NGHIÊN CỨU ỨNG DỤNG MÃ TURBO VÀO HỆ THỐNG DI ĐỘNG THẾ HỆ THỨ 3 IMT-2000

Phạm Hồng Liên – Trường Đại Học Bách Khoa, ĐHQG-HCM
Chung Thị Ngọc Hạnh – Công ty Cổ phần Dịch vụ Bưu chính Viễn thông Sài Gòn
(Bài nhận ngày 14 tháng 9 năm 2001)

TÓM TẮT: Mục tiêu của hệ thống thông tin di động thế hệ mới là cung cấp nhiều loại hình dịch vụ cho mọi người vào mọi lúc, mọi nơi. Các dịch vụ bao gồm các dịch vụ như truyền dữ liệu tốc độ cao, dữ liệu hình ảnh và dịch vụ đa truyền thông với tốc độ lên đến 2Mbps bên cạnh dịch vụ thoại truyền thống.

Với tốc độ truyền lên tới 2Mbps như vậy thì vấn đề cải tiến độ tin cậy của đường truyền trong hệ thống 3G là rất cần thiết và quan trọng. Mã hóa kênh thường được dùng trong hệ thống thông tin di động số nhằm để bảo vệ bit tin bị nhiễu và giảm số lượng lỗi bit. Một mã sửa sai lý tưởng là mã mà việc thực hiện của nó đạt đến giới hạn Shannon, nhưng trong thực tế không thể thực hiện được. Trong thực tế chỉ có một mã đạt gần đến giới hạn Shannon là mã Turbo RSC (Recursive Systematic Encoder).

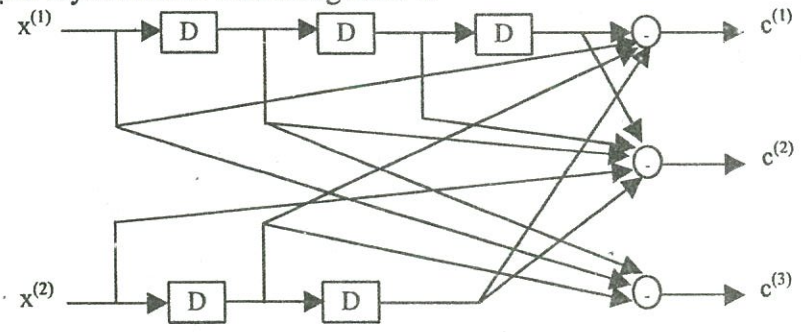
Chức năng thực hiện tuyệt vời của mã Turbo là nó đạt đến giới hạn Shannon trong trường hợp *kích cỡ khung lớn* ở kênh AWGN (nhiễu cộng)[10]. Trong tiêu chuẩn IMT-2000 thì mã Turbo được chấp nhận như là một phương pháp mã hóa kênh cho các dịch vụ có *kích thước khung lớn* được định nghĩa bởi 2 tổ chức chuẩn hóa 3GPP (W-CDMA) và 3GPP2 (cdma2000)[2] [10].

I. MÃ TURBO

Mã Turbo là mã tích chập mở rộng nên khái niệm mã tích chập được nêu ra ở đây. Nhằm giúp hiểu rõ mã Turbo, trong phần mã tích chập chỉ mô tả vắn tắt các nguyên tắc cơ bản.

I.1 Mã tích chập

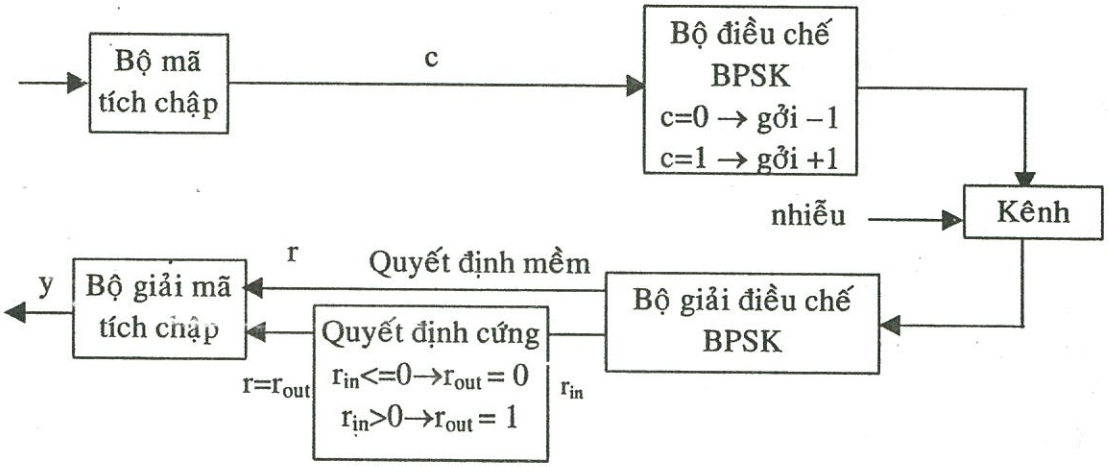
Một mã tích chập đưa các bit kiểm tra vào chùm bit dữ liệu thông qua việc sử dụng các bộ thanh ghi dịch tuyến tính D như trong hình 1.



