# Luận văn

## Ứng dụng mã Turbo trong hệ thống thông tin di động CDMA2000

# MỞ ĐẦU

Cùng với sự phát triển của Khoa Học và Công Nghệ, công nghệ viễn thông trong những năm qua cũng đã có những bước phát triển mạnh mẽ ngày càng đáp được nhu cầu của con người.

Đặc biệt là thông tin di động đóng một vai trò rất quan trọng.Nhu cầu trao đổi thông tin ngày càng tăng cả về số lượng, chất lượng và các loại hình dịch vụ kèm theo điều này đòi hỏi phải tìm ra phương thức trao đổi thông tin mới .Và công nghệ CDMA là mục tiêu hướng tới của lĩnh vực thông tin di động trên toàn thế giới .

Công nghệ CDMA bao gồm nhiều ưu điểm nhưng vấn đề đặt ra là trao đổi thông tin bằng cách nào cho hiệu quả nhất. Làm sao cho thông tin không bị mất mát trên đường truyền để đảm bảo chức năng trao đổi thông tin và mã hoá là một phần quan trọng của công nghệ CDMA.Chính vì thế mã TURBO được sử dụng trong CDMA2000 do những tính năng và cấu trúc ưu việt hơn những mã khác. Để hiểu rõ những ưu điểm của công nghệ này khi sử dụng mã Turbo và đây là lí do em chọn đề tài tốt nghiệp:"Ứng dụng mã Turbo trong hệ thống thông tin di động CDMA2000".

Nội dung đồ án gồm 4 chương :

- Chương 1: Khái niệm về mã Turbo: Nói về sự kết nối các bộ mã tích chập hệ thống đệ quy để tạo nên mã Turbo và đưa ra các thành phần và kỷ thuật chung của bộ mã hoá Turbo kết nối song song .
- Chương 2: Tìm hiểu về bộ giải mã, và hai thuật toán giải mã là MAP và SOVA .
- Chương 3: Trình bày những ứng dụng của mã Turbo:Ứng dụng trong truyền thông không dây và truyền thông đa phương tiện. đi vào chi tiết ứng dụng của nó trong cdma2000
- Chương 4: Chương trình mô phỏng bộ mã Turbo sử dụng trong hệ thống thông tin di động cdma2000 để rút ra nhận xét về mã Turbo

Trong quá trình làm đồ án tốt nghiệp, mặc dù đã cố gắng nhiều nhưng vẫn không tránh những sai sót, em mong được sự phê bình, chỉ bảo và giúp đỡ của thầy cô và bạn bè.

Em xin chân thành cảm ơn sự giúp đỡ tận tình của thầy Nguyễn Văn Cường và các thầy cô giáo trong khoa Điện Tử-Viễn Thông đã giúp em hoàn thành đồ án này.

Đà Nẵng thang 06 năm 2007

# Chương 1: Mã turbo

## 1.1. Giới thiệu mã turbo:

Mã Turbo là sự kết nối gồm hai hay nhiều bộ mã riêng biệt để tạo ra một mã tốt hơn và cũng lớn hơn. Mô hình ghép nối mã đầu tiên được Forney nghiên cứu để tạo ra một loại mã có xác suất lỗi giảm theo hàm mũ tại tốc độ nhỏ hơn dung lượng kênh trong khi độ phức tạp giải mã chỉ tăng theo hàm đại số. Mô hình này bao gồm sự kết nối nối tiếp một bộ mã trong và một bộ mã ngoài.

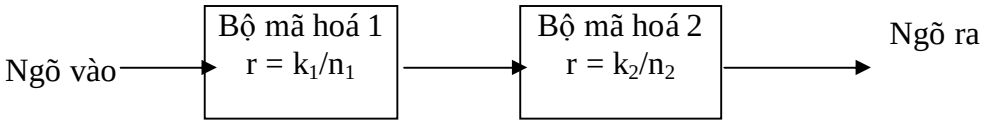
Chương này trình bày:

- Sự kết nối các mã và sự ra đời của mã Turbo( TC).
- Gới thiệu về mã chập hệ thống đệ quy (Recursive Systematic Convelutional Code_RSC), là cơ sở của việc tao ra mã TC.
- Chi tiết cấu trúc bộ mã hóa PCCC

## 1.2. Sự kết nối mã và ra đời của mã turbo (TURBO CODE*)*:

Forney đã sử dụng một bộ mã khối ngắn hoặc một bộ mã tích chập với giải thuật giải mã Viterbi xác suất lớn nhất làm bộ mã trong và một bộ mã Reed-Salomon dài không nhị phân tốc độ cao với thuật toán giải mã sửa lỗi đại số làm bộ mã ngoài.

Mục đích lúc đầu chỉ là nghiên cứu một lý thuyết mới nhưng sau này mô hình ghép nối mã đã trở thành tiêu chuẩn cho các ứng dụng cần độ lợi mã lớn. Có hai kiểu kết nối cơ bản là kết nối nối tiếp (**hình 1.1**) và kết nối song song ( **hình 1.2)**

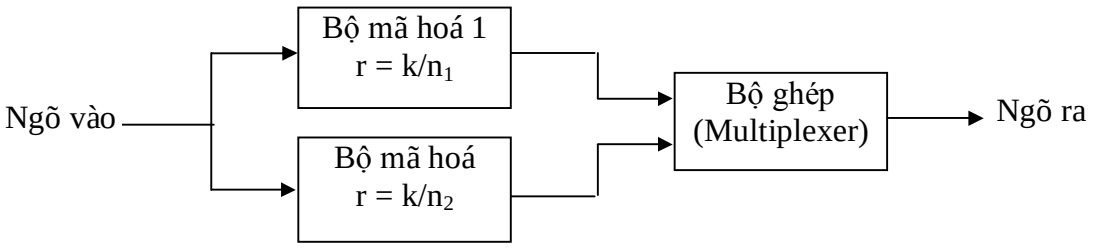



**Hình 1.1: Mã kết nối nối tiếp**

Bộ mã hoá 1 được gọi là bộ mã ngoài, còn bộ mã hoá 2 là bộ mã trong.

Đối với mã kết nối nối tiếp, tốc độ mã hoá: $R_{nt}=k_1k_2/n_1n_2$

Đối với mã song song, tốc độ mã hoá tổng: $R_{ss}=k/(n_1+n_2)$




**Hình 1.2: Mã kết nối song song**

Trên chỉ là các mô hình kết nối lý thuyết.Thực tế các mô hình này cần phải sử dụng thêm các bộ chèn giữa các bộ mã hoá nhằm cải tiến khả năng sửa sai.

Năm 1993, Claude Berrou**,** Alain Glavieux**,** Puja Thitimajshima đã cùng viết tác phẩm " Near Shannon limit error correcting coding and decoding:TURBO CODE" đánh dấu một bước tiến vượt bậc trong nghiên cứu mã sửa sai. Loại mã mà họ giới thiệu thực hiện trong khoảng 0.7dB so với giới hạn của Shannon cho kênh AWGN. Loại mã mà họ giới thiệu được gọi là mã Turbo, thực chất là sự kết nối song song các bộ mã tích chập đặc biệt cùng với các bộ chèn. Cấu hình này gọi là: "Kết nối song song các mã tích chập "( **P**arallel **C**oncatenated **C**onvolutional **C**ode-**PCCC**)

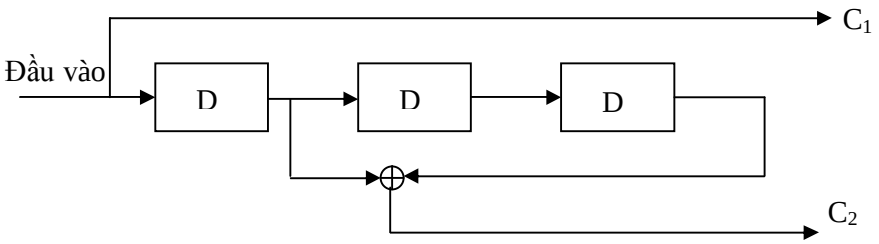
Ngoài ra cũng có "Kết nối nối tiếp các mã tích chập"(**S**erial **C**oncatenated **C**onvolutional **C**ode**_SCCC**) và dạng "Kết nối hổn hợp các bộ mã tích chập" ( **H**ybrid **C**oncatenated **C**onvolutional **C**ode**_HCCC**).Các loại mã này có nhiều đặc điểm tương tự nhau và cùng xuất phát từ mô hình của Berrou nên gọi chung là: **turbo code (TC)**

### 1.3. Bộ mã hóa tích chập hệ thống đệ quy RSC:

Trong bộ mã TC sử dụng một bộ mã tích chập đặc biệt: mã tích chập hệ thống đệ quy ( **R**ecursive **S**ystematic **C**onvolutional **Code_RSC** ).

#### 1.3.1. Mã tích chập hệ thống và không hệ thống:

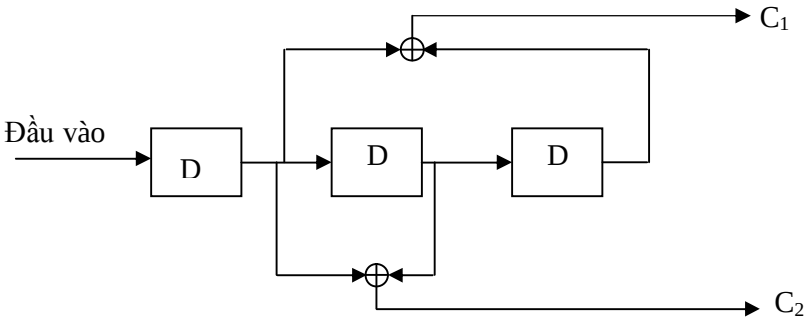
Mã tích chập có tính hệ thống là mã tích chập mà có một phần từ mã ở ngõ ra chính là dãy tin đầu vào, tức là đầu vào của dãy tin được đưa trực tiếp đến một trong những ngõ ra của bộ mã. Sơ đồ của bộ mã tích chập hệ thống như **hình 1.3**




**hình 1.3 Bộ mã hóa tích chập hệ thống**

đối với mã chập hệ thống thì ta có thể dễ dàng xác định từ mã ở ngõ ra hơn so với mã chập không hệ thống. Do cấu trúc như vậy nên yêu cầu của bộ mã hóa và giải mã ít phức tạp hơn so với mã không hệ thống

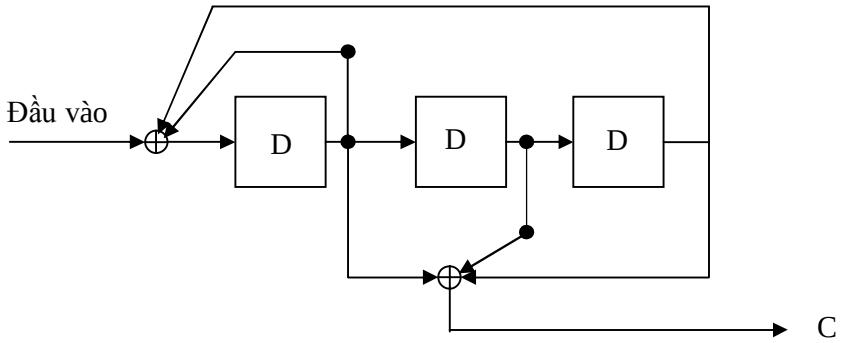
Mã chập không hệ thống có từ mã ngõ ra không phản ánh được dãy tin ở đầu vào, tức là đầu ra của bộ mã không nối trực tiếp đến dãy tin đầu vào. Sơ đồ của bộ mã chập không hệ thống như **hình 1.4**




**Hình 1.4 Bộ mã tích chập không hệ thống**

### 1.3.2. Mã tích chập đệ quy và không đệ quy:

Mã tích chập đệ quy có từ mã ở ngõ ra được đưa hồi tiếp trở lại dãy tin đầu vào. Sơ đồ như **hình 1.5**




**Hình 1.5 bộ mã tích chập đệ quy**

Mã tích chập không đệ quy có từ mã ở ngõ ra của bộ mã không được đưa hồi tiếp trở lại đầu vào. Sơ đồ như **hình 1.4**

### 1.3.3. Bộ mã tích chập hệ thống đệ quy:

Để mô tả bộ mã hóa mã chập người ta đưa ra các thông số của bộ mã hóa như sau : (n, k, K) trong đó:

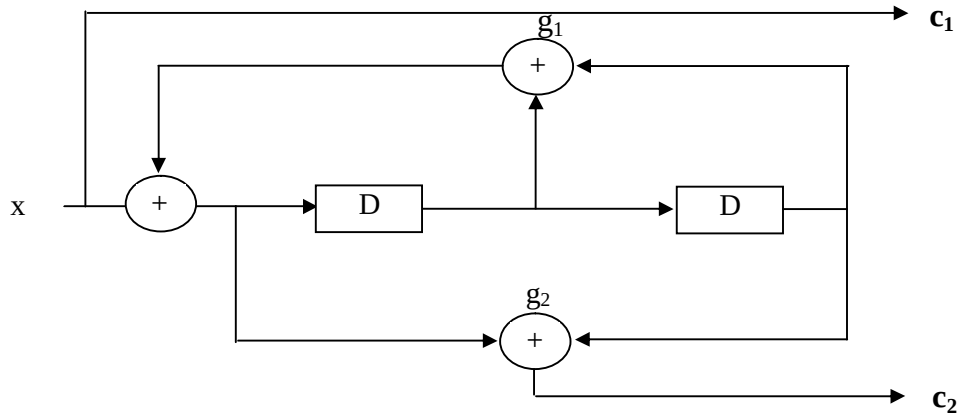
k : số đầu vào

n :số đầu ra

K:chiều dài constraint lengths (số ngăn lớn nhất trên thanh ghi)

Trong đó k < n để ta có thể thêm độ dư vào luồng dữ liệu để thực hiện phát hiện sai và sửa sai.

Một bộ mã tích chập thông thường được biểu diễn qua các chuỗi $g_1$= [1 1 1] và $g_2$ = [ 1 0 1] và có thể được viết là **G** = [ $g_1$,$g_2$] **.**Bộ mã hoá RSC tương ứng bộ mã hoá tích chập thông thường đó được biểu diễn là **G =** [ 1, $g_2/g_1$ ] trong đó ngõ ra đầu tiên ( biểu diễn bởi $\mathbf{g_1}$**)** được hồi tiếp về ngõ vào, $\mathbf{g_1}$ là ngõ ra hệ thống, $\mathbf{g_2}$ là ngõ ra feedforward. **Hình 1.6** trình bày bộ mã hoá RSC



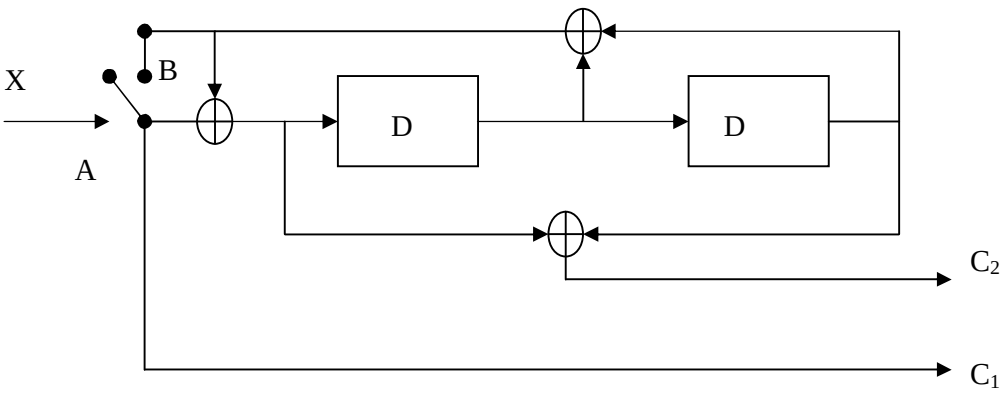

**Hình 1.6: Bộ mã hoá RSC với r=1/2 k=3**

Một bộ mã hoá tích chập đệ quy có khuynh hướng cho ra các từ mã có trọng số tăng so với bộ mã hoá không đệ quy, nghĩa là bộ mã tích chập đệ quy cho ra ít từ mã có trọng số thấp và cũng dẫn đến việc thực hiện sửa sai tốt hơn

Đối với mã Turbo, mục đích của việc thực hiện các bộ mã hoá RSC là tận dụng bản chất đệ quy của các bộ mã hoá và tận dụng sự kiện bộ mã hoá là hệ thống

### 1.3.4. kết thúc TRELLIS:

Đối với bộ mã tích chập thông thường, Trellis được kết thúc bằng( **m= k -1)** các bit zero thêm vào sau chuỗi ngõ vào. Các bit thêm vào này lái bộ mã tích chập thông thường đến trạng thái tất cả zero ( là trạng thái kết thúc trellis). Nhưng cách này không thể áp dụng cho bộ mã hoá RSC do có quá trình hồi tiếp. Các bit thêm vào để kết thúc cho bộ mã hoá RSC phụ thuộc vào trạng thái của bộ mã hoá và rất khó dự đoán. Ngay cả khi tìm được các bit kết thúc cho một trong các bộ mã hoá thành phần thì các bộ mã hoá thành phần khác có thể không được lái đến trạng thái tất cả zero với cùng các bit kết thúc do có sự hiện diện của bộ chèn giữa các bộ mã hoá thành phần. **Hình 1.7** là kết thúc trellis :




**Hình 1.7: Cách thức kết thúc trellis ở bộ mã RSC**

Để mã hoá chuỗi ngõ vào, khoá chuyển bật đến vị trí A, để kết thúc trellis thì khoá chuyển bật đến vị trí B.

### 1.4. Quyết định cứng và quyết định mềm:

Chuỗi tin sau khi truyền qua kênh truyền và được giải điều chế (dumodulate) thì sẽ được đưa đến bộ giải mã. Tín hiệu tại ngõ ra của bộ giải điều chế và ngõ vào của bộ giải mã sẽ quyết định quá trình giải mã là " cứng "hay "mềm ".

Nếu tín hiệu đến của bộ giải điều chế và được bộ điều chế ra quyết định từng bít là bít 0 hay 1 thì gọi là quyết định cứng. Ví dụ xét một hệ thống sử dụng tín hiệu đường dây là bipolar NRZ với biên độ là ± 1V. Nếu giá trị nhận được là 0,8V hoặc 0,03V thì đều được quyết định là bit 1. Còn nếu giá trị nhận được là -0,7V hoặc -0,02 thì đều được quyết định lá bít 0. như vậy ta thấy được phương pháp sai sót của quyết định cứng là dù 0,8V hay 0,03V thì bộ giải mã cũng nhận được bít 1 dù giá trị 0,8V có xác suất đúng là bit 1 cao hơn nhiều so với 0,03V. Như vậy, bộ giải mã không có thông tin nào về độ chính xác của quyết định từ bộ giải điều chế. Việc này sẽ làm cho chất lượng của bộ giải mã không chỉ phụ thuộc vào bộ giải mã mà còn phụ thuộc vào bộ giải điều chế và chất lượng không cao. Tuy nhiên quyết định cứng dễ dàng hơn cho việc giải mã.

Nếu bộ giải điều chế không tự quyết định xem giá trị lấy mẫu nhận được là bit 0 hay bit 1 mà đưa thẳng cho bộ giải mã để bộ giải có đầy đủ thông tin về bit sau khi

đã qua kênh truyền thì với cấu trúc phù hợp bộ giải mã sẽ cho các quyết định chính xác hơn, tức là chất lượng cao hơn. Bộ giải mã sẽ tính toán các giá trị để xét độ tin cậy của từng giá trị và cuối cùng mới quyết định. Điều này sẽ làm giảm khả năng có thể xẩy ra lỗi và độ lợi mã tổng cộng có thể tăng 2,5 dB so với giải mã cứng đối với môi trường có SRN thấp. Tuy nhiên, để đạt được độ lợi mã này thì bộ giải mã mềm sẽ có độ phức tạp cao hơn rất nhiều so với bộ giải mã cứng.

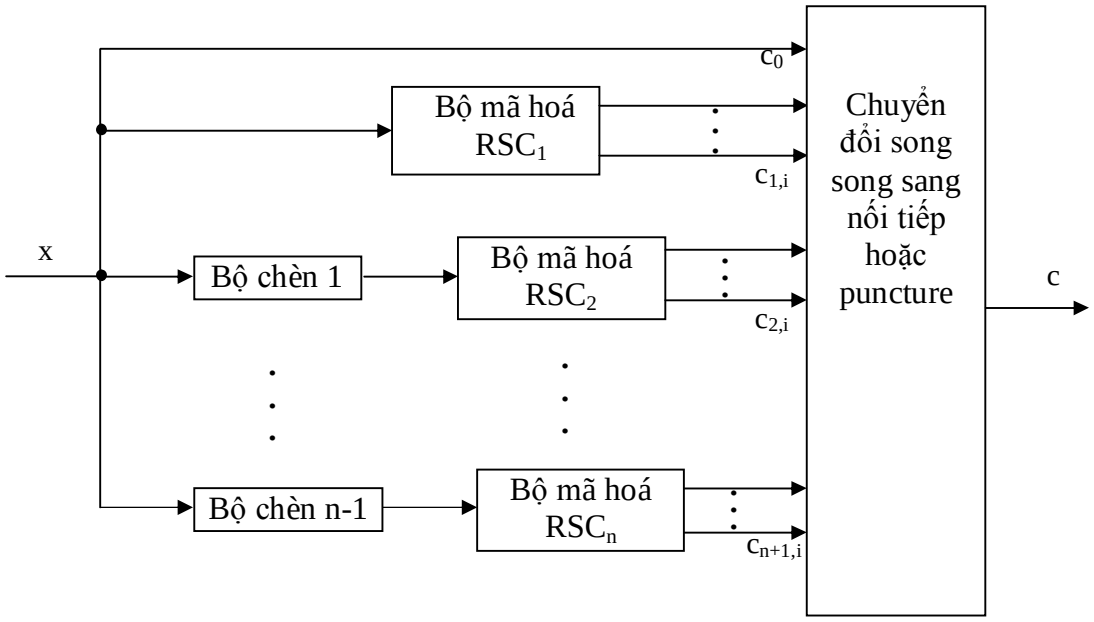
Với khả năng tính toán của các chíp vi xử lý hay các chíp DSP cùng với khối lượng bộ nhớ ngày nay thì sự phức tạp của bộ giải mã mềm không còn lá vấn đề lớn. vì thế xu hướng hiện nay trên thế giới là sử dụng bộ giải mã mềm, thậm chí có thể giải mã lại cho các loại mã khối và mã tích chập truyền thống bằng phương pháp giải mã mềm.

