การออกแบบโมเด็มการ์ดโดยใช้ DSP Chip

Design MODEM on DSP Chip

โดย

นายธนาฒน์ สุกนวล

นายธรรมศักดิ์ เหล่าพิเชียรพงษ์

อาจารย์ที่ปรึกษา

ดร. สมศักดิ์ ชุมช่วย

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของวิชา 01084102 Project II

ของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ประจำภาคเรียนที่ 2 ปีการศึกษา 2540

ปริญญานิพนธ์เรื่อง	การออกแบบโมเด็มการ์ดโดยใช้ DSP Chip Design MODEM on DSP Chip
จัดทำโดย	นายธนาฒน์ สุกนวล นายธรรมศักดิ์ เหล่าพิเชียรพงษ์
อาจารย์ที่ปรึกษา	อาจารย์ ดร. สมศักดิ์ ชุมช่วย

ปริญญานิพนธ์ฉบับนี้ได้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

ลงชื่อ.....อาจารย์ที่ปรึกษา

(ดร. สมศักดิ์ ชุมช่วย)

วันที่ 6 / June / 56

การออกแบบ MODEM บน DSP Chip

Design MODEM on DSP Chip

นายธนาฒน์ สุกนวล (38013228)
นายธรรมศักดิ์ เหล่าพิเชียรพงษ์ (38013230)
อาจารย์ ดร. สมศักดิ์ ชุมช่วย (อาจารย์ที่ปรึกษา)
ภาคเรียนที่ 2 ปีการศึกษา 2540

บทคัดย่อ

จากภาวะปัจจุบันนี้ การสื่อสารข้อมูลเป็นสิ่งที่มีความสำคัญอย่างยิ่ง ได้มีการนำอุปกรณ์อิเล็กทรอนิกส์มาประยุกต์ใช้งานกันอย่างกว้างขวาง และอุปกรณ์หนึ่งที่เป็นที่นิยมมากก็คือ "โมเด็ม" ซึ่งในที่นี้ได้กล่าวถึงโมเด็ม V.34 ที่มีความเร็วสูงสุดที่ 33.6 kb/s และยังได้กล่าวถึง TMS 320C50 DSP Chip ซึ่งในโครงงานนี้ได้นำ TMS 320C50 DSP มาประยุกต์ใช้งานให้ทำงานได้เหมือนโมเด็ม V.34

ในเนื้อหาของโครงงานนี้ได้มีการอธิบายหลักการทำงานตลอดจนการคำนวนค่าต่าง ๆ ในการดำเนินการของโมเด็ม V.34 และยังได้นำรายละเอียดของ TMS 320C50 DSP แต่จากการดำเนินงานแล้วพบว่าในรายละเอียดของโมเด็ม V.34 ได้มีการแบ่งภาคการทำงานไว้มากมาย แต่เนื่องด้วยระยะเวลาในการทำโครงงานมีจำกัดจึงได้กระทำเพียงในส่วนของภาค convolution encode, differential encode และภาค Viterbi decode ซึ่งเป็นภาคที่มีความสำคัญของโมเด็ม V.34 เป็นที่เรียบร้อย หากมีผู้สนใจที่จะดำเนินงานให้ TMS 320C50 DSP ให้ทำงานเป็นโมเด็ม V.34 ก็สามารถนำโครงงานนี้เป็นแนวทางในการปฏิบัติต่อไป

การออกแบบ MODEM บน DSP Chip

Design MODEM on DSP Chip

Mr. Thanat Sooknuan (38013228)

Mr. Thammasak Laopicheinpong (38013230)

Dr. Somsak Chumchui (Advisor)

2nd Semester , Year 1997

Abstract

The telecommunication is of necessary nowsaday where electronic devices are applicated various. One device that popular is Modem. And now we will refer to V.34 modem that update to 33.6 kb/s and description of TMS 320C50 DSP to design V.34 modem on DSP chip.

Recommendation of this project describes practice, calculate of V.34 modem and describe spec of TMS 320C50 DSP. Operations of V.34 modem are separated in various sections. A result of time in operate this project that won't do then we clarified convolution encode, differential encode section and viterbi decode section. These are important sections of V.34 modem already. The results and studies achieved are of beneficial background in future implementing a V.34 modem uses DSP chip.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

หน้า

สารบัญรูป (ต่อ)

สารบัญตาราง

บทที่1 บทนำ

1.1 การรับส่งข้อมูลแบบ Full Duplex และ Half Duplex

ในการรับส่งข้อมูลระหว่างกันของคอมพิวเตอร์สองเครื่องนั้น อาจแบ่งตามลักษณะของการรับส่งออกได้เป็น 3 วิธีใหญ่ ๆ คือ

1.1.1 การรับหรือส่งทางเดียว (Simplex)

1.1.2 การรับส่งแบบผลัดกันส่ง (Half Duplex)

1.1.3 การรับส่งสวนทางได้พร้อมกัน (Full Duplex)

การติดต่อสื่อสารแบบรับหรือส่งทางเดียวนั้น เรียกว่าเป็นการสื่อสารแบบ Simplex ตัวอย่างง่าย ๆ ที่เห็นได้ชัดก็คือ การรับส่งโทรทัศน์และวิทยุกระจายเสียงนั่นเอง สถานีโทรทัศน์จะเป็นตัวส่ง และเครื่องรับของเราก็จะทำหน้าที่รับแต่เพียงอย่างเดียว

1.2 Full duplex และ Half duplex

สำหรับการรับส่งแบบที่สองนี้ เราเรียกว่าการรับส่งแบบ Half duplex มีคุณสมบัติที่สามารถทำได้ทั้งรับและส่งข้อมูล แต่ว่าต้องสลับกันส่ง จะส่งพร้อมกันทั่งสองด้านไม่ได้ อุปกรณ์ที่ใช้ในการติดต่อในแบบ Half duplex ตัวอย่างเช่น วิทยุมือถือ (Walkie - talkie) และ Intercom เป็นต้น

การรับส่งข้อมูลแบบที่ซับซ้อนที่สุดก็คือ การรับส่งในแบบสวนทางได้พร้อมกัน ซึ่งเรียกว่า Full Duplex การรับส่งแบบนี้ ผู้รับและส่งสามารถรับและส่งพร้อมกันในเวลาเดียวกันได้ ไม่จำเป็นต้องรอให้อีกฝ่ายหนึ่งส่งจบเสียก่อนอย่างในการรับส่งแบบ Half duplex ตัวอย่างเช่น การพูดโทรศัพท์ของเรา ถึงแม้ปกติเมื่อผู้หนึ่งพูดอีกฝ่ายจะคอยฟัง แล้วตอบกลับมาเมื่อฝ่ายแรกพูดจบ ซึ่งเป็นลักษณะของการติดต่อแบบ Half duplex ก็ตาม แต่เราอาจจะพูดพร้อม ๆ กัน หรือพูดสวนกลับไปได้ทันที โดยยังคงฟังอยู่เหมือนเดิม ลักษณะเช่นนี้เราเรียกว่าติดต่อกันในแบบ Full Duplex

1.3 โมเด็มคืออะไร, มีการทำงานอย่างไร

การรับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ในระยะทางไม่ไกลมากนัก เราอาจใช้การรับส่งแบบอนุกรมแบบหนึ่งที่เรียกว่ามาตรฐาน RS - 232C ซึ่งส่งข้อมูลดิจิตอลของคอมพิวเตอร์ไปตามสายจนถึงผู้รับได้ กรณีนี้เราสามารถรับส่งข้อมูลได้ไกลถึง 35 เมตรตามคุณสมบัติของ RS - 232C หรือถ้าสายเคเบิลที่ใช้มีคุณภาพดี อาจส่งได้ไกลถึง 150 เมตร ที่ความเร็ว 9,600 บิทต่อวินาที แต่สำหรับระยะทางที่ไกลมาก ๆ เช่นหลายสิบกิโลเมตร, หลายร้อยกิโลเมตร จนถึงหลาย ๆ พันกิโลเมตร การส่งข้อมูลแบบดิจิตอลออกไปโดยตรงจะไม่เหมาะสมหลายอย่าง ปัญหาที่สำคัญก็คือคลื่นรูปสี่เหลี่ยมของสัญญาณดิจิตอล เมื่อส่งไปไกล ๆจะเพี้ยนหรือมีรูปร่างผิดไปจากเดิมได้ง่าย ทำให้สายส่ง และวงจรรับ

ส่งสัญญาณดิจิตอล ต้องถูกออกแบบมาเป็นอย่างดี ราคาของสายส่งสัญญาณแบบดิจิตอลจึงมีราคาแพงกว่าสายส่งสัญญาณแบบอนาล็อกมาก ในทางปฏิบัติเราอาจรับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์สองเครื่องโดยใช้สัญญาณดิจิตอลผ่านสายส่งได้ ซึ่งทั้งสายส่งและวงจรเชื่อมต่อทั้งหมดเป็นแบบดิจิตอล แต่ว่าค่าใช้จ่ายจะมีราคาแพงมากจนกระทั่งไม่ค่อยคุ้มที่จะทำเช่นนี้ วิธีหลีกเลี่ยงก็คือหาทางส่งข้อมูลไปตามสายส่งในแบบอนาล็อกแทน การทำเช่นนี้เราจำเป็นต้องใช้อุปกรณ์เข้าช่วยแปลงสัญญาณในการรับส่งข้อมูลทั้งสองด้าน ซึ่งเป็นที่มาของโมเด็ม (MODEM) นั่นเอง

โมเด็ม (MODEM) จะทำหน้าที่แปลงสัญญาณดิจิตอลจากคอมพิวเตอร์ที่ส่งมาทาง RS - 232C ให้กลายเป็นสัญญาณอนาล็อกแล้วส่งออกไปตามสายส่ง ขบวนการนี้เราเรียกว่าการ Modulate สัญญาณ เมื่อถึงปลายทางโมเด็มก็จะแปลงสัญญาณอนาล็อกที่ได้รับกลับมาเป็นสัญญาณดิจิตอล แล้วจึงส่งให้คอมพิวเตอร์ต่อไปในรูปแบบของสัญญาณดิจิตอลผ่านทาง RS - 232C เช่นกัน ขบวนการแปลงสัญญาณกลับนี้เรียกว่า Demodulation ชื่อของโมเด็มก็ได้จากการทำงานทั้งสองแบบนั่นเอง คือ Modulate - DEModulate → MODEM

สัญญาณอนาล็อกมีคุณสมบัติเหมาะที่จะส่งไปไกล ๆ มากกว่าสัญญาณแบบดิจิตอล เพราะว่าสัญญาณอนาล็อกจะเพี้ยนหรือมีรูปร่างผิดจากเดิมยากกว่า และสูญเสียกำลังในสายส่งน้อยกว่า ทำให้ส่งได้ระยะทางไกลมากขึ้น นอกจากนี้เรายังสามารถกรองสัญญาณรบกวนบางส่วนที่ไม่ต้องการ (filter) ออกได้อีกด้วย ราคาของสายส่งและอุปกรณ์เชื่อมต่อก็มีราคาถูก เราจึงจำเป็นต้องใช้โมเด็มในการรับส่งข้อมูลคอมพิวเตอร์ระยะทางไกลผ่านสายส่งอนาล็อก

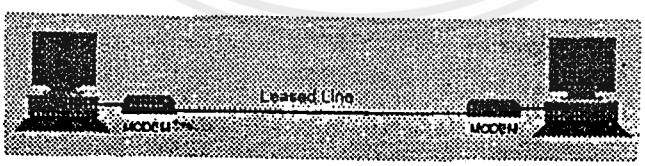
จากการที่โมเด็มแปลงสัญญาณดิจิตอลจากคอมพิวเตอร์ให้กลายเป็นสัญญาณอนาล็อก ในการรับส่งข้อมูลนี้เอง ถ้าโมเด็มแปลงสัญญาณออกมารอยู่ในรูปของเสียง ซึ่งเป็นสัญญาณอนาล็อกแบบหนึ่ง เราก็สามารถรับส่งข้อมูลผ่านทางสายโทรศัพท์ได้ โมเด็มทั่วๆไปที่เราใช้งานจะเป็นโมเด็มที่แปลงสัญญาณจากคอมพิวเตอร์ให้อยู่ในรูปคลื่นเสียงทั้งนั้น มีโมเด็มบางชนิดที่แปลงสัญญาณดิจิตอลเป็นสัญญาณอนาล็อกความถี่สูง แต่โมเด็มแบบนี้มีใช้งานน้อยและส่งข้อมูลโดยใช้สายส่งพิเศษ จะส่งผ่านสายโทรศัพท์ธรรมดาไม่ได้ เราจึงเน้นเฉพาะโมเด็มที่ทำงานในย่านความถี่เสียงเท่านั้น ไม่ว่าโมเด็มจะเป็นแบบไหนก็ตาม เมื่อได้รับข้อมูลดิจิตอลจากคอมพิวเตอร์มันจะเปลี่ยนให้กลายเป็นสัญญาณอนาล็อก จากนั้นก็นำสัญญาณอนาล็อกที่ได้นี้มารวมเข้ากับสัญญาณพาหะ (carrier wave) แล้วส่งออกไปทางสายส่งข้อมูล สัญญาณพาหะหรือคลื่นพาหะนี้จะทำหน้าที่พาข้อมูลที่อยู่ในรูปสัญญาณอนาล็อกไปจนถึงปลายทาง

ตามปกติเมื่อโมเด็มโทรติดต่อกันแล้วจะยังรับส่งข้อมูลไม่ได้ในทันที แต่จะต้องตกลงรายละเอียดของวิธีการรับส่งข้อมูลกันเสียก่อน เรียกว่าการทำ hand shaking ซึ่งมีขั้นตอนดังนี้คือ เมื่อโมเด็มปลายทางตอบรับแล้ว โมเด็มต้นทางจะทำการทดสอบสภาพสายก่อนว่าสามารถรับส่งข้อมูลได้ดีเพียงใด เพื่อที่จะได้เลือกความเร็วที่สูงสุดเท่าที่สายจะรับได้ จากนั้นจะทดสอบความเพี้ยนของ

สัญญาณต่อไป หลังจากขั้นตอนนี้โมเด็มจะตกลงกับปลายทางได้ว่าจะใช้ความเร็วในการรับส่งข้อมูลเท่าไหร่, ใช้การผสมสัญญาณแบบไหนที่ความถี่เท่าใด ถ้าใช้ความเร็วสูงสุดตามที่กำหนดไม่ได้โมเด็มก็จะทำการลดความเร็วลง และทดสอบสภาพสายใหม่ จนกระทั่งได้ความเร็วที่สามารถรับส่งข้อมูลได้อย่างถูกต้อง ซึ่งการลดความเร็วของโมเด็มจะลดลงไปเป็นขั้น ๆ ตามแต่ละมาตรฐานที่กำหนดไว้ จากนั้นโมเด็มจะตกลงกับปลายทางว่าจะใช้วิธีการตรวจสอบความผิดพลาด (Error Detection) แบบไหนและใช้การลดขนาดข้อมูล (Data Compression) หรือไม่ ขั้นตอนการทำ hand shaking ของโมเด็มทั้งหมดจะใช้เวลาหลายสิบวินาที เมื่อเสร็จเรียบร้อยแล้วจึงจะสามารถรับส่งข้อมูลกันได้

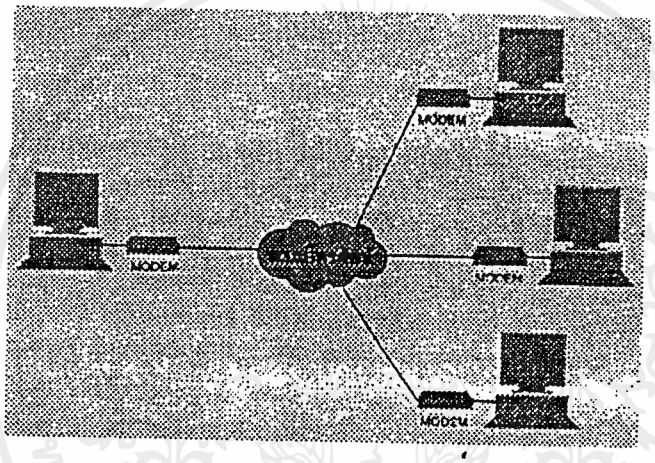
1.4โมเด็มสำหรับ Leased line,Dial-up line

โมเด็มแบ่งตามการใช้งานได้สองแบบคือ โมเด็มที่ใช้กับสายตรง (Leased Line) และโมเด็มที่ใช้กับสายโทรศัพท์ (Dial-up line) โมเด็มที่ใช้กับสายตรงหรือสายเช่าจะส่งข้อมูลด้วยความเร็วสูงจนถึงสูงมาก (9,600 บิตต่อวินาทีไปจนถึง 2 ล้านบิตต่อวินาที) ผ่านสายที่ลากตรงไปยังจุดหมายปลายทางตายตัว ซึ่งเป็นการติดต่อในลักษณะจุดถึงจุด (Point to Point) จะต่อไปยังจุดอื่น ๆ ไม่ได้ ส่วนมากจะเป็นการใช้ติดต่อส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ที่มีข้อมูลส่งไปมาเป็นจำนวนมาก เช่นเครือข่าย ATM ของธนาคาร,เทอร์มินัลของเครื่องคอมพิวเตอร์ขนาดใหญ่ ซึ่งอาจจะเป็นมินิคอมพิวเตอร์หรือเมนเฟรม เป็นต้น การส่งข้อมูลมักจะส่งเป็นกลุ่มและมีซอฟแวร์ควบคุมการรับส่งโดยเฉพาะ ที่เราเรียกว่าการรับส่งแบบ Synchromous นั่นเอง ข้อดีของโมเด็มแบบที่ใช้สายตรง คือส่งข้อมูลได้ที่ความเร็วสูง เนื่องจากสายส่งมีคุณภาพดี เหมาะกับงานส่งข้อมุลจำนวนมากระหว่างจุดสองจุด ข้อเสียก็คือไม่สามารถเปลี่ยนจุดรับส่งข้อมูลไปตามที่ต่าง ๆ ได้จึงขาดความคล่องตัว



รูปที่ 1.1 การรับส่งข้อมูลผ่านสายตรงจะเป็นการติดต่อกันระหว่างจุดต่อจุดตายตัว

โมเด็มแบบใช้กับสายโทรศัพท์ จะรับส่งข้อมูลด้วยความเร็วตั้งแต่ 300 บิตต่อวินาทีจนถึง 56,000 บิตต่อวินาที ผ่านเครือข่ายโทรศัพท์ที่เราใช้กันอยู่แล้ว เราสามารถรับส่งข้อมูลไปยังที่ต่าง ๆ ได้ตามต้องการ โดยไม่กำหนดจุดรับข้อมูลตายตัวเหมือนอย่างโมเด็มสายตรง เครื่องคอมพิวเตอร์ส่วนบุคคลและบริการรับส่งข้อมูลต่างๆ จะใช้โมเด็มแบบนี้ในการทำงานเกือบทั้งหมด การรับส่องข้อมูลจะเป็นการรับส่งทีละหนึ่งตัวอักษร ไม่ส่งเป็นกลุ่ม เรียกว่ารับส่งแบบ Asynchronous ซึ่งโดยปกติแล้วเราจะรับส่งข้อมูลผ่านสายโทรศัพท์ด้วยความเร็ว 9,600 ถึง 28,000 บิตต่อวินาทีเท่านั้น



รูปที่ 1.2 การรับส่งข้อมูลผ่านสายโทรศัพท์ เราสามารถรับส่งข้อมูลกับจุดต่าง ๆ ได้หลายแห่งโดยหมุนเบอร์ปลายทางผ่านชุมสาย

ความเร็วสูงสุดที่เชื่อถือได้ในการรับส่งข้อมูลของโมเด็มคือ 28,800 บิตต่อวินาที ถ้าใช้โมเด็มความเร็วสูงกว่านี้รับส่งข้อมูลผ่านสายโทรศัพท์ เช่น 33,600 บิตต่อวินาที ไปจนถึง 56,000 บิตต่อวินาทีอาจทำได้ในบางพื้นที่ แต่ถ้าส่งข้อมูลไกล ๆ ระหว่างจังหวัดหรือระหว่างประเทศอาจจะมีความผิดพลาดสูง จึงถือความเร็ว 28,800 บิตต่อวินาที เป็นความเร็วสูงสุดสำหรับส่งข้อมูลผ่านสายโทรศัพท์ในปัจจุบัน ข้อดีของการใช้โมเด็มแบบนี้ก็คือ มีความคล่องตัวสูง สามารถรับส่งข้อมูลไปยังที่ต่าง ๆ ได้ไม่จำกัด และไม่จำเป็นต้องจัดหาวงจรสายตรงมาเป็นพิเศษเพื่อส่งข้อมูล มีข้อเสียตรงที่ว่าความเร็วในการส่งข้อมูลต่ำกว่าโมเด็มแบบสายตรง ข้อเสียอีกอันหนึ่งก็คือ ถ้ารับส่งข้อมูลเป็นจำนวนมากไปยัง

จุดหมายปลายทางจุดเดียวในระยะทางไกล ค่าโทรศัพท์ทางไกลอาจจะแพงกว่าการเช่าวงจรสายตรงมาใช้ก็ได้ อันนี้ขึ้นกับเวลาที่ใช้ส่งข้อมูลเป็นหลัก เราต้องพิจารณาค่าใช้จ่ายเปรียบเทียบเอาเอง

โมเด็มที่มีขายในท้องตลาดปัจจุบัน บางรุ่นอาจทำงานได้เฉพาะกับสายตรงหรือบางรุ่นอาจทำงานได้กับสายโทรศัพท์เท่านั้น โมเด็มที่ใช้กับสายตรงได้เพียงอย่างเดียวจะนำมาต่อใช้กับสายโทรศัพท์ไม่ได้ และโมเด็มที่ใช้กับสายโทรศัพท์ได้อย่างเดียวก็จะนำมาใช้งานต่อกับสายตรงไม่ได้เช่นกัน ยกเว้นโมเด็มบางชนิด ซึ่งเลือกการทำงานในตัวได้ว่าจะต่อกับสายตรงหรือสายโทรศัพท์ โมเด็มที่ใช้งานได้ทั้งสองอย่างนี้ก็มักจะมีราคาแพงกว่าโมเด็มที่ใช้กับสายโทรศัพท์เพียงอย่างเดียว ถ้าการใช้งานของเราไม่ใช่การต่อพีซีเข้ากับเมนเฟรมหรือมินิคอมพิวเตอร์แล้วละก็ การใช้โมเด็มแบบต่อกับสายโทรศัพท์ได้อย่างเดียวก็นับว่าเพียงพอแล้วสำหรับรับส่งข้อมูลทั่วไป

1.5 Communications Parameter

จากการที่โมเด็มต่อกับคอมพิวเตอร์ผ่านทาง RS - 232C และส่งข้อมูลไปให้เครื่องคอมพิวเตอร์อีกเครื่องหนึ่ง ดังนั้นองค์ประกอบในการรับส่งข้อมูล (Communications Parameter) จึงต้องตรงกันตลอดเส้นทางที่รับส่ง องค์ประกอบดังกล่าวคือ ความเร็ว (Speed), จำนวนบิตของข้อมูล (Data Bit) จำนวนบิตเริ่มและบิตจบ (Start Bit, Stop Bit), การตรวจสอบพาริตีบิท (Parity Bit), และการเลือกใช้ Echo (Duplex) ในการรับส่ง

ความเร็ว (Speed), คืออัตราการรับส่งข้อมูลเป็นจำนวนบิตต่อวินาที เราต้องใช้ความเร็วเดียวกันตลอดเส้นทางการรับส่งข้อมูล เช่นเดียวกับองค์ประกอบอื่น ๆ ทั้งเครื่องคอมพิวเตอร์ที่ส่งข้อมูลออกไปโมเด็มที่ด้านส่ง, โมเด็มที่ด้านรับและคอมพิวเตอร์เครื่องรับจะต้องมีความเร็วในการรับส่งข้อมูลท่ากัน ความเร็วที่ใช้กันทั่วไปมีตั้งแต่ 2,400, 4,800, 9,600, 19,200 บิตต่อวินาทีไปจนถึง 38,400, 57,600 หรือ 115,000 บิตต่อวินาที ข้อสังเกตคือความเร็วนี้เป็นความเร็วในการส่งผ่านข้อมูลตลอดทั้งเส้นทาง แต่การส่งข้อมูลจริงในสายแต่ละช่วง เช่นระหว่างโมเด็มสองตัวอาจไม่สูงเท่านี้ เนื่องจากมีการใช้เทคโนโลยีในการย่อเข้ามาช่วยทำให้ปริมาณข้อมูลที่ต้องส่งลดลง

สำหรับจำนวนบิตของข้อมูล (Data Bit) นั้น คือในหนึ่งตัวอักษรจะใช้จำนวนข้อมูลกี่บิตในการส่ง ปรกติจะเลือกได้สองแบบคือ 7 บิตต่อหนึ่งตัวอักษรหรือ 8 บิตต่อหนึ่งตัวอักษร การรับส่งข้อมูลภาษาไทยเราจำเป็นต้องใช้แบบ 8 บิตต่อตัวอักษรเท่านั้น เนื่องจากภาษาไทยเราใช้รหัสครบทั้ง 8 บิต ถ้าหากส่งไปในแบบ 7 บิตต่อหนึ่งตัวอักษร รหัสภาษาไทยจะถูกตัดบิตที่ 8 บิตทิ้งไป กลายเป็นตัวภาษาอังกฤษซึ่งใช้งานไม่ได้ ส่วนจำนวนบิตเริ่มและบิตจบนั้นกำหนดไว้ให้ตัวรับและตัวส่งแยกออกว่าข้อมูลจะเริ่มต้นเมื่อไหร่ และจบลงเมื่อใด

บิตเริ่มต้น (Stsrt Bit) มักจะใช้หนึ่งบิตเสมอ ส่วน บิตจบข้อมูล (Stop Bit) จะมีสองแบบคือ หนึ่งบิตหรือสองบิต การใช้งานทั่วไปมักจะใช้หนึ่งบิตจบเช่นกัน

การตรวจสอบพาริตี (Parity Bit) เป็นการตรวจสอบความถูกต้องของการรับข้อมูลที่ส่งมา โดยสามารถเลือกให้ตรวจสอบว่าข้อมูลที่เป็น "1" เป็นจำนวนคู่ (Even) หรือจำนวนคี่ (Odd) หรือไม่ต้องตรวจสอบ (None) เช่นถ้าส่งข้อมูล 11000001 ไปให้ผู้รับและมีการตรวจสอบพาริตีแบบจำนวนคู่ พาริตีบิตในกรณีนี้จะมีค่าเป็น "1" เพื่อให้จำนวน "1" ทั้งหมดมีจำนวนเป็นเลขคู่ (Even) ถ้ากำหนดการตรวจสอบพาริตีเป็นจำนวนคี่พาริตีก็จะเป็น "0" เพื่อให้จำนวน "1" ของข้อมูลทั้งหมดเป็นจำนวนคี่ (Odd) และถ้าไม่มีการตรวจสอบเลย เครื่องส่งบิตก็จะส่งบิตจบ (Stop Bit) ปิดท้ายข้อมูลทันที ไม่มีการส่งพาริตีบิตไปให้ผู้รับส่วนมากถ้ารับส่งข้อมูลแบบ 7 บิต เรามักจะตั้งการตรวจสอง พาริตีเอาไว้ แต่ถ้าส่งรับข้อมูลแบบ 8 บิต ก็จะไม่มีการตรวจสอบพาริตีบิต สำหรับการเลือกใช้ Echoในการส่งก็เป็นการเลือกให้สอดคล้องกับ Full Duplex หรือ Half Duplex นั่นเอง ถ้าเป็นการรับส่งแบบ Full Duplex ก็ต้องเลือกใช้ Echo Off และถ้าเป็นแบบ Half Duplex เราก็จะเลือกใช้ Echo On เมื่อองค์ประกอบทั้งหมดนี้ตรงกัน การรับส่งข้อมูลจะผิดพลาดได้ ทั้งผู้รับและผู้ส่งจึงต้องตกลงกันให้แน่นอนว่าจะใช้องค์ประกอบในการรับส่งข้อมูลแต่ละตัวอย่างไร

1.6 ประเภทของโมเด็ม

การแบ่งประเภทโมเด็มสามารถแบ่งออกได้หลายวิธี ในตอนนี้เราจะแบ่งโมเด็มออกโดยใช้ความเร็วในการรับส่งข้อมูลเป็นหลัก โดยแบ่งออกตามความเร็วได้ 3 กลุ่ม คือ โมเด็มความเร็วต่ำ, โมเด็มความเร็วปานกลาง และโมเด็มความเร็วสูงตามลำดับ ความเร็วที่เราพูดถึงนี้คือความเร็วในการส่งข้อมูลซึ่งวัดเป็นบิตต่อวินาที (bit per second หรือ bps)

1.6.1 โมเด็มความเร็วต่ำ

โมเด็มความเร็วต่ำเป็นโมเด็มรุ่นแรก ๆ ที่ออกจำหน่ายในท้องตลาด โมเด็มความเร็วต่ำจะส่งข้อมูลที่ความเร็ว 300 บิตต่อวินาทีจนถึง 4,800 บิตต่อวินาที ใช้การผสมสัญญาณแบบเปลี่ยนความถี่ (Frequency Shift Keying - FSK) โดยแต่ละฝ่ายคือทั้งด้านรับและด้านส่งจะใช้ความถี่ ซึ่งไม่ซ้ำกันในการแทนค่าข้อมูล "0" และ "1"

1.6.2 โมเด็มความเร็วปานกลาง

โมเด็มความเร็วปานกลาง มีความเร็วในการส่งข้อมูลประมาณ 9,600 ถึง 14,400 บิตต่อวินาที โมเด็มความเร็วปานกลางนี้ใช้เทคนิคการผสมสัญญาณที่ก้าวหน้ากว่าโมเด็มความเร็วต่ำมาก คือแทนที่จะใช้การเปลี่ยนแปลงความถี่ไปมาเพื่อแทนข้อมูลของ "0" และ"1" โมเด็มความเร็วปานกลาง

จะผสมสัญญาณโดยเปลี่ยนแปลงมุมของช่วงคลื่น (Phase) และขนาดสัญญาณ (Amplitude) ไปพร้อม ๆ กัน

1.6.3โมเด็มความเร็วสูง

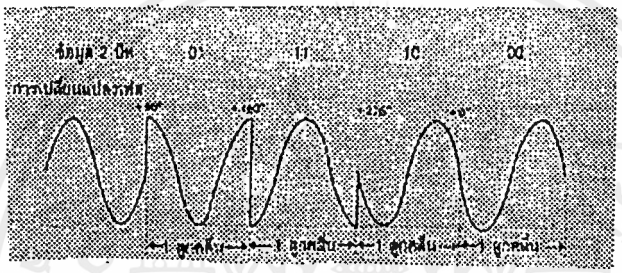
โมเด็มความเร็วสูง มีความเร็วในการรับส่งข้อมูลตั้งแต่ 19,200 จนถึง 28,800 บิตต่อวินาที หรือสูงกว่านั้น เช่นที่ 33,600 บิตต่อวินาที การที่โมเด็มสามารถรับส่งข้อมูลได้เร็วมากขนาดนี้ผ่านสายโทรศัพท์ได้ และมีการผสมสัญญาณที่ซับซ้อนกว่าโมเด็มความเร็วปานกลางมาก ในหนึ่งลูกคลื่นของสัญญาณที่ส่งออกไปอาจมีจำนวนบิตข้อมูลผสมอยู่ถึง 9 บิตก็ได้ และยังสามารถรับส่งในแบบ Full Duplex ได้อีกด้วย ซึ่งเมื่อเทียบกับความสามารถของสายโทรศัพท์ที่ส่งสัญญาณได้สูงสุด 3,400 Hz แล้ว ก็นับว่าโมเด็มความเร็วสูงเจาะทะลุขีดจำกัดต่าง ๆได้อย่างเหลือเชื่อ ถ้าเทียบเวลาที่ใช้ส่งข้อมูลขนาด 36 กิโลไบต์เท่าเดิมแล้ว ใช้เวลาส่งเพียงแค่ไม่ถึง 13 วินาทีเท่านั้นที่ความเร็ว 28,800 บิตต่อวินาที ฟังก์ชันการทำงานต่าง ๆ ของโมเด็มความเร็วสูงก็มีให้ใช้ใกล้เคียงกับโมเด็มความเร็วปานกลาง และสามารถลดความเร็วในการรับส่งข้อมูลลงไปกลายเป็นโมเด็มความเร็วปานกลางก็ได้ โมเด็มความเร็วสูงปัจจุบันมีความเร็วในการรับส่งข้อมูลสูงสุดที่ 33,600 บิตต่อวินาที

โมเด็มจำเป็นต้องมีมาตรฐานเช่นเดียวกับอุปกรณ์อื่น ๆ มาตรฐานของโมเด็มจะแบ่งออกเป็นสองส่วน คือมาตรฐานในส่วนของฮาร์ดแวร์ที่โมเด็มใช้ และอีกส่วนหนึ่งก็คือมาตรฐานในส่วนของซอฟแวร์ที่ควบคุมการทำงานของโมเด็มหรือคำสั่งของโมเด็ม โดยมาตรฐานนี้กำหนดขึ้นจากองค์การมาตรฐานสื่อสารสากล หรือ CCITT (International Telephone and Telegraph Consultative Committee) ที่ปัจจุบันเปลี่ยนชื่อมาเป็น ITU-T(International Telecommunication Union -Telecommunication) ซึ่งจะครอบคลุมเกี่ยวกับควาเร็วในการรับส่งข้อมูล,ความถี่ที่ใช้ และเทคนิคการผสมสัญญาณในสาย ฯลฯ

1.7 Bit rate กับ Baud rate

ความเร็วในการส่งข้อมูลของโมเด็มนั้นกำหนดเป็น บิตต่อวินาที (bit second,bps) หรือ Bit Rate ซึ่งแตกต่างจากอัตราการเปลี่ยนแปลงของสัญญาณไฟฟ้าในสายส่งหรือที่เรียกว่า Baud Rate ในสมัยก่อนการรับส่งข้อมูลใช้เทคนิคการผสมสัญญาณแบบง่าย ๆ เช่นการเปลี่ยนแปลงความถี่ตามข้อมูล “0” และ “1” ที่ได้รับ อัตราการส่งข้อมูลและอัตราการเปลี่ยนแปลงของสัญญาณในสายส่งจึงมีค่าเท่ากัน หรือพูดอีกอย่างว่าสัญญาณรูปคลื่น 1 ลูกจะแทนข้อมูลเพียง 1 บิต เราจึงถือว่าอัตราการเปลี่ยนแปลงของสัญญาณในสายส่ง (Baud Rate) ก็คืออัตราการส่งข้อมูลนั่นเอง ต่อมาเทคนิคการผสมสัญญาณซับซ้อนมากขึ้น ทำให้เราสามารถส่งข้อมูลได้มากขึ้นกว่าเดิม โดยอัตราการเปลี่ยนแปลงของสัญญาณในสายยังคงเท่าเดิม ดังนั้นเมื่อเราพูดว่า โมเด็มรับส่งข้อมูลที่ความเร็ว 1,200 Baud เราจะไม่

ทราบเลยว่าโมเด็มนั้นรับส่งข้อมูลได้กี่บิตต่อวินาที เนื่องจากว่าถ้าโมเด็มผสมสัญญาณ 1 บิตต่อหนึ่งลูกคลื่น โมเด็มนั้นจะรับส่งข้อมูลที่ความเร็ว 1,200 บิตต่อวินาที หรือถ้าโมเด็มผสมสัญญาณ 4 บิตต่อหนึ่งลูกคลื่นที่เปลี่ยนแปลงในสายส่ง โมเด็มจะรับส่งข้อมูลได้เร็วถึง 4,800 บิตต่อวินาที โดยยังคงมีอัตราการเปลี่ยนแปลงสัญญาณในสายส่งเท่ากับ 1,200 Baud เหมือนเดิม เพราะฉะนั้นเราจึงเลิกใช้คำว่า Baud Rate สำหรับบอกความเร็ว การรับส่งข้อมูลของโมเด็มและหันมาใช้คำว่า Bit Rate หรืออัตราการส่งข้อมูลเป็นบิตต่อวินาทีแทน ซึ่งสื่อความหมายได้เข้าใจตรงกันมากกว่า



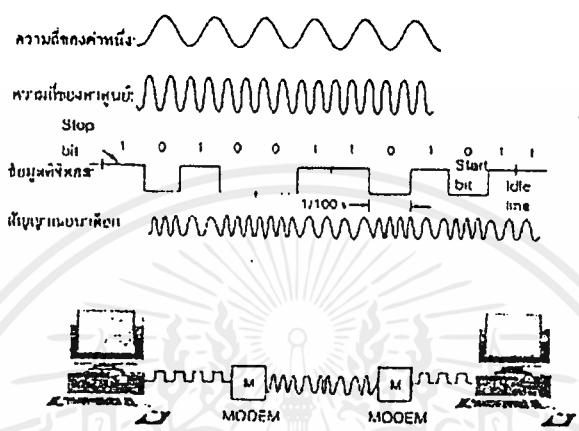
รูป 1.3 การผสมสัญญาณ 2 บิตต่อหนึ่งลูกคลื่นทำให้ Bit Rate มีค่าสูงกว่า Baud Rate ได้

