MỤC LỤC

[MỞ ĐẦU 2](#_Toc38028800)

[CHƯƠNG 1: MÃ KIỂM TRA CHẴN LẺ MẬT ĐỘ THẤP PHI NHỊ PHÂN (NB-LDPC NON BINARY LOW DENSITY PARITY CHECK) 4](#_Toc38028801)

[*1.1. Giới thiệu về mã kênh NB-LDPC* 4](#_Toc38028802)

[1.1.1. Mã kênh 4](#_Toc38028803)

[1.1.2. Mã LDPC 5](#_Toc38028804)

[1.1.3. Mã LDPC phi nhị phân 7](#_Toc38028805)

[1.2. Phương pháp xây dựng bộ mã NB-LDPC 8](#_Toc38028806)

[1.2.1. Các thuật toán xây dựng ma trận kiểm tra H 8](#_Toc38028807)

[1.2.2. Mã hóa bản tin từ ma trận kiểm tra H 11](#_Toc38028808)

[1.3. Thuật toán quyết định cứng (Hard Dicision) 15](#_Toc38028809)

[1.4. Thuật toán quyết định mềm – giải mã lặp (Soft Dicision) 16](#_Toc38028810)

[CHƯƠNG 2: THIẾT KẾ KHỐI GIẢI MÃ LAYERED TRELLIS MIN-MAX DECODER TRÊN PHẦN CỨNG FPGA 22](#_Toc38028811)

[2.1. Thiết kế khối giải mã Layered Trellis Min-Max Decoder 22](#_Toc38028812)

[2.1.1. Thuật toán xử lý node kiểm tra Trellis Min-Max 23](#_Toc38028813)

[2.1.2. Thuật toán Layered Schedule cho bộ giải mã 27](#_Toc38028814)

[2.2. Kết quả mô phỏng đánh giá 29](#_Toc38028815)

[2.2.1. Kết quả mô phỏng thuật toán trên phần mềm 29](#_Toc38028816)

[2.2.2. Đánh giá chất lượng bộ giải mã trên chíp FPGA xillinx sparta-6 31](#_Toc38028817)

[KẾT LUẬN 35](#_Toc38028818)

[*Tài liệu tham khảo:* 37](#_Toc38028819)

# MỞ ĐẦU

***Tính cấp thiết của đề tài***: Mã hóa kênh có vai trò quan trọng trong việc truyền dẫn thông tin số cũng như lưu trữ dữ liệu số. Mục đích của mã hóa kênh là nhằm tăng khả năng tái tạo dữ liệu bị can nhiễu ở phía đầu thu. Ngày nay, các dịch vụ trên mạng viễn thông gia tăng không ngừng trong khi nguồn tài nguyên của mạng viễn thông là hữu hạn. Vì vậy, việc khai thác nguồn tài nguyên của mạng viễn thông một cách hiệu quả là yêu cầu tiên quyết trong thiết kế hệ thống viễn thông số. Mã B-LDPC và NB-LDPC là những mã kênh tiên tiến nhất, cho khả năng sửa lỗi rất tốt (theo lý thuyết chỉ cách giới hạn Shannon 0.0045 dB), tuy nhiên, do thuật toán giải mã cho những mã này có độ phức tạp rất lớn nên vẫn chưa được áp dụng rộng rãi trong thực tế. Nhờ những cải tiến nhằm giảm độ phức tạp của thuật toán giải mã trong những năm gần đây, mã B-LDPC đã dần được đưa vào các ứng dụng viễn thông. Năm 2003, mã B-LDPC đã vượt qua 6 mã Turbo khác để trở thành mã sửa lỗi tiêu chuẩn ***DVB-S2*** cho truyền hình số vệ tinh. Năm 2008, B-LDPC đánh bại mã turbo xoắn và được chọn làm mã sửa lỗi cho tiêu chuẩn ***ITU-T G.hn***. Ngoài ra, mã B-LDPC cũng được sử dụng cho ***Ethernet T-10Gbase***, ***WI-FI 802.11ac*** (2009). Nhược điểm thứ 2 của mã B-LDPC là nó yêu cầu độ dài từ mã rất lớn (lớn hơn 1000.000 bit), khiến nó không phù hợp với hệ thống thông tin thời gian thực hay các hệ thống truyền nhận các gói tin kích thước nhỏ. Mã NB-LDPC (mã LDPC phi nhị phân) được chứng minh cho chất lượng tốt hơn cả mã B-LDPC ngay cả với độ dài từ mã ngắn hoặc trung bình (khoảng vài nghìn symbols), đồng thời cũng thuận tiện trong việc triển khai cùng với các phương pháp điều chế bậc cao(8-PSK, 16-QAM, 64-QAM...). Tuy nhiên, mã NB-LDPC có độ phức tạp tính toán lớn hơn nhiều so với mã B-LDPC, đặc biệt trên các trường bậc cao. Do đó, mã NB-LDPC đang được tích cực nghiên cứu nhằm có khả năng triển khai vào thực tế, mã NB-LDPC cũng được đề xuất là mã sửa lỗi tiêu chuẩn cho hệ thống di động 6G.

***Đề tài: “Nghiên cứu thuật toán giải mã MIN-MAX cho mã kênh LDPC phi nhị phân và thiết kế khối giải mã trên phần cứng FPGA”*** nhằm các mục tiêu sau:

* Nghiên cứu về thuật toán giải mã Min-Max cho mã NB-LDPC- là thuật toán tiên tiến nhất hiện nay, cải tiến rất nhiều về độ phức tạp và là thuật toán có tiềm năng nhất trong triển khai thực tế.
* Đề xuất một số cải tiến trong thuật toán nhằm giảm độ phức tạp và tăng khả năng chống lỗi của bộ giải mã.
* Từ thuật toán, nghiên cứu, thiết kế khối mã hóa và giải mã trên phần cứng FPGA.

***Đối tượng nghiên cứu của đề tài:***

* Các thuật toán mã hóa, giải mã NB-LDPC
* Thiết kế bộ giải mã trên chip Xillinx sparta-6, kiểm thử khả năng kiểm soát lỗi của bộ giải mã trên kênh AWGN.

***Phương pháp nghiên cứu:***

* Nghiên cứu lý thuyết về các thuật toán mã hóa, giải mã B-LDPC, NB-LDPC.
* Viết trương trình mô phỏng, đánh giá khả năng sửa lỗi của thuật toán trên ngôn ngữ Python.
* Thiết kế khối giải mã bằng ngôn ngữ VHDL, mô phỏng kết quả trên ModelSIM.
* Tổng hợp thiết kế và nạp thiết kế khối giải mã vào chíp sparta6 bằng phần mềm Xilinx-ISE.
* Kiểm thử khả năng hoạt động của khối giải mã.

***Cấu trúc của đề tài gồm mở đầu, 2 phần và phụ lục:***

* Phần I: Mã kiểm tra chẵn lẻ mật độ thấp phi nhị phân (NB-LDPC Non Binary Low Density Parity Check)
* Phần II: Thiết kế phần cứng bộ mã hóa và giải mã NB-LPBC

# MÃ KIỂM TRA CHẴN LẺ MẬT ĐỘ THẤP PHI NHỊ PHÂN (NB-LDPC NON BINARY LOW DENSITY PARITY CHECK)

## Giới thiệu về mã kênh NB-LDPC

### Mã kênh

Trong [Lý thuyết thông tin](https://vi.wikipedia.org/wiki/L%C3%BD_thuy%E1%BA%BFt_th%C3%B4ng_tin), **Định lý mã hóa trên kênh nhiễu** ([*tiếng Anh*](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh)*: noisy-channel coding theorem*) đề xuất rằng, cho dù một kênh truyền thông có bị ô nhiễm bởi nhiễu âm bao nhiêu đi chăng nữa, chúng ta cũng vẫn có thể truyền thông ([thông tin](https://vi.wikipedia.org/wiki/Th%C3%B4ng_tin)) dữ liệu số (*digital data*) không lỗi (*error-free*) tới một tỷ lệ tối đa nhất định qua một kênh truyền. Kết quả đáng ngạc nhiên này, đôi khi được gọi là *định lý nền tảng của lý thuyết thông tin* (*fundamental theorem of information theory*), hay chỉ đơn giản là *Định lý Shannon*, được giới thiệu lần đầu tiên bởi [Claude Shannon](https://vi.wikipedia.org/wiki/Claude_Shannon) vào năm [1948](https://vi.wikipedia.org/wiki/1948).

*Giới hạn Shannon* hoặc *Dung lượng Shannon* của một kênh truyền thông là tỷ lệ tối đa trên lý thuyết về lượng thông tin một kênh truyền thông có thể truyền tải, đối với một độ nhiễu nhất định.

Định lý được [Claude Shannon](https://vi.wikipedia.org/wiki/Claude_Shannon) chứng minh vào năm [1948](https://vi.wikipedia.org/wiki/1948), diễn tả hiệu quả tối đa mà [các phương pháp sửa lỗi](https://vi.wikipedia.org/w/index.php?title=M%C3%A3_s%E1%BB%ADa_l%E1%BB%97i&action=edit&redlink=1) (*error-correcting methods*) có thể đạt được, ngược lại mức độ nhiễu ô nhiễm và mức độ thoái hóa của dữ liệu (*data corruption*). Lý thuyết không diễn tả cách "làm thế nào để kiến tạo" một phương pháp sửa lỗi, song nó chỉ nói cho chúng ta biết mức độ hiệu quả có thể đạt được đối với một phương pháp "khả dĩ tốt nhất" nào đó mà thôi. Định lý Shannon có nhiều ứng dụng trên phạm vi rộng lớn, cả trong các ứng dụng truyền thông lẫn các ứng dụng về [lưu trữ dữ liệu](https://vi.wikipedia.org/w/index.php?title=Thi%E1%BA%BFt_b%E1%BB%8B_l%C6%B0u_tr%E1%BB%AF_d%E1%BB%AF_li%E1%BB%87u&action=edit&redlink=1) (*data storage applications*). Định lý này là nền tảng quan trọng đối với ngành [Lý thuyết thông tin](https://vi.wikipedia.org/wiki/L%C3%BD_thuy%E1%BA%BFt_th%C3%B4ng_tin) hiện đại.

