BỘ THÔNG TIN VÀ TRUYỀN THÔNG

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÁO CÁO NGHIÊN CỨU KHOA HỌC

Đề tài: "*Nghiên cứu, tìm hiểu một số phương pháp giải mã không gian, thời gian và thực thi trên phần cứng FPGA*"

Mã số: 26 – SV – 2024 - ĐT.

Giảng viên hướng dẫn : TS. Trần Thị Thúy Hà

Sinh viên thực hiện : Đoàn Long Vũ – B21DCDT247

: Nguyễn Sỹ Toản – B21DCDT223

HÀ NỘI – 2024

LỜI CẢM ƠN

Trong quá trình nghiên cứu em đã nhận được sự giúp đỡ, đóng góp ý kiến và chỉ bảo nhiệt tình của thầy cô, gia đình và bạn bè.

Đầu tiên, em xin được gửi lời cảm ơn đến Ban Giám hiệu Học viện Công nghệ Bưu chính Viễn thông đã tạo cho em môi trường rèn luyện tốt để em có thể học tập và tiếp thu được những kiến thức quý báu trong những năm qua.

Em xin cảm ơn tất cả các thầy cô giáo, đặc biệt là các thầy cô trong khoa Kỹ thuật điện tử 1 đã tận tình chỉ dạy những kiến thức quý báu để em có thể hoàn thành được đề tài cũng như những hành trang cần thiết để em có thể bước trên con đường sự nghiệp sau này.

Em xin được gửi lời cảm ơn sâu sắc nhất đến cô Trần Thị Thúy Hà, thầy Lê Minh Tuấn và thầy Phạm Văn Sự , những người đã trực tiếp hướng dẫn em thực hiện đề tài này. Mặc dù công việc rất bận rộn nhưng các thầy vẫn luôn dành thời gian hướng dẫn chỉ bảo tận tình để em có thể hoàn thành tốt đề tài.

Mặc dù đã cố gắng hết mình, đề tài không tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm và chỉ bảo tận tình của quý thầy cô và các bạn để em có thể hoàn thành tốt hơn báo cáo này.

Cuối cùng em xin kính chúc các thầy, các cô, gia đình và bạn bè dồi dào sức khỏe và thành công trong sự nghiệp.

Hà Nội, ngày 20 tháng 11 năm 2024

Sinh viên thực hiện 1 Sinh viên thực hiện 2

Đoàn Long Vũ Nguyễn Sỹ Toản

MỤC LỤC

MỤC LỤC HÌNH ẢNH

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

Từ viết tắt	Viết đầy đủ	Giải nghĩa
MIMO	Multiple Input- Multiple Output	Nhiều đầu vào – Nhiều đầu ra
STBC	Space Time Block Coding	Mã hóa Khối Không gian-Thời gian
O-STBC	Orthoronal-Space Time Block Coding	Bộ mã hóa Trực giao Khối Không gian-Thời gian
SM	Spatial Modulation	Điều chế không gian
SO-ML	Single-Stream Optimal Maximum Likelihood	Bộ Giải mã tối ưu Khả năng Tối đa với một Luồng
SO-SD	Single-Stream Optimal Sphere Decoder	Bộ Giải mã Hình cầu Tối ưu với một Luồng.
SC	Spatial Constellation	Chòm Sao Không Gian
FPGA	Field-Programmable Gate Array	Field-Programmable Gate Array
G–STSK	Generalized Space-Time Shift Keying	Điều chế dịch chuyển không gian-thời gian tổng quát
QAM	Quadrature Amplitude Modulation	Điều chế biên độ vuông góc

LỜI MỞ ĐẦU

Hệ thống truyền thông đã trở thành một trong những công nghệ hiện đại của chúng ta, đòi hỏi sự phát triển của các liên kết truyền thông tốc độ cao và đáng tin cậy với các hệ thống Nhiều Đầu Vào Nhiều Đầu Ra (Multiple Input Multiple Output - MIMO). MIMO là một trong những công nghệ chủ chốt có thể đạt được mục tiêu của mạng 4G và 5G. Vì các hệ thống MIMO sử dụng nhiều ăng-ten ở cả bộ phát và bộ thu, nên yêu cầu cầnthiết để giảm thiểu fading do sự dao động của tín hiệu trong kênh truyền là Tính Đa Dạng (Diversity). Do đó, các hệ thống MIMO tận dụng tính đa dạng không gian vốn có. Tuy nhiên, các hệ thống MIMO hoạt động trong môi trường nhiễu và fading phức tạp, dẫn đến sự suy giảm chất lượng tín hiệu đáng kể. Để giải quyết vấn đề này, kỹ thuật mã hóa không gian-thời gian (STBC - Space-Time Block Coding) được áp dụng để tối ưu hóa khả năng kháng fading và cải thiện độ tin cậy. STBC là phương pháp mã hóa tín hiệu qua nhiều ăng-ten phát theo cả hai chiều không gian và thời gian, tạo ra tính đa dạng không gian-thời gian. Nhờ vào tính đa dạng này, các bản sao của tín hiệu được truyền đi từ các ăng-ten khác nhau qua các kênh độc lập, cho phép bộ thu có thể khôi phục lại thông tin chính xác ngay cả khi một số kênh bị nhiễu. Trong số các phương pháp STBC, mã khối không gian-thời gian trực giao (OSTBC - Orthogonal Space-Time Block Coding) được đánh giá cao nhờ tính trực giao giữa các mã tín hiệu, giúp đơn giản hóa đáng kể quá trình giải mã tại bộ thu mà không làm giảm hiệu suất truyền dẫn.

Spatial Modulation (SM) là một kỹ thuật điều chế trong truyền thông không dây, đặc biệt dành cho các hệ thống MIMO (Multiple Input Multiple Output). SM sử dụng không gian phát sóng như một phương tiện để truyền tải thông tin, tận dụng sự sắp xếp và vị trí của các anten trong hệ thống phát.

Qua đó, mục tiêu của bài nghiên cứu này kết hợp kỹ thuật điều chế không gian(SM) và phương pháp mã khối trực giao không gian và thời gian (O-STBC), để thực hiện giải mã tín hiệu được phát tại vị nào trong không gian là gần nhất.

Sau thời gian tìm hiểu nghiên cứu, tác giả xin trình bày những nội dung đã nghiên cứu được trong luận văn gồm 3 chương:

- **Chương 1: Cơ sở lý thuyết.**
- **Chương 2: Thiết kế hệ thống.**
- **Chương 3: Kết quả mô phỏng mà hướng nghiên cứu tiếp theo.**

Dưới đây em xin trình bày chi tiết các phần trong nội dung của đề tài.

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

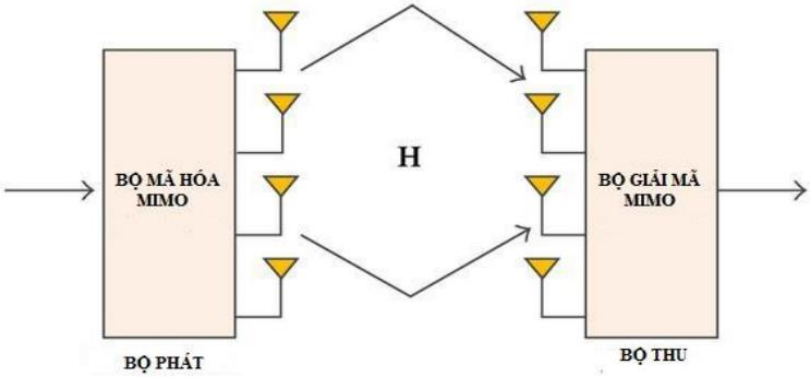
Giới thiệu chương : Nội dung chương 1 sẽ tập trung nghiên cứu tổng quan lý thuyết về hệ thống MIMO, mã hóa không gian, thời gian STBC, mã hóa không gian thời gian trực giao OSTBC, điều chế không gian SM và sự kết hợp SM-OSTBC và các thuật toán, phương pháp giải mã của chúng.

1.1 TỔNG QUAN VỀ HỆ THỐNG MIMO

1.1.1 Tổng quan về hệ thống MIMO

Do sự phát triển của các dịch vụ đa phương tiện, các thiết bị kết nối không dây trên toàn thế giới nên nhu cầu về dung lượng trong các hệ thống thông tin không dây đang tăng lên rất nhanh. Tuy nhiên phổ tần vô tuyến lại hạn chế, do vậy muốn tăng dung lượng ta bắt buộc phải tăng hiệu quả sử dụng tần số. Những tiến bộ trong mã hóa như mã Turbo, mã kiểm tra chẵn lẽ mật độ thấp (LDPC) đã có thể tiệm cận tới giới hạng dung lượng Shannon của hệ thống với một ăng-ten. Tuy nhiên có thể đạt được hiệu quả phổ tần cao hơn nữa bằng việc sử dụng hệ thống nhiều ăng-ten ở cả máy phát và máy thu.

MIMO là các hệ thống truyền dẫn vô tuyến sử dụng đồng thời nhiều ăngten ở máy phát và máy thu, nhằm tăng tốc độ truyền. Chuỗi tín hiệu phát được mã hóa theo cả hai miền không gian và thời gian nhờ bộ mã hóa không gian thời gian (STE: Space-Time Encoder). Tín hiệu sau khi được mã hóa không gian - thời gian được phát đi nhờ N ăng-ten phát. Máy thu sử dụng phân tập thu với M ăng-ten thu. Kênh tổng hợp giữa máy phát (Tx) và máy thu (Rx) có N đầu vào và M đầu ra được gọi là kênh MIMO M ×N . Trong các trường hợp đặc biệt khi N = 1 và M = 1, tương ứng chúng ta có các hệ thống phân tập thu SIMO và phân tập phát MISO. Để tránh ảnh hưởng giữa các ăng-ten phát và các ăng-ten thu với nhau, khoảng cách yêu cầu tối thiểu giữa các phần tử ăng-ten ở các mảng ăng-ten phát hoặc thu là lamda/2



Hình 1.1 Hệ thống truyền thông MIMO

1.1.2 Mô hình toán học của hệ thống MIMO

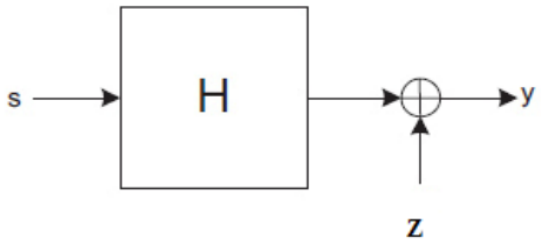
Bắt đầu từ trường hợp đơn giản là kênh truyền có hệ số truyền xác định (không có fading mà chỉ có hệ số suy giảm và ồn) và được biết trước qua phép ước lượng kênh, băng tần hẹp bất biến với thời gian. Một hệ thống gồm N ăng-ten phát và M ăng-ten thu thường được gọi là hệ thống MIMO MxN. Một hệ thống như vậy thường được mô tả mối quan hệ đầu vào – đầu ra như sau [1] :

$$\mathbf{Y = Hs + z}$$

z là vectơ tạp âm;

H là ma trận các đặc tính kênh truyền như thông tin về độ lớn, về pha của đường truyền giữa N ăng-ten phát và M ăng-ten thu.

Mô hình toán học được diễn tả như sau:



Hình 1.2 Mô hình toán học hệ thống MIMO

Ma trận H có dạng :

$$\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & \dots & h_{1N} \\ h_{21} & h_{21} & \dots & h_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ h_{M1} & h_{M1} & \dots & h_{MN} \end{bmatrix}$$

Định nghĩa các véc-tơ phát, thu và tạp âm tương ứng là:

$$s = [s_1, s_2, \dots, s_N]^T$$

$$y = [y_1, y_2, \dots, y_N]^T$$

$$z = [z_1, z_2, \dots, z_N]^T$$

Chúng ta có mối quan hệ giữa tín hiệu thu và phát biểu diễn qua phương trình hệ thống sau :

$$y = \sqrt{\left(\frac{P_t}{N}\right) H_s} + z$$

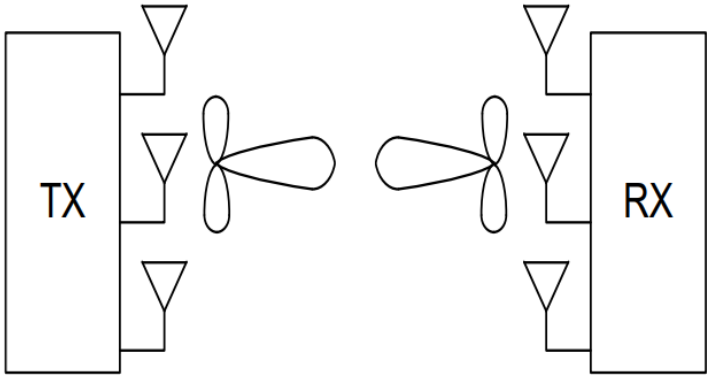
Với P_t là tổng công suất phát từ N ăng-ten phát.

1.1.3 Độ lợi trong hệ thống MIMO

1.1.3.1 Độ lợi tạo búp (Beamforming)

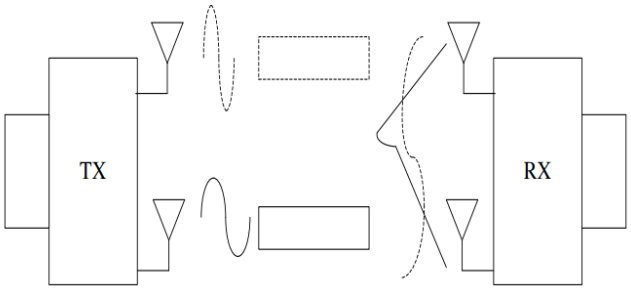
Tạo búp giúp hệ thống tập trung năng lượng bức xạ theo hướng mong muốn giúp tăng hiệu quả công suất, giảm can nhiễu và tránh được can nhiễu tới từ các hướng không mong muốn, từ đó giúp cải thiện chất lượng kênh truyền và tăng độ bao phủ của hệ thống. Để có thể thực hiện tạo búp, khoảng cách giữa các ăng-ten trong hệ thống MIMO

thường nhỏ hơn bước sóng (thông thường là), tạo búp thường được thực hiện trong môi trường ít tán xạ .Khi môi trường tán xạ mạnh hệ thống MIMO có thể cung cấp độ lợi ghép kênh không gian và độ lợi phân tập.



Hình 1.3 Kỹ thuật tạo búp

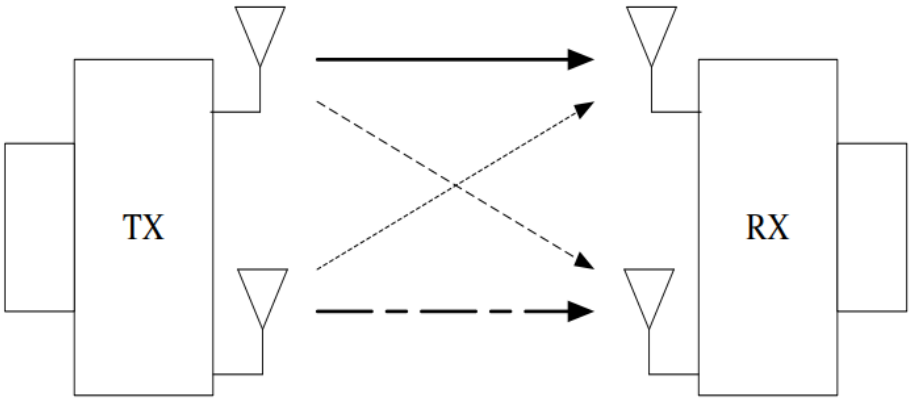
1.1.3.2 Độ lợi ghép kênh không gian



Hình 1.4 Độ lợi ghép kênh không gian

Tận dụng các kênh truyền song song có được từ đa ăng-ten tại phía phát và phía thu trong hệ thống MIMO, các tín hiệu sẽ được phát độc lập và đồng thời ra các ăng-ten (hình 1.5), nhằm tăng dung lượng kênh truyền mà không cần tăng công suất phát hay tăng băng thông hệ thống. Dung lượng hệ thống sẽ tăng tuyến tính theo số các kênh truyền song song trong hệ thống. Để cực đại độ lợi ghép kênh qua đó cực đại dung lượng kênh truyền thuật toán V-BLAST (Vertical- Bell Laboratories Layered Space-Time) được áp dụng.

1.1.3.3 Độ lợi phân tập



Hình 1.5 Sơ đồ khối hệ thống MIMO 2x2

Trong truyền dẫn vô tuyến, mức tín hiệu luôn thay đổi, bị fading liên tục theo không gian, thời gian và tần số, khiến cho tín hiệu tại nơi thu không ổn định, việc phân tập cung cấp cho các bộ thu các bản sao tín hiệu giống nhau qua các kênh truyền fading khác nhau (hinh 1.6), bộ thu có thể lựa chọn hay tổ hợp TX RX TX RX hay tổ hợp các bản sao tín hiệu này để giảm thiểu tốc độ lỗi biet BER, chống fading qua đó tăng độ tin cậy của hệ thống. Để cực đại độ lợi phân tập, giảm BER và chống lại fading, thuật toán STBC (Space-Time Block Code) và STTC (Space-Time Trellis Code) được áp dụng. Thực tế, để hệ thống có dung lượng cao, BER thấp, chống được fading, ta phải có sự tương quan giữa độ lợi phân tập và độ lợi ghép kênh trong việc thiết kế hệ thống

1.2 MÃ KHỐI KHÔNG GIAN VÀ THỜI GIAN - STBC

1.2.1 Giới thiệu chung về mã hõa không gian và thời gian

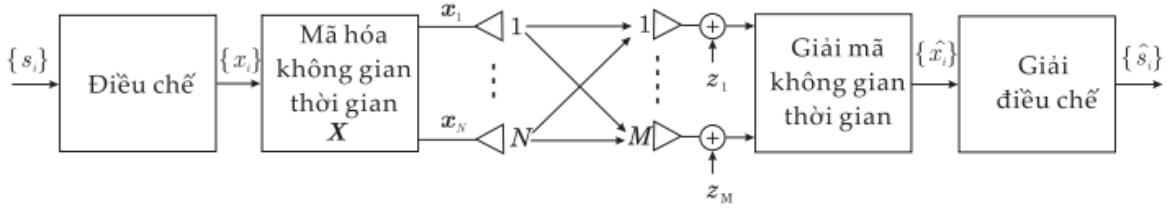
Mã hóa không gian – thời gian (STC: Space-Time Coding) [1],[2],[3],[4] là kỹ thuật mã hóa được sử dụng cho trường hợp máy phát được trang bị nhiều ăng-ten phát. Sự mã hóa được thực hiện trong cả miền không gian và miền thời gian nhằm tạo ra sự tương quan giữa các tín hiệu được phát đi từ các ăng-ten khác nhau và tại những thời điểm khác nhau. Nhờ đó ta có thể cực đại hóa hệ số tăng ích phân tập (diversity gain) và hệ số tăng ích mã hóa (coding gain). Nói cách khác, kỹ thuật mã hóa không gian thời gian là kỹ thuật cho phép chúng ta thực hiện phân tập không gian phát. Kỹ thuật phân tập

không gian - thời gian của Alamouti hay kỹ thuật phân tập phát giữ chậm có thể coi là các bộ mã không gian-thời gian đơn giản.

Nền móng của kỹ thuật mã hóa không gian-thời gian (gọi tắt là kỹ thuật STC) bắt nguồn từ sự ra đời của *mã lưới không gian–thời gian* (hay STTC) [2]. Một ưu điểm của mã STTC là chúng có thể cung cấp cả hệ số tăng ích phân tập và hệ số tăng ích mã hóa. Tuy nhiên, để giải mã ta phải sử dụng thuật toán giải mã Viterbi, dẫn tới mức độ phức tạp quá trình khôi phục tín hiệu lớn (biến thiên theo hàm mũ). Kỹ thuật STC trở nên ngày càng thông dụng hơn với sự ra đời của kỹ thuật *mã hóa khối không gian–thời gian* (STBC: Space–Time Block Coding).

Trong các phần sau đây, chúng ta sẽ nghiên cứu chi tiết thêm về các hai loại mã này. Đối với loại mã STBC phần trình bày sẽ chỉ giới hạn vào loại mã STBC tuyến tính trực giao do loại mã này cho phép đơn giản hoá việc giải mã ở máy thu thông qua các phép xử lý tín hiệu tuyến tính đơn giản

1.2.2 Mã khối không gian và thời gian STBC



Hình 1.6 Sơ đồ khối của một hệ thống mã hóa STBC.

Sơ đồ một hệ thống MIMO sử dụng STBC được mô tả Hình vẽ 9.1. Một cách tổng quát, bộ mã hoá (STE: Space-Time Encoder) của STBC có thể có như một phép ánh xạ của một tập Ns ký hiệu phát phức sang một ma trận phát **X** với N hàng và Nt cột. Nhìn chung, phép ánh xạ

$$\{\boldsymbol{s_1}; \boldsymbol{s_2}; \dots; \boldsymbol{sN_s} \rightarrow \boldsymbol{X}\}$$

có thể ở bất kỳ dạng nào như tuyến tính hoặc phi tuyến. Tuy nhiên, do STBC phi tuyến ít được quan tâm, trong phần sau đây chúng ta sẽ giới hạn trình bày vào loại mã STBC tuyến tính. Với mã STBC tuyến tính thì phép ánh xạ (9.1) có thể được biểu diễn ngắn gọn như sau [5]:

$$X = \sum_{k=1}^{N_s} \left(\tilde{S}_k A_k + j \tilde{S}_k B_k \right)$$

trong đó *s~k* = R{*sk*} và *s˘k* = I{*sk*} tương ứng biểu diễn phần thực và ảo của *sk*, còn ***A****k* và ***B****k* là các ma trận cố định cho trước được gọi là các *ma trận phân tán* (dispersion matrix). Ma trận ***X*** này được truyền tới máy thu thông qua *N* ăng-ten và *Nt* khe thời gian. véc-tơ tín hiệu phát từ ăng-ten *n* (thông qua *Nt* khe thời gian) được biểu diễn bởi hàng *n* của ***X*** và véc-tơ tín hiệu phát đồng thời qua *N* ăng-ten tại một khe thời gian *t* được biểu diễn bằng cột thứ *t* của ***X***. Như vậy, một phần tử *xn;t* biểu diễn một ký hiệu phát mã hoá được phát đi từ ăng-ten *n* tại khe thời gian *t*.

Do *Ns* ký hiệu được truyền đi trong khoảng thời gian *Nt* khe thời gian nên tốc độ truyền của mã STBC là :

$$R = \frac{N_s}{N_t}$$

Giả sử bộ điều chế được sử dụng có bậc M thì số bít được truyền đi trong mỗi ký hiệu phát là *mb* = log2 M. Như vậy, hiệu suất phổ tần của mã STBC có thể tính được như sau [6, Ch.3]

$$\eta = \frac{r_b}{B} = \frac{r_s m_b R}{r_s} = \frac{m_b N_s}{N_t} \text{ [bits/s/Hz]}$$

trong đó *rb* và *rs* tương ứng là các tốc độ truyền bit và truyền ký hiệu, còn *B* là băng tần tín hiệu.

Một ví dụ điển hình của mã STBC là bộ mã do Alamouti [1] đề xuất với ma trận phát

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \end{bmatrix} = \begin{vmatrix} s_1 & s_2^* \\ s_2 & -s_1^* \end{vmatrix}$$

Các ma trận ánh xạ ***A****k* và ***B****k* của mã Alamouti STBC có thể biểu diễn như sau [5]

$$A_1 = \begin{bmatrix} 1 & -0 \\ 0 & -1 \end{bmatrix} \quad A_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

Do *Ns* = *Nt* = 2 nên tốc độ truyền dẫn của mã Alamouti STBC là *R* = 1.