1.8 การผสมสัญญาณแบบ FSK,PSK

นอกจากมาตรฐานจะกำหนดความเร็วในการรับส่งข้อมูลให้ตรงกันแล้ว ยังกำหนดความถี่ของคลื่นพาหะที่ใช้ว่า ผู้ส่ง,ผู้รับ ใช้ความถี่เท่าไหร่ และตลอดจนถึงมาตรฐานการผสมสัญญาณอีกด้วย มาตรฐานการผสมสัญญาณที่ใช้กันมากในปัจจุบันคือ Frequency Shift Keying (FSK),Phase Shift Keying(PSK) และ Quadrature Amplitude Modulation(QAM)

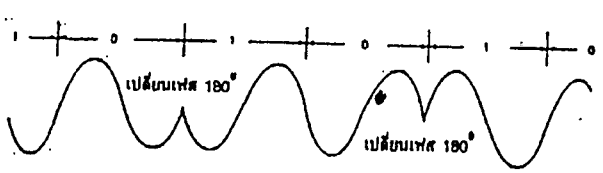
Frequency Shift Keying นั้น เราทราบกันดีแล้วว่าใช้ในโมเด็มความเร็วต่ำ โดยแทน "0" และ "1" ด้วยความถี่ต่างกัน ผู้ส่งจะใช้ความถี่สองความถี่ แทน "0" และ "1" ของด้านส่ง ส่วนผู้รับก็ใช้ความถี่อีกสองความถี่แทน "0"และ "1" ของด้านรับเช่นกัน ทั้งหมดจึงใช้ความถี่รวมสี่ความถี่ การผสมสัญญาณแบบ FSK มักใช้กับโมเด็มความเร็วประมาณ 300 ถึง 600 บิตต่อวินาที รวมทั้งใช้กับโมเด็มแบบ Acoustic Coupler ด้วย ความเร็วสูงสุดของโมเด็มที่ใช้เทคนิคนี้ในการผสมสัญญาณจะอยู่ที่ 1,200 บิตต่อวินาที แต่ตามปกติแล้วโมเด็มในท้องตลาดสำหรับเครื่องพีซีที่ใช้การผสมสัญญาณแบบ

FSK จะรับส่งข้อมูลด้วยความเร็ว 300 บิตต่อวินาที การผสมสัญญาณแบบ FSK นี้จะได้อัตราการส่งข้อมูล (Bit Rate) จะเท่ากับ Baud Rate เสมอ



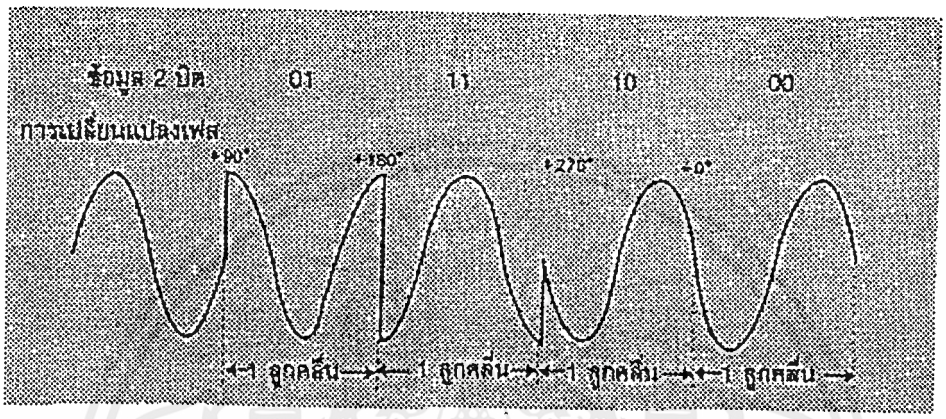
รูปที่ 1.4 แสดงการส่งข้อมูลโดยใช้ FSK

สำหรับการผสมสัญญาณแบบ Phase Shift Keying (PSK) นั้น ใช้หลักการที่ว่าแทนข้อมูล "0" และ "1" ด้วยการเปลี่ยนแปลงมุมของช่วงสัญญาณในสายส่ง (Phase) เช่นเราอาจกำหนดว่า "0" แทนด้วยมุมหรือองศาของคลื่นต่อเนื่องกันไป และ "1" แทนด้วยมุมที่มีการเปลี่ยนไปจากเดิม 180 องศา ถ้าเราส่งข้อมูล 10101 รูปร่างของคลื่นในสายส่งจะเป็นดังรูป



รูปที่ 1.5 การใช้ PSK ผสมสัญญาณแบบหนึ่งบิต

จากหลักการที่เราเปลี่ยน Phase ในสัญญาณส่งนี้เอง มุมทั้ง 360 องศาของคลื่นพาหะ เราสามารถแบ่งการเปลี่ยนแปลงของ Phase ให้เล็กลงไปได้อีก เช่นเปลี่ยน Phase ทีละ 90 องศา ซึ่งในหนึ่งลูกคลื่นจะเปลี่ยนได้ถึง 4 สภาวะ และกำหนดให้แต่ละสภาวะแทนด้วยข้อมูลสองบิตได้ จากการแทนค่า เราสามารถกำหนด 00,01,10 และ 11 ให้มีการเปลี่ยน Phase ได้ครั้งละ 90 องศา เมื่อเราส่งข้อมูลเช่น 01 11 10 00 รูปร่างของคลื่นในสายส่งจะเป็นดังรูป



รูปที่ 1.6 การใช้ PSK ผสมสัญญาณแบบ 2 บิต

ถ้าคลื่นพาหะมีความถี่ 600 Hz ข้อมูลที่ส่งออกไปจะมีค่าเท่ากับ 1,200 บิตต่อวินาที เนื่องจากการเปลี่ยนแปลง Phase หนึ่งครั้งเท่ากับข้อมูลสองบิต

เราจะเห็นว่าการผสมสัญญาณแบบ PSK นี้ อัตราการส่งข้อมูล (Bit Rate) จะมีค่าสูงกว่าอัตราการเปลี่ยนแปลงของสัญญาณในสายส่ง (Baud Rate) ได้ การผสมสัญญาณที่เราใช้กันมาก คือใช้ข้อมูล 2 บิต, 3 บิต และ 4 บิต ในการเปลี่ยน Phase โดยแบ่งมุมของคลื่นพาหะให้มีการเปลี่ยนแปลง เท่ากับ 90 องศา, 45 องศา และ 22.5 องศา ตามลำดับ ความเร็วสูงสุดที่ใช้ PSK ได้ก็คือ 9,600 บิต์ต่อวินาที ซึ่งมีการเปลี่ยนแปลงของมุมครั้งละ 22.5 องศา ซึ่งการเปลี่ยนแปลงมุม 22.5 องศานี้นับว่าน้อยมาก ทำให้ข้อมูลมักจะผิดพลาดอยู่เสมอ ส่วนมากจึงใช้งาน PSK ที่ความเร็ว 4,800 บิตต่อวินาทีแทน ส่วนที่ความเร็วสูงกว่านี้ เราจะใช้เทคนิคการผสมสัญญาณที่ซับซ้อนขึ้นไปอีกเข้ามาช่วย

Quadrature Amplitude Modulation(QAM) เป็นการผสมสัญญาณที่ใช้ทั้งการเปลี่ยน Phase และขนาดของสัญญาณควบคู่กันไป เป็นเทคนิคสำหรับใช้กับโมเด็มความเร็วสูง ซึ่งถ้าใช้การเปลี่ยน Phase เพียงอย่างเดียว มุมที่เปลี่ยนแปลงจะมีค่าน้อยเกินไปทำให้เกิดข้อผิดพลาดได้ง่าย ถ้าเราใช้การเปลี่ยน Phase และขนาดของสัญญาณประกอบเข้าด้วยกัน ก็จะช่วยให้วงจรทางผ่านรับสามารถแยกความแตกต่างระหว่างสัญญาณของข้อมูลต่างๆ กันได้อย่างชัดเจนยิ่งขึ้น ปกติที่มีใช้กันอยู่จะมี Phase

ต่างกัน 8 Phase และขนาดของสัญญาณต่างกัน 4 ระดับ ใช้แทนข้อมูล 16 สถานะ ซึ่งในหนึ่งลูกคลื่นจะสามารถส่งข้อมูลได้คราวละ 4 บิต การผสมสัญญาณของ QAM บางแบบจะใช้ Phase ต่างไปจากนี้ เช่นใช้ Phase ต่างกัน 12 Phase และขนาดของสัญญาณ 3 ระดับ หรืออาจใช้ Phase ต่างกัน 8 Phase และขนาดของสัญญาณต่างกัน 2 ระดับก็ได้ ขึ้นกับการออกแบบ แต่ว่าในมาตรฐานเดียวกันโมเด็มจะต้องใช้การแบ่ง Phase และระดับสัญญาณเท่ากันเสมอ ความเร็วการรับส่งข้อมูลของ QAM อยู่ที่ 9,600 บิตต่อวินาที โดยใช้ความถี่พาหะ 2,400 Hz และในหนึ่งลูกคลื่นแทนข้อมูลได้คราวละ 4 บิต เทคนิคการผสมสัญญาณแบบ QAM นี้อาจถูกนำไปใช้กับโมเด็มความเร็วปานกลางก็ได้ เช่นที่ความเร็ว 2,400 บิตต่อวินาที เป็นต้น

นอกจากนี้ยังมีการผสมสัญญาณแบบอื่น ๆ เช่น Trellis Coding Modulation (TCM) มักใช้กับโมเด็มความเร็วสูงรุ่นใหม่ ๆ เช่น V.32 bis,V.34 เป็นต้น ซึ่งสามารถผสมสัญญาณเข้าไปได้หลาย ๆ บิตต่อหนึ่งลูกคลื่น เช่น V.34 จะผสมสัญญาณ 9 บิต ต่อหนึ่งลูกคลื่น และใช้ความถี่พาหะ 3,200 Hz ทำให้รับส่งข้อมูลได้สูงถึง 28,800 บิตต่อวินาที ดังนั้นวงจรทางด้านฮาร์ดแวร์ ไม่ว่าจะเป็นวงจรด้านส่ง,วงจรด้านรับ และการแยกสัญญาณต่างๆ จึงยุ่งยากและมีราคาแพงกว่าโมเด็มที่ใช้การผสมสัญญาณแบบ QAM และ PSK อยู่มาก โดยหลักการแล้ว TCM จะใช้วิธีเข้ารหัสและผสมสัญญาณเหมือนกับ QAM นั่นเอง คือเข้ารหัสโดยเปลี่ยนทั้ง Phase และขนาดของสัญญาณพร้อมกัน แต่จะต่างกันตรงที่ TCM มีการเข้ารหัสของสัญญาณเป็นระบบดีกว่า ทำให้ด้านรับสามารถเดาได้ว่าข้อมูลถัดไปจะเป็นอะไร จึงต้านทานสัญญาณรบกวนได้สูงกว่า QAM มาก เหมาะกับการใช้งานในโมเด็มความเร็วสูงในปัจจุบัน

1.9 มาตรฐานของโมเด็มตาม CCITT(ITU-T) V-Series

มาตรฐานของโมเด็มที่เราใช้อยู่ในทุกวันนี้ เป็นไปตามที่องค์การมาตรฐานสื่อสารสากล หรือ CCITT (ITU-T) เป็นผู้กำหนดขึ้น โดยมีชื่อเรียกแต่ละมาตรฐานของโมเด็มขึ้นต้นด้วยอักษร "V" และตามด้วยตัวเลข เราจึงเรียกมาตรฐานอันนี้อีกชื่อหนึ่งว่า V-Series นอกจากมาตรฐานของโมเด็มแล้ว CCITT หรือ ITU-T ยังเป็นผู้กำหนดมาตรฐานทางการสื่อสารอื่น ๆ อีก เช่น มาตรฐานของการสื่อสารผ่านดาวเทียม มาตรฐานของโทรสาร (Facsimile), มาตรฐานการสื่อสารข้อมูลต่างๆ ทั้งในแบบดิจิตอลและอนาลอก รวมถึงมาตรฐานเกี่ยวกับระบบโทรศัพท์อีกด้วย มาตรฐานที่ CCITT หรือ ITU-T เป็นผู้กำหนดได้รับการยอมรับกันทั่วโลก การติดต่อสื่อสารระหว่างประเทศจึงเป็นไปได้อย่างไม่มีปัญหา เนื่องจากทุก ๆ คนต่างก็ทำตามมาตรฐานเดียวกัน

ก่อนอื่นขอทำความเข้าใจก่อนว่ามาตรฐานที่ขึ้นต้นด้วยอักษร "V" นี้ ไม่ใช่มาตรฐานของโมเด็มไปเสียทั้งหมด บางมาตรฐานอาจหมายถึงการเชื่อมต่อแบบอื่น ๆ ก็ได้ เช่น V.24 เป็นมาตรฐาน

การรับส่งข้อมูลแบบอนุกรมเทียบได้กับ RS-232C นั่นเอง และ V.35 หมายถึงการรับส่งข้อมูลแบบอนุกรมความเร็วสูง เป็นต้น ในที่นี้จะกล่าวถึงมาตรฐานของโมเด็มแบบต่าง ๆ ที่ใช้กันมากตาม CCITT(ITU-T) V-Series โดยทบทวนตั้งแต่ความเร็วต่ำไปจนถึงความเร็วสูง และรายละเอียดของแต่ละมาตรฐานดังนี้

V.21 เป็นมาตรฐานของโมเด็มความเร็ว 300 บิตต่อวินาที ใช้เทคนิคการผสมสัญญาณแบบ FSK (Frequency Shift Keying) รับส่งข้อมูลได้ในแบบ Full Duplex เป็นโมเด็มที่ใช้กับสายโทรศัพท์ ปัจจุบันนี้มีใช้กันน้อย เนื่องจากความเร็วในการรับส่งข้อมูลต่ำ

V.22 รับส่งข้อมูลด้วยความเร็ว 1,200 บิตต่อวินาที หรือลดความเร็วลงมาที่ 600 บิตต่อวินาที ได้การผสมสัญญาณใช้เทคนิคของ PSK(Phase Shift Keying) รับส่งข้อมูลในแบบ Full Duplex ใช้กับสายโทรศัพท์หรือสายตรงได้ ขึ้นอยู่กับโมเด็มว่าถูกออกแบบมาให้ต่อใช้กับสายตรงหรือไม่

V.22 bis รับส่งข้อมูลความเร็ว 2,400 บิตต่อวินาที หรือลดความเร็วลงมาที่ 1,200 บิตต่อวินาทีได้ การผสมสัญญาณใช้เทคนิค QAM รับส่งข้อมูลแบบ Full Duplex ใช้กับสายโทรศัพท์หรือสายตรงได้ V.22 bis เป็นมาตรฐานของโมเด็มที่เข้ามาแทนที่ V.22 ในสมัยก่อน เนื่องจากความเร็วสูง 2,400บิตต่อวินาที ซึ่งเร็วกกว่า V.22 ถึงสองเท่า

V.23 เป็นมาตรฐานที่คล้ายกับมาตรฐาน V.22 แต่รับส่งข้อมูลในแบบ Half Duplex คือมีความเร็ว 1,200 บิตต่อวินาที หรือลดความเร็วลงมาที่ 600 บิตต่อวินาที ใช้เทคนิคการผสมสัญญาณแบบ FSK ต่อใช้กับสายโทรศัพท์ก็ได้ มาตรฐาน V.23 นี้เราไม่ค่อยได้ใช้งานเท่าไหร่นัก เพราะว่าประสิทธิภาพของการรับส่งข้อมูลต่ำและเป็นการติดต่อแบบ Half Duplex จึงสู้มาตรฐานแบบ V.22 หรือ V.22bis ไม่ได้

V.26 เป็นมารฐานของโมเด็มสายตรงแบบใช้สาย 4 เส้น (4Wire คือ 2 คู่) รับส่งข้อมูลในแบบ Full Duplex ใช้เทคนิคการผสมสัญญาณชนิด PSK มีความเร็วในการรับส่งข้อมูล 2,400บิตต่อวินาที จะนำมาต่อใช้กับสายโทรศัพท์ไม่ได้ ดังนั้นเราจึงไม่ค่อยได้พบเห็นมาตรฐานนี้กันนัก ปัจจุบันก็มีใช้น้อยเนื่องจากความเร็วต่ำเกินไปสำหรับสายตรง ส่วนมากจะเลือกใช้มาตรฐานอื่นที่ความเร็วสูงกว่านี้

V.26bis เป็นมาตรฐานเหมือนกับ V.26 แต่สำหรับใช้กับสายโทรศัพท์แทน มีความเร็วในการรับส่งข้อมูลที่ 2,400 บิตต่อวินาที หรือลดความเร็วลงมาที่ 1,200 บิตต่อวินาทีได้ การรับส่งข้อมูลเป็นแบบ Half Duplex ใช้เทคนิคการผสมสัญญาณแบบ PSK

มาตราฐานนี้จึงสู้ V.22 bis ไม่ได้ โดยทั่วไปเราจะใช้ V.22 bis ที่ความเร็ว 2,400 บิตต่อวินาที Full Duplex มากกว่า

V.27 เป็นมาตรฐานสำหรับโมเด็มความเร็ว 4,800 บิตต่อวินาทีที่ใช้กับสายตรงเท่านั้น เทคนิคของการผสมสัญญาณเป็นแบบ PSK รับส่งข้อมูลในแบบ Full Duplex ได้ ซึ่งความเร็ว 4,800 บิตต่อวินาทีนี้ถือว่าเป็นความเร็วการรับส่งข้อมูลสูงที่สุดของเทคนิคการผสมสัญญาณแบบ PSK

V.27bis คล้ายกับมาตรฐานแบบ V.27 แต่ว่ารับส่งข้อมูลที่ 4,800 บิตต่อวินาที หรือ ลดความเร็วลงมา 2,400 บิตต่อวินาทีได้ ใช้สำหรับสายตรงแบบ 4 Wire เท่านั้น การผสมสัญญาณก็เป็นแบบ PSK สามารถรับส่งข้อมูลได้ทั้งในแบบ Full Duplex และ Half Duplex

V.27 ter เป็นมาตรฐานโมเด็มความเร็ว 4,800 บิตต่อวินาที หรือลดตวามเร็วลงมาที่ 2,400 บิตต่อวินาที สำหรับใช้กับสายโทรศัพท์ การรับส่งข้อมูลเป็นแบบ Half Duplex เท่านั้น เทคนิคการผสมสัญญาณชนิด PSK มาตรฐาน V.27 ter คล้ายกับ V.27 bis เพียงแต่ใช้กับสายโทรศัพท์แทนที่จะเป็นสายตรง มีใช้กันในเครื่องโทรสาร Group 3

V.29 เป็นมาตรฐานของโมเด็มที่ใช้กับสายตรงแบบ 4 Wire เท่านั้น การรับส่งข้อมูลใช้ได้ทั้ง Full Duplex และ Half Duplex สามารถรับส่งข้อมูลได้ตั้งแต่ 9,600 บิตต่อวินาที หรือลดความเร็วลงมาที่ 7200 บิตต่อวินาที และ 4800 บิตต่อวินาที ได้ ที่ความเร็ว 9600 บิตต่อวินาที จะใช้เทคนิคการผสมสัญญาณแบบ QAM ส่วนที่ความเร็ว 7200 และ 4800 บิตต่อวินาทีใช้การผสมสัญญาณแบบPSKมาตรฐาน V.29ใช้กันมากสำหรับการรับส่งข้มูลผ่านสายตรงระหว่างคอมพิวเตอร์กับคอมพิวเตอร์และใช้ในเครื่องโทรสาร Group 3

V.32 เป็นมาตรฐานโมเด็มสำหรับใช้กับสายโทรศัพท์ สารรับส่งข้อมูลได้ที่ความเร็ซ 9600 บิตต่อวินาที ในแบบ Full Duplex หรือลดความเร็วลงมาที่ 4800 บิตต่อวินาที มาตรฐาน V.32 นี้ยังใช้งานกับสายตรงแบบ 2 Wire ได้อีกด้วย เทคนิคการผสมสัญญาณเป็นแบบ QAM ทั้งที่ความเร็ว 9600 และ 4800 บิตต่อวินาทีการรับส่งข้อมูลความเร็วสูงผ่าน สาย 2 เส้น ของ V.32 ใช้เทคนิค Echo Cancellation แทนที่จะใช้การแบ่งความถี่อย่างในโมเด็มความเร็วต่ำ

V.32bis เป็นมาตรฐานโมเด็มสำหรับใช้กับสายโทรศัพท์ที่ปรับปรุงมาจากมาตรฐาน V.32 สามารถรับส่งข้อมูลได้ที่ความเร็ว 14,400 บิตต่อวินาที ใช้ได้กับทั้งสายโทรศัพท์และสายตรง เทคนิคการผสมสัญญาณเป็นแบบ Trellis Coding Modulation

(TCM) และลดความเร็วลงมาเชื่อมต่อกับ V.32,V.22 bis และ V.22 ได้ มีการใช้ Echo Cancellation ในการส่งข้อมูลผ่านสาย 2 เส้นเช่นเดียวกัน V.32 bis นับเป็นมาตรฐานโมเด็มของ CCITT หรือ ITU-T ที่มีผู้ใช้กันมากชนิดหนึ่งในปัจจุบัน

V.32 terbo เป็นมาตรฐานโมเด็มความเร็วสูงที่ไม่ได้รับการรับรองจาก CCITT หรือ ITU-T แต่เกิดจากผู้ผลิตโมเด็มรายใหญ่ตกลงมาตรฐานกันเอง โดยทำการปรับปรุงจาก มาตรฐาน V.32bis เล็กน้อย V.32terbo มีความเร็วในการรับส่งข้อมูล 19,200 บิตต่อ วินาที และมีคุณสมบัติอื่น ๆ เหมือนกับ V.32 bis ทุกอย่าง ต่างกันที่ความเร็วในการรับส่งข้อมูล เพิ่มขึ้นเท่านั้น โดยทั่วไปโมเด็มแบบ V.32bis จะรับส่งข้อมูลแบบ V.32terbo ได้ด้วย เราจึงเห็น แต่ V.32 bis ในท้องตลาดเท่านั้น

1.10 หลักการทั่วไปของ V.34

หนึ่งในหนทางสำคัญของ V.34 ซึ่งปรับปรุงมากจาก V-series ก่อนๆ ในทางที่จะใช้ในระบบ GSTN (General Switched Telephone network) ซึ่งการติดต่อสื่อสารทุกวันนี้นั้นต้องการการติดต่อที่มีความเร็วสูงและมีความผิดพลาดน้อย

ในการส่งสัญญาณภายใต้ระบบ GSTN นั้นมีข้อกำหนดมากมาย เช่น ในการติดต่อที่ใช้ Adaption Differential Pulse Code Modulation (ADPCM) ที่ 32 หรือ 40 kd/s สำหรับการส่งสัญญาณในสายโทรศัพท์ให้มีประสิทธิภาพ ซึ่งมาตราฐาน ITU-T สำหรับ ADPCM จะ support ประมาณดังนี้ SNR ประมาณ 21 dB ที่ 32 kb/s (G.721) หรือ ประมาณ 28 dB ที่ 40 kb/s (G726) ซึ่งโดยทั่วไปการส่งที่ 32 kb/s ADPCM encode / decode (codecs) ที่ support ในช่วง Bandwidth ที่น้อยกว่า 3200 Hz ที่ SNR ประมาณ 28 dB ซึ่งเป็นช่องที่ใช้งานมาก

โดยส่วนใหญ่ในการส่งสัญญาณ Analog แบบเก่า ๆ จะใช้ค่า bandwidth ที่ไม่ค่อยดีเท่าไร ซึ่งก็จะทำให้ SNR มีค่าไม่ดีตามไปด้วย เช่น ตัวอย่าง เช่น ในอเมริกาเหนือ ที่ยังคงใช้มาถึง 40 ปี N - carrier System ซึ่งจะ support น้อยกว่า 3000 Hz ของ band width ที่ SNR ประมาณ 24-28 dB ซึ่งอาจทำให้เกิดผลเสีย เช่น เกี่ยวกับ frequency offset phase jitter

ใน V.34 นั้นไม่เพียงแต่จะใช้การ Modutation แบบใหม่ (TCM) modem ที่เราติดต่อ ซึ่งทำให้ดีกว่า V.32 bis ในการใช้งานในการติดต่อ

ใน V.34 นั้นในการติดต่อตอนเริ่มต้นนั้น initial startup ใน bandwidth และ bit rate ที่สามารถใช้งานด้วยกันได้ (กับ V.serie อื่น ๆ)

ซึ่งผลของการที่ออกแบบ V.34 modem เป็นการที่ดีมากในการที่จะติดต่อ กันที่ bit rate สูง ๆ (ประมาณ 28.8 kb/s) ซึ่งโดยหลักการทั่วไปของ V.34 มีดังนี้

1.10.1 Adaptive Bandwidth

V.34 มีการปรับเปลี่ยน Bit rate ได้ ซึ่ง Bit Rate ที่เปลี่ยนไปนั้นทำให้มีประสิทธิภาพในการส่งข้อมูลที่ดีที่สุดในการติดต่อกับ modem อื่น ๆ ซึ่งการที่มีการปรับเปลี่ยน Bit Rate ได้ที่ทำให้มีการเปลี่ยน Band width ที่ดีที่สุดที่สามารถ จะติดต่อได้

ซึ่ง V.34 จะใช้ Quadratore amplitude modulation (QAM) ซึ่งองค์ประกอบ 2 องค์ประกอบของ 2-dimensional symbol คือ Amplitude-modulate บน in-phase และ quadrature sinusoidal carrier ที่ common carrier frequahcy ซึ่ง Nominal Nyquist bandwidth จะสมมูลกับ symbol rate และมีค่า band ที่ carrier frequency

ในมาตรฐาน modem รุ่นก่อนเช่น V.32 bis จะมี nominal band width (symbol rate) คงที่อยู่ที่ 2400 Hz โดยที่ค่าคงที่ของ frequency carrier มีค่า 1800 Hz ทำให้มีผลของ mominal transmittion band ที่ 600-3000 Hz แต่ใน V.34 นั้น Band width และ carrier frequency นั้น สามารถปรับเปลี่ยนได้ โดยค่าสูงสุดของ Band width มีค่าถึง 3429 Hz

ตัวอย่างเช่น ที่ rate 8 b/s/Hz (8 bits per QAM symbol) symbol rate มีค่า 2400 Hz และ bit rate มีค่า 19,200 b/s ในขณะที่ในการส่ง symbol rate ที่ 3429 Hz จะทำให้ได้ค่า bit rate 27,429 b/s ซึ่งในการที่เราส่ง bit rate ที่สูง ๆ การใช้ Band width จะต้องมีประสิทธิภาพมากตามไปด้วย

ใน 6 symbol rate (ซึ่งเป็น spec ใน V.34) มีค่าต่าง ๆดังนี้ : 2400, 2743, 2800, 3000, 3200, และ 3429 Hz ที่ใช้งาน และอีก 3 symbol rate ที่ใช้ในการอื่น ๆคือ 2400, 3000 และ 3200

ใน symbol rate และ carrier frequency นับเป็นจุดเริ่มต้นในการติดต่อ ซึ่งในตัวส่ง (Transmitter) จะส่ง "line probing" sequence ซึ่งจะทำให้ได้ค่า tone ที่ทำให้ได้ค่าสูงสุดของ band ดังนั้น SNR จะสามารถวัดได้โดย function ของความถี่ ซึ่ง symbol rate และ carrier frequency จะเป็นตัวที่ทำให้ได้ค่าที่ดีที่สุดที่จะทำการติดต่อกันได้ใน symbol rate ที่ยอมรับกันทั้ง 2 ฝ่าย

1.10.2 Adaptive Bit rate

V.34 สามารถ support bit rate ที่ 2.4 kb/s ถึง 28.8 kb/s และอาจสูงถึง 33.6 kb/s (ในการพัฒนาต่อไป) bit rate ที่ใช้งานกันอาจไม่เป็นเลขจำนวนจริงในหน่วย bit/symbol (ตัวอย่างเช่น 33.6 kb/s symbol rate มีค่า 3429 ซึ่งจะได้ 8.4 b/symbol) ซึ่งจะต้องใช้เทคนิคในการ mapping ที่ใหญ่ "Super frame" ซึ่งเพื่อที่จะได้ค่าที่เหมาะสมของ bit rate และsymbol rate

bit rate จะถูกเลือกโดย เครื่องรับ ซึ่งประมาณได้เป็นค่า สูงสุดของ bit rate ซึ่งสามารถ support ที่ law bit error probability เช่น 10^{-5} - 10^{-6}

Symbol Rate, S	a	c
2400	1	1
2743	8	7
2800	7	6
3000	5	4
3200	4	3
3429	10	7

ตารางที่ 1.1 Symbol rate

1.10.3 Trellis Coding

หนึ่งในหลักการที่เริ่มใช้มาตั้งแต่ V.32 9.6 kb/s modem standard (1984) เป็นการใช้ Trellis Code Modulation (TCM) ซึ่งในขณะนั้น TCM เป็นหลักการที่ใหม่อยู่และได้ใช้หลักการ Simple eight-state two-Dimentional (2 - D) trellis code ซึ่งCode ที่จะทำให้ได้ effective coding gain ประมาณ 3.6 dB แต่ Trellis Code ที่ใช้ใน V.34 เป็นแบบ four-dimentional (4-D) code ซึ่ง 4-D code จะมี constellation expansion ที่เล็กกว่า ซึ่งจะมีค่า Gain code ที่ดีมาก ๆ ประมาณ 4.2 dB หรือ ดีกว่า V.32 อยู่ 0.6 dB ซึ่งในการ decode นั้นจะใช้หลักการเดียวกัน ซึ่งค่าต่าง ๆ ของการ encode สามารถดูได้จากรายละเอียดใน V.34

1.10.3.1 Equalization and Precoding

ใน modem ที่มีความเร็วสูงเช่น V.32 bis จะใช้ adaptive Linear equalizen ใน reciever ที่ combat ISI ที่ modem ชนิดนี้ จึงการส่ง band width จะมีของเขตที่ "sweet spot" ของ 2400 Hz หรือน้อยกว่าจึงเป็นสิ่งที่รู้กันในเรื่อง channel attenuation will not be too severe

แต่ใน V.34 จำเป็นที่จะต้องใช้ bandwidth ที่มีประสิทธิภาพมากที่สุด จึงจำเป็นต้องการความถี่ที่ใกล้ bandedge ซึ่งจะมีค่า attenuation 10-20 dB ดังนั้นสิ่งที่ควรรู้คือ Decision-Feedback equalizer (DFE) อย่างไรก็ตามมันยังไม่สามารถที่จะ combine code ด้วย DFE ได้ ซึ่งยังเป็นปัญหาในการนำส่วน Feed back ของ DFE ที่จะมาใน transmitter ซึ่งต่อมาได้ถูกคิดค้นโดย Tomlinson และ Harathima ซึ่งเรียกว่า "equalization via precoding"

1.10.4 V 34 Transmitter

จากการส่งข้อมูลในระหว่างภายใน V34 ได้แสดงไว้ในรูปที่ 3.1จากรูปพบว่า รายละเอียดและข้อมูลของ bit rate และ symbol rate (28.8 kb/s ที่ 3200 Hz) และเครื่องส่งจะต้องส่งจำนวนบิทต่อ

QAM symbol ถ้าหากเลือก shaping แล้ว QAM Constellation จะขยายเป็น 75 เปอร์เซ็น ซึ่งจะเกิดกว่า Constellation ที่ต้องการนำไปรองรับการส่งแบบไม่เข้ารหัส และจะใช้เพียง 50 เปอร์เซ็นในระหว่างการเข้ารหัสแบบ trellis coding และ 25 เปอร์เซ็น ที่ shaping ที่ 9 b/symbol ดังตัวอย่าง เช่น สัญญาณ 896 point signal constelllation จะต้องการมากกว่า 512 point Constellation ที่ต้องการ กับการไม่เข้ารหัส

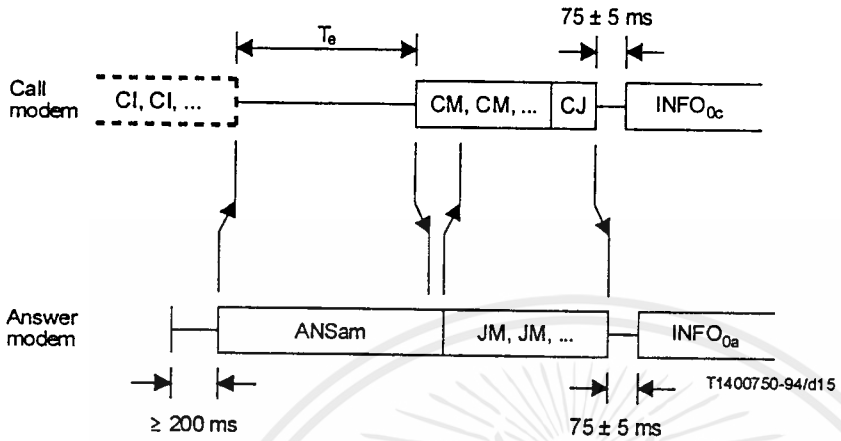
สำหรับในกรณีของ shell mapping จะทำการแบ่ง signal constellution ออกเป็นสัญญาณที่เป็นลักษณะวงกลม (Concentric ring) โดยแต่ละวงจะมี 64 point และแบ่งเป็น 14 ขนาดที่เท่า ๆ กัน ในการเข้ารหัส จะทำการรวบรวม user bits ไว้ใน กรอบที่มีขนาด 72 บิท ที่เกินกว่า 8 symbol periods. ในที่นี้ 28 user bits จะถูกนำมาใช้โดย shell mapper ซึ่งจะทำการเลือก 1 ใน 14 วง (ring) ของแต่ละ 8 QAM symbols ในกรอบของ user bits. โดยกลุ่มของวง (rings) จะถูกเลือกเป็น 16-dimensional signal และทำเป็นกึ่งวงกลม 16-D region. และในส่วนที่เหลือ 44 user bits จะถูกทำเป็น 4 coded โดย trellis encoder และจากนั้นจะใช้เวลาในการเลือก 1 ใน 64 จุด สัญญาณ u(n) ในการเลือกแต่ละวง (ring) ใน user bits บางส่วนจะเป็น differntially encoded เพื่อให้แน่ใจว่าระบบ มีการทำงานอย่างสม่ำเสมอใน precode และ trellis encoder นั้นจะถูกเชื่อมต่อในลักษณะแบบ feedback ดังแสดงในรูปที่ 2 โดย กระแสเอาท์พุทของ trellis encoder จะกำหนดเซตของ 4-D signal point pairs ว่าอาจจะถูกเลือกโดย precoder โดยที่ precoder จะเป็นตัวเลือก 1 ใน 4-D pair และ encoder ทำการส่งให้สอดคล้องกับคู่นั้น ๆ ดังนั้น จึงเป็นการกำหนดของ state ต่อไป

ในการจัดการนี้ จะบวกถึงระยะทางที่น้อยที่สุด ("dither") โดย d(n) = x(n)-u(n) ว่าต้องการเป็นตัวถูกรวมกับสัญญาณ u(n) โดย precode จะกำเนิดสัญญาณการส่ง x(n) โดย dither sequonce จะถูกเลือกหลังจากการส่งผ่านช่วงสัญญาณ ซึ่งเป็นที่รู้จักในชื่อของ linear distortion และเอาท์พุท sequence จะเท่ากับ valid trellis code sequence บวกกับสัญญาณรบกวน (noise) และ เนื่องจากสามารถถูกถอดรหัส โดยมาตราฐาน trellis decoder ค่าเฉลี่ยกำลังการส่งที่น้อยที่สุดของ précoder จะรวบอยู่ด้วย และแน่ใจว่า input data อาจจะถูกแสดงออกมาจากการถอดรหัสข้อมูลในเครื่องรับโดยการ feed-back free ดังนั้น error propagation จะถูกจำกัด

หลังจากทำการ precoding แล้ว sequence x(n) จะถูกปรับปรุงแก้ไขโดย a nonlinear encoder x(n) จะถูก filter โดย pulse-shaping fitter. ซึ่งขึ้นอยู่กับการเลือก โดยเครื่องรับ ซึ่งอาจเป็นมาตราฐาน square-root-of Nyquist filter กับ no spectral shaping หรือ filter ซึ่ง ขึ้นอยู่กับ pre-emphasis ตาม 1 ใน 5 spectral shapes ที่นิยามใน V.34 และ เอาท์พุทของ pulse-shaping filter จะถูก modulate ที่ความถี่พาหะที่ถูกเลือก และระดับกำลังของการส่งที่สายโทรศัพท์

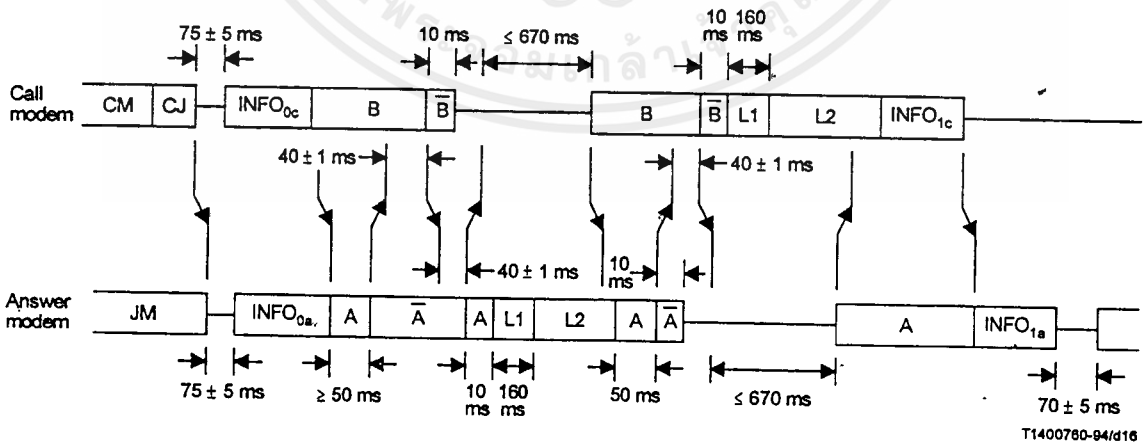
1.10.5 START-UP AND OPERATING PROCEDUPIES

ตามคำแนะนำ ของ V.34 dulex startup ประกอบด้วย 4 เฟส รูปที่ 1.7 แสดง ถึง seguence



รูปที่ 1.7 Network interaction with a CM/JM exchange

1.10.5.1 เฟสที่ 1 The network interaction เฟส ซึ่งเป็นพื้นฐาน ในการแนะนำของ V 8 โดยจะมีวัตถุประสงค์ เป็น disable network echo suppressors และ concellers เพื่อเตรียมสำหรับ จำกัดเทอร์มิเนอร์ ในการเลือกระหว่าง modem fax video text และ text telephone เพื่อเตรียมการปรับปรุงแก้ไข "automoding" และอนุญาตให้ network circuit multiplication equipment (CME) เพื่อทำงานเป็น demodulation/remodulation (demod/remod)



รูปที่ 1.8 Probing/Ranging

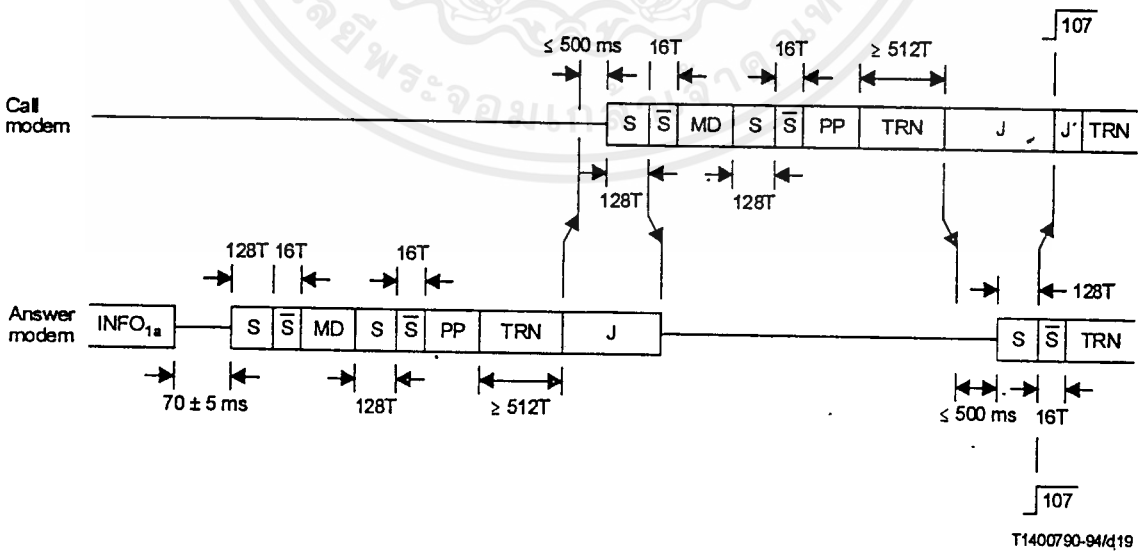
1.10.5.2 เฟสที่ 2 the ranging และ probing phase ซึ่งเป็นส่วนประกอบของ initial information exchange (INFO.) , ranging และ probing sequence และ การแลกเปลี่ยนข้อมูลข่าวสาร ครั้งที่ 2 ($INFO_1$) การเปลี่ยนข้อมูลข่าวสาร จะใช้ 600 bit/s Frequency-division multiplexed (FDM) differential phase shift keying (DPSK) modulation ที่ความถี่พาหะ 1200Hz และ 2400 Hz

INFO. จะถูกใช้ในการส่งความจุของข่าวสาร เช่น symbol rates ที่ถูกรองรับ regulatory bandwidth restrictions และการอนุญาตสูงสุดในการส่งและรับ ของ symbol rates.