Hình 1: Ví dụ bộ mã tích chập trong đó $x^{(i)}$ là chùm bit thông tin ngõ vào và $c^{(i)}$ là chùm bit được mã hóa ngõ ra

Việc giải mã của mã tích chập dựa theo thuật toán Viterbi, tùy theo loại lượng tử hóa được sử dụng ở các bit nhận được mà có giải mã Viterbi quyết định mềm và giải mã Viterbi quyết định cứng. Giải mã Viterbi quyết định cứng sử dụng lượng tử hóa 1-bit trên các giá trị kênh nhận được. Giải mã Viterbi quyết định mềm sử dụng lượng tử hóa nhiều-bit trên các giá trị

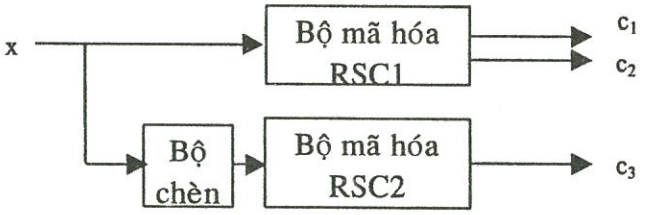
kênh nhận được. Đối với giải mã quyết định mềm lý tưởng (lượng tử hóa không xác định-bit), các giá trị kênh nhận được được sử dụng trực tiếp trong bộ giải mã kênh[3].



Hình 2: Giải mã quyết định cứng và mềm

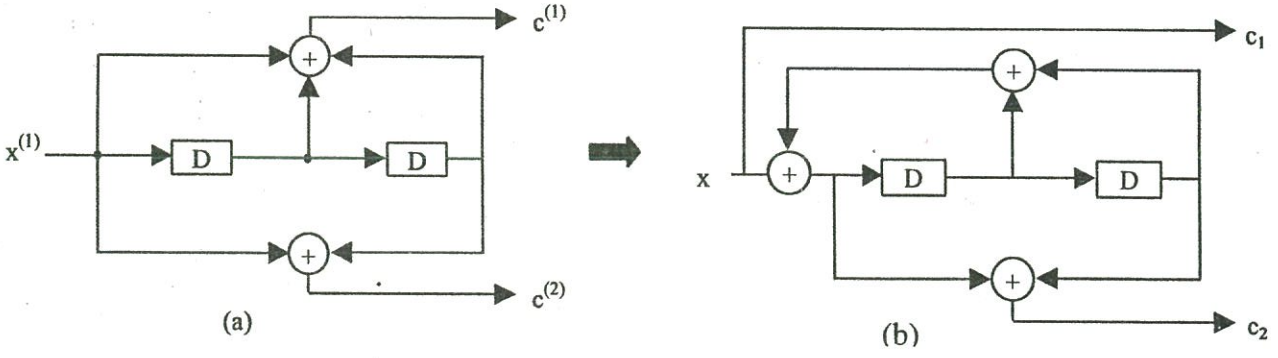
I.2 Mã Turbo

Mã Turbo là sự kết nối song song 2 hay nhiều bộ mã hoá tích chập hệ thống đệ qui (RSC)



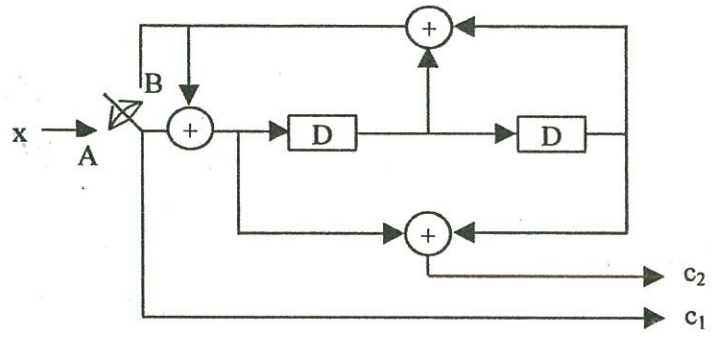
Hình 3: Mã Turbo tổng quát gồm 2 bộ mã hóa chập hệ thống đệ qui (RSC)

Bộ mã hóa chập hệ thống đệ qui (RSC) có được bằng cách hồi tiếp một trong những ngõ ra của bộ mã tích chập thông thường về ngõ vào chuỗi tin, và có một ngõ ra lấy trực tiếp từ ngõ vào chuỗi tin như trình bày ở hình 4.



Hình 4:cách thức lấy bộ mã hóa hệ thống đệ qui (RSC). (a) Bộ mã hóa tích chập thông thường. (b) Bộ mã hóa hệ thống đệ qui (RSC)

Kết thúc Trellis trong RSC phức tạp hơn so với mã chập (chỉ cần thêm m=k-1 bit zero vào sau chuỗi tin), trong khi đó kết thúc trong RSC phải dùng khóa chuyển.



Hình 5: Cách thức kết thúc trellis ở bộ mã RSC

Để mã hóa chuỗi ngõ vào, thì khóa chuyển được bật đến vị trí *A* và để kết thúc trellis, thì khóa chuyển được bật đến vị trí *B*.