Để thu được bậc phân tập toàn phần (full diversity) *N* do *N* ăng-ten phát cung cấp, ma trận mã truyền ***X*** được thiết kế trên cơ sở trực giao sao cho [4]

$$XX^H = C \sum_{k=1}^{N_S} |s_k|^2 I_N$$

trong đó *C* là một hằng số nào đó. Loại mã STBC được xây dựng trên cơ sở trực giao này được gọi là mã STBC trực giao (O-STBC: orthogonal STBC). Với mã STBC trực giao, các véc-tơ phát từ các ăng-ten trực giao với nhau. Tức là nếu chúng ta định nghĩa ***x****n* là các véc-tơ (hàng) phát từ ăng-ten *n*, tương ứng với hàng thứ *n* của ***X***,

thì

$$x_i x_j{}^H = \sum_{t=1}^{N_t} x_i x_{j,t}^* = 0, \quad , i, j \in \{1,2, \dots, N\}.$$

Tính chất trực giao của mã STBC này cho phép đạt được bậc phân tập toàn phần, đồng thời cho phép máy thu tách riêng các thành phần tín hiệu phát từ các ăng-ten phát, và vì vậy đơn giản hoá bộ tách tín hiệu thông qua tách sóng hợp lẽ tối ưu dựa trên các phép xử lý tuyến tính trên tín hiệu thu.

Tốc độ truyền dẫn tối đa của các mã STBC trực giao có bậc phân tập toàn phần là 1, tức là, $R \leq 1$ [4], trong đó mã Alamouti là bộ mã cho tín hiệu phức duy nhất cho phép đạt được đồng thời cả bậc phân tập toàn phần và tốc độ toàn phần (full rate).

Tuỳ theo tập tín hiệu (signal constellation), chúng ta có thể thiết kế được các bộ mã STBC trực giao khác nhau. Trong phần sau đây chúng ta sẽ tìm hiểu về các bộ mã dành cho tập tín hiệu thực và tập tín hiệu phức.

1.2.3. Mã khối không gian và thời gian trực giao O-STBC

Việc xây dựng mã STBC phải dựa vào mã trực giao. Ma trận truyền dẫn XnT có hàng là trực giao với nhau. Tức là trong mỗi khối, tính hiệu phát từ 2 anten phải là trực giao với nhau. Có nghĩa là trong mỗi khối các chuỗi tín hiệu từ 2 anten phát bất kỳ là trực giao nhau. Ví dụ chúng ta giả sử tín hiệu phát từ anten thứ i là

$$x_i = \left(x_{i,0}, x_{i,1} \dots \dots, x_{i,p}\right) \quad Với\ i = 1,2, \dots \dots, n_T\ .Khi\ \text{đó}$$

$$x_i . X_j = \sum_{t=1}^{p} x_{i,t} . x_{j,t} = 0 \quad i \neq j, i, j \in \{1,2, \dots, n_T\}$$

Trong đó xi , xj ký hiệu tích vô hướng của 2 chuỗi xi , xj . Tính trực giao có thể đạt được phân tập phát đầy đủ với một số anten phát cho trước. Nó giúp cho việc thu tách các tín hiệu được đơn giản hơn và do vậy giải mã ML đơn giản chỉ dựa trên xử lý tuyến tính tính các tín hiệu thu

1.2.3.1 Mã O–STBC cho tập tín hiệu thực

Một ví dụ điển hình của tín hiệu thực là tập tín hiệu sử dụng phương pháp điều chế biên độ xung (PAM: Pulse Amplitude Modulation), trong đó biên độ của tín hiệu sau điều chế thay đổi theo tín hiệu điều chế.

Để đạt được độ phân tập toàn phần, các bộ mã STBC cho các hệ thống *N* ăng-ten phát sử dụng tập tín hiệu thực cũng cần phải thoả mãn điều kiện trực giao (9.8)

$$X_N X^T{}_N = C \sum_{k=1}^{N_S} |s_k|^2 I_N.$$

Dựa định lý Hurtwitz-Radon về thiết kế trực giao, Tarokh, Jafarkhani và Calderbank [4] đã chứng minh được rằng các bộ mã vuông (*N* = *Nt*) có thể đạt được đồng thời cả bậc phân tập toàn phần và tốc độ toàn phần cho tập tín hiệu thực chỉ tồn tại với số ăng-ten phát *N* = 2;4;8. Ma trận truyền dẫn của các bộ mã này là [4]

$$X_2 = \begin{bmatrix} s_1 & -s_2 \\ s_2 & s_1 \end{bmatrix}$$

$$\boldsymbol{X_4} = \begin{bmatrix} s_1 & -s_2 & -s_3 & -s_4 \\ s_2 & s_1 & s_4 & -s_3 \\ s_3 & -s_4 & s_1 & s_2 \\ s_4 & s_3 & -s_2 & s_1 \end{bmatrix}$$

$$X_8 = \begin{bmatrix} s_1 & -s_2 & -s_3 & -s_4 & -s_5 & -s_6 & -s_7 & -s_8 \\ s_2 & s_1 & -s_4 & s_3 & -s_6 & s_5 & s_8 & -s_7 \\ s_3 & s_4 & s_1 & -s_2 & -s_7 & -s_8 & s_5 & s_6 \\ s_4 & -s_3 & s_2 & s_1 & -s_8 & s_7 & -s_6 & s_5 \\ s_5 & s_6 & s_7 & s_8 & s_1 & -s_2 & -s_3 & -s_4 \\ s_6 & -s_5 & s_8 & -s_7 & s_2 & s_1 & s_4 & -s_3 \\ s_7 & -s_8 & -s_5 & s_6 & s_3 & -s_4 & s_1 & s_2 \\ s_8 & s_7 & -s_6 & -s_5 & s_4 & s_3 & -s_2 & s_1 \end{bmatrix}$$

Để ý rằng các bộ mã $\boldsymbol{X}2$, $\boldsymbol{X}4$, $\boldsymbol{X}8$ sử dụng *Nt* khe thời gian để truyền đi *Ns* = *Nt* ký hiệu phát nên có tốc độ truyền *R* = 1. Các bộ mã này cũng thoả mãn tính chất trực giao do

$$x_i x^H{}_j = \sum_{k=1}^{N_s} |s_k|^2$$

$$x_i x^H{}_j = 0, i \neq j$$

Tổng quát hoá định lý Hurtwitz-Radon cho trường hợp ma trận không vuông, Tarokh, Jafarkhani và Calderbank [4] đã tìm được các ma trận phát $\boldsymbol{X}N$ cho bất kỳ số ăng-ten phát N nào. Các ví dụ ma trận phát cho $N \leq 8$ được trình bày dưới đây [4]

$$X_3 = \begin{bmatrix} s_1 & -s_2 & -s_3 & -s_4 \\ s_2 & s_1 & s_4 & -s_3 \\ s_3 & -s_4 & s_1 & s_2 \end{bmatrix}$$

$$X_5 = \begin{bmatrix} s_1 & -s_2 & -s_3 & -s_4 & -s_5 & -s_6 & -s_7 & -s_8 \\ s_2 & s_1 & -s_4 & s_3 & -s_6 & s_5 & s_8 & -s_7 \\ s_3 & s_4 & s_1 & -s_2 & -s_7 & -s_8 & s_5 & s_6 \\ s_4 & -s_3 & s_2 & s_1 & -s_8 & s_7 & -s_6 & s_5 \\ s_5 & s_6 & s_7 & s_8 & s_1 & -s_2 & -s_3 & -s_4 \end{bmatrix}$$

$$X_6 = \begin{bmatrix} s_1 & -s_2 & -s_3 & -s_4 & -s_5 & -s_6 & -s_7 & -s_8 \\ s_2 & s_1 & -s_4 & s_3 & -s_6 & s_5 & s_8 & -s_7 \\ s_3 & s_4 & s_1 & -s_2 & -s_7 & -s_8 & s_5 & s_6 \\ s_4 & -s_3 & s_2 & s_1 & -s_8 & s_7 & -s_6 & s_5 \\ s_5 & s_6 & s_7 & s_8 & s_1 & -s_2 & -s_3 & -s_4 \\ s_6 & -s_5 & s_8 & -s_7 & s_2 & s_1 & s_4 & -s_3 \end{bmatrix}$$

$$X_7 = \begin{bmatrix} s_1 & -s_2 & -s_3 & -s_4 & -s_5 & -s_6 & -s_7 & -s_8 \\ s_2 & s_1 & -s_4 & s_3 & -s_6 & s_5 & s_8 & -s_7 \\ s_3 & s_4 & s_1 & -s_2 & -s_7 & -s_8 & s_5 & s_6 \\ s_4 & -s_3 & s_2 & s_1 & -s_8 & s_7 & -s_6 & s_5 \\ s_5 & s_6 & s_7 & s_8 & s_1 & -s_2 & -s_3 & -s_4 \\ s_6 & -s_5 & s_8 & -s_7 & s_2 & s_1 & s_4 & -s_3 \\ s_7 & -s_8 & -s_5 & s_6 & s_3 & -s_4 & s_1 & s_2 \end{bmatrix}$$

So sánh (9.16) với (9.12), (9.17)–(9.19) với (9.13) chúng ta có thể nhận thấy rằng bộ mã $\boldsymbol{X}3$ được xây dựng trên cơ sở bộ mã vuông $\boldsymbol{X}4$ bằng cách lấy 3 hàng đầu tiên. Tương tự, các bộ mã $\boldsymbol{X}5$, $\boldsymbol{X}6$, $\boldsymbol{X}7$ tương ứng được xây dựng từ 5, 6, và 7 hàng đầu tiên của bộ mã vuông $\boldsymbol{X}8$. Thực tế thì chúng ta có thể xây dựng được các bộ mã cho các hệ thống với

N′ ăng-ten phát từ bộ mã vuông ***XN*** bằng cách lấy *N′* hàng bất kỳ từ ***XN*** [5]. Cũng giống như các bộ mã STBC trực giao vuông các bộ mã không vuông này sử dụng *Nt* khe thời gian để truyền đi *Ns* ký hiệu phát. Vì vậy, các bộ mã này cũng có tốc độ toàn phần, tức là *R* = 1, đồng thời cũng cho phép thu được độ phân tập toàn phần *N*

1.2.3.2 Mã O–STBC cho tập tín hiệu phức

Các tập tín hiệu phức được sử dụng rất phổ biến trong thông tin vô tuyến số như M-QAM hay M-PSK. Để đạt được độ phân tập toàn phần, các bộ mã O-STBC cho các hệ thống *N* ăng-ten phát sử dụng tập tín hiệu phức cũng cần phải thoả mãn điều kiện trực giao (9.8)

$$X_N X^H{}_N = C \sum_{k=1}^{N_S} |s_k|^2 I_N.$$

Bộ mã Alamouti STBC với ma trận phát (9.5)

$$X = \begin{bmatrix} s_1 & s^*{}_2 \\ s_2 & -s^*{}_1 \end{bmatrix}$$

là một ví dụ điển hình về bộ mã phức cho 2 ăng-ten phát. Đây cũng là bộ mã O-STBC phức duy nhất có thể đạt được cả tốc độ và độ phân tập toàn phần [4]. Với các hệ thống có nhiều hơn 2 ăng-ten phát thì một cách tổng quát chúng ta có thể xây dựng được các bộ mã phức cho bất kỳ số ăng-ten *N* nào với tốc độ *R* = 1/2 [4]. Một số ví dụ về các bộ mã phức tốc độ 1/2 cho *N* = 3;4 là

$$X_3 = \begin{bmatrix} s_1 & -s_2 & -s_3 & -s_4 & s_1^* & -s_2^* & -s_3^* & -s_4^* \\ s_2 & s_1 & s_4 & -s_3 & s_2^* & s_1^* & s_4^* & -s_3^* \\ s_3 & -s_4 & s_1 & s_2 & s_3^* & -s_4^* & s_1^* & s_2^* \end{bmatrix}$$

$$X_4 = \begin{bmatrix} s_1 & -s_2 & -s_3 & -s_4 & s_1^* & -s_2^* & -s_3^* & -s_4^* \\ s_2 & s_1 & s_4 & -s_3 & s_2^* & s_1^* & s_4^* & -s_3^* \\ s_3 & -s_4 & s_1 & s_2 & s_3^* & -s_4^* & s_1^* & s_2^* \\ s_4 & s_3 & -s_2 & s_1 & s_8^* & s_3^* & -s_2^* & s_1^* \end{bmatrix}$$

Một cách tổng quát, chúng ta có thể xây dựng các bộ mã O-STBC phức tốc độ *R* = 1/2 bằng cách xây dựng một bộ mã O-STBC phức có cấu trúc tương tự như một bộ mã O-STBC thực có tốc độ toàn phần rồi nối với ma trận liên hợp phức của nó.

Với tốc độ lớn hơn 1/2 có một số mã STBC trực giao phức với tốc độ *R* = 3/4 cho các hệ thống 3 và 4 ăng-ten phát được xây dựng trên cơ sở lý thuyết thiết kế "thoả hiệp" (admicable design) như sau [4]:

$$X_3^{\frac{3}{4}} = \begin{bmatrix} s_1 & -s_2^* & \frac{s_3^*}{\sqrt{2}} & \frac{s_3}{\sqrt{2}} \\ s_2 & s_1^* & \frac{s_3^*}{\sqrt{2}} & \frac{-s_3}{\sqrt{2}} \\ \frac{s_3}{\sqrt{2}} & \frac{s_3}{\sqrt{2}} & \frac{-s_1 - s_1^* + s_2 - s_2^*}{2} & \frac{-s_1 - s_1^* + s_2 + s_2^*}{2} \end{bmatrix}$$

$$X_4^{\frac{3}{4}} = \begin{bmatrix} s_1 & -s_2^* & \frac{s_3^*}{\sqrt{2}} & \frac{s_3}{\sqrt{2}} \\ s_2 & s_1^* & \frac{s_3^*}{\sqrt{2}} & \frac{-s_3}{\sqrt{2}} \\ \frac{s_3}{\sqrt{2}} & \frac{s_3}{\sqrt{2}} & \frac{-s_1 - s_1^* + s_2 - s_2^*}{2} & \frac{-s_{1-} s_1^* + s_2 + s_2^*}{2} \\ \frac{s_3}{\sqrt{2}} & \frac{-s_3}{\sqrt{2}} & \frac{-s_1 - s_1^* - s_2 - s_2^*}{2} & \frac{-s_{1-} s_1^* - s_2 + s_2^*}{2} \end{bmatrix}$$

Một số mã có tốc độ 3/4 khác dùng cho các hệ thống 3 và 4 ăng-ten phát được cho bởi các biểu thức sau [7]-[5]:

$$X_3 = \begin{bmatrix} s_1 & 0 & -s_2^* & s_3^* \\ 0 & s_1 & -s_3 & -s_2 \\ s_2 & s_3^* & -s_1^* & 0 \end{bmatrix}$$

$$X_4 = \begin{bmatrix} s_1 & 0 & -s_2^* & s_3^* \\ 0 & s_1 & -s_3 & s_2 \\ s_2 & s_3^* & s_1^* & 0 \\ -s_3 & s_2^* & 0 & s_1^* \end{bmatrix}$$

Ta nhận thấy rằng các mã này đơn giản hơn các mã trong (9.24) và (9.25) ở chỗ nó không đòi hỏi phải thực hiện các phép cộng và nhân.

1.2.4. Các thuật toán giải mã tối ưu cho mã O – STBC

Trong phần này, chúng ta sẽ điểm qua các phương pháp giải mã tối ưu dành cho mã hóa không gian-thời gian trực giao (O-STBC) nhằm cải thiện hiệu quả truyền dẫn trong các hệ thống truyền thông không dây.

Trước hết, với trường hợp lý tưởng khi máy thu có thông tin chính xác về kênh truyền (perfect channel state information), các thuật toán giải mã tối ưu sẽ được áp dụng để đảm bảo tính hiệu quả trong truyền dẫn. Khi kênh truyền biến đổi chậm (slow fading), kỹ thuật ước lượng kênh dựa trên phương pháp sai số bình phương tối thiểu (Least Square Error - LSE) có thể được áp dụng để tối ưu độ chính xác. Cuối cùng, trong môi trường kênh truyền biến đổi nhanh (fast fading), kỹ thuật giải mã kết hợp với ước lượng kênh theo chu kỳ (cyclic ML) sẽ giúp tăng cường khả năng thích ứng của hệ thống với các thay đổi nhanh chóng của kênh truyền.

1.2.4.1 Giải mã O-STBC cho tập tín hiệu thực.

Khi tín hiệu s_k là tín hiệu thực, các ma trận mã O-STBC có thể được biểu diễn như sau:

$$X = \sum_{k=1}^{N_s} s_k A_k$$

Trong đó A_k có kích thước NxN_t là các ma trận phân tán thỏa mãn các điều kiện sau:

$$A_k A_k^T = I_N,$$

$$A_k A_n^T = -A_n A_k^T, \ k \neq n \text{ với } k = 1,2,\dots,N_s, n = 1,2,\dots,N_s.$$

Giả thiết kênh pha- đinh chậm, tức là các hệ số kênh truyền $h_{m,n}$ không đổi trong khoảng N_t khe thời gian phát (một ma trận mã) , ta có

$$H_{m,n}(t) = h_{m,n}, t = 1,2,\dots,N_t$$

Và khối tín hiệu thu được tương ứng với từ mã phát đi được cho bởi biểu thức:

$$Y = HX + Z$$

Trong đó $H \in C^{MxN}$ là kênh truyền MIMO, $Y \in C^{M \times N_S}$ là ma trận tín hiệu thu $Z \in C^{M \times N_S}$ là tập âm cộng Gauss được giả thiết là tạp âm trắng cả trong miền không gian và miền thời gian với giá trị trung bình bằng 0 và phương sai σ^2.

Do X chỉ bao gồm các giá trị thực, ta có thể viết lại như sau:

$$\tilde{Y} = \tilde{H}X + \tilde{Z}$$

Với $\tilde{Y} = \begin{bmatrix} R\{Y\} \\ J\{Y\} \end{bmatrix}, \tilde{H} = \begin{bmatrix} R\{H\} \\ J\{H\} \end{bmatrix}, v\text{à } \tilde{Z} = \begin{bmatrix} R\{Z\} \\ J\{Z\} \end{bmatrix}$

Khi máy thu biết chính xác kênh truyền H hay $\tilde{H}$, nó thực hiện giải mã tối ưu véc-tơ thông tin s như sau:

$$\bar{s} = argmin \left\| \tilde{Y} - \tilde{H}X \right\|_F^2$$

Trong đó $|A|_F$ ký hiệu đợ lớn Frobenius (Frobenius norm) của ma trận A.

Dựa vào tính chất trực giao của ma trận X, ta có thể biến đối đối số trong vế phải của biểu thức trên như sau:

$$\begin{aligned} \left\| \tilde{Y} - \tilde{H}X \right\|_F^2 &= \left\| \tilde{Y} \right\|_F^2 - 2\left\{ trace\left(\tilde{Y}^T \tilde{H}X \right) + \left\| \tilde{H}X \right\|_F^2 \right\} \\ &= \left\| \tilde{Y} \right\|_F^2 - 2\sum_{k=1}^{N_S} trace\left(\tilde{Y}^T \tilde{H}A_k \right)_{s_k} + trace\left(\|s\|_2^2 \tilde{H} I_N \tilde{H}^T \right) \\ &= \sum_{k=1}^{N_S} \left[-2trace\left(\tilde{Y}^T \tilde{H}A_k \right)_{s_k} + |s_k|^2 \left\| \tilde{H} \right\|_F^2 \right] + const. \\ &= \left\| \tilde{H} \right\|_F^2 \sum_{k=1}^{N_S} [s_k - \bar{s}_k]^2 + const \end{aligned}$$

Trong đó $\tilde{s}_k = \frac{trace(\tilde{Y}^T \tilde{H} A_k)}{\|\tilde{H}\|_F^2}$ là biến thống kê quyết định (decision statistic) tương ứng với ký hiệu s_k, trace(A) ký hiệu vết của ma trận A được tính bằng tổng tất cả các phần tử nằm trên đường chéo chính của A.

Từ công thức rút ra ở trên, ta có thể viết lại luật quyết định tối ưu như sau:

$$\bar{s} = argmin_{s \in A} \sum_{k=1}^{N_s} (s_k - \bar{s}_k)^2$$

Biểu thức trên chứng tỏ rằng metric tối ưu được tách thành tổng của N_s thành phần, trong đó mỗi thành phụ thuộc duy nhất vào một ký hiệu thực $s_n, k \neq n$ theo luật sau:

$$\bar{s}_k = argmin_{s_k \in A} (s_k - \tilde{s}_k)^2$$

1.2.4.2 Giải mã O-STBC cho tập tín hiệu phức.

Khi tín hiệu s_k là tín hiệu phức, các ma trận mã O-STBC có thể được biểu diễn như trong công thức trong phần mã O-STBC cho tập tín hiệu phức, với các ma trận phân tán thỏa mãn các điều kiện sau:

$$A_k^H A_k = C I_N, \quad B_k^H B_k = C I_N, với\ C\ là\ hằng\ số\ bất\ kỳ$$

$$A_k^H A_n = -A_n^H A_k, \qquad B_k^H B_n = -B_n^H B_k, \qquad k \neq n$$

$$A_k^H B_k = B_k^H A_n$$

Trong đó k=1,2,...K, n=1,2,...K.

Tương tự như trong trường hợp tín hiệu thực, nếu chúng ta giả thiết kênh H không thay đổi trong thời gian một từ mã, ta có thể viết ma trận tín hiệu thu được.

Khi máy thu biết chính xác kênh truyền H, nó thực hiện giải mã tối ưu véc-tơ thông tin như sau:

$$\bar{s} = argmin_{s \in A} \|Y - HX\|_F^2$$

Khi các ma trận phân tán A_k và B_k thỏa mãn các điều kiện , thì ma trận X thỏa mãn điều kiện. Do đó ta có thể biến đổi đối số trong vế phải của biểu thức trên như sau:

$$\|Y - HX\|_F^2 = \|Y\|_F^2 - 2R\{trace\left(\tilde{Y}^T HX\right)\} + \|HX\|_F^2$$

$$= \|Y\|_F^2$$

$$- 2\sum_{k=1}^{N_s} R\{trace(Y^H HA_k)\}R\{s_k\} + +trace(C\|s\|_2^2 HI_N H^T)$$

$$= \sum_{k=1}^{N_s} \left[-2trace\left(\tilde{Y}^T \tilde{H} A_k\right)_{s_k} + |s_k|^2 \left\|\tilde{H}\right\|_F^2\right] + \ const.$$

$$= \left\|\tilde{H}\right\|_F^2 \sum_{k=1}^{N_s} [s_k - \ \bar{s}_k]^2 + \ const$$

1.3. ĐIỀU CHẾ KHÔNG GIAN – SM

1.3.1. Giới thiệu sơ lược về điều chế không gian

Các kỹ thuật truyền dẫn nhiều đầu vào – nhiều đầu ra (MIMO) với nhiều ăng ten được trang bị ở máy phát và/hoặc máy thu đã thu hút được sự quan tâm nghiên cứu rất lớn trong thập kỷ vừa qua, cả trong học thuật và ứng dụng thực tế **Error! Reference source not found.** Người ta nhận thấy rằng các hệ thống MIMO hiện nay có các nhược điểm chính là độ phức tạp và giá thành hệ thống cao. Các nguyên nhân chính dẫn đến việc tăng độ phức tạp và giá thành của các hệ thống MIMO bao gồm:

- Nhiễu xuyên kênh (ICI), gây ra bởi việc chồng lấn giữa nhiều luồng tín hiệu độc lập được phát đi từ nhiều ăng ten.
- Đồng bộ giữa các ăng ten (IAS), là giả thiết căn bản cho các phương pháp mã hóa không gian – thời gian.

- Nhiều máy thu phát tần số vô tuyến (RF), dùng để phát đi các tín hiệu một cách đồng thời và thường có giá thành cao.