Ranging จะใช้เฟสที่กลับในระหว่างโทนการส่งที่กำหนด the round-trip delay ของการเชื่อมต่อ ของระยะ echo canceller taps ของโมเด็ม

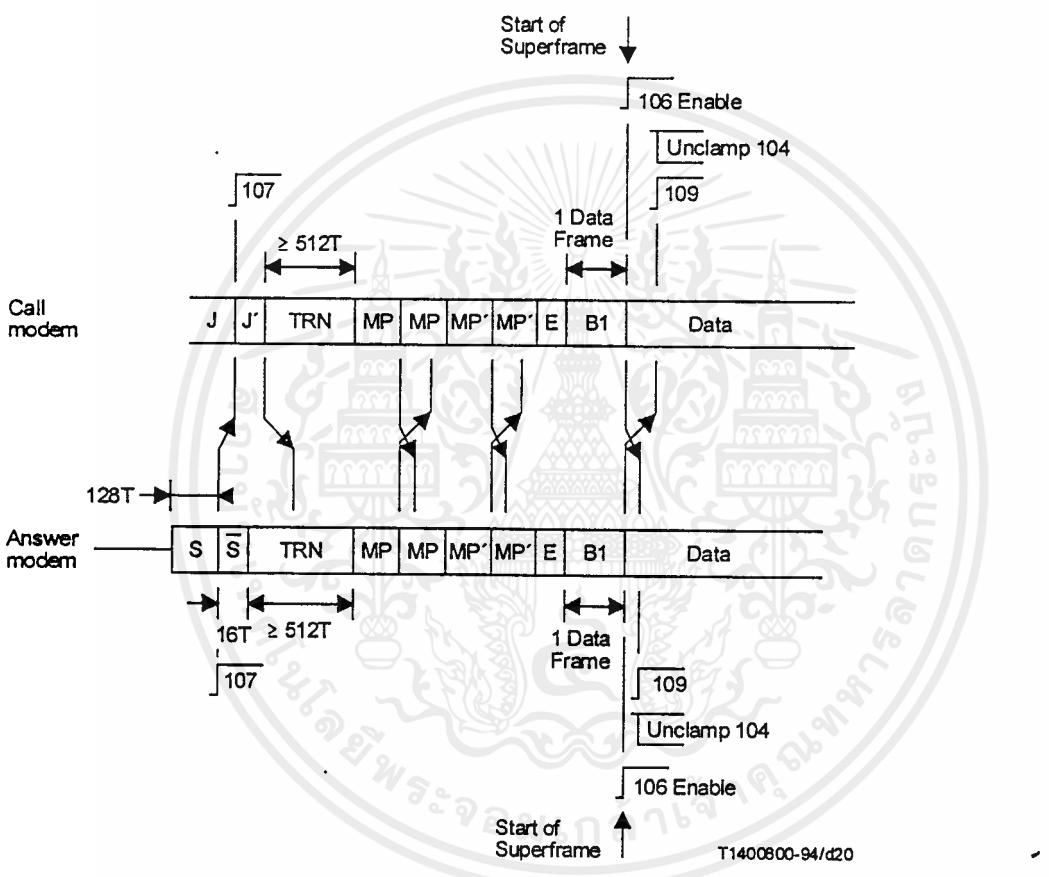
Probing จะถูกใช้เพื่อกำหนด ลักษณะเฉพาะตัวของช่องสัญญาณ ใน probing signal จะประกอบด้วย เซตของโทนที่มีช่วงแอมพลีจูดที่เท่ากัน 150 Hz ถึง 3750 Hz ในสัญญาณนี้อาจถูกใช้ในการวัดจำนวนของ Amplitude distortion across the band the SNR across the band และความถี่ออฟเซตสำหรับการจัดให้ช่องสัญญาณ โทนที่มีความถี่ 900, 1200, 1800 และ 2400 Hz จะถูกข้ามในการวัดของระดับของ intermodulation distortion . และสัญญาณ probing แรกจะถูกส่งที่ 6 dB มากกว่า nominal power level และที่ nominal power level จะอนุญาตการวัดของ nonlinearity ทั้งหมด

INFO ถูกใช้ในการส่งผลของการวัดของ probing ในเฟรมของ projected maximum bit rate และ เลือก symbol rate, ความถี่พาหะ pre-emphasis filter และช่วงของการลดกำลังงานที่ถูกใช้โดยโมเด็ม



รูปที่ 1.9 Equalizer and Echo Conceller Training

1.10.5.3 เฟส 3 The equalizer และ echo-canceller half-duplex training phase, ซึ่งจะประกอบด้วยชุดแรกของสัญญาณที่ส่ง โดย answer modem และ Call modem ตามลำดับ ในแต่ละชุดจะประกอบด้วย manufactorer-defined echo canceller training signal, periodic sequence สั้น ๆ สำหรับ equalizer training sequence ของ scramble ไบนารี 1s สำหรับการปรับละเอียดของ equalizer และ echo canceller และการ scrambled sequnce 16-bit ซ้ำอีกครั้ง

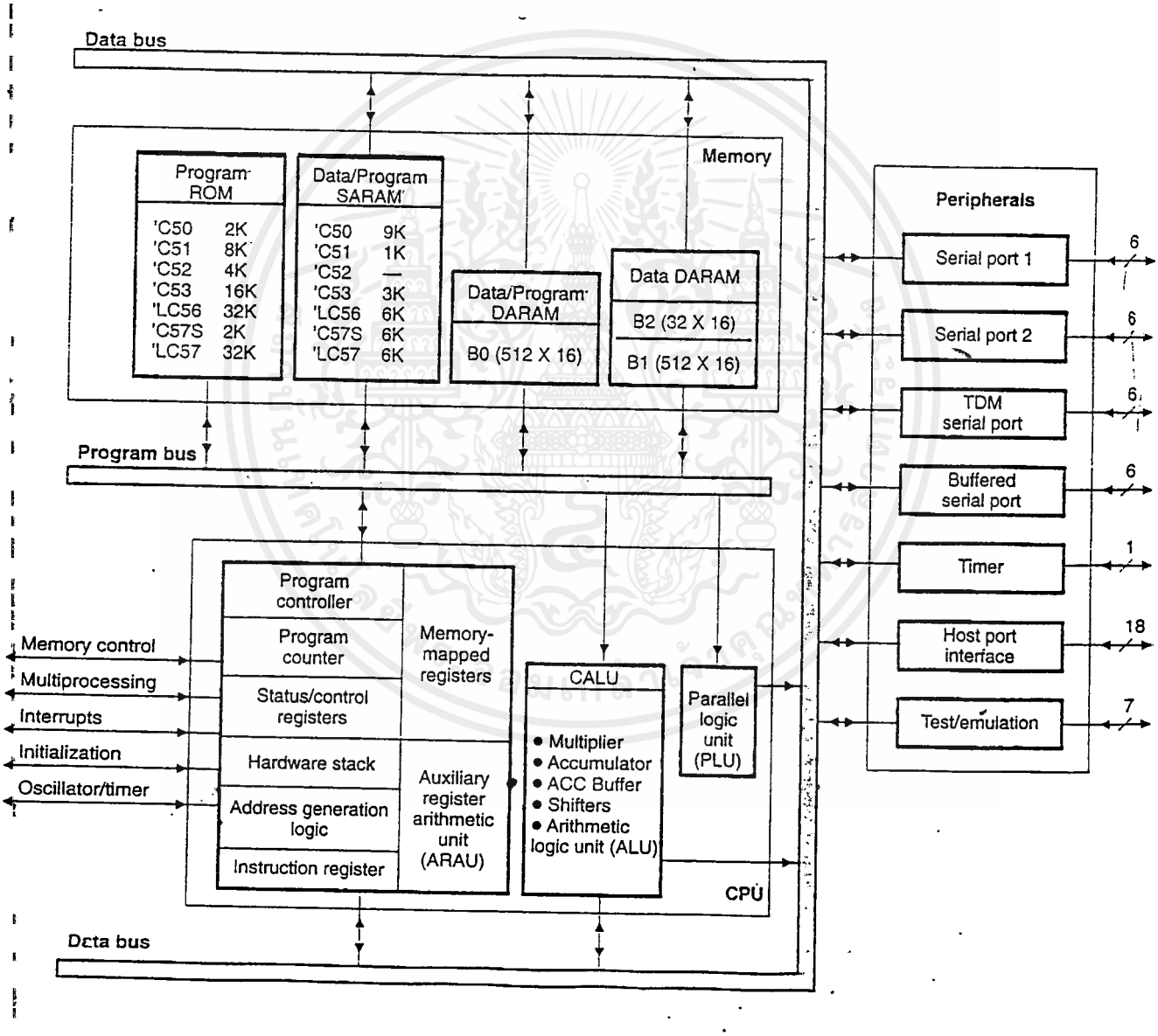


รูปที่ 1.10 Final Training

1.10.5.4 เฟสที่ 4 เป็น final duplex training phase, ซึ่งประกอบด้วย sequence ของ scramble binary 1s และการแลกเปลี่ยนมอดูเลชั้น พารามิเตอร์ ซึ่งจะเลือกลักษณะการมอดูเลชั้น ในส่วน sequence ของ scramble binary 1s จะใช้ทั้ง 4 หรือ 16-point QAM constellation และถูกใช้ใน train precoder coefficient และการปรับละเอียดของ equalizer และ echo canceller

บทที่ 2 TMS320C50

TMS320C5X เป็น CPU ในตระกูล Digital Signal Processors (DSPs) โดยตระกูล TM 320C5X ได้ปรับปรุงชั้นจาก 'C1X' และ 'C2X' โดยคุณสมบัติโครงสร้างพื้นฐานของ CPU C5X ทั่วไปจะคล้ายกับ CPU รุ่น C25 แต่ได้มีการเพิ่มสถาปัตยกรรม enhancements เข้าด้วยจึงเป็นผลให้ประสิทธิภาพการทำงานของ CPU C5X นี้เพิ่มความเร็วขึ้นเป็นประมาณ 2 เท่า



ตารางที่ 2.1 The' C5X gonceralion of static CMOS DSPs conists of the following devics

โดยทั่วไปตระกูล ประกอบไปด้วย C50, C51, C52, C53, C535, C6, C57 และ C575 โดยจากการใช้เทคโนโลยี CMOS และสถาปัตยกรรมพื้นฐานแบบรุ่น C25 จะทำให้มีความเร็วในการทำงานสูงมาก ซึ่งประมาณ 50 ล้านคำสั่งต่อวินาที (50 MIP'S)

2.1 คุณสมบัติโดยทั่วไปของ TMS 320C50

2.1.1 คอมแพททิบิลิตี้ (Compatibilily) ซอสโค๊ด (Source code) จะคอมแพททิเบิ้ลกับรุ่น C1X, C2X, C2XX

2.1.2 ความเร็ว (Speed) 20 ใช้ความเร็วในการทำงานแต่ละคำสั่งเพียง 20 nS และค่าส่งจะใช้เพียง 1 Cycle เท่านั้น

2.1.3 หน่วยความจำ

2.1.3.1 244 k-word x 16-bit mascimem addressable external menory space

2.1.3.2 1056-word x 16 bit dual-accen on-chip data RAM

2.1.3.3 9k-word x 16-bit single-accen on-chip program / 2ate RAM

2.1.3.4 2k-word x 16-bit single-accen on-chip boot ROM

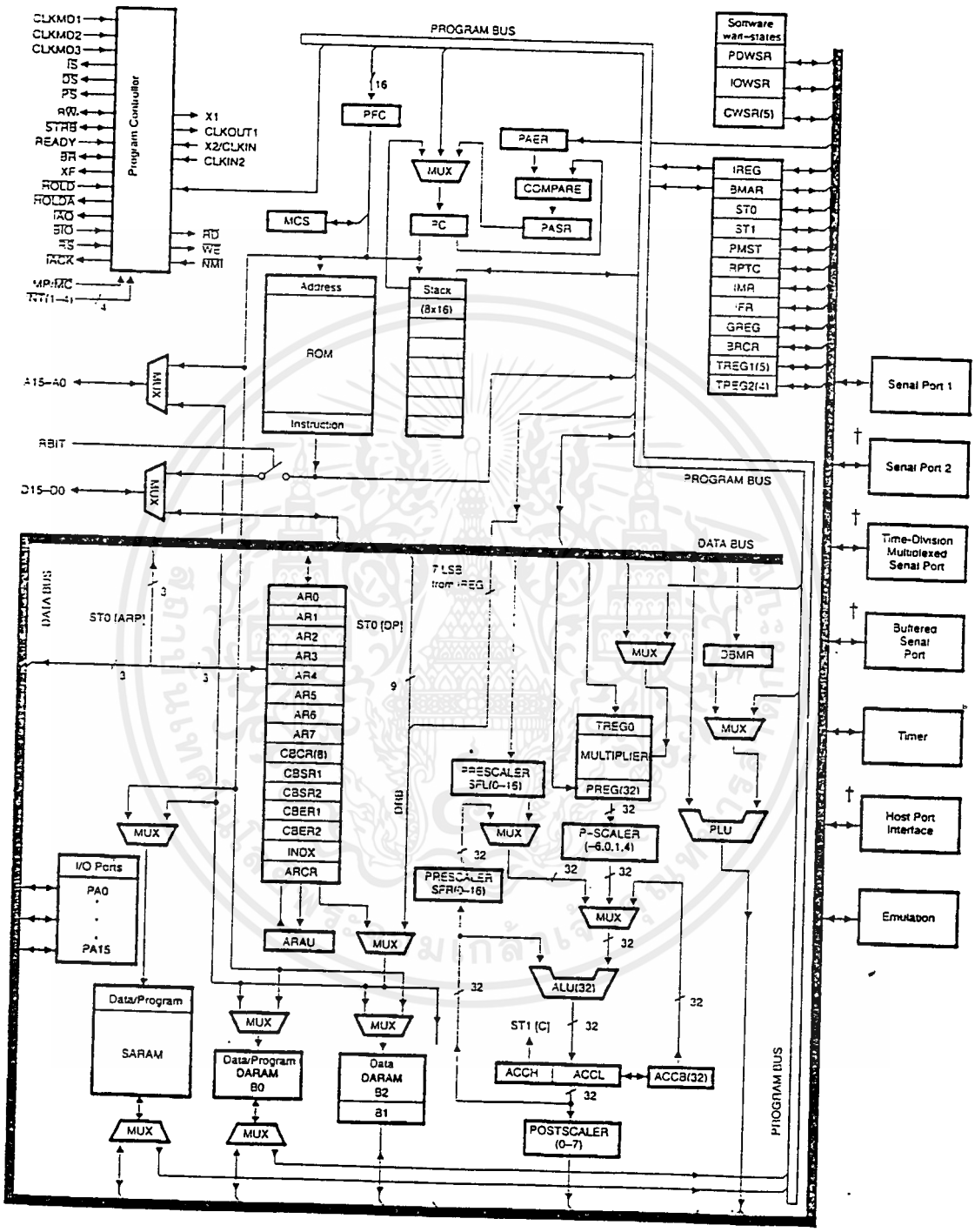
2.2 โครงสร้างโดยทั่วไปของ TM S320C50 DSP

สถาปัตยกรรมของ TM S320C50 สร้างขึ้นมาเพื่อความเร็วในการทำงานและโครงสร้างที่ทำงานได้ด้วยขีดความสามารถที่สูงขึ้น และเพื่อให้การทำงานของบัสไม่ขึ้นต่อกัน ซึ่งแยก บัส เป็น บัส ของโปรแกรมและบีตของข้อมูลออกจากัน โดยบัสของโปรแกรมจะเป็นทางเข้าออกรหัสคำสั่งและ Oper and ของคำสั่ง ส่วยบัสของข้อมูลจะเชื่อมต่อโดยตรงวงจรการทำงานการประมวลผล เช่น CALU (Central Arithmetic Logic Unit) และ Register ที่เป็น File ข้อมูลย่อย ARO-AR7 และยังมี ARAU (Auniliary Register Arithmetic Unit)

2.3 หน่วยประมวลผลกลาง (CPU)

CPU ของ TM S320C50 DSP จะประกอบด้วย

- หน่วยประมวลผลทางคณิตศาสตร์ และลอจิก (CALU)
- หน่วยลอจิกขนาน (PLU)
- Auxiliary register arithmetic unit (ARAU)
- Memory-mapped registers
- ตัวควบคุมโปรแกรม

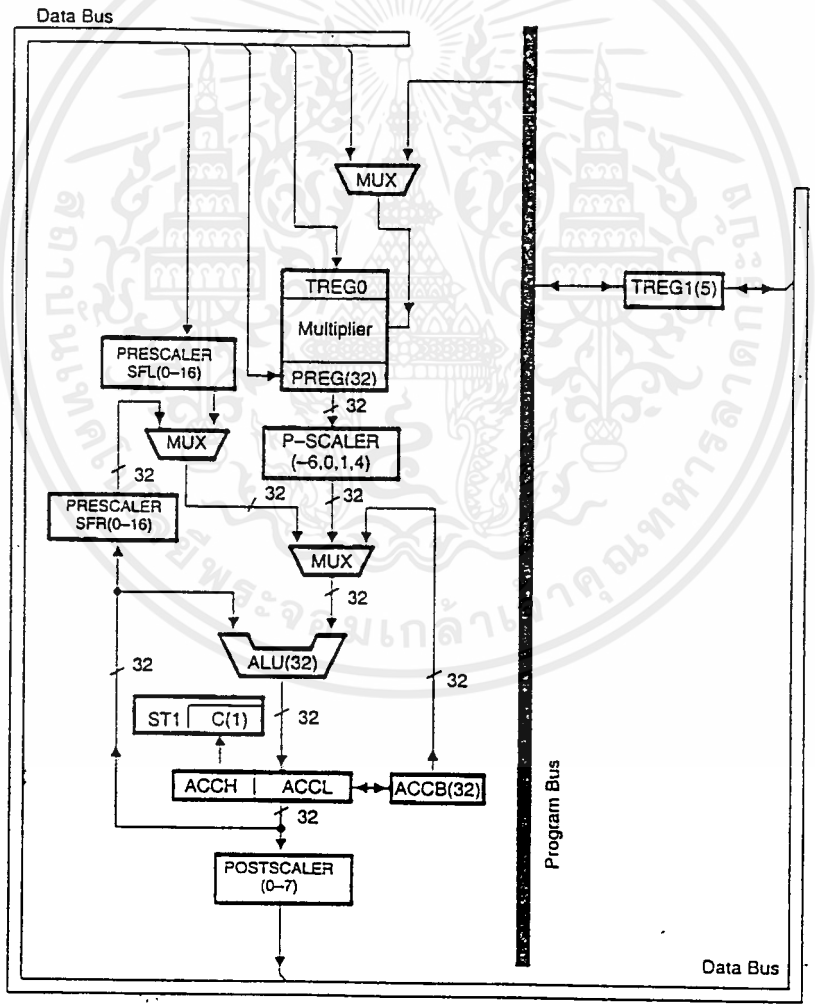


รูปที่ 2.1 Block Diagram of 'C5x DSP-Central Processing Unit (CPU)

2.3.1 หน่วยประมวลผลกลางทางคณิตศาสตร์และลอจิก (CALU)

ภายใน CPU จะใช้ CALU ในการแสดงผลทางคณิตศาสตร์ แบบ 25-complement ซึ่ง CALU จะมีคุณสมบัติดังนี้

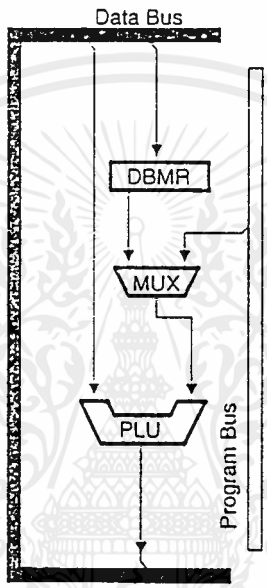
- วงจรคูณ X 16 บิท X 16 บิท
- หน่วยประมวลผลทางคณิตศาสตร์และลอจิก 32 บิท
- แอคคูมิวเลเตอร์ (Acc) 32 บิท
- บัฟเฟอร์แอคคูมิวเลเตอร์ (AccB) 32 บิท
- ทำการรวบเอาท์พุทของแอคคิวมูเลเตอร์ ทั้ง 2 และ TREG



รูปที่ 2.2 Central Arithmetic Logic Unit

2.3.2 หน่วยลอจิกขนาน (PLU)

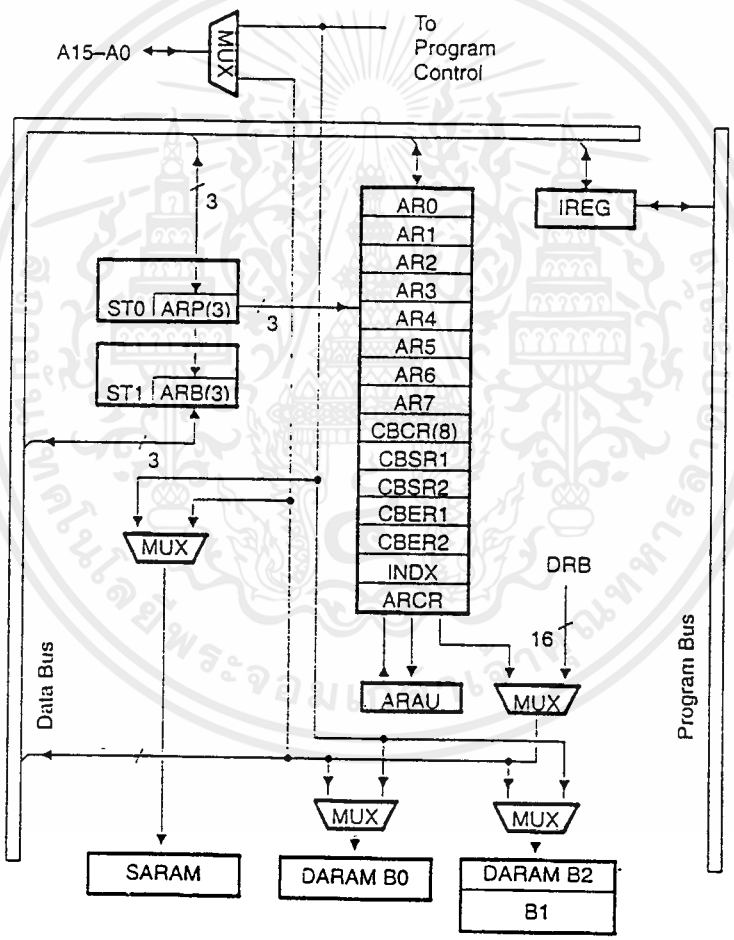
ภายใน CPU จะมี PLU ที่อิสระต่อกัน ซึ่ง PLU จะแสดงการดำเนินการ Boolean หรือบิทที่มีความต้องการใช้ของตัวควบคุมความเร็วสูง PLU จะสามารถ Set, clear, test หรือ toggle บิทต่าง ๆ ใน status register control register หรือที่ตั้งของ data memory ต่าง ๆ ได้ PLU จะทำการเตรียมการดำเนินงานทางลอจิกโดยตรงที่ค่า data memory โดยปราศจากผลกระทบของ Acc หรือ PREG



รูปที่ 2.3 Parallel Logic Unit Block Diagram

2.3.3 Auxiliary Register Arithmetic Unit (ARAU)

ภายใน CPU จะมีหน่วยประมวลผลทางคณิตศาสตร์และลอจิก 16 บิท แบบ unsigned ซึ่งจะทำการคำนวณแอดเดรททางอ้อมโดยการใช้อินพุทจาก auxiliary registers (Ars), index register (INDX) และ auxiliary register compare register (ARAU). ใน ARAU สามารถ autoindex ที่ AR ซึ่ง data memory location จะกลายเป็นแอดเดรท และสามารถบ่งชี้โดย หรือ ± 1 โดย INDX



รูปที่ 2.4 Auxiliary Register Arithmetic Unit

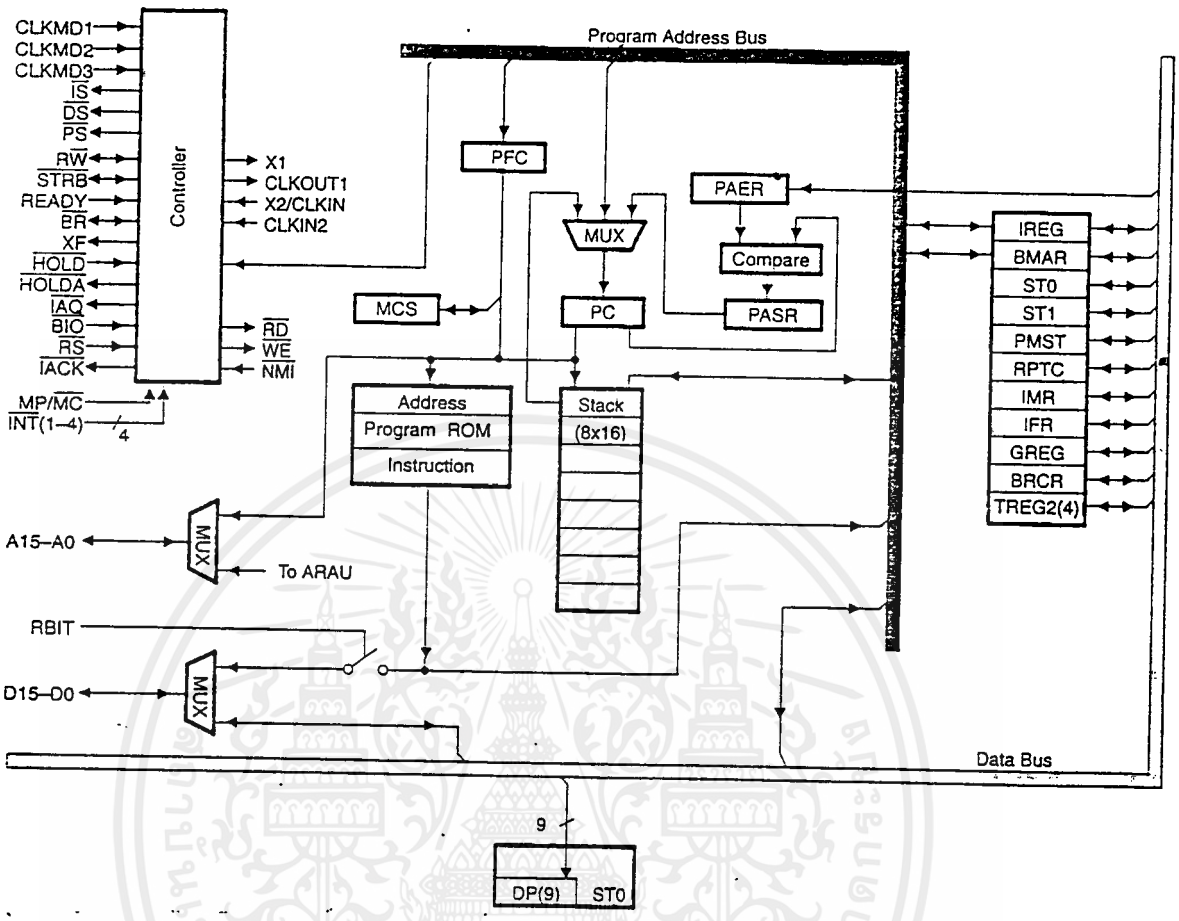
2.3.4 Memory-Mapped Register

TMS 320C50 DSP จะมี 96 registers mapped ที่ศูนย์ของช่องว่าง data memory ซึ่ง TMS 320C50 DSP จะมี 28 CPU registers และ 16 อินพุท/เอาท์พุท พอร์ท registers แต่จะมีความแตกต่างของจำนวนของ register ที่อยู่รอบนอก และ register เสริม ซึ่ง memory-mapped register เป็นส่วนประกอบของช่องว่าง data memory ซึ่งสามารถเขียนและอ่านจาก data memory location. ใน memory-mapped registers จะถูกใช้ในการชี้ data address ทางอ้อม, temporary storage, CPU status and control , หรือการประมวลผลทางคณิตศาสตร์โดยผ่าน ARAU

2.3.5 ตัวควบคุมโปรแกรม

ในตัวควบคุมโปรแกรมจะประกอบด้วยวงจรลอจิกที่ทำการถอดรหัสทางโครงสร้าง, ทำการจัด CPU, รักษาสถานะการดำเนินงานของ CPU, และถอดรหัสของเงื่อนไขการดำเนินงาน. ในตัวควบคุมโปรแกรมยังประกอบด้วย

- โปรแกรมเคาท์เตอร์
- status and control register
- ฮาร์ดแวร์แสตค
- Address generation logic
- Instruction register



รูปที่ 2.4 Program Control Functional Block Diagram

2.4 On-Chip Memory

ในสถาปัตยกรรมของ TMS 320C50 มีการพิจารณาจำนวนของ on-chip memory ที่ใช้ช่วยในระบบประกอบด้วย

- Program read-only memory (ROM)
- Data/program dual - access RAM (DARAM)
- Data/program single - access RAM (SARAM)

ใน TMS 320C50 มีผลรวมของแอดเดรสอยู่ในช่วง 224k word X 16 บิท ในช่องว่างของหน่วยความจำจะถูกแบ่งเป็น 4 ส่วน : 64 k-word program memory space, 64k-word locat data memory space, 64 k-word input/output ports, และ 32 k-word global data memory space.

