Thiết kế và tối ưu bộ giải mã TPC trên FPGA

Nguyễn Văn Phê[1*], Đặng Văn Bình[1], Nguyễn Thiên Tân[2], Lê Văn Hồng[1]

[1]Trung tâm Kỹ thuật Thông tin Công nghệ cao, số 9 Quan Nhân, Nhân Chính, Thanh Xuân, Hà Nội;
[2]Khoa Điện tử, Trường Đại học Kinh tế - Kỹ thuật Công nghiệp.
[*]Email: nvphe1905@gmail.com
Nhận bài: 04/10/2022; Hoàn thiện: 01/02/2023; Chấp nhận đăng: 02/02/2023; Xuất bản: 28/02/2023.
DOI: https://doi.org/10.54939/1859-1043.j.mst.85.2023.26-34

TÓM TẮT

Trong bài báo này, chúng tôi đề xuất giải pháp thực thi giải mã hóa mã TPC – Turbo Product Code – trên FPGA (Field Programmable Gate Array) cho các hệ thống thông tin liên lạc. Thiết kế dựa trên phương pháp giải mã lặp sử dụng thuật toán Chase-Pydiah nhằm nâng cao hiệu năng giải mã. Nhóm tác giả đơn giản hóa thuật toán giải mã, không sử dụng bộ nhân khi tính toán thông tin ngoại lai tại mỗi vòng lặp nhưng không làm giảm hiệu năng giải mã. Thực thi trên Xilinx Artix7 xc7a200tfbg484-1 tại clock 200 MHz, tốc độ giải mã đạt tới 800 Mbps.

Từ khoá: TPC; SISO; Thuật toán Chase-Pyndiah; QAM; FPGA.

1. MỞ ĐẦU

Mã TPC là mã sửa sai dạng Turbo được cấu tạo từ mã thành phần như mã Hamming [1], mã BCH (Bose-Chaudhuri-Hocquenghem) [2], mã SPC (Single Parity Check) [3]. So với các loại mã Turbo khác, mã TPC có cấu trúc đơn giản hơn, tốc độ hội tụ khi giải mã nhanh hơn nhưng lại có hiệu năng sửa sai tương đương. Mã TPC được ứng dụng rộng rãi trong các hệ thống thông tin chuẩn [4–6], thông tin vệ tinh, truyền hình ảnh và các lĩnh vực khác. Vào năm 1994, Pyndiah đề xuất thuật toán giải mã lặp với quyết định mềm SISO – Soft Input Soft Output [7]. Phương pháp SISO bao gồm hai bước: giải mã quyết định cứng HDD (Hard-Decision Decoding), tính toán thông tin ngoại lai cho vòng lặp tiếp theo dựa trên thuật toán Chase; qua đó nâng cao hiệu năng sửa sai của mã TPC gần đạt tới giới hạn Shannon [8]. Tiếp đó, tác giả cải tiến thuật toán giải mã lặp, sử dụng kiến trúc gần tối ưu để giảm độ phức tạp tính toán mà không ảnh hưởng tới hiệu năng giải mã nhằm dễ dàng thực thi trên phần cứng cho các bài toán ứng dụng thực tế [9]. Hiện nay, thuật toán Chase-Pydiah được sử dụng phổ biến nhất cho giải mã TPC.

Để đạt được thông lượng lớn khi ứng dụng mã TPC trên phần cứng thường sử dụng FPGA hoặc ASIC – Application Specific Integrated Circuit. Trong đó, FPGA là phương án linh hoạt hơn, tuy nhiên, thông lượng lại thấp hơn so với ASIC. Khi thiết kế bộ giải mã TPC trên FPGA, tăng thông lượng có thể thực hiện bằng cách tăng xung nhịp của hệ thống. Tuy nhiên, điều này yêu cầu kiến trúc bộ giải mã phải đơn giản.

