วิศวกรรมสาร ม.ข. ปีที่ 30 ฉบับที่ 4 (355-366) ตุลาคม-ธันวาคม 2546
KKU Engineering Journal Vol. 30 No. 4 (355-366) October-December 2003

การถอดรหัสรีด โซโลมอนโดยใช้ครีปเปอร์อัลกอริทึมและใช้การตัดสินใจแบบซอฟท์

วิระสิทธิ์ อิ่มถวิล จริยา พูลสวัสดิ์ และกรกฎ แสนชัย
ภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยขอนแก่น

บทคัดย่อ

บทความนี้เสนอวิธีการถอดรหัสรีดโซโลมอน โดยใช้วิธีการถอดรหัสแบบซีเควนเชียล (sequential decoding) ซึ่งนำเอาครีปเปอร์อัลกอริทึม (creeper algorithm) มาประยุกต์ใช้ และใช้การตัดสินใจแบบซอฟท์ (soft decision) สำหรับครีปเปอร์อัลกอริทึมเป็นอัลกอริทึมที่รวมเอาคุณสมบัติที่ดีของสแต็คอัลกอริทึม (stack algorithm) และเฟโนอัลกอริทึม (fano algorithm) มาไว้ด้วยกันคือ ครีปเปอร์อัลกอริทึมจะใช้หน่วยความจำในการถอดรหัสน้อยกว่า สแต็คอัลกอริทึม และมีการคำนวณโหนดเดิมซ้ำน้อยกว่าเฟโนอัลกอริทึม

ในบทความนี้จะทำการเข้ารหัสแบบรีด โซโลมอน และทำการถอดรหัสแบบซีเควนเชียล โดยใช้ครีปเปอร์อัลกอริทึม ซึ่งได้ทำการตัดสินใจแบบซอฟท์ มอดูเลตสัญญาณแบบ MPSK และใช้แบบจำลองช่องสื่อสารแบบ Rayleigh Fading ซึ่งจากผลการทดลองพบว่า การถอดรหัสโดยใช้ครีปเปอร์อัลกอริทึมให้ทางเลือกที่น่าสนใจอีกวิธีหนึ่งในการถอดรหัสสัญญาณแบบรีด โซโลมอน

Soft Decision Sequential Decoding for Reed Solomon Codes Using the Creeper Algorithm

Virasit Imtawil Jariya Poonsawat and Koragod Saenchai
Department of Electrical Engineering, Faculty of Engineering, Khon Kaen university

Abstract

A method for soft decision decoding of Reed-Solomon codes using the Creeper Algorithm (CA) is investigated. The CA is an algorithm for sequential decoding which combines the best properties of the conventional stack and Fano algorithms. It is originally developed for decoding convolutional codes. In this paper multilevel phase-shift keying (MPSK) system is used as the modulation scheme in memoryless Rayleigh fading channel. Simulation results show that the CA is a good candidate to the stack-based soft decision decoding algorithm since it requires much less memory and on the average provides much less computational effort than the stack algorithm.

บทนำ

ในระบบสื่อสารเชิงดิจิตอล มีวิธีการหนึ่งที่จะเพิ่มประสิทธิภาพของระบบสื่อสารเพื่อให้คุณภาพของสัญญาณที่เครื่องรับดีขึ้น คือการใช้เทคนิคที่เรียกว่า การเข้ารหัสควบคุมความผิดพลาด (Error Control Coding, ECC) หรือบางครั้งก็เรียกว่า การเข้ารหัสช่องสื่อสาร (Channel Coding) ในวิธีการนี้มีหลักการเบื้องต้นก็คือการเพิ่ม redundancy ให้กับข้อมูลดิจิตอลที่จะส่งผ่านช่องสื่อสาร ซึ่งปริมาณของ redundancy ที่ใส่เข้าไปจะควบคุมดีกรีในการป้องกันสัญญาณรบกวนของสัญญาณรหัส (Coded Signal) การเข้ารหัสเพื่อควบคุมความผิดพลาดสามารถแบ่งออกได้เป็น 2 วิธีใหญ่ๆ คือ การเข้ารหัสแบบบล็อก (Block Encoding) และ การเข้ารหัสแบบคอนโวลูชัน (Convolutional Encoding)