2.4.1 Program ROM

ใน TMS 320C50 จะมี boot loader code อยู่ประจำใน On-Chip ROM. ซึ่งหน่วยความจำนี้จะถูกใช้สำหรับ booting program code จาก เอ็กเทอร์นัล ROM หรือ EPROM อย่างช้า ๆ และเร็วขึ้นที่ On-Chip หรือเอ็กเทอร์นัล RAM

The on-chip ROM อาจจะมีหรือไม่มี boot loader code ก็ได้ แต่ถึงอย่างไรก็ตาม On-Chip ROM ถูกเตรียมไว้สำหรับโปรแกรมเฉพาะ

2.4.2 Data/Program Dual-Access RAM

ใน TMS 320C50 จะมี 1056-word 16 bit on-chip dual-accress RAM (DARAM). ซึ่ง DARAM จะถูกแบ่งแยกได้ 3 ส่วน : 512-word data หรือ program DARAM block BO, 512-wrod data DARAM block B1, และ 32-word data DARAM bolck B2. ใน DARAM มีหน้าที่ในการเก็บรักษาค่าของข้อมูล แต่ก็สามารถถูกใช้ในการรักษาโปรแกรมด้วย DARAM bolcks B1 และ B2 จะถูกรักษา data memory เสมอ อย่างไรก็ตาม DARAM block B0 สามารถถูกกำหนดโดยซอฟท์แวร์ data หรือ program memory

2.4.3 Data / Program Single-Access RAM

ใน TMS 320C50 จะมี 16-bit on-chip single-access RAM (SARAM). ใน SARAM สามารถถูกกำหนดโดยซอฟท์แวร์ได้ 1 ใน 3 ทาง

- SARAM ทั้งหมดถูกกำหนดโดย data memory
- SARAM ทั้งหมดถูกกำหนดโดย program memory
- SARAM ถูกกำหนดโดยทั้ง data memory และ program memory

SARAM จะถูกแบ่งได้เป็น 1k และ 2k-word block ต่อเนื่องในแอดเดรสที่เป็นช่องว่างของหน่วยความจำ ใน SARAM สามารถทำการ mapped ได้ทั้ง program และ data memory space

Bit values			ROM	SARAM	DARAM B0	
CNF	RAM	MP/$\overline{MC}$	(2K-words)	(9K-words)	(512-words)	Off-Chip
0	0	0	0000–07FF	Off-chip	Off-chip	0800–FFFF
0	0	1	Off-chip	Off-chip	Off-chip	0000–FFFF
0	1	0	0000–07FF	0800–2BFF	Off-chip	2C00–FFFF
0	1	1	Off-chip	0800–2BFF	Off-chip	0000–07FF, 2C00–FFFF
1	0	0	0000–07FF	Off-chip	FE00–FFFF	0800–FDFF
1	0	1	Off-chip	Off-chip	FE00–FFFF	0000–FDFF
1	1	0	0000–07FF	0800–2BFF	FE00–FFFF	2C00–FDFF
1	1	1	Off-chip	0800–2BFF	FE00–FFFF	0000–07FF, 2C00–FDFF

ตารางที่ 2.2 'C50 Program Memory Configuration

2.5 On-Chip Peripherals

ใน TMS 320C50 DSP on-chip periperalss ประกอบด้วย

- ตัวกำเนิดสัญญาณ นาฬิกา
- อาร์ดแวร์ไทม์เมอร์
- Soffware-programmable wait-state generators.
- พอร์ทขนาดอินพุท/เอาท์พุท
- พอร์ทอนุกรม
- Time-division multiplexed (TDM) serial port
- User-maskable interrupts

2.5.1 ตัวกำเนิดสัญญาณนาฬิกา

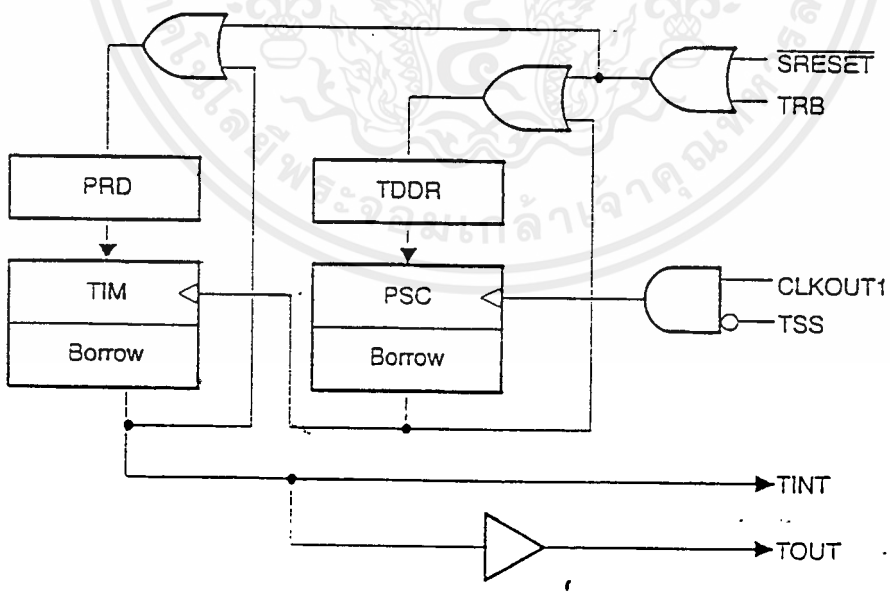
ในตัวกำเนิดสัญญาณนาฬิกาประกอบด้วย ออสซิเลเตอร์ภายในและ วงจรเฟสล็อคลูป (PLL).

CLKMD1	CLKMD2	Clock Mode
0	0	External divide-by-2 option with internal oscillator disabled.
0	1	Reserved for test purposes.
1	0	PLL clock generator option. - For 'C50, 'C51, 'C53, and 'C53S: multiply-by-1 option - For 'C52: multiply-by-2 option
1	1	External divide-by-2 option or internal divide-by-2 option with an external crystal.

ตารางที่ 2.3 Standard Clock Option

2.5.2 Hard ware Timer.

ใน Hard ware Time นี้จะใช้ 16-บิท กับ 4-บิท prescaler ซึ่งจะมีเรทอยู่ระหว่าง $^1/_2$ และ ของแมทชีนไซเคิ้ล (CLKOUT1) ซึ่งจะขึ้นอยู่กับอัตรา divide-down ของไทม์เมอร์ โดยไทม์เมอร์สามารถถูกทำให้หยุด, เริ่มใหม่, รีเซต หรือถูกอนุญาตโดย specific status bits



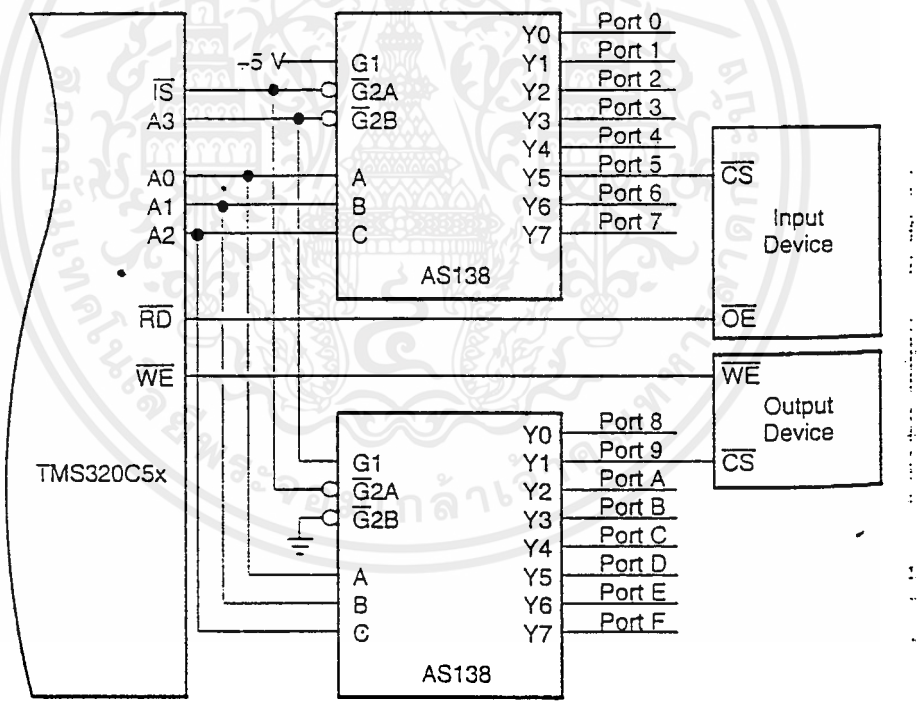
รูปที่ 2.6 Timer Block Diagram

2.5.3 Software-Programmable Wait-State Generators

ใน TMS 320c50 จะอนุญาตให้ wait-state generation โดยไม่จำเป็นต้องมีฮาร์ทแวร์ภายนอกมาต่อเชื่อมกับ slower off-chip memory และอุปกรณ์อินพุท/เอาท์พุท ซึ่งเป็นลักษณะของส่วนประกอบของการรวมวงจร wait-state generation โดยแต่ละวงจรผู้ใช้ programmable สามารถดำเนินการในการแตกต่างของ wait states เพื่อ off-chip memory

2.5.4 พอร์ทขนาดอินพุท/เอาท์พุท

ผลรวมทั้งหมด 64k พอร์ท อินพุท/เอาท์พุท จะถูกนำมาใช้ โดย 16 พอร์ท จะเป็น memory-mapped ใน data memory space โดยแต่พอร์ทของ อินพุท/เอาท์พุท สามารถถูกกำหนดตำแหน่งโดยคำสั่ง IN หรือ OUT ใน memory-mapped อินพุท/เอาท์พุทพอร์ท สามารถกระทำตามคำสั่งต่าง ๆ ว่าจะอ่านหรือเขียนจาก data memory โดย IS จะบ่งบอกถึงว่าจะอ่านหรือเขียนโดยผ่านทางพอร์ท อินพุท/เอาท์พุท



รูปที่ 2.7 I/O Port Ingerface Circuitry

2.5.5 พอร์ทอนุกรม

พอร์ททั้ง 3 ของพอร์ทอนุกรมจะถูกนำมาใช้ต่าง ๆ กันดังนี้

: พอร์ทอนุกรมที่ใช้ในงานทั่วไป, พอร์ทอนุกรม TDM, และบัฟเฟอร์พอร์ท (BSP) ในพอร์ทอนุกรมตัวส่งและตัวรับจะถูก double-buffered และจะถูกควบคุมโดยสัญญาณ maskable extermal interrupt

2.5.6 พอร์ทอนุกรม TDM

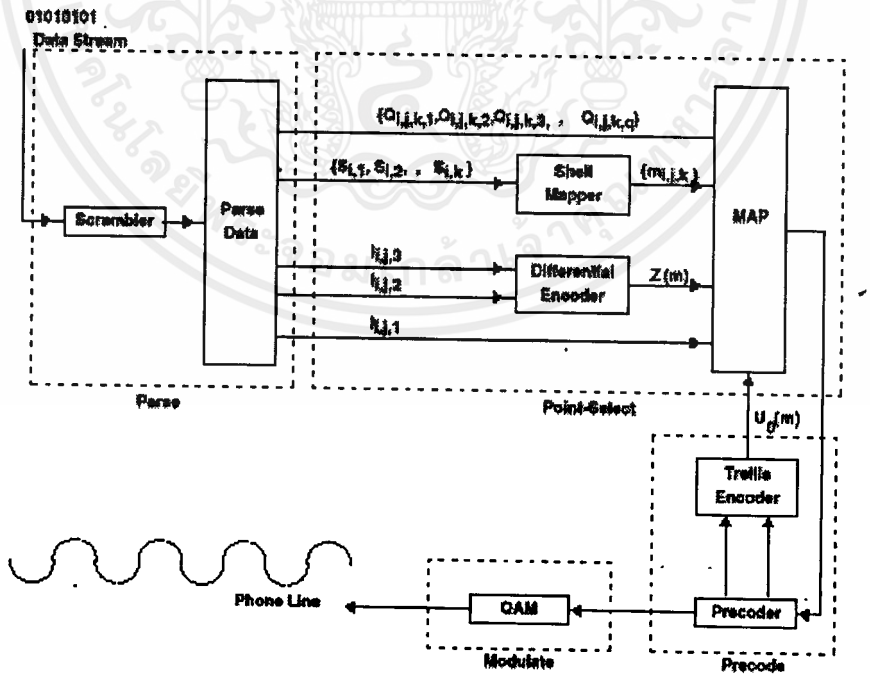
พอร์ทอนุกรม TDM จะถูกใช้ใน C50, C51 และ C53 ซึ่งเป็น full-duplexed อนุกรมพอร์ทสามารถกำหนดโดยซอร์ฟแวร์ของขบวนการ synchronous หรือขบวนการ TDM

2.5.7 User-Maskable Interrupt

จะประกอบด้วย 4 อินเตอร์รัฟท์จากภายนอก (INT1-INT4) และ 5 อินเตอร์รัฟท์จากภายใน, 1 ไทม์เมอร์อินเตอร์รัฟท์ และ 4 อินเตอร์รัฟท์จากพอร์ทอนุกรม ซึ่งเป็น User-maskable. เมื่อ interrupt service routine (ISR) ถูกทำการปฏิบัติ program counter จะถูกเก็บไว้ที่ 8-level hardware stack.

บทที่ 3 V.34 Transmitter

ใน V.34 Transmitter จะประกอบไปด้วย 4 ส่วนด้วยกัน ดังแสดงไว้ตามรูปที่ 3.1 ซึ่งเป็น Block Diagram โดยทั่วไปของ V.34 Transmitter ในส่วนแรกนั้นเป็นส่วนที่มีชื่อว่า Parse โดยส่วนนี้จะรับ input เป็น Binary Data แล้วจะทำการ Scramble โดยในส่วนของการ Scramble นี้จะก่อให้เกิดกลุ่มของข้อมูลต่างๆ ซึ่งจะเป็นอินพุทของกลุ่มถัดไป โดยส่วนที่สองนั้นมีชื่อว่า Point - Select โดยจะนำ Data bits ที่ออกมาจากส่วน Parse มาทำการเลือก Signal point จากกลุ่มของข้อมูล 2 - dimensional (2D) ซึ่งเป็นจุดเด่นที่ใช้ใน V.34 เท่านั้น ต่อจากนั้นเป็นส่วนที่ 3 ซึ่งเรียกว่า Precode โดยส่วนนี้จะทำการชดเชยสำหรับ noise - whitening filter ซึ่งอยู่ในภาค Reciever ซึ่งส่วนที่ 3 นี้จะมี Trellis Encoder ประกอบเป็น feed back ซึ่งจะทำให้การส่งข้อมูลได้ผลที่แน่นอนขึ้น และในส่วนที่สี่ซึ่งเป็นส่วนสุดท้ายของภาค Transmitter เป็นส่วนที่มีชื่อว่าส่วน Modulate เราจะใช้เป็นมาตรฐานทั่วไปของ Modem คือการใช้ Quadrature Amplitude Modulation (QAM) ซึ่งจะทำการ Modulate Signal Point ให้เป็น Wave from

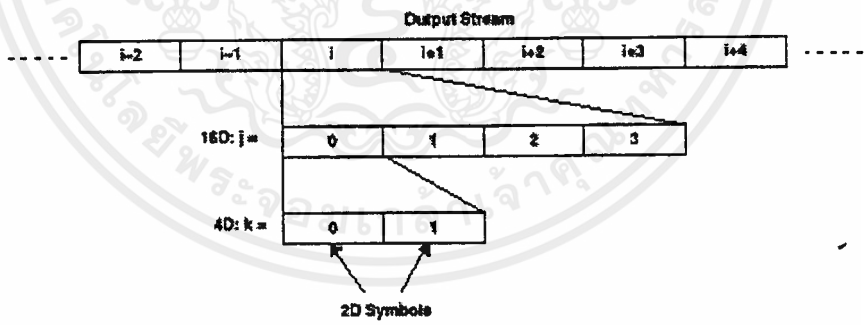


รูปที่ 3.1 บล๊อกไดอะแกรมของ V.34 Transmitter

3.1 กล่าวทั่วไป

โดยหลักการแล้ว V.34 transmitter นี้จะทำการ Map binary input data เพื่อให้เป็น output sequence 2D point และทำการ Modulate โดยใช้ QAM ที่คลื่นพาหะ ตามที่กำหนดไว้ในมาตรฐานการส่งผ่ายช่องสัญญาณอนาล็อก โดยความถี่ที่ออกจากโมเด็ม ต่อ output point หรือ Symbol จะเรียกว่า Symbol rate โดยรายละเอียดจะอยู่ในบทที่ 1 ซึ่งเป็นความเร็วในการส่งข้อมูลเป็นอัตราส่วนตัวอย่าง เช่น ในการ map data ไปเป็นหนึ่งใน 16 different 2D point โดยใช้ symbol rate ที่ 2400 symbol / sec นั้น Modem จะมีความเร็วในการส่ง 9600 bit / sec ซึ่งจะมีใช้ใน Modem แบบ high speed modem โดย V.34 modem จะใช้หลักการหรือเทคนิคพิเศษ ในการเลือก 2D point ให้มีการผิดพลาดน้อยที่สุดและทำให้ได้กำลังของสัญญาณมากที่สุดเพื่อทำให้ภาค Reciever decode ได้ถูกต้องที่สุด

ในส่วนของ V.34 modem นั้น output sequence 2D point จะทำการแบ่งออกเป็นส่วนย่อยๆ ได้ 8 - point เรียกว่า mapping frame โดย mapping frame นี้จะมองดูคล้ายกับ 4D point หรือ 16D point โดยส่วนนี้เองจะใช้เป็นเงื่อนไขในการจัดการ encode แบบ trellis encoder ซึ่งจะประมวลผลที่ 4D point และ shell mapper จะประมวลผลที่ 16D point โดยในที่นี้ mapping frame จะอ้างอิงโดย time index m ซึ่ง $m = 4i+j$; $(j = 0,1,2,3)$ และ 2D point จะอ้างอิงโดย time index n โดย $n = 2m+k$; $(k = 0,1)$ โดยความสัมพันธ์ดังกล่าวได้แสดงไว้ในรูปที่ 3.2



รูปที่ 3.2 Mapping Frame Conventionns

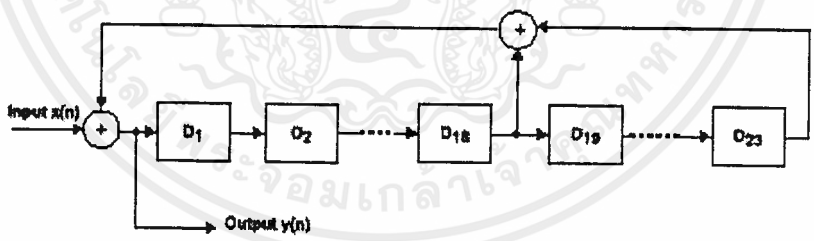
mapping frame จะเป็นหน่วยที่น้อยที่สุดของ output ของภาค Transmitter จำนวน data ของ encode ใน 1 mapping frame จะมีจำนวนมากที่ประกอบอยู่ใน data Transmitter rate และ Symbol rate ตัวอย่างเช่นที่ Maximum data transmittion rate ที่ 28800 bit / sec และ symbol rate ที่ 3200 Symbol / sec V.34 modem จะทำการ encode 72 bits ของ data ใน 1 mapping frame (28800 / 3200 / 8 = 72)

และลองส่งที่ 2400 Symbol / sec และทำการส่ง data transmittion rate ที่ 9600 , 14400 และ 19200 bits / sec สำหรับ rate ที่ transmitter จะทำการ encode 32 , 48 , 64 bits ต่อ mapping frame แม้ว่าจำนวนของบิตที่จะมาทำการ encode / mapping frame จะมีมากมายก็ตาม ทฤษฎีที่ใช้ในการ map data ไปเป็น signal point นี้จะเหมือนๆ กับใช้ใน Symbol rate และ data transmittion rate ที่ทุกๆ ความเร็ว ซึ่งแต่ละความเร็วก็จะใช้หลักการเดียวกันนั่นเอง

หลักการที่ใช้ใน V.34 transmitter ในการ encode block ของ data สำหรับ หนึ่ง mapping frame จะได้ทำการนำเสนอออกเป็นส่วนๆ และทฤษฎีหลักการในการคำนวณนั่นจะอยู่ในส่วนของภาคผนวก ก

3.2 Prase

ในส่วนของ Parse นี้ จะมีการทำงานหลักๆ 2 ประการคือ ประการแรก จะทำการ Scramble input data ที่เป็นอินพุทเข้ามาในตอนแรกก่อน ซึ่งจะทำให้ได้สัญญาณที่เป็นคาบ (Periodic output sicnal) เพื่อกำจัด loss ของ Symbol clock เพื่อเป็นผลดีใน Adaptive filter ของ V.34 ภาค Reciever ประการที่สอง เป็นกลุ่มของ Scramble bit ที่เป็น discrete ที่จะนำไปเข้าเป็น input ในส่วนของภาค point select ต่อไป



รูปที่3.3 Scramble diagram

3.2.1 Scramble

data scrambler ตามมาตรฐานของ V.34 นั่นจะใช้หลักการ linear - shift - clock feedback register ซึ่งประกอบไปด้วย Genrating Polynomial (GP)

$$(GP) = 1 + x^{-18} + x^{-23} \qquad \text{.... (3.1)}$$

จากรูป diagram ของ scramble data จะประกอบไปด้วยเทอมของ 23 - tap delay line ซึ่งแสดงในรูปที่ 3.3 โดยหลักการ scramble แบบนี้จะใช้ในการออกแบบ modem เป็นส่วนใหญ่ เพราะว่ามีคุณสมบัตินั้นดีมาก ข้อดีอีกประการหนึ่งของการ scramble แบบนี้ก็คือ จะเป็น self - synchronizing และจะให้ GP ที่คาบสูงสุดที่ $2^{23} \pm 1$

3.2.2 Parser

ในส่วนของ Parser block ของ binary data สำหรับ 1 mapping frame จากกลุ่มต่าง ๆ ของ บิต จะมาประมวลทำให้แยกออกเป็นส่วน ๆ ในการส่งให้ภาคต่อไป (Point Select) โดยจำนวนบิตกลุ่มหนึ่งจะไปยัง shell mapper ส่วนจำนวนบิตอีก 4 กลุ่มที่เหลือจะเข้าไปเป็นอินพุทของ Differential encoder และกลุ่มอื่น ๆ ที่ไม่ได้ผ่านการ encode จะไปยังส่วนของ Point Select โดยกลุ่มของจำนวนบิตนี้จะถูกจัดไว้เป็นพวก ๆ ดังนี้

$(S_{i,1}, S_{i,2}, \ldots, S_{i,K})$
$(I_{1i,0}, I_{2i,0}, I_{3i,0}), (Q_{i,0,0,1}, Q_{i,0,0,2}, \ldots, Q_{i,0,0,q}), (Q_{i,0,1,1}, Q_{i,0,1,2}, \ldots, Q_{i,0,1,q}),$
$(I_{1i,1}, I_{2i,1}, I_{3i,1}), (Q_{i,1,0,1}, Q_{i,1,0,2}, \ldots, Q_{i(,1,0,q)}), (Q_{i,1,1,1}, Q_{i,1,1,2}, \ldots, Q_{i,1,1,q}),$
$(I_{1i,2}, I_{2i,2}, I_{3i,2}), (Q_{i,2,0,1}, Q_{i,2,0,2}, \ldots, Q_{i,2,0,q}), (Q_{i,2,1,1}, Q_{i,2,1,2}, \ldots, Q_{i,2,1,q}),$
$(I_{1i,3}, I_{2i,3}, I_{3i,3}), (Q_{i,3,0,1}, Q_{i,3,0,2}, \ldots, Q_{i,3,0,q}), (Q_{i,3,1,1}, Q_{i,3,1,2}, \ldots, Q_{i,3,1,q}).$

โดย subscript ที่จะนำมาอ้างอิงก่อนคือ i ซึ่งเป็น time index ของ mapping frame โดย K และ S เป็น label ของ input ของ shell mapper โดยอีก 4 กลุ่ม ซึ่งมี label I จะเป็นอินพุทของ differential encoder และอีก 8 กลุ่มของ Q บิต จะไม่ถูก encode โดยค่าของ k และ q จะเปลี่ยนแปลงไปตาม data transmittion rate โดยค่าต่าง ๆ เหล่าที่จะหามาได้จากขนาดของ block ของบิต ของการ mapping frame โดยรายละเอียดต่าง ๆ จะได้แสดงไว้ในตารางที่ 3.1 สำหรับที่ 9600 bits/sec k=20 และ q=20 ซึ่งในหนึ่ง mapping frame จะประกอบไปด้วย 20+12=32 บิต สำหรับที่ 14400 จะประกอบไปด้วย k=28 และ q=1 (28+12+8 = 48 bit/mapping frame) และที่ 19200 bits/sce k=28 และ q=3 (28+12+24 = 64 bit/mapping frame)

Symbol Rate, S	Data Rate, R	K	M		L	
			Minimum	Expanded	Minimum	Expanded
2400	2400	0	1	1	4	4
	2600	0	1	1	4	4
	4800	4	2	2	8	8
	5000	5	2	2	8	8
	7200	12	3	4	12	16
	7400	13	4	4	16	16
	9600	20	6	7	24	28
	9800	21	7	8	28	32
	12 000	28	12	14	48	56
	12 200	29	13	15	52	60
	14 400	28	12	14	96	112
	14 600	29	13	15	104	120
	16 800	28	12	14	192	224
	17 000	29	13	15	208	240
	19 200	28	12	14	384	448
	19 400	29	13	15	416	480
	21 600	28	12	14	768	896
	21 800	29	13	15	832	960
2743	4800	2	2	2	8	8
	5000	3	2	2	8	8
	7200	9	3	3	12	12
	7400	10	3	3	12	12
	9600	16	4	5	16	20
	9800	17	5	5	20	20
	12 000	23	8	9	32	36
	12 200	24	8	10	32	40
	14 400	30	14	17	56	68
	14 600	31	15	18	60	72
	16 800	29	13	15	104	120
	17 000	30	14	17	112	136
	19 200	28	12	14	192	224
	19 400	29	13	15	208	240
	21 600	27	11	13	352	416
	21 800	28	12	14	384	448
	24 000	26	10	12	640	768
	24 200	27	11	13	704	832

Symbol Rate, S	Data Rate, R	K	M		L	
			Minimum	Expanded	Minimum	Expanded
2800	4800	2	2	2	8	8
	5000	3	2	2	8	8
	7200	9	3	3	12	12
	7400	10	3	3	12	12
	9600	16	4	5	16	20
	9800	16	4	5	16	20
	12 000	23	8	9	32	36
	12 200	23	8	9	32	36
	14 400	30	14	17	56	68
	14 600	30	14	17	56	68
	16 800	28	12	14	96	112
	17 000	29	13	15	104	120
	19 200	27	11	13	176	208
	19 400	28	12	14	192	224
	21 600	26	10	12	320	384
	21 800	27	11	13	352	416
	24 000	25	9	11	576	704
	24 200	26	10	12	640	768
3000	4800	1	2	2	8	8
	5000	2	2	2	8	8
	7200	8	2	3	8	12
	7400	8	2	3	8	12
	9600	14	4	4	16	16
	9800	15	4	5	16	20
	12 000	20	6	7	24	28
	12 200	21	7	8	28	32
	14 400	27	11	13	44	52
	14 600	27	11	13	44	52
	16 800	25	9	11	72	88
	17 000	26	10	12	80	96
	19 200	24	8	10	128	160
	19 400	24	8	10	128	160
	21 600	30	14	17	224	272
	21 800	31	15	18	240	288
	24 000	28	12	14	384	448
	24 200	29	13	15	416	480
	26 400	27	11	13	704	832
	26 600	27	11	13	704	832

Symbol Rate, S	Data Rate, R	K	M		L	
			Minimum	Expanded	Minimum	Expanded
3200	4800	0	1	1	4	4
	5000	1	2	2	8	8
	7200	6	2	2	8	8
	7400	7	2	2	8	8
	9600	12	3	4	12	16
	9800	13	4	4	16	16
	12 000	18	5	6	20	24
	12 200	19	6	6	24	24
	14 400	24	8	10	32	40
	14 600	25	9	11	36	44
	16 800	30	14	17	56	68
	17 000	31	15	18	60	72
	19 200	28	12	14	96	112
	19 400	29	13	15	104	120
	21 600	26	10	12	160	192
	21 800	27	11	13	176	208
	24 000	24	8	10	256	320
	24 200	25	9	11	288	352
	26 400	30	14	17	448	544
	26 600	31	15	18	480	576
	28 800	28	12	14	768	896
	29 000	29	13	15	832	960
3429	4800	0	1	1	4	4
	5000	0	1	1	4	4
	7200	5	2	2	8	8
	7400	6	2	2	8	8
	9600	11	3	3	12	12
	9800	11	3	3	12	12
	12 000	16	4	5	16	20
	12 200	17	5	5	20	20
	14 400	22	7	8	28	32
	14 600	23	8	9	32	36
	16 800	28	12	14	48	56
	17 000	28	12	14	48	56
	19 200	25	9	11	72	88
	19 400	26	10	12	80	96
	21 600	31	15	18	120	144
	21 800	31	15	18	120	144
	24 000	28	12	14	192	224
	24 200	29	13	15	208	240
	26 400	26	10	12	320	384
	26 600	27	11	13	352	416
	28 800	24	8	10	512	640
	29 000	24	8	10	512	640

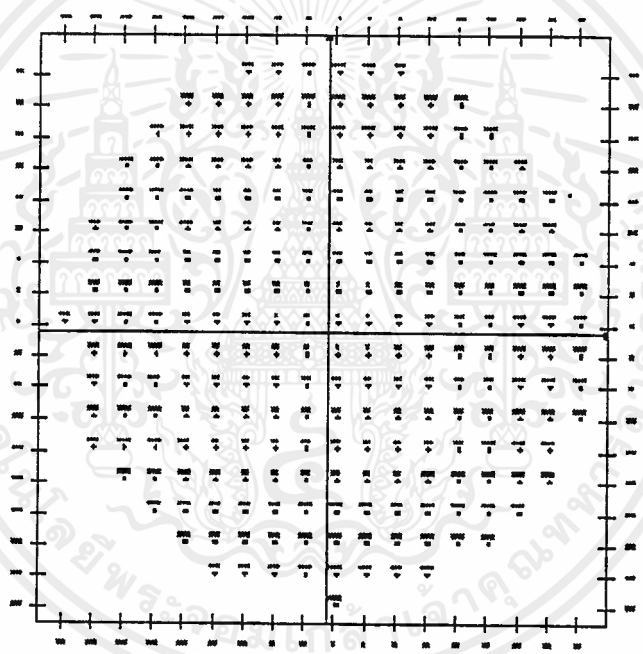
Data Rate, R	2400 sym/s		2743 sym/s		2800 sym/s		3333 sym/s		3600 sym/s		3692 sym/s	
	P = 12		P = 12		P = 14		P = 16		P = 16		P = 16	
	B	SWP	b	SWP	b	SWP	b	SWP	b	SWP	b	SWP
2400	8	FFF	-	-	-	-	-	-	-	-	-	-
2800	9	6DB	-	-	-	-	-	-	-	-	-	-
4800	16	FFF	14	FFF	14	1BB7	13	3DEF	12	FFFF	12	0421
5000	17	6DB	15	6DB	16	0489	14	1249	13	6666	12	3DEF
7200	24	FFF	21	FFF	21	16AB	20	0421	18	FFFF	17	3DEF
7400	25	6DB	22	6DB	22	0081	20	0777	19	6666	18	0489
9600	32	FFF	28	FFF	28	0A95	26	2E6B	24	FFFF	23	14A5
9800	33	6DB	29	6DB	28	3FFF	27	0C81	25	6666	23	3F7F
12 000	40	FFF	35	FFF	35	0409	32	7FFF	30	FFFF	28	7FFF
12 200	41	6DB	36	6DB	35	1FBF	33	2AAB	31	6666	29	1555
14 400	48	FFF	42	FFF	42	0C81	39	14A5	36	FFFF	34	2E6B
14 600	49	6DB	43	6DB	42	1BB7	39	3FFF	37	6666	35	0489
16 800	56	FFF	49	FFF	48	3FFF	45	3DEF	42	FFFF	40	0421
17 000	57	6DB	50	6DB	49	15AB	46	1249	43	6666	40	3DEF
19 200	64	FFF	56	FFF	56	1FBF	52	0421	48	FFFF	46	3DEF
19 400	65	6DB	57	6DB	56	0A95	52	0777	49	6666	46	1489
21 600	72	FFF	63	FFF	62	1BB7	58	2140	54	FFFF	51	14A5
21 800	73	6DB	64	6DB	63	0489	59	0081	55	6666	51	0F7F

ตารางที่ 3.1 Mapping Parameter K , M and L at difference data rate and symbol rate

3.3 Point - Selcet

เป็นหลักการที่แตกต่างจาก V - series ก่อน ๆ ในการเลือก 2D - point เนื่องจากว่า data transmittion rate ของ V.34 modem นั่นสามารถปรับเปลี่ยนให้สามารถติดต่อรับส่งกับ modem V.serie อื่น ๆ ได้ด้วย

โดย V.34 Transmitter จะทำการเลือก 2D point จากสับเซตของ 2 Z^2+(1,1) lattice โดยสับเซตนี้จะเรียกว่า superconstellation ซึ่งประกอบไปด้วย 960 point และ 240 point โดยสับเซตที่อยู่ใน superconstellation จะอยู่บน 4 Z^2+(1,1) lattice ซึ่งแสดงดังรูปที่ 3.4 โดย full constellation จะสามารถหาได้จากสับเซตที่หมุนไป 0̊ 90̊ 180̊ และ 270̊ ในการประกอบกันของ symbol rate และ data rate จะให้รายละเอียดต่าง ๆ ของสับเซตใน superconstellation



รูปที่ 3.4 240 Piont Quarter Constellation

จำนวน point ของ constellation จะเป็นลักษณะที่นำไปสู่ concentricring ที่มีระยะห่างเพิ่มขึ้นจากจุด origin โดยแต่ละ ring จะมีค่าเท่ากับจำนวนของ consteletion point โดยระยะห่างต่างๆ จะสามารถหาได้จาก Squared norm ซึ่งจะเป็นสัดส่วนของกำลังของ point เนื่องมาจากระยะทางที่ห่างไกลออกไปจากจุด original จะมีกำลังที่มากกว่า ซึ่งหลักการของ V.34 นี้จะทำการเฉลี่ยกำลังของ

สัญญาณที่ส่งไปให้มีค่าน้อยที่สุด ซึ่งสิ่งเหล่านี้จะนำไปใช้ในภาค point - selection ซึ่งจะทำการเลือก point ใน ring ก่อนที่จะเลือก point จากนอก ring

โดยใน V.34 point - selection นี้จะมีส่วนประกอบด้วยกัน 3 ส่วน คือในส่วนแรกเป็น shell mapper ซึ่งใช้บิต K,S ซึ่งจะทำให้ได้สัญญาณซีเควนท์ 8 ring ส่วนที่สอง กลุ่มของ uncode q และบิต Q จะใช้สำหรับการเลือก point ในแต่ละ ring จากหนึ่งควอเตอร์ของ superconstellation ดูรูปที่ 3.4 และสำหรับส่วนที่ 3 ในส่วนของบิต I (โดยส่วนหนึ่งเป็นผลมาจากเอาท์พุท U0 จาก trellis encoder) จะใช้กับ differenial encoder

3.3.1 Shell mapper

Shell mapping เป็นเทคนิคที่สมบูรณ์แบบในการจัดการกับ shaping gain ในระหว่างค่าต่ำสุดของกำลังของสัญญาณของการส่ง โดยหลักการของ shell mapping นี้จะทำการเลือกซีเควนท์ของ 8 ring สำหรับ 8 2D -point ในหนึ่ง mapping frame

โดยการส่ง contellation ทั้งหมดจะทำการแบ่งเป็น M ring โดยมี Label ตั้งแต่ 0 ถึง M±1 โดยแต่ละวงจะทำหน้าที่วัดเพื่อที่จะให้มีค่าเท่ากับ Label ของมันเอง โดยในหนึ่ง mapping frame จะมีค่าเท่ากับ $M+8^8$ โดยประมาณของ M ring และจากการที่จะต้อง map บล๊อก K บิตให้เป็น 2^K ซีเควนต์ของ 8 ring ด้วยพลังที่น้อยที่สุด โดยพลังงานของบล๊อก ๆ หนึ่งของ ring จะวัดในเทอมของผลรวมของลำดับความสำคัญแต่ละบล๊อก ในส่วนประกอบของ ring ซึ่งซีเควนต์ M^8 ของ ring จะมาจากการ์ดที่น้อยที่สุดหลาย ๆ ครั้ง ซึ่งแต่ละซีเควนต์จะมี Label ตั้งแต่ M^8±1 ซึ่ง ใน shell mapping นี้จะซีเควนต์ต่าง ๆ จะมีลำดับความสำคัญเท่า ๆ กัน หรือใกล้เคียงกันโดยในอันดับแรกซีเควนท์ที่มีความสำคัญที่น้อยที่สุดจะถูกเลือกโดย shell mapper โดยตอนนี้จะให้จำนวน K - bit เป็นอินพุทในหลักการ shell mapper เอาท์พุทของมันจะเป็นซีเควนท์ของ ring ซึ่งมีจำนวนเป็น label

ตัวอย่างเช่น มีจำนวนซีเควนท์ของ ring {0,0,0,0,0,0,0,0,} โดยจะมีลำดับความสำคัญจาก 0 และกำหนด label เป็น 0 ซึ่งในที่นี้จะไม่มีลำดับความสำคัญที่น้อยที่สุด ในที่นี้เองจากการที่มี อินพุท 0 shell mapper จะทำการย้อนกลับซีเควนท์ {0,0,0,0,0,0,0,0,} สำหรับความสำคัญหนึ่งจะมี 8 ซีเควนท์ที่เป็นไปได้ โดยซีเควนท์ต่าง ๆ มีดังนี้ {0,0,0,0,0,0,0,1,}, {0,0,0,0,0,0,1,0,}, ...{1,0,0,0,0,0,0,0,} ซึ่งมี label เป็น 1,2,....,8 ด้วยกัน ทั้งหมดนี้จากความสำคัญของ shell mappiing 1 ให้ label เป็นอินพุท shell mapper จะเลือก ซีเควนท์ ring ที่มีความหมายเหมือนกัน สุดท้าย ซีเควนท์ {5,5,5,5,5,5,5,5} ความสำคัญ 40 แต่ไม่ทำการเลือกเนื่องจากตามหลักการ $2^K < M^8$ ทำให้ซีเควนท์มี label มากกว่า 2^K±1 โดยทฤษฎีการคำนวนและตัวอย่างสามารถดูได้จากดรรชนี ก.