Ngày nay, các dịch vụ trên mạng viễn thông gia tăng không ngừng trong khi nguồn tài nguyên của mạng viễn thông là hữu hạn. Vì vậy, việc khai thác nguồn tài nguyên của mạng viễn thông một cách hiệu quả là yêu cầu tiên quyết trong thiết kế hệ thống viễn thông số. Các phương pháp sửa lỗi từ lâu đã được nghiên cứu, đề xuất và cải tiến không ngừng. Phương pháp sửa lỗi đơn giản nhất như "kế hoạch gửi thông điệp 3 lần và dùng hai cái tốt nhất trong ba cái được bầu, nếu các bản sao khác nhau" là những phương pháp sửa lỗi vô hiệu quả (*inefficient*), không thể nào đảm bảo chắc chắn rằng một khối dữ liệu được truyền thông là không có lỗi.  Những kỹ thuật tân tiến sử dụng mã kênh như [Mã Reed-Solomon](https://vi.wikipedia.org/wiki/K%E1%BB%B9_thu%E1%BA%ADt_s%E1%BB%ADa_l%E1%BB%97i_Reed-Solomon), mã xoắn, [mã Turbo](https://vi.wikipedia.org/w/index.php?title=M%C3%A3_Turbo&action=edit&redlink=1) (Turbo code) đã đạt được gần đến giới hạn giả thuyết của Shannon, với cái giá phải trả là sự phức tạp lớn trong tính toán (at a cost of high computational complexity). Với mã Turbo và với công suất tính toán của các [bộ xử lý tín hiệu số](https://vi.wikipedia.org/w/index.php?title=B%E1%BB%99_x%E1%BB%AD_l%C3%BD_t%C3%ADn_hi%E1%BB%87u_s%E1%BB%91&action=edit&redlink=1) (digital signal processors) hiện nay, người ta có thể đạt được 1 dB đơn vị [decibel](https://vi.wikipedia.org/wiki/Decibel) của giới hạn Shannon.

### Mã LDPC

1. *Lịch sử*

Mã kênh LDPC nhị phân (B-LDPC gọi tắt là LDPC) được Gallager đề xuất vào năm 1963. Gallager cũng chứng minh được khả năng sủa lỗi của mã LDPC có thể đạt gần với giới hạn shanon (Trong thực tế không tính đến độ phức tạp, mã LDPC với độ dài rất lớn, sử dụng phương pháp giải mã BPA cho phép truyền tin với băng thông chỉ cách giới hạn shanon 0.0045bB, tốt hơn nhiều so với mã Turbo tốt nhất là khoảng 1 dB). Tuy nhiên, vào thời điểm đó, do năng lực tính toán của các chíp vi xử lý còn hạn chế, mã LDPC được xem là bất khả thi nên bị bỏ quên và không được đầu tư nghiên cứu. Mãi đến năm 1995, nhờ các công trình nghiên cứu của Mackey giúp giảm thiểu chi phí tính toán của bộ giải mã, giúp nó khả thi trong thực tế, mã LDPC đã được quan tâm trở lại và được đầu tư nghiên cứu trên toàn thế giới.

1. *Cấu trúc mã LDPC*

Đặc trưng của một bộ mã LDPC thể hiện qua ma trận kiểm tra chẵn lẻ của nó. Mã kiểm tra chẵn lẻ mật độ thấp (LDPC), thuộc họ mã tuyến tính, được đặt tên bởi ma trận kiểm tra **H**(*mxn*) của nó có đặc tính thưa, nghĩa là có rất ít các phần tử trên ma trận **H** là khác 0. Ba tham số quan trọng của bộ mã là độ dài từ mã n, độ dài bản tin k, và số bít kiểm tra *m* = *n* – *k*. Số phần tử khác không trên 1 hàng gọi là trọng số hàng dc (*dc* <<*n*), số phần tử khác không trên một cột gọi là trọng số cột dv (*dv*<<*m*). Ma trận kiểm tra được dùng để kiểm tra một từ mã có là từ mã hợp lệ hay không, nếu từ mã là không hợp lệ có nghĩa là từ mã đã bị thay đổi trên đường truyền và cần phải sửa lại. Một từ mã **C** là hợp lệ khi **C**\***H**=0.

Có thể chia mã LDPC thành 2 lớp cơ bản:

• Nếu tất cả các hàng trên ma trận kiểm tra H có cùng trọng số hàng dc và tất cả các cột có cùng trọng số cột dv thì bộ mã được gọi là quy tắc.

• Ngược lại nếu trọng số trên các hàng và cột của ma trận **H** là khác nhau thì bộ mã được gọi là bất quy tắc.

Thông thường, các mã LDPC bất quy tắc cho chất lượng kiểm soát mã tốt hơn mã LDPC quy tắc. Một ví dụ về ma trận kiểm tra chẵn lẻ **H** của mã LDPC quy tắc:

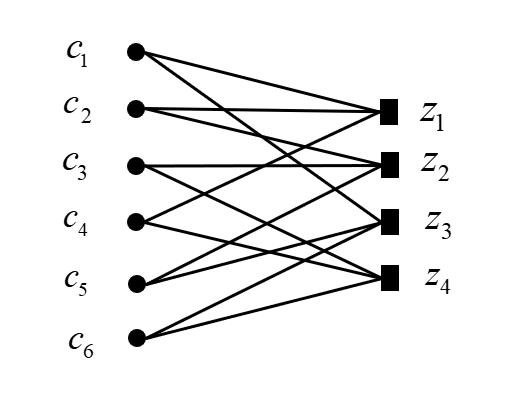


(1.1)

Chúng ta có thể thấy, mã LDPC trên có trọng số ở các hàng là *dc* =3, và trọng số ở các cột là *dv* =2.

1. *Biểu diên ma trận H trên giản đồ Tanner*

Một ma trận kiểm tra chẵn lẻ còn có thể biểu diễn dưới dạng một giản đồ 2 phía hay còn gọi là giản đồ Tanner. Mỗi hàng trên ma trận kiểm tra **H** biểu diễn một phương trình kiểm tra chẵn lẻ  với 1 ≤ *i* ≤ *m*, mỗi cột trên ma trận H biểu diễn một bít của từ mã  với 1 ≤ *j* ≤ *n*. Trên ma trận **H** (1.1), hàng đầu tiên của ma trận thể hiện phương trình kiểm tra : . Giản đồ Tanner biểu diễn ma trận kiểm tra **H**  bằng các nút kiểm tra (Check Node – CN) thay cho các hàng và các nút biến (Variable Node VN) thay cho các cột, các phần tử khác 0 của ma trận được biểu diễn bằng các liên kết giữa CN và VN. Ví dụ giản đồ Tanner cho ma trận kiểm tra **H** (1.1):



**Hình 1.1**. Giản đồ Tanner của ma trận **H** (1.1)

Biểu diễn ma trận **H** bằng giản đồ Tanner còn cho phép chúng ta dễ dàng phát hiện các *vòng* (short cycle), là 1 đường bắt đầu ở 1 node biến và kết thúc tại chính node đó (do đó, 1 vòng có độ dài tối thiểu là 4). Các vòng này ảnh hưởng đến chất lượng của bộ mã, vòng có độ dài càng ngắn càng làm xấu chất lượng bộ mã, đặc biệt là các vòng 4 hoặc 6. Ví dụ 1 vòng trên giản đồ là là một vòng có độ dài bằng 6.

### Mã LDPC phi nhị phân

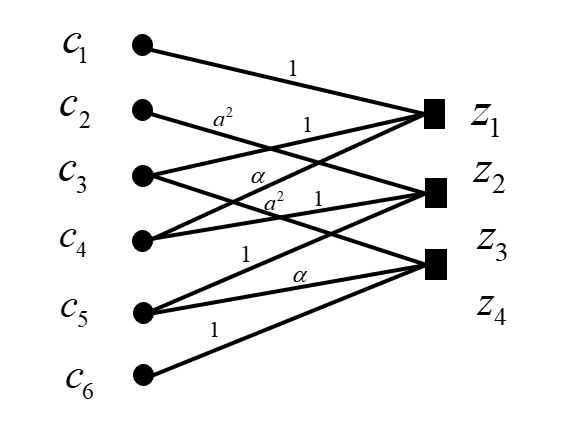
Mã NB-LDPC đơn giản là các mã B-LDPC với ma trận kiểm tra chẵn lẻ có nó có các phần tử khác không được định nghĩa trên các *nhóm*, các *vòng* hoặc *trường* bậc cao. Trong nghiên cứu này, tác giả chỉ tập trung nghiên cứu lớp mã NB-LDPC được định nghĩa trên các trường Galois giới hạn (finite fields) GF(2i), là các số dương có thể lơn hơn 1. Mã NB-LDPC được Mackey giới thiệu năm 1998, đồng thời ông đã chứng minh được khả năng kiểm soát lỗi vượt trội của nó so với các mã B-LDPC ngay cả với các bộ mã có độ dài từ mã ngắn.