วิธีการเข้ารหัสแบบบล็อกมีหลายวิธี หนึ่งในนั้นคือการเข้ารหัสแบบรีด โซโลมอน ซึ่งมีใช้แพร่หลายในระบบสื่อสารและการจัดเก็บข้อมูลเชิงดิจิตอล สำหรับการเข้ารหัสแบบรีด โซโลมอน จะสามารถส่งข้อมูลได้มากกว่าการเข้ารหัสแบบบล็อกชนิดอื่นๆ เนื่องจากเป็นการเข้ารหัสแบบ M-ary

ถึงแม้ว่ารหัสแบบรีด โซโลมอนจะมีคุณสมบัติที่สามารถแก้ไขข้อผิดพลาดได้อย่างมีประสิทธิภาพ แต่เนื่องด้วยโครงสร้างทางพีชคณิตของรหัสแบบรีด โซโลมอน ทำให้อัลกอริทึมที่ใช้สำหรับการถอดรหัสส่วนใหญ่เป็นวิธีการตัดสินใจแบบฮาร์ด (Hard

Decision Method) ซึ่งมีประสิทธิภาพไม่ดีเท่าที่ควร

Elke Offer and Michael G. Perkins (Offer and Perkins, 1991) ได้ทำการประยุกต์ใช้วิธีการตัดสินใจแบบซอฟท์ (Soft Decision Method) กับการถอดรหัสสัญญาณที่เข้ารหัสแบบบล็อกโดยใช้สแต็คอัลกอริทึม และต่อมา Andy Lau and Frank R. Kschischang (Lau and Kschischang, 1996) ก็ได้ประยุกต์ใช้วิธีการตัดสินใจแบบซอฟท์กับการถอดรหัส รีด โซโลมอนโดยใช้สแต็คอัลกอริทึมเช่นเดียวกัน สำหรับการถอดรหัสโดยใช้สแต็คอัลกอริทึมต้องใช้หน่วยความจำในการเก็บค่าโหนดไว้ในสแต็ค (Stack) จำนวนมาก ซึ่งทำให้การถอดรหัสแบบนี้ไม่เหมาะสำหรับการประยุกต์ใช้งานในระบบที่มีหน่วยความจำจำกัด และต่อมา Later Jeffery H. McClure and Laurie L. Joiner (McClure and Joiner, 2001) ได้ใช้วิธีการตัดสินใจแบบซอฟท์กับการถอดรหัส รีด โซโลมอนโดยใช้เฟโนอัลกอริทึมแต่ข้อด้อยของการใช้เฟโนอัลกอริทึมนี้คือเกิด backtracking นั่นคือการย้อนกลับไปกลับมาที่โหนดเดิมหลายๆครั้ง ก่อนที่จะได้ทางเดินที่ถูกต้องใน code tree ในบทความนี้ได้ทำการศึกษาการใช้ครีปเปอร์อัลกอริทึม ซึ่งเป็นอัลกอริทึมที่ใช้กับการถอดรหัสแบบซีเควนเชียลกับรหัสรีด โซโลมอน

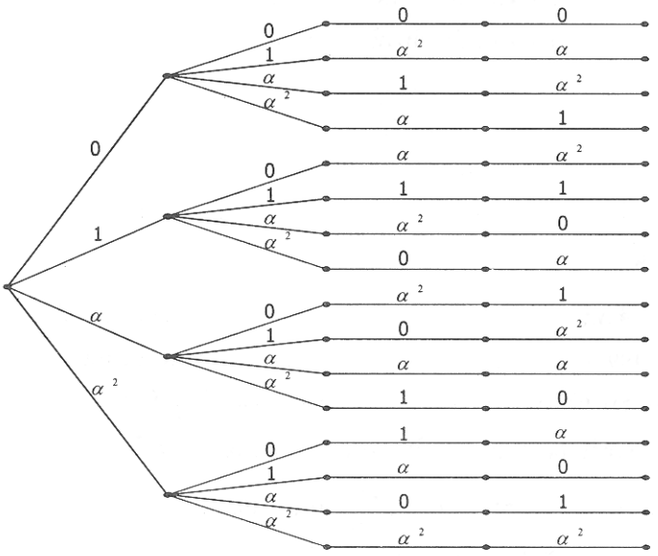
Johannesson and Zingangirov (Cedervall, Johannesson and Zigangirov, 1985) ได้เสนอครีปเปอร์อัลกอริทึมเป็นครั้งแรกในปี 1985 ซึ่งเป็นอัลกอริทึมที่นำเอาคุณสมบัติที่ดีที่สุดของสแต็คอัลกอริทึมและเฟโนอัลกอริทึมมาไว้ด้วยกัน กล่าวคือ ครีปเปอร์อัลกอริทึมจะไม่มี backward move เหมือนที่เกิดในเฟโนอัลกอริทึม และต้องการหน่วยความจำในการถอดรหัสน้อยกว่าสแต็คอัลกอริทึม