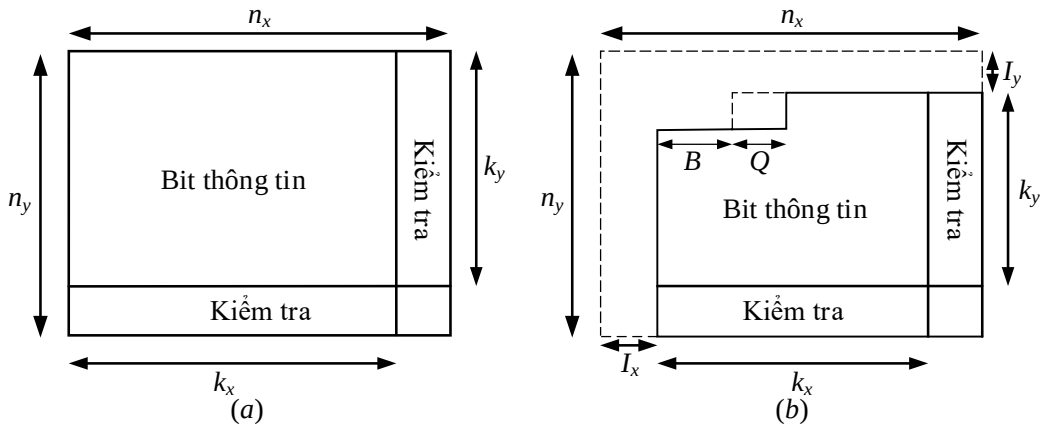
Trong bài báo này, nhóm tác giả thực thi thiết kế bộ giải mã TPC chuẩn IEEE 802.16 – 2017 [4]. Việc tối ưu thuật toán Chase-Pyndiah được thực thi bằng cách sử dụng các biến đổi xấp xỉ nhằm loại bỏ các phép nhân trong tính toán, qua đó giúp đơn giản hóa kiến trúc bộ giải mã.

2. MÃ TPC VÀ THUẬT TOÁN GIẢI MÃ TPC

2.1. Tổng quan về mã TPC

Giả sử rằng, mã TPC 2D được cấu thành từ hai mã khối thành phần $C_x = (n_x, k_x, d_x)$ và $C_y = (n_y, k_y, d_y)$. Trong đó, k_x, k_y là số lượng bit thông tin; n_x, n_y là độ dài mã; d_x, d_y là khoảng cách Hamming nhỏ nhất. Mã TPC $C_x \otimes C_y$ thu được có độ dài mã là $n_x \times n_y$, số bit thông tin là $k_x \times k_y$, khoảng cách Hamming nhỏ nhất là $d_x \times d_y$. Tỉ lệ mã của mã TPC là:

$$R = \frac{k_x k_y}{n_x n_y}, \quad (1)$$



Hình 1. *Cấu trúc mã TPC (a) và TPC rút gọn (b).*

Quá trình mã hóa mã TPC 2D được thực hiện qua ba bước như sau:

Bước 1: Các bit thông tin được điền vào ma trận $\boldsymbol{M}_1$ có kích thước $k_x \times ky$.

Bước 2: Mỗi hàng của ma trận $\boldsymbol{M}_1$ được mã hóa bởi mã Hamming $C_x = (n_x, k_x, d_x)$, kết quả thu được là ma trận $\boldsymbol{M}_2$.

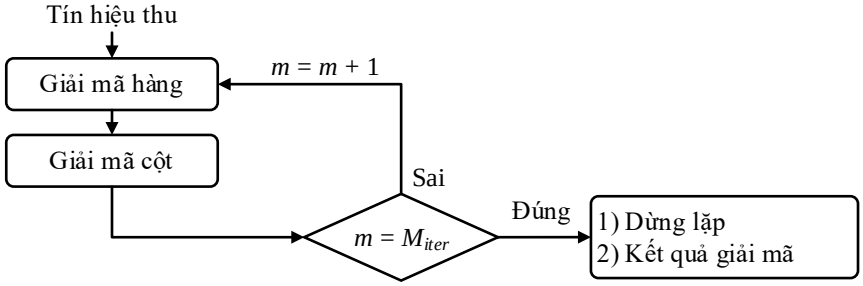
Bước 3: Mỗi cột của ma trận $\boldsymbol{M}_2$ được hóa bởi mã Hamming $C_y = (n_y, k_y, d_y)$, kết quả thu được là ma trận $\boldsymbol{M}_3$ có kích thước $n_x \times n_y$ như biểu diễn trên hình 1.*a*.

