์ต่างๆที่เอื้อเฟื้อข้อมูล
- เพื่อนๆที่ช่วยกันหาข้อมูลและร่วมกันแลกเปลี่ยนความคิด

ทิตยา เหลืองศิริธัญญะ

(นางสาวทิตยา เหลืองศิริธัญญะ)

นิธรรมนูญ ชัยปัญญากุล

(นายธรรมนูญ ชัยปัญญากุล)

ธิติ อึ้งอารีย์วิทยา

(นายธิติ อึ้งอารีย์วิทยา)

ผู้จัดทำ

วันที่ 21 มีนาคม 2544

บรรณานุกรม

1. พิพัฒน์พงศ์ เกื้อศิริกุล, **"ปริญญานิพนธ์ เรื่อง ตัวถอดรหัสเสียงเอ็มเป็ก"**, สาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง,110 หน้า.,2542

2. Bjorn Wesn,"**A DSP-based decompressor unit for high-fidelity MPEG-Audio over TCP/IP networks**",http://www.sparta.lu.se/~bjorn/whitney/index.htm,1997.

3. ISO/IE11172,"**Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s Part3: Audio**", The International Organization for Standardization and the International Electrotechnical Commission,143 p.,1993.

4. Kent Salomonsen,"**Design and Implement of an MPEG/Audio Layer III Bitstream Processor**",Aalborg University,237p.,1997.