สำหรับ tree structure ของ binary block codes ได้ถูกอธิบายไว้ใน (Offer and Perkins, 1991) และ tree structure ของรหัสรีด โซโลมอน ซึ่งเป็น nonbinary block codes ได้อธิบายไว้ใน (Lau and Kschischang, 1996) ในรูปที่ 1 เป็นตัวอย่างของ tree structure สำหรับ (4,2) RS code ใน $GF\left(2^m\right)$ เมื่อ $m=2$ และใช้ primitive polynomial $p(x)=1+x+x^2$ เช่น $GF\left(2^m\right)=\{00,01,10,11\}$ เขียนให้อยู่ในรูป power representation ได้ดังนี้ $GF\left(2^2\right)=\left\{0,1,\alpha,\alpha^2\right\}$

การถอดรหัสแบบซีเควนเชียลแบ่งตามลักษณะการทำงานได้ 2 รูปแบบคือ รูปแบบ metric-base และรูปแบบ depth-first โดยสแต็คอัลกอริทึมจะเป็นรูปแบบ metric-base ซึ่งตัวถอดรหัสจะพิจารณาตามเส้นทางเดินใน code tree ที่มีค่าเมทริกซ์ที่เก็บในสแต็คที่ดีที่สุดทุกครั้ง และทางเดินทั้งหมดนี้จะต้องถูกเก็บไว้ในสแต็ค ขณะที่เฟโนอัลกอริ

ทึมนั้นเป็นรูปแบบ depth-first นั่นคือตัวถอดรหัสจะเก็บค่าทางเดินใน code tree ทีละ 1 ทางเดิน โดยพิจารณาจากค่า threshold และค่าเมทริกซ์ของทางเดินนั้น แต่ข้อด้อยของเฟโนอัลกอริทึมคือตัวถอดรหัสมีการเคลื่อนที่ไปข้างหน้าและย้อนกลับหลายครั้ง ซึ่งทุกครั้งจะขึ้นอยู่กับค่าเมทริกซ์ของทางเดินที่ตัวถอดรหัสกำลังทำงานอยู่ ณ ขณะนั้น

สำหรับครีปเปอร์อัลกอริทึม ได้ถูกพัฒนาขึ้นมาเพื่อแก้ไขข้อด้อยของสแต็คอัลกอริทึมและเฟโนอัลกอริทึม กล่าวคือ แทนที่จะใช้วิธีการเก็บค่าทางเดินทุกๆ ทางเดิน ซึ่งถูกคำนวณโดยตัวถอดรหัสในสแต็ค ก็จะเลือกเก็บเฉพาะทางเดินที่มีค่าเมทริกซ์เหมาะสมเมื่อเปรียบเทียบกับค่า threshold ซึ่งจะทำให้ขนาดของหน่วยความจำที่จำเป็นต้องใช้น้อยลง ในขณะเดียวกันการเคลื่อนที่ถอยกลับของตัวถอดรหัสจะไม่เป็นการถอยกลับแบบผ่านโหนดเดิม แต่จะใช้วิธีการถอยกลับในลักษณะกระโดดข้ามไปยังโหนดที่มีค่าเมทริกซ์สูงสุดในสแต็ค เพื่อให้เกิดความเข้าใจมากขึ้น หัวข้อต่อไปจะเป็นการอธิบายวิธีการประยุกต์ใช้งานของครีปเปอร์อัลกอริทึม



รูปที่ 1 Tree diagram ของ (4,2) RS codes ใน $GF\left(2^2\right)=\left\{0,1,\alpha,\alpha^2\right\}$