3.3.2 Mapper

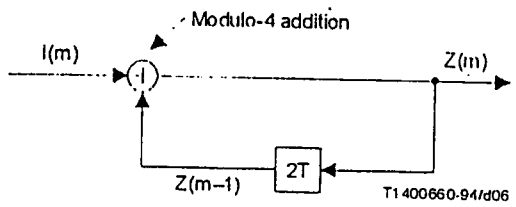
การ Mapping ทั้งแปดเซตของ Q บิต ที่ไม่ได้ทำการเข้ารหัสจะเข้าไปยังส่วนของ Mapper โดยตรง โดย point ในรูปที่ 3.4 ในแต่ละ label ของจำนวน ring จาก 0 ถึง 239 ซึ่งเป็นจำนวนของกำลังที่เพิ่มขึ้น และจำนวนนี้เองจะเป็นจำนวนของ point ที่ชี้ในตาราง การเลือก point ใน shell หาด้วยการเซตได้ q Q บิต = $\{Q_1, Q_2,...,Q_q\}$ จะใช้สมการดังนี้

$$\text{point index} = Q_1 + 2^1Q_2 + ,..., 2^{q-1}Q_q + 2q_m \qquad ....(3.2)$$

โดยหลัก ๆ แล้ว Q บิตจะประกอบไปด้วยจำนวนของไบนารีและการบวกกันของจำนวนไบนารีนี้เองจะเป็น point index ของ point แรกใน shell และผลของการบวกกันของ point นี้เองจะเป็นตัวชี้ point ในควอเตอร์ของ superconstellation โดย point ต่าง ๆ เหล่านี้จะสามารถดูได้จากตารางคุณสมบัติของ Differential encoder

3.3.3 Differential encoder

จากจุดเริ่มต้นของควอเตอร์ของ superconstellation ซึ่งทำการเลือกโดย shell mapper และส่วนของ mapper และจากภาค point-select จะต้องมีส่วนของ Differential encoder เพื่อเป็นการตรวจให้แน่นอนว่า ซีเควนท์ของ symbol เป็นการหมุนไป 90° อย่างสม่ำเสมอ โดยอินพุตของ differential encoder จะเป็นจะประกอบไปด้วย 4 บิต ซึ่งเป็นหนึ่งในสามเซตของ I บิต และเอาท์พุตบิต U_0 จาก trellis encoder ซึ่งจะกระทำบน 4D symbol (2 ของ 2D point) ที่หนึ่งช่วงเวลา ดังนั้นใน 1 mapping frame จะกระทำการประมวลผลทั้งหมด 4 ช่วงเวลา โดยการเปรียบเทียบให้เห็น shell mapper จะถูกเรียกว่า หนึ่งต่อ mapping frame และ 1 mapper จะใช้ 8 ช่วงเวลา ซึ่งบล๊อกไดอะแกรมของ Differential encoder สามารถดูได้จากรูปที่ 3.5



รูปที่ 3.5 Block Diagram of Differential Encoder

จุดมุ่งหมายของ Differential encoder ก็คือการกระทำให้ซีเควนท์ของ symbol ที่ทำการส่งนั้นมีการหมุนไปทุก ๆ 90° อย่างสม่ำเสมอ เพราะว่าภาค Reciever จะไม่สามารถ detect phase ที่ไม่ใช่ 90°

ในส่วนของอินพุทของ differential encoder ซึ่งประกอบไปด้วย $\{I_3, I_2, I_1, u_0\}$ และในการ encode จะมีคู่ของ point เริ่มต้น (เช่น 4D symbol 1 อัน) ทำการ encode ดังนั้นโดยการพิจารณา คู่บิต (I_3, I_2) ซึ่งเป็นจำนวนไบนารี่ 2 ดิจิต differential encoder จะทำการคำนวณโดยทำการรวม modulo - 4 ด้วย

$$Z_m = [(I_3, I_2) + z_{m-1}] \text{ mod4} \qquad \text{....(3.3)}$$

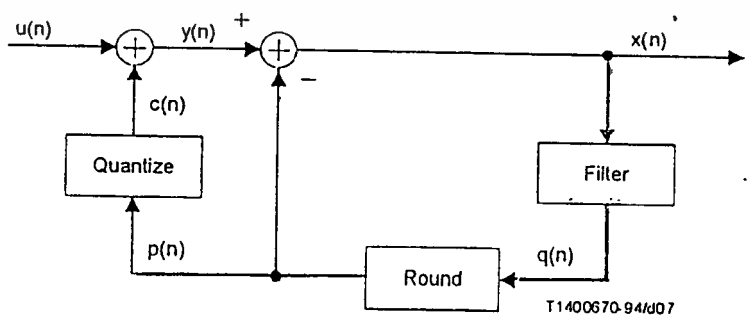
เมื่อ Z_{m-1} เป็นค่าของ Z_m ที่มาก่อน จากจำนวนบิตทั้งสอง Z_m จะมีค่า (0, 1, 2, 3) ซึ่งจะมีค่าทุก ๆ การหมุนไป 90° (Z_m + 90°) ตามเข็มนาฬิกา โดย point แรกของคู่ 4D จะหมุนโดยค่าของ Z_m และ point ที่สองในคู่ 4D จะหมุนโดยค่าเท่า ๆ กัน (ทุก ๆ 90°) ซึ่งค่า rotation factor สามารถคำนวณได้จาก

$$W_m = [(Z_m + (I_1, U_0)] \text{ mod4} \qquad \text{....(3.4)}$$

ซึ่ง (I_1, U_0) จะพิจารณาจากจำนวนไบนารี่ 2 ดิจิตนี้ โดยการหมุน point ที่ 2 นี้ จะทำการหมุนไป W^m * 90° ตามเข็มนาฬิกา

3.4 Precode

ในส่วน precode จะประกอบด้วย nonlinear precoder และ trellis encode ในการเปลี่ยนแปลงการรวมกันของทั้ง 2 เป็นลักษณะจำเพาะของ V.34 โมเด็ม โดยการรวบรวม trellis encoder เข้าไปด้วยการวนกลับ (จะแสดงดังรูปที่ 3.6) ซึ่งจะทำการชดเชย noise whitening prediction-error filter ของโมเด็ม V.34 ที่ภาครับโดยปราศจากการทำลายรูปทรง shaping gain โดยจะใช้ shell mapper.



รูปที่ 3.6 Block Diagram of Precoder

3.4.1 Nonlinear Precode

ในภาครับของโมเด็ม V.34 จะมีภาค 4-tap prediction error filter ซึ่งถูกใช้ในเรื่อง noise-whitening โดยฟังชั่นของ finite impulse response (FIR) จะประกอบด้วย

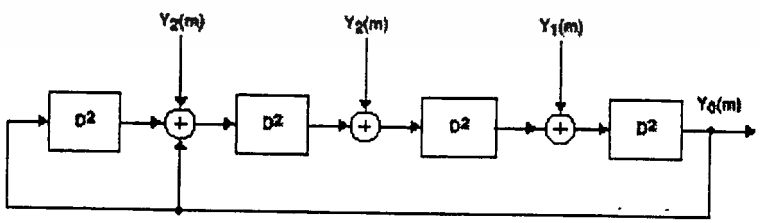
$$H(Z) = 1 + h_1Z_{-1} + h_2Z^{-2} + h_3Z^{-3} \quad \text{....(3.4)}$$

ในโมเด็ม V.34 จะมีแบบจำลองของฟิลเตอร์โดยอินเวอร์ส ของฟิลเตอร์นี้จะนำมาใช้ที่ precode สัญญาณที่ถูกส่งมา ในการ precode นี้จะมีผลกระทบจาก pre-emphasizing ซึ่งจุดข้อมูลต่าง ๆ จะมีคุณสมบัติในการ decode ที่ precode ของภาครับหลังจากกระทำ noise-whitening ในด้านล่างของ precode จะเป็น trellis ซีเควียน ซึ่งจะทำลายข้อมูลของการส่ง ในการแก้ไขปัญหานี้ V.34 จะทำการแก้ไขบิทที่จะใช้ในการคำนวนโดย nonlinear precoder เพื่อเป็นการรักษา trellis ซีเควียนในการแก้ไขบิน Co จะทำการคำนวณทุก ๆ 4D symbol ซึ่งในการคำนวณจะทำการ exclusive-OR กับ เอาท์พุท Yo ของ trellis encode ที่ได้จากบิท U_0 ซึ่งเป็นหนึ่งในอินพุทบิทของ differential encoder

3.4.2 Trellis Encoder

ใน trellis encoding จะเป็นวิธีที่จะกระทำการเพิ่มขึ้นของแกนและความหนาแน่นของ constellation ซึ่งจะเก็บรักษาระยะทางที่สั้นที่สุดระหว่างจุดที่เหมือน

ใน trellis encode จะมี 4D trellis encode อยู่ 3 แบบ คือ 16-state code, 32-state code, 64-state code ซึ่งในที่นี้จะใช้เพียง 16-state 4D code ของ L-F Wei เท่านั้น โดยที่จุดสำคัญของ trellis encode คือ 16-state convolutional encode ที่แสดงในรูปที่ 3.7 จากรูปพบว่าใน convolutional encode นี้มี 2 อินพุทที่ได้มาจาก differentail encode I bit. อย่างไรก็ตามใน V.34 การคำนวณ อินพุทจะได้จากเอาท์พุทของ nonlinear precoding ซึ่งได้แสดงในภาคผนวก ก. ผลของหลักการนี้ อินพุททั้ง 3 ของ encoder จะต้องใส่ state table ชุดใหม่เพื่อเปลี่ยน convolutional encoder



รูปที่ 3.7 16-state convolutional encoder

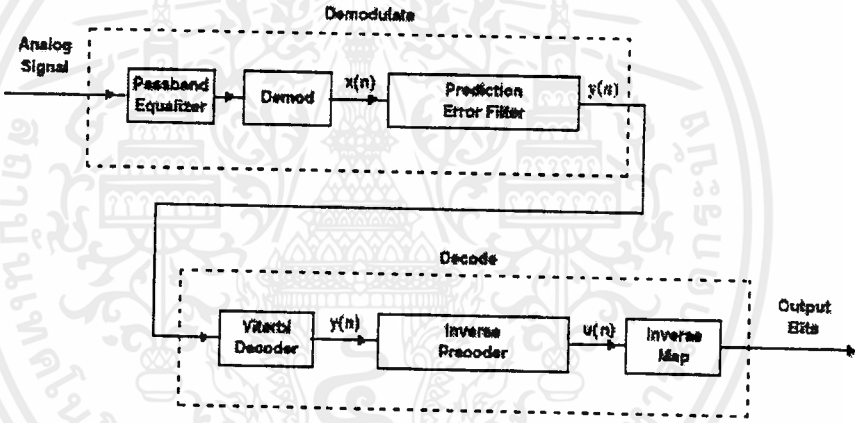
ที่เอาท์พุท Y_0 ของ convolutional encoder จะถูกคำนวณที่ทุก ๆ 4D symbol หลังจากทำการปรับปรุงโดยการแก้ไข bit C_0 แล้วผลของ bit U_0 จะถูกใช้เพื่อเลือก 4D point ชุดต่อไป เป็นที่น่าสังเกตว่าเอาท์พุทของ convolutional encoder จะขึ้นอยู่กับ 4D symbol ของชุดที่แล้วเท่านั้น

3.5 Modulate

ในโมเด็ม V.34 จะใช้การมอดดูเลทแบบ QAM โดยที่การแปลง digital-to-analog (D/A) จะดำเนินการในรูปแบบ 9600 samples per second และ symbol rate ของ 2400 symbols/sec. โดยจำนวนชุดทั้ง 8 ของ ของ 2D symbol จะถูกเก็บไว้ทุก ๆ หนึ่งช่วงเวลา ดังนั้นการมอดดูเลทจะใช้ baseband-shaping filter ที่ความกว้าง 32-sample โดยเอาท์พุทจะถูกมอดดูเลท ขึ้นเป็น passband รอบ ๆความถี่พาหะที่ 1800 H_z ที่ shaping filter ถูกใช้ให้เป็น cosine filter กับ bandwidth factor ที่เกินมาซึ่งมีค่าเป็น 0.12 อย่างไรก็ตามผลจากการคำนวณ passband-shaping filter 2 ค่าจะถูกเก็บไว้สำหรับประการแรกใช้ใน in-phase component และประการที่สองใช้ใน Quadrate component โดยทั้ง 2 ประการจะเป็น pre-modulate ของความถี่พาหะ ที่ 2D symbol จำเป็นที่จะถูก modulate ณ passband ก่อนที่จะทำการ filter เพราะว่าส่วนประกอบคือ symbol rate และความถี่พาหะ (2400 symbol/sec และ 1800 H_z) ซึ่งการ modulate ของ symbol โดยทั่วไปจะกระทำการ modulate หมุนทุก ๆ 90°

บทที่ 4 V.34 Receiver

ในส่วนบล็อคไดอะแกรมตัวรับของ V.34 ได้แสดงไว้ในรูปที่ 4.1 โดยที่ตัวรับจะประกอบ ด้วยลอจิก 2 ยูนิต ซึ่งจะมีความสัมพันธ์กับฟังก์ชั่นหลักทั้ง 2 ของตัวรับของโมเด็ม โดยที่ยูนิตแรกคือ "demodulate" ซึ่งจะทำการรับรูปคลื่นสัญญาณอะนาลอกจากอินพุทและทำการ demodulate สัญญาณจากพาสแบนด์ QAM ให้เป็นจุดสัญญาณเบสแบนด์ที่เป็นลักษณะอนุกรม ส่วนที่ยูนิตที่ 2 คือ "decode" ซึ่งจะนำเอาท์พุดที่จุด 2D ของdemodulate ยูนิต ซึ่งจะมีความคล้ายคลึงกับ trellis sequence ของ Constellation point และจะทำการ decode ในแต่ละ mapping frame จนกระทั่งเป็นบิทซีเควียนเดิม



รูปที่ 4.1 Block Diagram ของ Receiver.

4.1 กล่าวนำ

ในส่วนของตัวรับ V.34 transmitter. และจะทำการอินเวอร์สซีเควียนในการเข้ารหัสของการส่ง ซึ่งหลักการเหล่านี้จะยากเกินไปถ้าจะนำมาปฏิบัติเนื่องจากในการส่งสัญญาณแบบ QAM นั้นจะถูกรับกวนโดยสัญญาณต่าง ๆ มากมายในระหว่างการรับส่งสัญญาณ ซึ่งในส่วนของโมเด็มสัญญาณรบกวนนี้จะมาจากสายโทรศัพท์และ switching stations.

สำหรับโมเด็มในอุดมคติ receiver จะทำการรับสัญญาณอนาลอกที่ถูกส่งมาเป็น Symbol rate ที่แน่นอน ซึ่งการกระทำเช่นนี้จะทำให้สัญญาณ symbols ซีเควียนของ 2D สามารถกู้กลับมาได้อย่างสมบูรณ์ โดยจะมีปัญหาอยู่ 2 ประการ คือ ประการแรก Sampling rate ของตัวรับอาจจะไม่แมทซ์กับ Symbol rate ของตัวส่ง และ ประการที่สอง สัญญาณที่ตัวรับอาจจะมีรูปร่างไม่เหมือนกับสัญญาณที่ตัวส่ง ซึ่งในปัญหาประการแรกนั้นจะเป็นปัญหาที่ค่อยสำคัญ แต่จะทำให้ความหมายต่าง ๆ ของสัญญาณของตัวส่งและตัวรับมีความแตกต่างกัน

ในส่วนของ demodulate ของ V.34 receiver จะถูกออกแบบมาเพื่อทำการแก้ไขปัญหาข้อผิดพลาดต่าง ๆ ของสัญญาณ Sampling signal points โดยปัญหาประการแรกจะทำการแก้ไขโดยภาค symbol clock recovery ซึ่งขบวนการนี้จะทำการปรับ sampling rate ของตัวรับให้ตรงกับ sybol rate ของสัญญาณที่ได้รับ โดยปกติแล้ว sympling rate ของตัวรับมีค่า 7200 Hz. หรือประมาณ 3 sample ต่อ sybol. ใน symbol clock recovery จะทำการแก้ไข sampling rate โดยการคำนวนข้อผิดพลาดโดยการประมาณ ในภาค 144 - tap adaptive equalizer เป็นการแก้ไขปัญหาในประการที่ 2 ซึ่งเป็นปัญหาที่เกิดในช่องสัญญาณ

ในส่วนของ decoder ของ V.34 recevier จะทำการประมวลสัญญาณ 2D points ซึ่งจะถูกส่งกลับโดยส่วน demodulate ซึ่งโดยภาพรวมแล้วในส่วนนี้จะเป็นส่วนของ Viterbi decoder ซึ่ง Viterbi decoder เป็นหลักการที่ใช้ในการหาค่าของข้อมูลที่ถูกต้องที่สุด inverse - mapping และ unshell - mapping จะทำการแปลงข้อมูลให้กลับมาเป็นเหมือนเดิม

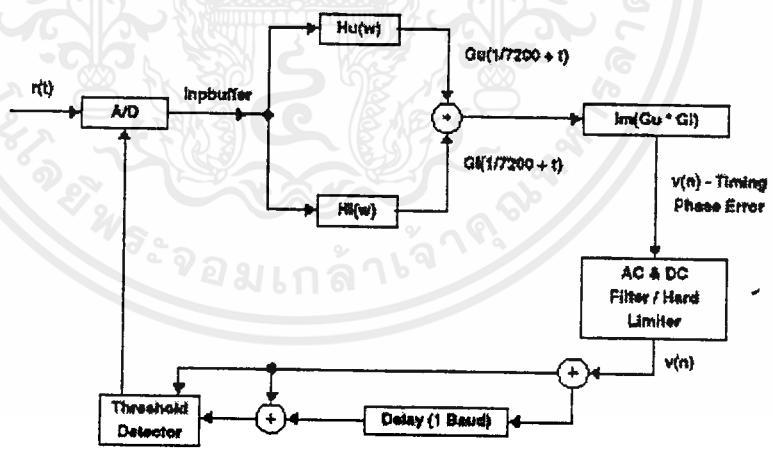
4.2 Demodulate

สัญญาณอนาลอกที่รับเข้ามาจะถูก demodulate ในส่วนนี้เพื่อให้ได้ข้อมูลที่แน่นอนของการ decoder ซึ่งการ decoder นี้จะได้ผลออกมาเป็น 2 ขั้นตอน ในขั้นตอนแรกส่วน demodulate จะทำการชดเชยส่วนที่ผิดพลาดของสัญญาณที่เข้ามา ซึ่งเป็นที่เรียกว่า symbol clock recovery ซึ่งในส่วนนี้จะทำการปรับแต่ง sampling rate ของสัญญาณ analog interface chip (AIC) เพื่อให้ได้ sample ที่แน่นอนเพื่อ

นำไปใช้ในภาค analog - to - digital (A/D) ซึ่งจะได้ตำแหน่งที่แน่นอน ปัญหาประการที่ 2 ซึ่งเกิดขึ้นในช่องสัญญาณจะถูกแก้ไขปัญหาโดยส่วนของ adaptive equalizer ในส่วนนี้จะทำการเก็บค่าสัมปสิทธิ์ สัญญาณอนาลอกที่เข้ามา และกระทำการกำจัดสัญญาณรบกวนที่เกิดจากช่องสัญญาณ

4.2.1 Symbol Colck Recovery

จาก Sybol rate ที่ 2400 Symbol / sec และ sampling rate ที่ 3 sample ต่อ symbol โดย AIC จะตั้งค่า sample ไว้ที่ความถี่ 7200 Hz. จะทำให้ได้ค่าตัวอย่างของ Sampling คือ 1 / 7200 t เมื่อ t เป็น clock phase. โดยเฟสจะเปลี่ยนแปลงตามค่า offset ระหว่าง transmitter และ receiver โดย symbol clock recovery ลูปนี้จะทำการปรับ AIC sampling frequency เพื่อให้แน่ใจว่า clock phase (t) มีค่าเป็นศูนย์ ในทุก ๆ สัญญาณ clock recovery ที่ผิดพลาดจะเป็นสาเหตุให้ภาพรับ ทำการเลื่อนตำแหน่งสัญญาณอ้างอิงจากศูนย์กลางของดีเลย์ไลท์ ซึ่งเป็นผลทำให้ การรับสัญญาณเกิดข้อผิดพลาด



รูปที่ 4.2 Symbol Clock Recovery Block Diagram

4.2.2 Phase - Splitting Fractionally - Spaced Equalizer

วัตถุประสงค์ของ Equalizer นี้จะทำการชดเชยทางแอมพลิจูด และ phase distortion ของสัญญาณ ซึ่งความถี่ของช่องสัญญาณและสัญญาณรบกวนไม่สามารถทราบได้ซึ่งเป็นสิ่งจำเป็นในการ Equalizer จะใช้ค่า least - mean - square (LMS) adaptive - tap- adjustment ซึ่งขบวนการนี้จะแสดงในรูปที่ 4.3 ในช่วงแรก ค่าผิดพลาดของสัญญาณเบสแบนด์ จะคำนวนโดยเปรียบเทียบ เอาท์พุทจาก demodulate filter กับ ideal constellation point. โดยหลักการแล้วจะมีผลกระทบน้อยที่สุดของสัญญาณผิดพลาด เบสแบนด์ จากการเพิ่มค่า complex tap h(m,n) = hr 1(m,n) + j* hi 1(m,n) โดยอยู่ในรูป gradient โดยมีสมการดังนี้

$$\frac{dB}{dh} = \frac{dB}{dhr1} + j * \frac{dB}{dhi1} = -2 * E\left\{e(nT) * inpbuffer\left(nT - m\frac{T}{3}\right)\right\}$$

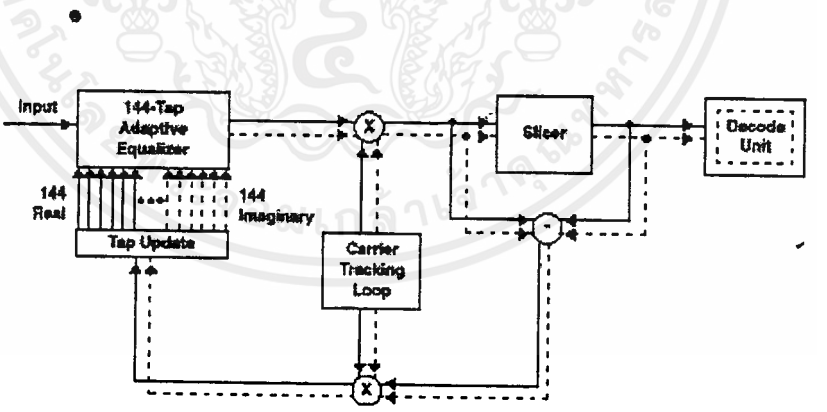
เมื่อ e(nT) เป็น passband error.

inpbuffer เป็น 144 - word buffer เมื่อได้รับข้อมูล sample data มาเก็บไว้

n เป็น ค่า time constant

T เป็น ค่าที่มากกว่า 1 baud rate

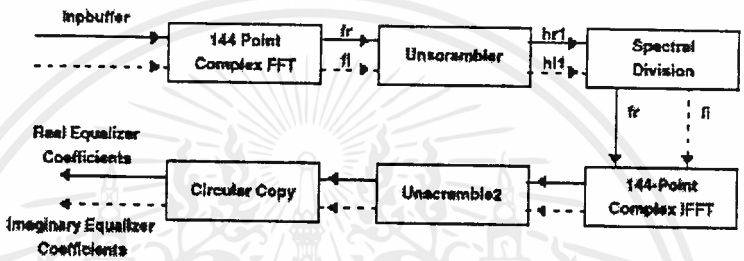
m เป็น ค่าตั้งแต่ 0 - 143



รูปที่ 4.3 Equalizer Adaptation - Loop Block Diagram

4.2.3 Fast Equalizer.

ใน Fast Equnlizer จะทำการปรับแต่งชุดข้อมูลโดยบล๊อคไดอะแกรม จากรูปที่ 4.4 โดยในช่วงแรกสัญญาณอินพุทจากดีเลย์ไลท์ ซึ่งได้มาจากการนับจำนวนของ Baud หลังจากดีเทค พาหะของสัญญาณโดยส่วนนี้จะทำการคำนวนค่าสัมประสิทธิ์ Equalizer โดยการแบ่งสเปกตรัม หลังจากนั้นจะได้ค่าสัมประสิทธิ์ของข้อมูลที่ได้คำนวนมา ซึ่งในส่วนนี้จะทำการ Cyclic equalization สัญญาณคาบ โดยคาบนี้จะมีความยาวเท่ากับ equalizer ดีเลย์ไลท์



รูปที่ 4.4 Fast Equalizer Block Diagram

จากบล๊อคไดอะแกรม input buffer จะมีแต่ส่วน real part ของข้อมูลและในส่วนของ immaginavy part จะถูกเซตให้เป็นศูนย์หมด หลังจากนั้นทำการ fast fourier transform ข้อมูลผลที่ได้จะส่งไปยังส่วนของ descramble โดยสมการที่ 4.2 นี้ จะเป็นค่าของความถี่ของสัมปสิทธิ์ equalizer C (i)

$$C(i) = \frac{B(i \bmod 48)^* D(i)^*}{D(i \bmod 48)^2 + D(48 + i \bmod 48)^2 + D(96 + i \bmod 48)^2}$$

เมื่อ i เป็น ค่าตั้งแต่ 0 - 143

B (i) เป็น FFT ของสัญญาณคาบ ซึ่งได้มาจกาการส่งของ transmitter

จากส่วนของ FFT จะใช้ในการคำนวน Inverse fourier transform (IFT) ของ C ซึ่งเป็นค่าสัมปสิทธิ์ equalizer ใน time domain ในส่วนของ Unscramble FFT output จะถูกเรียกว่า unscramble 2 (fr.) โดยเอาท์พุทจะถูกแทนเป็น gr ซึ่งจะถูกทำการคัดลอกที่ hr 1 ใน circular fashiun แล้วแทนค่าที่มากที่สุดจาก gr จนถึงศูนย์กลางของ hr1 ท้ายที่สุด unscramble 2 จะถูกเรียกใหม่เป็น fi และ ผลลัพท์ gr จะถูกคัดลอกเป็นวงกลมเป็น hi 1 ซึ่งมันจะหมุนเป็นจำนวนที่เท่ากัน

4.3 Decoder

เอาท์พุทของส่วน domodulate จะเป็นซีเควียนของ noise - corrupted 2D - points ในขั้นตอนแรกของส่วน decoder จะเป็นการตัดสินใจความเกี่ยวข้องกันของ constellation point ในอุดมคติกับสัญญาณรบกวน

4.3.1 Viterbi Decoder

ขบวนการของ Viterbi จะถูกใช้สำหรับอธิบาย ซีเควียนที่มีความคล้ายคลึงกันของการส่งสัญญาณ ซึ่งปัจจัยสำคัญในการดำเนินการของ Viterbi ก็คือ ซีเควียนในอุดมคติของ trillis ซึ่เป็นตัวกำเนิดของการส่งของ trellis encoder

ในการเลือกจุดสัญญาณในตัวส่งจะถูกควบคุมในส่วนการดำเนินการของ 16-state convolutional encoder ซึ่งใน convolutional encoder จะกระทำเพียง 2 อินพุทบิท ซึ่งก็หมายความถึงมีสเตทของการส่งของแต่ละสเตท โดยสามารถสร้างแบบจำลองได้ว่ามีความแตกต่างของแต่สเตท ในทุก ๆ เอนโคด์จะมีเอาท์พุทเพียง 1 เอาท์พุทเท่านั้นที่ใช้ในการเลือก 4D point จากอินพุททั้ง 2 ซึ่งเหล่านี้คือ 3 บิท 2 อินพุทบิท และ 1 เอาท์พุทบิท ซึ่งเป็นตัวเลขไบนารี่ของ ซับเซตของ eight different 4D โดยซับเซตนี้จะเป็นลักษณะเฉพาะของ trellis encode. ถ้าหากว่าตัวรับสามารถพิจารณาว่าสเตทของ trellis encoder นี้สามารถกำจัดซับเซตของจุดต่าง ๆ เมื่อมันมีซีเควียนที่คล้ายคลึงกัน

Viterbi เป็นวิธีที่ใช้ในการกำจัดจุดต่าง ๆ ซึ่งมันจะเก็บ large table ในหน่วยความจำ และพิจารณาความน่าจะเป็นของเหตุการณ์ของ 4D symbols อย่างจำกัด โดยทางปฏิบัติจะเก็บข้อมูลของ 16 4D symbols และที่แต่ละ 4D symbols มันจะเก็บ 16 entries ที่สอดคล้องไปยัง 16 สเตทใน convolutinal code. แต่ละสเตทที่เข้ามาจะบรรจุตัวชี้ที่มีความคล้ายคลึงของสเตทต่าง ๆ จนกระทั่ง ตัวชี้ 4D constellation ในอุดมคติให้สอดคล้องกับเส้นทางที่สเตทนั้น ๆ

ใน Viterbi algorithm จะทำการเริ่มขบวนการทุก ๆ ช่วงที่ตัวรับมีการ demodulate two noise - corrupte 2D symbol โดยจะสมมุติว่าตัวรับได้มีการเริ่มทำงานและ Viterbi table ก็จะทำการใส่ข้อมูลลงไป ในขั้นตอนแรกจะเป็นการควอนไตซ์ของ noise - corrupt point ที่อยู่ใกล้ที่สุดของแต่ละซับเซตของ 8 possible 4D ซึ่งข้อผิดพลาดของแต่ละ 8 subset จะถูกเรียกว่า branch error.

ในแต่ละสเตทจะมีการสะสม path metric และข้อมูลภายใน paht metric โดยที่ path metric จะทำการรวม branch error ทั้งหมดเข้าด้วยกันแล้วสร้างเป็น trellis path terminate ที่แต่ละ สเตท โดยที่ path branch ทั้งหลายจะสอดคล้องโดยตรงกับ 4D symbol และการหา path และ path metric ที่น้อยที่สุดจะเป็นผลลัพธ์ในซีเควียนของจุดที่คล้ายกัน

ในการคำนวนข้อมูลใน path metric ของแต่ละสเตท Viterbi algorithm จะทำการตรวจ four possible state ของแต่ละสเตท โดยข้อมูลใน path metric จะถูกปิดถ้าผลรวมของ four path metric และ branch error มีค่าน้อยที่สุดของ 16 สเตท ทั้งหมดโดย Viterbi algorithm

4.3.2 Inverse Precoder

เอาท์พุทของ Viterbi decoder จะสอดคล้องกับการประมาณของ trellis ซีเควียนที่ถูกส่งออกมาเพราะว่า non linear percode ของตัวส่ง แต่ถึงอย่างไรก็ตาม trellis ซีเควียนนี้ไม่จำเป็นต้องเป็นซีเควียนที่แท้จริง เพราะว่ามันจะถูกเลือกโดย point - select ของตัวส่ง ใน trellis ซีเควียนจาก Viterbi decode จะถูกวนกลับขบวนการโดยการวนกลับของ noise whitening filter ของตัวรับ

4.3.3 Inverse Mapper

หลักการของ inverse mapper คือการควบคุมอินพุทบิท (S,Q และ I bit groups) ที่เข้ามาใน mapper จะเป็นซีเควียนที่แม่นยำที่ได้รับเอาท์พุทจาก point - select state ของตัวส่ง ในการกู้ข้อมูลเดิมกลับมาจะมีความจำเป็นที่จะต้อง invert ขบวนการของ differential encoder โดยใช้ shell mapper, the simple mapper, scrambler และ parser

บทที่ 5 ทฤษฎีการทดลอง

5.1 TRELLIS (CONVOLUTIONAL) ENCODING

5.1.1 ทฤษฎี

คอนวูลูชั่นเอนโค๊ต เป็นฟังก์ชั่นของจำนวนอินพุทบิท (N) และจำนวนของเอาท์พุทบิท(M) และ constant length (K) ซึ่งสามารถนำค่าของ N, M, K นำมาสร้างฟังก์ชั่นได้โดยสมบูรณ์ ในการสร้างฟังก์ชั่นของการเข้ารหัสจะเป็น impulse response ของการเข้ารหัสซึ่งเมื่อลำดับของบิททางอินพุทเป็น 1 และตามด้วย 0 ดังนั้น สมการของการเข้ารหัสสามารถเขียนได้เป็น

$$V = U * G \qquad \ldots.(5.1)$$

เมื่อ * เป็นการคอนวูลูชั่น

V คือ เอาท์พุท

U คือ อินพุท

G คือ เจนเนอเรเตอร์โพลิโนเมียล

ในการคอนวูลูชั่นเอนโค๊ตนั้นจะนำ redundant บิท มารวมเข้าด้วยกันกับขบวนของสัญญาณข้อมูล (signal data stream) ซึ่งการรวมบิทของสัญญาณนี้จะเป็นการเพิ่ม BER (bit error rate) ซึ่งทำให้ตัวเข้ารหัสมีเอาท์พุทบิทต่ออินพุทบิทในการส่งมีกำลังเฉลี่ยเท่ากัน เมื่อเป็นเช่นนี้ การลดอัตราสัญญาณต่อสัญญาณรบกวน (SNR) จะเกิดขึ้นเมื่อทำให้กำลังของสัญญาณต่อบิทลดลง และยังเป็นการเพิ่ม BER ถึงอย่างไรก็ตาม เมื่อนำ คอนวูลูชั่นเอนโค๊ตรวมกับ redundant บิท ของขบวนข้อมูล ซึ่งจะเป็นการเพิ่ม SNR เกนของโค๊ตที่นำมาประกอบกับคอนวูลูชั่นเอนโค๊ตจะเป็นดังนี้

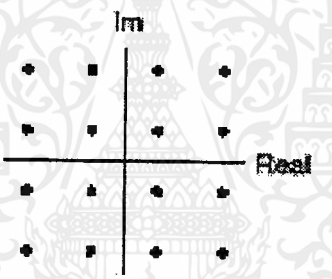
$$\text{Coding Gain} = 10 \log_{10} d^2\text{min/Pav)encoded}(d^2\text{min/Pav)unencode} \qquad \ldots.(5.2)$$

เมื่อ d^2min คือ ระยะทางที่สั้นที่สุดที่มีความเป็นไปได้

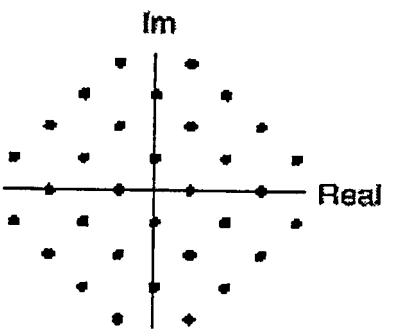
Pav คือ กำลังเฉลี่ยของที่ว่างของสัญญาณที่จะเข้ารหัส

ระยะทางที่สั้นที่สุดได้แสดงในรูปที่ 5.1 ในรูปที่ 5.1 คือ 16 QAM signal constellation ซึ่งนำ 4 บิท แทนเป็น 1 จุดบน constellation ส่วนในรูป 5.2 คือ 32 QAM constellation ซึ่งนำ 5 บิทแทนเป็น 1 จุดบน constellation สำหรับในกรณีของ 16 QAM นั้น ระยะทางที่สั้นที่สุดจากจุดหนึ่งถึงอีกจุดหนึ่งจะเป็น 2

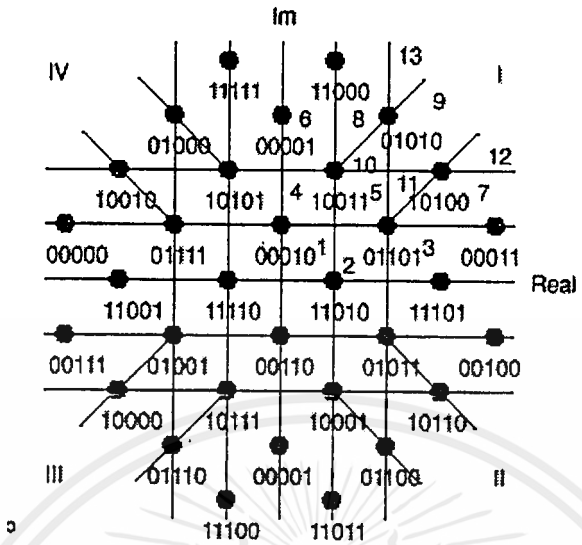
สำหรับในกรณีของ 32 QAM นั้น จะนำบิทที่มีค่ามากที่สุดเป็น redundant บิท ซึ่งได้ถูกนำมาคอนวูลูชั่นเอนโค๊ดแล้ว ดังนั้นอัตราข้อมูลที่แท้จริงจะเหมือนกันกับกรณี 16 QAM และระยะทางที่สั้นที่สุดของ 32 QAM นี้จะมีค่าเป็น $\sqrt{10}$ ดังนั้นค่าในสมการ (5.2) จะเป็นผลมาจากการนำค่าประมาณ 4 dB ของ coding Gain ใน 32 QAM นี้จะใช้ TCM ซึ่งจะมีผลทำให้ลดค่า BER ลงมากกว่าในกรณี 16 QAM ที่กำลังเฉลี่ยของสัญญาณเท่ากัน



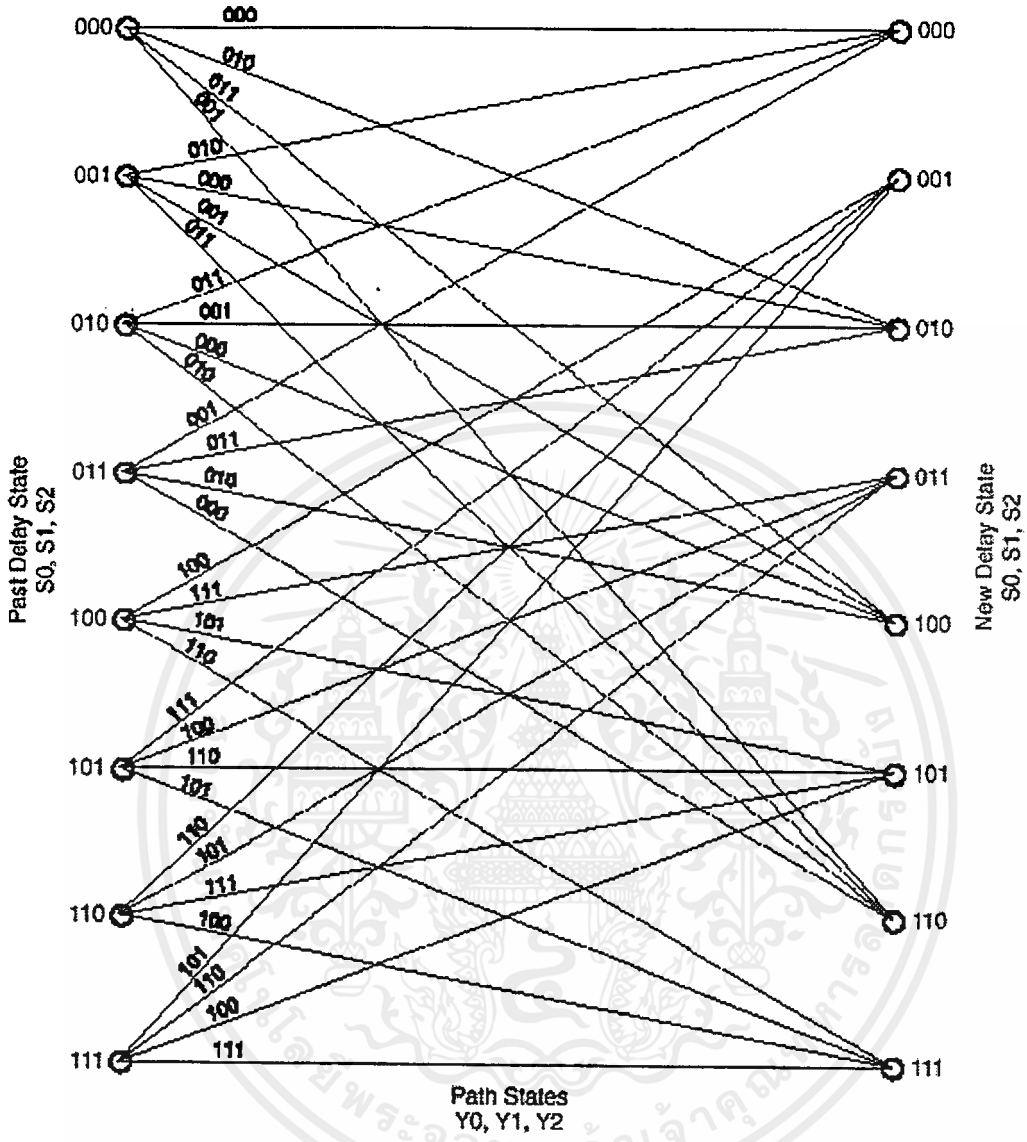
รูปที่ 5.1 16 QAM Constellation



รูปที่ 5.2 32 QAM Constellation



รูปที่ 5.3 Constellation Region



NOTE: Finite-state diagram for the convolutional encoder showing the relationship between delay and path states. Not all delay states can be reached from a previous delay state.