Ví dụ một ma trận kiểm tra của mã NB-LDPC trên trường GF(4) gồm các phần tử {0, 1, , } như sau:



(1.2)

Giản đồ Tanner biểu diễn ma trận kiểm tra của mã NB-LDPC cũng tương tự như với mã B-LDPC chỉ khác là các đường nối các nút biến và nút kiểm tra có thêm trọng số tương ứng.



**Hình 1.2** Giản đồ Tanner của ma trận (1.2)

## Phương pháp xây dựng bộ mã NB-LDPC

### Các thuật toán xây dựng ma trận kiểm tra H

Một mã LDPC được xây dựng từ một ma trận kiểm tra chẵn lẻ. Bài toán đặt ra là, biết trước các tham số về độ dài bản tin k, độ dài từ mã n và số bít kiểm tra *m* = *n*- *k*, làm thế nào để tạo ra được ma trận kiểm tra **H**(*mxn*)có chất lượng ( liên quan đến khả năng kiểm soát lỗi của bộ mã) tốt nhất. Rất khó để đánh giá một ma trận kiểm tra **H** cho trước có chất lượng thế nào nếu không thực hiện mã hóa, giải mã và chạy mô phỏng kiểm tra cụ thể nó. Nhưng việc xây dựng bộ mã hóa, giải mã từ một ma trận **H** và chạy mô phỏng kiểm tra nó là công việc rất phức tạp và tốn thời gian. Tuy nhiên, một số tham số đặc trưng của ma trận **H** có thể giúp chúng ta đánh giá sơ bộ chất lượng của nó bao gồm: tốc độ mã *r = k/n* càng thấp, khả năng sửa lỗi của mã càng cao nhưng làm giảm hiệu quả sử dụng băng thông; trọng số hàng dc càng lớn cũng cho chất lượng tốt hơn nhưng đánh đổi bởi độ phức tạp trong việc mã hóa và giải mã; các vòng ngắn (short cycle) như vòng 4, vòng 6, vòng 8... làm mất sự độc lập tuyến tính giữa các nút kiểm tra dẫn đến sự xuất hiện sàn lỗi (error floor) và làm giảm chất lượng bộ mã; mã LDPC quy tắc cũng có chất lượng kém hơn mã LDPC bất quy tắc.

Phương pháp sinh ma trận kiểm tra **H** bên cạnh các thuật toán giải mã và phương pháp thiết kế bộ giải mã là những vấn đề chính đang được tập trung nghiên cứu của mã LDPC.

1. *Phương pháp sinh ngẫu nhiên (Random Initialization)*

Phương pháp sinh ngẫu nhiên như tên gọi của nó, ma trận kiểm tra **H** được tạo ra bằng cách lựa chọn các tham số *m, n, dc* và một vài ràng buộc nào đó rồi được khởi tạo một cách ngẫu nhiên. Để tăng cường chất lượng, người ta thường sử dụng thêm các thuật toán tìm kiếm vòng ngắn để loại bỏ chúng. Phương pháp tạo ma trận kiểm tra này kết quả thường là các mã bất quy tắc, và ma trận H không có cấu trúc cụ thể. Nếu loại bỏ được hết các vòng, các ma trận này có chất lượng rất tốt. Mặc dù thế, phương pháp này vẫn gặp khó khăn khi nghiên cứu do thuật toán tìm kiếm vòng có độ phức tạp tính toán cao nên chỉ dùng để loại bỏ các vòng nhỏ như vòng 4, vòng 6, vòng 8, các vòng có độ dài lớn hơn mặc dù ít ảnh hưởng đến chất lượng mã hơn nhưng vẫn làm bộ mã bị suy giảm chất lượng. Bên cạnh đó, do không thể xác định rõ số vòng trên ma trận nên chúng ta cũng chỉ có thể đánh giá một cách ước lượng chất lượng của bộ mã. Hơn nữa, việc bộ mã là không quy tắc cũng gây khó khăn cho việc xây dựng thuật toán mã hóa, giải mã.

1. *Phương pháp sử dụng kỹ thuật phân rã mảng (Array Dispersion)*

Phương pháp xây dựng ma trận kiểm tra bằng kỹ thuật phân rã mảng[1] (Array Dispersion) được Bo Zhou, Shu Lin và các cộng sự đề xuất năm 2009. Trong các nghiên cứu của mình, Shu Lin cùng nhóm tác giả đã trình bày hàng loạt các phương pháp tạo ma trận sử dụng kỹ thuật phân rã như Euclidean và *hình học giới hạn* (finite geometries) cũng như sử dụng một từ mã đơn trong mã mật độ rất thấp Reed-Solomon[2]. Do nghiên cứu này tập trung vào các mã NB-LDPC nên tác giải chỉ trình bày các cấu trúc mã NB-LDPC từ một từ mã Reed-Solomon.

Phương pháp xây dựng ma trận kiểm tra bằng kỹ thuật phân rã mảng trên 1 trường giới hạn GF(*q*) bao gồm 3 bước:

* Bước 1: Chọn một từ mã Reed-Solomon có độ dài *q*-1. Từ mã này có trọng số là q-2 và các phần tử của từ mã không giống nhau, ví dụ:



* Bước 2: Xây dựng ma trận cơ sở **B** từ từ mã trên bằng cách các phép dịch vòng bên phải (RCS- Right Cyclic Shift) với hàng đầu tiên là từ mã Reed-Solomon đã trọn và các hàng sau được tính theo công thức:



Ví dụ ma trận cơ sở của một mã LDPC trên GF(8):



(1.3)

* Bước 3: Phân rã từng phần tử của ma trận cơ sở, từ 1 phần tử sẽ được phân rã thành ma trận con kích thước (*q*-1)x(*q*-1). Cuối cùng thu được ma trận có kích thước (*q*-1)2x(*q*-1)2. Ví dụ kết quả phân rã phần tử :



(1.4)

Ma trận cơ sở được cấu trúc như trình bày ở trên có thể chứng minh được là không tồn tại vòng (đối với mã NB-LDPC một vòng được tính khi các trọng số trên các đường kết lối phải là bằng nhau). Tiếp theo đó, kỹ thuật phân rã như trên cũng được chứng minh là không tạo ra thêm vòng, do đó ma cuối cùng thu được sẽ không có vòng. Cuối cùng, tùy vào bộ mã cần thiết kế, ta chọn ma trận kiểm tra là một phần của ma trận phân rã trên.

### Mã hóa bản tin từ ma trận kiểm tra H

1. Mã hóa bản tin bằng ma trận sinh G

Do mã LDPC thuộc họ mã khối tuyến tính, nên việc mã hóa một bản tin có thể đạt được bằng cách nhân vector bản tin với ma trận sinh **G**. Ma trận sinh **G** được tạo ra bằng phương pháp khử Gauss-Jordan ma trận **H** về dạng **H =** [**Im|P**], với **I**m là ma trận đơn vị kích thước *mxm*, **P** là ma trận chẵn lẻ. Ma trận sinh thu được **G =** [**PT|Ik**], với **PT**  là ma trận chuyển vị của **P**, và **Ik** là ma trận đơn vị kích thước kxk. Với định nghĩa như trên, ta luôn có **G\*H** = **0.**

**Ví dụ:** Ta có ma trận **H** như sau:

 (1.5)

Có thể thấy, hàng 4 là tổng module-2 của hàng 1, 2 và 3. Do đó, hàng 4 là không có giá trị trong kiểm soát lỗi (nếu từ mã thỏa mãn phương trình kiểm tra của hàng 1, 2 hoặc 3 thì tất nhiên cũng thỏa mãn 4). Vì thế, nếu **H** chưa độc lập tuyến tính với nhau, ta cần loại bỏ bớt các hàng của nó để các hàng còn lại đọc lập tuyến tính.

 (1.6)

Tiếp theo, sử dụng phép khử Gauss-Jordan để đưa **H** về dạng **H =** [**Im|P**].

 (1.7)

Từ đó, ta thu được ma trận sinh **G.**

 (1.8)

Giả sử bản tin cần mã hóa là **U** = [1, 0, 1]

Từ mã sau khi mã hóa bản tin là **C** = **U**\***G** = [1, 0, 0, 1, 0, 1].

Ta có **C\*H**= **0**

Có thể thấy, phương pháp mã hóa má bản tin bằng ma trận sinh có một nhược điểm là: Ma trận sinh G không còn giữ được tính chất ‘thưa” của H, do đó nó gây phức tạp trong việc thiết kế bộ mã hóa. Độ phức tạp của mã hóa gần tương đương 0n với n là độ dài từ mã. Đối với các mã LDPC có độ dài từ mã lớn thì bộ mã hóa sẽ trở lên cực kỳ phức tạp.