Một trong những kỹ thuật truyền dẫn MIMO mới được Mesleh và các cộng sự đề xuất gần đây là kỹ thuật điều chế không gian. Kỹ thuật này được phát triển với mục tiêu giảm độ phức tạp và giá thành của các hệ thống nhiều ăng ten mà không làm suy giảm phẩm chất lỗi bit đồng thời vẫn đảm bảo tốc độ dữ liệu đủ lớn. Trong hệ thống SM, máy phát kích hoạt một trong số n_T phần tử ăng ten phát tại mỗi chu kỳ tín hiệu và phát đi một tín hiệu được điều chế bằng các kỹ thuật điều chế truyền thống, như điều chế biên độ cầu phương (QAM) hoặc khóa dịch pha (PSK) với chòm sao tín hiệu M điểm. Bằng cách đó, thông tin được truyền đi bởi cả ký hiệu được điều chế và chỉ số của ăng ten được kích hoạt để phát đi ký hiệu này. Một trường hợp đặc biệt của SM, được gọi là khóa dịch không gian (SSK), đã được đề xuất bởi Jeganathan và các cộng sự trong **Error! Reference source not found.**bằng cách phát đi một mức năng lượng nào đó từ một ăng ten được lựa chọn phù hợp với các bit thông tin cần truyền đi mà không phát đi các ký hiệu được điều chế QAM hay PSK. Sự đơn giản và hiệu quả của kỹ thuật SM được thể hiện qua những đặc điểm sau:

- Tại mỗi chu kỳ tín hiệu, chỉ một ăng ten phát được kích hoạt để truyền dữ liệu. Điều này giúp cho SM tránh hoàn toàn hiện tượng ICI, loại bỏ yêu cầu đồng bộ giữa các ăng ten phát và chỉ cần sử dụng một máy thu phát RF. Hơn nữa, việc chỉ kích hoạt một ăng ten phát cho phép hệ thống SM thực hiện tách sóng hợp lẽ tối ưu (ML) tại máy thu với độ phức tạp thấp.
- Vị trí không gian của mỗi ăng ten phát trong mảng ăng ten được sử dụng như là một nguồn tin thông qua việc ánh xạ một – một giữa các bit thông tin cần truyền đi với chỉ số của các ăng ten phát. Vì vậy, kỹ thuật SM cho phép chúng ta thu được tăng ích ghép kênh theo không gian khi so với hệ thống đơn ăng ten truyền thống.

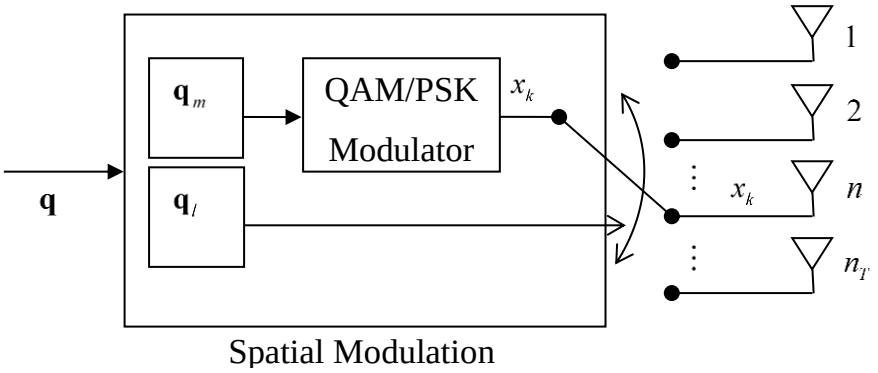
Như thảo luận trong, các hệ thống thông tin vô tuyến trước đây và hiện nay luôn tập trung để nâng cao hiệu suất phổ tần, một thông số có liên hệ chặt chẽ với dung lượng Shannon. Hiệu suất phổ tần được xem là một trong những chỉ số phẩm chất chính đóng vai trò quan trọng trong thiết kế và tối ưu các hệ thống thông tin vô tuyến nói chung và các mạng thông tin tế bào nói riêng. Vì vậy, hầu hết các công nghệ truyền dẫn và giao

thức trong các mạng thông tin di động và tế bào đang hoạt động đều được thiết kế dựa trên nhiều yếu tố như dung lượng, chất lượng dịch vụ, khả năng mở rộng, …, mà không xét đến năng lượng tiêu thụ. Với phương pháp thiết kế này, các hệ thống thông tin tế bào hiện tại chỉ có thể tiết kiệm được năng lượng tiêu thụ bằng cách đánh đổi chất lượng hoặc dung lượng. Do đó, các công nghệ truyền dẫn và giao thức cho các mạng thông tin tế bào thế hệ mới cần phải được thiết kế và tối ưu bằng cách sử dụng nhiều tiêu chí phẩm chất hơn, đặc biệt là tiêu chí về năng lượng tiêu thụ.

Trong bài báo này, chúng tôi giới thiệu về kỹ thuật SM với các nội dung chính bao gồm: i) mô tả phương thức điều chế, ánh xạ thông tin trong một hệ thống SM; ii) mô tả các thuật toán khôi phục tín hiệu trong một hệ thống SM đồng thời đánh giá hiệu suất năng lượng của kỹ thuật thông qua mô phỏng máy tính này nhằm làm rõ những ưu điểm của kỹ thuật SM.

1.3.2 Phương thức hoạt động và mô hình hệ thống

1.3.2.1 Phương thức hoạt động của hệ thống SM



Hình 1.7 Minh họa về máy phát điều chế không gian

Hình 1.7 minh họa nguyên lý hoạt động của một máy phát điều chế không gian. Trong mỗi chu kỳ tín hiệu, một ăng ten được kích hoạt và phát đi một ký hiệu trong chòm sao QAM hoặc PSK. Theo Hình 1.7, $\mathbf{q}$ là cụm dữ liệu nhị phân đến bao gồm $(m+l)=\log_2(Mn_T)$ bit thông tin, với M là kích thước chòm sao QAM/PSK. Cụm bit thông tin $\mathbf{q}$ sẽ được tách ra thành 2 phần: một phần là $\mathbf{q}_m$ bao gồm $m=\log_2(M)$ bit được ánh xạ vào một ký hiệu QAM/PSK x_k, $1\le k\le M$, phần còn lại $\mathbf{q}_l$ bao gồm $l=\log_2(n_T)$ bit sẽ xác định ăng ten nào được kích hoạt để phát đi x_k.

Với $n_T=4$ ăng ten phát và sử dụng điều chế 4-QAM, hệ thống SM có thể truyền đi 4 bit trong mỗi chu kỳ tín hiệu với phép ánh xạ được mô tả trên Bảng 1. Ví dụ, khi cụm bit đầu vào là $\mathbf{q}=(0\quad 0\quad 0\quad 0)$ thì ký hiệu $x_k=(1+j)$ sẽ được phát đi từ ăng ten số 1. Nếu 2 bit đầu đổi từ $(0\quad 0)$ thành $(0\quad 1)$ thì ký hiệu $x_k=(1+j)$ sẽ được phát đi từ ăng ten số 2.

Bảng 1: Minh họa phép ánh xạ các cụm bit thông tin vào chỉ số ăng ten và ký hiệu được điều chế 4-QAM

Cụm bit $\mathbf{q}$	***Cụm bit*** $\mathbf{q}_l$	***Chỉ số ăng ten*** n	***Cụm bit*** $\mathbf{q}_m$	***Ký hiệu phát*** x_k
0000	00	1	00	$1+j$
0001	00	1	01	$-1+j$

0010	00	1	10	$1-j$
0011	00	1	11	$-1-j$
0100	01	2	00	$1+j$
0101	01	2	01	$-1+j$
0110	01	2	10	$1-j$
0111	01	2	11	$-1-j$
1000	10	3	00	$1+j$
1001	10	3	01	$-1+j$
1010	10	3	10	$1-j$
1011	10	3	11	$-1-j$
1100	11	4	00	$1+j$
1101	11	4	01	$-1+j$
1110	11	4	10	$1-j$
1111	11	4	11	$-1-j$

1.3.2.2 Mô hình hệ thống

Trong hệ thống SM, ta biểu diễn véc tơ tín hiệu phát $\mathbf{s}$ như sau :

$$\mathbf{s} = \begin{bmatrix} 0 & 0 & \cdots & \underset{\text{vÞtrÝthø } n}{s_k} & \cdots & 0 \end{bmatrix}^T \qquad (1)$$

Trong đó $\mathbf{x}^T$ biểu thị chuyển vị của véc tơ $\mathbf{x}$. $\mathbf{s}$ bao gồm n_T phần tử trong đó chỉ có một phần tử duy nhất khác 0 là s_k tại vị trí ăng ten thứ n, $1 \le n \le n_T$. Chúng ta giả thiết

công suất phát được chuẩn hóa sao cho $E\{\mathbf{s}^H\mathbf{s}\}=1$. Phép ánh xạ giữa $\mathbf{q}$ và $\mathbf{s}$ là phép ánh xạ một – một. Véc tơ $\mathbf{s}$ được phát đi qua kênh MIMO để tới máy thu.

Véc tơ tín hiệu thu, $\mathbf{y}$, kích thước $n_R \times 1$ được cho bởi biểu thức sau:

$$\mathbf{y} = \sqrt{\gamma}\mathbf{H}\mathbf{s} + \mathbf{w} = \sqrt{\gamma}\mathbf{h}_{(v=n)}s_k + \mathbf{w} \quad (2)$$

Trong đó, $\mathbf{H}$ là ma trận kênh truyền kích thước $n_R \times n_T$ với $\mathbf{h}_v = \begin{bmatrix} h_{1,v} & h_{2,v} & \cdots & h_{n_R,v} \end{bmatrix}^T$ là cột thứ v. Mỗi phần tử của ma trận $\mathbf{H}$ được giả thiết là các biến ngẫu nhiên độc lập đồng nhất (i.i.d.) có phân bố Gauss với trị trung bình bằng 0 và phương sai bằng 1. Nói cách khác, $\mathbf{H}$ là một kênh MIMO pha đinh phẳng. $\mathbf{w}$ là véc tơ tạp âm kích thước $n_R \times 1$ với các phần tử được giả thiết là các biến ngẫu nhiên i.i.d. có phân bố Gauss với trị trung bình bằng 0 và phương sai 1. γ biểu thị tỷ số công suất tín hiệu trên công suất tạp âm (SNR) tại máy thu.

Trong phạm vi bài báo này, chúng ta giả thiết kênh truyền là cận tĩnh, tức là $\mathbf{H}$ giữ không đổi trong một khoảng η chu kỳ tín hiệu và thay đổi độc lập sau mỗi η chu kỳ.

1.3.3 Các thuật toán khôi phục tín hiệu

Sử dụng kỹ thuật điều chế không gian, dữ liệu được mã hóa trong một ký hiệu QAM hoặc PSK và một chỉ số ăng ten. Vì vậy, việc khôi phục tín hiệu phát được thực hiện bằng cách ước lượng cả chỉ số ăng ten phát ký hiệu QAM/PSK. Trong mục này chúng tôi trình bày hai thuật toán khôi phục tín hiệu trong điều chế không gian.

1.3.3.1 Thuật toán khôi phục tín hiệu i-MRC:

Trong thuật toán này, với giả thiết máy thu biết chính xác các hệ số kênh truyền, véc tơ tín hiệu thu $\mathbf{y}$ được lần lượt tính thử với các hệ số kênh truyền nhằm ước lượng cả ký hiệu thông tin từ phía phát và vị trí ăng ten phát. Thuật toán được mô tả theo các bước như sau **Error! Reference source not found.**:

Bước 1: Tính $g_n = \mathbf{h}_n^H \mathbf{y}$ *với* $n = 1 : n_T$.

Bước 2: Xác định chỉ số ăng ten phát theo biểu thức $\hat{n} = \arg\max_n |g_n|$.

Bước 3: Xác định ký hiệu phát theo biểu thức $\hat{s}_k = Q(g_{\hat{n}})$, *với* $Q(\ )$ *là toán tử quyết định hay lượng tử hóa.*

Bước 4: Khôi phục cụm bit thông tin truyền đi dựa vào $\hat{n}$ *và* $\hat{s}_k$.

Theo thảo luận và chứng minh trong **Error! Reference source not found.**, kết quả mô phỏng sử dụng thuật toán i-MRC chỉ đúng trong trường hợp các cột của ma trận kênh truyền thỏa mãn điều kiện $\frac{|\mathbf{h}_n^H \mathbf{h}_k|}{\|\mathbf{h}_n\|_F^2} \le 1$, trong đó $\|\mathbf{x}\|_F$ biểu thị chuẩn (norm) Frobenius của véc tơ $\mathbf{x}$. Áp dụng bất đẳng thức Cô si, ta thấy rằng $\|\mathbf{h}_k\|_F \le \|\mathbf{h}_n\|_F$ chính là điều kiện cần để việc ước lượng chỉ số ăng ten không bị sai trong trường hợp không có tạp âm. Một cách để đảm bảo điều kiện này là tạo ra một kênh truyền MIMO có ràng buộc thông qua việc chuẩn hóa kênh trước khi truyền tin **Error! Reference source not found.**, tức là kênh truyền được chuẩn hóa sao cho $\|\mathbf{h}_k\|_F^2 = c$, với c là một hằng số.

1.3.3.2 Thuật toán khôi phục tín hiệu tối ưu:

Bộ tách tối ưu tìm kiếm đồng thời cả chỉ số ăng ten và tín hiệu phát dựa trên nguyên tắc hợp lẽ tối ưu (ML) như sau **Error! Reference source not found.**:

$$\begin{aligned}\left[\hat{n}, \hat{k}\right] &= \arg\min_{n,k} p_Y(\mathbf{y} \mid \mathbf{s}, \mathbf{H}) \\ &= \arg\min_{n,k} \sqrt{\rho}\|\mathbf{g}_{nk}\|^2 - 2\,\mathrm{Re}\{\mathbf{y}^H \mathbf{g}_{nk}\}\end{aligned} \tag{3}$$

với $\mathbf{g}_{nk} = \mathbf{h}_n s_k, 1 \le n \le n_T, 1 \le k \le M$, $p_Y(\mathbf{y} \mid \mathbf{s}, \mathbf{H}) = \pi^{-n_R} \exp\left(-\left\|\mathbf{y} - \sqrt{\gamma}\mathbf{h}_n s_k\right\|\right)$ là hàm mật độ phân bố xác suất của $\mathbf{y}$ khi biết trước $\mathbf{s}$ và $\mathbf{H}$.

Công thức (3) cho ta thấy rằng, phương pháp điều chế SM cho phép thực hiện tách sóng ML một cách đơn giản do đã loại trừ hoàn toàn nhiễu đồng kênh.

1.4 KẾT HỢP ĐIỀU CHẾ KHÔNG GIAN VÀ MÃ HÓA KHỐI KHÔNG GIAN – THỜI GIAN

1.4.1 Giới thiệu về sự kết hợp SM và STBC

Mã hóa khối không gian-thời gian trực giao được điều chế theo không gian (SM–OSTBC), dựa trên khái niệm từ mã Chòm sao không gian (SC) được giới thiệu bởi thầy Lê Minh Tuấn. Trong sơ đồ đề xuất, ma trận từ mã truyền được tạo ra bằng cách nhân ma trận SC với các từ mã được xây dựng từ Mã khối không gian-thời gian trực giao (O–STBC). Hiệu suất phổ cực đại của sơ đồ đề xuất bằng $N_T - 2\log_2 M$ bpcu, trong đó n_T là số lượng ăng-ten phát, M là bậc điều chế.

Ma trận SC cung cấp phương tiện mang các bit thông tin cùng với các từ mã O–STBC và cho phép sơ đồ SM–OSTBC đạt được phân tập truyền thứ hai bằng cách đáp ứng đặc tính xác định không biến mất. Trình bày một phương pháp có hệ thống để thiết kế các từ mã SC cho số lượng anten phát chẵn lớn hơn 3. Bộ giải mã khả năng tối đa một luồng (ML), yêu cầu độ phức tạp tính toán thấp nhờ cấu trúc của từ mã SM–OSTBC và tính trực giao của O–STBC, và bộ giải mã hình cầu với độ phức tạp xử lý tín hiệu giảm hơn nữa được phát triển

Hệ thống MIMO đã được chứng minh về mặt lý thuyết và thực tế là cải thiện đáng kể dung lượng và độ tin cậy so với các hệ thống không dây ăng-ten đơn truyền thống.

Ví dụ : sơ đồ phân lớp không-thời gian (V–BLAST) của Phòng thí nghiệm Bell dọc có khả năng đạt được độ lợi ghép kênh cao bằng cách truyền các luồng dữ liệu song song từ các ăng-ten phát khác nhau, nhưng phải trả giá bằng độ phức tạp giải mã đáng kể cần thiết để giảm tác động của Nhiễu giữa các kênh (ICI). Ngược lại, các mã khối không-thời gian trực giao (O–STBC), 1, được phát minh ra để đạt được sự phân tập đầy đủ, tức là, thứ tự phân tập tối đa có thể đạt được. Tuy nhiên, O-STBC phân tập đầy đủ tốc độ một chỉ tồn tại cho 2 ăng ten phát.

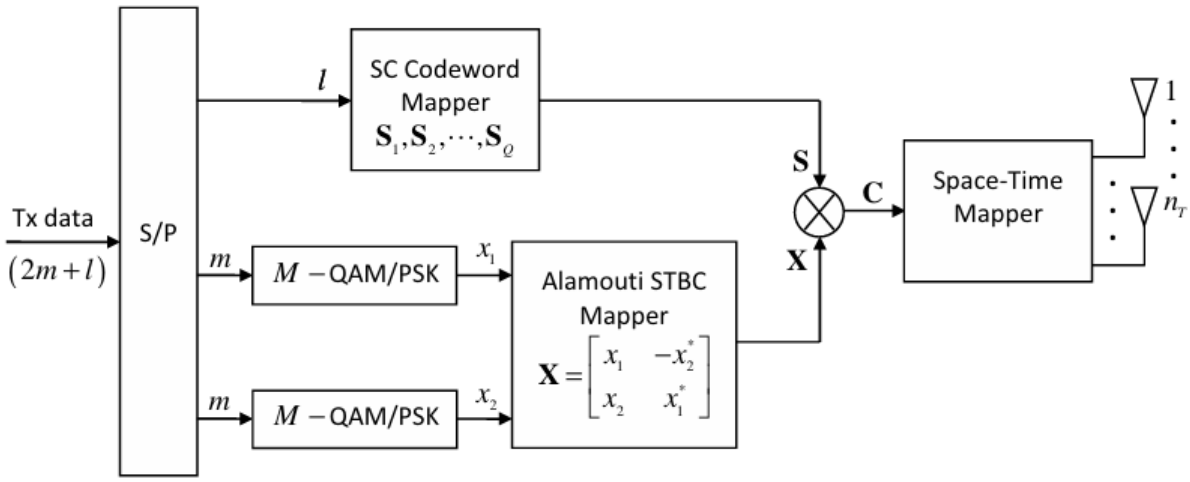
SM-OSTBC có thể được triển khai với số lượng ăng-ten phát chẵn bất kỳ $n_T \geq 4$ và cung cấp một phương pháp thiết kế mã SC (Spatial Constellation) hệ thống, không cần phải tìm kiếm bằng máy tính phức tạp.

SM-OSTBC có thể tích hợp vào các chuẩn truyền thông hiện tại, như LTE-Advanced, nhờ khả năng hỗ trợ nhiều cấu trúc mã hóa không gian-thời gian hiện đại.

Sử dụng các mã Alamouti STBC giúp giảm độ phức tạp trong việc phát hiện tín hiệu nhờ khả năng sử dụng bộ giải mã ML (Maximum Likelihood) với một luồng duy nhất (low-complexity single-stream ML decoder).

1.4.2 Mô hình hệ thống kết hợp SM-STBC

Mô hình hệ thống.



Hình 1.8 Sơ đồ khối máy phát SM-OSTBC

Hình 1.8 trên là sơ đồ kiến trúc của máy phát SM-OSTBC được đề xuất. Việc mã hóa được tóm tắt như sau. Cứ 2 chu kỳ tín hiệu liên tiếp ,(2m+1) bit đi vào bộ phát SM-OSTBC. L bit đầu tiên được đưa vào SC Codeword Mapper để chọn 1 trong số các Q SC,($S_1, \dots, S_Q, Q = 2^l$) là từ mã SC được truyền S. 2m còn lại ($m - \log_2 M$) được điều chế bởi bộ điều chế M-QAM/PSK, cung cấp cặp ký hiệu (x_1, x_2), sau đó được đưa vào Bộ ánh xạ Alammoti STBC để tạo ra từ mã Alammoti STBC X. Từ mã SM-OSTBC C

thu được bằng cách nhân S với X, tức là C = SX, và sau đó được truyền qua nT hoặc một tập hợp con của nT ăng ten có sẵn tại máy phát.

Bằng cách giả sử các ăng-ten n_R ở máy thu và kênh Fading Rayleigh gần như tĩnh, ma trận tín hiệu Y nR × 2 nhận được có thể được biểu diễn như sau:

$$Y = \sqrt{\gamma}HC + N$$

$$= \sqrt{\gamma}HSX + N$$

trong đó H và N lần lượt biểu thị ma trận kênh $n_R \times n_T$ và ma trận nhiễu $n_R \times 2$. H và N được coi là các biến ngẫu nhiên Gausian phức tạp độc lập và được phân phối giống hệt nhau với giá trị trung bình bằng 0 và phương sai đơn vị. Ngoài ra, H được coi là không đổi trong một từ mã có 2 chu kỳ ký hiệu và thay đổi độc lập từ từ mã này sang từ mã khác. C,X và S lần lượt biểu thị từ mã $n_T \times 2$ SM-OSTBC, từ mã 2x2 Alamouti STBC và từ mã $nT \times 2$ SC. Từ mã truyền C được chuẩn hóa sao cho trung bình tổng thể của trace $C^H C$ bằng 2, tức là $E\{tr(C^H C)\} = 2\,.\gamma$ là SNR trung bình tại mỗi ăng-ten thu.

Tuy nhiên, việc thiết kế các sơ đồ SM–OSTBC dựa trên các OSTBC có thứ tự đa dạng cao có thể dẫn đến những thách thức thiết kế sau:

- Tìm kiếm trên máy tính có lẽ là cách duy nhất để xây dựng các từ mã SC cho số lượng ăng-ten phát tùy ý. Trong thực tế, rất khó để thiết kế các từ mã SC một các có hệ thống khi số khi số lượng cột trong ma trận mã SC tăng lên.
- Đối với ví dụ trong [41], việc tăng hiệu suất phổ lên 1 bpcu đòi hỏi số lượng từ mã SC phải tăng theo hệ số 2T nếu sử dụng OSTBC có chu kỳ ký hiệu T có độ dài mã. Điều này dẫn đến sự gia tăng không đáng kể về độ phức tạp phát hiện, khối lượng tính toán cao để tìm kiếm các từ mã SC, cũng như giảm độ lợi mã hóa, đặc biệt đối với T ≥ 4.

➢ Những nhược điểm trên có thể làm cho việc kết hợp sơ đồ đề xuất với OSTBC có độ dài mã lớn hơn hoặc bằng 4 trở nên kém hấp dẫn hơn.

1.4.3 Các thuật toán giải mã trong hệ thống kết hợp.

Nếu sử dụng điều chế QAM, sơ đồ SM–OSTBC được đề xuất có thể được biểu diễn dưới dạng sơ đồ G–STSK. Do đó, quá trình giải điều chế có thể được thực hiện với sự hỗ trợ của bộ giải mã ML đa luồng từ tài liệu [38]. Kết quả là, nó yêu cầu một tìm kiếm toàn diện qua tất cả các tổ hợp của ma trận phân tán và các ký hiệu điều chế, dẫn đến độ phức tạp cao trong giải điều chế. Để tránh vấn đề này, trong phần này, hai bộ giải điều chế đơn luồng có độ phức tạp thấp được đề xuất cho sơ đồ SM–OSTBC. Bộ giải điều chế đầu tiên là ML tối ưu và được gọi là SO-ML. Bộ thứ hai dựa trên nguyên lý SD [47], [48] và được gọi là SO-SD. Độ phức tạp thấp của cả hai bộ giải điều chế bắt nguồn từ cấu trúc của mã SM–OSTBC cũng như từ tính trực giao của mã STBC Alamouti.