วิธีการใช้ครีปเปอร์อัลกอริทึมกับรหัสรีด โซโลมอน

ครีปเปอร์อัลกอริทึม เป็นอัลกอริทึมที่มีประสิทธิภาพสำหรับการถอดรหัสสัญญาณแบบคอนโวลูชัน มีความต้องการในการใช้หน่วยความจำต่ำ กล่าวคือ จะใช้หน่วยความจำในการเก็บค่ามากที่สุดเท่ากับ $2^m L$ โหนด (เมื่อ L เป็นค่าความยาวของ code tree และ m เป็นจำนวนบิตต่อ symbol) โดยจะมีสแต็คที่ใช้เก็บค่า 2 สแต็คคือ node object stack จะใช้ node object stack pointer (NP) สำหรับอ้างอิงถึงตำแหน่งของ object ใน node stack และ threshold object stack จะใช้ threshold object stack pointer (TP) สำหรับอ้างอิงถึงตำแหน่งของ object ใน threshold stack

ให้ sibling nodes หมายถึง โหนดที่จะถูกคำนวณต่อไป และ N เป็น จำนวนของ sibling nodes ซึ่งนิยาม sibling nodes เป็น $n_1, n_2, ..., n_N$

ให้ T-nodes หมายถึงโหนดที่มีค่าเมทริกซ์มากกว่าค่า threshold (T) ถ้า sibling nodes ทั้ง N โหนดเป็น T-nodes แล้ว ค่า flag (F) เท่ากับ 1 แต่ถ้าไม่เป็น T-nodes ค่า flag เท่ากับ 0

ให้ $n(NP)$ เป็น node object และ $\mu(TP)$ เป็น threshold object ซึ่งแต่ละ object จะเก็บค่าได้หลายค่า โดยจะเก็บค่าควบคู่กันไปตามโหนดที่ยาวเท่าๆ กัน

ให้ μ_τ^1, μ_τ^2,....., μ_τ^N เป็นค่าเมทริกซ์ที่ถูกเก็บไว้ใน threshold object stack และ N เป็นจำนวนของ T-nodes ซึ่งเมทริกซ์นี้จะเรียงค่าตามความสัมพันธ์ $\mu_\tau^i \geq \mu_\tau^{i+1}$ เมื่อ $i = 1, 2, ..., N-1$

สมมุติให้ n_{cur} หมายถึง โหนดที่ถูกคำนวณในขณะปัจจุบัน และ successor nodes หมายถึง โหนดที่จะถูกคำนวณต่อไป มีค่าเท่ากับ 2^m โหนด ประกอบด้วย $n_1, n_2, ...n_{2^m}$ เรียงตามค่าเมทริกซ์ $\mu_1 \geq \mu_2 \geq .... \geq \mu_{2^m}$

ให้ $n(NP)N$ เป็นจำนวนของ sibling nodes ใน node object ที่ตำแหน่งสูงที่สุด

ขั้นตอนการถอดรหัสโดยใช้ครีปเปอร์อัลกอริทึม

1. การถอดรหัสจะเริ่มต้นที่ $TP = NP = 0$ แล้วเก็บค่า node object, $N(NP)$ ไว้บน node object stack ซึ่ง $n(NP)N = 2$, $n(NP)F = 1$,

$n(NP)n_1 = root$ (โหนดเริ่มต้น) และ $n(NP)n_2$ เป็น dummy node (โหนดก่อนหน้าโหนดเริ่มต้น) และ $\mu(TP)\mu_\tau^1 = \mu(TP)\mu_\tau^2 = -\infty$, $T = -\infty, n_{cur} = root$, และ $\mu_{\max} = -\infty$

2. คำนวณค่า threshold $T = Q\left[\mu(TP)\mu_\tau^2\right]$ เมื่อ $Q(x) = \left\lfloor \frac{x}{\Delta} \right\rfloor \Delta$ และเรียงค่าเมทริกซ์ของ successor nodes ทั้งหมด และให้ N เป็นจำนวนของ successor nodes ที่มีค่าเมทริกซ์ไม่น้อยกว่า T
3. ใช้กฎข้อใดข้อหนึ่งในตารางที่ 1 ซึ่งมีเงื่อนไขที่เหมาะสม
4. ถ้าตัวถอดรหัสไปถึงจุดสิ้นสุดของ code tree แล้วให้หยุด จะได้เอาท์พุทของโหนดปัจจุบันเป็นข้อมูลที่ต้องการ ถ้ายังไม่ถึงจุดสิ้นสุด ให้ย้อนกลับไปทำตามขั้นตอนที่ 2 ใหม่