1. Mã hóa bản tin trực tiếp từ ma trận khiển tra H

Có một cách tiếp cận tốt hơn cho mọi ma trận kiểm tra **H** là không xây dựng ma trận G, mà thay vào đó là thực hiện mã hóa ngay bằng ma trận kiểm tra H nhờ phép biến đổi về dạng tam giác dưới và thay thế ngược trở lại. Ý tưởng của phương pháp mã hóa dùng ma trận kiểm tra **H** là sử dụng chủ yếu các hoán vị hàng và cột, sao cho vẫn giữ được tính chất thưa của ma trận H. Trước hết, chỉ hoán vị hàng và cột để đưa ma trận **H** về ma trận **Ht**, dạng gần như tam giác dưới.



trong đó, **T** là ma trận tam giác dưới, nghĩa là, **T** có các giá trị 1 trên đường

chéo từ trái qua phải, các phần tử ở trên đường chéo bằng 0, và **T** có kích thước (*m*−*g*)×(*m*−*g*). Nếu ma trận **Ht** có hạng đầy đủ thì **B** là ma trận kích thước (*m*−*g*)×*k* và **A** là ma trận kích thước (*m*−*g*)×*k*. Các ma trận **C**, **D**, **E** có g hàng, và g được gọi là phần khuyết, nếu *g* càng nhỏ thì độ phức tạp của mã hóa càng thấp.

**Ví dụ** : Thực hiện mã hóa bản tin u = [1 1 0 0 1], với chiều dài từ mã bằng 10, tỉ lệ mã hóa bằng 1/2, và ma trận kiểm tra **H**:

 (1.9)

Để đưa ma trận kiểm tra **H** về dạng gần tam giác dưới, thực hiện hoán vị hàng 2 với hàng 3; cột 6 với cột 10, và chọn g bằng 2. Khi đó, kết quả nhận được là ma trận **Ht**.

 (1.10)

Quá trình định dạng tam giác dưới, phép khử Gauss-Jordan được dùng một lần, tương đương với việc nhân ma trận **Ht** với ma trận .

 (1.11)

Trong đó, **C2 = - ET-1A + C; D2 = -ET-1B + D.**

Ở ví dụ đang xét t có:

 (1.12)

Suy ra:

 (1.13)

Khi thực hiện phép khử Gauss-Jordan để xóa **E** thì chỉ có **C2**, **D2** bị ảnh hưởng, còn các phần khác của ma trận kiểm tra vẫn giữ nguyên đặc tính thưa.

Cuối cùng, để mã hóa bản tin sử dụng ma trận **Htd’** thì từ mã **c** = [c1,c2,...,cn**]** được chia thành **c** = [**u**, **p1**, **p2**], với **u** = [u1,u2,...uk]là k bit thông tin, **p1** là g bit kiểm tra đầu và **p2** =[p21, p22,...p2*m*−*g*] là các bit kiểm tra còn lại. Từ mã **c** = [**u**, **p1**, **p2**] phải thỏa mãn phương trình kiểm tra chẵn lẻ **C\*H**= **0**. Do đó:



(1.14)

Vì ma trận **E** đã được khử thành toàn 0, các bit kiểm tra **p1** chỉ còn phụ thuộc vào các bit thông tin có thể tính độc lập. Nếu D2 là nghịch đảo, thì **p1**, **p2** được tính theo công thức:



. (1.15)

Các ma trận A, B và T rất thưa, do đó độ phức tạp của phương trình này là rất thấp.

Trong ví dụ này:

 (1.16)



Vậy: **C** = [1 1 0 0 1 1 0 1 0 0].

## Thuật toán quyết định cứng (Hard Dicision)

Bản tinh sau khi được mã hóa sẽ thu được từ mã. Từ mã này được đưa đến bộ điều chế và được truyền đi trên kênh truyền. Ở phía máy thu, tín hiệu được giải điều chế, với máy thu tối ưu (biết đặc tính kênh truyền), ta thu được xác suất lợp lẽ hậu nghiệm của từng symbol. Với thuật toán quyết định cứng, ta sẽ quyết định giá trị symbol dựa trên xác suất hậu nghiệm lớn nhất. Bằng cách tìm symdrone **S** = **C**\***H**, và sử dụng bảng tra để tìm lỗi ứng với **S**, ta sẽ biết được lỗi nằm ở đâu và sửa từ mã lại cho đúng. Phương pháp này chỉ có thể phát hiện và sửa các lỗi có số bit lỗi hạn chế và yêu cầu một bảng tra rất lớn. Với độ dài từ mã là *n*, số bít lỗi tối đa sửa được là *f,* bảng tra phải chứa  syndrome, tức là 2*n*\*  bits nhớ. Đây là một số rất lớn nên phương pháp này không khả thi với mã LDPC.

## Thuật toán quyết định mềm – giải mã lặp (Soft Dicision)

Thuật toán quyết định mềm hay giải mã lặp là phương pháp sử dụng các phương trình kiểm tra của ma trận kiểm tra để tính toán lại xác suất hợp lẽ của các symbols, thuật toán quyết định mềm có cơ sở toán học là *thuật toán lan truyền niềm tin* BPA*.* Về ý tưởng, thuật toán này sử dụng các phép lặp để tính toán lại xác suất của các symbol trong từ mã, mỗi một symbol sẽ được cập nhật xác xuất tin cậy hơn nhờ vào các thông tin của symbols khác, cứ thế sau nhiều lần *lặp,* xác suất hợp lẽ của chúng sẽ hội tụ và thu được từ mã phù hợp nhất. Người ta chứng minh được rằng, nếu trên ma trận **H** không tồn tại các vòng kín thì từ mã cuối cùng thu được sẽ luôn là một từ mã thỏa mãn **C\*H**= **0** .

1. *Thuật toán giải mã Sum-Product*

Thuật toán Sum-Product là triển khai của thuật toán lan truyền niềm tin đối với mã LDPC. Thuật toán này được Gallager trình bày năm 1963 và là thuật toán có chất lượng giải mã tốt nhất, tuy nhiên nó yêu cầu độ phức tạp tính toán rất lớn đến nỗi không khả thi trên phần cứng thực tế. Các thuật toán đề xuất sau này như Min-Sum, Min-Max đều có cở sở từ thuật toán Sum-Product, tuy nhiên đã được biến đổi một phần để giảm bớt sự phức tạp và phù hợp với phần cứng hơn, đánh đổi của các thuật toán sau này là chất lượng giải mã bị giảm, tuy nhiên sự suy giảm này là nhỏ và có thể chấp nhận được. Do là thuật toán có chất lượng giải mã tốt nhất, thuật toán Sum-Product vẫn được dùng để đánh giá chất lượng một bộ mã.

Trong trương này, chúng ta sẽ trình bày lần lượt 3 thuật toán tiêu chuẩn để giải mã LDPC là Sum-Product, Min\_Sum, Min-Max. Để thuận tiện, ta thông nhất ký hiệu như sau:

**Kí hiệu cho các thành phần thuộc trường Galois:**

* GF(q) = {0, 1, ...., q-1}, trường Galois với q phần tử. Mỗi phần tử của trường sẽ được gọi là symbol, để phân biệt với các số nguyên thông thường.
* a, s, x được dùng để kí hiệu cho các symbol trên GF(q).
* **a, s, x** được dùng để kí hiệu cho các vector trên trường GF(q). Ví dụ **a**  = (a1, a2, a3... ) GF(q).

**Kí hiệu cho bộ mã LDPC:**

* H MM,N(GF(q)), là ma trận kiểm tra của bộ mã
* , tập các từ mã hợp lệ của bộ mã
* , tập các từ mã với symbol thứ n bằng a.
* x = (x1, x2, x3 ...., xN) là từ mã truyền qua kênh.

**Kí hiệu cho giản đồ Tanner:**

Giản đồ Tanner tương ứng của bộ mã chứa N node biến và M node kiểm tra, biểu diễn cho M hàng và N cột của ma trận **H.**

* H, giản đồ Tanner của bộ mã.
* *n* {1, 2, ..., N} là một node biến của H.
* *m* {1, 2,..., M} là một node kiểm tra của H.
* H(*n*), tập hợp các node kiểm tra nối với node biến *n*.
* H(*m*), tập hợp các node biến nối với node kiểm tra *m*.
* L(*m*), tập các cấu hình cục bộ (local configurations) thỏa mãn node kiểm tra m; vd: tập hợp các từ mã a = , thỏa mãn phương trình kiểm tra của node m:

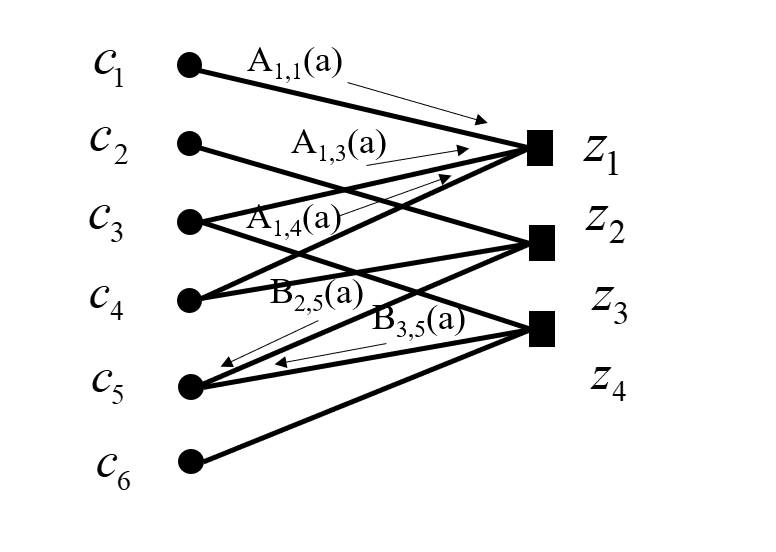


* L(*m|an*=a), tập các cấu hình cục bộ a thỏa mãn m, với an=a.