รูปที่ 5.4 Trellis Diagram

5.2 VITERBI DECODING

5.2.1 ทฤษฎี

ในการถอดรหัสแบบ VITERBI จะใช้รูปแบบของ trellis และ ข้อมูลทางอินพุท จะมีนิยามคล้ายกับทาง trellis โดยที่เวลาเอาท์พุท (to) จะมีผลมาจากการออกแบบโดยตัวถอดรหัสที่ข้อมูลของการรับที่เวลา N time periods ที่คาดว่าจะเกิดขึ้นนั้น ก็หมายถึงว่า เวลาที่เอาท์พุท (to) มีความสำคัญอย่างยิ่งในการดีเลย์โดย N time periods โดยที่ N จะหมายถึง constant length ของโค๊ต และการดีโค๊ตในระยะที่ใกล้และเหมาะสมที่สุด ซึ่งก็คือ 4 หรือ 5 ครั้งของ constant length

สิ่งที่มีความคล้ายกับ trellis ก็คือ ระยะทางที่สั้นที่สุด (minium - distance) ของข้อมูลอินพุทที่อยู่ใกล้กับตัวรับที่สุดใน Euclidean distance หรือ กล่าวอีกนัยหนึ่งว่า Viterbi algorithm ที่ระยะทางที่น้อยที่สุดคือ

$$d(r,v) = \sum_{i=0}^{N-1} d(ri,vi) \qquad \ldots.(5.3)$$

เมื่อ ri = ตัวรับ

vi = ซีเควนสัญญาณดีโค๊ต

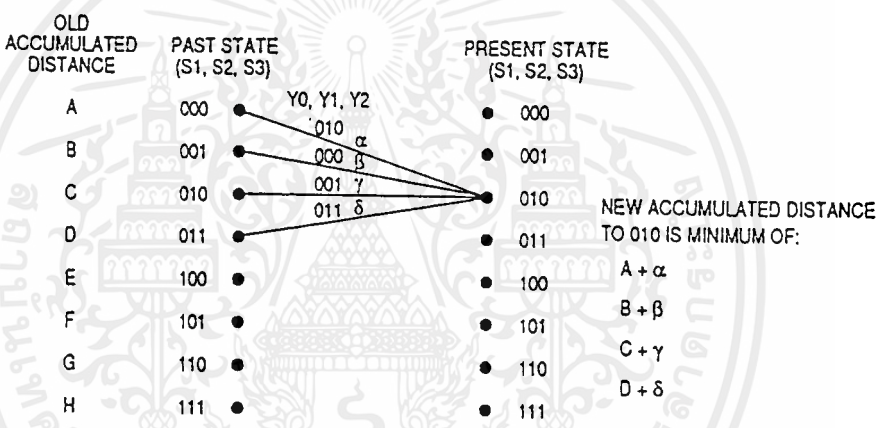
ในแต่ละ time period ที่ทุกดีเลย์สเตทใน trellis จะประกอบด้วยเส้นทางมากมาย แต่จะมีเพียงหนึ่งเส้นทางเท่านั้นที่จะมีระยะทางที่น้อยที่สุดในแต่ละดีเลย์สเตท ดังนั้นดีเลย์สเตทและระยะทาง แอกคิวมูเลตที่น้อยที่สุดจะเป็นจุดเริ่มต้นและใน time period ดังกล่าว จะเป็นเส้นทางที่มีระยะทางสั้นที่สุดของเส้นทางที่แล้ว past N-1 time periods ของ trellis ซึ่งระยะทางที่สั้นที่สุดของดีเลย์ต่อไป จะถูกนิยามโดย การหาค่าของอินพุทที่ซึ่งเป็นจุดบน constellation ในแต่ละเส้นทางที่ใกล้กันที่สุด ดังนั้นพื้นฐานของรูปแบบ trellis และ ระยะทางที่สั้นที่สุดจะนิยามได้โดยระยะแอคคิวมูเลทที่สั้นที่สุดในแต่ละดีเลย์สเตท โดยขั้นตอนของการดีโค๊ตข้อมูล สามารถแบ่งได้ 3 ขั้น ดังนี้

1. ที่แต่อินพุทให้ทำการคำนวณสเตทของเส้นทางที่มีระยะสั้นที่สุดที่สอดคล้องกับ Eudideand distances และทำการเก็บรักษาค่าไว้ในแต่ละสเตท
2. ทำการคำนวณระยะ แอคคิวมูเลทของแต่ละดีเลย์สเตท โดยการรวมระยะทางของแต่ละสเตทที่ไปสู่ดีเลย์สเตท ถ้าเมื่อเส้นทางของสเตทอยู่ที่จุดเริ่มจะเป็นระยะทางที่สั้นที่สุดและเก็บรักษาค่าเส้นทางของสเตทและ สเตทดีเลย์ ไว้

3. ทำการค้นหาดีเลย์สเตท กับระยะแฮคคิวมูเลทที่น้อยที่สุด และ ช่วงดึงกลับ N time ในการอ่านของแต่ละเส้นทางสเตท ซึ่งจะเป็น เอาท์พุทของตัวดีโค๊ดของ time period นั้น

ในรูปที่ 5.5 แสดงเส้นทางดีเลย์สเตท 010 ของ V.34 trellis และระยะทางสั้นที่สุดของ 010 ที่ถูกปิดจากเส้นทางต่าง ๆ

เมื่อเส้นทางของระยะทางที่สั้นที่สุดที่พบในแต่ละดีเลย์สเตท โดยจะถูกนำมาจาก ดีเลย์สเตทล่าสุด ซึ่งจะถูกนำไปเก็บ (เช่น 001 ในรูป 5.5 แทนด้วย C + r ซึ่งเป็นส่วนที่น้อยที่สุด) ดังนั้นใน N time period ที่เอาท์พุทสามารถดูได้จากจุดสิ้นสุดของเส้นทางของระยะทางที่สั้นที่สุดที่ to ÷ N โดยจะเก็บค่าของเส้นทางที่มีระยะทางสั้นที่สุด (Y0n, Y1n, Y2n) ในแต่ละดีเลย์สเตท และดีเลย์สเตท (S1,S2,S3)



รูปที่ 5.5 Possible Path to State 010

5.2.2 Initialization

ในการเริ่มต้น X และ Y ซึ่งเป็นส่วนประกอบของจุดใน constellation จะถูกเก็บไว้ในภายในหน่วยความจำ วงจรโมดูโลจะทำการเซตไว้เพื่อการโค๊ดของแต่ละเส้นทาง โดย distance table จะเริ่มจากจุดนี้ โดยจะเริ่มจากสเตท 000 นั้นคือ ระยะทางแฮคคิวมูเลทจะทำการบรรจุค่า 0 และก็คือการเริ่มต้นเงื่อนไขของดีเลย์ในตัวเข้ารหัสเป็น 000 ในการเซตระยะทางแฮคคิวมูเลทที่มีค่ามากจะทำให้แน่ใจว่าจะเป็นเส้นทาง 000

5.2.2.1 การหาระยะทางที่สั้นที่สุด

ใน routine นี้จะทำการวิเคราะห์ถึงข้อมูลอินพุท และ Euclidean distance ที่อยู่ใกล้จุดที่สุดของแต่ละ 8 เส้นทางของสเตท เมื่อ Euclidean distance ได้นิยามดังนี้

$$d = \sqrt{(Xc-Xi)^2 + (Yc-Yi)^2} \quad \ldots(5.4)$$

เมื่อ Xc และ Yc คือ โคออดิเนท X และ Y ของจุดบน constellation

Xi และ Yi คือ โคออดิเนทของข้อมูลอินพุท

ซึ่งในการคำนวณนี้สามารถบอกถึง Euclidean distance ของแต่ละจุดในแต่ละเส้นทางของสเตท และ ระยะทางที่น้อยที่สุดที่ทำการคำนวณที่จะเก็บไว้ในเส้นทางของสเตท ในรูปที่ 5.6 เป็นที่ตั้งของจุดบนเส้นทางของสเตท (Y0n, Y1n, Y2n) 110 โดยขอบเขตจะเท่ากับการแยก 4 จุดที่แสดงไว้ใน dashed lines ในรูปที่ 5.6 แสดงถึงจุดของเส้นทางของสเตทที่อยู่ชิดกับอินพุทที่ใช้ในขอบเขต ในรูปที่ 5.7 เป็น 8 จุดที่อยู่ใกล้กันอาจเขียนเป็น 00010, 00101, 01010, 01101, 10011, 10101, 11000, และ11111

5.2.2.2 การหาระยะทางแอคคิวมูเลทของแต่ละสเตท

ในการวิเคราะห์ V34 trellis ที่กล่าวมาแล้วเป็นเครื่องแสดงให้เห็นว่าจำนวนของเส้นทางของสเตท (4) ที่แต่ละดีเลย์สเตทจาก time period ต่าง ๆ ตารางที่ 3-1 แสดงถึงการนำมารวมกันของแต่ละดีเลย์สเตท และเส้นทางของสเตทที่กระทำแต่ละดีเลย์สเตทที่ time period ต่าง ๆ ซึ่งตารางนี้จะทำการจัดข้อมูลของโค๊ดที่ต้องการยกระดับให้เหมาะสมที่น้อยที่สุดของดีเลย์สเตท ถ้าของเป็น Even ดีเลย์สเตทจะดูได้ที่ 000, 001, 010, 011 แต่ถ้าเป็น Odd ดีเลย์สเตทจะดูได้ที่ 100, 101, 110, 111 ในทำนองเดียวกันเส้นทางของสเตทแต่ละ Even ดีเลย์ตัวใหม่ก็จะเป็น 000, 001, 010, 011 และเส้นทางของสเตทแต่ละ Odd ดีเลย์ตัวใหม่ก็จะเป็น 100, 101, 110, 111 ในกรณีของ Even ถ้าเส้นทางของสเตทถูกใส่ค่า 000, 010, 011, 001 โดยการเพิ่มสเตท 0 และสเตท 4 และทำการลดที่สเตท 2 และ สเตท 6 ซึ่งจะเป็นการง่ายในการคำนวณหาระยะทางแอคคิวมูเลทในแต่ละดีเลย์สเตท สำหรับดีเลย์สเตทที่ 2 (010) ตัวชี้จะถูกเริ่มต้นที่ locate 2 (เส้นทางของสเตท 010) หากทำการลดดีเลย์สเตทที่ 4 ตัวชี้จะถูกเริ่มต้นที่ locate 3 (เส้นทางของสเตท 011) ถ้าหากทำการเพิ่มที่ ดีเลย์สเตท 6 ตัวชี้จะเริ่มต้นที่ locate 4 (เส้นทางของสเตท 001) ดังนั้น การสลับเส้นทางของระยะทางแอคคิวมูเลทจะแสดงได้ดังนี้

$$d_{new} = \beta \, dold + (1-\beta) \, d\,path \quad \ldots(5.5)$$

เมื่อ $0 << \beta < 1$ คือ smoothing parameter

ในวิธีนี้ทำให้แน่ใจว่า ระยะแอคคิวมูเลทล่าสุดจะมีค่าอยู่ในขอบเขต ถึงแม้ว่าในสมการ (5.5) นำมาใช้คำนวณค่าระยะแอคคิวมูเลทที่ผ่านไปแล้ว และค่าของ β จะมีความสัมพันธ์กับ ค่าคงที่เวลา "τ" โดยที่

$$\tau = \frac{2}{1-\beta} \qquad \ldots.(5.6)$$

ในการใช้สมการนี้ 85 % ของ d_{new} จะมาจากจุดต่าง ๆ ในค่าคงที่เวลา และ ที่เหลืออีก 15 % จะถูกสนับสนุนโดยจุดต่าง ๆ ที่ τ โดยค่าของ $\beta = 0.9$ จะถูกใช้ในโค๊ดนี้ ถึงอย่างไรก็ตามควรจะพิจารณาถึงคุณสมบัติของพารามิเตอร์ด้วยว่า สามารถจะประยุกต์ใช้ได้เพียงไร

5.2.2.3 Data Out

ที่ เอาท์พุทของ Viterbi ดีโค๊ต สามารถดูได้จากเส้นทางที่สอดคล้องกับ 1 ใน 8 เส้นทาง (Y0n, Y1n, Y2n) ของแต่ละจุดบน constellation ที่แทนด้วย 5 bits โดยทั้ง 3 บิทแรกที่สอดคล้องกับเส้นทาง (Y0n, Y1n, Y2n) และ 2 บิทสุดท้าย ที่สอดคล้องกับบิทที่ไม่ได้ทำการเข้ารหัส (Q3n และ Q4n) จุดที่ใกล้กันที่สุดของ 4 จุดจากเส้นทางนี้ที่อินพุทในช่วงเวลานั้นก็คือ เอาท์พุท ของการถอดรหัส

5..2.2.4 Differential Decoding

ในการถอดรหัสของ differential encoding ถูกกระทำโดย 2 บิทแรกของ Viterbi ดีโค๊ต เอาท์พุท ที่แสดงได้ดังนี้

$$Q1n = Y1n \underline{V} Y1n\text{-}1 \qquad \ldots.(5.7)$$

$$Q2n = (Q1n \wedge Y1n\text{-}1) \underline{V} Y2_{2n-1} \underline{V} Y2n \qquad \ldots.(5.8)$$

เมื่อ Y1n และ Y2n คือ MSB ของ Viterbi ดีโค๊ตเอาท์พุท

Q1n และ Q2n คือ การรวมของ 2 บิทหลังของ Viterbi เอาท์พุท ที่สมบูรณ์ของขบวนการถอดรหัส

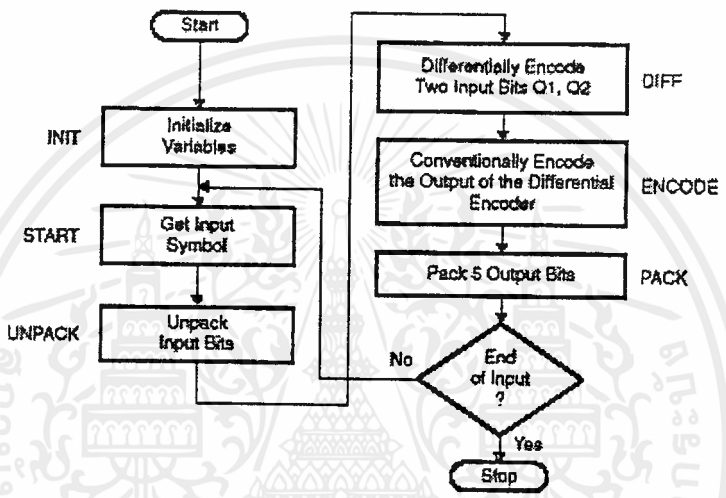
$\underline{V}$ คือ exclusive OR

$\wedge$ คือ AND

บทที่ 6 ขั้นตอนการทดลองและผลการทดลอง

6.1 Encoder Implementation

ในส่วนของโปรแกรม Encoder จะประกอบไปด้วย differential encoder และ convolutional encoder ซึ่งโปรแกรมโฟรว์ชาร์ดของการ encode ได้แสดงดังรูปที่ 6.1



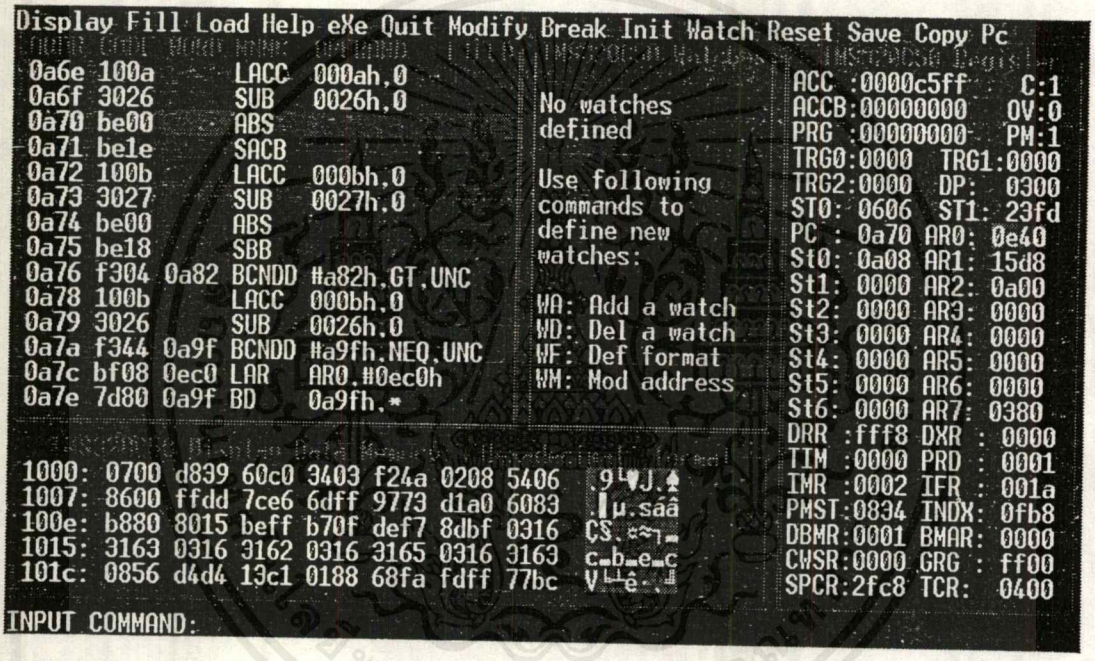
รูปที่ 6.1 Encode Flowchart

โดยในตอนแรกสับรูทีน INIT จะทำการตั้งค่า auxiliary register ให้เป็นอินพุท และเอาท์พุท และทำการรีเซตดีเรย์เสตท (50,51,52) ให้เป็น 0 เพื่อให้เสตทเริ่มแรกมีค่าที่กำหนดไว้ เพราะว่า ในการ decode point ตอนแรกจะมีดีเรย์เสตท เป็น 0

ในการ encoder อินพุท symbol จะถูกเก็บค่าไว้ในตาราง PCK-IP โดยแต่ละค่าในต่างนี้จะประกอบไปด้วย 4 bit symbol โดยการที่เราทำการเซตค่าต่างๆ ให้เป็นรูปตารางเพื่อที่จะให้ง่ายต่อการใช้งานแบบ Real-time แต่ถ้าข้อมูลที่ใช้ในการเข้ารหัสมาจาก ADC เราจะต้องใช้ Buffer ประมาณ 2 Buffer โดยจะใช้เพื่ออ่านและประมวลฟังก์ชั่นต่างๆ ของการ encode และอีกอันหนึ่งจะใช้เพื่อเป็นการอินเตอร์รัพท์ โดยการประมวลผลในกรณีของการ encoder นี้ ซึ่งข้อมูลจะต้อง synchronous กัน ข้อมูลไม่จำเป็นต้องใช้ Buffer

ในหลักการ encode ซึ่งจะประมวลผลกับไบบารี่อินพุท โดยค่าอินพุท symbol แต่ละค่าจะถูกนำออกมาจาก 4 words (โดยในแต่ละ word จะประกอบไปด้วยจำนวนบิท) โดยหลักการอันนี้จะอยู่ใน สับรูทีน UNPACK

```
UNPACK:     LACC   LOCATE
            RPTB   LOOP1
            SACL   *
            APL    * -
LOOP1:      SFR
```



รูปที่ 6.2 Program Debugger ของ UNPACK

ในส่วนของ Differential encode จะอยู่ในสับรูทีน DIFF ซึ่งจะใช้อินพุทบิท 2 อินพุท ซึ่งจะเป็นไปตามสมการดังนี้

$$Y1_n = Q1_n \oplus Y1_{n-1}$$
$$Y2_n = (Q1_n \cdot Y1_{n-1}) \oplus Y2_{n-1} \oplus Q2_n$$

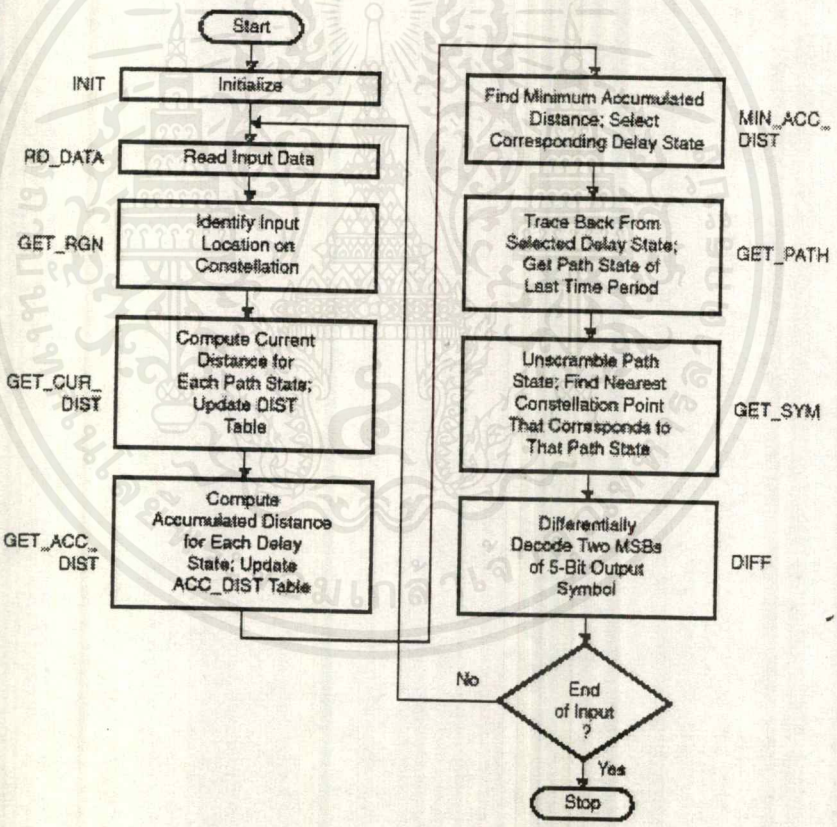
ต่อมาในส่วนของ convolution encoder จะทำการประมวลผลเอาท์พุทบิทที่ออกมาจาก Differential encoder ซึ่งจะทำให้ได้รีดันแดนท์บิท Yo และในของการ encoder เสตท (50,51,52)

จะถูกนำมาเก็บไว้ใน STATMEN และจะเปลี่ยนค่าไปเรื่อยๆ ในขณะที่เวลาในการประมวลผลผ่านไปในแต่ละครั้ง

ในตอนสุดท้าย ค่าของเอาท์พุทบิททั้ง 5 ค่า (OUTPUT + INPUT) จะถูกเก็บไว้ให้เป็น 1 word ในสับรูทีน PACK ซึ่งเอาท์พุท word นี้จะประกอบไปด้วยบิท YO,Y1,Y2,Q3,Q4 และเอาท์พุท word นี้จะถูกนำมาเก็บไว้ใน PCKD-OP

6.2 Viterbi Decoder Implementation

Viterbi Decoding จะทำการ decoding เอาท์พุทที่ออกมาจาก convolution encoder โดยหลักการของ Viterbi decode จะเป็นไปตาม Viterbi Flowchart ดังรูปที่ 6.3



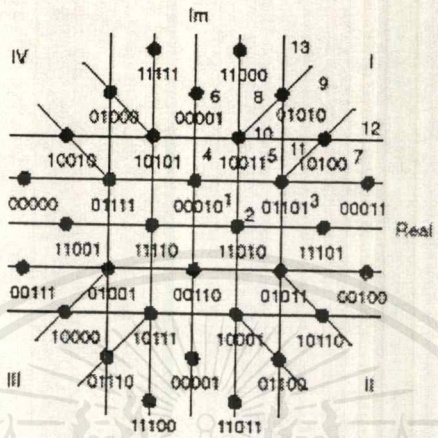
รูปที่ 6.3 Decode Flowchart

ในโปรแกรม Decoder flowchart ที่แสดงดังรูปที่ 6.3 นี้ ในแต่ละการประมวลผลจะประกอบไปด้วย ฟังก์ชั่นการทำงานต่างๆ กันดังนี้

ในตอนแรกของการเริ่มต้นโปรแกรมสับรูทีน INIT จะถูกตั้งค่าไว้เป็นตัวแปรโดยค่าระยะห่างของแอคคิวมูเลทต่างๆ จะถูกนำออกมาจากสับรูทีน ACCDIST ของทุกๆ ดีเรย์เสตท โดยเสตทแรกของการ encode จะเป็น (0 ,0 ,0) เสมอ ซึ่งเป็นเสตทที่ 0 เพื่อที่จะให้การ decoder จะทำการ decode เสตทที่ 0 เสมอในตอนแรกแอคคิวมูเลทเริ่มแรกของเสตท 0 นี้ จะถูกเซตให้เป็น 0 และส่วนอื่นๆ ที่เหลือของเสตทจะถูกตั้งค่าเป็น 0.5

ส่วนข้อมูลที่เข้ามาจะถูกนำมาอ่านโดยสับรูทีน RD_DATA โดยสับรูทีนนี้เพียงแค่ทำให้ข้อมูลสามารถนำไปใช้กับสับรูทีนอื่นๆ ได้เท่านั้นเอง โดยข้อมูลจะถูกทำการทดสอบก่อนที่จะนำไปทำการ decode โดยสับรูทีน TST_INP โดยอินพุทซึ่งจะมีค่าเท่ากับ 5 บิท symbol ซึ่งเป็นเอาท์พุทที่ออกมาจากเอาท์พุทของ encoder จะถูกมองเป็น 2 ค่า ใน XLOC และ YLOC ซึ่งจะทำการแปลในแต่ละ symbol เป็นค่าจำนวนจริง และจำนวนจินตภาพ (ซึ่งเรียกว่า ค่า XY หรือค่า IQ) โดยค่า XY นี้จะมีค่าของสัญญาณรบกวนในช่องสัญญาณรวมอยู่ด้วย ซึ่งค่าของจำนวนเชิงซ้อนเหล่านี้จะถูกเก็บไว้ในตัวแปร CURR-X และ CURR-Y ตามลำดับ โดยในสับรูทีน
RD-DATA จะทำการเก็บค่าจากตัวแปร CURR-X และ CURR-Y โดยค่า X และ Y แต่ละอันจะมีจำนวนบิทสูงสุดได้ 16 บิท

จากค่า X และ Y ที่ได้ซึ่งเป็นตำแหน่งใน constellation โดยจาก 8 constellation Point จะประกอบไปเป็นเสตทโดยแต่ละเสตทจะประกอบไปเป็น 4 constellation Point ดูรูปที่ ... โดยทฤษฎีแล้วเพื่อที่จะเลือกแต่ละกลุ่มจาก 4 Point แล้ว จะทำการคำนวณระยะห่างในแต่ละ Point แล้วเลือกไปเป็นอินพุท โดยโครงงานนี้จะทำการเลือก Point 32 Point ซึ่งประกอบเป็น constellation จากอินพุท Symbol วิธีการในการเลือกอีกอย่างหนึ่งก็คือ การกำหนดเป็นตารางแล้วทำการเลือกจากตาราง ซึ่งตำแหน่งของ constellation point จะสามารถทราบได้ล่วงหน้า ซึ่งจะทำให้ง่ายเข้าในกรณีที่ region คล้ายกัน จากรูปที่ 6.4 ในควอแดนซ์ที่ 1 จะประกอบไปด้วย 13 region ในแต่ละควอแดนซ์ของ constellation ในแต่ละ region จะถูกตั้งค่าเป็น constellation point (ซึ่งประกอบไปด้วย 8 เสตท) โดยตารางนี้จะอยู่ในรูปของสับรูทีน RECION ซึ่งเก็บค่าไว้ใน data memory ซึ่งประกอบไปเป็น 13 ส่วน ซึ่งแต่ละส่วนจะมีส่วนประกอบย่อยอีก 4 ส่วน ซึ่งแต่ละส่วนใน 4 ส่วนประกอบย่อยของโปรแกรมสับรูทีน REGION นี้ก็คือ 1 ควอแดนซ์นั่นเอง โดยแต่ละส่วนประกอบย่อยนี้จะถูกตั้งค่าให้เป็น 8 Pointer เพื่อใช้หา constellation points



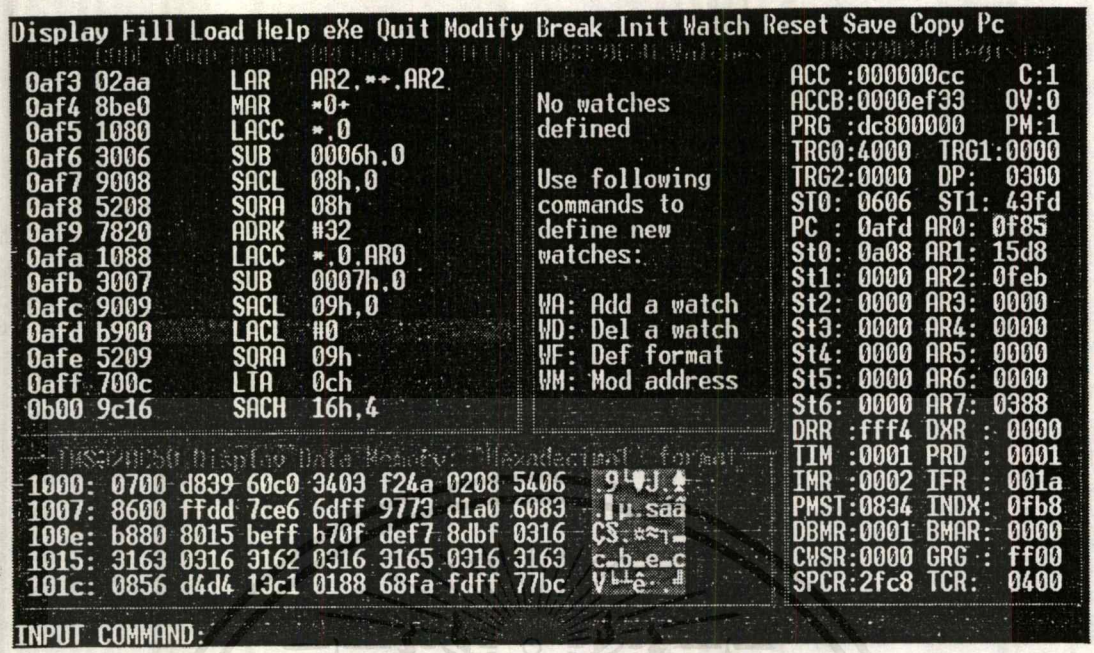
รูปที่ 6.4 Constellation Region

สำหรับสับรูทีน GET-RGN นี้จะเป็นฟังก์ชั่นที่ใช้สำหรับหา Source code ของการ decoder ซึ่งผลของ GET_RGN ก็คือ Pointer ของตารางที่อยู่ในสับรูทีน REGION

ซึ่งผลของ GET_RGN นี้จะถูกเซตให้มีค่าใน 8 constellation point ถ้า (X,Y) เป็นอินพุทที่เข้ามาในเวลาต่างๆ และ (Xk , Yk) เป็น 8 constellation point ที่อยู่ในแต่ละเสตท k (เมื่อ k=0....7) โดยระยะห่างของค่าในตารางสามารถหาค่าโดย

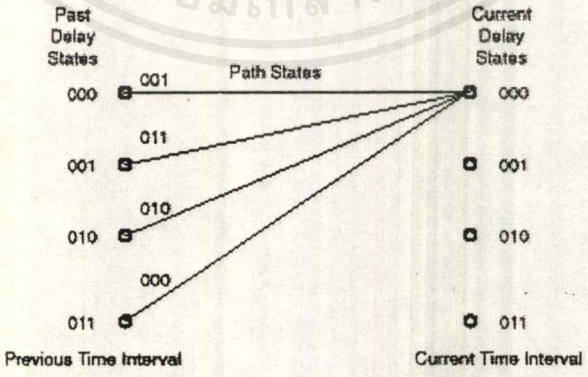
$$DIST(K) = (Xk-X)^2 + (Yk-Y)^2 \quad ; \; k = 0... 7 \qquad ....(6.3)$$

```
STATE0:
        LAR     AR2,*+,AR2
        MAR     *0+
        LACC    *
        SUB     CURR_X
        SACL    DIFF_X
        SQRA    DIFF_X
        ADRK    #32
        LACC    *,0,AR0
        SUB     CURR_Y
        SACL    DIFF_Y
        LACL    #0
        SQRA    DIFF_Y
        LTA     SMALL
        SACH    DIST,4
        MPY     DIST
        SPH     DIST
```

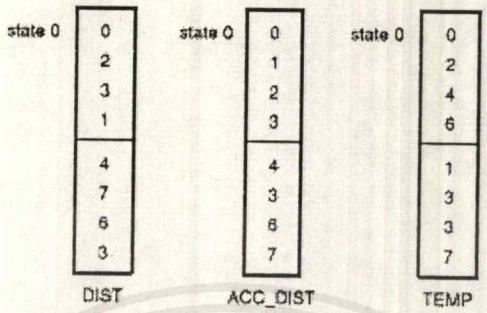


รูปที่ 6.5 Program Debugger ของ State 0

โดยค่าระยะห่างนี้จะถูกเก็บไว้ใน DIST ซึ่งมี 8 words ซึ่งใน DIST จะประกอบไปด้วยเสตทต่างๆ ซึ่งแสดงไว้ดังรูปที่ 6.7