1.4.3.1 Bộ giải mã SO-ML

Giả sử rằng thông tin trạng thái kênh hoàn hảo có sẵn tại bộ thu. Gọi ΩS là không gian tìm kiếm tương ứng với các từ mã SC S Gọi ΩX là không gian tìm kiếm tương ứng với các từ mã Alamouti STBC X. Khi đó, bộ giải điều chế ML tối ưu của sơ đồ SM–OSTBC được đề xuất có thể được biểu diễn như sau:

$$\left(\hat{S}, \hat{X}\right) = \arg \min_{S \in \Omega_S, X \epsilon \Omega_X} \left\| Y - \sqrt{\gamma}\, HSX \right\|^2$$

Cho một ma trận $S_q \in \Omega_S, q = 1, \dots, Q$. Sau đó tương ứng n_R×2 tương đương ma trận $\widetilde{H}_q = HS_q$có thể thu được.

Do đó, (1) rút gọn về sơ đồ Alamouti nổi tiếng:

$$Y = \sqrt{\gamma}\widetilde{H}_q X + N$$

và quy tắc giải điều chế ML trong (9) giảm xuống quy tắc đó cho X điều kiện theo Sq, như sau:

$$\left(\hat{X}\right)_q = \arg \min_{X \in \Omega_x} \left\| Y - \sqrt{\gamma}\widetilde{H}_q X \right\|^2$$

Từ công thức trên đẳng thức sau xảy ra:

$$\left\|Y-\sqrt{\gamma}\widetilde{H}_qX\right\|^2=\|Y\|^2-2\sqrt{\gamma}R\{tr(Y^H\widetilde{H}_qX)\}+\gamma\left\|\widetilde{H}_qX\right\|^2$$

Bằng cách sử dụng biểu diễn phân tán tuyến tính của từ mã Alamouti STBC X trong phương trình (2), tính trực giao của X, và bỏ qua một số hằng số không liên quan, phương trình (12) có thể được viết lại như sau:

$$\left\|Y-\sqrt{\gamma}\widetilde{H}_qX\right\|^2=\sum_{n=1}^{2}[\gamma\left\|\widetilde{H}_q\right\|^2(\tilde{x}_n^2+\check{x}_n^2)\ -\ 2\sqrt{\gamma}R\{tr(Y^H\widetilde{H}_q(A_n\tilde{x}\ +\ jB_n\check{x}_n))\}]$$

$$=\gamma\left\|\widetilde{H}_q\right\|^2\sum_{n=1}^{2}[(\tilde{x}_n^2+\check{x}_n^2)\ -\ 2\frac{R\{tr(Y^H\widetilde{H}_qA_n\}}{\sqrt{\gamma}\left\|\widetilde{H}_q\right\|^2}\tilde{x}_n+\ 2\frac{R\{tr(Y^H\widetilde{H}_qB_n\}}{\sqrt{\gamma}\left\|\widetilde{H}_q\right\|^2}\check{x}_n]$$

Với sự trợ giúp của một số phép biến đổi đại số, phương trình (13) có thể được đơn giản hóa như sau:

$$\left\|Y_q-\widetilde{H}_qX\right\|^2=\gamma\left\|\widetilde{H}_q\right\|^2[(\sum_{n=1}^{2}d_q(\tilde{x}_n)+\sum_{n=1}^{2}d_q(\check{x}_n))-\ R_q]$$

Ở đó $d_q(\tilde{x}_n)$và $d_q(\check{x}_n)$là khoảng cách Euclide của phần thực và phần ảo của ký hiệu truyền x_n:

$$d_q(\tilde{x}_n)=\left(\tilde{x}_n-\ \bar{x}_{I,n}^q\right)^2$$

$$d_q(\check{x}_n)=\left(\check{x}_n-\ \bar{x}_{Q,n}^q\right)^2$$

Và

$$R_q=\sum_{n=1}^{2}[\left(\bar{x}_{I,n}^q\right)^2+\left(\bar{x}_{Q,n}^q\right)^2]$$

Do đó, quy tắc giải mã ML có điều kiện với Sq trong phương trình (11) có thể được viết lại như sau:

$$(\hat{x}_1^q,\hat{x}_2^q)=\arg\min_{(x_1,x_2)\in\Omega_x}(\sum_{n=1}^{2}d_q(\tilde{x}_n)+\sum_{n=1}^{2}d_q(\check{x}_n)\ )$$

Hoặc tương đương với phương trình dưới đây:

$$\hat{x}_n^q = \arg \min_{x_n \in \Omega_x} d_q(x_n)$$

ở đó:

$$d_q(x_n) = d_q(\tilde{x}_n) + d_q(\check{x}_n)$$

Cho n=1,2 and $\Omega_x$là một tập hợp M-QAM/PSK

Từ $(\hat{x}_1^q, \hat{x}_2^q), q = 1,2, \dots, Q$ được suy ra từ công thức trên chỉ số qˆtương ứng với từ mã

S=S_q được tính như sau:

$$\hat{q} = \arg \min_q D_q$$

ở đó:

$$D_q = \gamma \left\| \widetilde{H}_q \right\|^2 [\sum_{n=1}^{2} d_q(\hat{x}^q{}_n) - R_q]$$

Cuối cùng từ mã được truyền được ước tính là $\hat{S} = S_{\hat{q}}(\hat{x}_{1,}\hat{x}_2) = (\hat{x}_1^{\hat{q}}, \hat{x}_2^{\hat{q}})$.

Do đó, bộ giải mã SO–ML được đề xuất có thể được xây dựng như sau:

1. Với mỗi ma trận $\widetilde{H}_q$ và với mỗi cặp tín hiệu $x_{1,m}, x_{2,m}$ trong chòm sao phát Ω_x, hãy tính hai khoảng cách Euclide từ (*) như sau:

$$d_{1,q}^m = d_q(x_{1,m}) \; với \; m = 1, \dots, M.$$

$$d_{2,q}^m = d_q(x_{2,m}) \; với \; m = 1, \dots, M$$

2. Tìm mức tối thiểu trong số các giá trị M của $d_{1,q}^m$ và $\hat{x}_1^q$ tương đương với $d_{1,q}^{min}$.
3. Tìm giá trị nhỏ nhất giữa M giá của $d_{2,q}^m$ và $\hat{x}_2^q$ tương đương với $d_{2,q}^{min}$.
4. Tình $d_q = d_{1,q}^{min} + d_{2,q}^{min}$ với q=1,...Q.
5. Tìm vị trí $\hat{q}$ tương ứng với khoảng cách nhỏ nhất
   d_q^{min} giữa các giá trị Q của giá trị d_q

6. Tính toán ma trận SC ước tính và các ký hiệu được truyền đi như : $\hat{S} = S_{\hat{q}}$ và $(\hat{x}_1, \hat{x}_2) = \hat{x}^{\hat{q}}{}_1 \hat{x}_2^{\hat{q}}$ tương ứng.
7. Từ $\hat{S}$ và $(\hat{x}_1 \hat{x}_2)$, ước tính $(2m+1)$ bit thông tin

Các sơ đồ giải điều chế có độ phức tạp thấp tương tự đã được đề xuất trong [34] và [36] nhưng dành cho các sơ đồ SM-MIMO khác nhau.

1.4.3.2 Bộ giải mã SO-SD

Độ phức tạp xử lý tín hiệu của bộ giải điều chế ML của phần trên có thể được giảm bớt với sự trợ giúp của SD [48]. Chúng ta hãy bắt đầu bằng cách viết lại (10) như sau:

$$y = \bar{H}_q x + n \ (*)$$

Trong đó $x = [x_1 \quad x_2]^T$ và $\bar{H}_q$ có dạng

$$\bar{H}_q = \begin{bmatrix} \tilde{h}_{q,11} & \tilde{h}_{q,12} \\ \tilde{h}^*_{q,12} & -\tilde{h}^*_{q,11} \\ \vdots & \vdots \\ \tilde{h}_{q,nR^1} & \tilde{h}_{q,nR^2} \\ \tilde{h}^*_{q,nR^2} & -\tilde{h}^*_{q,nR^2} \end{bmatrix} = \begin{bmatrix} \bar{h}_{q,1} & \bar{h}_{q,2} \end{bmatrix}$$

Từ công thức (*), quy tắc giải mã ML cho x dựa trên S_q là

$$\left(\hat{X}\right)_q = \arg \min_{x \in \Omega_x} \left\| y - \bar{H}_q x \right\|^2 \ (**)$$

Với sự trợ giúp của thuật toán Gram-Schmidt (MGS) đã sửa đổi, tức là phân tách QR, $\bar{H}_q$ có thể được viết lại như sau:

$$\bar{H}_q = Q_q R_q$$

Trong đó Q_q là ma trận cột trực giao chuẩn đơn vị $2n_R \times 2$ tức là $Q_q^H Q_q = I_2$ và R_q là à ma trận tam giác trên 2x2. Vì $\bar{h}_{q,1}^H \bar{h}_{q,2} = \bar{h}_{q,2}^H \bar{h}_{q,1}$ và $\left\|\bar{h}_1\right\| = \left\|\bar{h}_2\right\|$ nên R_q có thể được viết như sau:

$$R_q = \begin{bmatrix} R_q & 0 \\ 0 & R_q \end{bmatrix}$$

Hãy nhân trước cả hai vế của (*) với Qq. Chúng tôi có được:

$$v_q = R_q x + w_q$$

Trong đó v_q và w_q lần lượt là vectơ tín hiệu nhận được 2 × 1 và vectơ nhiễu 2 × 1 sau khi phân tách QR. Bằng cách sử dụng đẳng thức:

$$\left\|v_q - R_q x\right\|^2 = \left\|y - H_q x\right\|^2 - y^h y + v_q^H v_q$$

quy tắc giải mã ML trong (**) có thể được viết lại như sau:

$$\left(\hat{X}\right)_q = \arg\min_{x\in\Omega_x}\left\|v_q - R_q x\right\|^2$$

SD có thể được áp dụng cho công thức trên để tìm kiếm $\left(\hat{X}\right)_q$ Sau khi tìm được $\left(\hat{X}\right)_q$ với q=1,2,...,Q. thì phải xác định các nghiệm tối ưu của $\hat{q}$ và $\hat{X} = \left(\hat{X}\right)_{\hat{q}}$ Điều này có thể thực hiện được như sau:

$$\hat{q} = \arg\min_{q\in 1,\ldots,Q}\left\|v_q - R_q\left(\hat{X}\right)_q\right\|^2 + y^H y - v_q^H v_q$$

Để tiếp tục giảm độ phức tạp tính toán của SD, số hạng $y^H y - v_q^H v_q$ trong công thức trên có thể được đưa vào. ài khoản ngay từ đầu. Cụ thể hơn, chúng ta hãy biểu thị bán kính hình cầu ban đầu bằng C0. SD có thể tìm kiếm các vectơ $\left(\hat{X}\right)_q$ trong số các vectơ $x \in \Omega_x$ thỏa mãn điều kiện sau:

$$\left\|v_q - R_q x\right\|^2 \le C_0 - y^H y - v_q^H v_q$$

Nếu $C_0 - y^H y - v_q^H v_q \le 0$ thì không cần áp dụng SD cho (33), vì các điểm tín hiệu nằm ngoài phạm vi bán kính C_0.

Dựa trên bộ giải mã PMLD và QMLD được phát minh để phát hiện tín hiệu được điều chế PSK và QAM [45], ương ứng, bộ giải mã SO-SD cho sơ đồ SM–OSTBC được đề xuất có thể được tóm tắt như sau:

1. Đặt q=1, $D_{opt} = C_0, \hat{q} = \emptyset, \hat{x} = \emptyset$.

2. Tính $\widetilde{H}_q = HS_q, \overline{H}_q, \overline{H}_q = QR$ và $v = Q^H y$
3. Tính bán kính hình cầu $R = D_{opt} - (y^H y - v_q^H v_q)$ và đặt $D_{q,min} = \infty$.
4. Nếu $R > 0$ áp dụng bộ giải mã QMLD hoặc bộ giải mã PMLD tới (33) và tìm kiếm $\left(\hat{X}\right)_q$ bằng cách sử dụng R là bán kính . Sau đó tính $D_{q,min} = \left\| v_q - R_q\left(\hat{X}\right)_q \right\|^2 + (y^H y - v_q^H v_q)$
5. Nếu $D_{q,min} < D_{opt}$ lưu đáp án mới $\hat{x} = (\hat{x})_q, \hat{q} = q$ và $gán\ D_{opt} = D_{q,min}$
6. Đặt $q = q + 1$. $Nếu\ q \leq Q$ sau đó chuyển sang bước 2. Nếu $q > Q$ và $\hat{q} = \emptyset$ tức không thấy nhiệm. tăng bán kính hình cầu ban đầu C_0 và chuyển sang Bước 1. Ngược lại, dừng và đưa ra nghiệm $\hat{x}$ và $\hat{q}$.

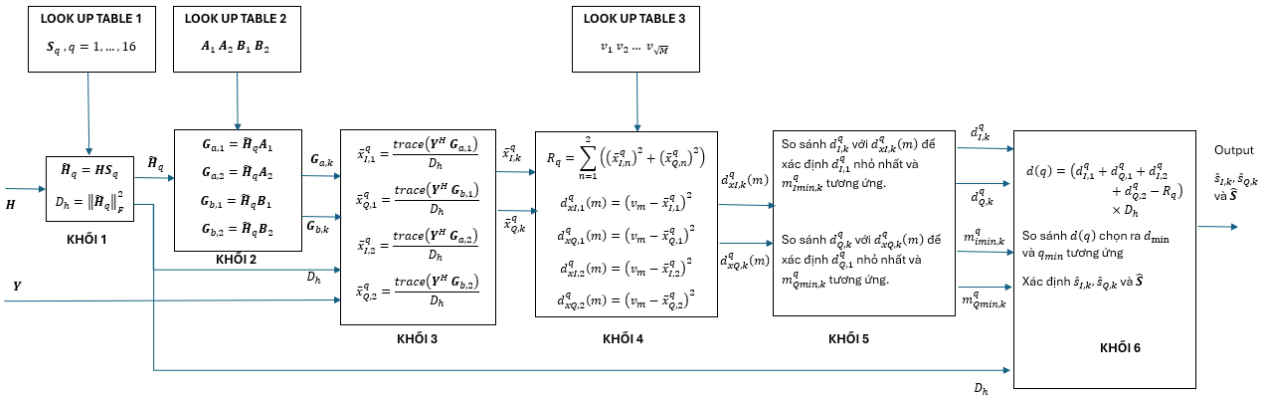
1.5 KẾT LUẬN CHƯƠNG.

Chương 1 đã trình bày cơ sở lý thuyết về hệ thống MIMO, bao gồm mô hình toán học, các loại độ lợi, và vai trò của MIMO trong cải thiện hiệu năng hệ thống. Các phương pháp mã hóa STBC, OSTBC và điều chế không gian (SM) được phân tích để làm rõ khả năng tăng độ tin cậy và hiệu suất truyền thông. Cuối cùng, trình bày xuất một sơ đồ truyền MIMO mới với hiệu suất cao, hiệu suất tốt và độ phức tạp thấp, được gọi là SM–OSTBC cùng các thuật toán giải mã tối ưu như SO-ML, SO-SD đã được đề xuất, tạo nền tảng cho các nghiên cứu và thiết kế trong các chương tiếp theo.

CHƯƠNG 2 THIẾT KẾ HỆ THỐNG.

Giới thiệu chương : Chương 2 sẽ tập trung thiết kế các sơ đồ khối cho, các khối tính toán, lựa chọn các phép tính phù hợp cho bộ bộ giải mã SO—ML cho hệ thống MIMO SM-OSTBC 4 × 4 ($n_T = 4, n_R = 4$), sử dụng điều chế 16-QAM

2.1 SƠ ĐỒ TỔNG QUAN CHỨC NĂNG HỆ THỐNG



Hình 2.1 Tổng quan sơ đồ khối chức năng hệ thống

Mô tả hệ thống :

- Đầu vào nhận ma trận kênh $\boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}$ (trong đó h_{ij} là các số phức); và ma trận tín hiệu thu $\boldsymbol{Y} = \begin{bmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \\ y_{31} & y_{32} \\ y_{41} & y_{42} \end{bmatrix}$
- Khối xử lý tính toán sẽ kết hợp với dữ liệu đầu vào vào dữ liệu được lưu trong Look-Up-Table để tính toán các giá trị cần thiết.
- Đầu ra của hệ thống gồm chuỗi 12 bit thông tin bao gồm : 8 bit thông tin tương ứng với 2 ký hiệu (x_1, x_2) trong ma trận Alamouti và 4 bit thong tin tương ứng với 16 từ mã SC.

2.2 SỬ DỤNG LOOK-UP-TABLE ĐỂ LƯU TRỮ CÁC MA TRẬN CẦN THIẾT.

Để lưu trữ các ma trận trên, em sử dụng ROM. Các giá trị được lưu dưới dạng số Fixed point number 16bit (8bit thập phân, 7 bit nguyên và 1 bit dấu). Các giá trị được lưu dưới dạng mã hex.

- Lưu trữ tập các số nguyên $v = \{-3 \quad -1 \quad 1 \quad 3\}$ dùng để tạo ra các ký hiệu điều chế 16-QAM (x_1, x_2).
- Lưu trữ các ma trận tán xạ để tạo mã Alamouti STBC kích thước 2×2:

$$\boldsymbol{A}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \boldsymbol{A}_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \quad \boldsymbol{B}_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \boldsymbol{B}_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- Lưu trữ 16 từ mã SC, mỗi từ mã là một ma trận kích thước 4×2 với các thành phần là các số phức, được định nghĩa như sau:

$$\boldsymbol{S}_1 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & 1 \\ -1 & 1 \end{bmatrix} \quad \boldsymbol{S}_2 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & j \\ j & 1 \end{bmatrix} \quad \boldsymbol{S}_3 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \quad \boldsymbol{S}_4 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ 1 & -j \\ -j & 1 \end{bmatrix}$$

$$\boldsymbol{S}_5 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix} \quad \boldsymbol{S}_6 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & j \\ j & -1 \end{bmatrix} \quad \boldsymbol{S}_7 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & -1 \\ 1 & -1 \end{bmatrix} \quad \boldsymbol{S}_8 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -1 & -j \\ -j & -1 \end{bmatrix}$$

$$\boldsymbol{S}_9 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ j & 1 \\ -1 & j \end{bmatrix} \quad \boldsymbol{S}_{10} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ j & j \\ j & j \end{bmatrix} \quad \boldsymbol{S}_{11} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ j & -1 \\ 1 & j \end{bmatrix} \quad \boldsymbol{S}_{12} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ j & -j \\ -j & j \end{bmatrix},$$

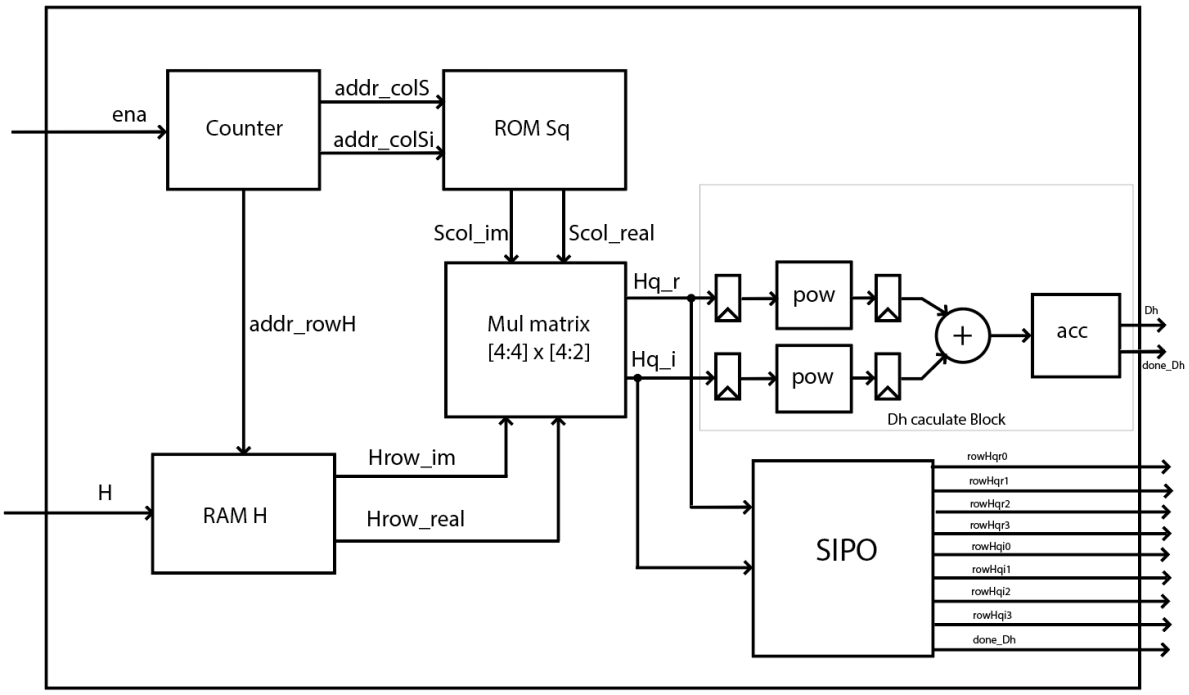
$$\boldsymbol{S}_9 = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -j & 1 \\ -1 & -j \end{bmatrix} \quad \boldsymbol{S}_{10} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -j & j \\ j & -j \end{bmatrix} \quad \boldsymbol{S}_{11} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -j & -1 \\ 1 & -j \end{bmatrix} \quad \boldsymbol{S}_{12} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & 1 \\ -j & -j \\ -j & -j \end{bmatrix}$$

- Lưu trữ các chuỗi bit dùng để ánh xạ từ bit sang các ký hiệu 16-QAM (B_v) và từ mã SC (B_s):

$$B_v = \{00 \quad 01 \quad 11 \quad 10\}$$

$$B_s = \begin{Bmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \end{Bmatrix}$$

2.3 KHỐI TÍNH TOÁN MA TRẬN HQ VÀ GIÁ TRỊ DH



Hình 2.2 Sơ đồ khối chức năng tính ma trận Hq và giá trị Dh

- **Vào ra hệ thống :**

Đầu vào : Ma trận H, tín hiệu ena, tín hiệu clk và tín hiệu rst.

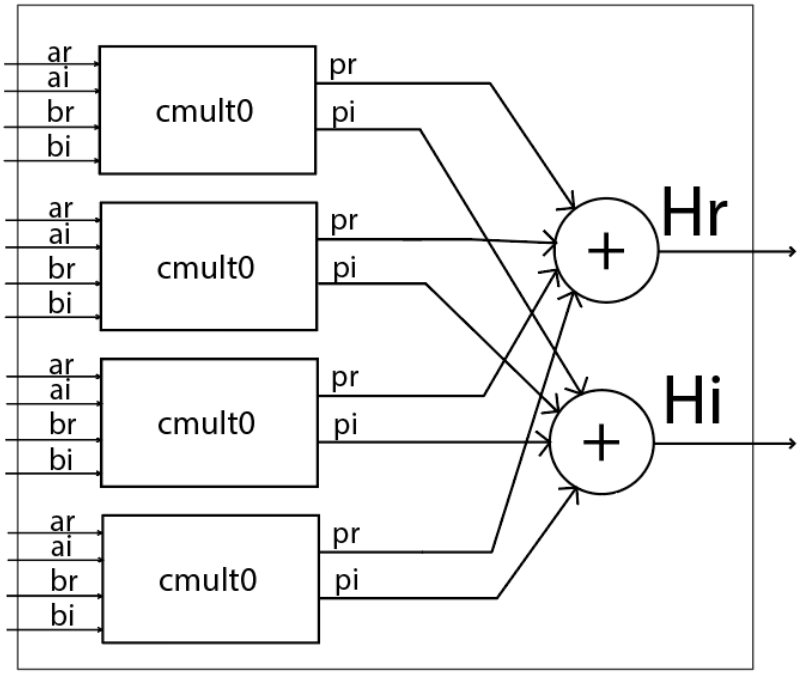
Đầu ra : Giá trị Dh, các hàng của ma trận Hq, tín hiệu done_Dh và tín hiệu done_Hq.