## 1.5. Mã hóa mã turbo PCCC (parallel concatenated convolutional code)

### 1.5.1. Bộ mã hóa:

Mã PCCC là sự kết nối song song của 2 hay nhiều mã RSC. Thông thường người ta sử dụng tối thiểu 2 bộ mã hoá tích chập .Sơ đồ khối mã PCCC tổng quát được trình như **hình 1.7**

Mỗi bộ mã hoá $RSC_i$ được gọi là các bộ mã thành phần (constituent code).Các bộ mã thành phần có thể khác nhau, tốc độ mã khác nhau nhưng có cùng cỡ khối bit ngõ vào là k ,các chuỗi mã hoá ngõ ra bao gồm một chuỗi hệ thống (chuỗi bit vào).Ở các bộ mã hoá thứ hai trở đi, chuỗi bit nhận vào để mã hoá trước hết phải qua một bộ chèn.Tất cả các chuỗi mã hoá ngõ ra sẽ được hợp lại thành một chuỗi bit duy nhất n bit trước khi truyền .



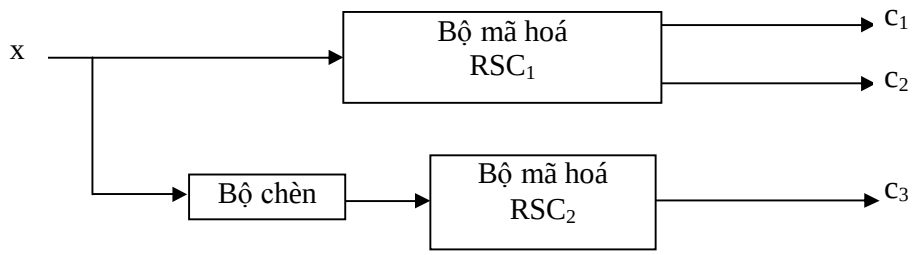

**Hình 1.8: Bộ mã hoá PCCC tổng quát**

Tốc độ mã hoá (code rate) của bộ mã hoá PCCC là: r = k/n

Mỗi bit thông tin ngõ vào sẽ trở thành một phần của từ mã ngõ ra (tính hệ thống) và sẽ được kèm theo bằng ( 1/r - 1) bit ( gọi là bit parity) để sửa lỗi nếu có. Nếu r càng nhỏ tức số bit parity đi kèm sẽ lớn và dẫn đến khả năng sửa lỗi cao hơn rất nhiều nhưng tốc độ truyền giảm đi, số bit truyền nhiều hơn có nghĩa là băng thông lớn hơn và độ trễ tăng lên. Theo khuyến cáo của các tổ chức định chuẩn thì giá trị r chỉ nên nhỏ nhất là 1/6 **.**

Trong quá trình hợp các chuỗi mã hoá thành một chuỗi mã hoá duy nhất ta có thể dùng một kỹ thuật khá mới mẻ đó là kỹ thuật xoá (puncture) .

Một mã Turbo tiêu biểu là loại được kết nối theo kiểu PCCC. Sơ đồ khối được biểu diễn trong **hình 1.9**




**Hình 1.9: Mã PCCC tốc độ 1/3 gồm 2 bộ mã hoá chập hệ thống đệ quy**

Bộ mã hoá Turbo cơ bản được thiết kế bằng cách kết nối song song hai bộ mã hoá hệ thống đệ quy tích chập lại với nhau, hai bộ mã hoá thành phần được phân cách nhau bởi một bộ chèn ( interleaving). Chỉ có một trong ba đầu ra của hai bộ RSC trên là đầu ra của hệ thống, đầu ra của hệ thống có được bằng cách thay đổi thứ tự vị trí của bit đầu vào. Tốc độ mã hoá của bộ mã này là r =1/3, bộ mã hoá RSC đầu tiên cho ra chuỗi hệ thống $c_1$ và chuỗi chập đệ quy $c_2$, trong khi bộ mã hoá RSC thứ hai thì bỏ qua chuỗi hệ thống của nó và chỉ cho ra chuỗi chập đệ quy $c_3$ .

### 1.5.2. Kỷ thuật xóa (punture):

Kỹ thuật xoá là kỷ thuật dùng để tăng tốc độ mã của một bộ mã hoá mà không làm thay đổi cấu trúc của bộ mã hoá.Tốc độ mã càng thấp thì chất lượng càng cao nhưng băng thông tăng.Ví dụ bộ mã tốc độ 1/3 có thể trở thành bộ mã hoá tốc độ 1/2 bằng cách thay vì 1 bit ngõ vào sẽ có tương ứng 3 bit ngõ ra mã hoá thì ta cho ngõ ra mã hoá chỉ còn 2 bit. Bản chất của kỷ thuật puncture là làm giảm n theo một qui luật nào đó để tốc độ mã hoá r tăng lên.

Ví dụ: bộ mã trong **hình 1.9**, nếu chuỗi hệ thống $c_1$ vẫn giữ nguyên và các chuỗi $c_2$ và $c_3$ sẽ được lấy xen kẽ. Chuỗi $c_2$ sẽ lấy các bit lẻ và các bit chẵn của chuỗi $c_3$ thì bộ mã sẽ có tốc độ 1/2. Khi bộ giải mã nhận được chuỗi bit đến thì nó sẽ thêm vào chuỗi này các bit 0 tại những chỗ đã bị xoá bớt. Như vậy có thể làm sai lệch bit parity nên giảm chất lượng.

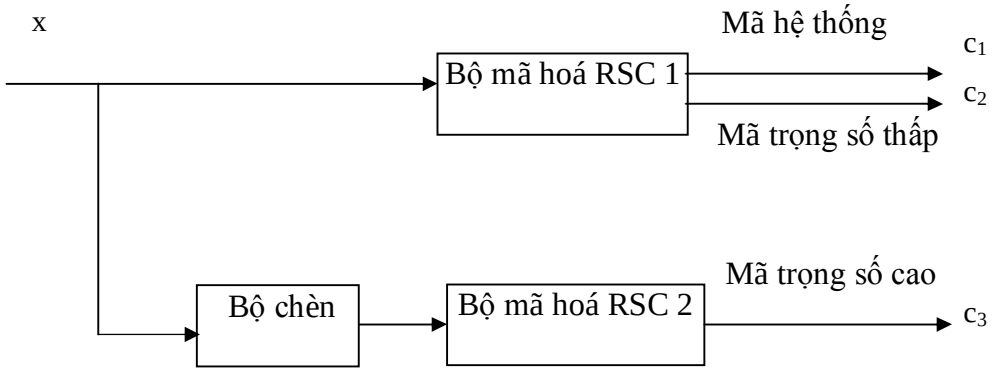
### 1.5.3. Bộ chèn (interleaver):

Đối với mã Turbo, có thể có một hay nhiều bộ chèn được sử dụng giữa các bộ mã hoá thành phần. Bộ chèn được sử dụng tại bộ mã hoá nhằm mục đích hoán vị tất cả các chuỗi ngõ vào có trọng số thấp thành chuỗi ra có từ mã ngõ ra trọng số cao

hay ngược lại. Luôn đảm bảo với một chuỗi ngõ vào thì ngõ ra một bộ mã hoá sẽ cho từ mã trọng số cao còn bộ mã hoá kia sẽ cho ra từ mã trọng số thấp để làm tăng khoảng cách tự do tối thiểu.

Bộ chèn không những được sử dụng tại bộ mã hoá mà nó cùng với bộ giải chèn (deinterleaver) có trong bộ giải mã đóng một vai trò quan trọng. Vai trò của bộ chèn chính tại bộ giải mã mới bộc lộ hết. Một bộ chèn tốt sẽ làm cho các ngõ vào của bộ giải mã SISO ít tương quan với nhau tức là mức độ hội tụ của thuật toán giải mã sẽ tăng lên, đồng nghĩa với việc giải mã chính xác hơn.

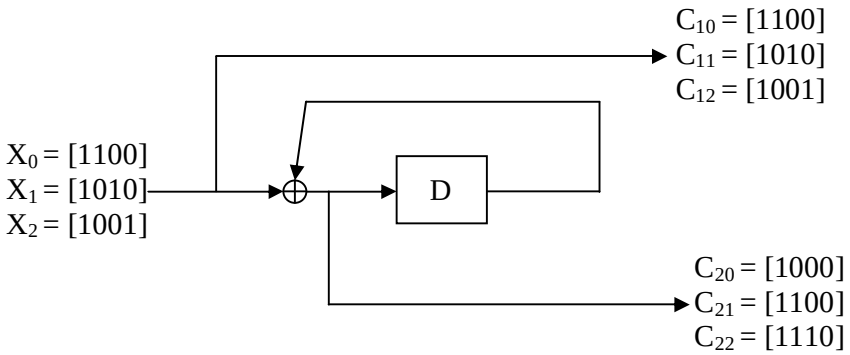
Ví dụ bộ chèn được sử dụng để tăng trọng số của các từ mã như trong **hình 1.10**




**Hình 1.10: Bộ chèn làm tăng trọng số mã của bộ mã hoá RSC2 khi so sánh với bộ mã hoá RSC1**

Từ **hình 1.10**, đối với bộ mã hoá RSC1 thì chuỗi ngõ vào x cho ra chuỗi mã tích chập đệ quy có trọng số thấp $c_2$. Để tránh bộ mã hoá RSC2 cho ra chuỗi ngõ ra đệ quy khác cũng có trọng số thấp, bộ chèn hoán vị chuỗi ngõ vào x thành 1 chuỗi mới với hi vọng cho ra chuỗi mã tích chập đệ quy có trọng số cao $c_3$. Vì vậy, trọng số mã của mã PCCC là vừa phải, nó được kết hợp từ mã trọng số thấp của bộ mã hoá 1 và trọng số cao của bộ mã hoá 2. **hình 1.11** là một ví dụ minh họa.

Theo **hình 1.11** chuỗi ngõ vào $x_i$ cho ra các chuỗi ngõ ra $c_{1i}$ và $c_{2i}$ tương ứng. các chuỗi ngõ vào $x_1$ và $x_2$ là các chuỗi hoán vị khác nhau của $x_0$. **bảng 1.1** trình bày kết quả của các từ mã và trọng số của các từ mã




**Hình 1.11. Ví dụ minh họa khả năng của bộ chèn**

| | Chuỗi ngõ vào $x_i$ | Chuỗi ngõ ra $C_{1i}$ | Chuỗi ngõ ra $C_{2i}$ | Trọng số của từ mã i |
|---|---|---|---|---|
| i = 0 | 1100 | 1100 | 1000 | 3 |
| i = 1 | 1010 | 1010 | 1100 | 4 |
| i = 2 | 1001 | 1001 | 1110 | 5 |

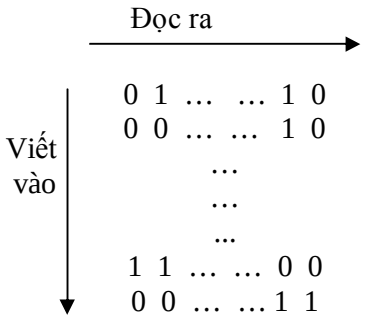
**Bảng 1.1 các chuỗi ngõ vào và ngõ ra của bộ mã hóa trong hình 1.11**

Từ bảng trên cho thấy trọng số của từ mã có thể tăng bằng cách sử dụng bộ chèn.

Bộ chèn ảnh hưởng đến việc thực hiện mã vì nó ảnh hưởng trực tiếp đến đặc tính khoảng cách của mã. Bằng cách tránh các từ mã có trọng số thấp, BER của mã turbo có cải tiến đáng kể. Vì vậy có nhiều bộ chèn khác nhau đã được nghiên cứu thiết kế. phần sau đây trình bày các bộ chèn tiêu biểu thường sử dụng trong việc thiết kế mã turbo.

**1.5.3.1. Bộ chèn ma trận (bộ chèn khối):**

Bộ chèn ma trận là bộ chèn thường được sử dụng nhất trongg các hệ thống liên lạc. Nó viết vào theo cột từ trên xuống dưới, từ trái sang phải và đọc ra theo hàng từ trái sang phải và từ trên xuống dưới. hoặc có thể viết vào theo hàng và đọc ra theo cột. như hình dưới




| | | |
|---|---|---|
| $x_1$ | $x_7$ | $x_{13}$ |
| $x_2$ | $x_8$ | $x_{14}$ |
| $x_3$ | $x_9$ | $x_{15}$ |
| $x_4$ | $x_{10}$ | $x_{16}$ |
| $x_5$ | $x_{11}$ | $x_{17}$ |
| $x_6$ | $x_{12}$ | $x_{18}$ |

Với chuỗi vào ($x_1, x_2, x_3, \ldots\ldots x_{17}, x_{18}$ ) dùng ma trận bộ chèn $6\times3$ ở trên thì chuỗi ra là:

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $x_1$ | $x_7$ | $x_{13}$ | $x_2$ | $x_8$ | $x_{14}$ | … | … | $x_{12}$ | $x_{18}$ |

**1.5.3.2. Bộ chèn helical:**

Tương tự bộ chèn ma trận (hàng cột ), bộ chèn helical cũng ghi vào theo hàng (hoặc cột ) nhưng lại đọc ra theo đường chéo.

Ví dụ : các gia trị đọc vào là như bảng sau

| | | |
|---|---|---|
| $x_1$ | $X_6$ | $x_{11}$ |
| $x_2$ | $X_7$ | $x_{12}$ |

| | | |
|---|---|---|
| $x_3$ | $X_8$ | $x_{13}$ |
| $x_4$ | $X_9$ | $x_{14}$ |
| $x_5$ | $X_{10}$ | $x_{15}$ |

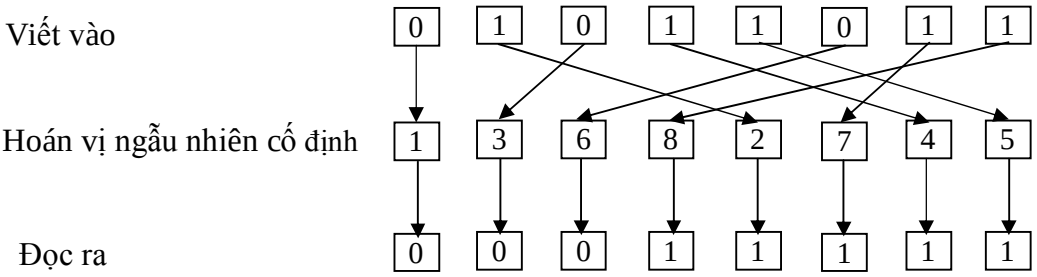
Các giá trị đọc ra là:

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $X_5$ | $X_9$ | $X_{13}$ | $X_3$ | $X_7$ | $X_{11}$ | $X_1$ | $X_{10}$ | $X_{14}$ | … | $X_{15}$ |

Một điều cần lưu ý là ma trân chèn helical có số hàng lẻ

### 1.5.3.3. Bộ chèn giả ngẫu nhiên:

Bộ chèn giả ngẫu nhiên sử dụng tính ngẫu nhiên cố định tức là sắp xếp các chuỗi ngõ vào theo một thứ tự hoán vị. Giả thiết độ dài của chuỗi ngõ vào là L. hình sau trình bày bộ chèn ngẫu nhiên với L = 8.




**Hình 1.12: Bộ chèn giả ngẫu nhiên với độ dài chuỗi ngõ vào L= 8**

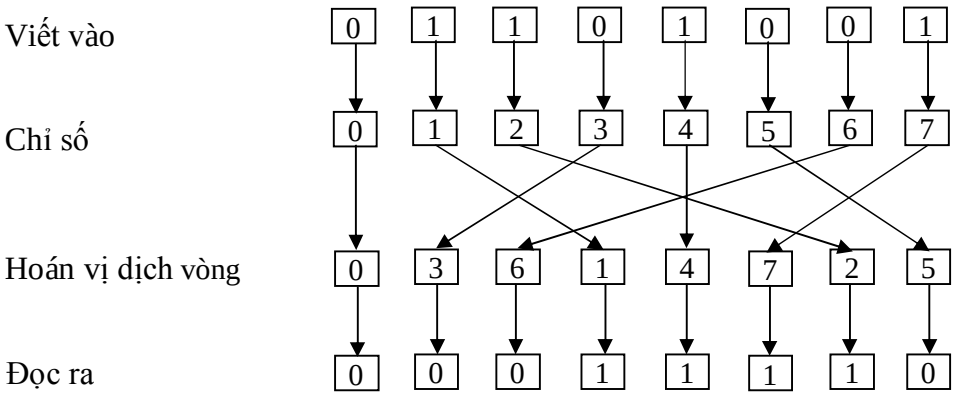
Bộ chèn giả ngẫu nhiên sử dụng tính ngẫu nhiên cố định và sắp xếp chuỗi ngõ vào theo thứ tự hoán vị. Như hình trên bộ chèn viết vào [ 01011011] và đọc ra [ 00011111]

### 1.5.3.4. Bộ chèn dịch vòng:

Phép hoán vị p của bộ chèn dịch vòng được định nghĩa:

$P(i)= (ai + s) \bmod L$

Yêu cầu $a$ < L, $a$ gần bằng $\sqrt{L}$ và $s$ < L trong đó i là chỉ số, a là kích cỡ của bước và s là phần bù (offset). **Hình 1.13** trình bày bộ chèn dịch vòng với L = 8, a=3 và s=0



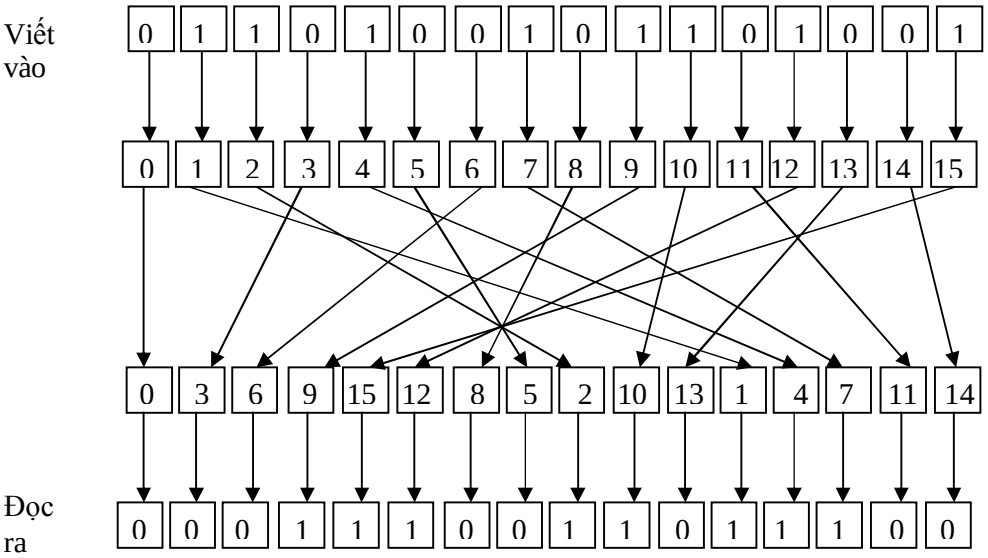

**Hình 1.13: Bộ chèn dich vòng với L=8, a=3, s=0**

Từ hình trên bộ chèn viết vào [01101001] và đọc ra [00011110]. Việc tách bit lân cận là 3 hay 5. bộ chèn này được đưa ra để hoán vị các chuỗi ngõ vào có trọng số 2 có các trọng số từ mã thấp thành các chuỗi ngõ vào trọng số 2 có các trọng số từ mã cao. Tuy nhiên, bởi vì tính quy tắc vốn có của bộ chèn này, sẽ khó hoán vị các chuỗi ngõ vào trọng số cao hơn có các trọng số từ mã thấp thành các chuỗi ngõ vào khác có các trọng số từ mã cao.

### 1.5.3.5. Bộ chèn bán ngẫu nhiên :

Bộ chèn bán ngẫu nhiên là sự thỏa hiệp giữa bộ chèn ngẫu nhiên và bộ chèn được thiết kế như là các bộ chèn khối và dịch vòng. Thuật toán hoán vị cho bộ chèn bán ngẫu nhiên được mô tả như sau:

1. chọn chỉ số ngẫu nhiên $t \in [0, L-1]$
2. chọn số nguyên dương $s < \sqrt{\frac{L}{2}}$
3. so sánh i với các số dương trước đó. Đối với mọi số nguyên s, so sánh i nếu nó nằm trong khoảng $\pm$ s thì giữ i, nếu nó không nằm trong khoảng $\mp$s thì trở lại từ đầu.
4. trở lại từ đầu cho đến khi tất cả các vị trí L được lấp đầy.




Hình trên trình bày bộ chèn bán ngẫu nhiên với L=16 và s=2. bộ chèn viết vào [0110100101101001] và đọc ra [0001110011011100]. Bộ chèn bán ngẫu nhiên cố gắng đưa ra vài tính ngẫu nhiên để khắc phục tính quy tắc của việc hoán vị. Tuy nhiên, thuật toán không đảm bảo kết thúc một cách thành công.

### 1.5.3.6. Bộ chèn chẵn lẻ:

Bộ chèn chẵn lẻ là đặc trưng cho mã PCCC r = 1/2. Một mã PCCC r =1/2 được lấy bằng cách kết hợp 2 chuỗi ngõ ra của mã PCCC r = 1/3 thành một chuỗi ngõ ra của mã PCCC r = 1/2. Tuy nhiên, bằng cách kết hợp 2 chuỗi ngõ ra được mã hóa này, có thể một bit thông tin sẽ không có các bit mã hóa của nó ( hoặc cả hai bit mã hóa kết hợp lại cho ra sửa sai cho cùng một bit tin). Cũng có thể một bit tin có một hay cả hai bít được mã hóa của nó. Vì vậy, nếu một lỗi xẩy ra cho bit tin không được bảo vệ ( không có một bit nào của nó được mã hóa của nó), thì chất lượng của bộ bộ giải mã TC có thể bị giảm hay BER của nó có thể tăng.