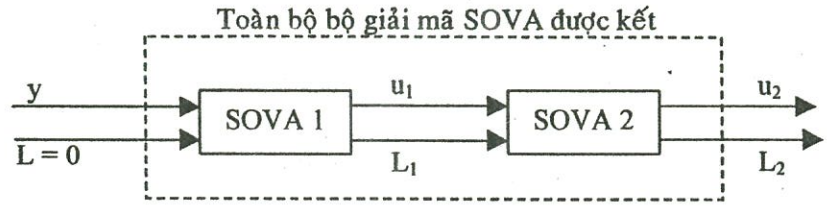
I.3 Bộ chèn

Bộ chèn trong mã Turbo được sử dụng nhằm để cung cấp tính ngẫu nhiên cho các chuỗi ngõ vào. Mục đích của bộ chèn là để hoán vị tất cả các chuỗi ngõ vào "xấu" (các chuỗi này cho ra các từ mã trọng số thấp) thành chuỗi cho ra có các từ mã ngõ ra trọng số cao. Ta có các kiểu chèn sau

- Bộ chèn khối
- Bộ chèn ngẫu nhiên (giả ngẫu nhiên).
- Bộ chèn dịch vòng.
- Bộ chèn bán ngẫu nhiên.
- Bộ chèn chẳn-lẽ.
- Bộ chèn tối ưu (gần tối ưu).
- ...

I.4 Bộ giải mã mã turbo lặp

Nguyên lý của bộ giải mã lặp Viterbi ngõ ra mềm (SOVA)



Hình 6: Bộ giải mã SOVA kết nối trong đó y biểu diễn các giá trị kênh nhận được, u biểu diễn các giá trị ngõ ra quyết định cứng, và L biểu diễn các giá trị tin cậy liên kết

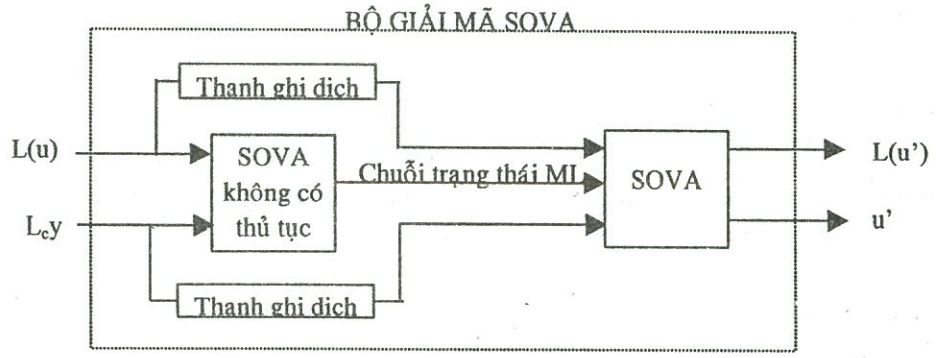
Thủ tục cập nhật giá trị tin cậy

Thủ thục cập nhật này được tích hợp vào trong thuật toán Viterbi như sau:

Đối với nút $S_{k,t}$ trong biểu đồ trellis (đáp ứng đến trạng thái k tại thời điểm t),

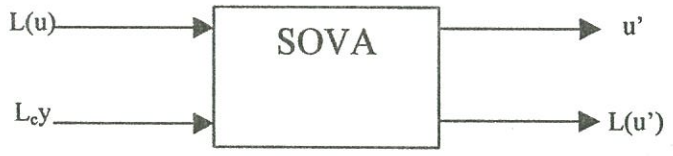
1. Lưu $L(t)=|V_s(S_{k,t})-V_c(S_{k,t})|$. (ký hiệu bằng Δ).
   Nếu có nhiều hơn một con đường cạnh tranh, thì sau đó nhiều giá trị tin cậy phải được tính và giá trị tin cậy nhỏ nhất được lấy là L(t).
2. Khởi tạo giá trị tin cậy $S_{k,t}$ bằng $+\infty$ (tin cậy nhất).
3. So sánh các con đường survivor và cạnh tranh tại $S_{k,t}$ và lưu lại các cấp độ nhớ (*MEM*) trong đó các quyết định nhị phân được ước đoán của 2 con đường là khác nhau.
4. Cập nhật các giá trị tin cậy tại các *MEM* này với thủ tục như sau:
   (a) Tìm *MEM* thấp nhất >0, coi như là MEM_{low}, mà giá trị tin cậy của nó không được cập nhật.
   (b) Cập nhập giá trị tin cậy của MEMlow L(t-MEM_{low}) bằng cách gán giá trị tin cậy thấp nhất giữa MEM = 0 và MEM = MEM_{low}

I.5 Sơ đồ khối của bộ giải mã SOVA



Hình 7: Sơ đồ khối của bộ giải mã SOVA

Bộ giải mã thành phần SOVA



Hình 8: Bộ giải mã thành phần SOVA.

Thuật toán Viterbi ngõ ra mềm (SOVA) thực hiện như sau:

1. (a) Khởi tạo thời điểm t=0
   (b) Khởi tạo $M_0^{(m)} = 0$ đối với trạng thái zero trong biểu đồ trellis.

$$M_0^{(m)} = \begin{cases} 0 & zerostate \\ -\infty & otherstate \end{cases}$$

2. (a) Lấy thời điểm t=t+1
   (b) Tính metric cho mỗi trạng thái trong sơ đồ trellis.

$$M_t^{(m)} = M_{t-1}^{(m)} + u_t^m L_c y_{t,1} + u_t^{(m)} L(u_t) + \sum_{j=2}^{N} x_{t,j}^{(m)} L_c y_{t,1}$$

trong đó

m : nhánh trellis nhị phân cho phép

$M_t^{(m)}$: metric được tích lũy cho thời điểm t trên nhánh m.