เพื่อให้เข้าใจขั้นตอนของการถอดรหัสรีด โซโลมอนโดยใช้ครีปเปอร์อัลกอริทึม ต่อไปจะแสดงตัวอย่างการถอดรหัส (4,2) RS codes ซึ่งมี tree diagram ดังรูปที่ 1 สมมุติให้ สัญญาณรหัสที่รับได้ (received symbols) คือ $\{\alpha^2\ 1\ 0\ \alpha^2\}$ มีค่าเมทริกซ์ $M(r_i = v_i) = 1$, $M(r_i \neq v_i) = -5$ และ step size Δ เท่ากับ 1

ขั้นตอนการถอดรหัสได้อธิบายตามรูปที่ 2-5 แล้วดังนี้

รูปที่ 2: ในสถานะเริ่มต้นจะใช้กฎข้อที่ 1, $n_{cur} = \alpha^2$ และ $\mu_{\max}$ เท่ากับ 2

รูปที่ 3: threshold T หาได้จาก $Q(-4) = -4$, แล้ว $N = 0$ และ $\max\{-10, -\alpha\}$ ไม่น้อยกว่า $Q\left[\mu(0)\mu_\tau^2\right] = -\infty$ ดังนั้น จะใช้กฎข้อที่ 4

รูปที่ 4: $n_{cur} = 1$ และจากการเรียง $\mu(1)\mu_\tau^2 = -4$, $N(0)n_2 = \alpha$, คำนวณ $T = Q(-4) = -4$, แล้ว $N = 0$ ดังนั้น จะใช้กฎข้อที่ 4

รูปที่ 5: $n_{cur} = \alpha$ และ $T = Q(-4) = -4$, แล้ว $N = 1$ ดังนั้น จะใช้กฎข้อที่ 3, ซึ่ง $n_{cur} = n_1 = \alpha\ 1\ 0\ \alpha^2$ เป็นจุดสิ้นสุดของ code tree,หยุด ดังนั้นจะได้ข้อมูลที่ถอดรหัสแล้วคือ $\alpha\ 1\ 0\ \alpha^2$

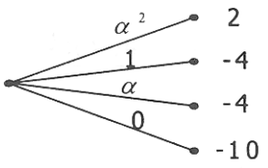
ตารางที่ 1

กฎ 6 ข้อสำหรับครีปเปอร์อัลกอริทึมที่ใช้กับรหัสรีด โซโลมอน

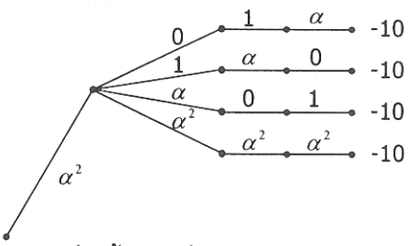
Rule	Condition	Decoder actions
One	$N \geq 2$ and $\mu_1 > \mu_{\max}$	$n_{cur} = n_i$ $N(NP+1).n_i = n_i, i = 1,2,...,N$ $N(NP+1)N = N$ $N(NP+1)F = 1$ $NP = NP+1$ $\mu(TP+1)\mu_\tau^1 = -\infty$ $\mu(TP+1)\mu_\tau^i = \mu_i \quad i = 2,3,...,N$ if $N < 2^b$, then $\mu(TP)\mu_\tau^1 = \max\{\mu_{N+1}, \mu(TP)\mu_\tau^1\}$ $TP = TP+1$ $\mu_{\max} = \max\{\mu_{\max}, \mu_1\}$
Two	$N \geq 2$ and $\mu_1 \leq \mu_{\max}$	$n_{cur} = n_i$ $N(NP+1)n_i = n_i, i = 1,2,...,N$ $N(NP+1)N = N$ $N(NP+1)F = 0$ $NP = NP+1$ If $N < 2^m$, then $\mu(TP)\mu_\tau^1 = \max\{\mu_{N+1}, \mu(TP)\mu_\tau^1\}$
Three	$N = 1$	$n_{cur} = n_i$ $\mu_{\max} = \max\{\mu_{\max}, \mu_1\}$ $\mu(TP)\mu_\tau^1 = \max\{\mu_2, \mu(TP)\mu_\tau^1\}$
Four	$N = 0$ and $N(NP)F = 1$ and $\max\{\mu_2, \mu(TP)\mu_\tau^1\}$ $\geq Q[\mu(TP-1)\mu_\tau^2]$	$n_{cur} = N(NP)n_2$ $\mu(TP)\mu_\tau^1 = \max\{\mu_1, \mu(TP)\mu_\tau^1\}$ Resort the $N(NP)N$ thresholds in $\mu(TP)$ Resort the $N(NP)N$ node in $N(NP)$ accordingly $\mu(TP)\mu_\tau^1 = -\infty$