Bộ chèn chẵn lẻ có thể khắc phục được vẫn đề này bằng cách cho phép mỗi bit tin có một trong các bít được mã hóa của nó một cách chính xác. Như kết quả của

bộ chèn này, khả năng sửa sai của mã được phân bố đồng nhất trên tất cả các bít tin. Thực sự bộ chèn này giống như một cách cải tiến của kỷ thuật punture.

Ví dụ bộ chèn chăn lẻ sau

Chuỗi tin x = $c_1$ của L = 9 sau khi đi qua bộ mã hoá RSC1 thì cho ra chuỗi mã hóa $c_2$. Từ chuỗi $c_2$, chỉ có các bit mã hoá ở vị trí lẻ được lưu trữ như trong bảng. Chỉ số dưới là vị trí bit trong chuỗi bit

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $x_7$ | $x_8$ | $x_9$ |
|---|---|---|---|---|---|---|---|---|
| $c_{21}$ | - | $c_{23}$ | - | $c_{25}$ | - | $c_{27}$ | - | $c_{29}$ |

Một bộ chèn khối $3\times3$ được dùng để hoán vị chuỗi tin tức x cho bộ mã hóa $RSC_2$ như sau:

| | | |
|---|---|---|
| $x_1$ | $x_4$ | $x_7$ |
| $x_2$ | $x_5$ | $x_8$ |
| $x_3$ | $x_6$ | $x_9$ |

Chuỗi tin tức x được viết theo cột và đọc ra theo hàng. Chuỗi tin được hoán vị cho ra chuỗi mã hóa $c_3$. Từ chuỗi $c_3$ chỉ có các bit mã hoá vị trí chẵn được lưu trữ như trong bảng dưới.

Các bit mã hóa lẻ của chuỗi $c_3$ được lưu trữ với chuỗi tin hoán vị x

| $x_1$ | $x_4$ | $x_7$ | $x_2$ | $x_5$ | $x_8$ | $x_3$ | $x_6$ | $x_9$ |
|---|---|---|---|---|---|---|---|---|
| - | $c_{34}$ | - | $c_{32}$ | - | $c_{38}$ | - | $c_{36}$ | - |

Đối với mã hoá PCCC r = 1/2, các chuỗi bit mã hoá sau đó phải được ghép với nhau như trong bảng sau

Chuỗi tin x và chuỗi mã hóa được ghép

| $x_1$ | $x_4$ | $x_7$ | $x_2$ | $x_5$ | $x_8$ | $x_3$ | $x_6$ | $x_9$ |
|---|---|---|---|---|---|---|---|---|
| $c_{21}$ | $c_{34}$ | $c_{27}$ | $c_{32}$ | $c_{25}$ | $c_{38}$ | $c_{23}$ | $c_{36}$ | $c_{29}$ |

Từ bảng trên ta thấy mỗi bit tin có bit mã hóa riêng của nó

**1.5.3.7. Bộ chèn simile:**

Bộ chèn chẵn lẻ như trình bày ở trên cho duy nhất một bit kiểm tra đi kèm theo một bit mã hoá. Bằng cách này khả năng sửa lỗi của bộ mã được phân bố đều trên tất cả các bit thông tin. Bây giờ ta xem xét một hạn chế khác khi thiết kế bộ chèn : sau khi mã hoá cả hai chuỗi bit thông tin (một chuỗi gốc và một chuỗi sau khi qua bộ chèn) trạng thái của hai bộ mã hoá phải giống nhau. Ngoài cách kết thúc trellis như đã trình bày ở phần trước, ta vẫn có thể thêm vào sau chuỗi thông tin một số bit (gọi là "tail bits") để làm cho hai bộ mã hoá kết thúc ở cùng một trạng thái zero bằng cách dùng một bộ chèn đặc biệt gọi là simile.

Ý tưởng của bộ chèn này xuất phát từ ý tưởng một khối thông tin N bit có thể được chia thành m + 1 chuỗi với m là số ô nhớ của bộ mã hóa. Ví dụ với m = 2 ta có

Chuỗi 0 = $\{u_k | k \bmod(m+1) = 0\}$

Chuỗi 1 = $\{u_k | k \bmod(m+1) = 1\}$

Chuỗi 2 = $\{u_k | k \bmod(m+1) = 2\}$

Ví dụ như với bộ RSC đã nói trên với một N cho trước, trạng thái cuối cùng của bộ mã hoá mô tả bằng trạng thái của hai D flip-flop sẽ là sự kết hợp của hai chuỗi vừa nêu trên thể hiện trong bảng sau.

| *N mod (m+1)* | $S^0_N$ | $S^1_N$ |
|---|---|---|
| 0 | Chuỗi 1 + chuỗi 2 | Chuỗi 0 + chuỗi 1 |
| 1 | Chuỗi 2 + chuỗi 0 | Chuỗi 1 + chuỗi 2 |
| 2 | Chuỗi 0 + chuỗi 1 | Chuỗi 2 + chuỗi 0 |

Một kết luận quan trọng là từ quan điểm trạng thái kết thúc của bộ mã hóa, thứ tự của các bit đơn lẻ trong mỗi chuỗi không còn quan trọng, chỉ cần chúng ở trong cùng một chuỗi. Một bộ chèn simile phải thực hiện việc hoán vị các bit trong mỗi chuỗi để đưa được bộ mã hoá về cùng trạng thái như khi không sử dụng bộ chèn. Vì cả hai bộ mã hoá kết thúc ở cùng một trạng thái nên ta chỉ cần một "tail" là có thể đưa cả hai bộ mã hoá về trạng thái zero cùng một lúc.

**1.5.3.8. Bộ chèn khung:**

Nếu bộ chèn simile cần sử dụng thêm tail bit để lái cả hai bộ mã hoá đến cùng một trạng thái thì bộ chèn khung lại không cần tail bit. Mỗi một bộ RSC do tính hồi quy của nó có thể đặc trưng bằng một đa thức sinh chu kỳ L. Trong trường hợp này N bit thông tin sau khi được chèn sẽ được lưu hai lần trong bộ nhớ kích thước 2N tại những địa chỉ mà việc đọc chúng ra sau này bị ngăn cách bằng một khoảng thời gian bằng với số nguyên lần của L. Bằng cách này, nếu bộ mã hóa bắt đầu bằng một trạng thái zero thì sẽ kết thúc tại một trạng thái zero mà không cần thêm bất kỳ một tail bit nào.

**1.5.3.9. Bộ chèn tối ưu ( gần tối ưu):**

Bộ chèn tối ưu có thể được mô tả như bộ chèn cho ra các chuỗi mã hóa ngõ ra có trọng số thấp ít nhất. Bộ chèn này thiết kế dài dòng và phức tạp, thuật toán sau chỉ mô tả sơ lược việc thiết kế này :

1. Phát ra chèn ngẫu nhiên

2. Phát tất cả các chuỗi tin ngõ vào có thể

3. Đối với tất cả các chuỗi tin ngõ vào có thể mã hoá thành từ mã và xác định kết quả trọng số của từ mã để tìm được phân bố trọng số của từ mã

4. Xác định trọng số từ mã nhỏ nhất và số các từ mã với trọng số đó

Thuật toán này lặp lại với "số lần hợp lý" và giữ bộ chèn có trọng số từ mã nhỏ nhất là lớn nhất với số lượng các từ mã có trọng số đó thấp nhất.

**1.5.3.10. kết luận :**

Trên đây liệt kê một số các kiểu chèn được sử dụng trong mã TC. Với cùng một chuỗi tin ngõ vào, có thể đối với bộ chèn này sẽ cho ra các từ mã có trọng số cao

nhưng với bộ chèn kiểu khác lại cho ra các từ mã có trọng số thấp hơn. Trong thực tế để có BER thấp người ta không sử dụng thuần tuý một kiểu chèn mà người ta thiết kế một bộ chèn mới bằng cách kết hợp các ưu điểm của các bộ chèn khác nhau. Các cách thiết kế này làm tăng đáng kể chất lượng của mã TC đối với từng hệ thống cụ thể.

# Chương 2: Giải mã mã turbo

## 2.1. Giới thiệu chương:

Chương này sẽ trình bày hai thuật toán giải mã Turbo đó là :

- Thuật toán giải mã MAP
- Thuật toán giải mã SOVA
- So sánh chất lượng mã PCCC với các loại mã ra đời trước

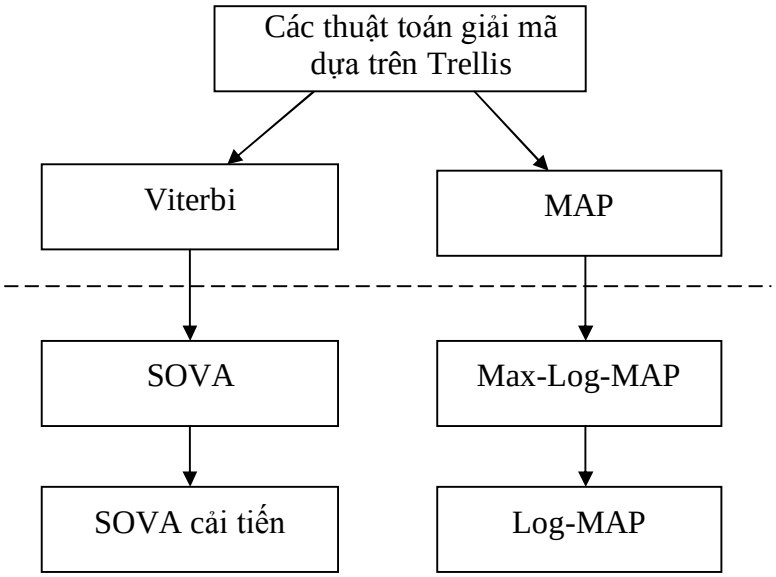
## 2.2. Tổng quan về các thuật toán giải mã:

Ngoài sự kết nối các bộ mã tích chập cùng việc sử dụng một thành phần đặc biệt là các bộ chèn, còn một thành p hần quan trọng khác trong chất lượng Turbo là qui trình giải mã mềm được thực hiện lặp đi lặp lại và độ phức tạo chỉ tăng tuyến tính theo kích thước khung. Mã PCCC có cấu trúc mã hoá kết nối song song tuy nhiên quá trình giải mã PCCC lại dựa trên sơ đồ giải mã kết nối nối tiếp. Mã Turbo sử dụng bộ giải mã kết nối nối tiếp vì sơ đồ kết nối nối tiếp có khả năng chia xẻ thông tin giữa các bộ giải mã kết nối, trong khi đó các bộ giải mã có sơ đồ kết nối song song chủ yếu giải mã độc lập nhau. Các thông tin này nhờ đặc tính mềm, được trao đổi, khai thác nhiều lần qua các vòng lặp sẽ làm tăng đáng kể chất lượng giải mã.

Trong khi thực hiện một vòng lặp giải mã các thông tin mềm được trao đổi giữa các bộ giải mã thành phần, Forney đã chứng minh được rằng ngõ ra mềm tối ưu cho bộ giải mã phải là xác suất a posteriori (APP) là xác suất của một bit nào đó được truyền dựa trên tín hiệu nhận được. Vì độ phức tạp của các mã TC chủ yếu là do bộ

giải mã lặp nên điều cần thiết trước nhất là tìm hiểu các thuật toán giải mã và tìm ra cách tốt nhất để giải mã mà không làm giảm chất lượng.

Phát triển các thuật toán giải mã hiệu quả là mối quan tâm hàng đầu khi cải tiến mã TC. **Hình 2.1** trình bày cái nhìn tổng quan về các họ thuật toán giải mã dựa trên sơ đồ trellis.




**Hình 2.1 :** Tổng quan các thuật toán giải mã

Họ thứ nhất là họ các thuật toán MAP còn gọi là thuật toán BCJR (Bahl-Cocke-Jelinek-Raviv, tên bốn người đã tìm ra thuật toán này). Thuật toán này liên quan đến các thuật toán giải mã khả năng xảy ra lớn nhất (ML) nhằm làm giảm tối đa xác suất lỗi bit. Họ này bao gồm các thuật toán symbol-by-symbol MAP, là phương pháp tối ưu để tính các thông tin APP, đây là thuật toán dạng tích, độ phức tạp rất cao. Trong họ này còn có hai loại thuật toán làm gần đúng thuật toán MAP để trở thành thuật toán dạng tổng độ phức tạp ít hơn mà chất lượng giải mã gần như tương đương là Log-MAP và phiên bản gần đúng của Log-MAP là Max-log-MAP. Một họ thuật toán giải mã khác là một họ thuật toán dựa trên việc sửa đổi thuật toán Viterbi (VA) có sử dụng thêm metric bổ sung vì VA truyền thống không tính các thông tin APP, metric bổ sung làm điều đó. Họ thuật toán giải mã này bao gồm thuật toán nổi tiếng là thuật toán Viterbi ngõ ra mềm (SOVA) và thuật toán ít được biết đến hơn là

thuật toán Viterbi ngõ ra liệt kê nối tiếp (SLVA). Ngoài hai họ thuật toán giải mã này còn có một số kỹ thuật giải mã lặp khác.

Tuy cùng là các thuật toán ngõ ra mềm dựa trên sơ đồ trellis nhưng khác với VA là một thuật toán giải mã trellis ML và giảm thiểu xác suất lỗi từ mã, thuật toán MAP lại nhắm tới giảm tối đa xác suất lỗi bit. MAP là một phương pháp tối ưu để ước đoán các trạng thái và ngõ ra của các quá trình Markov trong điều kiện nhiễu trắng. Tuy nhiên MAP ít khả năng được ứng dụng thực tế bởi các khó khăn về số học liên quan đến việc biểu diễn xác suất, các hàm phi tuyến cùng một số các phép nhân và cộng khi tính toán các giá trị này.

Log-MAP là một biến thể của MAP, chất lượng gần như tương đương mà không gặp trở ngại trong việc ứng dụng trong thực tế. Log-MAP được thực hiện hoàn toàn trong miền logarit, nhờ đó phép nhân chuyển thành phép cộng và ta có được một hàm tương đối dễ thực hiện hơn.

Max-Log-MAP và SOVA là thuật toán gần tối ưu dùng để giảm bớt độ phức tạp tính toán nhưng trong kênh nhiễu Gauss thì chất lượng hai loại này cũng không cao, đặc biệ trong vùng SNR thấp. Max -Log-MAP hầu như giống với Log-MAP chỉ có duy nhất một điểm khác là sử dụng một hàm đơn giản hơn rất nhiều. Các nghiên cứu cho thấy Max-Log-MAP làm giảm chất lượng khoảng 0.5 dB so với MAP/Log-MAP trong kênh nhiễu Gauss.

Các khác biệt trong việc thực hiện giữa các thuật toán giải mã này có thể giúp giải thích được sự khác biệt về chất lượng. Tại mỗi bước thứ k trong một trellis, MAP/Log-MAP chia tất cả các đường ra thành hai tập ; một tập các đường khi bit thông tin ngõ vào bằng 1 và một tập các đường khi bit thông tin ngõ vào bằng 0. MAP/Log-MAP sẽ tính tỉ số xác suất log (LLR) của hai tập này theo công thức. Ngược lại Max -Log-MAP sẽ tìm trong tất cả các đường để chọn các đường thích hợp, một đường có khả năng lớn nhất cho bit thông tin ngõ vào bằng 0. Ngõ ra mềm của Max-Log-MAP là LLR của hai đường này.

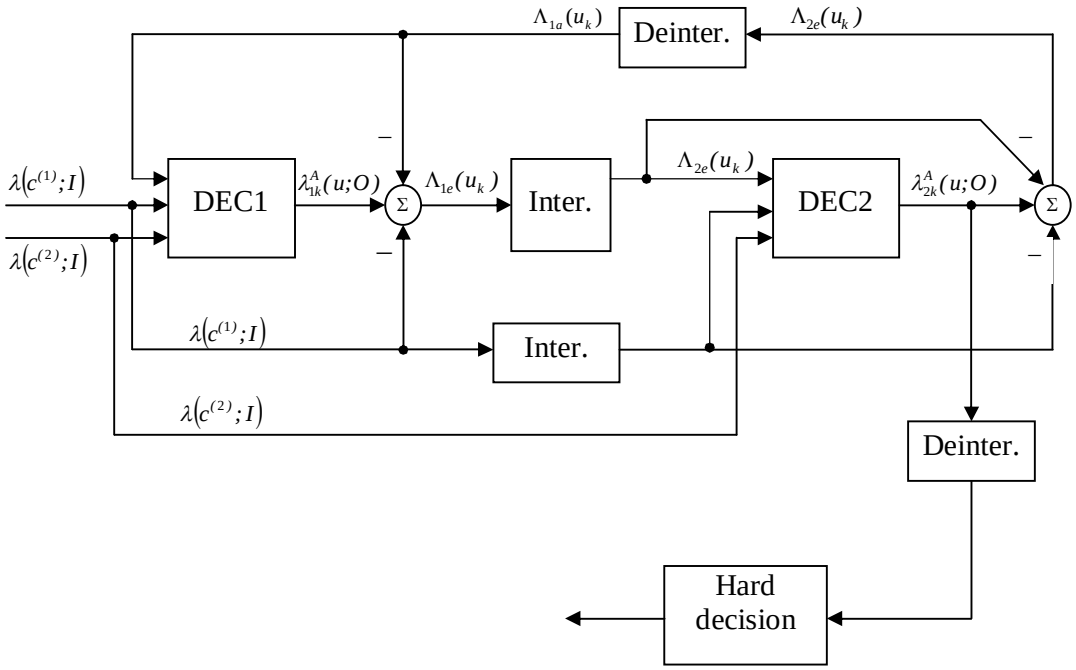
Còn SOVA thì bổ sung vào VA một số giá trị thực và lưu giữ . Thuật toán này chỉ tìm đường "tồn tại" và một đường cạch tranh với đường "tồn tại" đó. Về bản

chất, SOVA sử dụng cùng một loại metric và có quyết định cứng như Max-log-MAP. Mặc dù, SOVA luôn tìm đường có khả năng lớn nhất nhưng đường cạnh tranh tốt nhất có thể bị loại ra trước khi kết hợp với đường ML. Kết quả là ngõ ra mềm của SOVA có thể bị sai đường so với ngõ ra mềm của Max-Log-MAP và chất lượng của bộ giải mã lặp SOVA kém hơn Max -Log-MAP.

Mặc dù thuật toán MAP tốt hơn thuật toán SOVA nhưng nó có cấu trúc phần cứng và quá trình tính toán giải mã lại phức tạp hơn nhiều.

**2.3. Giải thuật MAP:**

Bộ giải mã là sự kết hợp của nhiều bộ giải mã (thường là hai bộ giải mã) và giải mã lặp (interatively). Phần lớn tập trung ở giải thuật Viterbi cung cấp giá trị ra mềm (soft output or reliability information) cho một bộ so sánh giá trị ra mềm được dùng để quyết định bit ngõ ra. Một giải thuật khác cũng được quan tâm là symbolby-*symbol Maximum A Posteriori (MAP) của Balh được công bố.*




**Hình 2.2:** Bộ giải mã lặp MAP

Giải thuật giải mã được thực hiện như sau:

1. Tách tín hiệu nhận ra thành 2 chuỗi tương ứng cho bộ giải mã 1 và bộ giả mã 2 .

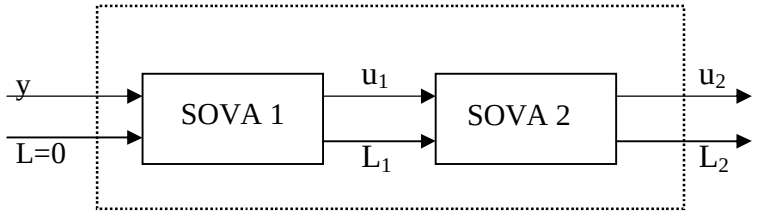
2. Ở vòng lặp đầu tiên ,thông tin *a priori* của bộ giải mã 1 được đưa về 0. Sau khi bộ giải mã 1 đưa ra được thông tin *extrinsic* thì sẽ được chèn và đưa tới bộ giải mã 2 đóng vai trò là thông tin *a priori* của bộ giải mã này. Bộ giải mã 2 sau khi đưa ra thông tin *extrinsic* thì vòng lặp kết thúc.Thông tin *extrinsic* của bộ giải mã thứ 2 sẽ được giải chèn và đưa về bộ giải mã 1 như là thông tin *a priori* .

3. Quá trình giải mã giải mã cứ lặp lại như vậy cho đến khi thực hiện đủ số lần lặp đã qui định .

4. Sau vòng lặp cuối cùng, giá trị ước đoán có được tính bằng cách giải chèn thông tin ở bộ giải mã thứ 2 và đưa ra quyết định cứng.