$u_t^{(m)}$: bit hệ thống cho thời gian t trên nhánh m (nghĩa là $x_t^{(m)}$)

$x_{t,j}^{(m)}$: bit thứ j trong N bit cho thời điểm t trên nhánh m ($2 \le j \le N$)

$y_{t,j}^{(m)}$: giá trị nhận được qua kênh tương ứng với $x_{t,j}^{(m)}$

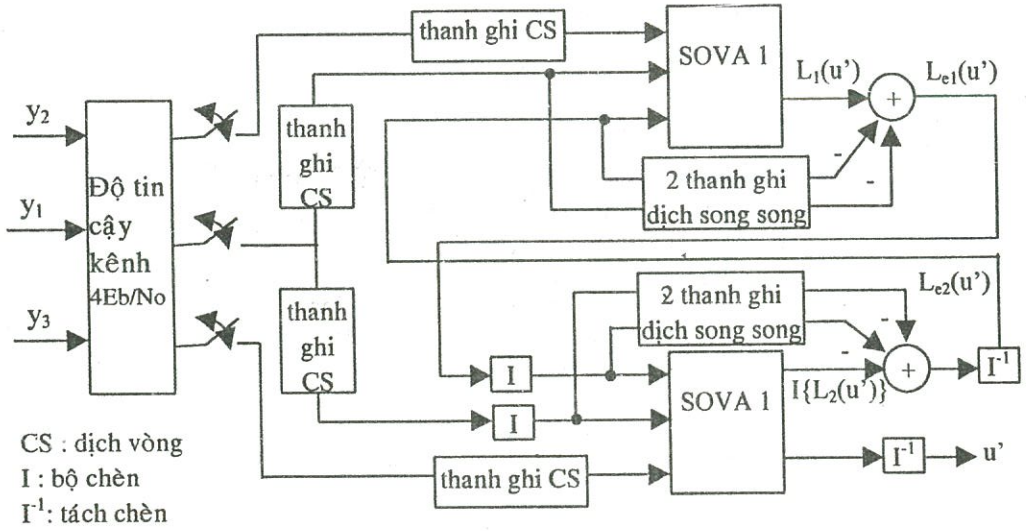
$L_c = 4\frac{E_b}{N_0}$: giá trị tin cậy của kênh

$L(u_t)$: giá trị tin cậy ưu tiên cho thời điểm t. Giá trị này lấy từ bộ giải mã có trước đó. Nếu không có bộ giải mã trước đó, thì sau đó giá trị này được lấy bằng 0

3. Tìm $\max_m M_t^{(m)}$ cho mỗi trạng thái. Để đơn giản, lấy $M_t^{(1)}$ biểu thị metric của con đường survivor và $M_t^{(2)}$ biểu thị metric của đường cạnh tranh.
4. Lưu $M_t^{(1)}$ và các con đường trạng thái và các con đường bit survivor kết hợp của nó.
5. Tính $\Delta_t^{(0)} = \frac{1}{2}\left|M_t^1 - M_t^2\right|$.
6. So sánh các con đường survivor và cạnh tranh tại mỗi trạng thái ở thời điểm t và lưu các MEM mà ở đó các quyết định nhị phân được ước đoán của 2 con đường khác nhau.
7. Cập nhật $\Delta_t^{MEM} \approx \min_{k=0,\ldots,MEM}\left\{\Delta_t^k\right\}$ cho tất cả các *MEM* từ *MEM* nhỏ nhất đến *MEM* lớn nhất.
8. Trở lại bước 2 cho đến khi kết thúc chuỗi nhận.
9. Cho ra chuỗi bit ước đoán *u'* và chuỗi *giá trị-L* hay "*mềm*" kết hợp của nó $L(u')=u' \bullet \Delta$, trong đó phép • định nghĩa là phép nhân từng phần với nhau. $L(u')$ sau đó được xử lý và đi tiếp như là chuỗi ưu tiên $L(u)$ đối với bộ giải mã kế tiếp.

Bộ giải mã turbo lặp lại SOVA

Bộ giải mã Turbo lặp lại bao gồm 2 bộ giải mã thành phần *SOVA* kết nối lại.



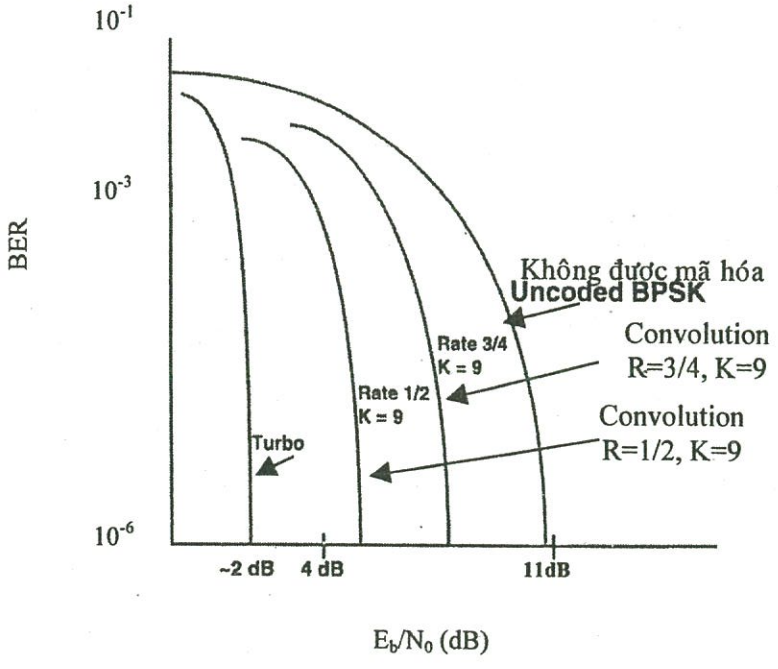
Hình 9: Bộ giải mã mã Turbo lặp lại SOVA