**Kí hiệu cho thuật toán lặp:**

Một thuật toán giải mã lặp bao gồm bước khởi tạo và sau đó là các bước lặp nhằm cập nhật thông tin giữa các node biến và node kiểm tra liên kết với nhau trên giản đồ Tanner.

* , thông tin tiên nghiệm (priori information) của node biến thứ n với symbol a.
* , thông tin hậu nghiệm (posteriori information) của node biến thứ n với symbol a.
* Am,n(a), thông tin gửi từ node kiểm tra m đến node biến *n*.
* Bm,n(a), thông tin từ node biến *n* đến mode kiểm tra m với symbol a.

**

**Hình 1.3** *Biểu diễn trao đổi thông tin giữa node biến*

*và node kiểm tra trên đồ thị Tanner*

\* *Thuật toán giải mã Sum-Product*:

**Khởi tạo:**

* Thông tin tiên nhiệm :  = Pr(xn=a|channel)
* Bản tin của node biến: Am,n(a) = 

**Lặp:**

* Xử lý ở node kiểm tra:

 (1.17)

* Xử lý ở node biến:

 (1.18)

* Tính toán thông tin hậu nghiệm:

 (1.19)

**Nhận xét:**

* Ở các vòng lặp, công đoạn xử lý ở node kiểm tra (check node processing) là phức tạp nhất và là điểm cổ chai của bộ giải mã, các thuật toán sau này đều cố gắng đẩy nhanh tốc độ ở bước này. Ở bước này, ta phải tìm tất cả các cấu hình cục bộ thỏa mãn một node biến, số lượng cấu hình cục bộ tăng theo cấp số nhân với trọng số hàng của của phương trình kiểm tra đó và số phần tử thuộc trường GF(q).
* Thuật toán được đặt tên là Sum-Product do sử dụng các phép cộng và nhân ở bước xử lý node kiểm tra. Các phép toán này được thực hiện trên trường số thực nên không phù hợp cho thiết kế mạch trong thực tế, đặc biệt là các bộ nhân số thực yêu cầu nhiều về tài nguyên cổng logic.
* Thuật toán được Mackey cải tiến bằng các thay các phép nhân bằng phép tính Furrier đã giảm bớt được rất nhiều độ phức tạp tính toán. Tuy nhiên, để hiểu được bản chất của thuật toán nên ở đây chúng ta vẫn xử dụng công thức gốc.

1. *Thuật toán giải mã Min-Sum*

Thuật toán Min-Sum được phát triển từ thuật toán Sum-Product, các thông tin( xác suất tiên nhiệm, hậu nhiệm) được chuyển về dạng logarith để thay thế các phép nhân bằng phép cộng. Phép cộng được thay thế bằng phép lấy min. Các phép nhân là tương đương với phép cộng ở dạng logarith tuy nhiên phép lấy min không tương đương với phép cộng ở dạng xác suất bình thường, vì thế thuật toán này có chất lượng thấp hơn so với thuật toán Sum-product ban đầu, sự suy giảm chất lượng này là không đáng kể, người ta chứng minh được rằng nếu không tồn tại các vòng trên ma trận kiểm tra thì thuật toán cũng cho giá trị hội tụ về kết quả tối ưu như thuật toán Sum-Product.

Cụ thể thuật toán Min-Sum:

**Khởi tạo:**

* + Thông tin tiên nhiệm :  = -ln(Pr(xn=a|channel))
  + Bản tin của node biến: Am,n(a) = 

**Lặp:**

* Xử lý ở node kiểm tra:

 (1.20)

* Xử lý ở node biến:

 (1.21)

* Tính toán thông tin hậu nghiệm:

 (1.22)

**Nhận xét:**

* Các phép toán cộng có thể thực hiện dễ dàng trên mạch logic, các thông tin là số thực thường được biểu diễn trên mạch ở dạng số thực dấu phẩy tĩnh để giảm độ phức tạp của phép cộng.
* Thuật toán được gọi là Min-Sum do sử dụng phép lấy Min và phép cộng ở bước xử lý node kiểm tra.
* Tác giả của thuật toán Min-Sum cũng đề xuất một giải pháp nữa nhằm giảm độ phức tạp ở bước xử lý node kiểm tra là: Thay vì phải tính toán tất cả các cấu hình cục bộ, ta chỉ xử lý một phần nhỏ các trường hợp đem lại nhiều thông tin nhất. Chi tiết về ý tưởng này sẽ được trình bày trong **phần II**-thiết kế bộ giải mã NB-LDPC với thuật toán Min-Max.

1. *Thuật toán giải mã Min-Max*

Thuật toán giải mã Min-Max được xây dựng trên cơ sở thuật toán Min-Sum. Tư tưởng của thuật toán là: Để tính toán xác suất của một cấu hình cục bộ, thay vì lấy tổng các thông tin của các node biến liên quan (trên miền logarith-tương đương với tích các xác xuất xảy ra của từng phần tử trong cấu hình), thì ta chỉ lấy Max của các thông tin đó. Phép tìm Max của 2 số thực là đơn giản và có độ trể thấp hơn phép cộng, điều này đẩy nhanh tốc độ của thuật toán.

* Xử lý ở node kiểm tra:

 (1.23)

# THIẾT KẾ KHỐI GIẢI MÃ LAYERED TRELLIS MIN-MAX DECODER TRÊN PHẦN CỨNG FPGA

Ở phần này, chúng ta sẽ thiết kế khối giải mã của một bộ mã NB-LDPC. Do hạn chế về số lượng cổng logic trên bo mạch sparta 6, nên bộ mã chỉ được xây dựng trên trường GF(8), độ dài từ mã ngắn. Việc thiết kế khối giải mã đối với các trường GF bậc cao hơn gần như tương tự trên GF(8). Cụ thể mã NB-LDPC sẽ thiết kế có một số thông số quan trọng như sau:

* Bộ mã được xây dựng trên trường GF(8)
* Độ dài từ mã n= 35, độ dài bản tin k = 23, số bit kiểm tra m = n-k = 12, tốc độ mã r = k/n  0.66.
* Ma trận kiểm tra của mã được xây dựng theo phương pháp phân rã ma trận như trình bày ở mục 2.1.2.
* Bản tin sẽ được mã hóa trực tiếp bằng ma trận **H** như trình bày ở mục 2.2.2. Tác giả sử dụng một số thuật toán tham lam để biến đổi ma trận **H** về dạng ma trận tam giác dưới sau đó mã hóa theo công thức của thuật toán.

## Thiết kế khối giải mã Layered Trellis Min-Max Decoder

Kiến trúc Layered Trellis Min-Max Decoder [3] được đề xuất vào năm 2014, đạt được băng thông 660Mb/s với số vòng nặp cực đại là 9. Bộ giải mã này cho hiệu quả sử dụng phần cứng tốt nhất so với các thiết kế trước đó và trở thành một tiêu chuẩn trong thiết kế khối giải mã NB-LDPC. Đúng với tên gọi của mình, kiến trúc của bộ giải mã là sự kết hợp của 3 nghiên cứu quan trọng trước đó liên quan đến việc xây dựng khối giải mã NB-LDPC đó là: Min-Max-thuật toán giải mã được sử dụng; Trellis-Phương pháp triển khai thuật toán dưới dạng lưới (sẽ được trình bày trong mục sau); Layered- Layered Schedule thay cho Floor Schedule, bộ giải mã sẽ xử lý lý lần lượt ở từng node kiểm tra và đùng kết quả cập nhật mới nhất để xử lý ở node tiếp theo thay vì xử lý toàn bộ các node kiểm tra rồi mới cập nhật. Điều này trên thực tế làm giảm số vòng lặp cần thiết để kết quả hội tụ.

### Thuật toán xử lý node kiểm tra Trellis Min-Max

**Thuật toán:**

**Input:** Am,n , 

**1.****for**   **do**



**end**

**2.** 

**3.** 

**4.** 

**5. for**  **do**

**if**  **and**  **then**



**else if**  **then**



**else**



**End**

1. Mô tả thuật toán

**Input:** đầu vào của thuật toán là thông tin tiên nhiệm của các node biến *nj* ở miền logarith chuẩn hóa , có kết nối đến node kiểm tra *m*  và vị trí symbol có giá trị nhỏ nhất (tương ứng với xác suất xảy ra là lớn nhât). Ví dụ:

 (2.1)



**1.** Ở bước đầu tiên, tất cả thông tin được chuyển sang miên delta, kí hiệu , với  . Việc chuyển đổi này nhằm đảm báo tất cả các thông tin tin cậy nhất (bằng 0) đều nằm ở vị trí đầu tiên của .

 (2.2)

**2**. Bước 2 dùng để tính toán syndrome  ứng với các thông tin tin cậy nhất *zn.* Việc tính syndrome ở bước này sẽ coi như mọi phần tử khác không trên ma trận kiểm tra **H** là bằng 1 vì thông tin trước khi đi vào khối xử lý node kiểm tra đều được hoán vị theo giá trị phần tử đó.

 (2.3)

**3.** Thủ tục  sử dụng trên từng hàng của để tìm ra giá trị phần tử nhỏ nhất m1(a), giá trị phần tử nhỏ thứ 2 m2(a) và vị trí giá trị phần tử nhỏ nhất m1col(a).

(2.3)

**4.** Mỗi một đường liên kết từ m1 đến m4 đi qua các symbol trên bảng được gọi là 1 cấu hình (confm) của node *m,* ở đây ta chỉ xét các cấu hình có tối đa 2 điểm khác 0 (vị trí hàng đầu tiên), mỗi vị trí khác không là giá trị nhỏ nhất khác không thuộc hàng đó (conf\*(1, 2)).Cấu hình gồm toàn các điểm 0 ở hàng đầu được gọi là cấu hình bậc 0 (zero order configuration).