- **Mô tả hoạt động :**

Như tiêu đề, khối này có chức năng tính toán ma trận Hq và giá trị Dh. Ma trận Hq được tính bằng công thức Hq = H.Sq (q có giá tị từ 0 đến 15). Trước khi tính toán, ta lưu tra ma trận H theo các hàng (gồm 4 hàng) vào RAM H và các ma trận Sq theo các cột (mỗi ma trận S gồm 2 cột) vào ROM Sq. Sau đó, khi có tín hiệu ena được kích lên mức cao, bộ counter sẽ hoạt động và tạo ra các tín hiệu địa chỉ dùng để truy cập vào hàng và cột tương ứng của 2 ma trận H và S sau đó làm dữ liệu đầu vào cho khối nhân ma trận tín toán. Tại khối nhân ma trận, khi có dữ liệu đầu vào tới, nó sẽ bắt đầu tính toán và cho ra giá trị thực và ảo (Hq_r và Hq_im) đầu tiên của ma trận kết quả sau 5 xung nhịp clock. Tại giai đoạn này 2 giá trị Hq_r và Hq_im sẽ được chuyển tiếp tới khối SIPO và khối

Dh caculate. Ở khối SIPO, giá trị Hq_R và Hq_im sẽ được tích lũy cho tới khi đủ 1 ma trận kết quả Hq thì sẽ xuất ra các hàng của ma trận Hq. Còn đối với khối Dh caculate, 2 giá trị Hq_r và Hq_im sẽ lần lượt được bình phương, cộng và tới bộ cộng tích lũy, bộ cộng tích lũy sẽ cộng 8 giá trị sau đó xuất ra giá trị Dh là kết quả cuối cùng.

2.3.1 Khối nhân ma trận [4x4] với ma trận [4x2]



Hình 2.3 Sơ đồ khối nhân ma trận [4x4] với ma trận [4x2]

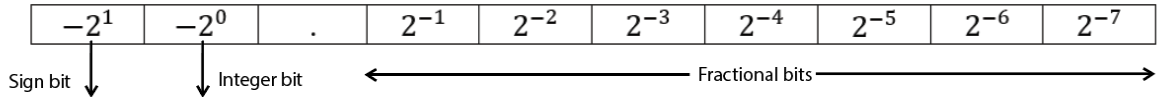
Khối nhân 3 trận [4x4] với ma trận [4x2] gồm 4 phép nhân và 2 phép cộng số phức. Khối này hoạt động theo pipeline 5 stage. Sau xung clk thứ 5 sẽ xuất ra kết quả đầu tiên đồng nghĩa với đã nhân xong 1 hàng và 1 cột. Và sau đó cứ mỗi 1 clk sẽ xuất ra 1 giá trị mới.

2.2.1.1 Biểu diễn số fixed-point

Số phẩy tĩnh là số có số bit biểu diễn phần nguyên và phần thập phân cố định. Chúng ta thường sử dụng Q notation để biểu diễn nó.

UQm.n để biểu diễn số phẩy tĩnh không dấu

Qm.n để biểu diễn số phẩy tĩnh có dấu Mỗi dạng có 2 tham số m và n trong đó : m là biểu diễn số bit bên trái và m biểu diễn số bit bên phải dấu chấm.



2.2.1.2 Phép nhân 2 số phức fixed-point theo pipeline

Thực hiện nhân 2 số phức có dạng : (Xr+jXi) và (Yr+jYi) được định nghĩa như sau :

$$Zr = XrYr - XiYi$$

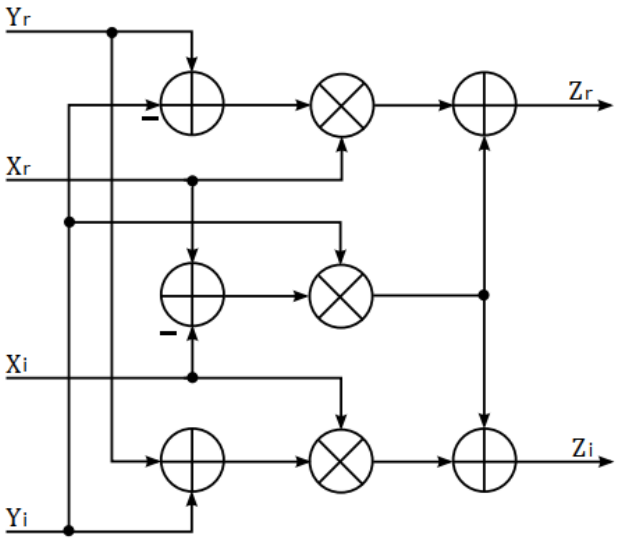
$$Zi = XrYi + YrXi$$

Trong đó Zr là phần thực của kết quả và Zi là phần ảo. Nếu thức hiện theo cách tính trực tiếp, ta sẽ cần sử dụng 4 bộ nhân và 2 bộ cộng. Tuy nhiên sẽ biến đổi biểu thức trên để chỉ cần sử dụng 3 bộ nhân như sau :

$$Zr = Xr . (Yr - Yi) + Yi . (Xr - Xi),$$

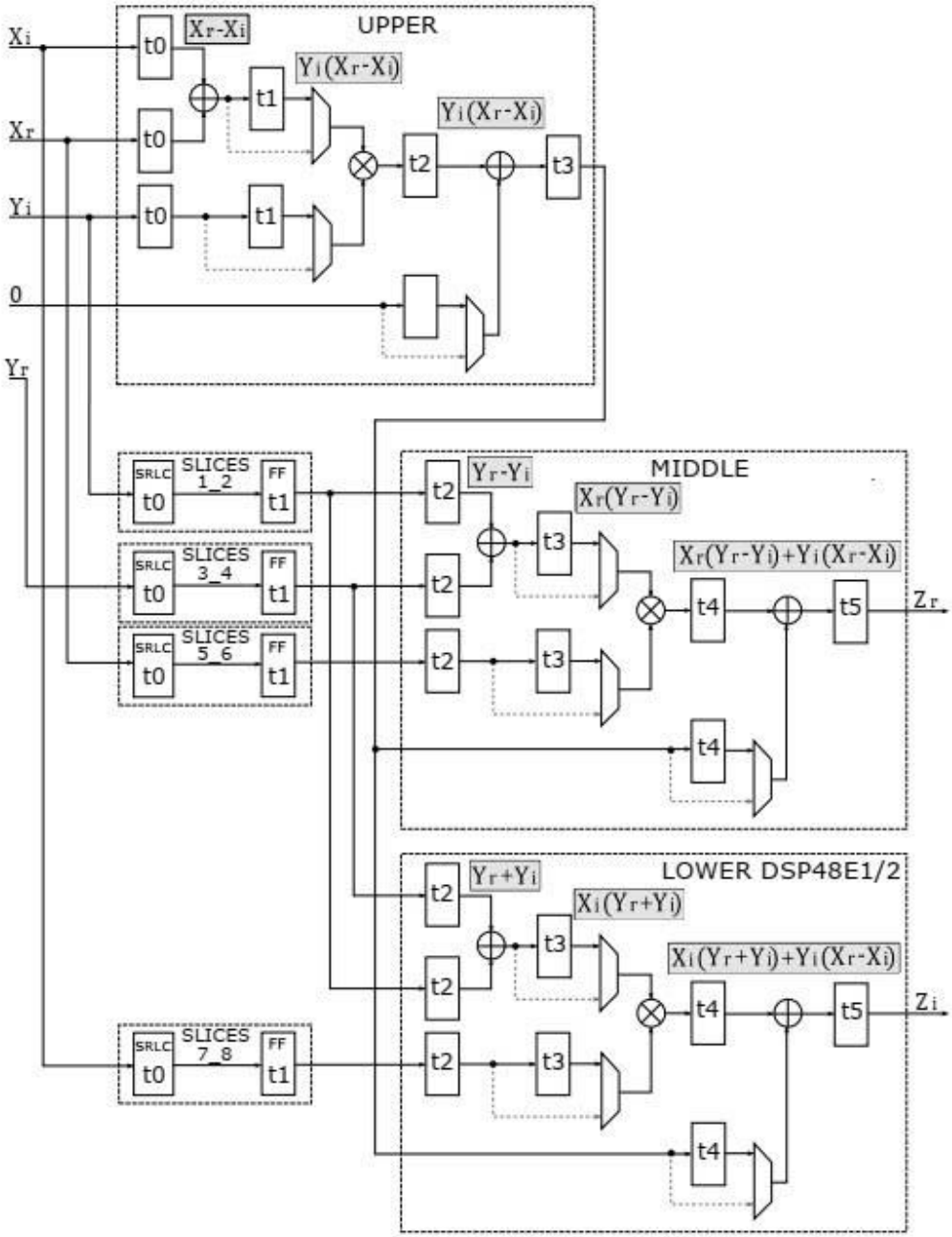
$$Zi = Xi . (Yr + Yi) + Yi . (Xr - Xi).$$

Ta có thể thấy 2 phép tính trên có cùng 1 biểu thức chung đó là Yi . (Xr - Xi). Do đó ta chỉ cần thực hiện tính 1 lần đối với biểu thức này. Như vậy ta sẽ sử dụng 3 bộ nhân và 5 bộ cộng cho nên sẽ giảm được diện tích bởi vì bộ nhân yêu cầu nhiều diện tích hơn bộ cộng



Hình 2.4 Sơ đồ khối triển khai 1 bộ nhân số phức

- **Sơ đồ khối thực hiện**



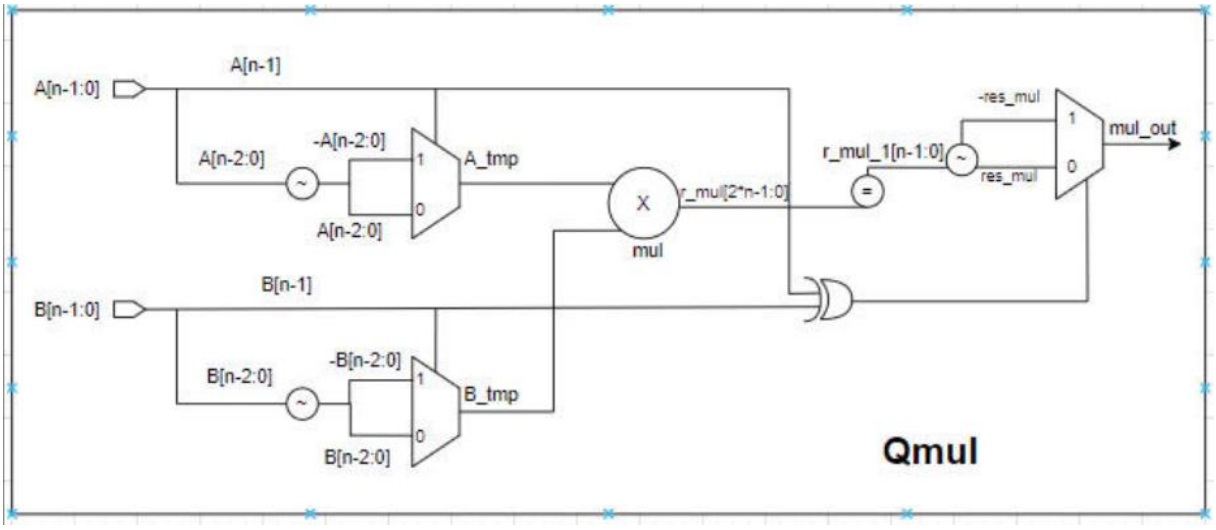
Hình 2.5 Sơ đồ khối thực hiện1 bộ nhân số phức theo pipeline

Hình trên thể hiện sơ đồ của bộ nhân số phức được đề xuất để đạt thông lượng tối đa.

- Bộ nhân số phức này cần ba khối DSP và 8 khối bổ sung để cân bằng các đường dẫn pipeline. Mất 5clk để tính toán xong được 1 giá trị đầu tiên sau đó cứ mỗi clk sẽ cho ra 1 giá trị mới.

- Khối DSP UPPER tính toán hạng tử Yi·(Xr−Xi)
- Vì hạng tử Yi·(Xr−Xi) được chia sẻ giữa phần thực và phần ảo, nên kết quả tạm thời này cần được cung cấp cho hai khối DSP khác.
- Khối DSP MIDDLE tính toán biểu thức Xr·(Yr−Yi) trong bộ cộng đầu tiên kết hợ với bộ nhân. Bộ cộng cuối cùng cộng biểu thức này với kết quả từ khối DSP UPPER, tạo ra phần thực của phép nhân số phức.
- Tương tự, trong khối DSP LOWER, hạng tử Xi·(Yr−Yi) được tính toán và sau đó được cộng với kết quả từ khối DSP UPPER, tạo ra phần ảo của phép nhân số phức.
- Để đạt được tần số xung nhịp tối đa, kết quả từ khối DSP UPPER cần được ghi lại, cả ở đầu ra của khối DSP UPPER và ở đầu vào của hai khối DSP khác. Nếu không, tần số xung nhịp tối đa có thể đạt được sẽ bị giảm do độ trễ do sự kết nối giữa các khối DSP gây ra.
- Để tính toán phép nhân số phức chính xác, các khối DSP MIDDLE và LOWER cần kết quả từ khối DSP UPPER sau ba chu kỳ xung nhịp kể từ khi dữ liệu được đưa vào các khối DSP này. Kết quả của khối DSP UPPER được chuyển đến bộ cộng thứ hai trong các khối DSP MIDDLE và LOWER sau năm chu kỳ xung nhịp kể từ khi dữ liệu đầu vào được chuyển đến khối DSP UPPER, tức là bốn chu kỳ xung nhịp từ độ trễ của khối DSP UPPER cộng thêm một chu kỳ từ thanh ghi đầu vào trong đường dẫn C đến P của các khối DSP MIDDLE và LOWER."

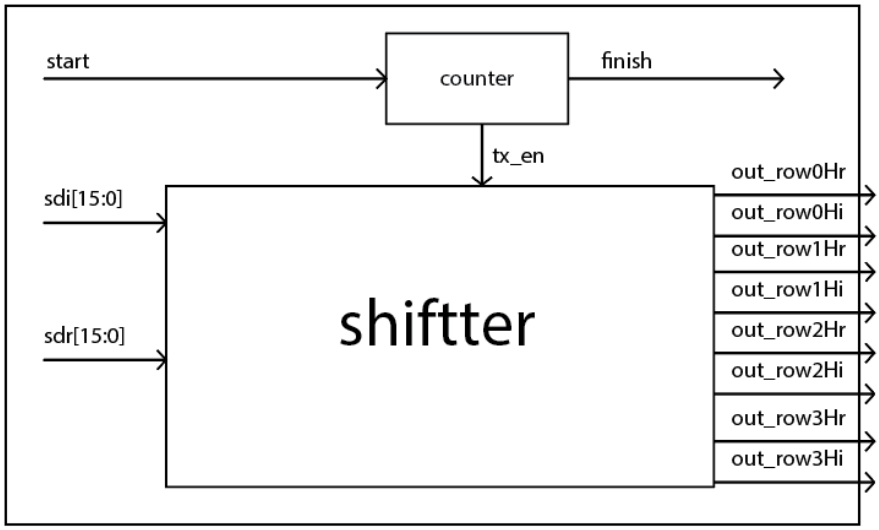
2.3.2 Khối Pow



Hình 2.6 Sơ đồ khối của khối pow

Khối pow là mạch tổ hợp có chức năng bình phương giá trị đầu vào và xuất ra giá trị tương ứng. Đầu ra của khối pow là giá trị có dạng fixed-point.

2.3.3 Khối SIPO



Hình 2.7 Sơ đồ khối của khối SIPO

- **Vào ra hệ thống :**

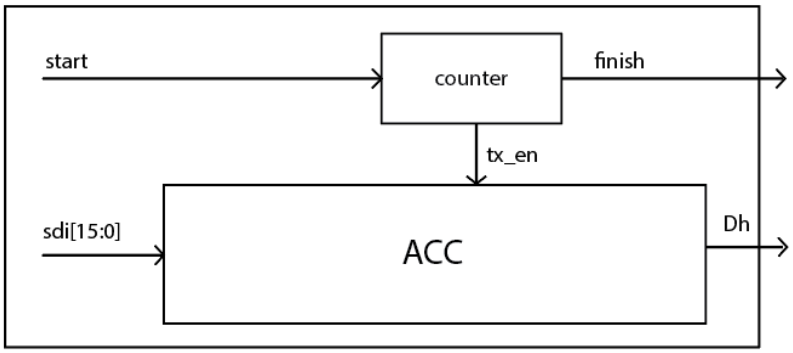
Đầu vào : Start, sdi, sdr, clk, rst

Đầu ra : Tín hiệu finish, out_row0Hr, out_row0Hi, out_row1Hr, out_row1Hi, out_row2Hr, out_row2Hi, out_row3Hr, out_row3Hi.

- **Mô tả hoạt động :**

Khi có tín hiệu start được kích lên mức cao, khi đó khối counter sẽ hoạt động và động thời tín hiệu tx_en sẽ ở mức cao và khối shifter sẽ hoạt động. Ở khối shifter đầu vào là các tín hiệu sdr, sdi và sau 8 chu kỳ dịch sẽ xuất ra các giá trị tương ứng với các hàng của ma trận Hq.

2.3.4 Khối cộng tích lũy



Hình 2.8 Sơ đồ khối của khối cộng tích lũy

- **Vào ra hệ thống :**

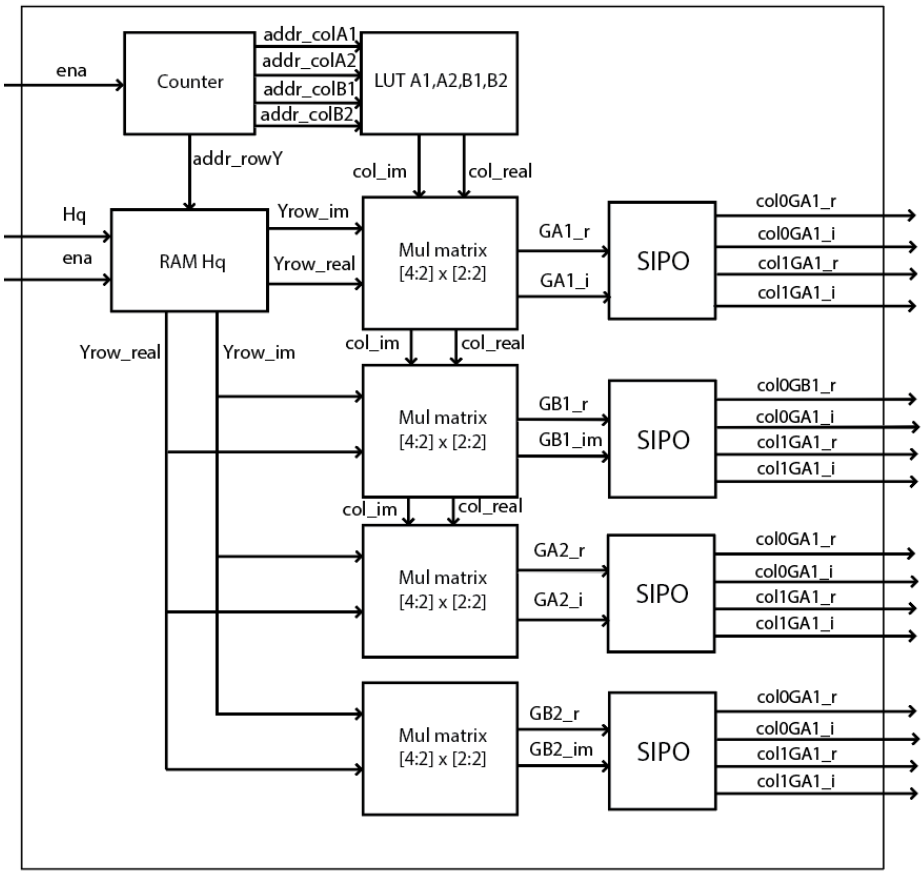
Đầu vào : Start, sdi, clk, rst

Đầu ra : Tín hiệu finish và giá trị Dh.

- **Mô tả hoạt động :**

Khi có tín hiệu start được kích lên mức cao, khi đó khối counter sẽ hoạt động và động thời tín hiệu tx_en sẽ ở mức cao và khối accsẽ hoạt động. Ở khối accđầu vào là các tín hiệu sdr, khối sdr có thanh ghi tích lũy sẽ cộng tích lũy giá trị đầu vào sau 8 chu kỳ dịch và sẽ xuất ra giá trị Dh.

2.4 KHỐI TÍNH MA TRẬN GA1, GA2, GB1, GB2



Hình 2.9 Sơ đồ khối tính ma trận Ga1, Ga2, Gb1, Gb2

- **Vào ra hệ thống**

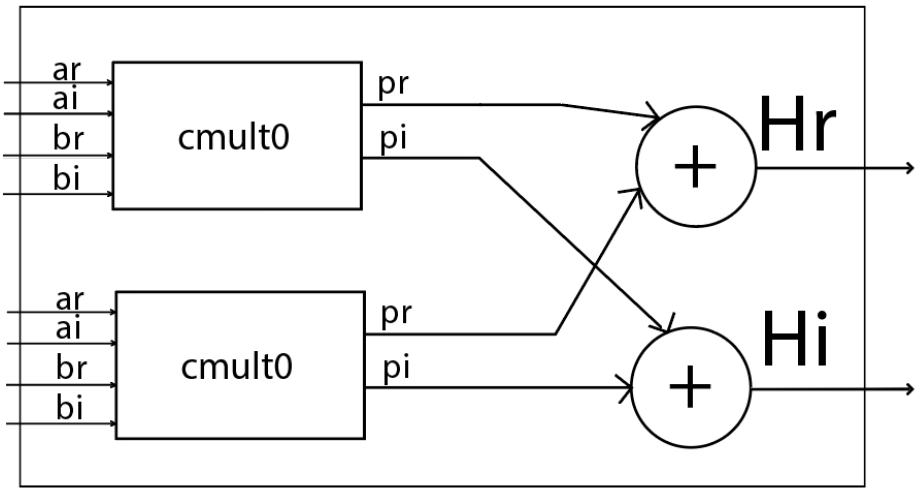
Đầu vào : tín hiệu clk, rst, ena và ma trận Hq.

Đầu ra : Các cột của ma trận GA1, GA2, GB1, GB2 và tín hiệu finish.

- **Mô tả hoạt động**

Khi có tín hiệu ena thì dữ liệu của ma trận H sẽ được nạp vào RAM Hq đồng thời khối counter cũng sẽ hoạt động và xuất ra tín hiệu hiệu địa chỉ dùng để truy cập vào LUT để lấy ra các cột của ma trận A1, A2, B1, B2 và truy cập vào RAM Hq để lấy ra các hàng của ma trận Hq. Sau đó khối nhân ma trận tính toán ra các giá trị tương ứng của ma trận kết quả. Kết quả được đưa vào khối SIPO để tích lũy và khi tính đủ giá trị sẽ xuất ra các cột của các ma trận kết quả GA1, GA2, GB1, GB2.