Thuật toán mã Turbo lặp với lần lặp thứ *n* như sau

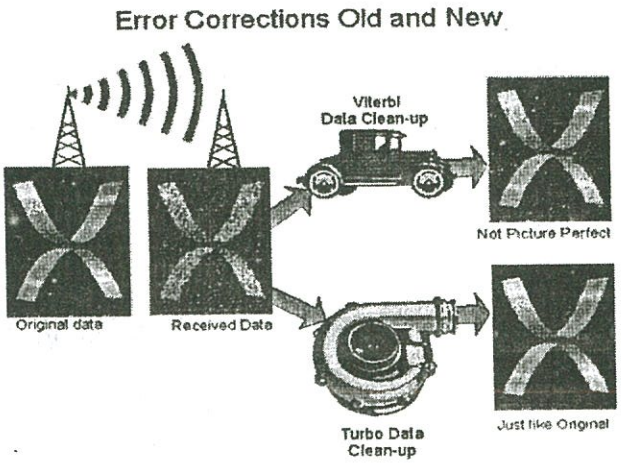
1. Bộ giải mã *SOVA1* có ngõ vào là các chuỗi $4\frac{E_b}{N_0}y_1$ (hệ thống), $4\frac{E_b}{N_0}y_2$ (kiểm tra chẵn lẽ), và $L_{e2}(u')$ và cho ra chuỗi $L_1(u')$. Đối với lần lặp đầu tiên, chuỗi $L_{e2}(u')=0$ bởi vì không có giá trị ưu tiên (không có giá trị bên ngoài từ *SOVA2*).
2. Thông tin bên ngoài từ *SOVA1* được lấy bằng
   $L_{e1}(u')=L_1(u')-L_{e2}(u')-L_c y_1$ trong đó $L_c=4\frac{E_b}{N_0}$
3. Các chuỗi $4\frac{E_b}{N_0}y_1$ và $L_{e1}(u')$ được chèn và là $I\left\{4\frac{E_b}{N_0}y_1\right\}$ và $I\{L_{e1}(u')\}$.
4. Bộ giải mã *SOVA2* có ngõ vào là các chuỗi (hệ thống), và $I\left\{4\frac{E_b}{N_0}y_3\right\}$ (kiểm tra chẵn lẽ đã được chèn ở bộ giải mã mã Turbo) và $I\{L_{e1}(u')\}$ (thông tin ưu tiên) và cho ra các chuỗi $I\{L_2(u')\}$ và $I\{u'\}$.
5. Thông tin bên ngoài từ *SOVA2* được lấy là
   $I\{L_{e2}(u')\}=I\{L_2(u')\}-I\{L_{e1}(u')\}-I\{L_c y_1\}$
6. Các chuỗi $I\{L_{e2}(u')\}$ và $I\{u'\}$ được giải chèn và là $L_{e2}(u')$ và u'. $L_{e2}(u')$ được hồi tiếp về *SOVA1* như là thông tin ưu tiên cho lần lặp kế tiếp và *u'* là ngõ ra của các bit được ước đoán cho lần lặp thứ *n*.

II. SO SÁNH SỰ KHÁC NHAU GIỮA MÃ TÍCH CHẬP VÀ MÃ TURBO

Trong cdma2000 sử dụng 2 loại mã: mã Turbo và mã tích chập có đặc tính như hình dưới. Độ lợi mã của mã Turbo gần bằng 9dB trong khi độ lợi mã của mã tích chập cao nhất là gần bằng 6dB [1].