**2.4. Nguyên lý của bộ giải mã viterbi ngõ ra mềm:**

Đối với các mã tích chập thì thuật toán Viterbi cho ra chuỗi ngõ ra ML. Còn đối với các mã Turbo, chúng ta gặp hai trở ngại khi sử dụng các bộ giải mã Viterbi thông thường. Thứ nhất, bộ giải mã Viterbi bên trong cho ra một loạt lỗi bit làm giảm đi việc thực hiện của các bộ giải mã Viterbi bên ngoài. Thứ hai, bộ giải mã Viterbi bên trong cho ra các ngõ ra quyết định cứng làm ngăn chặn bộ giải mã Viterbi bên ngoài nhận được các lợi điểm của các quyết định mềm. Cả hai trở ngại này có thể được khắc phục và việc thực hiện giải mã có thể được cải tiến một cách đáng kể nếu các bộ giải mã Viterbi có thể cho ra các giá trị tin cậy. Các giá trị tin cậy này đi qua các bộ giải mã Viterbi tiếp sau đó và được xem như là một thông tin ưu tiên nhằm để cải tiến việc thực hiện giải mã. Bộ giải mã Viterbi bổ sung này được tham khảo như là bộ giải mã thuật toán Viterbi ngõ ra mềm (SOVA)

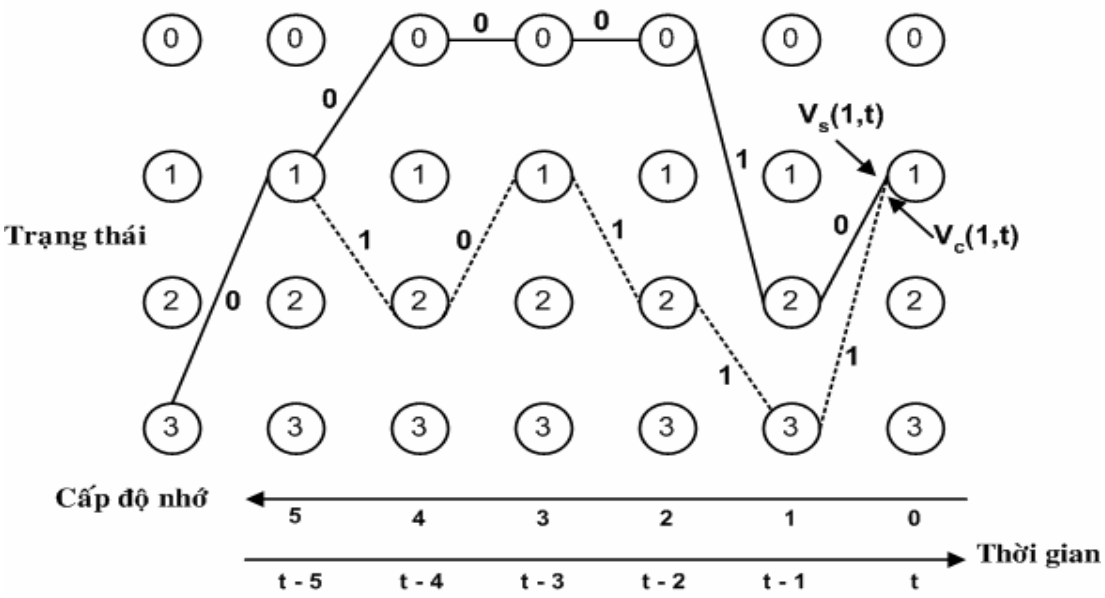



**Hình 2.3 Bộ giải mã SOVA kết nối**

Trong hình trên y biểu diễn các giá trị kênh nhận được, u biểu diễn các giá trị ngõ ra quyết định cứng, L biểu diễn các giá trị tin cậy liên kết.

**2.4.1. Độ tin cậy của bộ giải mã SOVA tổng quát:**

Độ tin cậy trong giải mã SOVA được tính toán từ biểu đồ trellis như **hình:2.4**




**Hình 2.4**: Các đường survivor và đường cạnh tranh để ước đoán độ tin cậy

Trong **Hình 2.4** trình bày biểu đồ trellis 4 trạng thái. Đường liền nét chỉ ra đường survivor (giả thiết ở đây là một phần của đường ML) và đường đứt nét chỉ ra đường cạnh tranh (xảy ra đồng thời) tại thời điểm t đối với trạng thái 1. Để đơn giản thì các đường survivor và cạnh tranh cho các nút khác không được vẽ ra. Nhãn $S_{1,t}$ biểu diễn trạng thái 1 tại thời điểm t. Cũng vậy, các {0,1} được viết trên mỗi đường chỉ ra quyết định nhị phân được ước đoán cho các đường. Một metric tích lũy $V_s(S_{1,t})$ gán cho đường survivor đối với mỗi nút và metric tích lũy $V_c(S_{1,t})$ gán cho đường cạnh tranh đối với mỗi nút. Thông tin cơ bản cho việc gán giá trị tin cậy L(t) đến đường survivor của nút $S_{1,t}$ là giá trị tuyệt đối của 2 metric tích lũy.

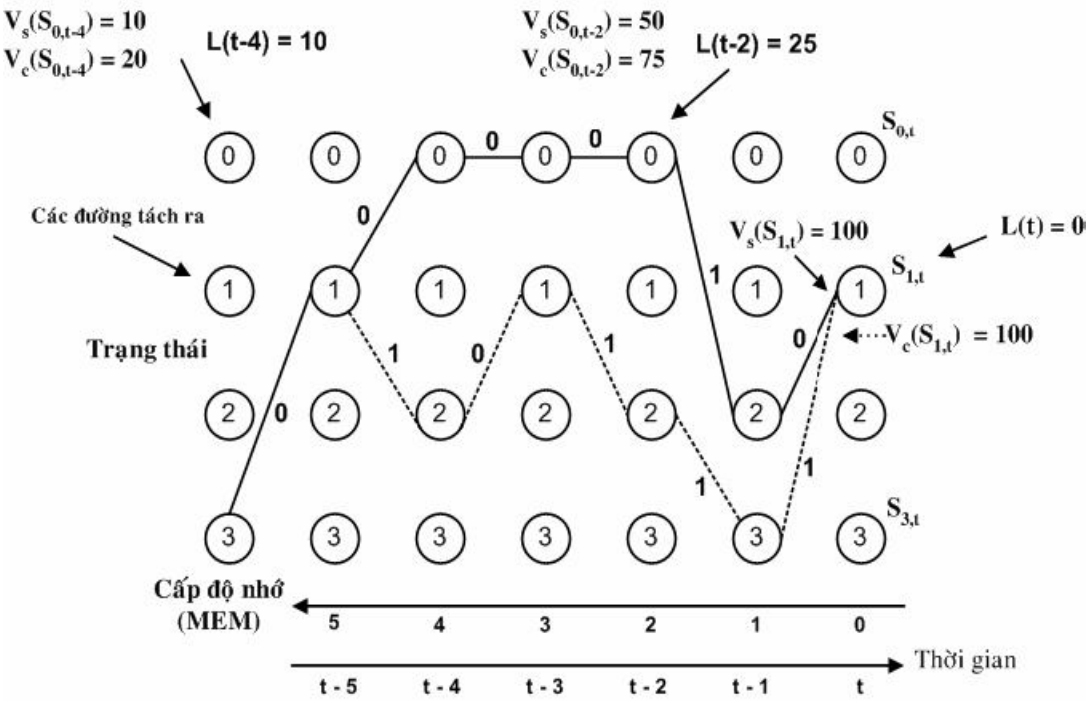
$$L(t) = |V_s(S_1,t) - V_c(S_1,t)| \qquad (2.1)$$

Giá trị này càng lớn thì đường survivor càng đáng tin cậy. Để tính toán độ tin cậy này, giả thiết metric tích lũy của survivor thì luôn luôn lớn hơn metric tích lũy của cạnh tranh. Để giảm độ phức tạp, các giá trị tin cậy chỉ cần được tính cho đường survivor ML và không cần thiết tính cho các đường survivor khác bởi vì chúng sẽ được bỏ qua sau này.

Để minh hoạ rõ hơn khái niệm độ tin cậy, hai ví dụ sau đây được đưa ra. Trong các ví dụ này, thuật toán Viterbi chọn đường survivor như là đường có metric tích lũy nhỏ nhất. Trong ví dụ đầu tiên, giả thiết tại nút S1,t có metric survivor tích lũy là Vs(S1,t) = 50 và metric cạnh tranh tích lũy là Vc(S1,t) = 100. Giá trị tin cậy liên kết đến việc chọn đường survivor này là L(t) = |50 - 100| = 50. Trong ví dụ thứ hai, giả thiết metric survivor tích lũy không đổi Vs(S1,t)= 50 và metric cạnh tranh tích lũy là *Vc(S1,t) = 75. Kết quả giá trị tin cậy là L(t) = |50 - 75| = 25. Mặc dù trong cả hai ví* dụ này, đường survivor có cùng metric tích lũy, nhưng giá trị tin cậy được liên kết với đường survivor thì khác nhau. Giá trị tin cậy trong ví dụ đầu tiên có nhiều tin tưởng hơn (gấp 2 lần) trong việc chọn đường survivor hơn là giá trị trong ví dụ thứ hai. **Hình 2.5** minh họa vấn đề sử dụng trị tuyệt đối giữa các metric survivor và cạnh tranh tích lũy như là phép đo độ tin cậy của quyết định.

Trong hình, các đường survivor và các đường cạnh tranh tại S1,t tách ra tại thời điểm t-5. Các đường survivor và các đường cạnh tranh cho ra các quyết định nhị phân ước đoán đối lập tại các thời điểm t, t - 2 và t - 4 như các chữ in đậm ở trong hình. Để minh họa, chúng ta giả thiết các metric tích lũy của survivor và cạnh tranh tại S1,t là bằng nhau, Vs(S1,t) = Vc(S1,t) = 100. Điều này có nghĩa là cả hai đường survivor và đường cạnh tranh có cùng xác suất là đường ML. Hơn nữa chúng ta giả thiết là metric tích lũy survivor thì tốt hơn metric tích lũy cạnh tranh tại thời điểm t - *2 và t - 4 được trình bày trong hình. Để giảm bớt độ phức tạp của hình vẽ, các* đường cạnh tranh này tại các thời điểm t - 2 và t - 4 không đưa ra. Từ giả thiết này, chúng ta thấy rằng giá trị tin cậy gán cho đường survivor tại thời điểm t là L(t) = 0, điều này có nghĩa là không có độ tin cậy liên kết với việc chọn đường survivor. Tại các thời điểm t - 2 và t - 4, các giá trị tin cậy gán cho đường survivor thì lớn hơn 0

(L(t-2) = 25 và L(t-4) = 10) nghĩa là kết quả các metric tích lũy "tốt hơn" cho đường survivor. Tuy nhiên, tại thời điểm t, đường cạnh tranh cũng có thể là đường survivor bởi vì chúng có cùng metric. Vì vậy có thể có các quyết định nhị phân được ước đoán trái ngược nhau tại các thời điểm t, t- 2, t - 4 mà không làm giảm các giá trị tin cậy liên kết suốt dọc theo đường survivor.




**Hình 2.5** *:* Ví dụ trình bày việc gán độ tin cậy bằng cách sử dụng các giá trị metric trực tiếp

Để cải tiến các giá trị tin cậy của đường survivor, một phép tính truy ngược để cập nhật các giá trị tin cậy được giả thiết. Thủ tục 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), lưu $L(t) = | V_s(S_{1,t}) - V_c(S_{1,t})|$.

* Nếu có nhiều hơn một 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)

* Khởi tạo giá trị tin cậy $S_{k,t}$ bằng $+\infty$ (tin cậy nhất)

* 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 hai con đường là khác nhau.

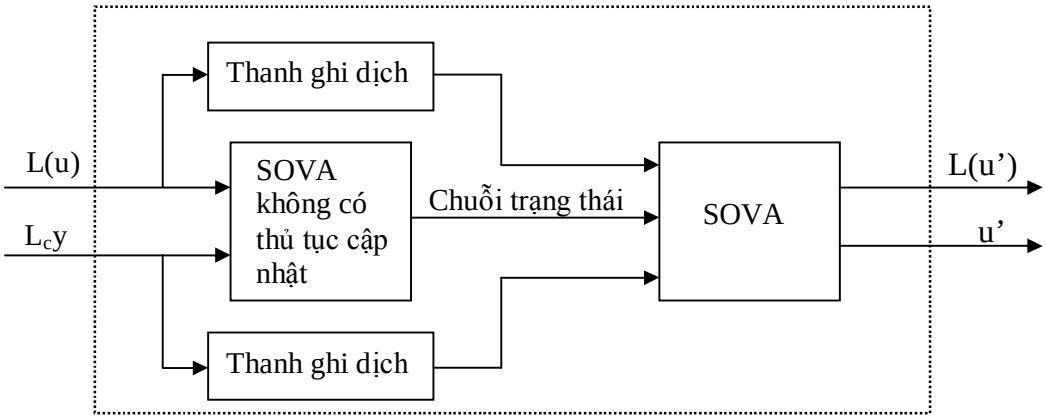
* Cập nhật các giá trị tin cậy tại các MEM này với thủ tục như sau :

+ Tìm MEM thấp nhất lớn hơn 0, coi như là $MEM_{low}$ mà giá trị tin cậy của nó không được cập nhật.

+ Cập nhật giá trị tin cậy của $MEM_{low}$ $L(t\text{-}MEM_{low})$ bằng cách gán giá trị tin cậy thấp nhất giữa MEM = 0 và MEM = $MEM_{low}$

### 2.4.2. Sơ đồ khối của bộ giải mã SOVA:

Bộ giải mã SOVA có thể được thực hiện theo nhiều cách khác nhau. Nhưng có lẽ theo khuynh hướng tính toán thì dễ dàng thực hiện bộ giải mã SOVA cho các mã có chiều dài bắt buộc K lớn và kích cỡ khung dài bởi vì sự cần thiết cập nhật tất cả các đường survivor. Do thủ tục cập nhật chỉ có ý nghĩa cho đường ML, nên việc thực hiện của bộ giải mã SOVA chỉ thực hiện thủ tục cập nhật đối với đường ML được trình bày trong **hình 2.6**

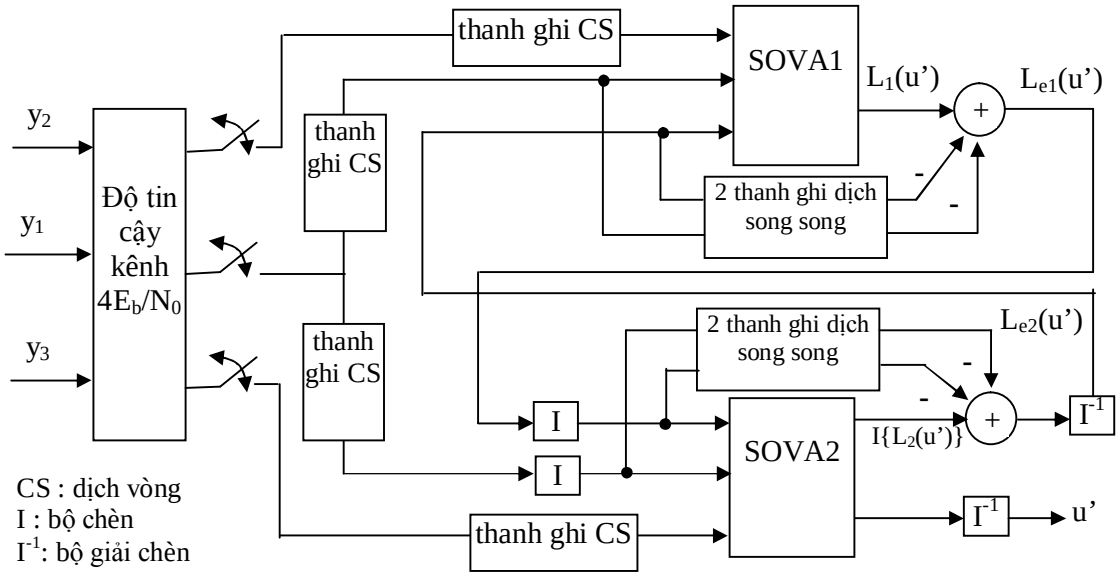



**Hình 2.6: Sơ đồ khối bộ giải mã SOVA**

Bộ giải mã SOVA lấy ngõ vào là L(u) và Lcy, là giá trị tin cậy và giá trị nhận được đã qua cân bằng tương ứng, và cho ra u' và L(u'), tương ứng là các quyết định bit ước đoán và các thông tin a posteriori L(u'). Việc thực hiện bộ giải mã SOVA

này bao gồm hai bộ giải mã SOVA riêng biệt. Bộ giải mã SOVA đầu tiên chỉ tính các metric của đường ML và không tính (giữ lại) các giá trị tin cậy. Các thanh ghi dịch được sử dụng để đệm cho các ngõ vào trong khi bộ giải mã SOVA đầu tiên đang xử lý đường ML. Bộ giải mã SOVA thứ hai (có thông tin đường ML) tính lại đường ML và cũng tính và cập nhật các giá trị tin cậy. Ta thấy rằng phương pháp thực hiện này làm giảm độ phức tạp trong tiến trình cập nhật. Thay vì truy ngược và cập nhật 2m đường survivor, thì chỉ có đường ML cần được xử lý.

Một sơ đồ chi tiết của một bộ giải mã SOVA lặp được trình bày ở **Hình 2.7**




**Hình 2.7: Bộ giải mã SOVA lặp**

Bộ giải mã xử lý các bit kênh nhận được trên một khung cơ bản. Như được trình bày trong **Hình 2.7**, các bit kênh nhận được tách thành dòng bit hệ thống $y_1$ và *2* dòng bit parity $y_2$ và $y_3$ từ các bộ mã hóa 1 và 2 tương ứng. Các bit này được cân bằng bởi giá trị tin cậy kênh và được lấy ra qua các thanh ghi CS. Các thanh ghi trình bày trong hình được sử dụng như các bộ đệm để lưu trữ các chuỗi cho đến khi chúng ta cần. Các khóa chuyển được đặt ở vị trí mở nhằm ngăn ngừa các bit từ các khung kế tiếp đợi xử lý cho đến khi khung hiện hành được xử lý xong.

Bộ giải mã thành phần SOVA cho ra thông tin *a posteriori* L($u_t$') và bit được ước đoán $u_t$' (ở thời điểm *t*). Thông tin *a posteriori* L($u_t$') được phân tích thành 3 số hạng

$$L(u'_t)=L(u_t) + L_c y_{t,1} + L_e(u_t') \quad (2.2)$$

$L(u_t)$ là giá trị *a priori* và được sinh ra bởi bộ giải mã thành phần SOVA trước đó.

$L_c y_{t,1}$ là giá trị kênh hệ thống nhận được đã qua cân bằng.

$L_e(u_t')$ là giá trị *extrinsic* được sinh ra bởi bộ giải mã thành phần SOVA hiện tại. Tin tức đi xuyên qua giữa các bộ giải mã thành phần SOVA là giá trị *extrinsic*.

$$L_e(u_t')=L(u'_t) - L_c y_{t,1} - L(u_t) \quad (2.3)$$

Giá trị *a priori* $L(u_t)$ được trừ đi từ số bị trừ là thông tin *a osteriori* L($u_t$') để ngăn ngừa tin tức đi ngược lại bộ giải mã mà từ đó sinh ra nó. Cũng vậy, giá trị kênh hệ thống nhận được đã qua cân bằng $L_c y_{t,1}$ được trừ đi nhằm để xóa tin tức "thông thường" trong các bộ giải mã thành phần SOVA. **Hình 2.7** trình bày bộ giải mã mã PCCC là sự kết nối theo thứ tự vòng kín của các bộ giải mã thành phần SOVA. Trong sơ đồ giải mã vòng kín này, mỗi một bộ giải mã thành phần SOVA ước đoán chuỗi tin bằng cách sử dụng dòng bit parity đã qua cân bằng. Hơn nữa, bộ giải mã PCCC thực hiện giải mã lặp nhằm cho ra các ước đoán *a priori* /độ tin cậy đáng tin tưởng hơn từ 2 dòng bit parity đã qua cân bằng khác nhau, với hy vọng thực hiện giải mã tốt hơn. Thuật toán mã Turbo lặp với lần lặp thứ n như sau:

1. Bộ giải mã SOVA1 có ngõ vào là chuỗi $L_c y_1$(hệ thống), $L_c y_2$ (parity), và cho ra chuỗi $L_{e2}$(u'). Đối với lần lặp đầu tin, chuỗi $L_{e2}(u')=0$ bởi vì không có giá trị *a priori* (không có giá trị *extrinsic* từ SOVA2). Thông tin *extrinsic* từ SOVA1 được tính bằng

$$L_{e1}(u')= L_1(u') - L_{e2}(u')- L_c y_1 \quad (2.4)$$

trong đó

1. Lc = $4\frac{E_b}{N_o \times rate}$

2. Các chuỗi $L_c y_1$ và $L_{e1}(u')$ được chèn là $I\{(L_c y_1)\}$ và $I\{L_{e1}(u')\}$.

3. Bộ giải mã SOVA2 có ngõ vào là các chuỗi $L_c y_1$ (hệ thống), và $I(L_c y_3)$(parity đã được chèn ở bộ giải mã) và $I\{L_{e1}(u')\}$ (thông tin *a priori*) và cho ra các chuỗi $I\{L_2(u')\}$ và $I\{u'\}$ .

4. Thông tin *extrinsic* từ SOVA2 được lấy là:

$$I\{L_{e2}(u')\} = I\{L_2(u')\} - I\{L_{e1}(u')\} - I(L_c y_1)$$

Các chuỗi $I\{L_{e2}(u')\}$ và$I\{u'\}$ được giải chèn và là $L_{e2}(u')$ và u'.

5. $L_{e2}(u')$ được hồi tiếp về SOVA1 như là thông tin *a priori* cho lần lặp kế tiếp và u' là ngõ ra của các bit được ước đoán cho lần lặp thứ *n*.

## 2.5. Sự khác nhau giữ mã chập và mã PCCC:

Ta có sự khác nhau như **bảng 2.1** dưới:

| Tiêu chuẩn | Mã tích chập | Mã PCCC |
|---|---|---|
| Chiều dài bắt buộc lớn hơn | Tốt | Xấu |
| Khoảng cách tự do lớn hơn | Tốt | Không khác |
| Tốc độ mã hoá thấp hơn | Không khác | Tốt |
| Các bộ mã hoá đệ quy | Không khác | Tốt |
| Các bộ giải mã ngõ ra mềm | Không khác | Tốt |

**Bảng 2.1 So sánh mã chập và mã PCCC**

## 2.6. So sánh chất lượng của các hệ thống mã hóa:

Tỉ lệ lỗi bit (BER) và tỉ số tín hiệu/nhiễu (SNR) của quá trình truyền dẫn xác định chất lượng của kênh truyền. Tuy nhiên, sự khác nhau là bao nhiêu công suất (SNR) thì cần thiết để thực hiện BER thấp. Tỉ lệ lỗi bit là xác suất của bất cứ bit nào bị lỗi trong quá trình truyền. Tỉ số tín hiệu/nhiễu ( $E_b/N_o$) là tỉ số công suất kênh/ công suất nhiễu. BER càng thấp thì cáng tốt bởi vì có nghĩa là có một vài lỗi trong dữ liệu cuối cùng SNR càng thấp thì càng tốt bởi vì có nghĩa là ít công suất cần thiết để truyền tín hiệu.