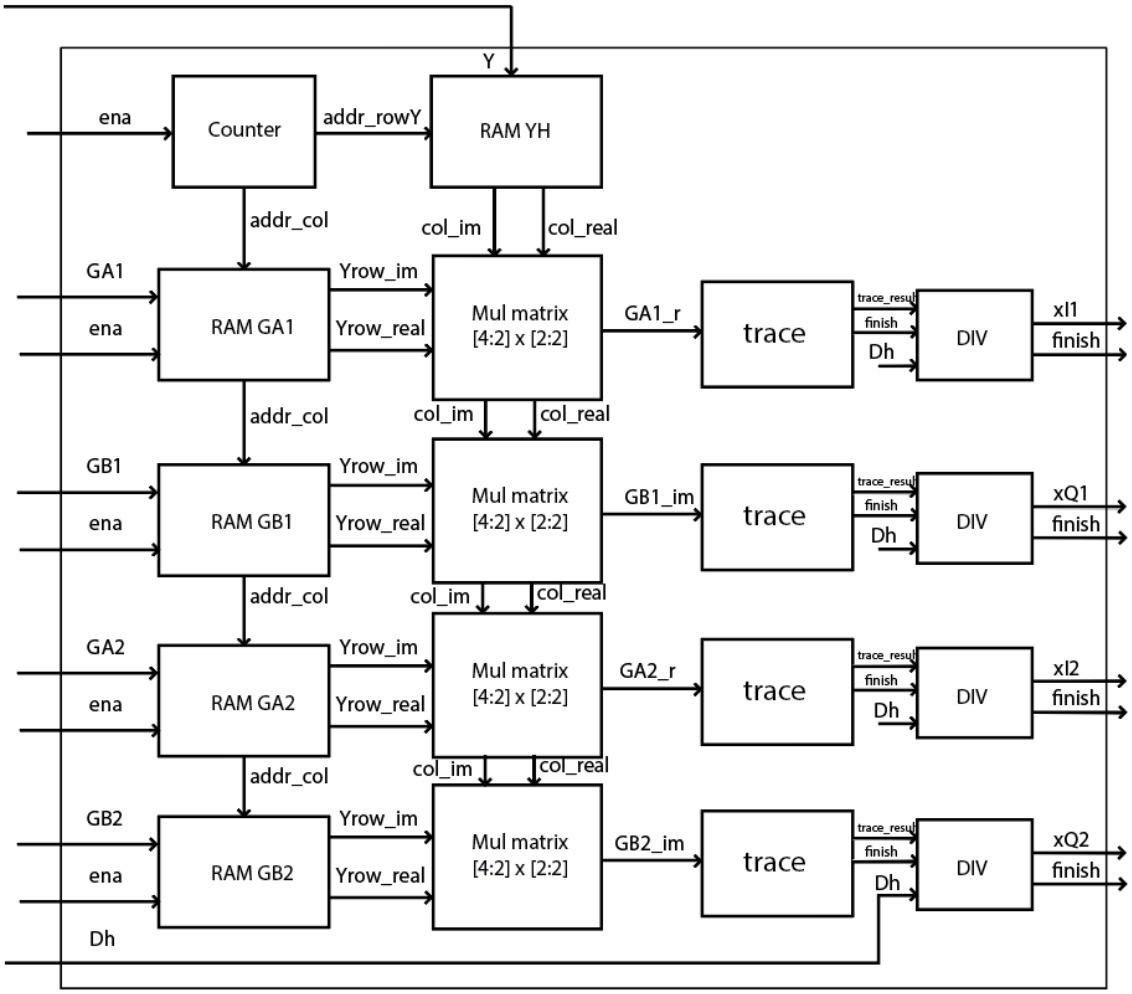
❖ ***Khối nhân ma trận [4:2] x [2:2]***



Hình 2.10 Sơ đồ khối tính ma trận [4:2] x [2:2]

Khối nhân 2 trận [4x2] với ma trận [2x2] gồm 2 phép nhân và 2 phép cộng số phức. Kết quả sẽ cho ra ma trận có kích thước [4x2]. Khối này hoạt động theo pipeline 5 stage. Sau xung clk thứ 5 sẽ xuất ra kết quả đầu tiên đồng nghĩa với đã nhân xong 1 hàng và 1 cột. Và sau đó cứ mỗi 1 clk sẽ xuất ra 1 giá trị mới.

2.5 KHỐI TÍNH GIÁ TRỊ XI, XQ



Hình 2.11 Sơ đồ khối tính ma trận Ga1, Ga2, Gb1, Gb2

- **Mô tả hệ thống**
  - Đầu vào: Tín hiệu clk, rst, ena, các hàng của ma trận GA1, GA2, GB1, GB2, và giá trị Dh.
  - Đầu ra: Giá trị xI1, xQ1, xI2, xQ2 và tín hiệu finish.
- **Quy trình hoạt động:**
  - Khi tín hiệu ena được kích hoạt (mức cao), dữ liệu từ ma trận Y, GA1, GA2, GB1, và GB2 sẽ được ghi vào bộ nhớ RAM.
  - Một bộ đếm (counter) sẽ tạo các tín hiệu địa chỉ để truy cập dữ liệu trong RAM.

- Dữ liệu được truy xuất từ RAM sẽ được truyền đến khối nhân ma trận để thực hiện các phép tính toán.
- Kết quả tính toán từ khối nhân ma trận sẽ được gửi tới khối Trace, nơi thực hiện phép tính tổng các phần tử trên đường chéo chính của ma trận.
- Kết quả của phép tính Trace sẽ được đưa vào khối Chia, nơi thực hiện phép chia giá trị Trace cho Dh.
- Cuối cùng, các giá trị xI1, xQ1, xI2, xQ2 được xuất ra. Tín hiệu finish sẽ được kích hoạt (mức cao) sau khi hoàn tất phép chia.

2.5.1 Khối trace

- **Chức năng:**

Khối Trace thực hiện phép tính tổng các phần tử trên đường chéo chính của ma trận đầu vào.

Kết quả của phép tính này là giá trị Trace, được sử dụng làm đầu vào cho khối Chia.

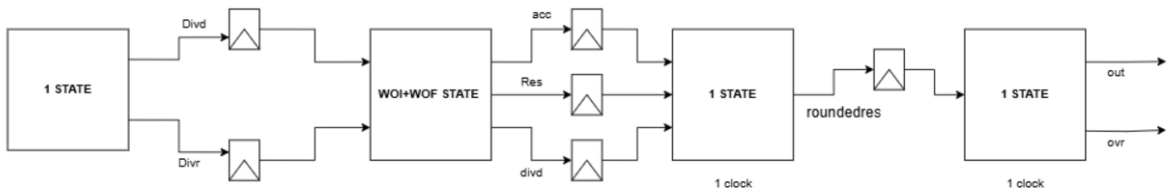
- **Quy trình hoạt động:**

Nhận dữ liệu đầu vào từ khối nhân ma trận (dữ liệu là ma trận đã được tính toán).

Xác định và trích xuất các phần tử nằm trên đường chéo chính của ma trận (các phần tử có chỉ số hàng bằng chỉ số cột).

Tính tổng các phần tử này và xuất kết quả Trace.

2.5.2 Khối chia



Hình 2.12 Sơ đồ khối chức năng tính các giá trị xI, xQ

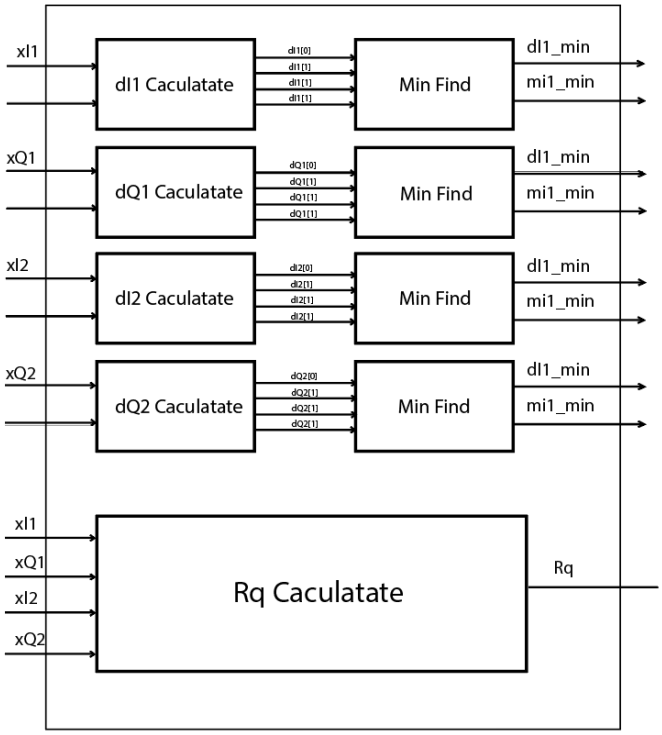
- Mô tả :

Trên là sơ đồ bộ chia 2 số fixed-point được thực hiện theo Pipeline gồm a + b + 3 stage. Trong đó a là số bit biểu diễn phần thập và b là số bit biểu diễn phần nguyên.

Tính Toán Thương bằng Phép Dịch (Division Using Shift-Add)

Chức năng: Tính toán thương (quotient) thông qua các phép dịch (shift) và cộng/trừ (add/subtract).

2.6 KHỐI TÍNH RQ, DXI, DXQ VÀ TÌM RA GIÁ TRỊ NHỎ NHẤT CỦA DI, DQ



Hình 2.13 Sơ đồ khối chức năng tính các giá trị xI, xQ

2.6.1 Khối D Caculate

Ở khối này ta sẽ thực hiện các phép tính :

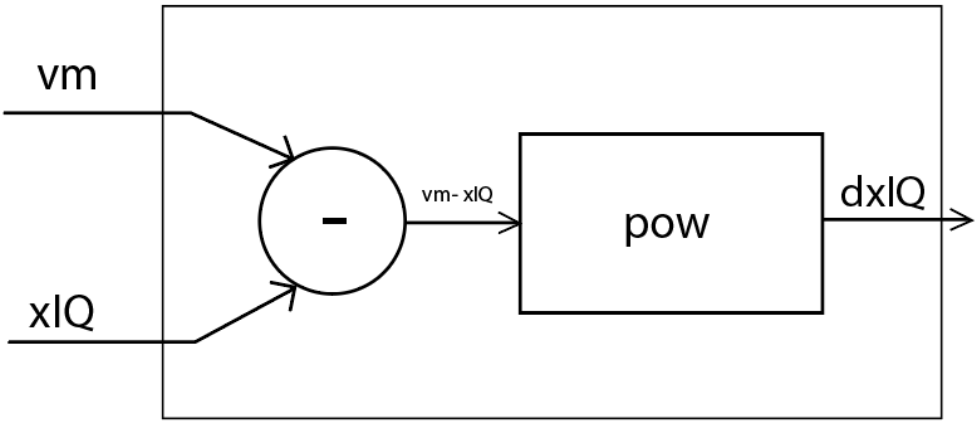
$$d_{xI,1}^{q}(m) = \left(v_m - \bar{x}_{I,1}^{q}\right)^2$$

$$d_{xQ,1}^{q}(m) = \left(v_m - \bar{x}_{Q,1}^{q}\right)^2$$

$$d_{xI,2}^{q}(m) = \left(v_m - \bar{x}_{I,2}^{q}\right)^2$$

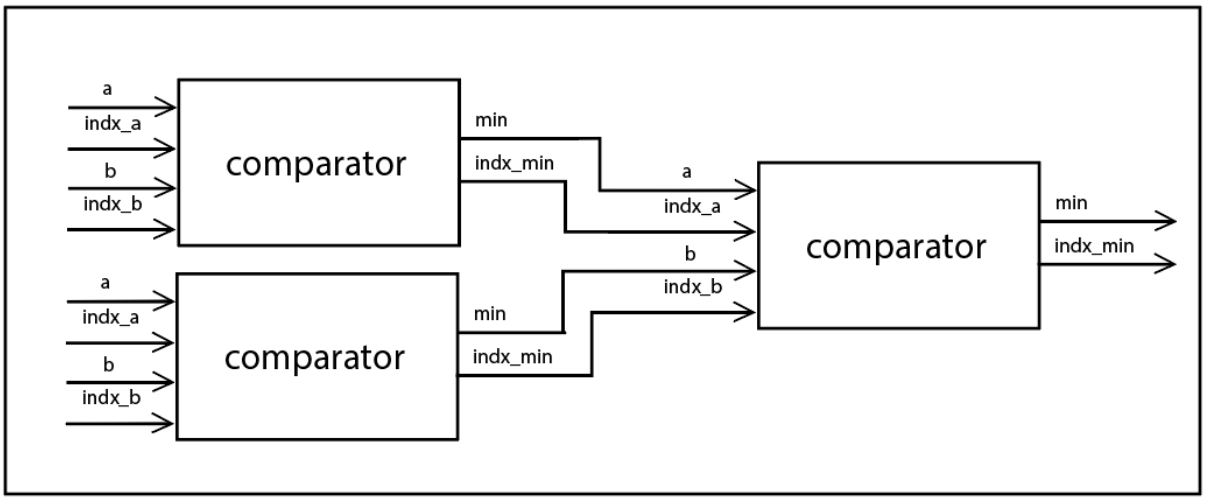
$$d_{xQ,2}^{q}(m) = \left(v_m - \bar{x}_{Q,2}^{q}\right)^2$$

(Với m có giá trị từ 1 tới 4.)



Hình 2.14 Sơ đồ khối khối D caculate

2.6.2 Khối Min Find



Hình 2.15 Sơ đồ khối khối D caculate

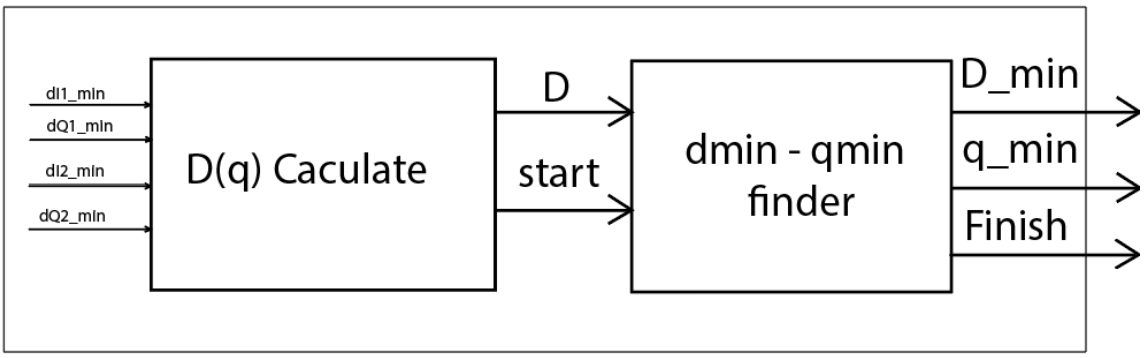
Khối Min Find nhận đầu vào là 4 giá trị kèm theo chỉ số của nó sau đó tìm ra ra giá trị bé nhất cùng chỉ số đi kèm.

2.6.3 Khối Rq Caculate

Khối Rq Caculate là mạch tổ hợp gồm 4 phép cộng 2 đầu vào với phép tính là

$$R_q = \sum_{n=1}^{2} \left(\left(\bar{x}_{I,n}^{q} \right)^2 + \left(\bar{x}_{Q,n}^{q} \right)^2 \right)$$

2.7 KHỐI TÍNH GIÁ TRỊ D(Q) VÀ TÒM GIÁ TRỊ NHỎ NHẤT CỦA D VÀ CHỈ SỐ Q TƯƠNG ỨNG



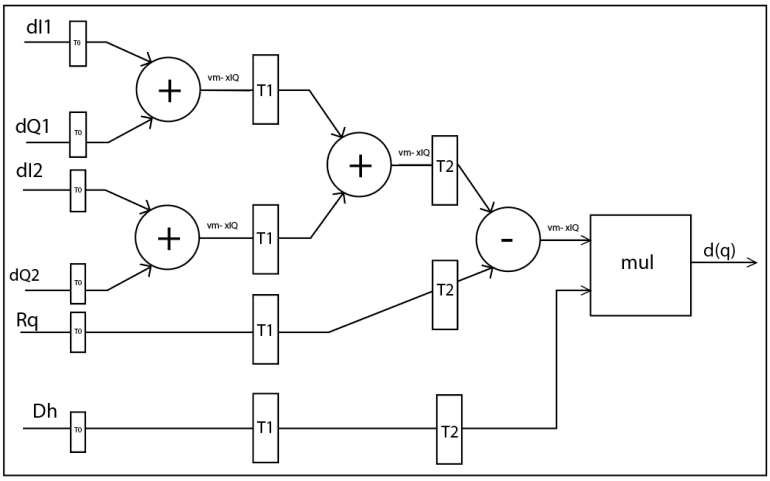
Hình 2.16 Sơ đồ khối chức năng tính giá trị d(q) và tìm giá trị nhỏ nhất của d và chỉ số q tương ứng

- **Mô tả :**
  - Đầu vào : Tín hiệu clk, rst, enable và các giá trị dI1_min, dQ1_min, dI2_min
  - Đầu ra : Giá trị D_min, chỉ số q_min và tín hiệu finish.

2.7.1 Khối D(q) Caculate

Kối D(q) Caculate có chức năng thục hiện phép tính sau :

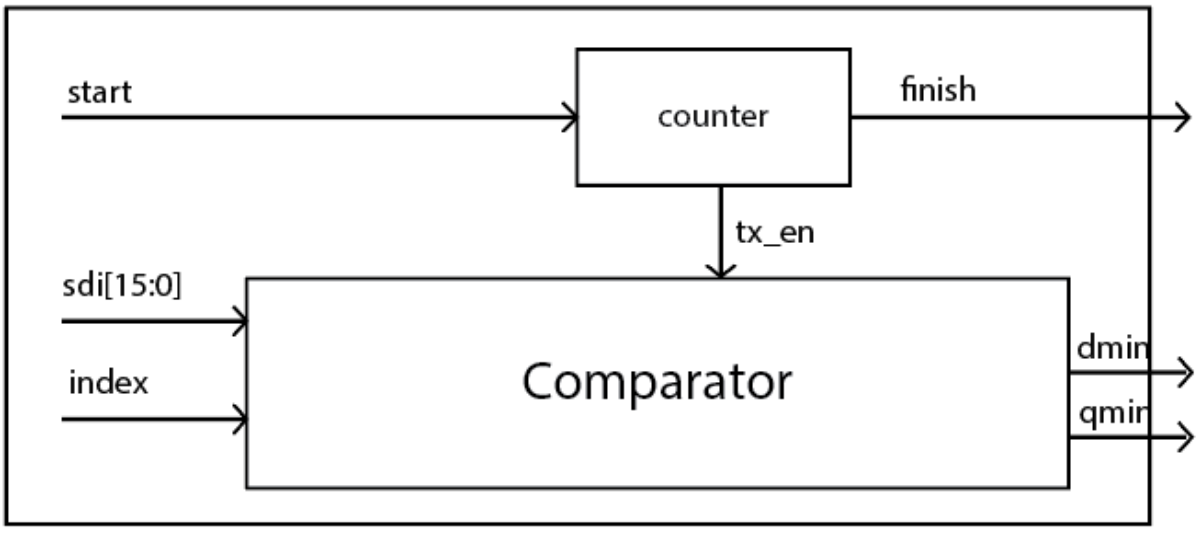
$$d(q) = \left(d_{I,1}^{q} + d_{Q,1}^{q} + d_{I,2}^{q} + d_{Q,2}^{q} - R_q \right) \times D_h$$



Hình 2.17 Sơ đồ khối chức năng tính giá trị d(q)

Phép tính trên được tính theo dạng pipeline 5 stage. Stage thứ nhất là nạp các giá trị vào thanh ghi, sau đó thức hiện tính $d_{I,1}^{q} + d_{Q,1}^{q} v$à $d_{I,2}^{q} + d_{Q,2}^{q}$ Sau đó lấy kết quả trừ cho giá trị Rq và sau đó nhân kết quả đó với Dh. Tổng cộng phép tính trên sẽ mất 8 clk để tính ra giá trị đầu tiền và sau đó cứ mỗi clk thì sẽ tính ra được 1 giá trị mới.

2.7.2 Khối tìm d min và qmin



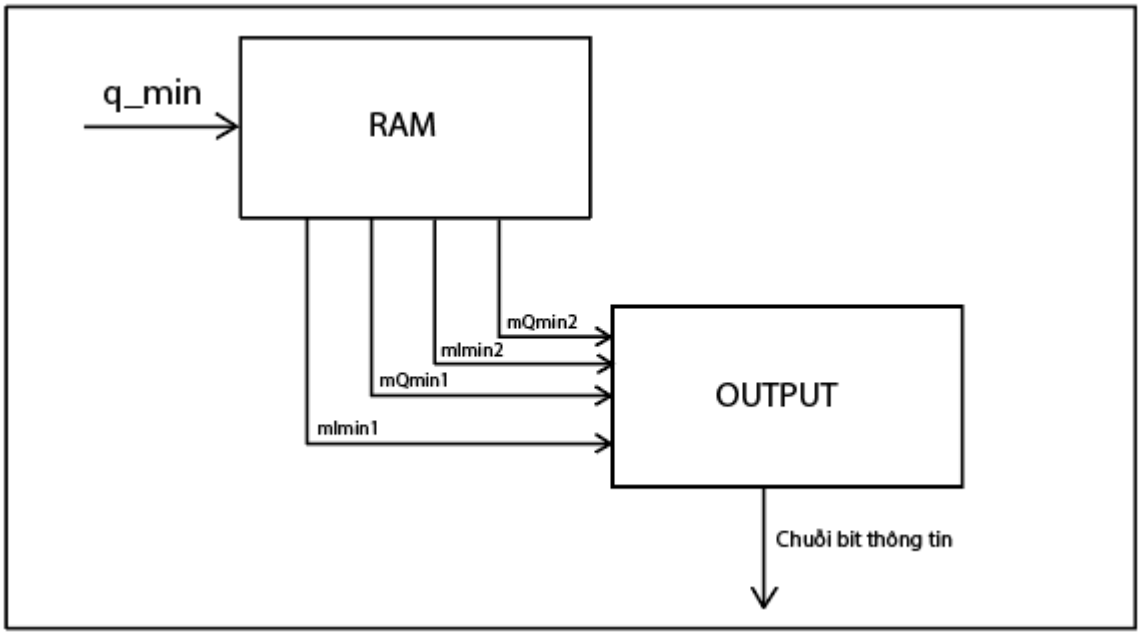
Hình 2.18 Sơ đồ khối chức năng tính giá trị d(q)

- **Mô tả hoạt động**

Khi có tín hiệu start được kích lên mức 1 thì khối counter và khối Comparator sẽ hoạt động. Cứ mỗi 1 clk khối Comparator sẽ lấy 1 giá trị mới từ input và so sánh với

giá trị hiện tại để tìm ra giá trị bé hơn. Sau 16 lần so sánh sẽ tìm ra giá trị bé nhất đó là d min và chỉ số q min tương ứng8.

2.8 KHỐI XÁC ĐỊNH TÍN HIỆU PHÁT VÀ CHUỖI BIT THÔNG TIN TƯƠNG ỨNG



Hình 2.19 Sơ đồ khối chức năng xác định output

- **Mô tả hoạt động**

Khối gồm 2 khối chính. 1 khối Ram được sử dụng để lưu các giá $m^q_{Imin,1}$, $m^q_{Qmin,1}$, $m^q_{Imin,2}$, $m^q_{Qmin,2}$(Với chạy từ 1 đến 16).