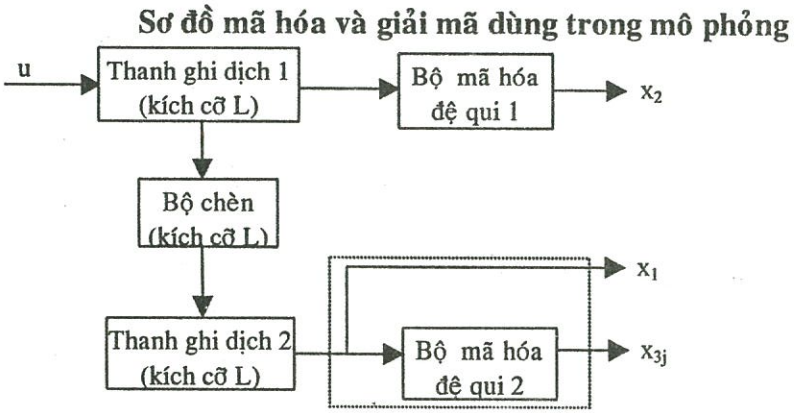


Hình 10: So sánh độ lợi mã của mã tích chập và mã turbo trong hệ thống cdma2000

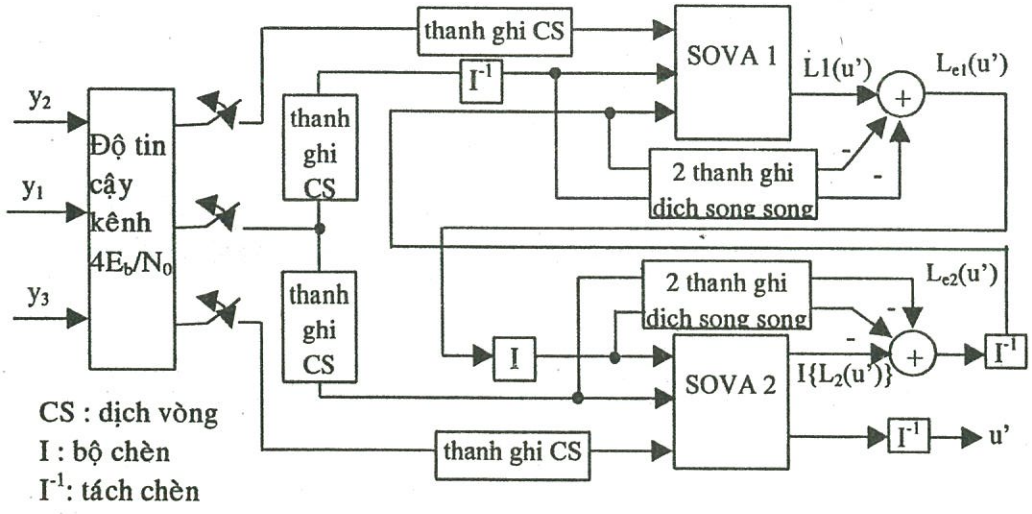


Hình 11: Mã hóa sửa sai cũ (tích chập) và mới (Turbo)

III. KẾT QUẢ MÔ PHỎNG



Hình 12: Sơ đồ mã hóa dùng để mô phỏng



Hình 13: sơ đồ giải mã dùng trong mô phỏng

Phân tích kết quả mô phỏng

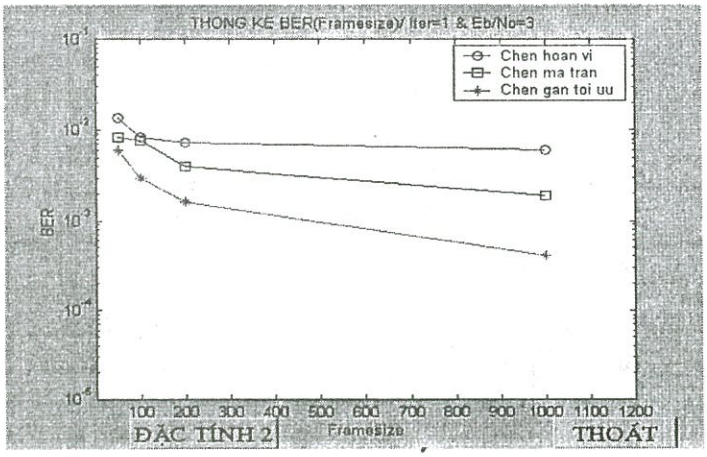
Sau khi chạy mô phỏng chương trình nhiều lần với số lượng bit ngõ vào là 1000 và 5000 bit tương ứng với 3 kiểu chèn, Eb/No, số lần lặp, kích cỡ khung khác nhau. Ta có kết quả như sau:

III.1 Thực hiện mã turbo với cùng số lần lặp iter=3, eb/no = 3db với 3 kiểu chèn khác nhau

Bảng 1: BER của mã Turbo với số lần lặp Iter=1, Eb/No=3dB

Kiểu chèn	Tổng số bit ngõ vào	Framesize (bit)	Số bit sai	BER
Hoán vị	1000	50	10/18/13	$13,6.10^{-3}$
	1000	100	9/4/12	$8,3.10^{-3}$
	1000	200	7/8/7	$7,3.10^{-3}$
	5000	1000	27/34	$6,1.10^{-3}$
Ma trận	1000	50	5/6/14	$8,3.10^{-3}$
	1000	100	12/4/7	$7,6.10^{-3}$
	1000	200	6/3/6/1	4.10^{-3}
	5000	1000	6/21/2	$1,9.10^{-3}$
Gần tối ưu	1000	50	7/8/3	6.10^{-3}
	1000	100	3/3/3	3.10^{-3}
	1000	200	3/2/0	$1,6.10^{-3}$
	5000	1000	2/2/2	4.10^{-4}

Hình sau trình bày biểu đồ thực hiện của mã Turbo có số lần lặp Iter=1, Eb/No=3dB với 3 kiểu chèn khác nhau.



Nhận xét:

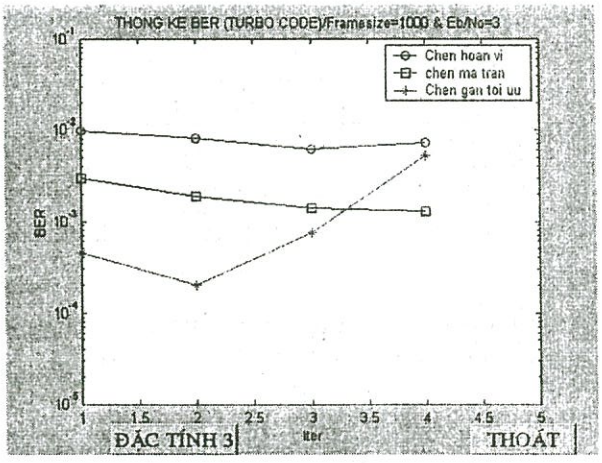
- Ở 3 kiểu chèn, kết quả cho thấy việc thực hiện của mã Turbo sẽ được cải tiến khi kích cỡ khung framesize tăng
- Kiểu chèn gần tối ưu cho kết quả tốt nhất là do bộ chèn này đã chọn được vị trí hoán vị ngẫu nhiên tốt nhất để được mã có khoảng cách tối thiểu lớn nhất (d_{min}), nghĩa là không có bất kỳ 2 từ mã có thể nào có giá trị gần bằng nhau. Khoảng cách tối thiểu càng lớn thì càng dễ dàng phân biệt được 2 đôi từ mã bất kỳ và việc thực hiện giải mã sẽ tốt hơn.