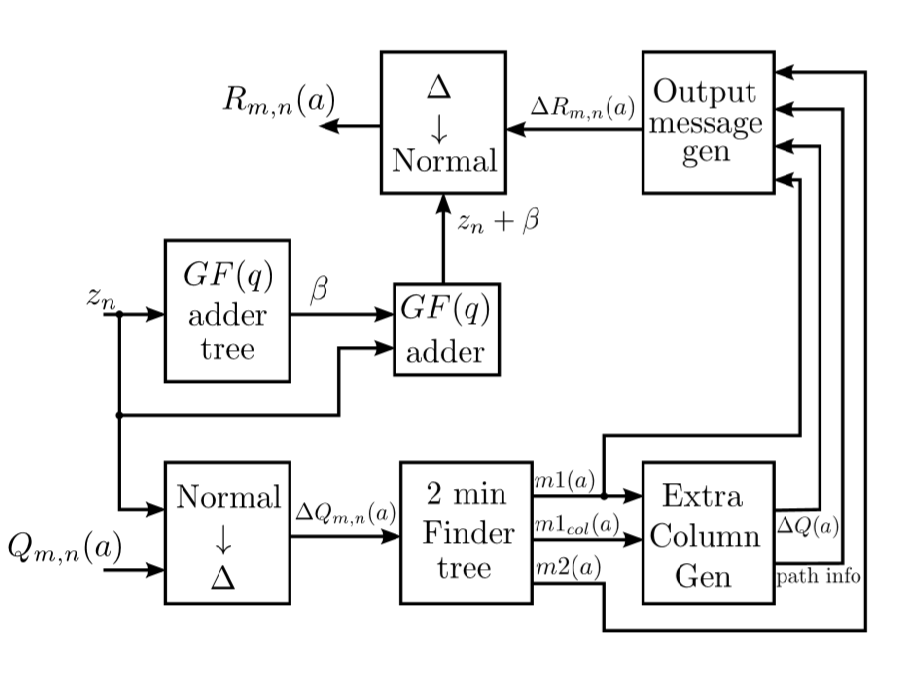
(2.4)

Việc mô tả các cấu hình dưới dạng các lưới được gọi là Trellis (lưới). Ở bước 4, thông tin lấy được ở các cấu hình là giá trị max của các vị trí bản tin khác 0, như đã trình bày trong thuật toán Min-Max ở chương trước.

**5.** Tính toán giá trị node kiểm tra như trong thuật toán. Công thức tính giá trị cập nhật của node kiểm tra ở bước này đã được cải tiến so với thuật toán gốc được đề xuất ở trong [3], công thức mới này giúp chúng ta có thể xem xét toàn bộ các cấu hình cục bộ con (1, 2) mà còn thêm một số cấu hình cục bộ con (2, 2) tuy nhiên không làm tăng yêu cầu về phần cứng quá lơn. Việc này thực hiện được nhờ vào bước 3 thay vì chỉ tìm giá trị min của mỗi hàng ta còn tìm cả giá trị nhỏ thứ 2.

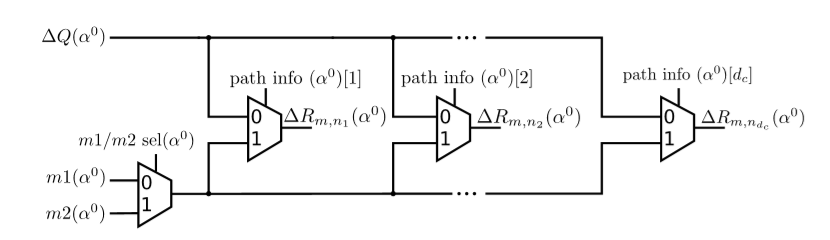
b, Thiết kế khối xử lý node kiểm tra (CN) trên phần cứng

Dựa trên thuật toán đã trình bày ở phần a, cấu trúc của khối CN được thiết kế như sau:

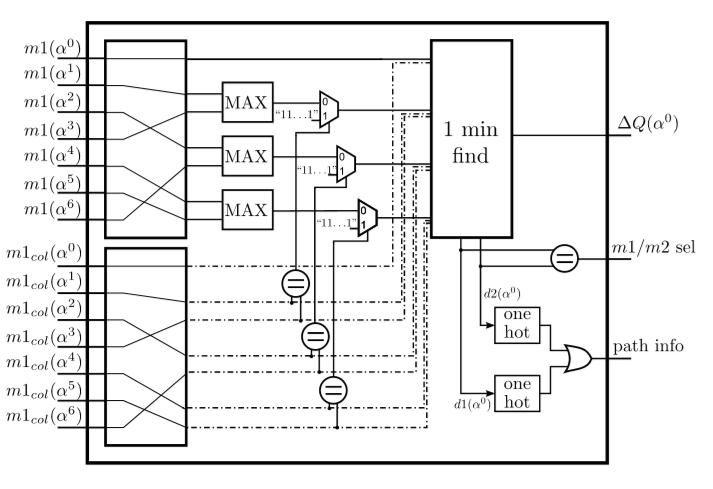


**Hình 2.1** Cấu trúc tổng quan của khối CN

* ***Khối Output Message Gen cho 1 symbol***

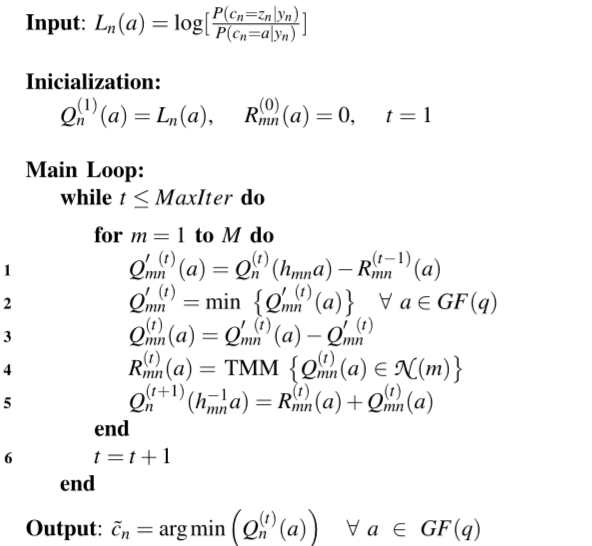


* ***Khối Extra Column Gen*** (ví dụ cho một symbol trên trường GF(8))