ตารางที่ 1 (ต่อเนื่อง)

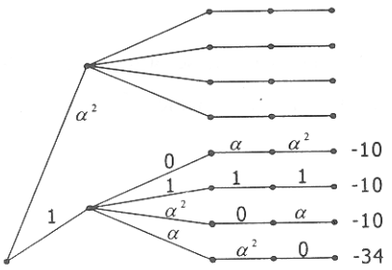
Rule	Condition	Decoder actions
Five	$N=0$	$n_{cur}=N(NP)n_2$
	and	$\mu(TP-1)\mu_\tau^1=\max\left\{\mu_1,\mu(TP)\mu_\tau^1,\mu(TP-1)\mu_\tau^1\right\}$ Delete $N(NP)n_1$
	$N(NP)F=1$	Delete $\mu(TP)\mu_\tau^1$
	and	Renumber the remaining $N(NP)N-1$ thresholds in $\mu(TP)$
	$\max\left\{\mu_2,\mu(TP)\mu_\tau^1\right\}$	Renumber the remaining $N(NP)N-1$ node in $N(NP)$
	$<Q\left[\mu(TP-1)\mu_\tau^2\right]$	$N(NP)N=N(NP)N-1$
		if $N(NP)N=1$, then $NP=NP-1$, and $TP=TP-1$
Six	$N=0$	$n_{cur}=N(NP)n_2$
	and	$\mu(TP)\mu_\tau^1=\max\left\{\mu_1,\mu(TP)\mu_\tau^1\right\}$
	$N(NP)F=0$	Delete $N(NP)n_1$
		Renumber the remaining $N(NP)N-1$ node in $N(NP)$
		$N(NP)N=N(NP)N-1$
		if $N(NP)N=1$, then $NP=NP-1$



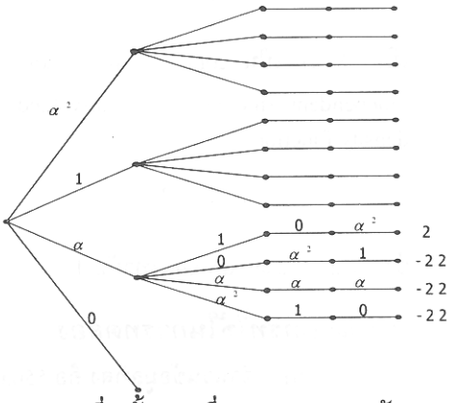
รูปที่ 2 ขั้นตอนที่ 1ของการถอดรหัส



รูปที่ 3 ขั้นตอนที่ 2 ของการถอดรหัส



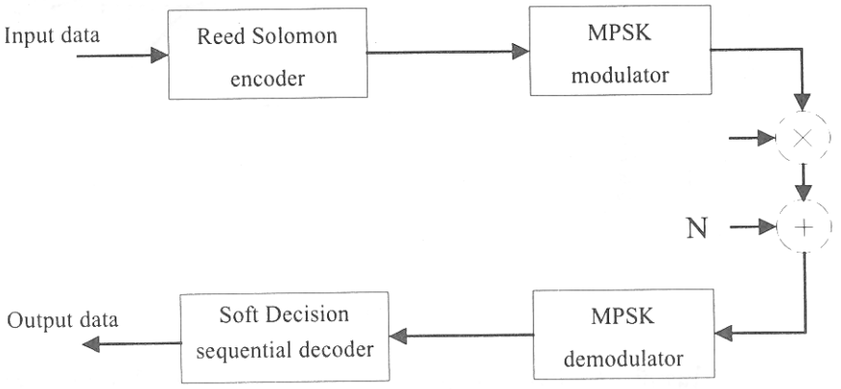
รูปที่ 4 ขั้นตอนที่ 3 ของการถอดรหัส



รูปที่ 5 ขั้นตอนที่ 4 ของการถอดรหัส

ผลการทดลอง

หัวข้อนี้จะเป็นการแสดงวิธีการทดสอบประสิทธิภาพของการถอดรหัสแบบครีปเปอร์ในรหัสแบบรีด โซโลมอน ซึ่งในการทดลองจะใช้ระบบที่มีการมอดูเลตสัญญาณแบบ 8-PSK ผ่านช่องสื่อสารแบบ Rayleigh fading รูปที่ 6 แสดงบล็อกไดอะแกรมของระบบรับส่งข้อมูลที่ใช้ในการทดลอง