Thay đổi tỉ lệ mã được thực hiện bằng cách chèn bit "0" vào I_y hàng, I_x cột và $B + Q$ bit tại hàng thứ $I_y + 1$ (trong đó, Q là số bit để số lượng byte trong mỗi gói thông tin là số nguyên dương) khi thực hiện mã hóa như hình 1.*b*. Sau khi mã hóa xong các bit "0" chèn thêm này (trừ Q bit) sẽ không được truyền đi. Độ dài của khối mã thu được là $(n_x - I_x) \times (n_y - I_y) - B$, số lượng bit thông tin là $(k_x - I_x)(k_y - I_y) - B - Q$. Khi đó tỉ lệ mã là:

$$R = \frac{(k_x - I_x)(k_y - I_y) - B - Q}{(n_x - I_x)(n_y - I_y) - B}, \tag{2}$$

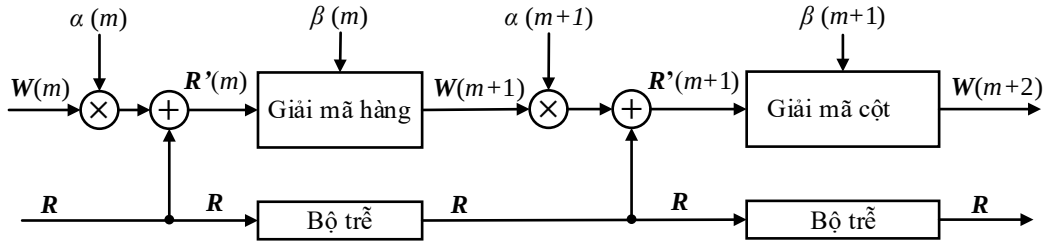
2.2. Kiến trúc giải mã lặp

Mã TPC về bản chất là một dạng mã kết nối nối tiếp nên có hiệu năng sửa sai tốt nếu sử dụng thuật toán giải mã lặp [9]. Với mã TPC 2D quá trình giải mã được thực hiện tuần tự như quá trình mã hóa: giải mã hàng, giải mã cột. Tuy nhiên, thứ tự giải mã cũng có thể thực hiện ngược lại: giải mã cột, giải mã hàng. Lưu đồ thuật toán giải mã lặp được mô tả trên hình 2, trong đó, m là chỉ số vòng lặp, M_{iter} là số vòng lặp tối đa.



Hình 2. *Lưu đồ thuật toán giải mã lặp.*

Trên hình 3 mô tả kiến trúc giải mã lặp theo thứ tự giải mã hàng – giải mã cột. Kiến trúc giải mã lặp cho mã TPC được chia thành hai phần: phần giải mã đầu vào mềm, đầu ra mềm SISO và phần tính toán thông tin ngoại lai.



Hình 3. *Kiến trúc giải mã lặp.*

Giả sử rằng, các bit mã hóa trong ma trận $\boldsymbol{M}_3$ được truyền đi trên kênh truyền Gauss sử dụng ký tự {–1, +1} bởi ánh xạ 0 → 1 và 1 → –1. Ma trận $\boldsymbol{R}$ có kích thước $n_x \times n_y$ là tín hiệu ở đầu thu dưới tác động của nhiễu. Giả sử $\boldsymbol{W}(m)$ là thông tin ngoại lai tại vòng lặp thứ m. Thông tin mềm đầu vào $\boldsymbol{R}'(m)$ của bộ giải mã hàng (hoặc cột) được tính như sau:

$$\mathbf{R}'(m) = \mathbf{R} + \alpha(m)\mathbf{W}(m) \tag{3}$$

với $\alpha(m)$ là hệ số tỉ lệ do độ lệch chuẩn của các phần tử trong ma trận $\boldsymbol{R}$ và $\boldsymbol{W}$ là khác nhau [9].