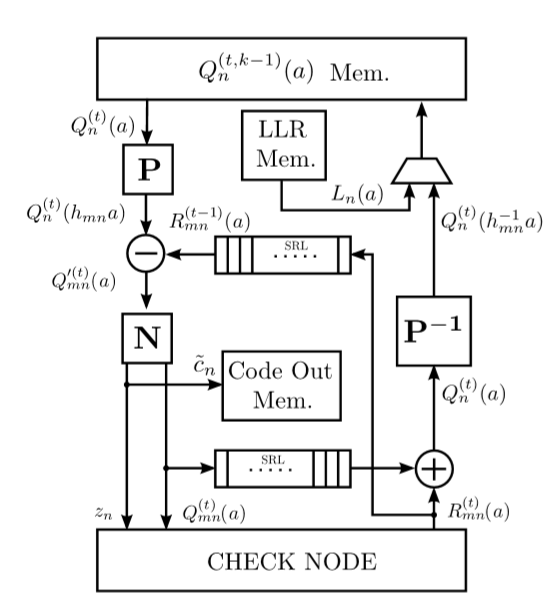
******

### Thuật toán Layered Schedule cho bộ giải mã

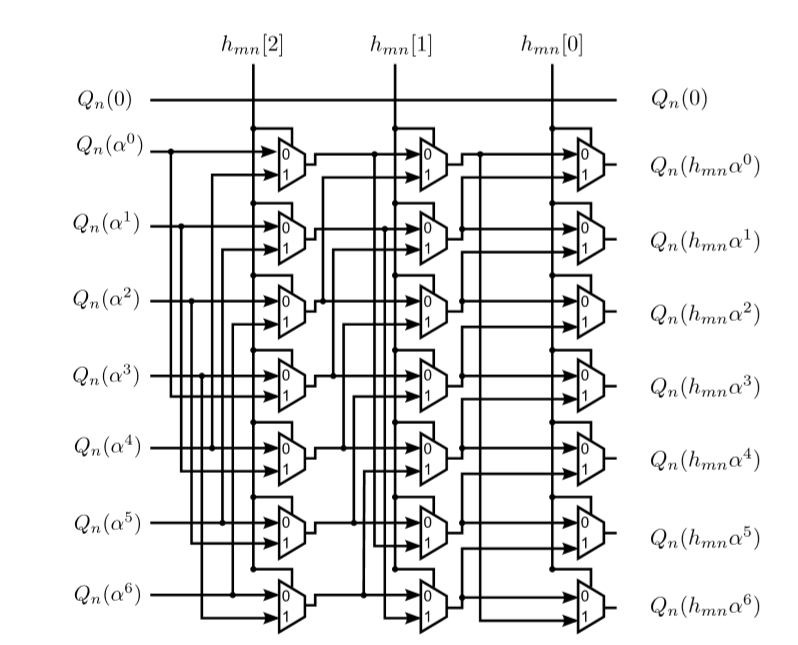
1. Thuật toán

****

1. Thiết kế trên phần cứng

******

***Hình 2.2*** *Tổng quan khối giải mã*

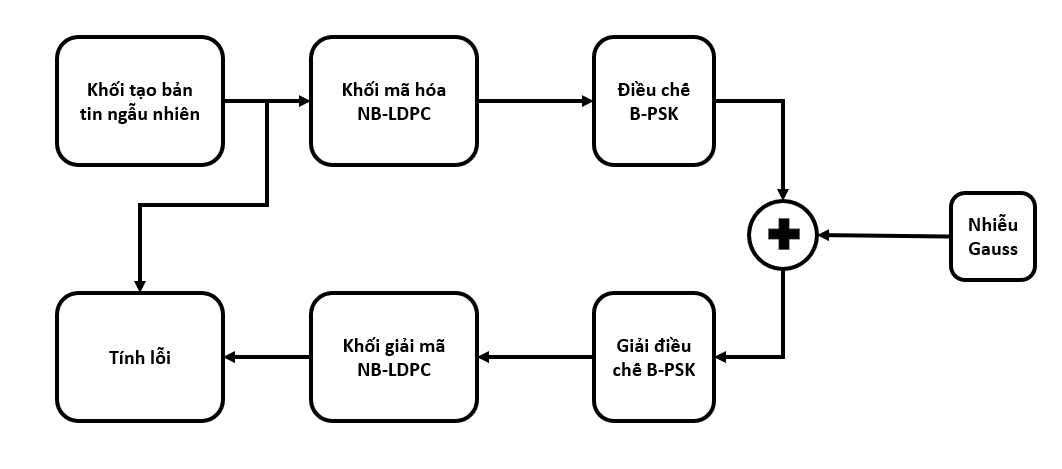
******

***Hình 2.3*** *Khối hoán vị* ***P***

## Kết quả mô phỏng đánh giá

### Kết quả mô phỏng thuật toán trên phần mềm

Thuật toán được viết và chạy mô phỏng trên ngôn ngữ python để đánh giá chất lượng. Sơ đồ mô phỏng như sau:



***Hình 2.3*** *Sơ đồ mô phỏng đánh giá chất lượng*

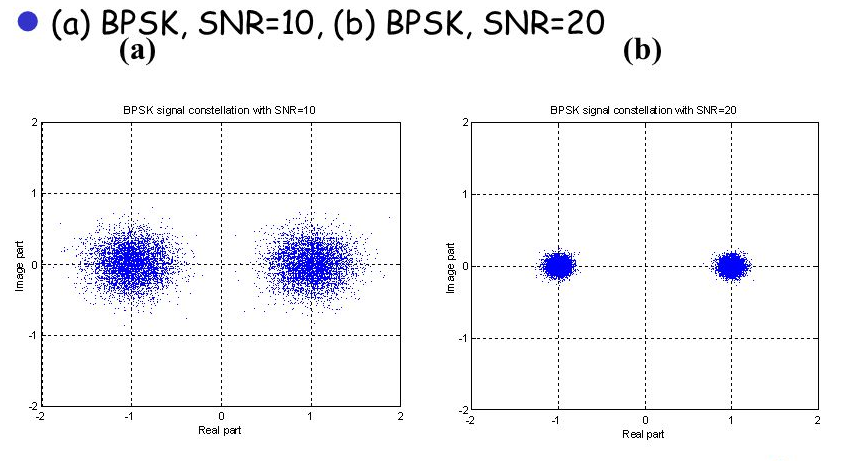
*thuật toán giải mã (phần mềm)*

Mô phỏng được thiết kế theo mô hình Monto-Carlo để tính lỗi frame (1 từ mã được xem là 1 frame). Trong đó:

* **Khối tạo bản tin ngẫu nhiên**: Sử dụng hàm random để tạo ngẫu nhiên bản tin cần truyền. Vì bộ mã được xây dựng trên trường GF(8), mỗi một symbol ứng với 3 bits, độ dài bản tin là 23 nên 1 bản tin có độ dài 3\*23=69 bits.
* **Khối mã hóa NB-LDPC:** Bản tin được ánh xạ sang thành dạng symbols rồi được mã hóa theo thuật toán được trình bày ở mục 2.2.2.
* **Khối điều chế:** Bản tin được ánh xạ lại dưới dạng bits có độ dài 3\*35=105 bits và được đưa vào điều chế B-PSK, đây là dạng điều chế thường được sử dụng để đánh giá chất lượng của mã kênh.
* **Kênh truyền:** Là kênh AWGN, nhiễu được sinh ngẫu nhiên theo phân bố Gauss và được cộng trực tiếp vào tín hiệu.
* **Khối giải điều chế:** Tín hiệu thu được sau khi truyền đi trên kênh truyền và được giải điều chế, đầu ra của bộ điều chế là xác suất hậu nhiệm của các bit tin. Ở đây, ta giả sử đã biết đặc tính kênh truyền AWGN, tỉ số SNRdB, xác suất tiên nhiệm Px=0 = Px=1 = 0.5. Từ đó ta tính toán được xác suất tiên nhiệm ( của bộ giải điều chế) ứng với mỗi symbol của từ mã nhận được.
* **Khối giải điều chế:** Đầu vào là xác suất tiên nghiệm ứng với mỗi symbol của từ mã, sau khi giải mã lặp theo thuật toán min-max đã trình bày ở trên, ta thu được từ mã.
* **Khối tính lỗi:** Kêt quả của bộ giải mã sẽ được loại bỏ các bit kiểm tra, chuyển về dạng bit và so sánh với chuỗi bit bản tin ban đầu.

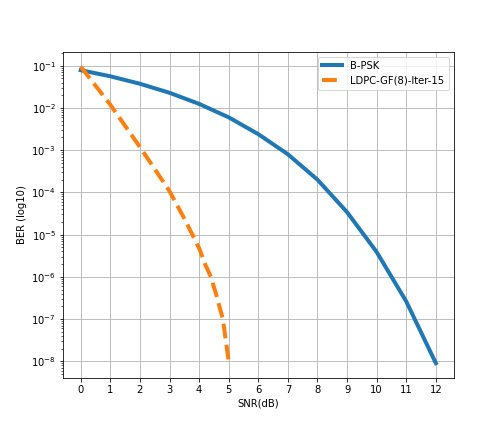
Do toàn bộ quá trình mã hóa, giải mã, điều chế, giải điều chế đều được thực hiện trên phần mềm bị hạn chế về tốc độ so với xử lý trên phần cứng nên chỉ có thể khảo sát tới tỉ lệ lỗi bít ở mức 10-8(yêu cầu tối thiểu 10-10 bit tin truyền đi). Thời gian thực hiện toàn bộ quá trình mô phỏng tốn 20h.

Kết quả:

**

***Hình 2.4*** *Chòm sao tín hiệu BPSK với các mức tỉ*

*lệ công suất tín hiệu trên tạp âm*



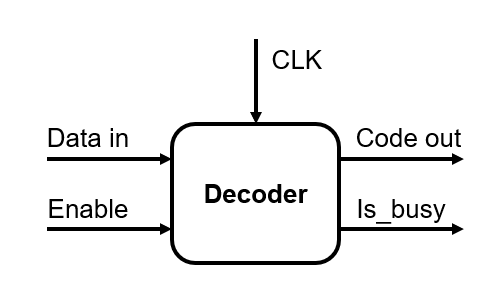
***Hình 2.5*** *So sánh tỉ lệ lỗi bit (BER) khi sử dụng mã kênh*

*và khi không sử dụng*

***Nhận xét:*** Sử dụng mã kênh để bảo vệ có tỉ lệ lỗi bit thấp hơn nhiều so với khi không sử dụng. Cụ thể, ở yêu cầu tỉ lệ lỗi bit 10-3(đối với tín hiệu thoại), Tỉ lệ tín trên tạp (SNRdB) khi không dùng mã kênh cần thiết là 8dB, khi dùng mã kênh là 2dB, vậy độ lợi là 6dB tương đương với 4 lần công suất phát, đây là con số rất có ý nghĩa rất lớn trong thực tiễn. Ở mức yêu cầu lỗi bít 10-6(tín hiệu truyền hình), khi không sử dụng mã kênh SNRdB cần thiết là 10.5 dB, khi dùng mã kênh là 4.3 dB cho độ lợi 6,2 dB. Ở yêu cầu lỗi bit 10-8, độ lợi công suất của bộ mã là 7 dB và độ lợi càng tăng khi yêu cầu về tỉ lệ lỗi bít càng thấp. Độ lợi công suất này có thể giúp chúng ta tiết kiệm công suất phát, tăng khoảng cách liên lạc đối với thông tin vô tuyến hay tăng tốc độ truyền dữ liệu.

### Đánh giá chất lượng bộ giải mã trên chíp FPGA xillinx sparta-6

Sau khi mô phỏng thành công thuật toán trên phần mềm, ta chuyển sang thiết kế bộ giải mã trên phần cứng thực, mục tiêu là triển khai bộ giải mã trên phần cứng sparta 6 hoạt động đúng chức năng giải mã như thuật toán trình bày. Hoạt động của khối giải mã sẽ thiết kế như sau:

**

***Hình 2.6*** *Khối Decoder*

Dữ liệu vào là xác suất hậu nghiệm của từng symbol của từ mã đã được chuẩn hóa và chuyển sang dạng logarith.

Chân Enable được đẩy lên mức *high* trong 1 xung clock, bộ Decoder sẽ bắt đầu nạp dữ liệu vào bộ nhớ (RAM), đồng thời đưa chân is\_busy lên mức *high* để báo bộ decoder đang bận.