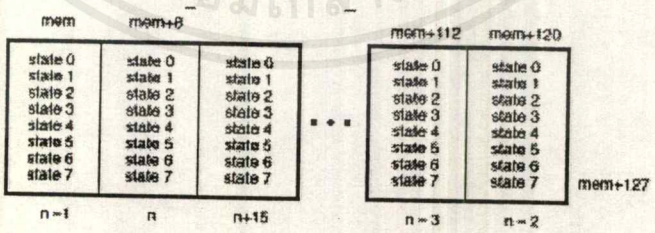
รูปที่ 6.6 Delay State Linking



รูปที่ 6.7 DIST Table Structure

จากนั้นค่าต่างๆ ใน ACCUMULATE (ค่าของระยะห่าง) ของแต่ละเสตท ซึ่งมีทั้งหมด 8 ดีเลย์ เสตท (50,51,52) จะทำการ link ไปทีละ 4 ดีเลย์เสตท ซึ่งแสดงไว้ในรูปที่ 6.6 ซึ่งค่าที่น้อยที่สุดจากการ link กันและค่าของระยะห่างของการ link กัน จะถูกบวกเข้ากับค่าของ แอคคิวมูเลทของดีเลย์เสตท

สำหรับ Pointer และ ดีเลย์เสตท จะถูกเก็บค่าไว้ใน DAST_PTH และ PAST_DLY ซึ่งการ Decode จะทำ path ก่อนหน้านำ 15 พีเรียดมาใช้ก่อนจากรูปที่ 6.8 จะมีความยาว 128 word (16 time periods x 8 state) ซึ่งสับรูทีน GET_ACC_DIST จะทำการรวมค่าข้อมูลใหม่กับแปดเสตทชุดเดิม ซึ่งมีรูปแบบดับตารางในรูปที่ 6.7



รูปที่ 6.8 128-Words Circular Buffer – Format of PAST PATH and PAST_DLY Table

ทั้ง 2 ตารางจะตั้งค่าเป็น 128-word circular buffers โดยแต่ละส่วนจะถูกแบ่งเป็น 16 ส่วนย่อย ประกอบเป็น 16 ช่วงเวลา โดย pointer จะถูกตั้งค่าตำแหน่งของช่วงเวลานั้นๆ

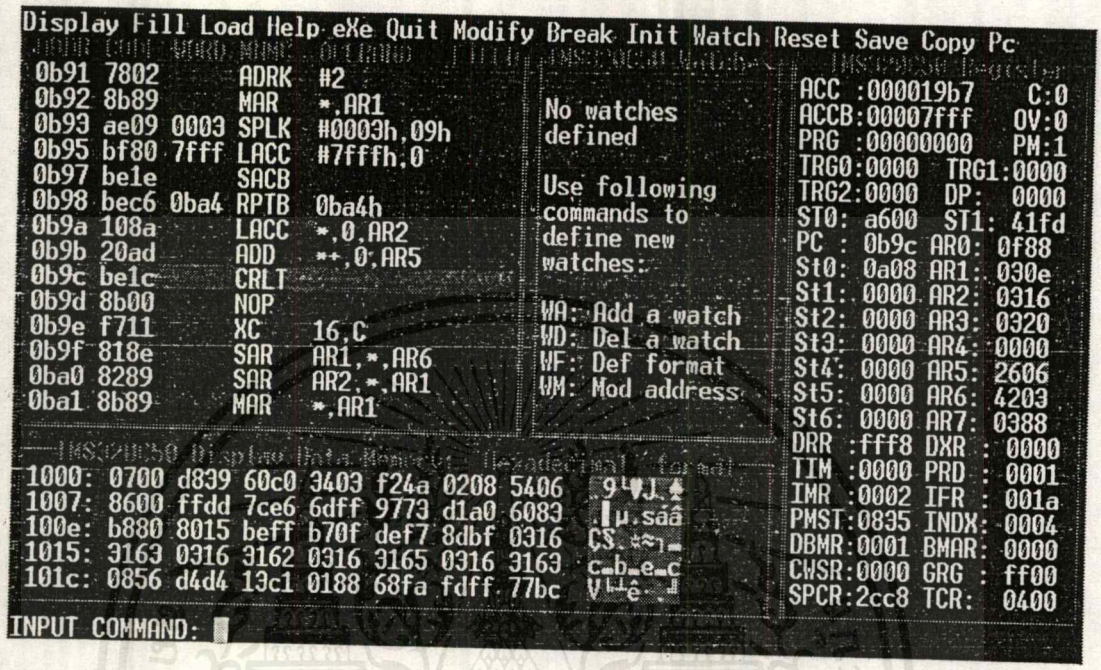
ถ้าส่วนประกอบย่อยของ DIST ที่แสดงในรูป 6.7 Path-State จะมีค่าที่เหมือนกัน 4 - word circular buffer จะถูกตั้งโดยที่ข้อมูลภายใน DIST จะถูกเพิ่มค่าขึ้น ขณะที่ข้อมูลภายใน ACC_DIST จะถูกลดค่าลง ซึ่งของการเพิ่มขึ้นและลดลงที่ผ่าน circular buffer นี้ ทำให้ delay state sequence สร้าง delay state ชุดใหม่ขึ้น (ดูได้จาก สับรูทีน GET_ACC_DIST) ซึ่ง delay state ชุดใหม่จะยังคงต้องการเส้นทางของ delay state ชุดเดิมอยู่ และที่ตาราง delay state ชุดเดิม (ACC_DIST) จะถูกตั้งเป็น circular buffer แล้ว pointer จะทำการีเซตตัวเองเพื่อที่จะรับเสตทใหม่

ในการ link แปดเสตทดีเลย์จะเก็บค่าไว้ในสับรูทีน MIN_ACC_DIST แล้วระยะห่างของแอคคิวมูเลท ACC_DIST จะถูกเปลี่ยนค่าระยะห่างแอคคิวมูเลทใหม่ ซึ่งสามารถคำนวณได้ดังนี้

$$\text{new acc dist} = 0.9 \times \text{old acc dist} + 0.1 \times \text{dist} \qquad \ldots.(6.4)$$

จากค่าต่างๆ ที่เก็บไว้ในสับรูทีน PAST_PTH และ PAST_DLY จะถูกเปลี่ยนค่าใหม่ โดยชุดคำสั่งในสับรูทีน GET_PATH

```
        RPTB    TLOOP-1
        MAR     *0+
        LACC    *0-
        SUB     #ACCDIST
        SAMM    INDX
        SBRK    7
        SBRK    1
TLOOP:
```



รูปที่ 6.9 Program Debugger ของ GETPATH

ที่ 3-bit path state (Y0,Y1,Y2) ก็คือ out put ของ decoder

สุดท้ายหลักการของ Differential decoding ซึ่งก็คือ สับรูทีน DIFF จะทำการเปลี่ยนค่า Y1 และ Y2 ไปเป็น Q1 และ Q2 ตามสมการดังนี้

$$Q1_n = Y1_n \oplus Y1_{n-1}$$

$$Q2_n = (Q1_n - Y1_{n-1}) \oplus Y2_{n-1} \oplus Y2_n$$

โดยตารางที่จะทำการเก็บค่าของสมการดังกล่าวนี้ก็คือ สับรูทีน DIFF_TBL

บทที่ 7 สรุปผลการทดลอง

จากหลักการของ V.34 โมเด็ม transmitter ซึ่งจะประกอบไปด้วย

1. ส่วนของ Parse ซึ่งมีภาค Scrambler และ Parse Data
2. ส่วนของ Point select ซึ่งประกอบด้วย Shell mapper, Differentail encoder และ Map
3. ส่วนของ Pre code ซึ่งประกอบด้วย Trellis encoder และ precoder
4. ส่วนของ Modulate ซึ่งจะทำการ modulate แบบ QAM และจากหลักการของ V. 34 และโมเด็ม Receiver ซึ่งประกอบไปด้วย

1. ภาค Demodulate ซึ่งประกอบด้วย Passband equalizer , demod และ Prediction Eror Filter
2. ภาค Decoder ซึ่งประกอบด้วย Viterbi decoder , Inverse decoder และ Inverse Map

โดยในโครงงานนี้ ได้กระทำการ encode ที่ภาค differential encoder และ trellis encoder โดยใช้ 16-state convolution code และได้กระทำการ decode ที่ภาค viterbi decoder และ differential decoder ซึ่งในการทดลองได้ใช้ DSP TMS320C50 dsk board ในการประมวลผลของโปรแกรม ซึ่งจากโปรแกรมที่ได้กระทำสามารถประมวลผลบน dsp chip ตัวนี้ได้ แต่เนื่องจาก dsp ที่ใช้อยู่มีภาคอินพุท และเอาท์พุทเป็นแบบอนาลอก จากส่วนประกอบของ V.34 trasmitter และ receiver จึงทำให้โครงงานนี้สามารถประมวลผลได้เพียงบางส่วน คือส่วนของ trellis encode , differential encode และ Viterbi decode , differential decodeได้กระทำไว้เป็นที่เรียบร้อยแล้ว

ปัญหาและอุปสรรค สัญญาณที่ได้ยังเป็นสัญญาณ Sequenceอยู่ โดยหลักกาที่แท้จริงของ V.34 โมเด็มแล้ว โครงงานนี้ จะยังขาดในส่วนของ scrambler <=> descrambler , shell mapper และ mapper <=> Inverse map , precuder <=>Inverse precude และในการ modulate แบบ QAM <=> demudulate โครงงานนี้จึงจะสมบูรณ์ แต่ว่าเวลาที่ใช้ในการทำโครงงาน จะต้องทำการศึกษาหลักการต่างๆ ของ V.34 และ TMS320C50 dsk จึงทำให้ไม่สามารถส่งผ่านข้อมูลโดยใช้หลักการของ V.34 โมเด็ม ได้สมบูรณ์

ดังนั้น ในการที่จะทำให้มีการส่งผ่านข้อมูลโดยใช้หลักการของ V.34 โมเด็มบน dsp chip เป็นไปอย่างสมบูรณ์ จำเป็นต้องพัฒนาในส่วนที่เหลือของภาค encode และ decode ต่อไป

ภาคผนวก ก

VII Constellation Sharping by Shell Mapping

Selected excerpts from Dr. Steven A. Tretter's "Fundamentals of Trellis Shaping and Precoding" on the subject of shell mapping. Used with permission.

A technique known as shell mapping or SVQ shaping has recently been proposed for constellation shaping in V.fast modems. It achieves higher shaping gains with comparable or less computational complexity than trellis shaping or precoding. Shell mapping can be easily combined with trellis channel coding. Constraints on constellation expansion ratio (CER_s) and peak-to-average ratio (PAR_2) are easily included. Shell mapping achieves shaping gains close to that of N-sphere shaping if there are no PAR_2 or CER_s constraints. Also, LTF precoders can be cascaded with shell mapped constellations to perform channel equalization at the transmitter without destroying shaping gain.

The initial use of shell mapping appears to be in a commercial modem manufactured by ESE[1] of Canada to map data blocks to 24-dimensional constellation points selected from the Leech lattice. This mapping method was also suggested in a Ph. D. thesis by Frank Robert Kschischang[2] of the University of Toronto and he also thoroughly analyzed the problem of CER_S and PAR[2] constraints using multidimensional truncated polydisks. Khandani and Kabal[3,4] also independently studied the properties of truncated polydisks but did not give them a name and they discuss some constellation addressing schemes that are not the same as the shell mapping method proposed for V.fast. Also independently, Laroia, Farvardin, and Tretter[5] at the University of Maryland discovered the same shell mapping and truncated polydisk ideas of Kschischang which they called structured vector quantizer (SVQ) shaping. In addition, they suggest grouping the shells into rings as suggested by Calderbank and Ozarow[6] to simplify the mapping complexity yet retain most of the possible shaping gain. In May 1992 Motorola Information Systems[7] (Codex) presented a paper to the CCITT V.fast committee also proposing the use of shell mapping and rings to achieve shaping gain relatively easily and stated that "Trellis shaping is no longer required."

1. G. Lang and F. Longstaff, "A Leech Lattice Modem," *Journal on Selected Areas in Communications*, Aug. 1989.
2. Frank Robert Kschischang, "Shaping and Coding Gain Criteria in Signal Constellation Design," Ph.D. Thesis, University of Toronto, Canada, Department of Electrical Engineering, Communications Group Technical Report, June 1991.
3. A.K. Khandani and P. Kabal, "Shaping Multi-dimensional Signal Spaces — Part I: Optimum Shaping Shell Mapping," Submitted to *IEEE Trans. Information Theory*, July 5, 1991, Revised April 1, 1992. Presented in part at the IEEE Int. Symp. Inform. Theory, June 24–28, 1991.
4. A.K. Khandani and P. Kabal, "Shaping Multi-dimensional Signal Spaces — Part II: Shell-addressed Constellations," Submitted to *IEEE Trans. Information Theory*, July 5, 1991, Revised April 1, 1992.
5. Rajiv Laroia, Nariman Farvardin, and Steven A. Tretter. "On SVQ Shaping of Multidimensional Constellations — High-Rate Large-Dimensional Constellations," Proceedings of the Princeton Conference on Information Sciences and Systems, March 1992. Also submitted to the *IEEE Trans. on Information Theory*, January 1992.
6. A.R. Calderbank and L.H. Ozarow. "Nonequiprobable Signaling on the GAussian Channel." *IEEE Transactions on Information Theory*, Vol. 36, No. 4, July 1990, pp. 726–740.
7. Motorola Information Systems, "Signal mapping and shaping for V.fast." CCITT working paper, Question 3/XVII, WP XVII/1, May 1992.

A. System Discription

The block diagram of the transmitter for a system using shell mapping for constellation shaping is shown in Figure VIII–1. The diagram shows the Wei 16-state 4D channel code but can easily be generalized to use any channel code. As described in Section IV, the transmitted 2D constellation is a subset of $z^2+(1/2,1/2)$ and this constellation is partitioned into four 2D subsets A, B, C, and D. The Wei encoder takes in 3 bits every 4D symbol and generates 4 output bits per 4D symbol. The first and second pairs of output bits specify the pair of 2D symbols that form the 4D subset selected by the Wei encoder.

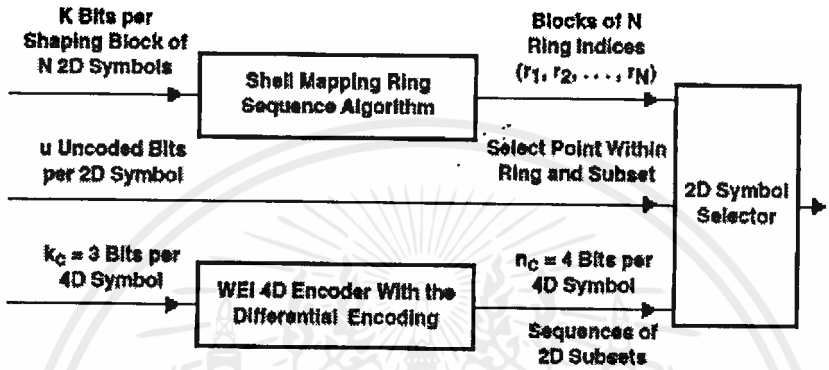


Figure VIII–1. Constellation Shaping With Shell Mapping

The 2D constellation is partitioned into M "rings" so that each ring contains the same number of points according to the approach of Calderbank and Ozarow. Furthermore, if the transmitter accepts u uncoded bits per 2D symbol, each ring must contain 2^u points from each of the four 2D subsets. Thus the total number of points in the 2D constellation is

$$L = 4M\,2^u = M\,2^{u+2} \qquad \text{(VIII–1)}$$

The ring closest to the origin should contain the 2^{u+2} constellation points of least energy. The next ring should contain the next 2^{u+2} points in order of energy, etc.

The sequence of 2D rings is determined on a block basis by the shell mapping algorithm. The transmitter takes in K bits every N 2D symbols to select a block of N rings. The shell mapping algorithm for selecting the ring sequences will be described in detail in the following sections. Basically, the algorithm maps blocks of K bits to the 2^K least energy blocks of N rings out of the M^N possible ring blocks. The algorithm requires a reasonable amount of computation and memory for tables.

A relationship between the number of rings M and the number of shaping bits K will now be determined. The K shaping bits specify 2^K blocks of N rings. Each 2D constellation is partitioned into M rings, so there are M^N possible ring blocks. Therefore, it is necessary that

$$2^K \leq M^N \text{ or } 2^{K/N} \leq M \qquad \text{(VIII–2)}$$

To achieve shaping gain, the constellation size must be expanded. Forney and Wei[8] show that most of the available shaping gain can be obtained with a constellation expansion ratio of $CER_s = 1.5$. Each shaping block the transmitter accepts Nu uncoded bits, K shaping bits, and 3(N/2) coded bits which the Wei encoder converts to 4(N/2) bits. Thus the total number of bits for constellation point selection per block is

$$B = Nu + K + 4(N/2) = N(u + 2) + K \qquad \text{(VIII–3)}$$

In an unshaped system this requires a 2D constellation of size

$$[2^{N(u+2)+K}]^{1/N} = 2^{u+2+K/N} \qquad \text{(VIII–4)}$$

According to (VIII–1), the number of points in the 2D constellation for the shell mapped system is $L = M\,2^{u+2}$ so the constellation expansion ratio is

$$CER_s = M\,2^{u+2}/2^{u+2+K/N} = M2^{-K/N} \qquad \text{(VIII–5)}$$

If CER_s is required to be no greater than 1.5, this upper bounds M by

$$M \leq 1.5\ 2^{K/N} \qquad \text{(VIII–6)}$$

Combining (VIII–2) and (VIII–6), we see that M must be limited to the range

$$2^{K/N} \leq M \leq 1.5\ 2^{K/N} \qquad \text{(VIII–7)}$$

8. G.D. Forney and L-F. Wei, "Multidimensional Constellations, Part I," *IEEE J. SAC*, Vol. 7, No. 6, August 1989, p. 887.

The selection of the number of rings M allows a tradeoff between constellation expansion and shaping gain. Using the smallest M gives the smallest constellation expansion and smallest shaping gain, while the largest M gives the largest constellation expansion and shaping gain.

The Motorola CCITT paper[7] on page 7 has some additional formulas relating M, L, and K. We will now see where they come from. First, the number of shaping bits K can be divided by the shaping block length N to give a quotient p and remainder k. Thus K can be expressed as

$$K = Np + k \text{ for } 0 \le k \le N-1 \qquad \text{(VIII–8)}$$

Substituting this form for K into (VIII–3) gives

$$B = N(u+2) + K = N(u+2) + (Np+k) = N(u+2+p) + k$$
$$= nN + k \text{ where } n = u+2+p \text{ or } u+2 = n-p \qquad \text{(VIII–9)}$$

According to (VIII–1) the number of points in the 2D constellation is

$$L = M\,2^{u+2} = M\,2^{n-p} \qquad \text{(VIII–10)}$$

The number of uncoded bits must be positive, so from (VIII–9) we find that

$$u = n-2-p \ge 0$$
$$\text{or } 0 \le p \le n-2 \qquad \text{(VIII–11)}$$
$$\text{and } n \ge 2$$

For a fixed 2D constellation size L, increasing p forces the number of rings M to increase and results in greater complexity. For a block size of N=8, Motorola recommends using $p \le 3$.

B. Weights of Ring Block, The Basic Concepts of Shell Mapping, and Some Data Table Required for Efficient Implementations

Suppose that the rings are labelled from 0 to M–1 with ring 0 being closest to and ring M–1 furthest from the origin. Let a block of ring indices be denoted by

$$r = [r_1, r_2, \cdots, r_N] \qquad \text{(VIII–12)}$$

with $r_i \in \{0, 1, \ldots, M-1\}$ for $i=1, \ldots, N$. Each ring must be assigned an integer weight $W_1(r_i)$ where the subscript 1 indicates that the argument is a one-dimensional vector. For example, assuming that the constellation point coordinates are integers, the ring weight could be the average squared distance of points in the ring from the origin. Motorola[7] suggests using the weight function $w_1(i) = i$ for $i = 0, \ldots, M-1$. Justification is given in Appendix VIII–A. The ring weights must form a nondecreasing sequence, that is,

$$w_1(0) \leq w_1(1) \leq \cdots \leq w_1(M-1) \qquad \text{(VIII–13)}$$

The weight of a sequence will be defined as the sum of the component weights, that is

$$w_N(r) = \sum_{i=1}^{N} w_1(r_i) \qquad \text{(VIII–14)}$$

The basic idea behind shell mapping is that the M^N possible ring blocks are arranged in an ordered list with lower weight blocks appearing closer to the beginning of the list. There may be many blocks with the same weight and these are called a shell. We will see how to lexicographically order the blocks in a shell. The block at the beginning of the list will be labelled or indexed by 0 and the one at the end by $M^N - 1$. The 2^K blocks closest to the beginning of the list which are also the 2^K lowest weight blocks are used as the shell sequences. Encoding is performed by using the binary K-tuples of shaping bits as the indexes of the list elements. Decoding is performed, basically, by observing that given a ring block, its index is the number of blocks below it on the list. We will see how to achieve reasonable table storage requirements by a "divide and conquer" approach where the N dimensional problem is divided into two N/2 dimensional problems, etc. The problem, then, is to efficiently assign shaping K-tuples to ring blocks and vice versa.

To perform shell mapping, the number of vectors with a given weight must be known. Let $M_p(j)$ be the number of p-tuples of ring indices $\mathbf{r} = \{r_1, \ldots, r_p\}$ with total weight $j = w_1(r_1) + \ldots + w_1(r_p)$. The weight generating function is defined to be

$$G_p(z) = \sum_k M_p(k)\, z^k \qquad \text{(VIII–15)}$$

For example, with the Motorola weight function $w_1(i)=i$, the number of 1-tuples of weight j is $M_1(j) = 1$ for $j = 0, \ldots, M-1$ and is 0 otherwise. The corresponding generating function is

$$G_1(z) = \sum_{k=0}^{M-1} z^k = \frac{1-z^M}{1-z} \qquad \text{(VIII–16)}$$

The number of (2p)-tuples of weight j can be computed from the number of p-tuples of each weight in the following way. Each 2p-tuple can be considered to be the concatenation of a pair of p-tuples. The weight of the 2p-tuple is the sum of the weights of the two p-tuples. If the first p-tuple has weight k, then the second p-tuple must have weight j–k to make the total weight equal to j. Thus the number of 2p-tuples of weight j must be

$$M_{2p}(j) = \sum_{k=0}^{j} M_p(k)\, M_p(j-k) \qquad \text{(VIII–17)}$$

This sum is just a convolution so the corresponding generating function is

$$G_{2p}(z) = G_p^2(z) \qquad \text{(VIII–18)}$$

If the shaping block length N is a power of 2, $M_N(j)$ can be found by successively applying (VIII–17) for $2p = 2, 4, \ldots, N$ since the initial sequence $M_1(j)$ is easily determined. The shell mapping encoding and decoding algorithms assume that the intermediate results $M_k(j)$ have been stored in memory for $k = 1, 2, 4, \ldots, 2^n, \ldots, N/2$ and all relevant j.

Another sequence that will be used in shell mapping is the number of ring blocks with weight less than or equal to j. This will be designated by $C_N(j)$ and can be computed from $M_N(j)$ by

$$C_N(j) = \sum_{k=0}^{j} M_N(k) \qquad \text{(VIII–19)}$$

These values should also be stored in a table.

The number of ring blocks required is 2^K. Thus, the maximum value, J, required for j is the smallest j such that $C_N(j) \geq 2^K$. Then 2^K blocks can be selected from the $C_N(J)$ blocks. J is called the SVQ threshold.

C. The Decoding Algorithm

The shell mapping technique can be best understood by first looking at the decoding algorithm, which is the method for mapping a received block of N rings back into the original input block of K shaping bits. In the receiver, the received signal is demodulated and Viterbi decoded to give a maximum likelihood estimate of the transmitted sequence of constellation points. Then blocks of N estimated 2D constellation points are quantized into blocks of ring indexes.

Given a received block r of N ring indexes, the decoding function $D_N(r)$ computes the index of r in the list. The decoding function can also be expressed as

$$D_N(r) = \{\text{number of blocks below } r \text{ on the list}\} \qquad \text{(VIII–20)}$$

Binary shaping K-tuples can be assigned to ring blocks in many ways. The method we will examine is based on a splitting algorithm that successively divides blocks into pairs of half the length until blocks of length 1 are reached. Therefore, we will assume that the original block length is a power of 2, that is, $N = 2^q$.

At each step, the i-tuples of rings will be ordered according to rules which will be given shortly. The decoding, ordering, or indexing function on i-tuples will be called $D_i(\cdot)$.

The ordering of 1-tuples is easy. According to (VIII–13), the ring weights must form a nondecreasing sequence. Therefore, 1-tuples will be listed in numerical order. That is, the one-dimensional decoding function is

$$D_1(r) = r \text{ for } r = 0, \ldots, M-1 \qquad \text{(VIII–21)}$$

This is the starting point for building higher dimensional decoding functions.

As a matter of notation, let an i-tuple of ring indexes be

$$r^{[i]} = [r_1, \ldots, r_i] \qquad \text{(VIII–22)}$$

The first and second halves of $r^{[i]}$ will be designated by

$$r_1^{[i]} = [r_1, \ldots, r_{i/2}] \text{ and } r_2^{[i]} = [r_{i/2+1}, \ldots, r_i] \qquad \text{(VIII–23)}$$

We will now see how to define a decoding function for i-tuples in terms of one for i/2-tuples. Assume $D_{i/2}(\cdot)$ is known. First, order i-tuples of rings according to the following rules for the three possible cases:

Case 1: Different Weight i-tuples

An i-tuple $u^{[i]}$ is listed below $v^{[i]}$ if $w_i(u^{[i]}) < w_i(v^{[i]})$ where $w_i(\cdot)$ is the i-dimensional weight function.

Case 2: Equal Weight i-tuples, but First Halves have Different i/2-dimensional Indexes

When $w_i(u^{[i]}) = w_i(v^{[i]})$, then $u^{[i]}$ is listed below $v^{[i]}$ if

$$D_{i/2}(u_1^{[i]}) < D_{i/2}(v_1^{[i]})$$

Case 3: Equal Weight i-tuples. Indexes of 1st Halves are the same

When $w_i(u^{[i]}) = w_i(v^{[i]})$ and $u_1^{[i]} = v_1^{[i]}$, then list $u^{[i]}$ below $v^{[i]}$ if $D_{i/2}(u_2^{[i]}) < D_{i/2}(v_2^{[i]})$.

The i-dimensional decoding function can also be expressed in terms of the following function:

$$N_i(v^{[i]}) = \{\text{number of i-tuples } u^{[i]} \text{ with } w_i(u^{[i]}) = w_i(v^{[i]}) \text{ and } u^{[i]} \text{ below } v^{[i]}\} \quad \text{(VIII–24)}$$

This quantity will be called the offset into the shell. Then, the i-dimensional decoding function can be expressed as

$$\begin{aligned} D_i(v^{[i]}) &= \{\text{number of i-tuples below } v^{[i]}\} \\ &= \{\text{number of i-tuples } u^{[i]} \text{ with } w_i(u^{[i]}) < w_i(v^{[i]})\} \\ &\quad + \{\text{number of i-tuples } u^{[i]} \text{ with } w_i(u^{[i]}) = w_i(v^{[i]}) \text{ and } u^{[i]} \text{ below } v^{[i]}\} \\ &= C_i\left[w_i\left(v^{[i]}\right) - 1\right] + N_i(v^{[i]}) \end{aligned} \quad \text{(VIII–25)}$$

The quantities $D_i(\cdot)$ for $i < N$ do not have to be computed. Also, $C_i(\cdot)$ is needed only for $i=N$. Finally, we will see in the next paragraph how to recursively compute $N_i(\cdot)$ for $i = 2, 4, \ldots, N$ using $N_{i/2}(\cdot)$ and $M_{i/2}(\cdot)$. Then $D_N(\cdot)$ can be computed by (VIII–25) for $i=N$.

The vectors counted in $N_i(v^{[i]})$ can be partitioned into the following three types:

Type 1: 1st Halves Differ in Weight

Consider the set

$$\left\{ u^{[l]} \;\middle|\; w_i\left(u^{[l]}\right) = w_i\left(v^{[l]}\right) \bigcap w_{i/2}\left(u_1^{[l]}\right) < w_{i/2}\left(v_1^{[l]}\right) \right\} \tag{VIII–26}$$

The number of elements in this set is

$$a_1 = \sum_{k=0}^{w_{i/2}\left(v_1^{[l]}\right)-1} M_{i/2}(k)\, M_{i/2}\left[w_i\left(v^{[l]}\right)-k\right] \tag{VIII–27}$$

Type 2: 1st Halves Differ but Have the Same Weight

Consider the set

$$\left\{ u^{[l]} \;\middle|\; w_i\left(u^{[l]}\right) = w_i\left(v^{[l]}\right) \bigcap w_{i/2}\left(u_1^{[l]}\right) = w_{i/2}\left(v_1^{[l]}\right) \bigcap D_{i/2}\left(u_1^{[l]}\right) < D_{i/2}\left(v_1^{[l]}\right) \right\} \tag{VIII–28}$$

According to (VIII–24), the number of choices for $u_1^{[l]}$ is $N_{i/2}(v_1^{[l]})$. The total weight must be $w_i(v^{[l]})$, so the number of choices for $u_2^{[l]}$ is $M_{i/2}[w_i(v^{[l]}) - w_{i/2}(v_1^{[l]})]$. Thus the number of type 2 vectors is

$$a_2 = N_{i/2}\left(v_1^{[l]}\right) M_{i/2}\left[w_i\left(v^{[l]}\right)-w_{i/2}\left(v_1^{[l]}\right)\right] \tag{VIII–29}$$

Type 3: 1st Halves Identical

The number of vectors in the set

$$\left\{ u^{[l]} \mid w_i\left(u^{[l]}\right) = w_i\left(v^{[l]}\right) \bigcap u_1^{[l]} = v_1^{[l]} \bigcap D_{i/2}\left(u_2^{[l]}\right) < D_{i/2}\left(v_2^{[l]}\right) \right\} \tag{VIII–30}$$

is

$$a_3 = N_{i/2}\left(v_2^{[l]}\right) \tag{VIII–31}$$

Notice, also, that the weight of $v_2^{[l]}$ is

$$w_{i/2}\left(v_2^{[l]}\right) = w_i\left(v^{[l]}\right) - w_{i/2}\left(v_1^{[l]}\right) \tag{VIII–32}$$

Adding the numbers for the three cases gives

$$N_i\left(v^{[i]}\right) = \sum_{k=0}^{w_{i/2}\left(v_1^{[i]}\right)-1} M_{i/2}(k)\, M_{i/2}\left[w_i\left(v^{[i]}\right) - k\right] + N_{i/2}\left(v_1^{[i]}\right) M_{i/2}\left[W_{i/2}\left(v_2^{[i]}\right)\right] + N_{i/2}\left(v_2^{[i]}\right) \tag{VIII–33}$$

The decoding operation is performed iteratively. First, $r = v^{[N]}$ is divided into N/2 pairs of ring indexes and $N_2(\cdot)$ is computed for each of the pairs using (VIII–33). Adjacent pairs are then combined into N/4 4-tuples and $N_4(\cdot)$ is computed for each. The doubling procedure is repeated until $N_N(r)$ is computed. Then the index $D_N(r)$ is computed by (VIII–25) with i=N.

Example VIII–1. N = 2 and Motorola Weight Function

Let the one-dimensional weight function be

$$w_1(r) = r \text{ for } r = 0, \ldots, M-1 \tag{VIII–34}$$

and designate 2-tuples by $v^{[2]} = [r1, r2]$. The number of 1-tuples of each weight is

$$M_1(k) = \begin{cases} 1 & \text{for } k = 0, \ldots, M-1 \\ 0 & \text{elsewhere} \end{cases} \tag{VIII–35}$$

For this example,

$$v_1^{[2]} = [r_1] \text{ and } v_2^{[2]} = [r_2]$$

Then, according to (VIII–33), the shell offset is

$$N_2\left(v^{[2]}\right) = \sum_{k=0}^{w_1[r_1]-1} M_1(k)\, M_1\left[w_2\left(v^{[2]}\right) - k\right] + N_1(r_1)\, M_1\left[w_2\left(v^{[2]}\right) - w_1(r_1)\right] + N_1(r_2) \tag{VIII–36}$$

Since there is a single 1-tuple of each weight, $N_1(r) = 0$, so the last line of (VIII–36) is zero and

$$N_2\left(V^{[2]}\right) = \sum_{k=0}^{r_1-1} M_1(k)\, M_1(r_1 + r_2 - k) = \sum_{k=0}^{r_1-1} M_1(r_1 + r_2 - k) \qquad \text{(VIII–37)}$$

From Figure VIII–2 it can be seen that this sum is

$$N_2\left(\left[r_1, r_2\right]\right) = \begin{cases} r_1 & \text{for } 0 \le r_1 + r_2 \le M-1 \\ M-1-r_2 & \text{for } M \le r_1 + r_2 \le r_1 + M - 2 \end{cases} \qquad \text{(VIII–38)}$$

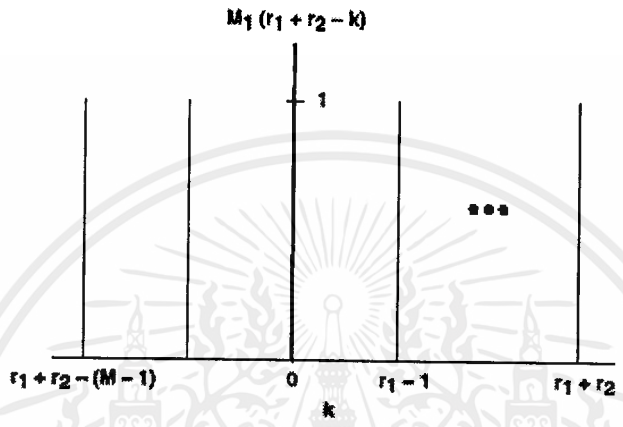
Example of Shell Sequence Ordering for N = 4 and M = 3

$D_1(r)$	r	
0	0	$M_1(0) = 1$
1	1	$M_1(1) = 1$
2	2	$M_1(2) = 1$
$D_2(r)$	r	
0	00	$M_2(0) = 1$
1	01	$M_2(1) = 2$
2	10	
3	02	$M_2(2) = 3$
4	11	
5	20	
6	12	$M_2(3) = 2$
7	21	
8	22	$M_2(4) = 1$
$D_4(r)$	r	
0	0000	$M_4(0) = 1$
1	0001	$M_4(1) = 4$ $\quad C_4(0) = 1$
2	0010	
3	0100	
4	1000	

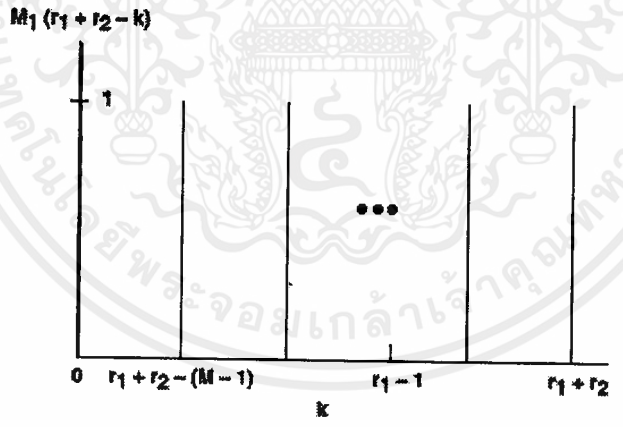
5	0002	$M_4(2) = 10$	$C_4(1) = 5$
6	0011		
7	0020		
8	0101		
9	0110		
10	1001		
11	1010		
12	0200		
13	1100		
14	2000		
15	0012	$M_4(3) = 16$	$C_4(2) = 15$
16	0021		
17	0102		
18	0111		
19	0120		
20	1002		
21	1011		
22	1020		
23	0201		
24	0210		
25	1101		
26	1110		
27	2001		
28	2010		
29	1200		
30	2100		
31	0022	$M_4(4) = 19$	$C_4(3) = 31$
32	0112		
33	0121		
34	1012		
35	1021		
36	0202		
37	0211		
38	0220		
39	1102		
40	1111		

41	1120		
42	2002		
43	2011		
44	2020		
45	1201		
46	1210		
47	2101		
48	2110		
49	2200		
50	0122	$M_4(5) = 16$	$C_4(4) = 50$
51	1022		
52	0212		
53	0221		
54	1112		
55	1121		
56	2012		
57	2021		
58	1202		
59	1211		
60	1220		
61	2102		
62	2111		
63	2120		
64	2201		
65	2210		
66	0222	$M_4(6) = 10$	$C_4(5) = 66$
67	1122		
68	2022		
69	1212		
70	1221		
71	2112		
72	2121		
73	2202		
74	2211		
75	2220		
76	1222	$M_4(7) = 4$	$C_4(6) = 76$

77	2122		
78	2212		
79	2221		
80	2222	$M_4(8) = 1$	$C_4(7) = 80$
			$C_4(8) = 81$



(a) $0 \le r_1 + r_2 \le M - 1$



(b) $M \le r_1 + r_2 \le r_1 + M - 2$

D. The Encoding Algorithm

The encoding algorithm maps binary K-tuples of shaping bits into N-tuples of ring indexes. The mapping is based on the ordering of ring blocks described in the previous section. Let the shaping bit K-tuple be the binary representation for the index $D_N(v^{[N]})$. The problem is to find $v^{[N]}$. According to (VIII–25)

$$D_N\left(v^{[N]}\right) = C_N\left[w_N\left(v^{[N]}\right) - 1\right] + N_N\left(v^{[N]}\right) \qquad \text{(VIII–39)}$$

Remember that $N_N(v^{[N]})$ is the number of blocks with the same weight as $v^{[N]}$ that are below it on the list. Also, $M_N(w_N(v^{[N]}))$ is the number of blocks with the weight of $v^{[N]}$. Therefore,

$$0 \leq N_N\left(v^{[N]}\right) < M_N\left[w_N\left(v^{[N]}\right)\right] \tag{VIII–40}$$

Also,

$$C_N\left[w_N\left(v^{[N]}\right)\right] = C_N\left[w_N\left(v^{[N]}\right) - 1\right) + M_N\left[w_N\left(v^{[N]}\right)\right] \tag{VIII–41}$$

Thus,

$$C_N\left[w_N\left(v^{[N]}\right) - 1\right] \leq D_N\left(v^{[N]}\right) < C_N\left[w_N\left(v^{[N]}\right)\right] \tag{VIII–42}$$

This shows that the index D_N will always fall in an interval bracketed by a pair of successive C_N's.