Để giải mã mềm, Pydiah sử dụng thuật toán Chase được giới thiệu lần đầu tiên vào năm 1972 [8]. Đồng thời tác giả đề xuất sử dụng các công thức tính xấp xỉ để tính toán thông tin ngoại lai nhằm giảm độ phức tạp tính toán. Hiện nay, thuật toán Chase-Pyndiah trở thành thuật toán phổ biến để giải mã TPC.

2.3. Thuật toán Chase-Pyndiah

Như đã trình bày ở trên, mỗi hàng hoặc cột của mã TPC là một mã khối Hamming. Để giảm độ phức tạp, việc giải mã đầu vào mềm, đầu ra mềm và tính toán thông tin ngoại lại riêng rẽ cho từng khối. Do kích thước mã khối hàng và cột có thể khác nhau, để đơn giản, chúng tôi giả sử mã Hamming dùng để mà hóa là $C(k, n)$. Khi đó, sử véc tơ $\boldsymbol{u} = (u_1, u_2, \ldots, u_k)$ bit thông tin sẽ được mã hóa thành véc tơ $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ bởi mã $\mathbf{C}(n, k)$ sau đó truyền đi qua kênh Gauss sử dụng ký tự {-1, +1} cho điều chế QAM-4. Véc tơ $\boldsymbol{r} = (r_1, r_2, ..., r_n)$ là tín hiệu ở đầu thu dưới tác động của nhiễu, $\boldsymbol{y} = (y_1, y_2, ..., y_n)$ là quyết định cứng dựa trên $\boldsymbol{r}$. Thuật toán Chase-Pyndiah gồm các bước sau [9]:

1. Xác định vị trí của p phần tử có khả năng lỗi cao nhất của $\boldsymbol{y}$ tương ứng với p giá trị nhỏ nhất của $|\boldsymbol{r}|$.

2. Tạo tập hợp 2^p véc tơ kiểm tra $\boldsymbol{T}$, với $\mathbf{T}^i = \left(t_1^i, t_2^i, \ldots, t_n^i\right)$, $i = 1, \ldots, 2^p$. $\boldsymbol{T}$ là tập hợp các véc tơ kích thước n với bit "0" hoặc "1" tại p vị trí có khả năng lỗi cao nhất tìm được tại bước 2 và bit "0" tại các vị trí còn lại.

3. Tạo tập hợp 2^p chuỗi kiểm tra $\boldsymbol{Z}$, với $\boldsymbol{Z}^i = \boldsymbol{y} \oplus \boldsymbol{T}^i$.

4. Thực hiện giải mã quyết định cứng cho $\boldsymbol{Z}^i$ như khi giải mã Hamming thông thường. Kết quả giải mã là các ứng viên $\mathbf{C}^i = (c_1^i, c_2^i, ..., c_n^i)$.

5. Tính khoảng cách Euclidean giữa từ mã $\boldsymbol{C}^i$ và tín hiệu thu được $\boldsymbol{r}$:

$$\left\|\mathbf{r} - \mathbf{C}^i\right\|^2 = \sum_{j=1}^{n}\left[r_j - c_j^i\right]^2, i = 1, ..., 2^p. \tag{4}$$

6. Tìm từ mã $\boldsymbol{d}$ có khoảnh cách Euclidean từ $\boldsymbol{r}$ là nhỏ nhất giữa 2^p từ mã ứng viên $\boldsymbol{C}^i$:

$$\left\|\mathbf{r} - \mathbf{d}\right\|^2 = \min\left\|\mathbf{r} - \mathbf{C}^i\right\|, \; i = 1, ..., 2^p. \tag{5}$$