Khi bộ Decoder đã nạp xong dữ liệu, nó tiến hành giải mã theo thuật toán đã thiết kế. Khi từ mã thỏa mãn phương trình kiểm tra chẵn lẻ H hoặc hết số vòng lặp tối đa, bộ giải mã sẽ đây kết quả từ mã ra ngoài đồng thời hạ chân is\_busy xuống mức *low.*

Thiết kế được viết bằng ngôn ngữ mô tả phẩn cứng VHDL. Việc thiết kế được chia làm 2 giai đoạn.

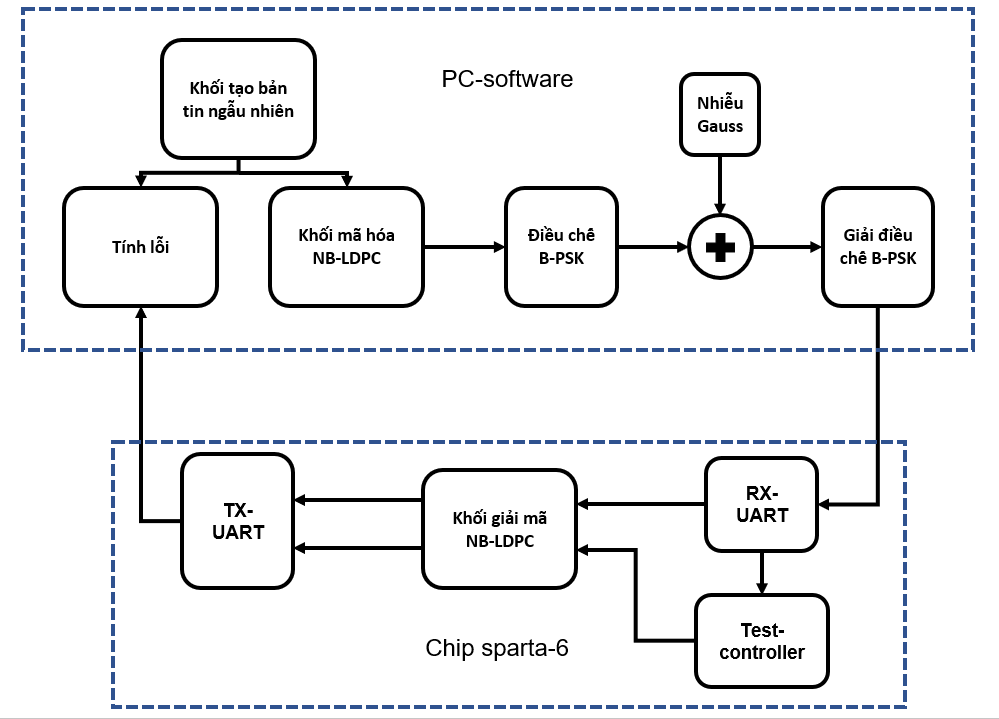
Giai đoạn 1: thiết kế và mô phỏng hoạt động trên phần mềm ModelSim để kiểm tra hoạt động của khối giải mã so với thuật toán trên phần mềm.

Giai đoạn 2; Sử dụng phần công cụ ISE của Xilinx để nạp thiết kế vào chíp sparta-6 và kiểm tra hoạt động của khối giải mã trên phần cứng thực tế.

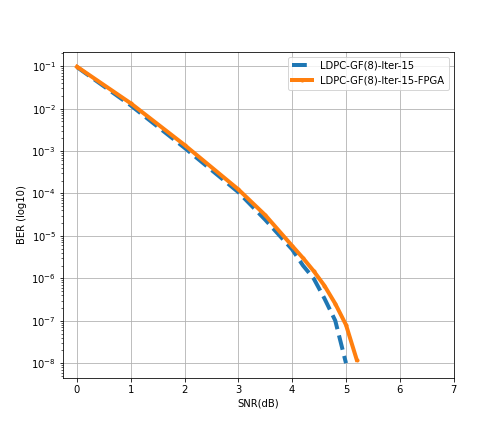
|  |  |  |  |
| --- | --- | --- | --- |
| **Device Ultilization Summary** | | | |
| **Logic Ultilization** | **Used** | **Avaiable** | **Ultilization** |
| Number of Slice Registers | 6753 | 30064 | 22% |
| Number of Slice LUTs | 10113 | 15032 | 67% |
| Number of fully used LUT-FF pairs | 5792 | 11074 | 52% |
| Number of bonded IOBs | 6 | 186 | 3% |
| Number of BUFG/BUFGCTRLs | 8 | 16 | 50% |

*Tài nguyên sử dụng của khối giải mã đã thiết kế:*

Sơ đồ kiểm tra thiết kế như sau:



***Hình 2.7*** *Sơ đồ kiểm tra khả năng sửa lỗi của bộ giải mã trên phần cứng*



***Hình 2.8*** *So sánh chất lượng bộ giải mã trên phần mềm*

*và bộ giải mã được thiết kế trên phần cứng FPGA*

***Nhận xét:*** Bộ giải mã đã thiết kế trên phần cứng có chất lượng hoạt động gần như giống với bộ giải mã đã mô phỏng trên phần mềm. Chất lượng của bộ giải mã trên phần cứng bị suy giảm một lượng không đáng kể do việc sử dụng số thực dấu phẩy tĩnh trên phần cứng (12 bit) so với số thực dấu phẩy động trên phần mềm, sự đánh đổi này là hoàn toàn chấp nhận được.

**KẾT LUẬN**

Qua thời gian tìm hiểu, nghiên cứu các tài liệu, bằng việc vận dụng kiến thức tổng hợp của các môn học cơ sở và chuyên ngành, cùng với sự nỗ lực của bản thân và sự giúp đỡ tận tình của thầy giáo hướng dẫn, các giáo viên trong bộ môn, tôi đã hoàn thành đề tài cảu mình.

*Kết quả đạt được của đề tài như sau:*

***-*** Tìm hiểu về lý thuyết tổng quan của mã kênh, lý thuyết thông tin về kiểm soát lỗi sử dụng mã kênh, cơ sở toán học về trường Galois, cơ sở xây dựng các bộ mã hóa phi nhị phân trên trường Galois bậc cao.

- Đã tìm hiểu được lý thuyết tổng quan cũng như các phương pháp mới nhất trong xây dựng, mã hóa, giải mã cho mã NB-LDPC.

- Thiết kế hệ thống mô phỏng trên phần mềm để đánh giá chất lượng của bộ mã, kết quả giống với báo cáo trong các bài báo đã tham khảo.

- Cụ thể hóa thuật toán lý thuyết trên phần cứng dựa trên cơ sở tổng quan về thiết kế khối giải mã Layered Trellis Min-Max decoder của mã LDPC (23, 35) trên trường GF(8). Đánh giá kết quả, khối giải mã trên phần cứng thực hiện đúng chức năng thuật toán đã đề suất. Quá trình thiết kế đã đề xuất các phương án thiết kế các khối chức năng khả thi để xây dựng trên phần cứng FPGA.

*Hướng phát triển của đề tài:*

- Xây dựng bộ giải mã trên trường GF bậc cao hơn (32, 64...), từ mã dài hơn.

- Đề xuất các cải tiến về thuật toán và các phướng án thiết kế nhằm tăng thông lượng khối giải mã, giảm độ phức tạp giải mã và tăng hiệu quả sử dụng phần cứng của bộ giải mã.

- Nghiên cứu tích hợp bộ mã hóa, giải mã vào các hệ thống thông tin hiện có cũng như trong các hệ thống thông tin mới sắp được triển khai như hệ thống thông tin di động 6G...

Do thời gian nghiên cứu cũng như kiến thức và kinh nghiệm của bản thân còn hạn chế nên đề tài không tránh khỏi những thiếu sót trong quá trình hoàn thiện, tôi rất mong nhận được các ý kiến đóng góp của các thầy cô và bạn đọc để có thể hoàn thiện đề tài hơn nữa.

Một lần nữa, tôi xin chân thành cảm ơn sự giúp đỡ nhiệt tình của các thầy cô trong bộ môn Thông tin- Khoa Vô tuyến điện tử - HVKTQS, đặc biệt thầy giáo TS. Phạm Xuân Nghĩa và thầy Đàm Đức Thuận đã giúp tôi hoàn thành đồ án này.

Tôi xin chân thành cảm ơn!

***Tài liệu tham khảo:***

1. Bo Zhou, Shu Lin, Jingyu Kang, Shumei Song...(2009) Construction of Non-Binary Quasi-Cyclic LDPC Codes by Array and Array Dispersion.
2. Shu Lin, S. Song, Bo Zhou (2007) Algebraic constructions of non-binary quasi-cyclic LDPC codes: array masking and dispersion.
3. Lacruz, Hererom David Declercq (2014), Simplified Trellis Min-Max Decoder Architecture for NB-LDPC codes.
4. Carassco, Martin Johnston (2008) Non-Binary Error control coding for wireless communication and data storage.
5. Nguyễn Văn Duẫn (2015), *Luận án tiến sĩ* Nghiên cứu cải tiến thuật toán giải mã lặp LDPC và ứng dụng trong hệ thống FH/NC-BPSK
6. Erbao Li, David Declercq (2013) , Trellis-based extended Min-Sum Algorithm for Non-Binary LDPC Codes and its hardware structure.
7. Valentin Savin, Grenoble (2008) Min-Max decoding for non binary LDPC codes.