รูปที่ 6 บล็อกไดอะแกรมของระบบรับส่งข้อมูล

ช่องสื่อสารนี้จะรับค่าอินพุทซึ่งเป็น complex value $x(n)$ เข้ามาที่เวลา n และสร้างเอาท์พุท

$$r(n) = \alpha(n)x(n) + N(n) \quad (1)$$

เมื่อ $N(n)$ เป็น complex-value additive white Gaussian noise และ $\alpha(n)$ เป็น independent (memoryless) real-valued Rayleigh random variable, มี probability density function

$$f_\alpha(x) = 2xe^{-x^2} \quad (2)$$

ซึ่ง mean – squared value เท่ากับ 1

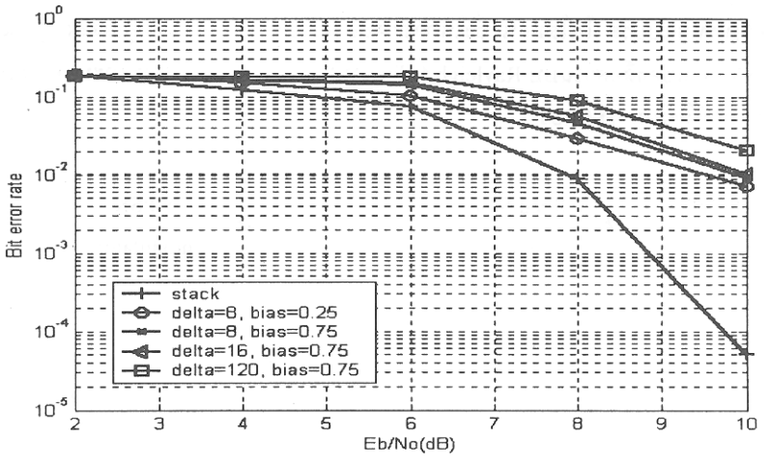
พารามิเตอร์ที่ใช้ในการทดลอง

ก) จำนวนข้อมูลที่ส่ง คือ 550,000 บิต

ข) (31, 11) RS codes ($m = 5$)

ค) 8-level soft decision decoding

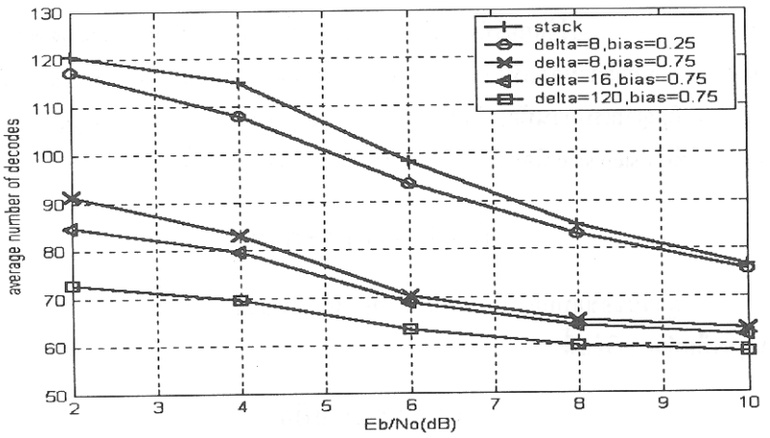
ง) Eb/No จาก 2 – 10 dB



รูปที่ 7 การวัดค่าความผิดพลาดของตัวถอดรหัส

ในรูปที่ 7 เป็นการวัดค่าความผิดพลาดของตัวถอดรหัสที่ได้จากการทดลองโดยใช้สแต็คอัลกอริทึมและครีปเปอร์อัลกอริทึม ซึ่งจะเห็นว่าค่าความผิดพลาดของครีปเปอร์อัลกอริทึมจะมากกว่าสแต็คอัลกอริทึม ซึ่งอาจจะมีสาเหตุมาจากในสแต็คอัลกอริทึมใช้ขนาดของสแต็คที่มากพอสำหรับการป้องกัน erasure จึงทำให้มีค่าความผิดพลาดน้อย ส่วน ครีปเปอร์อัลกอริทึม จะมีค่าความผิดพลาดมากขึ้นเมื่อ step size มากขึ้น ซึ่งการใช้