Như trên đã giới thiệu mã PCCC RSC là mã có thực tế tốt nhất bởi vì nó có thể thực hiện SNR thấp tại BER thấp và gần với giới hạn Shannon – lý thuyết tối đa của thực hiện kênh. Độ lợi mã xác định độ thực hiện mã hóa như thế nào. Độ lợi mã là sự khác nhau trong SNR giữa 1 kênh được mã hóa và 1 kênh không được mã hóa. Độ lợi mã có thể được xác định bằng cách đo khoảng cách giữa giá trị SNR của bất kỳ một trong những kênh được mã hóa nào và kênh không được mã hóa tại BER cho trước. Ví dụ độ lợi mã đối với mã PCCC RSC có tốc độ 1/2 tại BER $10^{-5}$ là khoảng 8,5 dB. Điều này có nghĩa là tín hiệu được mã hóa PCCC có thể hoặc là được nhận 2,65 lần xa hơn tín hiệu không được mã hóa (ở cùng một công suất phát), hoặc là nó chỉ cần 1/7 công suất phát (cho cùng một khoảng cách).

### 2.7. Kết luận chương:

Qua chương này chúng ta biết được rõ hơn về mã turbo, cũng như những ưu điểm của nó so với các loại mã khác. Nên nó được ứng dụng vào các hệ thống thông tin yêu cầu chất lượng, tốc độ cao.

## Chương 3: ứng dụng mã turbo trong hệ thông tin di động cdma2000

### 3.1. Giới thiệu chương:

Trong lĩnh vực viễn thông thì hai hệ thống gây nhiều khó khăn nhất là truyền thông không dây (wireless communication) và truyền thông đa phương tiện (multimedia communication -MMC) do một số điểm đặc thù của hai loại hệ thống này gây nhiều khó khăn cho việc truyền thông. Mã TC ra đời đã thúc đẩy một quá trình tìm tòi, phát triển mới nhờ những đặc tính ưu việt của nó. chương này trình bày các ứng dụng chung của mã Turbo trong hệ thống truyền thông và trình bày chi tiết ứng dụng trong hệ thống thông tin di động cdma2000.

### 3.2. Các ứng dụng truyền thông đa phương tiện:

Ứng dụng trong truyền thông đa phương tiện là đề tài mới được nghiên cứu gần đây. Vì thế có một số nét chính về các vấn đề gặp phải và một số đề nghị khi ứng dụng TC trong truyền thông đa phương tiện.

### 3.2.1. Các hạn chế khi ứng dụng TC vào hệ thống truyền thông đa phương tiện:

Các ứng dụng MMC gặp phải các ràng buộc sau đây :

#### 3.2.1.1. Tính thời gian thực

Một hạn chế quan trọng nhất của các ứng dụng MMC là thời gian eo hẹp. Ví dụ như xét một ứng dụng MMC là Video-On-Demand (VOD), máy chủ VOD truyền dữ liệu phim đến các khách hàng. Mỗi khung dữ liệu có thể là thông tin về một khung hình của bộ phim. Nếu các dữ liệu phim đến chậm thì khách hàng sẽ có cảm giác chất lượng phim không tốt, phim không được chiếu trơn tru. Từ đây ta có thể thấy dữ liệu multimedia có bản chất thời gian thực, sự chậm trễ của dữ liệu sẽ làm mất giá trị của thông tin. Vấn đề thời gian chính là một rào cản lớn trong các ứng dụng MMC.

Trong khi TC cần phải có một cấu trúc giải mã lặp để tăng chất lượng thì tính thời gian thực quả là một thách thức khi phải giải quyết mâu thuẫn giữa thời gian đáp ứng và tỉ lệ lỗi bit (BER). Đặc tính thời gian thực này cho thấy các mã TC ứng dụng trong MMC không thể có số vòng lặp lớn.

#### 3.2.1.2. Khối lượng dữ liệu lớn:

Một đặc tính khác của các ứng dụng MMC là các khối dữ liệu lớn. Chỉ cần một hình ảnh trong bộ phim cũng cần tới cỡ hàng Megabit để biểu diễn. Cộng với đặc tính thời gian thực thì một số lượng lớn các dữ liệu sẽ phải được xử lý trong một khoảng thời gian giới hạn và rất ngắn, nếu không hệ thống sẽ gây lỗi. Kết quả là yêu cầu đối với các bộ mã hóa và giải mã TC rất cao.

#### 3.2.1.3. Băng thông giới hạn:

Băng thông là vấn đề luôn được quan tâm hàng đầu, nhất là trong các ứng dụng thực tiễn trong thời gian gần đây vì lượng thông tin con người mong muốn được truyền tải ngày càng lớn mà một tài nguyên quốc gia như băng thông không thể tăng. Băng thông sử dụng trong các ứng dụng MMC rất lớn (ví dụ như VOD thường sử dụng ATM), tuy nhiên do khối lượng dữ liệu cần truyền lớn nên băng thông lớn

(so với các ứng dụng trong hệ thống khác) vẫn trở thành một giới hạn cho các ứng dụng MMC. Kết quả là các mã tốc độ thấp sẽ không hiệu quả.

**3.2.1.4. Tìm hiểu các đặc tính của kênh truyền:**

Các đặc tính của các kênh truyền trong MMC đơn giản và bất biến hơn nhiều so với môi trường không dây. Vì vậy ta có thể tìm hiểu được các đặc tính của kênh truyền và đưa ra các giải pháp thích hợp cho từng hệ thống. Một phương pháp để tìm hiểu các đặc tính của kênh truyền là dùng mạng Bayes. Các phương pháp thực hiện có thể tóm lược như sau :

* Dữ liệu được truyền qua kênh truyền và sử dụng nhiều loại mô hình mã TC với các thông số khác nhau.

* Ghi lại các giá trị BER

* Thành lập một mạng Bayes với các độ chính xác của mã kết quả và các yếu tố ảnh hưởng là các nút mạng. Một đường nối trực tiếp sẽ đi từ các yếu tố đến các độ chính xác của mã kết quả.

* Sử dụng các giá trị BER ghi nhận để thử cho mạng này.

* Tìm ra một tập hợp các thông số để tối ưu hóa các sự điều chỉnh

**3.2.2. Các đề xuất khi ứng dụng TC vào truyền thông đa phương tiện:**

**3.2.2.1.Kích thước khung lớn:**

Như đã đề cập ở trên, một đặc tính quan trọng của ứng dụng MCC là khối dữ liệu lớn. Từ đây gợi ra ý tưởng sử dụng kích thước khung lớn cho mã TC. Kích thước khung lớn đồng nghĩa với kích thước bộ chèn lớn và sẽ làm tăng đáng kể chất lượng của mã TC.

Với một băng thông lớn như của MMC thì một khối lượng dữ liệu lớn có thể truyền với một độ trễ chấp nhận được. Với kích thước khung lớn này độ lợi mã của TC có thể tăng bằng các cách sau :

* Giảm BER của kênh truyền

* Tăng thời gian đáp ứng bằng cách giảm số lần lặp giải mã hay sử dụng một số cải tiến giải mã trình bày dưới đây.

**3.2.2.2.Cải tiến quá trình giải mã:**

**3.2.2.2.1 Giải mã động:**

Phương pháp giải mã động gói gọn trong hai điểm sau :

* Đặt một ngưỡng vòng lặp, tức là số lần lặp tối đa cho một khung.

* Số vòng lặp thực sự để giải mã một khung sẽ nhỏ hơn hay bằng giá trị ngưỡng này và phụ thuộc vào kết quả giải mã. Điều kiện để ngưng quá trình giải mã là khung đã hết lỗi. Trong quá trình giải mã, kết quả giá trị ước lượng của vòng lặp giải mã trước được lưu lại và so sánh với kết quả của vòng lặp giải mã kế tiếp. Nếu hai kết quả giống nhau thì hết lỗi và tiếp tục giải mã cho khung kế tiếp.

Ý tưởng ở đây là một số khung chỉ cần số vòng lặp rất ít (chỉ khoảng 2 hay 3 vòng) đã loại bỏ hoàn toàn lỗi sai, trong khi một số khung khác rất nhiều lỗi thì cần số vòng lặp giải mã nhiều hơn để đạt được chất lượng cao hơn. Vì thế số vòng lặp thay đổi sẽ ảnh hưởng trực tiếp đến việc giảm độ trễ và có thể còn làm tăng chất lượng. Ví dụ như một hệ thống sử dụng số lần lặp giải mã cố định là 10. Khi sử dụng hệ thống này với phương pháp giải mã động có số vòng lặp tối đa là 15 thì số vòng lặp giải mã trung bình sẽ giảm rất nhiều, chỉ khoảng 5 -7 vòng. Như vậy ta đã tiết kiệm được rất nhiều thời gian, tăng thời gian đáp ứng của hệ thống. Thậm chí có một số khung nhiều lỗi sai thì giải mã lặp đến 15 vòng có thể sẽ cho chất lượng cao hơn chỉ lặp 10 vòng cố định.

**3.2.2.2.2 Giải mã ưu tiên:**

Khối dữ liệu được truyền ngoài đặc tính là có số lượng bit lớn còn có một số đặc tính khác như :

* Dữ liệu nhận không cần chính xác 100%. Ví dụ như trong VOD, nếu một số phần nào đó của các khung nhận bị lỗi thì có thể gây ra một số suy giảm chất lượng trên một vài phần nào đó trong hình ảnh bộ phim. Nhưng nếu những sự suy giảm này khá nhỏ thì mắt người cũng khó nhận biết hoặc dễ dàng chấp nhận. Điều đó có nghĩa là MMC có thể chấp nhận một mức lỗi nhất định.

* Các dữ liệu truyền có tầm quan trọng khác nhau. Cũng xét ví dụ trên, nếu các lỗi xảy ra trong ở vùng trung tâm của hình ảnh thì khách hàng có thể phát hiện dễ dàng. Nhưng nếu các lỗi gây sự suy giảm chất lượng ở các vùng lân cận biên của

hình ảnh thì khó gây sự chú ý hơn. Điều đó có nghĩa là các dữ liệu có tầm quan trọng thấp sẽ chấp nhận mức lỗi cao hơn.

Các đặc tính này làm nảy sinh thêm một ý tưởng là giải mã theo mức ưu tiên. Các ứng dụng MMC sẽ thêm các thông tin về độ ưu tiên vào trong khung tùy theo tầm quan trọng của khung. Sau khi nhận được chuỗi tin từ kênh truyền, bộ giải mã sẽ giải mã tìm các từ mã. Sau vòng lặp đầu tiên bộ giải mã có thể nhận được thông tin về mức độ ưu tiên của khung và sẽ quyết định số vòng lặp (hay phương pháp lặp) phù hợp với khung.

Theo mô hình giải mã này, lượng thời gian tiết kiệm được từ các khung có độ ưu tiên thấp sẽ được dùng để :

* Giảm BER của các khung có độ ưu tiên cao

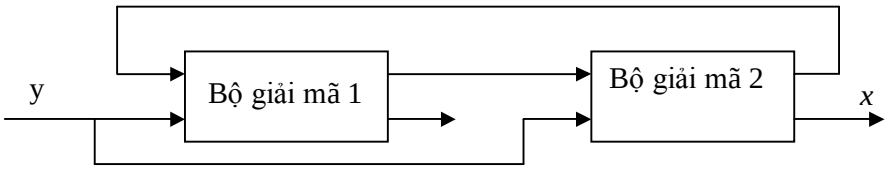
* Tăng tốc độ đáp ứng của hệ thống nhờ giảm được số vòng lặp cho các khung có độ ưu tiên thấp.

Mô hình này không làm tăng chất lượng trung bình của hệ thống. Tỉ số BER có thể lớn hơn hay nhỏ hơn so với các phương pháp giải mã khác nhưng hiệu quả thực tế thì hơn hẳn. Ví dụ như trong trường hợp cụ thể trên thì hình ảnh sẽ được khách hàng đánh giá là tốt hơn.

**3.2.2.2.3 Cấu trúc giải mã Pipeline:**

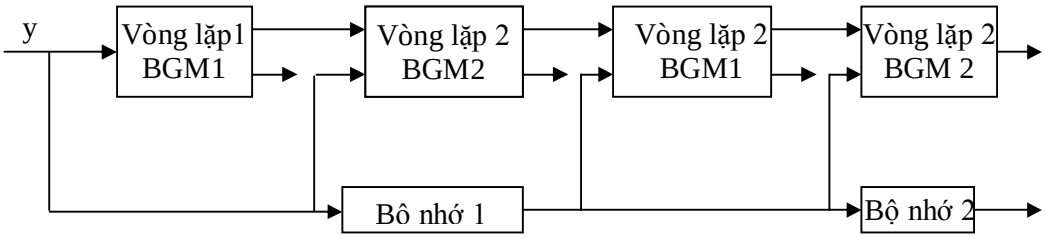
Đây là một phương pháp giải mã khác cũng với mục đích có thể làm giảm tối đa độ trễ của hệ thống do các vòng lặp giải mã gây ra.

Sơ đồ đơn giản hóa của một bộ giải mã lặp như sau:




Bộ giải mã thông thường này sẽ lặp lại quá trình giải mã n lần cho mỗi từmã y để tìm được ước đoán gần đúng với từ mã x nhất.

Cấu trúc đơn giản hóa của bộ giải mã pipeline như sau




BGM : Bộ giải mã

Bộ chuỗi các bộ giải mã được sử dụng cho mỗi vòng lặp. Bộ nhớ dùng để lưu trữ thông tin hệ thống cho các vòng lặp tương ứng. Theo mô hình trên, độ trễ như sau :

* n vòng lặp cho từ mã đầu tiên

* 1 vòng lặp cho mỗi từ mã tiếp theo

**3.3. Các ứng dụng truyền thông không dây:**

Truyền thông không dây ngày càng trở nên thông dụng nhờ những ưu điểm như số lượng dịch vụ lớn, kích thước thoại nhỏ và giá cả tương đối chấp nhận được so với những lợi ích của nó. Không như các tiến bộ vượt bậc về kỹ thuật trong kích thước thoại và số lượng dịch vụ, các giao thức hiện nay như GSM, CDMA vẫn sử dụng các mô hình đơn giản như các mã tích chập. Với tốc độ phát triển như hiện nay, các thành phần này sẽ được thay thế bằng loại mã chất lượng hơn như mã TC mà gần đây nhất là các hệ thống thông tin thế hệ thứ ba (cdma2000).

Đặc biệt trong truyền thông không dây còn phải kể đến truyền thông vệ tinh hay thám hiểm vũ trụ. Hiện nay đã có xu hướng gia tăng cả về số lượng lẫn chất lượng các loại hình thông tin vệ tinh cũng như thông tin vũ trụ do khoa học kỹ thuật đã tiến bộ ở rất nhiều nước. Các hệ thống tiêu biểu cho thông tin vệ tinh là hệ thống định vị toàn cầu (GPS), hệ thống thông tin địa lý (GIS), hệ thống truyền hình qua vệ tinh.

**3.3.1. Các hạn chế khi ứng dụng TC trong truyền thông không dây:**

**3.3.1.1.Kênh truyền:**

Đối với nhiều kênh truyền thì mô hình kênh AWGN với nhiễu tĩnh rất thích hợp. Tuy nhiên, trong môi trường không dây thì thường không tĩnh do có fading của các tín hiệu truyền. Fading là hậu quả bản chất vật lý của kênh truyền với độ tăng biên độ là một quá trình ngẫu nhiên biểu diễn bởi một hàm mật độ xác suất và một hàm tự tương quan.

Trong kênh AWGN, các bit chỉ bị tác động bởi nhiễu :

*$Y_k = ax_k + n_k$ với $n_k$ là nhiễu và a = 1 đối với kênh AWGN*

Trong fading Rayleigh, các từ mã bị tác động bởi cả nhiễu và fading biến đổi theo thời gian trong kênh vô tuyến di động.

*$y_k = ax_k + n_k$ với $n_k$ là nhiễu và a là một biến ngẫu nhiên của phân bố fading* Rayleigh.

Phân bố fading Rayleigh thường được sử dụng để mô tả bản chất thay đổi theo thời gian theo thống kê của đường bao nhận được của một tín hiệu fading phẳng, hay đường bao của một thành phần riêng lẻ trong hệ thống đa đường. Phân bố Rayleigh là một hàm mật độ xác suất cho bởi :

$P(r) = \frac{r}{\sigma^2} e^{-\frac{r^2}{\sigma^2}}$ với r<0

P(r) = 0 với r ≥ 0

Kênh truyền trong truyền thông không dây có mức nhiễu cao hơn ở môi trường truyền dây. Vì thế các mã kênh phải có đủ khả năng đương đầu với mức nhiễu lớn. Đặc biệt nếu dùng trong công tác nghiên cứu vũ trụ thì mức nhiễu còn cao hơn nữa.

Để triệt fading thì còn có nhiều cách khác nhau ví dụ như trải phổ. Khi đã triệt được một phần fading và sử dụng thêm mã TC nữa thì chất lượng đạt được sẽ rất cao.

Ngoài ra, môi trường truyền còn luôn luôn biến đổi. Ví dụ như một thuê bao điện thoại di động có thể vừa đàm thoại vừa di chuyển, môi trường truyền xung quanh cũng biến đổi, thông số môi trường cũng thay đổi. Chính vì sự bất ổn định của kênh truyền mà việc tìm được một loại mã thích hợp là một việc rất khó khăn. Và đây chính là lĩnh vực ứng dụng chủ yếu của TC nhờ các đặc tính ưu việt.

**3.3.1.2. Hạn chế về thời gian:**

Cũng như các ứng dụng thời gian thực khác, truyền thông không dây cũng có những yêu cầu về thời gian rất khắt khe. Nhất là các thông tin thoại yêu cầu phải đáp ứng nhanh. Thông tin thoại mà đáp ứng chậm sẽ trở nên vô giá trị.

**3.3.1.3. Kích thước khung nhỏ:**

Trong truyền thông không dây thì kích thước khung truyền không được lớn vì:

* Kênh truyền không tin cậy, nếu truyền khung lớn thì tỉ lệ lỗi trong khung sẽ cao hơn. Nếu khung bị mất hay không thể khôi phục thì dữ liệu tại đầu nhận sẽ bị mất.

* Do đặc tính thời gian thực nên không chấp nhận độ trễ lớn khi truyền một khung có kích thước lớn.

Như vậy, với kích thước khung nhỏ thì không tận dụng được các đặc tính ưu việt của TC.

**3.3.1.4. Băng thông giới hạn:**

Truyền thông không dây chỉ sử dụng một khoảng phổ tần số đã được phân, mỗi công ty điện thoại di động lại chỉ được phân cho một khu vực trong khoảng này để cung cấp dịch vụ cho khách hàng. Như vậy băng thông rất hạn chế có nghĩa là mô hình mã hóa phải có càng ít bit redundant càng tốt, tức là đòi hỏi tốc độ mã cao.

**3.4. Mã hóa turbo trong cdma2000:**

Bộ mã hóa turbo thực hiện mã hóa số liệu, chỉ thị chất lượng khung (CRC) và hai bit dành trước cho mã turbo và cộng chuỗi đuôi mã hóa đầu ra. Nếu tổng các bit số liệu, các bit chất lượng khung và các bit dành trước là $N_{turbo}$, thì bộ mã hóa tạo ra $N_{turbo}/R$ các ký hiệu số liệu cùng với 6/R các ký hiệu đuôi ở đầu ra, trong đó R là tỷ lệ mã bằng 1/2, 1/3 hay 1/4. Bộ mã hóa turbo sử dụng hai bộ mã hóa tích chập hệ thống, đệ quy mắc song song kết hợp với bộ chèn, trong đó bộ chèn đứng trước bộ mã tích chập thứ hai, hai mã tích chập đệ quy này được gọi các mã thành phần của mã turbo. Các đầu ra của các bộ mã hóa thành phần được trích bỏ và được lặp để đạt được $(N_{turbo} +6)/R$ các ký hiệu ra

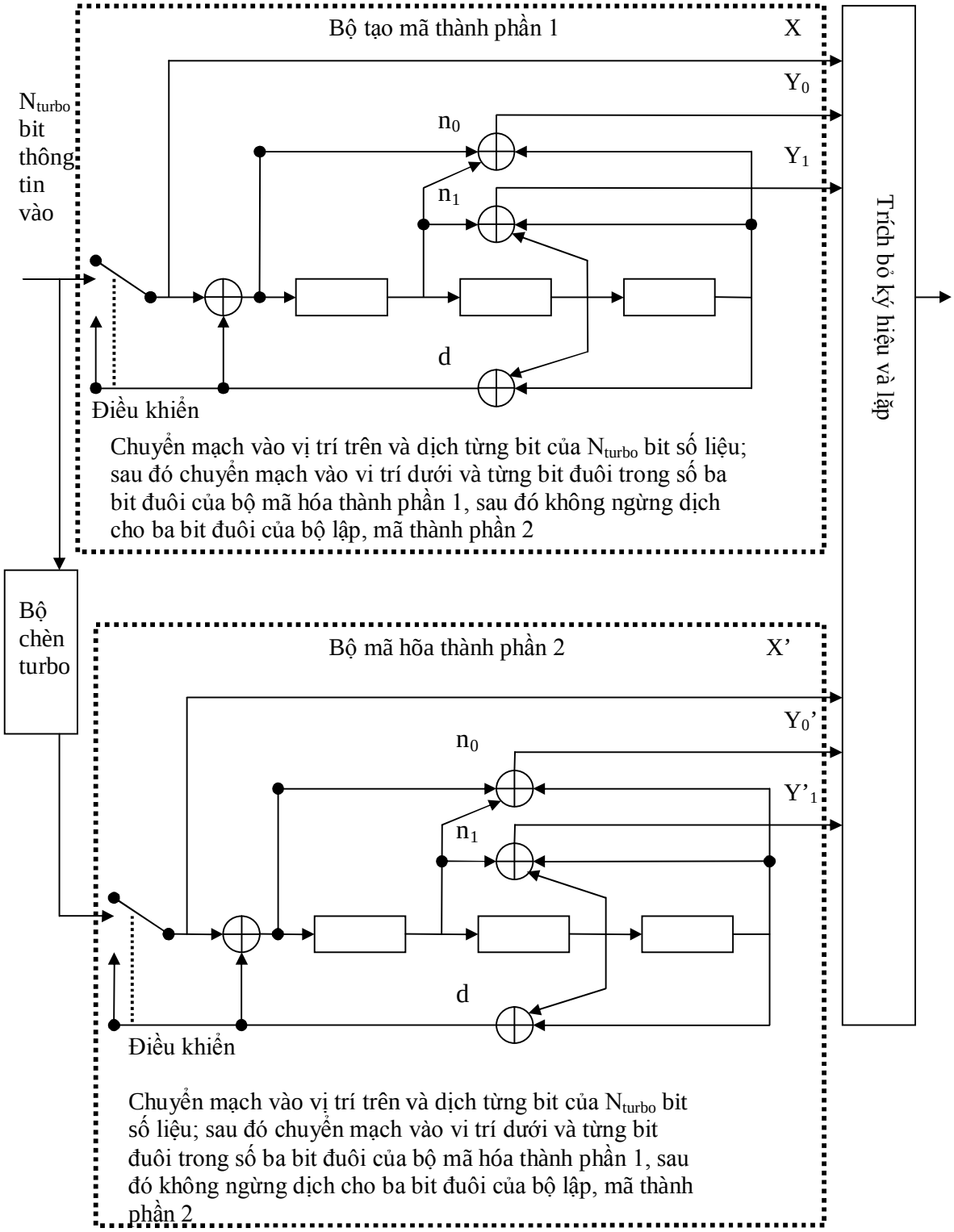
**3.4.1 Các bộ mã hóa turbo tỷ lệ 1/2, 1/3, 1/4 :**

Một mã thành phần chung được sử dụng cho các mã turbo tỷ lệ 1/2, 1/3, và 1/4. Hàm truyền đạt của mã này có dạng sau:

$$G(D)=\left[1 \frac{n_0(D)}{d(D)} \frac{n_1(D)}{d(D)}\right] \quad (3.1)$$

Trong đó: $d(D) = 1 + D^2 + D^3$ , $n_0(D) = 1+D + D^3$ và $n_1(D) = 1 +D + D^2 + D^3$. bộ tạo mã turbo này sẽ tạo ra chuỗi ký hiệu đầu ra giống như chuỗi được tạo ra bởi bộ mã cho ở **hình 3.1**

Khởi đầu các trạng thái của các thanh ghi dịch trong các bộ mã hõa thành phần được đặt về "0". Sau đó, các bit được dịch vào các bộ mã hóa thành phần theo vị


Bộ tạo mã thành phần 1
X
$Y_0$
$N_{turbo}$ bit thông tin vào
$n_0$
$Y_1$
$n_1$
d
Điều khiển
Chuyển mạch vào vị trí trên và dịch từng bit của $N_{turbo}$ bit số liệu; sau đó chuyển mạch vào vi trí dưới và từng bit đuôi trong số ba bit đuôi của bộ mã hóa thành phần 1, sau đó không ngừng dịch cho ba bit đuôi của bộ lập, mã thành phần 2
Trích bỏ ký hiệu và lặp
Bộ chèn turbo
Bộ mã hõa thành phần 2
X'
$Y_0$'
$n_0$
$Y'_1$
$n_1$
d
Điều khiển
Chuyển mạch vào vị trí trên và dịch từng bit của $N_{turbo}$ bit số liệu; sau đó chuyển mạch vào vi trí dưới và từng bit đuôi trong số ba bit đuôi của bộ mã hóa thành phần 1, sau đó không ngừng dịch cho ba bit đuôi của bộ lập, mã thành phần 2

**Hình 3.1**

trí của các chuyển mạch trên hình vẽ. mạch thay đổi chu kỳ từng bit số liệu và bit đuôi.