Khối output dùng để lưu các các chuỗi bit dùng để ánh xạ từ bit sang các ký hiệu 16-QAM (B_v) và từ mã SC (B_s):

$$B_v = \{00 \quad 01 \quad 11 \quad 10\}$$

$$B_s = \begin{Bmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \end{Bmatrix}$$

Từ ta xác định input như sau :

Xác định tín hiệu phát :

$$\hat{s}_{I,1} = v_{m_{Imin,1}^{q_{min}}}, \hat{s}_{Q,1} = v_{m_{Qmin,1}^{q_{min}}}, \hat{s}_{I,2} = v_{m_{Imin,2}^{q_{min}}}, \hat{s}_{Q,2} = v_{m_{Qmin,2}^{q_{min}}}$$

$$\widehat{\boldsymbol{S}} = \boldsymbol{S}_{q_{min}}$$

Xác định chuỗi thông tin tương ứng :

$$\boldsymbol{b}_1 = \left[B_v\left(m_{Imin,1}^{q_{min}}\right) \quad B_v\left(m_{Qmin,1}^{q_{min}}\right) \quad B_v\left(m_{Imin,2}^{q_{min}}\right) \quad B_v\left(m_{Qmin,2}^{q_{min}}\right)\right]$$

$$\boldsymbol{b}_2 = B_s(q_{min})$$

CHƯƠNG 3. KẾT QUẢ MÔ PHỎNG VÀ HƯỚNG NGHIÊN CỨU TIẾP THEO

Giới thiệu chương : Chương 3 sẽ tập trung vào việc mô phỏng thuật toán trên matlab và viết code Verilog mô tả và mô phỏng từng khối trên trình mô phỏng.

3.1 MÔ PHỎNG TRÊN MATLAB

3.1.1 Lập trình

Chương trình mô phỏng hiệu suất của hệ thống MIMO SM-OSTBC 4×4 với thuật toán giải mã Sphere Decoding ML (SO-ML). Điều chế 16-QAM được sử dụng để mã hóa dữ liệu, kết hợp với mã khối không gian thời gian (STBC) kiểu Alamouti. Hệ thống truyền qua kênh Rayleigh fading, và hiệu suất được đánh giá thông qua tỷ lệ lỗi bit (BER) theo tỷ số tín hiệu trên nhiễu (SNR).

```
% Program to simulate performance of SM-OSTBC in 4x4 MIMO system
% 16-QAM Modulation.
% SO-SD detector
% (C)2024 Le Minh Tuan, FEE1 - PTIT
% Date: March 2026

%*******************************************************
clear all;
% close all;
clc;

nT=4;                           % No of TxAnt
nR=4;                           % No of RxAnt
nA=4;                           % Number of active antenna
T=2;                            %No of symbol periods per block
nd=2;                           % Number of QAM symbol per Alamouti Code
block
```

```
TM=16;                          %16-QAM Modulation
ml=log2(TM);                    %Number of bits per 16-QAM symbol
SM=16;                          %Number of SC codewords
m=log2(SM);                     %Number of bits per SC codeword


SNRdB=[15];             % Signal-to-noise power ratio (in dB):
10*log10(Ps/Pn)
SNR=10.^(SNRdB/10);             % Convert to SNR (Ps/Pn)
S_SNR=sqrt(SNR);


%Total number of received signal blocks to be written in files
IterNo=[1000];


%-------Store integer values for 16-QAM modulation -----
v=[-3 -1 1 3];


%------Store bit sequence for 16-QAM symbol mapping ---
Bv=[0 0;0 1;1 1;1 0];
%-----Store dispersion matrices for Alamouti OSTBC----
A(:,:,1)=[1 0;0 1];
A(:,:,2)=[0 -1;1 0];
B(:,:,1)=[1 0;0 -1];
B(:,:,2)=[0 1;1 0];
%-----Store 16 SC codewords --------
S(:,:,1)=(1/2)*[1  1;-1  1;1  1;-1  1];
S(:,:,2)=(1/2)*[1  1;-1  1;1  1i;1i  1];
S(:,:,3)=(1/2)*[1  1;-1  1;1  -1;1  1];
S(:,:,4)=(1/2)*[1  1;-1  1;1  -1i;-1i  1];
S(:,:,5)=(1/2)*[1  1;-1  1;-1  1;-1  -1];
S(:,:,6)=(1/2)*[1  1;-1  1;-1  1i;1i  -1];
S(:,:,7)=(1/2)*[1  1;-1  1;-1  -1;1  -1];
S(:,:,8)=(1/2)*[1  1;-1  1;-1  -1i;-1i  -1];
S(:,:,9)=(1/2)*[1  1;-1  1;1i  1;-1  1i];
S(:,:,10)=(1/2)*[1  1;-1  1;1i  1i;1i  1i];
```

```
S(:,:,11)=(1/2)*[1  1;-1  1;1i  -1;1  1i];

S(:,:,12)=(1/2)*[1  1;-1  1;1i  -1i;-1i  1i];

S(:,:,13)=(1/2)*[1  1;-1  1;-1i  1;-1  -1i];

S(:,:,14)=(1/2)*[1  1;-1  1;-1i  1i;1i  -1i];

S(:,:,15)=(1/2)*[1  1;-1  1;-1i  -1;1  -1i];

S(:,:,16)=(1/2)*[1  1;-1  1;-1i  -1i;-1i  -1i];

%------Store bit sequence for SC codeword mapping ---

Bs=[0 0 0 0;0 0 0 1;0 0 1 0;0 0 1 1;0 1 0 0;0 1 0 1;0 1 1 0;0 1 1 1;...

    1 0 0 0;1 0 0 1;1 0 1 0;1 0 1 1;1 1 0 0;1 1 0 1;1 1 1 0;1 1 1 1;];

Es = 10; % Symbol energy of 16-QAM

S_SNR=S_SNR./sqrt(2*Es);  %Normalized S_SNR with symbol energy

% [SMConst,SM]=SM_Codeword_Gen_SubsetnA_ver5(nT,TM,'psk',nA,4);

%

% bin1=dec2bin(0:SM-1);

% bin2=dec2bin(0:TM-1);

txt=['b-p'];

Tx_SCcodeword=['SM_OSTBC_Tx_SC_Codewords_' int2str(nT) 'Tx' int2str(T)
'Symbol_Periods.dat'];

fidSC = fopen(Tx_SCcodeword,'w');

Tx_SCTxData=['SM_OSTBC_Tx_SC_Data_' int2str(nT) 'Tx' int2str(T)
'Symbol_Periods.dat'];

fidSCTxData = fopen(Tx_SCTxData,'w');

Tx_OSTBC=['SM_OSTBC_Tx_OSTBCs_' int2str(T) 'Tx' int2str(T)
'Symbol_Periods.dat'];

fidOSTBC = fopen(Tx_OSTBC,'w');

Tx_SMOSTBC=['SM_OSTBC_Tx_SMOSTBCs_' int2str(nT) 'Tx' int2str(T)
'Symbol_Periods.dat'];

fidSMOSTBC = fopen(Tx_SMOSTBC,'w');

Tx_OSTBCTxData=['SM_OSTBC_Tx_OSTBC_Data_' int2str(nT) 'Tx' int2str(T)
'Symbol_Periods.dat'];

fidOSTBCTxData = fopen(Tx_OSTBCTxData,'w');

Rx_Noise=['SM_OSTBC_Rx_Noise_'  int2str(SNRdB) 'dBSNR_'  int2str(nR) 'Rx'
int2str(T) 'Symbol_Periods.dat'];

fidNoise = fopen(Rx_Noise,'w');

Channel=['SM_OSTBC_RayleighChannel_' int2str(nR) 'Rx' int2str(nT)
'Tx.dat'];

fidCH = fopen(Channel,'w');
```

```
Rx_Signal=['SM_OSTBC_RxSignal_' int2str(SNRdB) 'dBSNR_' int2str(nR) 'Rx' int2str(T) 'Symbol_Periods.dat'];

fidRxSig = fopen(Rx_Signal,'w');

% Pre-generate data for simulation

Len=5000;   % Length of OSTBC blocks, channel matrix and noise

%--Generate SC codeword indices for data mapping

Sid=randi(SM,1,Len);

%--Generate noise matrices, each element has zero min and unit variance ----

Noise=(randn(nR,T,Len)+1i*randn(nR,T,Len))./sqrt(2);

%--Generate Rayleigh fading channel matrices, each element has zero min and unit variance ----

H=(randn(nR,nT,Len)+1i*randn(nR,nT,Len))/sqrt(2);

%--Generate data indices for real part of nd 16-QAM symbols

I_data=randi(sqrt(TM), nd,Len);

%--Generate data indices for real part of nd 16-QAM symbols

Q_data=randi(sqrt(TM), nd,Len);

Sindx=[1:SM];

% NTMConst=repmat(TMConst,nR,SM);

Index2=[1:4:2*T*T*SM;4:4:2*T*T*SM];

for n=1:length(SNR)

    ne1=0;

    ne2=0;

    ns=0;

    Smg=0;

    Nmg=0;

    for i=1:IterNo(n)

        %Check if more data need to be generated for simulation

        nn=mod(i,Len)+1;

        if nn==Len %More data need to be generated

            Sid=randi(SM,1,Len);

            Noise=(randn(nR,T,Len)+j*randn(nR,T,Len))./sqrt(2);

            H=(randn(nR,nT,Len)+j*randn(nR,nT,Len))/sqrt(2);
```

```
            %--Generate data indices for real part of nd 16-QAM symbols
            I_data=randi(sqrt(TM), nd,Len);

            %--Generate data indices for real part of nd 16-QAM symbols
            Q_data=randi(sqrt(TM), nd,Len);
        end
        %Select SC codeword to be transmitted
        Sc=Sid(nn);
        St=S(:,:,Sid(nn));
        %Print the real and imaginary part of SC codewords
        fprintf(fidSC,'%3.6f\t%3.6f\n',real(St));
        fprintf(fidSC,'%3.6f\t%3.6f\n',imag(St));
        %Print SC codewords index and Tx Data
        fprintf(fidSCTxData,'%d\t%d\t%d\t%d\t%d\n',Sc,Bs(Sc,:));
        %Select the real and imaginary part of 16-QAM symbols
        xI=v(I_data(:,nn));
        xQ=v(Q_data(:,nn));
        %Print 16-QAM index and Tx Data
        fprintf(fidOSTBCTxData,'%d\t%d\t%d\t%d\n',[I_data(1,nn)
Bv(I_data(1,nn),:)]);
        fprintf(fidOSTBCTxData,'\n%d\t%d\t%d\t%d\n',[I_data(2,nn)
Bv(I_data(2,nn),:)]);
        fprintf(fidOSTBCTxData,'\n%d\t%d\t%d\t%d\n',[Q_data(1,nn)
Bv(Q_data(1,nn),:)]);
        fprintf(fidOSTBCTxData,'\n%d\t%d\t%d\t%d\n',[Q_data(2,nn)
Bv(Q_data(2,nn),:)]);
        fprintf(fidOSTBCTxData,'\n');
        %Generate OSTBC matrix using dispersion matrices

X=A(:,:,1)*xI(1)+1i*B(:,:,1)*xQ(1)+A(:,:,2)*xI(2)+1i*B(:,:,2)*xQ(2);

        %Print the real and imaginary part of OSTBC
        fprintf(fidOSTBC,'%3.6f\t%3.6f\n',real(X));
        fprintf(fidOSTBC,'%3.6f\t%3.6f\n',imag(X));

        %Compute noise with given SNR
```

```
        Z=Noise(:,:,nn)./S_SNR(n);


        %Print the real and imaginary part of Noise
        fprintf(fidNoise,'%3.6f\t%3.6f\n',real(Z));
        fprintf(fidNoise,'%3.6f\t%3.6f\n',imag(Z));


        % Generate received signal at the receiver
        C=St*X;
        %Y=H(:,:,nn)*C+Z;
        % Ma tran H
        H = [
      -0.1365 + 0.7543i,  1.5116 - 0.0461i,  0.3007 + 0.2327i,  1.0309 -
1.1578i;
      -0.2736 - 0.7099i,  0.7132 + 0.2463i,  0.5120 - 0.2975i,  1.0679 +
0.2900i;
      -0.3907 - 0.8371i,  0.0189 - 1.3633i,  0.7558 + 0.5909i, -0.7756 +
0.2938i;
       0.8572 + 0.0470i, -0.3225 - 0.0850i,  0.6719 - 0.2820i, -0.9127 +
0.0101i
        ];
    % Ma tran Y
        Y = [
      -1.3256 - 1.1671i,  0.6867 + 3.1935i;
       0.5562 + 0.1468i,  0.8196 + 0.3086i;
      -1.3336 + 4.4254i,  0.6308 - 0.3941i;
       0.2515 + 0.6614i,  0.9831 + 0.3040i
    ];
        %Print the real and imaginary part of SM-OSTBC
        fprintf(fidSMOSTBC,'%3.6f\t%3.6f\n',real(C));
        fprintf(fidSMOSTBC,'%3.6f\t%3.6f\n',imag(C));


        %Print the real and imaginary part of Rayleight Channel
        fprintf(fidCH,'%3.6f\t%3.6f\n',real(H));
        fprintf(fidCH,'%3.6f\t%3.6f\n',imag(H));
```

```
%Print the real and imaginary part of Rx Signal
fprintf(fidRxSig,'%3.6f\t%3.6f\n',real(Y));
fprintf(fidRxSig,'%3.6f\t%3.6f\n',imag(Y));

%Calculate transmit power and noise power
Smg=Smg+trace(C'*C);
Nmg=Nmg+trace(Z'*Z);
%-------Signal Detection using SO-ML detector--------
%Compute equivalent channels for all 16 SC codewords
Hq=(H*S(:,1:T*SM)); %4x2
%Compute Dh for all 16 SC codewords
Hn=(sum(Hq.*conj(Hq),1));
Dh=(sum(reshape(Hn,T,SM)));

%Compute xI_q, xQ_q for all 16 SC codewords
YH=Y'*Hq;
AYH(:,:,1)=A(:,:,1)*YH;
AYH(:,:,2)=A(:,:,2)*YH;
BYH(:,:,1)=B(:,:,1)*YH;
BYH(:,:,2)=B(:,:,2)*YH;
%Compute the trace of real(A*Y'*Hq)
xI_q=sum(real(AYH(Index2)));
%Divide by Dh to get xI_q
xI_q=reshape(xI_q,SM,2)'./Dh(ones(nd,1),:);
%Compute the trace of imag(B*Y'*HH)
xQ_q=sum(imag(BYH(Index2)));
%Divide by Dh to get xI_q
xQ_q=-reshape(xQ_q,SM,2)'./Dh(ones(nd,1),:);
%Compute Rq for all 16 SC codewords
Rq=sum(xI_q.*xI_q)+sum(xQ_q.*xQ_q);
%Detect 16-QAM symbols corresponding to each SC codewords
for k1=1:nd
    Ttemp=xI_q(k1,:);
```

```
            dI_q=v(ones(SM,1),:).'-Ttemp(ones(sqrt(TM),1),:);
            dI_q=dI_q.*dI_q;
            [dtemp2(:,2*k1-1) mIQ(:,2*k1-1)]=min(dI_q);
            Ttemp=xQ_q(k1,:);
            dQ_q=v(ones(SM,1),:).'-Ttemp(ones(sqrt(TM),1),:);
            dQ_q=dQ_q.*dQ_q;
            [dtemp2(:,2*k1) mIQ(:,2*k1)]=min(dQ_q);
        end
        % ---Compute d_q ---
        d_q=(sum(dtemp2,2)-Rq').*Dh';
        %---find d_q min and the corresponding index
        [dmin qmin]=min(d_q);
        %--Determine transmitted 16-QAM symbols
        m_min=mIQ(qmin,:)';
        %Compute number of bit errors for SC codewords
        ne1=ne1+length(find(Bs(qmin,:)~=Bs(Sc,:)));
        %Compute number of bit errors for 16-QAM symbols
        ne2=ne2+length(find([Bv(m_min(1:2:3),:)
Bv(m_min(2:2:4),:)]~=[Bv(I_data(:,nn),:) Bv(Q_data(:,nn),:)]));%42s
    end
        %----Compute BER ----
     ber(n)=(sum(ne1)+sum(ne2))/(m+2*ml)/IterNo(n);
     %Compute SNR at each receive antenna of the receiver
     Smg=Smg/IterNo(n);
     Nmg=Nmg/IterNo(n)/nR;
     10*log10(Smg/Nmg)
end
% Plot BER vs SNR (in dB) in logarithm scale
fclose('all');
semilogy(SNRdB,ber,txt)
hold on
xlabel('SNR (dB)')
ylabel('BER')
legend('BER')
```

3.2 MÔ PHỎNG TRÊN TRÌNH MÔ PHỎNG HDL

Trong quá trình mô phỏng, chúng em sử dụng Iverilog để biên dịch và GTKWave để quan sát wave form.

Từ chương trình matlab ở trên, chúng em chọn những test case đầu vào như sau :

$$\boldsymbol{H} = \begin{bmatrix} -0.1365 + 0.7543i & 1.5116 - 0.0461i & 0.3007 + 0.2327i & 1.0309 - 1.1578i \\ -0.2736 - 0.7099i & 0.7132 + 0.2463i & 0.5120 - 0.2975i & 1.0679 + 0.2900i \\ -0.3907 - 0.8371i & 0.0189 - 1.3633i & 0.7558 + 0.5909i & -0.7756 + 0.2938i \\ 0.8572 + 0.0470i & -0.3225 - 0.0850i & 0.6719 - 0.2820i & -0.9127 + 0.0101i \end{bmatrix}$$

$$\boldsymbol{Y} = \begin{bmatrix} -1.3256 - 1.1671i & 0.6867 + 3.1935i \\ 0.5562 + 0.1468i & 0.8196 + 0.3086i \\ -1.3336 + 4.4254i & 0.6308 - 0.3941i \\ 0.2515 + 0.6614i & 0.9831 + 0.3040i \end{bmatrix}$$

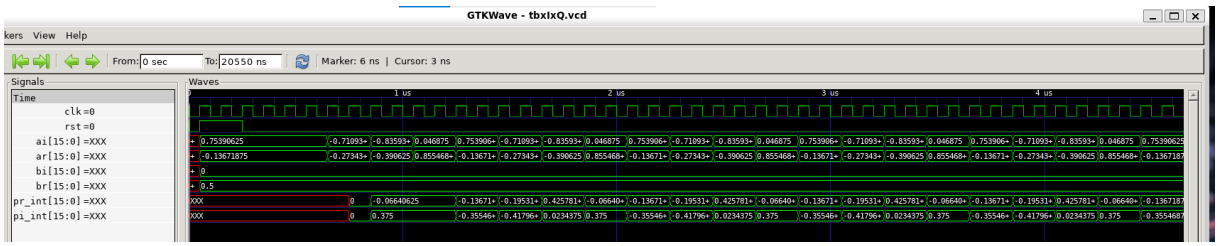
3.2.1 Xây dựng và mô phỏng khối nhân 2 số phức theo pipeline

```
module cmult # (
    parameter Q = 8,
    parameter N = 16
)
(
    input clk,
    input rst,
    input signed  [N-1:0] ar, ai,
    input signed  [N-1:0] br, bi,
    output signed [N-1:0] pr, pi
);

reg signed [N-1:0] ar_d, ar_dd, ar_ddd, ar_dddd;
reg signed [N-1:0] ai_d, ai_dd, ai_ddd, ai_dddd;
reg signed [N-1:0] br_d, br_dd, br_ddd, bi_d, bi_dd, bi_ddd;
reg signed [N-1:0] addcommon ;
reg signed [N-1:0] addr, addi ;
reg signed [N-1:0] mult0, multr, multi;
reg signed  [N-1:0] common, commonr1, commonr2;
reg signed  [N-1:0] pr_int, pi_int;

wire signed [N-1:0] tmp_mult0, tmp_multr, tmp_multi;
wire ovr_mult0, ovr_multr, ovr_multi; // Overflow flags for multipliers
// Pipeline stage 1: Input register
always @(posedge clk) begin
        if(rst) begin
                ar_d <= 0;
                ai_d <= 0;
                br_d <= 0;
                bi_d <= 0;
```

```
        end
        else begin
                ar_d <= ar;
                ar_dd <= ar_d;
                ai_d <= ai;
                ai_dd <= ai_d;
                br_d <= br;
                br_dd <= br_d;
                br_ddd <= br_dd;
                bi_d <= bi;
                bi_dd <= bi_d;
                bi_ddd <= bi_dd;
        end
end

// Pipeline stage 2: Common factor multiplication
always @(posedge clk) begin
        addcommon <= ar_d - ai_d;
        mult0 <= tmp_mult0;//addcommon * bi_dd;
        common <= mult0;
end

qmult #(.Q(Q), .N(N)) qmult_common (
    .i_multiplicand(addcommon),
    .i_multiplier(bi_dd),
    .o_result(tmp_mult0),
    .ovr(ovr_mult0)
);
/*
always @(posedge clk) begin
    common <= mult0;
end
*/
// Pipeline stage 3: Real product calculation
always @(posedge clk) begin
        ar_ddd <= ar_dd;
        ar_dddd <= ar_ddd;
        addr <= br_ddd - bi_ddd;
        multr <=  tmp_multr;//addr * ar_dddd;
        commonr1 <= common;
        pr_int <= multr + commonr1;
end

/*
always @(posedge clk) begin
        addr <= br_ddd - bi_ddd;

end
*/
qmult #(.Q(Q), .N(N)) qmult_real (
    .i_multiplicand(addr),
    .i_multiplier(ar_dddd),
    .o_result(tmp_multr),
    .ovr(ovr_multr)
);
/*
always @(posedge clk) begin
    commonr1 <= common;
```

```
    pr_int <= multr + commonr1;
end
*/
// Pipeline stage 4: Imaginary product calculation

always @(posedge clk)
begin
        ai_ddd <= ai_dd;
        ai_dddd <= ai_ddd;
        addi <= br_ddd + bi_ddd;
        multi <= tmp_multi;//addi * ai_dddd;
        commonr2 <= common;
        pi_int <= multi + commonr2;
end
/*
always @(posedge clk) begin
        addi <= br_ddd + bi_ddd;
end
*/
qmult #(.Q(Q), .N(N)) qmult_imag (
    .i_multiplicand(addi),
    .i_multiplier(ai_dddd),
    .o_result(tmp_multi),
    .ovr(ovr_multi)
);
assign pr = pr_int;
assign pi = pi_int;

endmodule // cmult
```

➢ Kết quả mô phỏng



Hình 3.1 Kết quả mô phỏng khối nhân

➢ Nhận xét

Quan sát trên waveform ta có thấy trong 5 clk đầu tiên giá trị đầu vào được nạp vào trong thanh ghi, sau khi đến clk thứ 6 kết quả của phép nhân được tính xong và cứ sau đó mỗi 1 clk thì lại cho ra 1 kết quả mới. Như vậy, phép nhân khá chính xác.