III.2 Thực hiện của mã Turbo có cùng framesize=1000, Eb/No=3dB với 3 kiểu chèn khác nhau:

Bảng 2: BER của mã Turbo với số kích cỡ khung framesize=1000, Eb/No=3dB

Kiểu chèn	Tổng số bit ngõ vào	Số lần lặp Iter	Số bit sai	BER
Hoán vị	5000	1	40/74/35	$9{,}9.10^{-3}$
	5000	2	34/62/27	$8{,}2.10^{-3}$
	5000	3	33/44/27	$6.2.10^{-3}$
	5000	4	31/43/34	$7{,}2.10^{-3}$
Ma trận	5000	1	5/13/28	3.10^{-3}
	5000	2	0/8/21	$1{,}9.10^{-3}$
	5000	3	0/6/16	$1{,}4.10^{-3}$
	5000	4	0/0/20	$1{,}3.10^{-3}$
Gần tối ưu 102/127/87	5000	1	1/8/0/0	$4{,}5.10^{-4}$
	5000	2	1/0/2/1	2.10^{-4}
	5000	3	5/2/4/2	$7{,}5.10^{-4}$
	5000	4	27/21/28/30	$5{,}3.10^{-3}$

Hình sau trình bày biểu đồ thực hiện của mã Turbo có cùng kích cỡ khung framesize=1000, Eb/No=3dB với 3 kiểu chèn khác nhau.



Hình 14: Thực hiện mã turbo với framesize=1000, Eb/No=3dB

Nhận xét:

- Khi số lần lặp tăng từ 1 → 2 thì việc thực hiện mã Turbo được cải tiến nhiều. Điều này là do sau khi thông tin được chia sẽ giữa các bộ giải mã với nhau thì các bộ giải mã có nhiều thông tin về ngõ vào và vì vậy đưa ra quyết định chính xác hơn.
- Khi số lần lặp tăng lớn hơn 2 thì việc thực hiện của mã Turbo cũng được cải tiến. Tuy nhiên, mức độ cải tiến không được cao. Điều này là do sau vài lần lặp, các bộ giải mã đã lấy được hết thông tin (bức tranh) của mã ngõ vào và do đó, không cho ra ở ngõ ra các giá trị biến đổi nữa như trong lần lặp thứ nhất. Vì vậy, có thể nói thực hiện của mã Turbo sẽ đạt đến mức ngưỡng sau vài lần lặp.

- Nếu số lần lặp tăng hơn mức ngưỡng thì việc thực hiện mã turbo sẽ bị giảm xuống. Sau mức ngưỡng thì các lần lặp sau không đem đến thông tin khác hơn đến các bộ giải mã.
- Như vậy, việc thực hiện mã turbo tăng khi số lần lặp tăng và thời gian sử dụng giải mã cũng tăng tuyến tính theo số lần lặp. Vì vậy, người thiết kế phải điều chỉnh số lần lặp sao cho phù hợp giữa việc thực hiện của mã và thời gian giải mã.
- Tuy nhiên, trong quá trình giải mã, thuật toán SOVA phải chịu 2 loại méo. Méo thứ nhất là các ngõ ra mềm vượt quá tối ưu thường được bù bằng hệ số chia mức (scale factor). Thứ hai là sự tương quan giữa thông tin bên ngoài và bên trong hay sự tương quan giữa ngõ ra mềm của mỗi bộ giải mã tương ứng với các bit kiểm tra chẵn lẻ của nó và chuỗi dữ liệu ngõ vào thông tin. Mặc dù ảnh hưởng méo thì rất nhỏ, nhưng sau nhiều lần lặp thì các méo này sẽ được tích lũy và có thể ảnh hưởng đến việc thực hiện của mã Turbo. Để khắc phục điều này có nhiều cách như : thiết kế bộ chèn mới S 2 bước [4], ... Chương trình mô phỏng chỉ trình bày đúng theo thuật toán mã hóa và giải mã của mã Turbo vì thiết kế bộ chèn này rất phức tạp.