Các ký hiệu ra của số liệu sau mã hóa được tạo ra bằng cách dịch các bộ mã hóa thành phần $N_{turbo}$ lần khi các khóa ở vị trí trên và trích bỏ các đầu ra theo như quy định ở **bảng 3.1.** '0' ở mẫu trích bỏ có nghĩa là ký hiệu này sẽ bị xóa và '1' có nghĩa là ký hiệu này được cho qua. Đối với mỗi bit vào, đầu ra của các bộ lập mã thành phần sẽ được đặt vào chuỗi X, $Y_0$,$Y_1$, X', $Y_0$', $Y_1$'. Trong quá trình tạo ra các ký hiệu từ số liệu vào mã hóa sẽ không thực hiện lặp

**3.4.2 Kết cuối mã turbo:**

Bộ mã hóa turbo tạo ra 6/R các ký hiệu đuôi đầu ra tiếp sau các ký hiệu của các bit số liệu được mã hóa. Chuỗi ký hiệu đuôi đầu ra cũng giống như chuỗi được bộ mã hóa tạo ra ở **hình 3.1.** Các ký hiệu ra được tạo ra sau khi $N_{turbo}$ bit được dịch vào các bộ mã hóa thành phần với các khóa ở vị trí trên. 3/R ký hiệu đuôi ra đầu tiên được tạo ra bằng cách dịch bộ mã hóa thành phần 3 lần với khóa tương ứng ở vị trí dưới và đồng thời trích bỏ cũng như lặp các ký hiệu ra của bộ mã hóa thành phần này. 3/R các ký hiệu đuôi ra nhận được bằng cách dịch bộ mã hóa thành phần 2 ba lần với khóa tương ứng của nó ở vị trí dưới quá trình này kết hợp với trích bỏ và lặp các ký hiệu đầu ra của bộ mã hóa thành phần này. Các đầu ra của các bộ mã hóa thành phần đới với từng chu kỳ bit đuôi sẽ được đặt vào chuỗi X, $Y_0$, $Y_1$, X', $Y'_0$, $Y'_1$ với X ra trước.

Mẫu trích bỏ và lặp ký hiệu ra của bộ mã hóa thành phần được quy định ở **bảng 3.2** . Trong mẫu trích bỏ, '0' nghĩa là ký hiệu bị xóa còn '1' nghĩa là ký hiệu được cho qua. Đối mã turbo 1/2, các ký hiệu đuôi ra đối với 3 chu kỳ bit đuôi đầu tiên sẽ là $XY_0$ còn các ký hiệu đuôi ra đối với ba chu kỳ bit còn lại sẽ là $X'Y'_0$. Đối với mã turbo 1/3, các ký hiệu đuôi ra đối với 3 chu kỳ bit đuôi đầu tiên sẽ là $XXY_0$ còn các ký hiệu đuôi ra đối với ba chu kỳ bit đuôi còn lại sẽ là $X'X'Y'_0$ . Đối với mã turbo 1/4, các ký hiệu đuôi ra đối với 3 chu bit đuôi đầu tiên sẽ là $XXY_0Y_1$ còn các ký hiệu đuôi đối với 3 chu kỳ bit còn lại sẽ là $X'X'Y'_0Y_1$.

### 3.4.3. Các bộ chèn Turbo:

Bộ chèn turbo là một bộ phận của bộ mã hóa turbo có nhiệm vụ chèn khối cho số liệu, chỉ thị chất lượng khung (CRC) và các bit dành trước nhận được ở đầu vào của bộ mã hóa Turbo.

Bộ chèn turbo hoạt động như sau. Toàn bộ chuỗi bit đầu vào của bộ chèn turbo được viết vào ma trận nhớ lần lượt theo một trình tự các địa chỉ và sau đó toàn bộ chuỗi này được đọc ra từ bộ nhớ theo một trình tự các địa chỉ được xác định theo thủ tục trình bày dưới đây

| Đầu ra | Tỷ lệ mã | | |
|---|---|---|---|
| | 1/2 | 1/3 | 1/4 |
| X | 11 | 11 | 11 |
| $Y_0$ | 10 | 11 | 11 |
| $Y_1$ | 00 | 00 | 10 |
| X’ | 00 | 00 | 00 |
| $Y'_0$ | 01 | 11 | 01 |
| $Y'_1$ | 00 | 00 | 11 |

**Bảng 3.1. Mẫu trích bỏ cho các chu kỳ của bit số liệu**

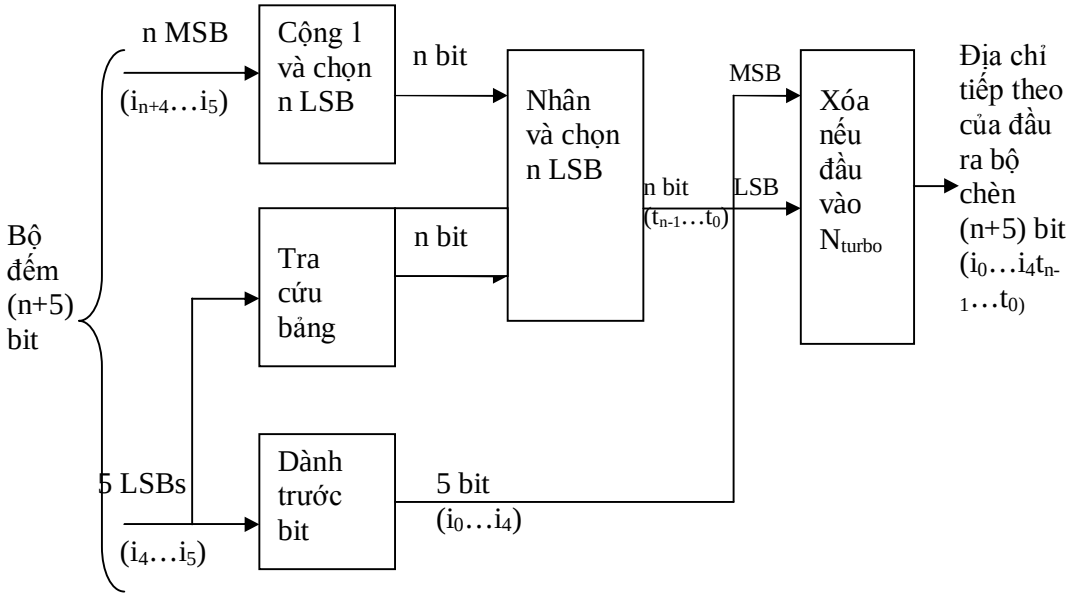
Lưu ý : Đối với từng tỷ lệ mã bảng trích bỏ sẽ được đọc từ trên xuống dưới sau đó từ trái sang phải

| Đầu ra | Tỷ lệ mã | | |
|---|---|---|---|
| | 1/2 | 1/3 | 1/4 |
| X | 111000 | 111000 | 111000 |
| $Y_0$ | 111000 | 111000 | 111000 |
| $Y_1$ | 000000 | 000000 | 111000 |
| X’ | 000111 | 000111 | 000111 |
| $Y'_0$ | 000111 | 000111 | 000111 |
| $Y'_1$ | 000000 | 000000 | 000111 |

**Bảng 3.2. Mẫu trích bỏ cho các chu kỳ bit đuôi**

Lưu ý: Đối với mã turbo 1/2, bảng trích bỏ được đọc từ trên xuống dưới sau đó từ trái sang phải. Đối với các mã turbo 1/3 và 1/4 bảng trích bỏ được đọc từ trên xuống dưới đồng thời với lặp X và X' sau đó đọc từ trái sang phải

Giả sử trình tự của các địa chỉ vào là từ 0 đến $N_{turbo}$-1, trong đó $N_{turbo}$ là số các ký hiệu ở bộ chèn sẽ được xác định theo thủ tục được cho ở **hình 3.2** như sau:




**Hình 3.2** thủ tục tinh toán đỉa chỉ đầu ra bộ chèn xen turbo

1. xác định thông số bộ chèn: n, trong đó n là số nguyên nhỏ nhất để $N_{turbo} \leq 2^{n+5}$ **. Bảng 3.3** cho các thông số này.
2. khởi đầu bộ đếm (n+5) bit vào "0".
3. lấy ra n bit trọng số cao nhất (MSB) từ bộ đếm và cộng 1 để được giá trị mới. Sau đó xóa tất cả trừ n bit trọng số thấp nhất (LSB) của giá trị này.
4. tra cứu bảng 3.4 theo địa chỉ đọc bằn năm bit trọng số thấp nhất (LSB) của bộ đếm. lưu ý rằng bảng này phụ thuộc vào giá trị n.
5. nhân các giá trị nhận được ở bước 3 và 4 rồi xóa tất cả trừ n bit trọng số thấp nhất (LSB).
6. đảo vị trí cho năm bit trọng số thấp nhất (LSB) của bộ đếm.
7. tạo địa chỉ ra thử với các bít trọng số cao (MSBs) nhận được ở bước 6 và các bit trọng số thấp (LSB) nhận được ở bước 5.

8. tiếp nhận địa chỉ ra thử này nếu nó không lớn $N_{turbo}$ , ngược lại xóa bỏ.
9. tăng bộ đếm và lặp lại các bước từ 3 đến 8 cho đến khi nhận được tất cả $N_{turbo}$ đỉa chỉ ra cho bộ chèn.

| Kích thước khối của bộ chèn Turbo Turbo ($N_{turbo}$) | Thông số của bộ chèn Turbo n |
|---|---|
| 378 | 4 |
| 570 | 5 |
| 762 | 5 |
| 1.146 | 6 |
| 1.530 | 6 |
| 2.298 | 7 |
| 3.066 | 7 |
| 4.602 | 8 |
| 6.138 | 8 |
| 9.210 | 9 |
| 1.282 | 9 |
| 20.730 | 10 |

**Bảng 3.3**. Thông số của bộ chèn turbo

| Bảng chỉ số | n=4 | n=5 | n=6 | n=7 | n=8 | n=9 | n=10 |
|---|---|---|---|---|---|---|---|
| 0 | 5 | 27 | 3 | 15 | 3 | 13 | 1 |
| 1 | 15 | 3 | 27 | 127 | 1 | 335 | 349 |
| 2 | 5 | 1 | 15 | 89 | 5 | 87 | 303 |
| 3 | 15 | 15 | 13 | 1 | 83 | 15 | 721 |
| 4 | 1 | 13 | 29 | 31 | 19 | 15 | 973 |
| 5 | 9 | 17 | 5 | 15 | 179 | 1 | 703 |
| 6 | 9 | 23 | 1 | 61 | 19 | 333 | 761 |
| 7 | 15 | 13 | 31 | 47 | 99 | 11 | 327 |
| 8 | 13 | 9 | 3 | 127 | 23 | 13 | 453 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 9 | 15 | 3 | 9 | 17 | 1 | 1 | 95 |
| 10 | 7 | 15 | 15 | 119 | 3 | 121 | 241 |
| 11 | 11 | 3 | 31 | 15 | 13 | 155 | 187 |
| 12 | 15 | 13 | 17 | 57 | 13 | 1 | 497 |
| 13 | 3 | 1 | 5 | 123 | 3 | 175 | 909 |
| 14 | 15 | 13 | 39 | 95 | 17 | 421 | 769 |
| 15 | 5 | 29 | 1 | 5 | 1 | 5 | 349 |
| 16 | 13 | 21 | 19 | 85 | 63 | 509 | 71 |
| 17 | 15 | 19 | 27 | 17 | 131 | 215 | 557 |
| 18 | 9 | 1 | 15 | 55 | 17 | 47 | 197 |
| 19 | 3 | 3 | 13 | 57 | 131 | 425 | 499 |
| 20 | 1 | 29 | 45 | 15 | 211 | 295 | 409 |
| 21 | 3 | 17 | 5 | 41 | 173 | 229 | 259 |
| 22 | 15 | 25 | 33 | 93 | 231 | 427 | 335 |
| 23 | 1 | 29 | 15 | 87 | 171 | 83 | 253 |
| 24 | 13 | 9 | 13 | 63 | 23 | 409 | 677 |
| 25 | 1 | 13 | 9 | 15 | 147 | 387 | 717 |
| 26 | 9 | 23 | 15 | 13 | 243 | 193 | 313 |
| 27 | 15 | 13 | 31 | 15 | 213 | 57 | 757 |
| 28 | 11 | 13 | 17 | 81 | 189 | 501 | 189 |
| 29 | 3 | 1 | 5 | 57 | 51 | 313 | 15 |
| 30 | 15 | 13 | 15 | 31 | 15 | 489 | 75 |
| 31 | 5 | 13 | 33 | 69 | 67 | 391 | 163 |

**Bảng 3.4**. quy định bảng tra cứu cho bộ chèn Turbo

### 3.4.4. Phối hợp tốc độ trong hệ thống cdma2000:

Phối hợp tốc độ có nghĩa là lặp hoặc trích bỏ các ký hiệu ở kênh truyền tải (viết tắt là TrCH) để đạt được tốc độ ký hiệu như nhau cho các kênh có tốc độ bít khác nhau ở các kênh vô tuyến khác nhau. Lớp cao sẽ ấn định thuộc tính của phối

hợp tốc độ cho từng TrCH. Thuộc tính này là bán cố định và chỉ có thể thay đổi theo thông báo của lớp cao. Thuộc tính phối hợp tốc độ được sử dụng để tính số bit cần lặp hoặc trích bỏ.

Trong hệ thống cdma2000, lặp ký hiệu và trích bỏ ký hiệu được thực hiện ở sau bộ mã hóa kênh. Mẫu lặp và trích bỏ phụ thuộc vào cấu hình vô tuyến (RC: Radio configuration). Dưới đây ta xét một số thí dụ về lặp và trích bỏ ký hiệu ở cdma2000

Lặp ký hiệu mã

Các ký hiệu mã ở đầu ra của bộ mã hóa hiệu chỉnh lỗi thuận sẽ được lặp theo quy định ở **bảng 3.5**

| Kiểu kênh | | Số ký hiệu mã sau lặp/ ký hiệu mã |
|---|---|---|
| Kênh truy nhập (chỉ cho tốc độ trải phổ một) | | 2 |
| Kênh truy nhập cải tiến | | 4 (9600 bit/s)<br>2 (19200 bit/s)<br>1 (38400 bit/s) |
| Kênh điều khiển chung đường lên | | 4 (9600 bit/s)<br>2 (19200 bit/s)<br>1 (38400 bit/s) |
| Kênh điều khiển riêng đường lên | | 2 |
| Kênh cơ bản đường lên | RC 1 hay 2 | 8 (1200 hay 1800 bit/s)<br>4 (2400 hay 3600 bit/s)<br>2 (4800 hay 7200 bit/s)<br>1 (9600 hay 14400 bit/s) |
| | RC 3,4,5 hay 6 | 16 (1500 hay 1800 bit/s)<br>8 (2700 hay 3600 bit/s)<br>4 (4800 hay 7200 bit/s)<br>2 (9600 hay 14400 bit/s) |
| Kênh mã bổ sung đường lên | (RC 1hay 2) | 1 |

| | | |
|---|---|---|
| Kênh bổ sung đường l | | 1 |

**Bảng 3.5** quy định bảng tra cứu cho bộ đan xen turbo

### 3.4.5. chèn trong cdma2000:

Chèn thường đi với mã hóa kênh để tăng hiệu quả của sửa lỗi. trong thông tin di động do pha đinh sâu, các bit thường xẩy ra từng cụm dài. Tuy nhiên mã hóa kênh đặc biệt là mã tích chập chỉ hiệu quả nhất khi sửa các lối ngẫu nhiên đơn lẻ hoặc các cụm lỗi không quá dài. Để đối phó với vấn đề này người ta chia khối bản tin cần gửi thành các cụm ngắn rồi hoán vị các cụm này với các cụm của các khối bản tin khác, nhờ vậy khi xẩy ra cụm lỗi dài mỗi khối bản tin chỉ mất đi một cụm nhỏ và phần còn lại của khối bản tin vẫn cho phép các dạng mã hóa kênh khôi phục được khối đúng sau khi đã sắp xếp lại các cụm của khối bản tin theo thứ tự như phía phát. Chẳng hạn ta có 4 khối bản tin **hình 3.3** ta chia mỗi khối thành 4 cụm và đánh số cho các cụm này từ 1 đến 4, sau đó hoán vị các cụm với nhau bằng cách ghép chung các cụm 1 vào một khối và cụm vào một khối… giả sử ở phía thu các cụm 2 bị mắc lỗi, sau khi sắp xếp lại các khối bản tin các cụm 1, 3, 4 còn lại sẽ cho phép mã hóa kênh khôi phục lại khối đúng

#### 3.4.5.1. Chèn khối:

Đối với kênh đồng bộ, các kênh tìm gọi, các kênh quảng bá, kênh ấn định chung, kênh điều khiển và các kênh lưu lượng đường xuống, tất cả các ký hiệu sau lặp và trích bỏ (nếu có) sẽ được chèn khối.

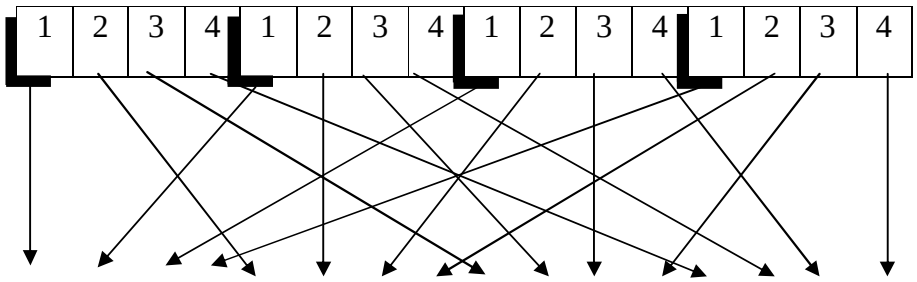
Các ký hiệu đầu vào lần lượt được viết vào bộ đan xen theo địa chỉ từ 0 đến kích thước khối (N) trừ một. Các ký hiệu sau đan xen được đọc ra theo trính tự hoán vị từ địa chỉ $A_i$ như sau:

$$A_i = 2^m(i \bmod j) + BRO_m([i/j]) \qquad (3.2)$$

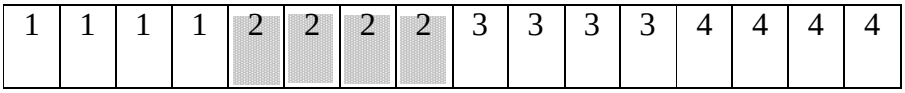
Trong đó :

i = 0 đến N-1

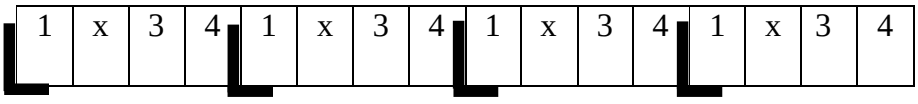
[x] biểu thị số nguyên lớn nhất nhỏ hơn x và $BRO_m(y)$ biểu thị giá trị m bit đảo của y (chẳng hạn BRO3(6) = 3).