Encoding Step 1: Compute the Weight of $v^{[N]}$

Based on (VIII–42), the weight of $v^{[N]}$ is

$$w_N\left(v^{[N]}\right) = \max_n\left\{n \mid C_N(n-1) \leq D_N\left(v^{[N]}\right)\right\} \tag{VIII–43}$$

Encoding Step 2: Compute the Offset

Once the weight is known, we can compute the offset as

$$N_N\left(v^{[N]}\right) = D_N\left(v^{[N]}\right) - C_N\left[w_N\left(v^{[N]}\right) - 1\right] \tag{VIII–44}$$

According to (VIII–33),

$$\begin{aligned} N_N\left(v^{[N]}\right) = & \left\{\sum_{k=0}^{w_{N/2}\left(v_1^{[N]}\right)-1} M_{N/2}(k)\, M_{N/2}\left[w_N\left(v^{[N]}\right) - k\right]\right\} \\ & + \left\{N_{N/2}\left(v_1^{[N]}\right) M_{N/2}\left[W_{N/2}\left(v^{[N]}\right)\right]\right\} \\ & + \left\{N_{N/2}\left(v_2^{[N]}\right)\right\} \end{aligned} \tag{VIII–45}$$

Of the weight $w_N(v^{[N]})$ N-tuples, the number with the same weight as $v^{[N]}$ in the first half, that is, $w_{N/2}(u_1^{[N]}) = w_{N/2}(v_1^{[I]})$, is

$$M_{N/2}\left[w_{N/2}\left(v_1^{[N]}\right)\right] M_{N/2}\left[w_{N/2}\left(v_2^{[N]}\right)\right] \qquad \text{(VIII–46)}$$

where $w_{N/2}(v_2^{[N]}) = w_N(v^{[N]}) - w_{N/2}(v_1^{[N]})$. Thus, the sum of the second and third terms in the curly braces in (VIII–45), which is the number of N-tuples below $v^{[N]}$ with the same total weight as $v^{[N]}$ and the same weight in the first half as $v_1^{[N]}$, must satisfy,

$$0 \le N_{N/2}\left(v_1^{[N]}\right) M_{N/2}\left[w_{N/2}\left(v_2^{[N]}\right)\right] + N_{N/2}\left(v_2^{[N]}\right) < M_{N/2}\left[w_{N/2}\left(v_1^{[N]}\right)\right] M_{N/2}\left[w_N\left(v_2^{[N]}\right)\right] \qquad \text{(VIII–47)}$$

Rearranging this last inequality gives

$$\left\{N_{N/2}\left(v_1^{[N]}\right) M_{N/2}\left[w_{N/2}\left(v_2^{[N]}\right)\right] + N_{N/2}\left(v_2^{[N]}\right)\right\} - M_{N/2}\left[w_{N/2}\left(v_1^{[N]}\right)\right] M_{N/2}\left[w_N\left(v_2^{[N]}\right)\right] \qquad \text{(VIII–48)}$$

$$= \left\{N_N\left(v^{[N]}\right) - \sum_{k=0}^{w_{N/2}\left(v_1^{[N]}\right)-1} M_{N/2}(k)\, M_{N/2}\left[w_N\left(v^{[N]}\right) - k\right]\right\}$$

$$- M_{N/2}\left[w_{N/2}\left(v_1^{[N]}\right)\right] M_{N/2}\left[w_{N/2}\left(v_2^{[N]}\right)\right] < 0$$

Encoding Step 3: Finding the Weights of $v_1[N]$ and $v_2[N]$

From the inequality (VIII–48), we see that the weight of the first half of $v[N]$ must be

$$W_{N/2}\left(v_1^{[N]}\right) = \max_n \left\{n \;\middle|\; \sum_{k=0}^{n-1} M_{N/2}[k]\, M_{N/2}\left[w_{N/2}\left(v^{[N]}\right) - k\right] \le N_N\left(v^{[N]}\right)\right\} \qquad \text{(VIII–49)}$$

The weight of the second half can then be computed as

$$w_{N/2}\left(v_2^{[N]}\right) = w_N\left(v^{[N]}\right) - w_{N/2}\left(v_1^{[N]}\right) \qquad \text{(VIII–50)}$$

Encoding Step 4: Compute the Partial Offset

Now the following partial offset d can be computed:

$$d \leftarrow N_{N/2}\left(v_1^{[N]}\right) M_{N/2}\left[w_{N/2}\left(v_2^{[N]}\right)\right] + N_{N/2}\left(v_2^{[N]}\right)$$

$$= N_N\left(v^{[N]}\right) - \sum_{k=0}^{w_{N/2}\left(v_1^{[N]}\right)-1} M_{N/2}(k)\, M_{N/2}\left[w_N\left(v^{[N]}\right) - k\right] \qquad \text{(VIII–51)}$$

Encoding Step 5: Compute the Offsets of the 1st and 2nd Halves

Remember that $N_{N/2}\left(v_2^{[N]}\right)$ is the number of weight $w_{N/2}\left(v_2^{[N]}\right)$ N/2-tuples below $v_2^{[N]}$. The total number of N/2-tuples with weight $w_{N/2}\left(v_2^{[N]}\right)$ is $M_{N/2}\left[w_{N/2}\left(v_2^{[N]}\right)\right]$, so

$$0 \le N_{N/2}\left(v_2^{[N]}\right) < M_{N/2}\left[w_{N/2}\left(v_2^{[N]}\right)\right] \qquad \text{(VIII–52)}$$

Using the same reasoning, we see that the following inequality must also be true:

$$0 \le N_{N/2}\left(v_1^{[N]}\right) < M_{N/2}\left[w_{N/2}\left(v_1^{[N]}\right)\right] \qquad \text{(VIII–53)}$$

Using the Euclidean division algorithm, $N_{N/2}\left(v_2^{[N]}\right)$ and $N_{N/2}\left(v_1^{[N]}\right)$ can be found by dividing d by $M_{N/2}\left[w_{N/2}\left(v_2^{[N]}\right)\right]$. $N_{N/2}\left(v_1^{[N]}\right)$ is the remainder and $N_{N/2}\left(v_2^{[N]}\right)$ is the quotient.

Thus, to complete step 5, compute the following:

$$N_{N/2}\left(v_1^{[N]}\right) = \text{int}\left\{d / M_{N/2}\left[w_{N/2}\left(v_2^{[N]}\right)\right]\right\} \qquad \text{(VIII–54)}$$

and

$$N_{N/2}\left(v_2^{[N]}\right) = d - M_{N/2}\left[w_{N/2}\left(v_2^{[N]}\right)\right] N_{N/2}\left(v_1^{[N]}\right) \qquad \text{(VIII–55)}$$

Encoding Step 6: Iterate the Procedure

Steps 3, 4, and 5 can now be applied to the two N/2-tuples to find the weights and offsets of four N/4-tuples, and the procedure can be repeated until 1-tuples are reached. Then the ring index for a 1-tuple will be obvious from its weight $w_1(\cdot)$ and offset $N_1(\cdot)$.

Example 1. (Continued)

For the Motorola ring weight assignment, it is not necessary to decompose 2-tuples into 1-tuples since the results have been analytically computed and can be found from (VIII–38). Let $v = [r_1, r_2]$ so $w_2(v) = r_1 + r_2$. Then from the first part of Example 1, it follows that

$$r_1 = \begin{cases} N_2(v) & \text{for } w_2(v) \le M-1 \\ W_2(v) + N_2(v) - (M-1) & \text{for } M-1 < W_2(v) \end{cases} \qquad \text{(VIII–56)}$$

$$r_2 = w_2(v) - r_1$$

Encoding Example for N = 4 and M = 3

Suppose $D_4\left(r^{[4]}\right) = 13$

Step 1

From the ordered list of 4-tuples, we see that

$$C_4(1) = 5 \le 13 < C_4(2) = 15$$

Thus, according to Step 1, $w_4\left(r^{[4]}\right) = 2$.

Step 2

The offset is $D_4\left(r^{[4]}\right) - C_4(1) = 13 - 5 = 8$

Step 3 Find Weights of 1st and 2nd Halves

$$M_2(0)\,M_2(2) + M_2(1)\,M_2(1) = 1 \times 3 + 2 \times 2 = 7 \le 8$$

$$< M_2(0)\,M_2(2) + M_2(1)\,M_2(1) + M_2(2)\,M_2(0) = 10$$

Thus

$$w_2\left(r_1^{[4]}\right) = 2 \text{ and } w_2\left(r_1^{[4]}\right) = w_4\left(r^{[4]}\right) - w_2\left(r_1^{[4]}\right) = 2 - 2 = 0$$

Step 4

The partial offset is $d = 8 - 7 = 1$

Step 5

Find Offsets of 1st and 2nd Halves

$$M_2\left(w_2\left(r_2^{[4]}\right)\right) = M_2(0) = 1$$

$$N_2\left(r_1^{[4]}\right) = \text{int}\left[d \,/\, M_2\left(w_2\left(r_2^{[4]}\right)\right)\right] = 1$$

$$N_2\left(r_2^{[4]}\right) = d - M_2\left(w_2\left(r_2^{[4]}\right)\right) N_2\left(r_1^{[4]}\right) = 1 - 1 \times 1 = 0$$

Using (VIII–56) we find that

$$r_1 = N_2\left(r_1^{[4]}\right) = 1, \quad r_2 = w_2\left(r_1^{[4]}\right) - r_1 = 2 - 1 = 1$$

$$r_3 = N_2\left(r_2^{[4]}\right) = 0, \quad r_4 = w_2\left(r_2^{[4]}\right) - r_3 = 0 - 0 = 0$$

So the encoded ring block is $r^{[4]} = [\,1100\,]$

ภาคผนวก ข

A. Program Convolution Encode

```
                    .mmregs
                    .long 55
                    .word 120
        STATMEM     .set    60h
        OUTPUT      .set    63h
        INPUT       .set    64h
        YPAST       .set    68h
        LOCATE      .set    6ah
        PCKD_IP     .set    1000h
        PCKD_OP     .set    3000h
        COUNT       .set    48
*                   .data
*PCKD_IP
*                   .include "d:\matlab\viterbi\enc_inp.dat"
                    .text

INIT    LAR     AR1,#PCKD_IP
        LAR     AR2,#PCKD_OP
        LAR     AR3,#COUNT-1
        LDP     #0
*
        LAR     AR0,#STATMEM
        MAR     *,AR0
        LACL    #0
        RPT     #20h
        SACL    *+
*
```

```
START MAR   *,AR1
      LACC  *+,0,AR0
      SACL  LOCATE
      LAR   AR0,#INPUT+3
      LACL  #3
      SAMM BRCR
      LACL  #1
      SAMM DBMR

UNPACK      LACC  LOCATE
      RPTB  LOOP1-1
      SACL  *
      APL   *-
      SFR
LOOP1
      CALL  DIFF
      CALL  ENCODE

PACK  LAR   AR0,#OUTPUT
      MAR   *,AR0
      LACL  #3
      SAMM BRCR
      LACC  *+

      RPTB  LOOP2-1
      SFL
      ADD   *+
      NOP
LOOP2
      MAR   *,AR2
      SACL  *+,0,AR3
```

```
*
        BANZ  START
        RET

*****************************************************************

DIFF    LACC  YPAST
        AND   INPUT
        XOR   INPUT+1
        XOR   YPAST+1
        SACL  INPUT+1
        SACL  YPAST+1
        LACC  YPAST
        XOR   INPUT
        RETD
        SACL  INPUT
        SACL  YPAST

*****************************************************************

ENCODE      LACC  STATMEM
        SACL  OUTPUT
        LACC  INPUT+1
        XOR   STATMEM+1
        SACB
        LACC  OUTPUT
        AND   INPUT
        XORB
        SACL  STATMEM
```

```
LACC  OUTPUT
ANDB
SACB
LACC  INPUT
XOR   INPUT+1
XOR   STATMEM+2
XORB
SACL  STATMEM+1
RETD
LACC  OUTPUT
SACL  STATMEM+2
```

B. Program Vitebi Decoder

```
kfour       .set  02000h
kthree      .set  01800h
ktwo        .set  01000h
kone        .set  00800h
kzero       .set  00000h
kmone       .set  0F800h
kmtwo       .set  0F000h
kmthree     .set  0E800h
kmfour      .set  0E000h

            .data

region1     .word 02h,06h,0bh,0dh,13h,15h,1ah,1eh  ;1st quadrant (+,+)
            .word 02h,06h,09h,0fh,13h,15h,1ah,1eh  ;2nd quadrant (-,+)
            .word 02h,06h,09h,0fh,11h,17h,1ah,1eh  ;3rd quadrant (-,-)
            .word 02h,06h,0bh,0dh,11h,17h,1ah,1eh  ;4th quadrant (+,-)

region2     .word 02h,06h,0bh,0dh,13h,14h,1ah,1dh
            .word 02h,06h,09h,0fh,12h,15h,19h,1eh
            .word 02h,06h,09h,0fh,10h,17h,09h,1eh
            .word 02h,06h,0bh,0dh,11h,16h,1ah,1dh

region3     .word 03h,04h,0bh,0dh,13h,14h,1ah,1dh
            .word 00h,07h,09h,0fh,12h,15h,19h,1eh
            .word 00h,07h,09h,0fh,10h,17h,19h,1eh
            .word 03h,04h,0bh,0dh,11h,16h,1ah,1dh

region4     .word 02h,05h,0ah,0dh,13h,15h,1ah,1eh
```

```
                .word 02h,05h,08h,0fh,13h,15h,1ah,1eh
                .word 01h,06h,09h,0eh,11h,17h,1ah,1eh
                .word 01h,06h,0bh,0ch,11h,17h,1ah,1eh

region5         .word 02h,05h,0ah,0dh,13h,14h,1ah,1dh
                .word 02h,05h,08h,0fh,12h,15h,19h,1eh
                .word 01h,06h,09h,0eh,10h,17h,19h,1eh
                .word 01h,06h,0bh,0ch,11h,16h,1ah,1dh

region6         .word 02h,05h,0ah,0dh,13h,15h,18h,1fh
                .word 02h,05h,08h,0fh,13h,15h,18h,1fh
                .word 01h,06h,09h,0eh,11h,17h,1bh,1ch
                .word 01h,06h,0bh,0ch,11h,17h,1bh,1ch

region7         .word 03h,04h,0ah,0dh,13h,14h,1ah,1dh
                .word 00h,07h,08h,0fh,12h,15h,19h,1eh
                .word 00h,07h,09h,0eh,10h,17h,19h,1eh
                .word 03h,04h,0bh,0ch,11h,16h,1ah,1dh

region8         .word 02h,05h,0ah,0dh,13h,14h,18h,1fh
                .word 02h,05h,08h,0fh,12h,15h,18h,1fh
                .word 01h,06h,09h,0eh,10h,17h,1bh,1ch
                .word 01h,06h,0bh,0ch,11h,16h,1bh,1ch

region9         .word 03h,05h,0ah,0dh,13h,14h,18h,1dh
                .word 00h,05h,08h,0fh,12h,15h,19h,1fh
                .word 01h,07h,09h,0eh,10h,17h,19h,1ch
                .word 01h,04h,0bh,0ch,11h,16h,1bh,1dh

region10        .word 02h,05h,0ah,0dh,13h,14h,18h,1dh
                .word 02h,05h,08h,0fh,12h,15h,19h,1fh
```

```
                .word 01h,06h,09h,0eh,10h,17h,19h,1ch
                .word 01h,06h,0bh,0ch,11h,16h,1bh,1dh

region11        .word 03h,05h,0ah,0dh,13h,14h,1ah,1dh
                .word 00h,05h,08h,0fh,12h,15h,19h,1eh
                .word 01h,07h,09h,0eh,10h,17h,19h,1eh
                .word 01h,04h,0bh,0ch,11h,16h,1ah,1dh

region12        .word 03h,04h,0ah,0dh,13h,14h,18h,1dh
                .word 00h,07h,08h,0fh,12h,15h,19h,1fh
                .word 00h,07h,09h,0eh,10h,17h,19h,1ch
                .word 03h,04h,0bh,0ch,11h,16h,1bh,1dh

region13        .word 03h,05h,0ah,0dh,13h,14h,18h,1fh
                .word 00h,05h,08h,0fh,12h,15h,18h,1fh
                .word 01h,07h,09h,0eh,10h,17h,1bh,1ch
                .word 01h,04h,0bh,0ch,11h,16h,1bh,1ch

scrmb_tbl
                .word 0,2,3,1,4,7,6,5
dly_ptr         .set  0066h
pth_ptr         .set  0067h
data_ptr        .set  0068h
tmploc          .set  0069h
count           .set  006Ch

past_dly        .set  0200h

past_pth        .set  0280h

oldx            .set  0300h
```

```
oldy        .set   0301h
odiffx      .set   0302h
odiffy      .set   0303h
output      .set   0304h
prv_ip      .set   0305h
curr_x      .set   0306h
curr_y      .set   0307h
diff_x      .set   0308h
diff_y      .set   0309h
abs_x       .set   030ah
abs_y       .set   030bh
small       .set   030ch
large       .set   030dh
accdist     .set   030eh
dist        .set   0316h
temp        .set   031eh
two         .set   0326h
one         .set   0327h

            .sect "diff"



diff_tbl    .word  0
            .word  1
            .word  2
            .word  3
            .word  1
```

```
        .word   0
        .word   3
        .word   2
        .word   3
        .word   2
        .word   0
        .word   1
        .word   2
        .word   3
        .word   1
        .word   0

xloc    .word kmfour, kzero, kzero, kfour, kfour, kzero, kzero, kmfour
        .word kmtwo, kmtwo, ktwo, ktwo, ktwo, ktwo, kmtwo, kmtwo
        .word kmthree, kone, kmthree, kone, kthree, kmone, kthree, kmone
        .word kone, kmthree, kone, kone, kmone, kthree, kmone, kmone

yloc    .word kone, kmthree, kone, kone, kmone, kthree, kmone, kmone
        .word kthree, kmone, kthree, kmone, kmthree, kone, kmthree, kone
        .word kmtwo, kmtwo, ktwo, ktwo, ktwo, ktwo, kmtwo, kmtwo
        .word kfour, kzero, kzero, kmfour, kmfour, kzero, kzero, kfour

path_tbl .set   0380h


tst_inp .set    03000h
        .text
```

```
main  CALL   init,*,AR0
             LAR   AR0,#30h

             SPLK  #0,IMR
             SPLK  #20H,TCR

top    SAR   AR0,count
             CALL  rd_data,*,AR0
             CALL  get_rgn,*,AR0
CALL   get_path,*,AR0
       CALL  diff,*,AR0
             LAR   AR0,count
             MAR   *,AR0

             LACC  tim

             BANZ  top
             RET

       init  LDP   #6
             CLRC  cnf
             SPM   1
             SPLK  #ktwo,two
             SPLK  #kone,one
             SPLK  #00CCCh,small
             SPLK  #07333h,large
```

```
                LAR   AR0,#accdist
                LACL  #0
                SACL  *+
                LACC  #04000h
                RPT   #6
                SACL  *+


                LDP   #0
                OPL   #4,pmst
                SPLK  #path_tbl,AR7
                SPLK  #past_dly,dly_ptr
                SPLK  #past_pth,pth_ptr
                SPLK  #1,dbmr
                RETD
                SPLK  #tst_inp,data_ptr

                              DP=0
rd_data         LAR   AR0,data_ptr
                SPLK  #xloc,indx
                LAR   AR1,*+,AR1
                SAR   AR0,data_ptr
                MAR   *0+
                LACC  *

                ADRK  #32

                LDP   #6

                SACL  curr_x
```

```
                LACC  *

                SACL  curr_y
                RET


get_rgn         ;DP=6
                LACC  curr_y
                ABS
                SACL  abs_y
                LACC  curr_x
                ABS
                SACL  abs_x
                SUB   one
                BCND  xgt1,gt
                LACC  abs_y
                SUB   one
                BCNDD go ,lt
                LAR   AR0,#region1
                LACC  abs_y
                SUB   two
                BCNDD go ,lt
                LAR   AR0,#region4
                BD    go
                LAR   AR0,#region6

xgt1            LACC  abs_x
                SUB   two
                BCNDD xgt2,gt
                LACC  abs_y
```

```
                SUB    one
                BCNDD  go ,lt
                LAR    AR0,#region1
                LACC   abs_y
                SUB    two
                BCNDD  go ,lt
                LAR    AR0,#region5
                LACC   abs_y
                SUB    one
*
xgt2            BCNDD  go ,lt
                LAR    AR0,#region3
                LACC   abs_x
                SUB    two
                ABS
                SACB
                LACC   abs_y
                SUB    one
                ABS
                SBB
                BCNDD  abov1,gt
                LACC   abs_y
                SUB    two
                BCNDD  go ,lt
                LAR    AR0,#region7
                BD     go
                LAR    AR0,#region12

abov1           ABS
                SACB
                LACC   abs_x
```

```
                SUB    one
                ABS
                SBB
                BCNDD  abov2,leq
                LACC   abs_x
                SUB    two
                BCNDD  go ,lt
                LAR    AR0,#region10
                LACC   abs_y
                SUB    two
                BCNDD  go ,lt
                LAR    AR0,#region11
                BD     go
                LAR    AR0,#region9

abov2           LAR    AR0,#region8
                XC     2,geq
                LAR    AR0,#region13

go
                LACC   curr_x
                BCND   xlt0,lt
                LACC   curr_y
                xygt0  BD     get_cur_dist
                XC     1,leq
                ADRK   #24
xlt0            LACC   curr_y
ygt0            ADRK   #8
                XC     1,lt
xylt0           ADRK   #8
```

```
get_cur_dist    ;DP=6
                LACL  #8
                SAMM  indx
                LACC  #xloc
                MAR   *,AR7
                SAR   AR0,*BR0+,AR0

                SAMM  indx
                NOP

state0          LAR   AR2,*+,AR2
                MAR   *0+
                LACC  *
                SUB   curr_x
                SACL  diff_x
                SQRA  diff_x
                ADRK  #32
                LACC  *,0,AR0
                SUB   curr_y
                SACL  diff_y
                LACL  #0
                SQRA  diff_y
                LTA   small
                SACH  dist,4
                MPY   dist
                SPH   dist

state1          LAR   AR2,*+,AR2
                MAR   *0+
                LACC  *
```

```
        SUB    curr_x
        SACL   diff_x
        SQRA   diff_x
        ADRK   #32
        LACC   *,0,AR0
        SUB    curr_y
        SACL   diff_y
        LACL   #0
        SQRA   diff_y
        LTA    small
        SACH   dist + 3,4
        MPY    dist + 3
        SPH    dist + 3

state2  LAR    AR2,*+,AR2
        MAR    *0+
        LACC   *
        SUB    curr_x
        SACL   diff_x
        SQRA   diff_x
        ADRK   #32
        LACC   *,0,AR0
        SUB    curr_y
        SACL   diff_y
        LACL   #0
        SQRA   diff_y
        LTA    small
        SACH   dist + 1,4
        MPY    dist + 1
        SPH    dist + 1
```

```
state3      LAR   AR2,*+,AR2
            MAR   *0+
            LACC  *
            SUB   curr_x
            SACL  diff_x
            SQRA  diff_x
            ADRK  #32
            LACC  *,0,AR0
            SUB   curr_y
            SACL  diff_y
            LACL  #0
            SQRA  diff_y
            LTA   small
            SACH  dist + 2,4
            MPY   dist + 2
            SPH   dist + 2

state4      LAR   AR2,*+,AR2
            MAR   *0+
            LACC  *
            SUB   curr_x
            SACL  diff_x
            SQRA  diff_x
            ADRK  #32
            LACC  *,0,AR0
            SUB   curr_y
            SACL  diff_y
            LACL  #0
            SQRA  diff_y
            LTA   small
```

```
                SACH   dist + 4,4
                MPY    dist + 4
                SPH    dist + 4

state5          LAR    AR2,*+,AR2
                MAR    *0+
                LACC   *
                SUB    curr_x
                SACL   diff_x
                SQRA   diff_x
                ADRK   #32
                LACC   *,0,AR0
                SUB    curr_y
                SACL   diff_y
                LACL   #0
                SQRA   diff_y
                LTA    small
                SACH   dist + 7,4
                MPY    dist + 7
                SPH    dist + 7

state6          LAR    AR2,*+,AR2
                MAR    *0+
                LACC   *
                SUB    curr_x
                SACL   diff_x
                SQRA   diff_x
                ADRK   #32
                LACC   *,0,AR0
                SUB    curr_y
                SACL   diff_y
```

```
                LACL  #0
                SQRA  diff_y
                LTA   small
                SACH  dist + 6,4
                MPY   dist + 6
                SPH   dist + 6

state7          LAR   AR2,*+,AR2
                MAR   *0+
                LACC  *
                SUB   curr_x
                SACL  diff_x
                SQRA  diff_x
                ADRK  #32
                LACC  *,0,AR1
                SUB   curr_y
                SACL  diff_y
                LACL  #0
                SQRA  diff_y
                LTA   small
                SACH  dist + 5,4
                MPY   dist + 5
                SPH   dist + 5


get_acc_dist    ;DP=0

                LDP   #0
                LAR   AR1,#accdist
                LAR   AR3,#temp
                LAR   AR2,#dist
```

```
**
                LAR   AR5,dly_ptr
                LAR   AR6,pth_ptr
**
                SPLK  #4,indx
                SPLK  #accdist,cbsr1
                SPLK  #accdist + 3,cber1
                SPLK  #dist,cbsr2
                SPLK  #dist + 3,cber2
                SPLK  #0A9h,cbcr
                SPLK  #3,brcr
                LACC  #07FFFh
                SACB

*       DELAY STATES: 0 1 2 3
*       PATH STATES:  0 2 3 1

stat0           RPTB  endb0 - 1
                LACC  *,0,AR2
                ADD   *+,0,AR5
                CRLT
                NOP
                XC    2,C
                SAR   AR1,*,AR6
                SAR   AR2,*,AR1
                MAR   *,AR1
                MAR   *+,AR2
                MAR   *+,AR1
endb0
                MAR   *,AR3
```

```
                SACL   *+,0,AR5
                MAR    *0+,AR6
                MAR    *0+,AR2

*       DELAY STATES: 0 1 2 3
*       PATH STATES:  3 1 0 2
stat4           ADRK   #2
                SPLK   #3,brcr
                LACC   #07FFFh
                SACB
                MAR    *,AR1

                RPTB   endb4 - 1
                LACC   *,0,AR2
                ADD    *+,0,AR5
                CRLT
                NOP
                XC     2,C
                SAR    AR1,*,AR6
                SAR    AR2,*,AR1
                MAR    *,AR1
                MAR    *+,AR2
                MAR    *+,AR1
endb4
                MAR    *,AR3
                MAR    *+
                SACL   *-,0,AR2

*
                LACC   cbsr2
                LMMR   cbsr2,cbsr2
```

```
                SACL   cber2
*
                MAR    *-,AR5
                SBRK   2
                MAR    *,AR6
                SBRK   2

*       DELAY STATES: 0 1 2 3
*       PATH STATES:  2 0 1 3
stat2           SPLK   #3,brcr
                LACC   #07FFFh
                SACB
                MAR    *,AR1

                RPTB   endb2 - 1
                LACC   *,0,AR2
                ADD    *+,0,AR5
                CRLT
                NOP
                XC     2,C
                SAR    AR1,*,AR6
                SAR    AR2,*,AR1
                MAR    *,AR1
                MAR    *+,AR2
                MAR    *-,AR1
endb2
                MAR    *,AR3
                SACL   *+,0,AR5
                MAR    *0+,AR6
                MAR    *0+,AR2
                ADRK   2
```

```
                MAR   *,AR1

*       DELAY STATES: 0 1 2 3
*       PATH STATES:  1 3 2 0
stat6
                SPLK  #3,brcr
                LACC  #07FFFh
                SACB

                RPTB  endb6 - 1
                LACC  *,0,AR2
                ADD   *+,0,AR5
                CRLT
                NOP
                XC    2,C
                SAR   AR1,*,AR6
                SAR   AR2,*,AR1
                MAR   *,AR1
                MAR   *+,AR2
                MAR   *-,AR1
endb6
                MAR   *,AR3
                MAR   *+
                SACL  *+,0,AR5
                LACC  #accdist,4

                SAMM  cbsr1
                SAMM AR1
                ADD   #3
                SAMM  cber1
```

```
        LACC   #dist,4
        SAMM   cbsr2
        SAMM AR2
        ADD    #3
        SAMM   cber2

        LAR    AR5,dly_ptr
        MAR    *+,AR6
        LAR    AR6,pth_ptr
        MAR    *+,AR1

*       DELAY STATES: 4 5 6 7
*       PATH STATES:  4 7 6 5
stat1
        SPLK   #3,brcr
        LACC   #07FFFh
        SACB

        RPTB   endb1 - 1
        LACC   *,0,AR2
        ADD    *+,0,AR5
        CRLT
        NOP
        XC     2,C
        SAR    AR1,*,AR6
        SAR    AR2,*,AR1
        MAR    *,AR1
        MAR    *+,AR2
        MAR    *+,AR1

endb1
```

```
                MAR   *,AR3
                SACL  *+,0,AR2

*       DELAY STATES: 4 5 6 7
*       PATH STATES:  6 5 4 7
stat7           ADRK  2
                MAR   *,AR5
                ADRK  6
                MAR   *,AR6
                ADRK  6

                SPLK  #3,brcr
                LACC  #07FFFh
                SACB
                MAR   *,AR1

                RPTB  endb7 - 1
                LACC  *,0,AR2
                ADD   *+,0,AR5
                CRLT
                NOP
                XC    2,C
                SAR   AR1,*,AR6
                SAR   AR2,*,AR1
                MAR   *,AR1
                MAR   *+,AR2
                MAR   *+,AR1
endb7
                MAR   *,AR3
                ADRK  2
                SACL  *-,0,AR2
```

```
                MAR   *-,AR5
                MAR   *0-,AR6
                MAR   *0-,AR1



                LACC  cbsr2
                LMMR  cbsr2,cber2
                SACL  cber2

*       DELAY STATES: 4 5 6 7
*       PATH STATES:  7 4 5 6

stat3
                SPLK  #3,brcr
                LACC  #07FFFh
                SACB
                RPTB  endb3 - 1
                LACC  *,0,AR2
                ADD   *+,0,AR5
                CRLT
                NOP
                XC    2,C
                SAR   AR1,*,AR6
                SAR   AR2,*,AR1
                MAR   *,AR1
                MAR   *+,AR2
                MAR   *-,AR1
endb3
                SPLK  #2,indx
                MAR   *,AR3
                MAR   *-
```

```
                SACL   *+,0,AR5
                MAR    *0+,AR6
                MAR    *0+,AR2
                MAR    *0+,AR1

*       DELAY STATES: 4 5 6 7
*       PATH STATES:  5 6 7 4
stat5           SPLK   #3,brcr
                LACC   #07FFFh
                SACB

                RPTB   endb5 - 1
                LACC   *,0,AR2
                ADD    *+,0,AR5
                CRLT
                NOP
                XC     2,C
                SAR    AR1,*,AR6
                SAR    AR2,*,AR1
                MAR    *,AR1
                MAR    *+,AR2
                MAR    *-,AR1
endb5
                MAR    *,AR3
                SACL   *,0,AR0

                SPLK   #0,cbcr
                LMMR   treg0,large
                SPLK   #2,indx
                LAR    AR0,#temp
                LAR    AR1,#accdist
```

```
                SPLK   #2,brcr
                LACC   #07FFFh,15
                SFL
                SACB


min_acc_dist
                MPY    *+,AR1
                PAC
                SACH   *+,0,AR0
                RPTB   endtbl1 - 1
                CRLT
                MPY    *+,AR1
                XC     1,C
                SAR    AR1,AR2
                PAC
                SACH   *+,0,AR0
endtbl1
                CRLT
                SPLK   #2,brcr
                XC     1,C
                SAR    AR1,AR2
                LAR    AR1,#accdist + 1
                MPY    *+,AR1
                PAC
                SACH   *+,0,AR0
                RPTB   endtbl2 - 1
                CRLT
```

```
                MPY    *+,AR1
                XC     1,C
                SAR    AR1,AR2
                PAC
                SACH   *+,0,AR0
endtbl2
                CRLT
                MAR    *,AR2
                XC     1,C
                SAR    AR1,AR2
*
                RETD
                MAR    *0-
                LAMM AR2

get_path        ;DP=0

                LAR    AR0,dly_ptr
                LAMM AR2
                SUB    #accdist
                SAMM   indx
                SPLK   #14,brcr
                SPLK   #past_dly + 120,cbsr1
                SPLK   #past_dly - 7,cber1
                SPLK   #08h,cbcr
*
                RPTB   tloop - 1
                MAR    *0+
                LACC   *0-
                SUB    #accdist
                SAMM   indx
```

```
                SBRK  7
                SBRK  1


tloop                   ; back (circular addressing)
                SPLK  #0,cbcr
                SAR   AR0,dly_ptr
                ADRK  128
                SAR   AR0,pth_ptr
                MAR   *0+
                LACC  *


get_sym         SUB   #dist
                ADD   #scrmb_tbl
                SACL  tmploc
                LAR   AR0,tmploc
                LACC  *,0,AR7
*
                ADD   *,0,AR0
                SAMM  AR0
                RETD
                LACL  #3
                SAMM  brcr


diff            LDP   #6
                LACC  *
                AND   #0Fh
*
```

```
SACL   output
BSAR   2
ADD    prv_ip
BLDD   #output,prv_ip
APL    #0Ch,prv_ip
ADD    #diff_tbl
SAMM AR0
LACC   output
AND    #03h
ADD    *,2
RETD
SACL   output
LDP    #0
```

เอกสารอ้างอิง

ISO 2110:1989, *Information technology – Data communications – 25-pole DTE/DCE interface connector and contact number assignments.*

ISO/IEC 11569:1993, *Information technology – Telecommunications and information exchange between systems – 26-pole interface connector mateability and contact number assignments.*

ITU-T (CCITT) Recommendation T.30 (1988) (Amended, March 1991), *Procedures for document facsimile transmission in the general switched telephone network.*

ITU-T Recommendation V.8 (1994), *Procedures for starting and ending sessions of data transmission over the general switched telephone network.*

ITU-T (CCITT) Recommendation V.10 (1993), *Electrical characteristics for unbalanced double-current interchange circuits operating at data signalling rates nominally up to 100 kbit/s.*

ITU-T (CCITT) Recommendation V.11 (1993), *Electrical characteristics for balanced double-current interchange circuits operating at data signalling rates up to 100 kbit/s.*

ITU-T (CCITT) Recommendation V.14 (1988), *Transmission of start-stop characters over synchronous bearer channels.*

ITU-T (CCITT) Recommendation V.21 (1988), *300 bit per second duplex modem standardized for use in the general switched telephone network.*

ITU-T (CCITT) Recommendation V.24 (1988), *List of definitions for interchange circuits between data terminal equipment (DTE) and data circuit-terminating equipment (DCE).*

ITU-T (CCITT) Recommendation V.25 (1984), *Automatic answering equipment and/or parallel automatic calling equipment on the general switched telephone network including procedures for disabling of echo control devices for both manually and automatically established calls.*

ITU-T (CCITT) Recommendation V.28 (1993), *Electrical characteristics for unbalanced double-current interchange circuits.*

ITU-T (CCITT) Recommendation V.32 (1988), *A family of 2-wire, duplex modems operating at data signalling rates of up to 9600 bit/s for use on the general switched telephone network and on leased telephone-type circuits.*

ITU-T (CCITT) Recommendation V.32 *bis* (1991), *A duplex modem operating at data signalling rates of up to 14 400 bit/s for use on the general switched telephone network and on leased point-to-point 2-wire telephone-type circuits.*

ITU-T (CCITT) Recommendation V.42 (1993), *Error-correcting procedures for DCEs using asynchronous-to-synchronous conversion.*

ITU-T (CCITT) Recommendation V.54 (1988), *Loop test devices for modems.*