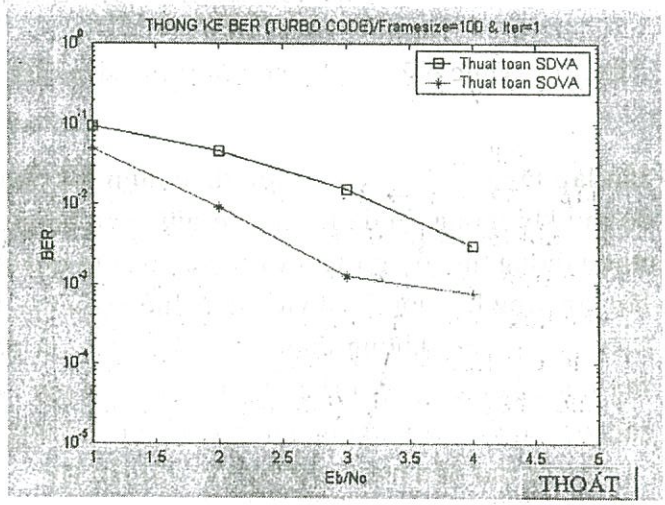
III.3 So sánh phương pháp giải mã Viterbi SDVA (mã tích chập) và giải mã SOVA (mã Turbo)

Mô phỏng dựa trên bộ chèn gần tối ưu, với cùng ngõ vào u_{in}=1000bit, kích cỡ khung Framesize=100, và SOVA có số lần lặp Iter=1.

Bảng 3: So sánh phương pháp giải mã SDVA và giải mã SOVA

Eb/No	Số bit sai dùng SDVA	BER dùng SDVA	Số bit sai dùng SOVA	BER dùng SDVA
1	98/92/98/87	$9,3.10^{-2}$	44/59/51/45	$4,97.10^{-2}$
2	35/52/58/40	$4,62.10^{-2}$	10/7/2/18	$9,25.10^{-3}$
3	21/14/6/19	$1,5.10^{-2}$	4/1/0/0	$1,23.10^{-3}$
4	8/3/8/9	3.10^{-3}	0/0/3/0	$7,5.10^{-4}$

Hình sau trình bày biểu đồ so sánh thực hiện của 2 phương pháp giải mã SDVA và SOVA có cùng chuỗi ngõ vào u_{in}, kích cỡ khung framesize=100 (hay cùng chuỗi mã hóa).



Hình 15: So sánh phương pháp giải mã SDVA và giải mã SOVA

Nhận xét :

- Với cùng chuỗi mã hóa , nếu dùng thuật toán SDVA thì cho ra kết quả BER cao hơn thuật toán SOVA hay nói cách khác, thuật toán SOVA giải mã cho ra kết quả lỗi sai ít hơn.
- Độ lợi giải mã của thuật toán SOVA so với SDVA tại BER= khoảng bằng dB
- Thuật toán giải mã mã Turbo gồm 2 bộ SOVA, mỗi bộ SOVA bao gồm 2 bộ SOVA1 và SOVA2. Thuật toán SOVA khác thuật toán SDVA ở chỗ nó có tính đến giá trị cập nhật. Do đó, thuật toán giải mã Turbo dùng SOVA có độ phức tạp tính toán cao gấp 4 lần thuật toán giải mã Viterbi. Khi số lần giải mã tăng thì độ phức tạp của bộ giải mã cũng tăng và dẫn đến thời gian giải mã tăng.

APPLYING TURBO CODE TO THIRD GENERAL MOBILE SYSTEM IMT-2000

Pham Hong Lien – Chung Thi Ngoc Hanh

ABSTRACT: The third general mobile system (3G)'s purpose supports many kinds of services for everyone at anytime and anywhere. These include such as: high speed data transmission, figure data and multimedia service up to 2Mbps beside traditional voice service.

With data rate up to 2Mbps, it is very necessary and important to enhance the transmission, line reliability in 3G. Channel coding is often used in digital mobile system in order to protect the information bits against interference and reduce the number of error bits. A code with Shannon limit performance is ideal, but so far, has not been achieved in practical. The only practical code that comes close to the Shannon limit is RSC Turbo code (Recursive Systematic Encoder).

The excellent performance funtion of Turbo code achieve near Shannon limit in case the large frame size on the AWGN channel (Additive White Gaussian Noise). According to IMT-2000 specification, Turbo code is accepted as a channel coding method for large frame data services by 3GPP(W-CDMA) and 3GPP2 (cdma2000).

TÀI LIỆU THAM KHẢO

[1]. Ericsson Wireless Communication Inc, cdma2000, ECT-360, Student Guide, pp.2.58-2.71(2000).

[2]. 3GPP2, cdma 2000 Standard, TIA/EIA, Vol.2, pp.2.67-2.77 (2000)

[3]. Fu-hua Huang, Evaluation of Soft Output Decoding for Turbo Codes, Blacksburg, Virginia, pp.24-67(1997).

[4]. H. R. Sadjadpour, N. J. A. sloane, M. Salehi, and G. Nebe, Interleaver Design for Turbo Codes, Department of electrical engineering, Northeast University, pp.2-12(2000) .

[5]. Berrou, Glavieux, Near Optimum Error Correct Coding and Decoding: Turbo Code, IEEE Transactions on Communication (1996).

[6]. Joachim Hagenauer, The Turbo Principle Tutorial Introduction and State of the Art, Technical University of Munich, Department of Communication Engineering, Germany (1999)

[7]. William E. Ryan, A Turbo Code Tutorial, New Mexico State University (1999).

[8]. D. Divsalar, S. Dolinar, and F. Pollara, Transfer Function Bounds on the Performance of Turbo Codes, Communication System and Research section (1998).

[9]. Nasir Ahmed, An Investigation into Turbo Codes, Rice University.

[10]. Im-Bum Chang, Tea-Young Chang, Jae-Hwan Kwon, Performance Improvement of Turbo Code with Increased Free Distances and Scaling Factors for Decoding, Korea University (2000).