Các khối bản tin được phân thành 4 cụm nhỏ




Các khối bản tin sau đan xen và phát




Các khối bản tin thu sau khi giải đan xen

Cụm bi tác động xấu bởi pha đinh

X: là cụm bị hỏng

**Hình 3.3**. Nguyên lý đan xen

Các thông số của bộ đan xen: m và j được quy định ở **bảng 3.6 hình 3.4** cho thấy cấu hình của bộ đan xen đối với các chế độ DS non – OTD (direct spreading – non Orthogonal Transmit diversity: đơn sóng mang không sử dụng phân tập phát trực giao), DS OTD (đơn sóng mang với phân tập phát trực giao) và MC (Multicarrier: đa sóng mang).

**3.4.5.2. chen đa khung:**

Khi số bit của kênh bổ sung đường xuống là 360 hay lớn hơn, BS phải cho phép đan xen kênh bổ sung ở hai hoặc khung liên tiếp theo quy định được xác định bởi MULTI_FRAME_LENGTH.

Cấu trúc của bộ chèn khối n khung (n=2 hay 4) giống như cấu trúc của bộ chèn đơn khung. Thông số đan xen cho bộ chèn n khung được quy định ở **bảng 3.7**

| Kích thước đan xen | m | j | Kích thước đan xen | m | j |
|---|---|---|---|---|---|
| 48 | 4 | 3 | 144 | 4 | 9 |
| 96 | 5 | 3 | 288 | 5 | 9 |
| 192 | 6 | 3 | 576 | 5 | 18 |
| 384 | 6 | 6 | 1.152 | 6 | 18 |
| 768 | 6 | 12 | 2.304 | 6 | 36 |
| 1.536 | 6 | 24 | 4.608 | 7 | 36 |
| 3.072 | 6 | 48 | 9.216 | 7 | 72 |
| 6.144 | 7 | 48 | 18.432 | 8 | 72 |
| 12.288 | 7 | 96 | 36.864 | 8 | 144 |
| 72 | 3 | 9 | 128 | 7 | 1 |

**Bảng 3.6 các thông số chèn**

| TTI | Số cột $C_1$ | Các mẫu hoán vị giữa các cột |
|---|---|---|
| 10 ms | 1 | {0} |
| 20 ms | 2 | {0,1} |

| | | |
|---|---|---|
| 40 ms | 4 | {0,2,1,3} |
| 80 ms | 8 | {0,4,2,6,1,5,3,7} |

**Bảng 3.7. các mẫu hoán vị giữa các cột**

**3.4.5.3. chen OTD:**

Khi sử dụng chế độ OTD, bộ đan xen khối sẽ phân luồng các ký hiệu vào thành hai luồng. Mỗi luồng được đan xen theo thủ tục được trình bày ở **3.3.1** . Khối thứ hai được dịch vòng N/4 ký hiệu sau đó hai khối được ghép chung. Dịch vòng được thực hiện bằng cách dịch 3N/4 ký hiệu đến cuối khối này và N/4 ký hiệu đến đầu khối này. Thủ tục dịch vòng được cho ở **hình 3.4**

**3.4.5.4 chèn MC:**

Đối với chế độ MC, bộ đan xen khối sẽ phân luồng các ký hiệu đầu vào thành ba khối N/3 ký hiệu.

Các ký hiệu vào k bộ đan xen khối (k = 0,1,2) lần lượt được ghi vào bộ nhớ ở các địa chỉ từ 0 đến N/3 – 1. các ký hiệu sau đan xen được đọc ra theo trình tự hoán vị từ địa chỉ $A_i$ như sau:

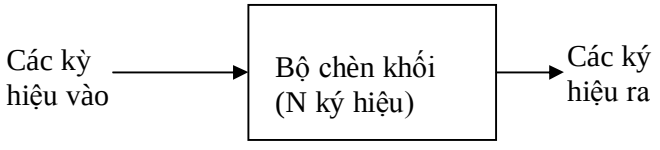
$A_i = 2^m((i+[kN/9] \bmod J) + BRO_m([(i+[kN/9])/J])$ (3.3)

Trong đó:

i = 0 đến N/3-1

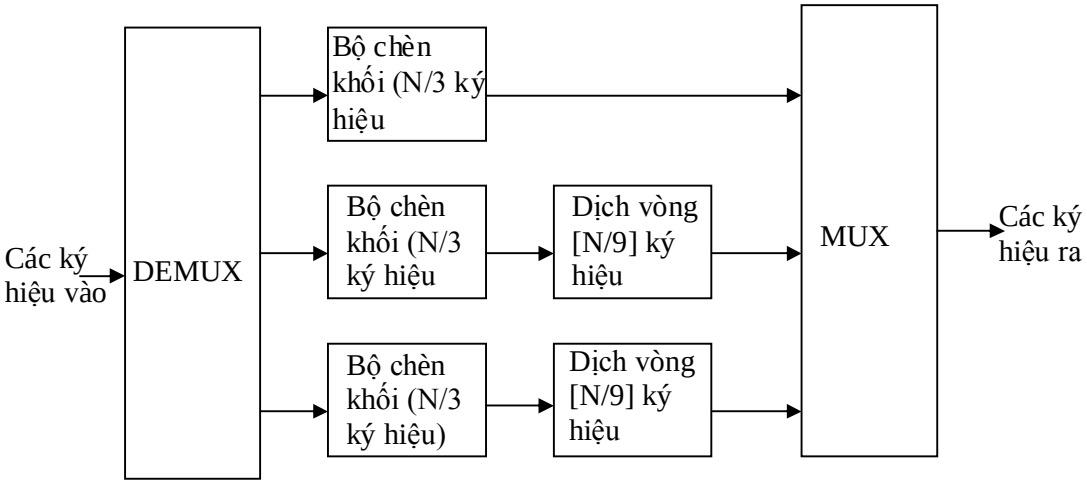
[x] chỉ thị số nguyên lớn nhất nhỏ hơn hay bằng x và $BRO_m(y)$ chỉ thị giá trị m bị đảo vị trí ( chẳng hạn $BRO_3(6)=3$).

Sau đó ba đầu ra của các khối đan xen ghép chung với nhau. Quá trình đan xen khối MC được thực hiện đồng thời cho cả đan xen khối và dịch vòng khối như ở **hình 3.4**




**a) chế độ DS Non - OTD**

**b) chế độ DS OTD**




**c) các chế độ MC**

chức năng của DEMUX là phân phối lần lượt các ký hiệu vào cho đường trên và đường dưới.

chức năng của MUX là kết hợp lần lượt các ký hiệu vào từ đường trên và đường xuống dưới.

**hình 3.4. cấu trúc các bộ đan xen khối N ký hiệu**

**3.5 kết luận:**

Qua chương này ta biết được những ứng dụng của mã TC vào trong truyền thông đa phương tiện và truyền thông không dây, cụ thể là ứng dụng trong cdma2000. Những khó khăn khi thực hiện để tìm ra những cách khắc phục những nhược điểm đó nhằm mang lại những tiện ích hơn nữa của công nghệ CDMA đối với cuộc sống con người. Bộ mã sử dụng trong cdma2000 sẽ được mô phỏng trong chương tiếp theo giúp ta hiểu sâu hơn về nó.

# CHƯƠNG 4: CHƯƠNG TRÌNH MÔ PHỎNG MÃ TURBO TRONG HỆ THỐNG THÔNG TIN DI ĐỘNG CDMA2000

## 4.1 Giới thiệu chương:

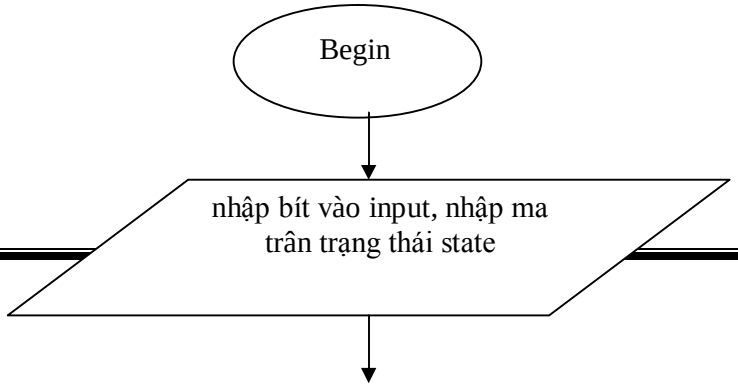
Trong chương này trình bày chương trình mô phỏng bộ mã turbo sử dụng trong hệ thống thông tin di động thế hệ 3 theo chuẩn CDMA2000. Chương trình được viết bằng ngôn ngữ Matlab, thông qua chương trình mô phỏng giúp ta kiểm tra lại lý thuyết và hiểu sâu hơn về mã turbo,cũng như khả năng ứng dụng của mã turbo khi tốc độ bít cao Qua đó cho chúng ta đánh giá được những đặc điểm như khả năng sửa lỗi ...mà các loại mã hóa kênh khác không có. Trong chương trình mô phỏng ta nhập các bit số liệu vào khác nhau, số lần lặp giải mã khác nhau, cũng như số bit khung để thu được kết quả giải mã, BER khác nhau. Bộ mã này có hàm truyền như sau:

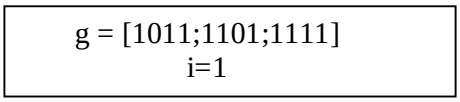
$$G(D)=\left[1 \quad \frac{n_0(D)}{d(D)} \quad \frac{n_1(D)}{d(D)}\right]$$

Trong đó $d(D) = 1+D^2+D^3$ , $n_0(D) = 1+D+D^3$ và $n_1(D) = 1+D+D^2+D^3$

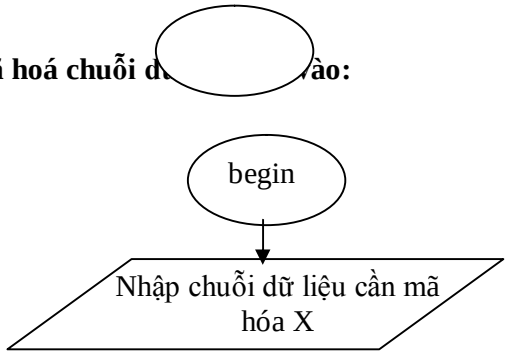
## 4.2. Lưu đồ thuật toán:

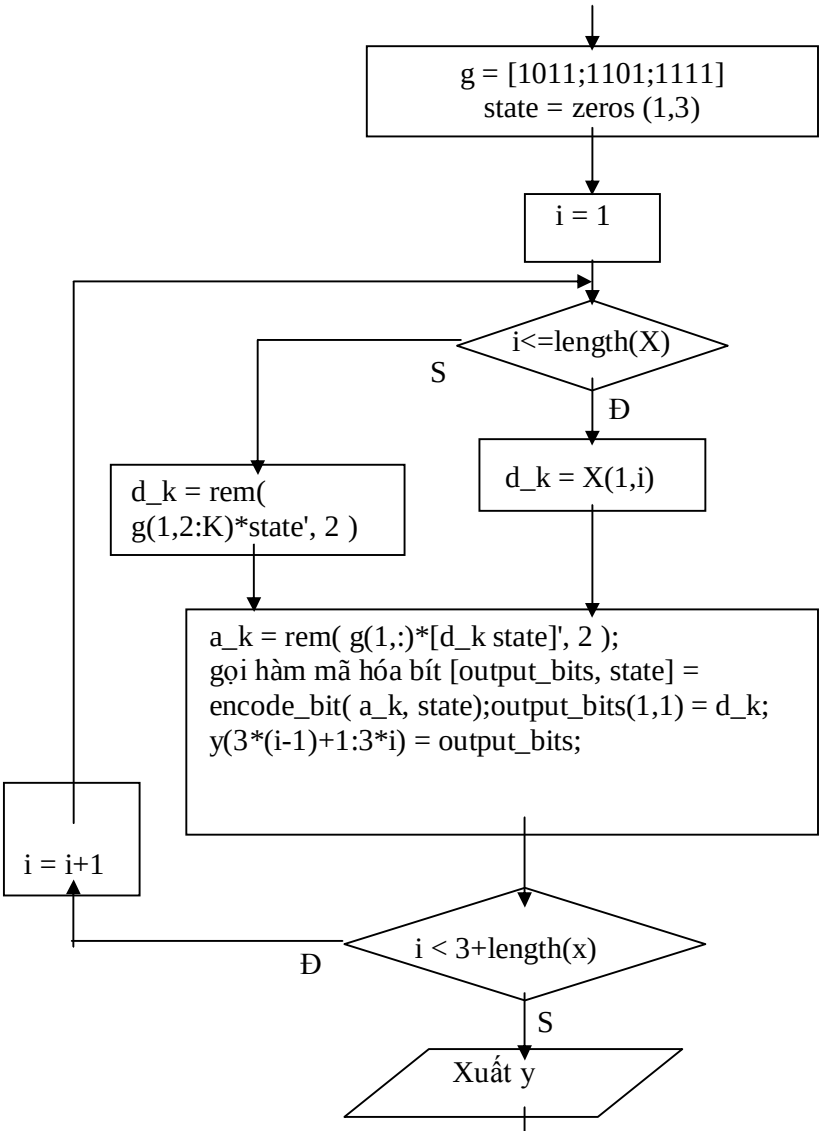
### 4.2.1. Lưu đồ thuật toán chương trình mã hoá theo bít:



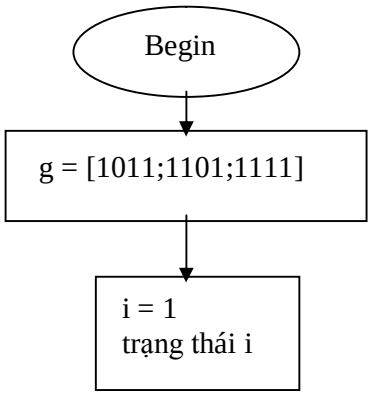


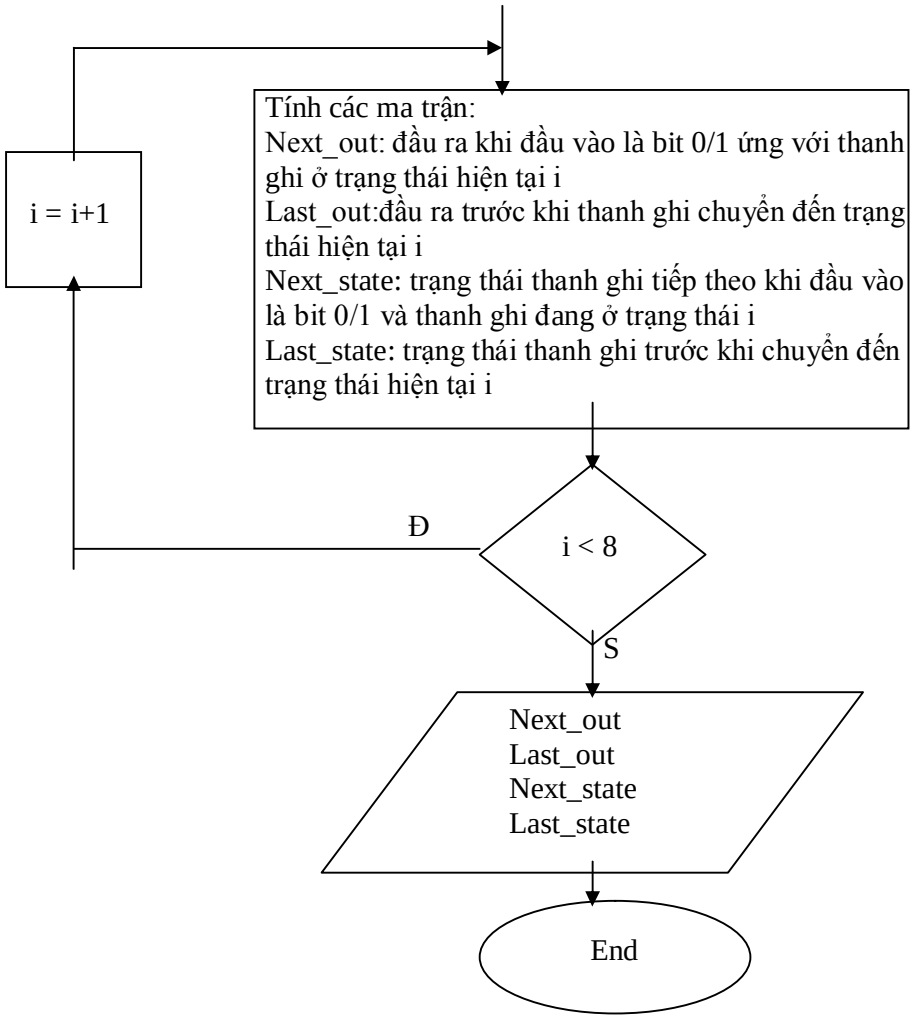

**4.2.2. Lưu đồ thuật toán mã hoá chuỗi d[illegible]ào:**



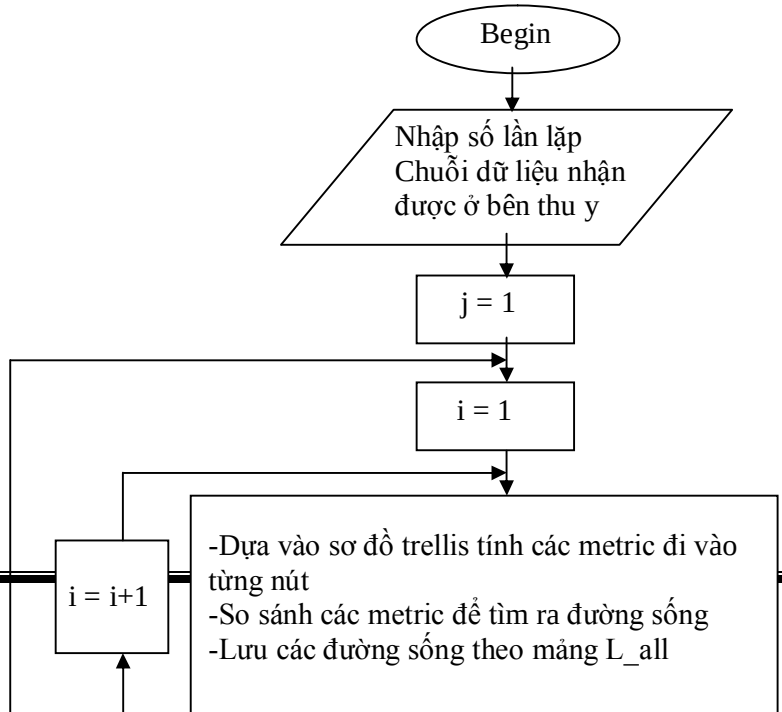



**4.2.3. Lưu đồ thuật toán tính cá[illegible] thái trellis:**

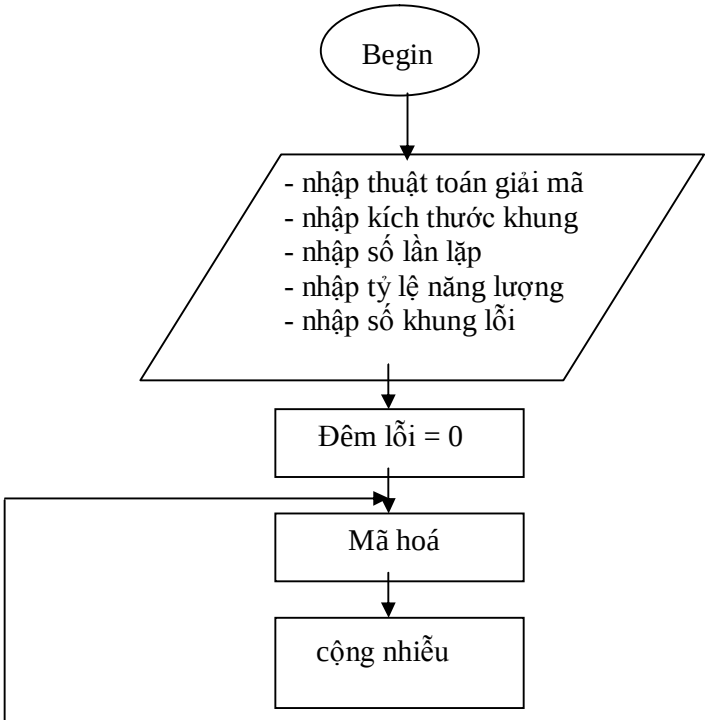


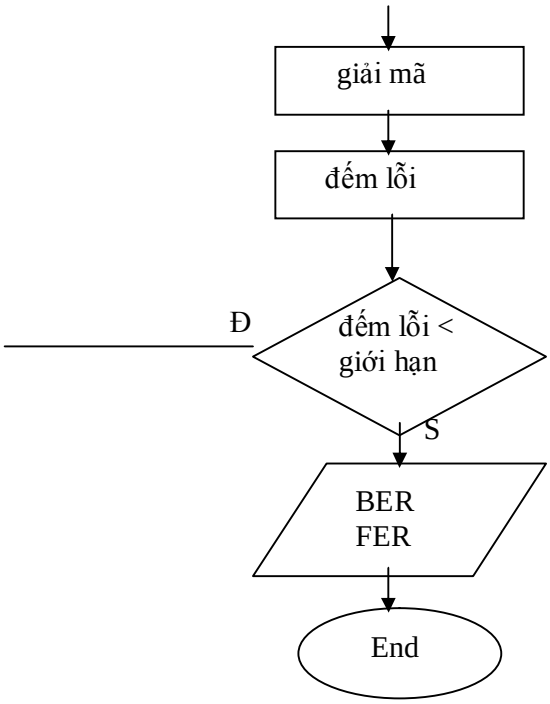



**4.2.4. Lưu đồ thuật toán giải mã turbo:**



**4.2.5. Lưu đồ thuật toán tính lỗi bit và lỗi khung:**

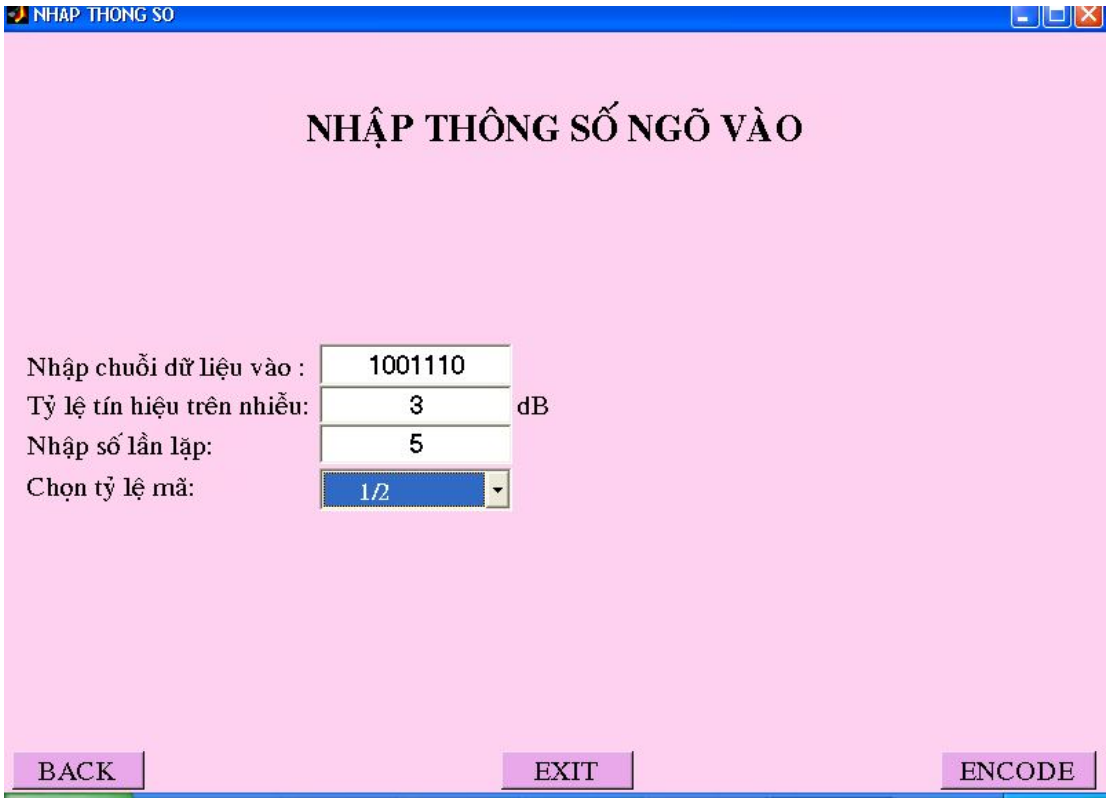




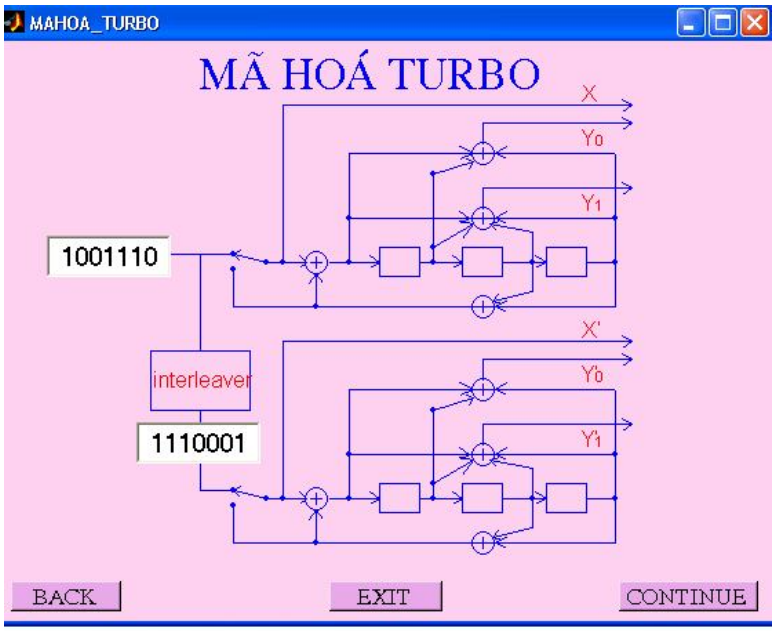

**4.3. giao diện và kết quả chương trình mô phỏng:**

**Giao diện chương trình**

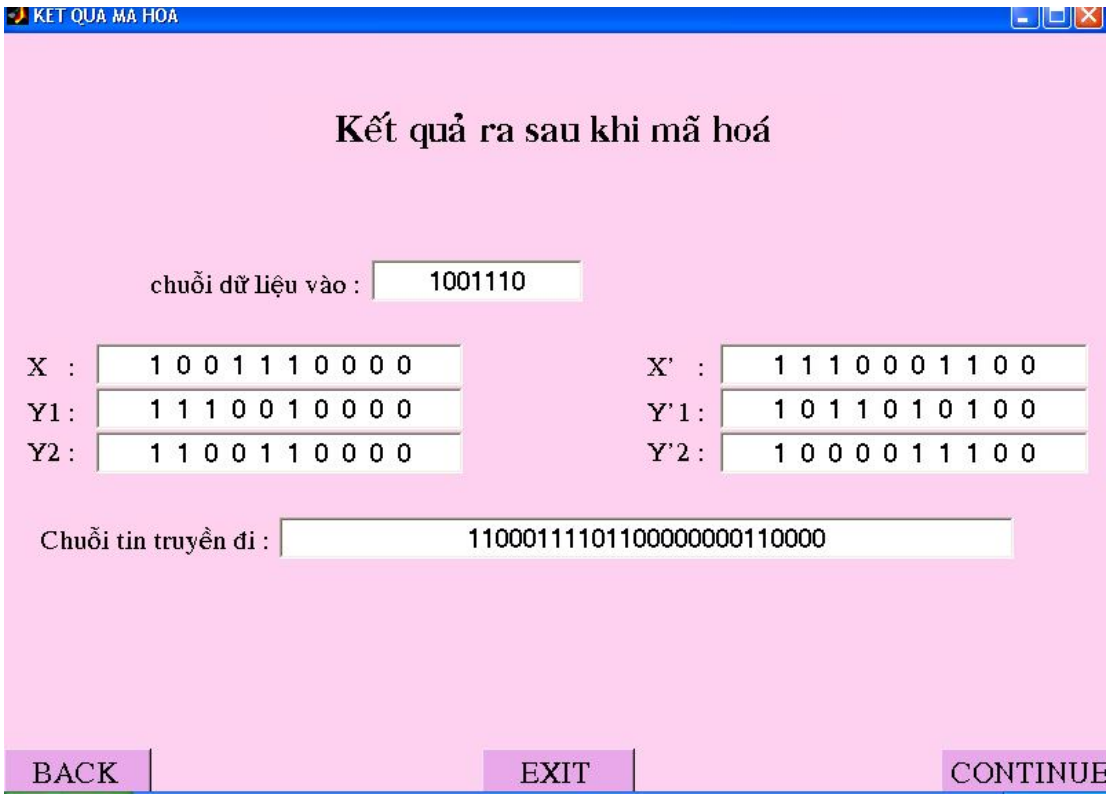

Khi chọn “Exit ” chương trình sẽ thoát còn chọn “continue” chương trình sẽ tiếp tục cho ra trang nhập thông số vào



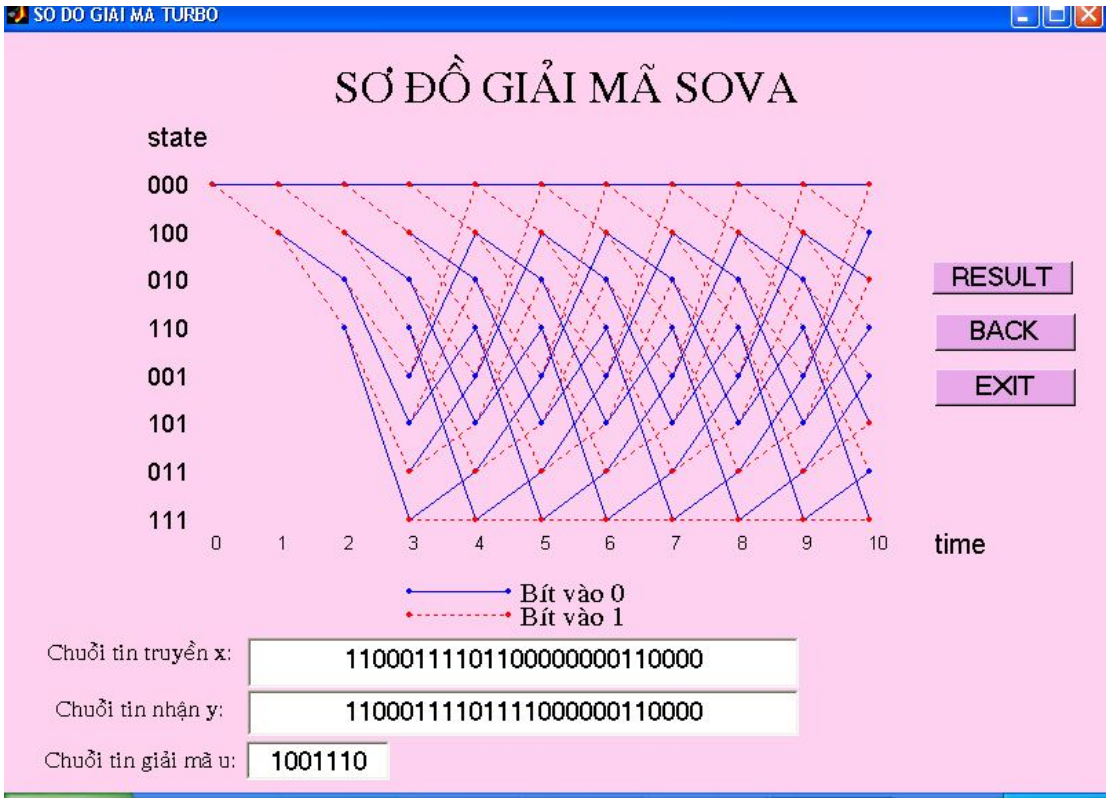

Ta nhập chuỗi dữ liệu vào, tỷ lệ tín hiệu trên nhiễu, số lần lặp giải mã, tỷ lệ mã truyền đi có 3 tỷ lệ là 1/2, 1/3, 1/4. chọn "ENCODE" để tiếp tục tới trang mã hoá



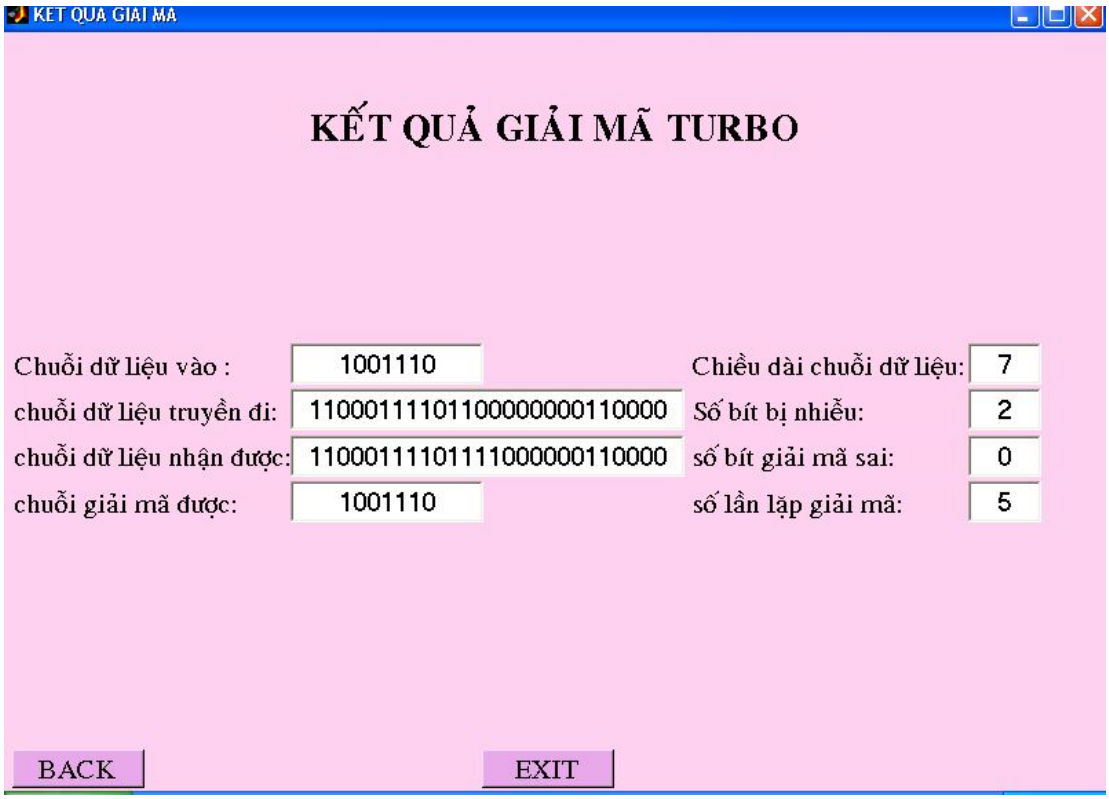
xuất hiện bộ mã hoã Turbo CDMA2000 chuỗi dữ liệu đưa vào sau khi qua bộ chèn hoán vị ngẫu nhiên cho ra chuỗi mới để đưa vào bộ mã hoá thành phần thứ hai. Ta chọn "CONTINUE" để đưa ra kết quả mã hoá .



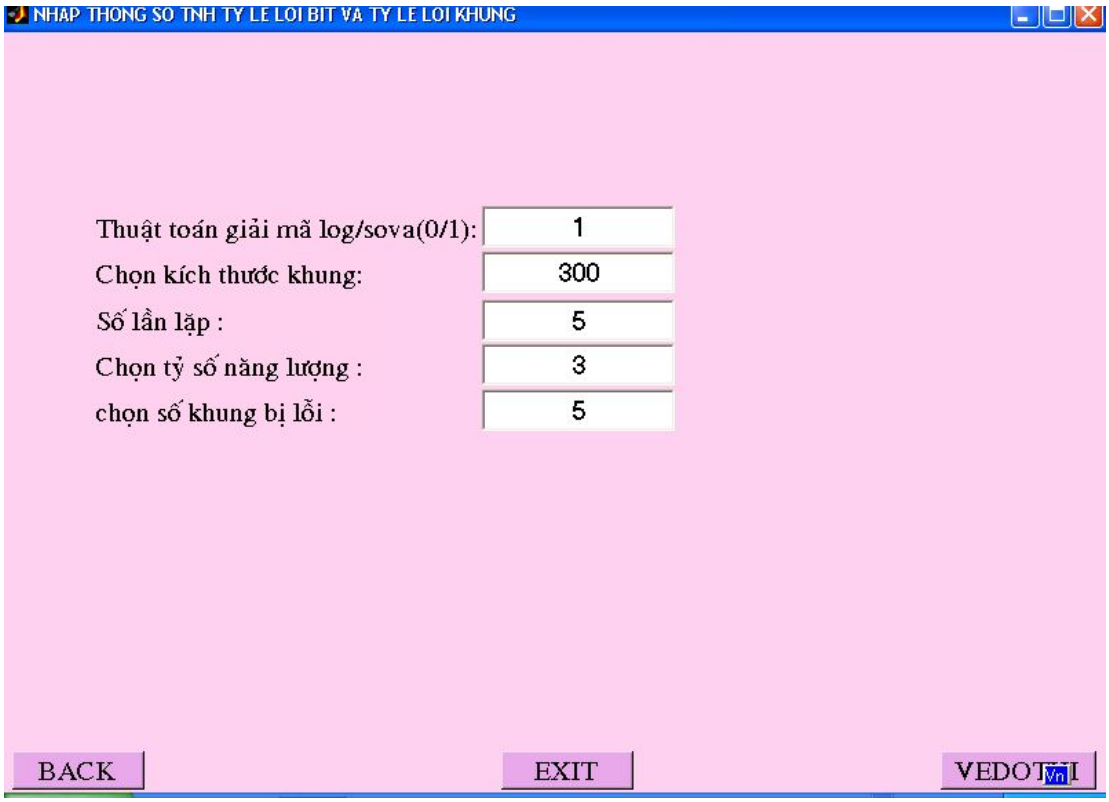

Ta thu được kết quả mã hoá như trên hình và đưa ra chuỗi tin cần truyền đi phụ thuộc vào việc chọn tỷ lệ mã trước. chọn "BACK" để quay về trang trước, chọn "EXIT" để thoát, chọn "CONTINUE" để tiếp tục đến trang sau.



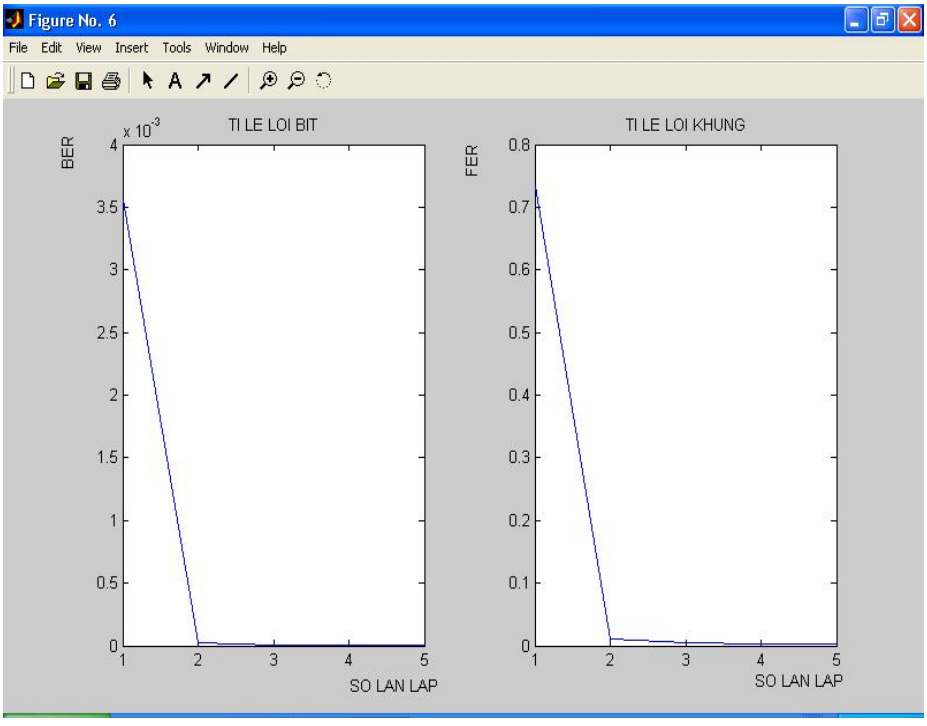

xuất hiện sơ đồ lưới dùng để giải mã, tiến hành giải mã chuỗi tin nhận được y chuỗi này có một số bít lỗi khác với chuỗi truyền. ta tiếp tục chọn “RESULT” để đưa ra kết quả giải mã.




ta tiếp tuc chạy chương trinh tính tỷ lệ lỗi bit và lỗi khung khi nhập các thông số đầu vào khác nhau thì ta thu được những gia trị lỗi bít và lỗi khung khác nhau. Và có đồ thị khác nhau




Sau khi tính lỗi xong nó sẽ đưa kết quả đến để vẽ đồ thị




Ta chạy chương trình mô phỏng nhiều lần ta đưa ra một số nhận xét như sau:

+ Khi số lần lặp tăng từ thì tỉ lệ lỗi bit cũng như tỉ lệ lỗi khung đều giảm. 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 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 lần lặp ,các bộ giải mã đã lấy được hết thông tin của mã ngõ vào và do dó : không cho ra ở ngõ ra các giá trị biến đổi nữa như trong lần lặp thứ nhất .Vì vậy, có thể nói việc 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 giãm

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 p hù 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 Méo 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 chẳn lẽ của nó và chuỗi dữ liệu ngõ vào thông tin

+ nếu số lượng khung đưa vào càng lớn thì BER và FER càng thấp

+ mã sẽ hoạt động tốt khi ta lựa chọn kích thước khung lớn.

+ tỉ lệ lỗi khung(FER) thường lớn hơn tỉ lệ lỗi bit(BER) nhưng lần lặp càng lớn thì BER~FER

## 4.4 Kết luận và hướng phát triển của đề tài

Do quy mô và thời gian thực hiện đồ án có hạn nên trong đồ án không thể trình bày hết mọi vấn đề của mã Turbo mà chỉ tập trung vào các vấn đề cốt lõi của mã Turbo kết nối song song. Ngoài ra do hạn chế về thời gian nên việc phân tích kỹ về các yếu tố ảnh hưởng đến chất lượng bộ mã cũng hạn chế và các biện pháp để cải tiến chất lượng bộ mã TC cũng chỉ mới được một số ít và chưa đi sâu vào chi tiết. Ở đây chỉ xin nêu một số hướng có thể

nghiên cứu tiếp về mã Turbo theo các công trình nghiên cứu trên thế giới và người viết đề nghị

+ Nghiên cứu các thuật toán gần tối ưu khác

+ Áp dụng trong các hệ thống Hybrid ARQ

+ Ứng dụng trong truyền thông không dây thế hệ thứ ba và có thể là thứ tư

+ Nghiên cứu bổ sung các mã TC có chiều dài các mã thành phần biến đổi

+ Nghiên cứu sử dụng "Lý thuyết biến đổi Wavelets trên trường hữu hạn" để tạo ra các loại ECC mới hay cải tiến đơn giản hóa các mã ECC đã có Ngoài ra có thể mở rộng nghiên cứu về FEC hơn nữa bằng các mã khác cũng đang được nghiên cứu áp dụng vào thực tiễn như :

+ Mã Woven : một dạng gần tương tự với mã Turbo

+ Mã Turbo Block Code : mã Turbo cải tiến từ mã khối

+ Mã GC (Generic Codes)

+ Họ mã SPC (Sparse Graph Codes)

+ Mã LDPC (Low Density Parity check Codes), PA (Product Accumulate Codes) hay GPA (Generalized Product Accumulate Codes).

+ Mã TCM (Turbo Codes Modulation) : Kết hợp TC và điều chế.

+ Các loại kết hợp mã hóa nguồn và mã hóa kênh

+ Tìm hiểu về mã Turbo kết nối nối tiếp( SCCC).

+ Tìm hiểu về mã Turbo kết nối hổn hợp(HCCC).

+ Ứng dụng mã Turbo vào các hệ thống truyền thông thế hệ thứ 4.

+ Thiết kế bộ chèn tối ưu sử dụng cho từng bộ mã Turbo cụ thể.