3.2.2 Xây dựng và mô phỏng khối chia

```
module fxp_div_pipe #(
    parameter WIIA = 8,
    parameter WIFA = 8,
    parameter WIIB = 8,
    parameter WIFB = 8,
    parameter WOI  = 8,
    parameter WOF  = 8,
    parameter ROUND= 1
)(
    input  wire                 rstn,
    input  wire                 clk,
    input  wire [WIIA+WIFA-1:0] dividend,
    input  wire [WIIB+WIFB-1:0] divisor,
    output reg  [WOI +WOF -1:0] out,
    output reg                  overflow
);
initial {out, overflow} = 0;
localparam WRI = WOI+WIIB > WIIA ? WOI+WIIB : WIIA;
localparam WRF = WOF+WIFB > WIFA ? WOF+WIFB : WIFA;
wire [WRI+WRF-1:0]  divd, divr;
reg  [WOI+WOF-1:0] roundedres = 0;
reg                rsign = 1'b0;
reg                 sign [WOI+WOF:0];
reg  [WRI+WRF-1:0]  acc  [WOI+WOF:0];
reg  [WRI+WRF-1:0] divdp [WOI+WOF:0];
reg  [WRI+WRF-1:0] divrp [WOI+WOF:0];
reg  [WOI+WOF-1:0]  res  [WOI+WOF:0];
localparam  [WOI+WOF-1:0] ONEO = 1;

integer ii;

// initialize all regs (khoi tao cac reg banwgf 0)
```

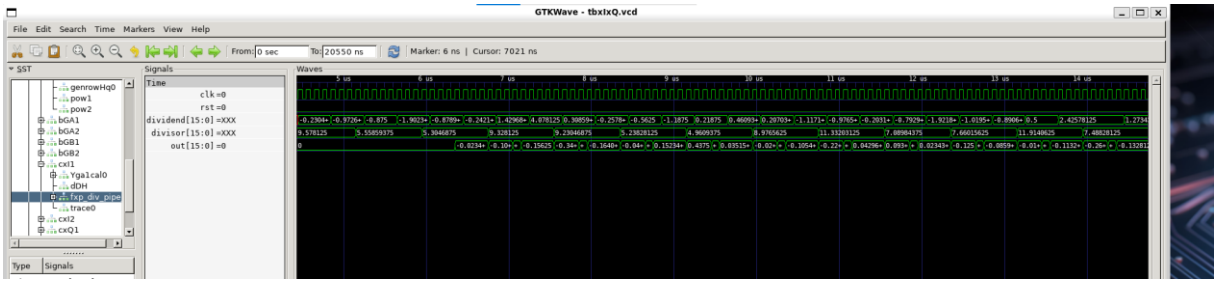
```
initial for(ii=0; ii<=WOI+WOF; ii=ii+1) begin
            res  [ii] = 0;
            divrp[ii] = 0;
            divdp[ii] = 0;
            acc  [ii] = 0;
            sign [ii] = 1'b0;
        end


wire [WIIA+WIFA-1:0] ONEA = 1;
wire [WIIB+WIFB-1:0] ONEB = 1;


// convert dividend and divisor to positive number (chuyen 2 gias trij
veef gias trij duongw)
wire [WIIA+WIFA-1:0] udividend = dividend[WIIA+WIFA-1] ? (~dividend)+ONEA
: dividend;
wire [WIIB+WIFB-1:0]  udivisor =  divisor[WIIB+WIFB-1] ? (~ divisor)+ONEB
: divisor ;


fxp_zoom # (
    .WII      ( WIIA      ),
    .WIF      ( WIFA      ),
    .WOI      ( WRI       ),
    .WOF      ( WRF       ),
    .ROUND    ( 0         )
) dividend_zoom (
    .in       ( udividend ),
    .out      ( divd      ),
    .overflow (           )
);


fxp_zoom # (
    .WII      ( WIIB      ),
    .WIF      ( WIFB      ),
    .WOI      ( WRI       ),
    .WOF      ( WRF       ),
```

```
    .ROUND    ( 0          )
)  divisor_zoom (
    .in       ( udivisor  ),
    .out      ( divr      ),
    .overflow (           )
);

// 1st pipeline stage: convert dividend and divisor to positive number
always @ (posedge clk or negedge rstn)
    if(~rstn) begin
        res[0]   <= 0;
        acc[0]   <= 0;
        divdp[0] <= 0;
        divrp[0] <= 0;
        sign [0] <= 1'b0;
    end else begin
        res[0]   <= 0;
        acc[0]   <= 0;
        divdp[0] <= divd;
        divrp[0] <= divr;
        sign [0] <= dividend[WIIA+WIFA-1] ^ divisor[WIIB+WIFB-1];
    end

reg [WRI+ WRF-1:0] tmp;

// from 2nd to WOI+WOF+1 pipeline stages: calculate division
always @ (posedge clk or negedge rstn)
    if(~rstn) begin
        for(ii=0; ii<WOI+WOF; ii=ii+1) begin
            res  [ii+1] <= 0;
            divrp[ii+1] <= 0;
            divdp[ii+1] <= 0;
            acc  [ii+1] <= 0;
            sign [ii+1] <= 1'b0;
```

```
        end
    end else begin
        for(ii=0; ii<WOI+WOF; ii=ii+1) begin

            res  [ii+1] <= res[ii];
            divdp[ii+1] <= divdp[ii];
            divrp[ii+1] <= divrp[ii];
            sign [ii+1] <= sign [ii];
            if(ii<WOI)
                tmp = acc[ii] + (divrp[ii]<<(WOI-1-ii));
            else
                tmp = acc[ii] + (divrp[ii]>>(1+ii-WOI));
            if( tmp < divdp[ii] ) begin
                acc[ii+1] <= tmp;
                res[ii+1][WOF+WOI-1-ii] <= 1'b1;
            end else begin
                acc[ii+1] <= acc[ii];
                res[ii+1][WOF+WOI-1-ii] <= 1'b0;
            end
        end
    end


// next pipeline stage: process round
always @ (posedge clk or negedge rstn)
    if(~rstn) begin
        roundedres <= 0;
        rsign      <= 1'b0;
    end else begin
        if( ROUND && ~(&res[WOI+WOF]) &&
(acc[WOI+WOF]+(divrp[WOI+WOF]>>(WOF))-divdp[WOI+WOF]) < (divdp[WOI+WOF]-
acc[WOI+WOF]) )
            roundedres <= res[WOI+WOF] + ONEO;
        else
            roundedres <= res[WOI+WOF];
```

```
        rsign      <= sign[WOI+WOF];
    end

// the last pipeline stage: process roof and output
always @ (posedge clk or negedge rstn)
    if(~rstn) begin
        overflow <= 1'b0;
        out <= 0;
    end else begin
        overflow <= 1'b0;
        if(rsign) begin
            if(roundedres[WOI+WOF-1]) begin
                if(|roundedres[WOI+WOF-2:0]) overflow <= 1'b1;
                out[WOI+WOF-1] <= 1'b1;
                out[WOI+WOF-2:0] <= 0;
            end else
                out <= (~roundedres) + ONEO;
        end else begin
            if(roundedres[WOI+WOF-1]) begin
                overflow <= 1'b1;
                out[WOI+WOF-1] <= 1'b0;
                out[WOI+WOF-2:0] <= {(WOI+WOF){1'b1}};
            end else
                out <= roundedres;
        end
    end

endmodule
```

- Kết quả mô phỏng

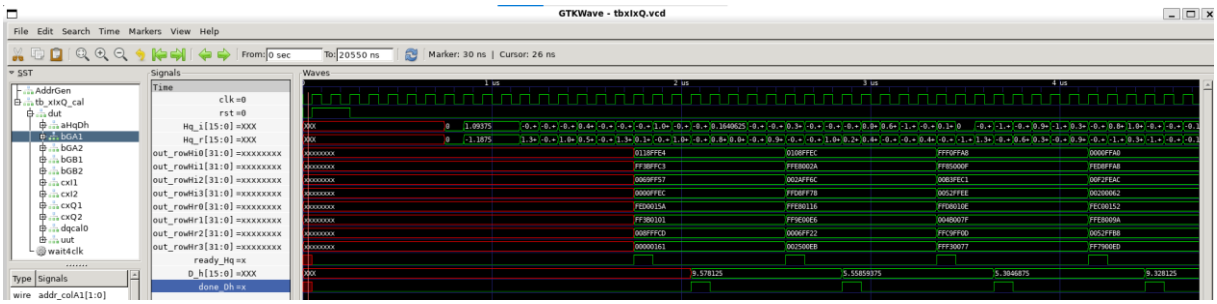


Hình 3.2 Kết quả mô phỏng khối chia pipeline

- Kết quả

Như đã đề cập phần trước, khối chia được thiết kế theo cấu trúc pipeline a+b+3 Stage. Trong trường hợp, em sử dụng số fixed point có dạng gồm 8 bit để mô tả thập phân và 8 bit để mô tả phần nguyên, vì vậy khối chia gồm 19 giai đoạn. Và sau 19clk đầu tiên sẽ cho ra được kết quả của phép chia đầu tiên.

3.2.3 Mô phỏng khối tính ma trận Hq và giá trị Dh.



Hình 3.3 Kết quả mô phỏng tính ma trận Hq và giá trị Dh

- Phân tích kết quả :

Ta có thể quan sát trên Waveform thấy sau reset thì sau đó 6clk sẽ tính được giá trị đầu tiên của ma trận Hq sau đó cứ mỗi clk lại cuất ra 1 giá trị của ma trận Hq. Sau đó các phần từ Hq được dịch dần thành các hàng của ma trận Hq. Khi đủ 1 hàng tín hiệu ready_Hq sẽ lên mức 1 trong 1 chu kỳ. Và tương tự khi tín hiệu done_Dh lên mức 1 là thông báo đã tính xong giá trị Hd.

- Tổng hợp kết quả

Ma trận Hq tính được trên mô phỏng Verilog :

$$\boldsymbol{Hq} = \begin{bmatrix} -1.18375 + 1.09375\text{i} & 1.35156 - 0.109375\text{i} \\ -0.76953 - 0.76953\text{i} & 1.0039 + -0.23828\text{i} \\ 0.558537 + 0.4101\text{i} & -0.199218 - 0.66015\text{i} \\ 1.3789 - 0.078125\text{i} & 0.14453 - 0.15625\text{i} \end{bmatrix}$$

Ma trận Hq tính được trên mô phỏng Matlab:

$$\boldsymbol{Hq} = \begin{bmatrix} -1.1892 + 1.0955\text{i} & 1.3533 - 0.1084\text{i} \\ -0.7713 - 0.7718\text{i} & 1.0097 - 0.2356\text{i} \\ 0.5609 + 0.4116\text{i} & -0.1958 - 0.6579\text{i} \\ 1.3822 - 0.0800\text{i} & 0.1469 - 0.1549\text{i} \end{bmatrix}$$

Giá trị Dh tính được trên mô phỏng Verilog:

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
9.578	5.558	5.034	9.328	9.230	5.238	4.960	8.976	11.332	7.089	7.66	11.914	7.488	3.734	2.621	6.394

Giá trị Dh tính được trên mô phỏng Matlab:

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
9.6407	5.6108	5.3508	9.327	5.2980	5.0380	5.038	9.067	11.399	7.150	7.725	11.9754	7.568	3.758	2.663	6.473

- Nhận xét

Giá trị tính được ở hệ thống có sai số nhỏ so mới kết quả mô phỏng trên matlab do ở hệ thống sử dụng số fixed-point nên sẽ hạn chế trong việc biểu diễn phần thập phân.

3.2.1 Mô phỏng khối

3.3 HƯỚNG NGHIÊN CỨU TRONG TƯƠNG LAI

3.3.1 Hoàn thiện tích hợp hệ thống

- **Mục tiêu**: Tích hợp các khối riêng lẻ (giải mã SO-ML, mã hóa, giải mã kênh, điều chế, giải điều chế) thành một hệ thống hoàn chỉnh.
- **Công việc cần làm**:
    - Xây dựng module liên kết giữa các khối để đảm bảo tương thích dữ liệu và giao tiếp tín hiệu.

- Mô phỏng toàn bộ hệ thống trên công cụ mô phỏng HDL để kiểm tra hiệu năng và tính chính xác.
- Tối ưu hóa thiết kế để giảm độ phức tạp và tăng tốc độ xử lý.

3.3.2 Triển khai trên FPGA

- **Mục tiêu**: Thực hiện hóa thiết kế trên FPGA để kiểm tra hiệu năng thực tế.
- **Công việc cần làm**:
  - Ánh xạ (synthesize) toàn bộ hệ thống lên FPGA, tối ưu việc sử dụng tài nguyên logic và bộ nhớ.
  - Thực hiện kiểm tra tín hiệu đầu vào/đầu ra trên phần cứng bằng cách sử dụng các tín hiệu testbench thực tế.
  - Xử lý các vấn đề thực tế như trễ xử lý (latency), đồng bộ xung nhịp (clock synchronization), và nhiễu tín hiệu (signal interference).

3.3.3 Phân tích hiệu năng toàn hệ thống

- **Mục tiêu**: Đánh giá hiệu năng của hệ thống qua các chỉ số như:
  - **Tốc độ xử lý (throughput)**: Đo lường dữ liệu được xử lý trong một giây.
  - **Độ chính xác giải mã (Bit Error Rate - BER)**: Đánh giá mức độ chính xác của tín hiệu đầu ra so với tín hiệu gốc.
  - **Tiêu thụ năng lượng**: Đo lường năng lượng tiêu thụ trên FPGA, đặc biệt là trong các ứng dụng thực tế.

3.3.4 Tối ưu hóa thiết kế

- **Mục tiêu**: Giảm độ phức tạp của thuật toán và tối ưu hóa tài nguyên FPGA.
- **Công việc cần làm**:
  - Sử dụng các chiến lược nén dữ liệu (data compression) hoặc pipeline để tăng tốc độ xử lý.
  - Phân tích và cải tiến các thành phần có độ trễ cao hoặc tiêu thụ nhiều tài nguyên.

- So sánh hiệu năng của các thuật toán khác nhau (ví dụ: ML, ZF, hoặc các thuật toán heuristic).

3.4 KẾT LUẬN CHƯƠNG

Trong chương 3, các kết quả mô phỏng của hệ thống MIMO SM-OSTBC 4×4 với thuật toán giải mã SO-ML đã được trình bày và phân tích chi tiết. Trên môi trường MATLAB, hệ thống được đánh giá qua chỉ số BER theo SNR với các thông số kênh Rayleigh fading và điều chế 16-QAM. Trong khi đó, mô phỏng trên trình mô phỏng HDL đã xác nhận khả năng thực thi của từng khối trong hệ thống, với test case cụ thể được lựa chọn từ mô phỏng MATLAB.

Kết quả so sánh giữa các giá trị tính toán trên MATLAB và Verilog cho thấy sai số nhỏ, chủ yếu do hạn chế trong việc biểu diễn số fixed-point. Điều này khẳng định độ chính xác của thiết kế HDL so với mô phỏng lý thuyết.

KẾT LUẬN

Trong đề tài *"Nghiên cứu, tìm hiểu một số phương pháp giải mã không gian, thời gian và thực thi trên phần cứng FPGA"*, chúng em đã tập trung nghiên cứu hệ thống MIMO với các phương pháp mã hóa không gian - thời gian (OSTBC), điều chế không gian (SM), và thuật toán giải mã tối ưu (SO-ML).

Các nội dung nghiên cứu đã được trình bày và triển khai theo các bước cụ thể:

Cơ sở lý thuyết: Đề tài đã làm rõ các khái niệm và phương pháp toán học cơ bản của hệ thống MIMO, OSTBC, và SM. Đặc biệt, phần lý thuyết đã cung cấp nền tảng vững chắc để phát triển và mô phỏng hệ thống.

Thiết kế hệ thống: Chúng em đã bước đầu xây dựng các module chức năng, bao gồm khối tính toán ma trận, khối nhân, chia, tích lũy, và khối tìm giá trị tối ưu. Tuy nhiên, do tính phức tạp của hệ thống, việc hoàn thiện tất cả các module vẫn chưa được thực hiện đầy đủ. Đây chính là một trong những mục tiêu chính cho các giai đoạn nghiên cứu tiếp theo.

Mô phỏng: Đề tài đã mô phỏng thành công trên cả MATLAB và trình mô phỏng HDL (Iverilog và GTKWave). Kết quả cho thấy sự nhất quán giữa các giá trị tính toán trên lý thuyết và mô phỏng, với sai số nhỏ xuất phát từ việc sử dụng số fixed-point.

Hướng nghiên cứu tiếp theo: Đề tài đã đề xuất các bước hoàn thiện, bao gồm tích hợp đầy đủ các module còn thiếu, triển khai hệ thống trên FPGA, đánh giá hiệu năng thực tế, và tối ưu hóa thiết kế.

Nhìn chung, kết quả nghiên cứu đã đạt được mục tiêu ban đầu là nghiên cứu lý thuyết, xây dựng và mô phỏng một phần hệ thống giải mã không gian - thời gian. Tuy nhiên, trong quá trình thực hiện, vẫn còn một số hạn chế như:

Chưa hoàn thiện đầy đủ tất cả các module thiết kế do tính phức tạp của hệ thống.

Sai số nhỏ do sử dụng số fixed-point.

Hệ thống hiện mới được kiểm chứng trên môi trường mô phỏng, chưa triển khai thực tế trên FPGA.

Những hạn chế này sẽ là tiền đề để tiếp tục hoàn thiện và tối ưu hệ thống trong tương lai.

Cuối cùng, đề tài không chỉ là cơ hội để chúng em tìm hiểu sâu hơn về lĩnh vực truyền thông không dây và thiết kế phần cứng, mà còn có thể là tài liệu tham khảo hữu ích trong giảng dạy các môn học như Xử lý tín hiệu số hay thiết kế logic số.

Chúng em xin chân thành cảm ơn sự hướng dẫn tận tình của các thầy cô giáo, và mong nhận được những góp ý để tiếp tục cải thiện trong các nghiên cứu sau này.

PHỤ LỤC CÔNG THỨC.

$$\mathrm{B}(\mathrm{C},\hat{C}) = \mathrm{C} - \hat{C} = SX - \hat{S}\hat{X} \quad (40)$$

$$A(C,\hat{C}) = B^H(C,\hat{C})B(C,\hat{C}) \quad (41)$$

$$\mathrm{B}(\mathrm{C},\hat{C}) = \mathrm{C} - \hat{C} = S(X - \hat{X}) \quad (42)$$

$$A(C,\hat{C}) = (X,-\hat{X})^H S^H S(X,\hat{X}) \quad (43)$$

$$A(C,\hat{C}) = (X - \hat{X})^H (X,\hat{X}) = \sum_{n=1}^{2} |x_n - \hat{x}_n|^2 I_2 \quad (44)$$

$$D_1 = \left[\sum_{n=1}^{2} |x_n - \hat{x}_n|^2\right]^2 \geq |x_1 - \hat{x}_1|^4 \geq D_{1,min} \quad (45)$$

$$D_{1,min} = \begin{cases} 16 & với\ M-QAM \\ 16\sin^4\dfrac{\pi}{M} & với\ M-PSK \end{cases} \quad (46)$$

$$\mathrm{B}(\mathrm{C},\hat{C}) = (\mathrm{C} - \hat{C}) = (\mathrm{S} - \hat{S})X \quad (47)$$

$$\mathrm{A}(\mathrm{C},\hat{C}) = X^H(\mathrm{S} - \hat{S})^H(\mathrm{S} - \hat{S})X \quad (48)$$

$$(\mathrm{S} - \hat{S})^H(\mathrm{S} - \hat{S}) = (G(s) - G(\hat{s}))^H(G(s) - G(\hat{s})) = \frac{1}{\mathbb{\Gamma}^2}\sum_{n=1}^{n_T} |s_n - \hat{s}_n|^2 I_2 \quad (49)$$

$$A(\mathrm{C},\hat{C}) = \frac{1}{\mathbb{\Gamma}^2}\sum_{n=1}^{n_T} |s_n - \hat{s}_n|^2 X^H X = \frac{1}{\mathbb{\Gamma}^2}\sum_{n=1}^{n_T} |s_n - \hat{s}_n|^2 \sum_{k=1}^{2} |x_k|^2 I_2 \quad (50)$$

$$D_2 = \frac{1}{\mathbb{\Gamma}^4}\left[\sum_{n=1}^{n_T} |s_n - \hat{s}_n|^2\right]^2 \left[\sum_{k=1}^{2} |x_k|^2 \right]^2 \quad (51)$$

$$\left[\sum_{n=1}^{n_T} |s_n - \hat{s}_n|^2\right]^2 \geq |s_b - \hat{s}_b|^4 \geq 4 \quad (52)$$

$$\left[\sum_{k=1}^{2}|x_k|^2\right]^2 \geq \begin{cases}16 & với\ M-QAM\\ 4 & với\ M-PSK\end{cases} \quad (53)$$

$$D_2 \geq D_{2,min} \quad (54)$$

$$D_{2,min} = \begin{cases}\frac{64}{\mathbb{\Gamma}^4} & với\ M-QAM\\ \frac{16}{\mathbb{\Gamma}^4} & với\ M-PSK\end{cases} \quad (55)$$

$$B(C,\hat{C}) = C-\hat{C} = \frac{1}{\mathbb{\Gamma}}\begin{bmatrix} c-\widehat{c_1} & -c_2^*+\hat{c}_2^* \\ c-\hat{c}_2 & c_1^*-\hat{c}_1^* \\ \vdots & \vdots \\ c_{n_T-1}-\widehat{c_1}_{n_T-1} & -c_{n_T}^*+\hat{c}_{n_T}^* \\ c_{n_T}-\widehat{c_1}_{n_T} & c_{n_T-1}^*-\hat{c}_{n_T-1}^*\end{bmatrix} \quad (56)$$

$$A(C,\hat{C}) = B^H(C,\hat{C})B(C,\hat{C}) = \frac{1}{\mathbb{\Gamma}^2}\sum_{n=1}^{n_T}|c_n-\hat{c}_n|^2 I_2 \quad (57)$$

$$\sum_{n=1}^{n_T}|c_n-\hat{c}_n|^2 \geq \sum_{n=1}^{2}|c_n-\hat{c}_n|^2 = |x_1-\hat{x}_1+x_2-\hat{x}_2|^2+|-(x_1-\hat{x}_1)+x_2-\hat{x}_2|^2$$

$$= 2\sum_{n=1}^{2}|x_n-\hat{x}_n|^2 > 0 \quad (58)$$

$$D_3 = \frac{1}{\mathbb{\Gamma}^4}\left[\sum_{n=1}^{n_T}|c_n-\hat{c}_n|^2\right]^2 \quad (59)$$

$$D_3 \geq \frac{4}{\mathbb{\Gamma}^4}\sum_{n=1}^{2}|x_n-\hat{x}_n|^2 \geq D_{3,min} \quad (60)$$

$$D_{3,min} = \frac{4}{\mathbb{\Gamma}^4}D_{1,min} = \begin{cases}\frac{64}{\mathbb{\Gamma}^4} & với\ \ M-QAM\\ \frac{64}{\mathbb{\Gamma}^4}\sin^4\left(\frac{\pi}{M}\right) & với\ M-PSK\end{cases} \quad (61)$$

$$\delta_{min} = \min(D_{1,min}, D_{2,min}, D_{3,min}). \quad (62)$$

TÀI LIỆU THAM KHẢO

Giáo trình & bài giảng Tiếng Việt:

[1] TS. Lê Minh Tuấn, TS. Trần Xuân Nam, Xử lý tín hiệu không gian – thời gian.

[2] ThS. Trần Thúy Hà, ThS. Đỗ Mạnh Hà, Giáo trình điện tử số, Nhà xuất bản Bưu điện, 2009.

[3] TS Đỗ Mạnh Hà, TS. Đặng Hoài Bắc, Thiết kế Logic số, Nhà xuất bản Thông tin và Truyền Thông, 2015.

Tài liệu tham khảo Tiếng Anh:

[1] M.-T. Le, V.-D. Ngo, H.-A. Mai, and X.-N. Tran, "High rate space time block coded spatial modulation," in Proc. 2012 Int. Conf. Advanced Technol. Commun., pp. 278-282.

[2] G. J. Foschini and M. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," Wireless Per. Commun., vol. 6, pp. 311-335, 1998.

[3] E. Telatar, "Capacity of multi-antenna Gaussian channels," European Trans. Telecommun., vol. 10, no. 6, pp. 558-595, Nov./Dec. 1999.

[4] G. D. Golden, G. J. Foschini, R. A. Valenzuela, and P. W. Wolniansky, "Detection algorithm and initial laboratory results using the V-BLAST space-time communication architecture," Electron. Lett., vol. 35, no. 1, pp. 14-15, Jan. 1999.

[5] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," IEEE J. Sel. Areas Commun., vol. 16, no. 8, pp. 1451- 1458. Oct. 1988.

[6] V. Tarokh, H. Jafarkhani, and A. R. Calderbank, "Space-time block codes from orthogonal designs," IEEE Trans. Inf. Theory, vol. 45, no. 7, pp. 1456-1467, Jul. 1999.