step size มากขึ้นจะทำให้การถอดรหัสไปเร็วขึ้น ซึ่ง path ที่ถอดรหัสได้อาจจะไม่เป็น maximum likelihood path แต่อย่างไรก็ตามการใช้ step size สูงก็จะทำให้จำนวนเฉลี่ยของการคำนวณค่าในการถอดรหัสลดลงดังรูปที่ 8



รูปที่ 8 จำนวนเฉลี่ยของการคำนวณในการถอดรหัส

ตารางที่ 2

แสดงจำนวนหน่วยความจำสูงสุดที่ใช้ในการถอดรหัส

Eb/No (dB)	maximum storage memory	
	stack	Creeper,delta=8
3dB	786	245
7dB	563	198

ตารางที่ 2 แสดงจำนวนหน่วยความจำสูงสุดที่ใช้ในการถอดรหัสของทั้งสองอัลกอริทึม ซึ่งพบว่า ครีปเปอร์ อัลกอริทึมจะใช้หน่วยความจำน้อยกว่าสแต็คอัลกอริทึม

สรุป

บทความนี้แสดงให้เห็นว่า ครีปเปอร์อัลกอริทึมเป็นอัลกอริทึมที่น่าสนใจสำหรับการถอดรหัสรีด โซโลมอนโดยใช้การตัดสินใจแบบซอฟท์ ถึงแม้ว่าค่าความผิดพลาดของครีปเปอร์อัลกอริทึมจะมากกว่าสแต็คอัลกอริทึม ซึ่งถ้าใช้ code rate ต่ำลง (redundancy

สูง ๆ) ค่าความผิดพลาดนี้ก็จะดีขึ้น

ครีปเปอร์อัลกอริทึมจะมีข้อดีเหนือสแต็คอัลกอริทึมคือ การใช้หน่วยความจำจะใช้มากที่สุดเท่ากับ $2^m L$ ขณะที่สแต็คอัลกอริทึมจะต้องการหน่วยความจำอย่างน้อยที่สุดเท่ากับ $2^m(L-1)$ ซึ่งถ้าระบบมีสัญญาณรบกวนก็จะทำให้ใช้หน่วยความจำมากกว่านี้มาก และครีปเปอร์อัลกอริทึมจะเป็นการแลกเปลี่ยนกันระหว่างค่าความผิดพลาดและจำนวนเฉลี่ยในการคำนวณเพื่อให้การถอดรหัสสำเร็จด้วยการเปลี่ยนค่า step size นั่นคือ ถ้าระบบมีสัญญาณรบกวนน้อย ก็ให้ใช้ค่า step size มากได้ แต่ถ้าระบบมีสัญญาณรบกวนมาก ก็ให้ใช้ค่า step size น้อย เพื่อที่ทำการถอดรหัสได้ละเอียดขึ้น

กิตติกรรมประกาศ

ผู้วิจัยขอขอบคุณ บัณฑิตวิทยาลัย มหาวิทยาลัยขอนแก่น ที่ให้การสนับสนุนเงินทุนเพื่อทำการวิจัยนี้

เอกสารอ้างอิง

M. Cedervall, R. Johannesson and K.Sh. Zigangirov. 1985. Creeper-a new algorithm for sequential decoding. **IEEE Proceedings Inform. Theory**. Brighton : England.

R. Johannesson and K.Sh. Zigangirov. 1999. Fundamentals of Convolutional Coding, **The institute of Electrical and Electronics Engineers**.

A. Lau and F.R. Kschischang. A sequential Soft-Decision Decoder for Reed- Solomon Codes Applied to Encoded PSK in Rayleigh-Fading Channels, **IEEE Trans Vehicular Technology**. 45: 97-104,

J.H. McClure, L.L. Joiner. **2001.** Soft Decision Decoding of Reed-Solomon codes using the Fano sequential algorithm, **IEEE Proceedings Southeast Con.** :131-135, 2001.

E. Offer and M.G. Perkins. 1991. Soft Decision Decoding of Block Codes and Concatenated Block-Convoltional Codes Using the Stack Algorithm. **IEEE.**