7. Thông tin ngoại lai $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ được tính như sau:

$$w_j = \begin{cases} \left(\dfrac{\|\mathbf{r}-\mathbf{v}\|^2 - \|\mathbf{r}-\mathbf{d}\|^2}{4} \right) d_j - r_j, \text{ khi } d_j \neq v_j \\ \beta d_j, \qquad\qquad\qquad\qquad \text{khi } d_j = v_j \end{cases} \tag{6}$$

với $j = 1, ..., n$; β là hệ số tin cậy được giới thiệu trong [9]; $\boldsymbol{v}$ là từ mã cạnh tranh với từ mã $\boldsymbol{d}$ (tức là từ mã có khả năng tin cậy thứ nhì, sau từ mã $\boldsymbol{d}$) và được tìm theo nguyên tắc sau:

$$\mathbf{v} = \mathbf{C}^i \text{ if } \left|\mathbf{r}-\mathbf{C}^i\right|^2 \leq \left|\mathbf{r}-\mathbf{C}^j\right|^2, \ \forall \mathbf{C}^j \neq \mathbf{d}. \tag{7}$$

8. Giá trị hợp lý vào cho vòng lặp tiếp theo được tính bởi công thức:

$$\mathbf{r}'(m+1) = \mathbf{r} + \alpha(m)\mathbf{w}(m), \tag{8}$$

Các hệ số tin cậy α, β thu được thông qua thực nghiệm. Giá trị của các hệ số này tại mỗi vòng lặp được liệt kê trong bảng 1 [9].

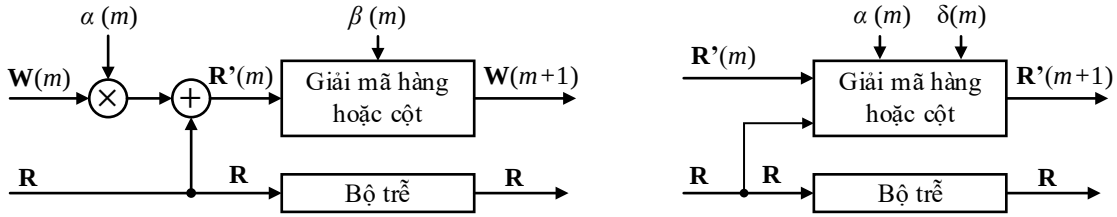
Bảng 1. *Giá trị α, β theo số vòng lặp m.*

m	1	2	3	4	5	6	7	8
α	0,0	0,2	0,3	0,5	0,7	0,9	1,0	1,0
β	0,2	0,4	0,6	0,8	1,0	1,0	1,0	1,0

2.4. Thiết kế bộ giải mã TPC trên FPGA

Để thiết kế bộ giải mã TPC trên FPGA, nhóm tác giả sử dụng phần mềm Xilinx Vivado 2019.1 với ngôn ngữ mô tả phần cứng verilog. Trước khi thực thi trên FPGA, thuật toán được được khảo sát bằng công cụ Matlab, trong đó, các biến của chương trình được viết dưới dạng dấu phảy tĩnh. Quá trình này giúp kiểm tra được hiện tượng tràn bộ nhớ, giúp tối ưu được tài nguyên chiếm dụng khi thực thi trên FPGA. Đồng thời chương trình trên Matlab hỗ trợ kiểm tra kết quả tính toán từng khối logic được thiết kế.

Theo truyền thống, để tính toán thông tin ngoại lai và giá trị hợp lý vào cho bộ giải SISO cần sử dụng hai bộ nhân khi thực thi trên FPGA, một bộ nhân với hệ số α, một bộ nhân với hệ số β (hình 4.*a*). Để đơn giản hóa quá trình tính toán trên FPGA, giảm độ trễ do phải tính giá trị thông tin ngoại lai và giá trị tỉ lệ hợp lý cho vòng lặp tiếp theo, chúng tôi đề xuất kết hợp công thức (6) và (8) với nhau. Khi đó, sơ khối bộ giải mã SISO được mô tả trên hình 4.*b*.



(*a*) (*b*)

Hình 4. *Cấu trúc bộ xử lý SISO truyền thống (a) và cải tiến (b).*

Giá trị hợp lý vào cho bộ giải mã SISO đề xuất tại vòng lặp tiếp theo là:

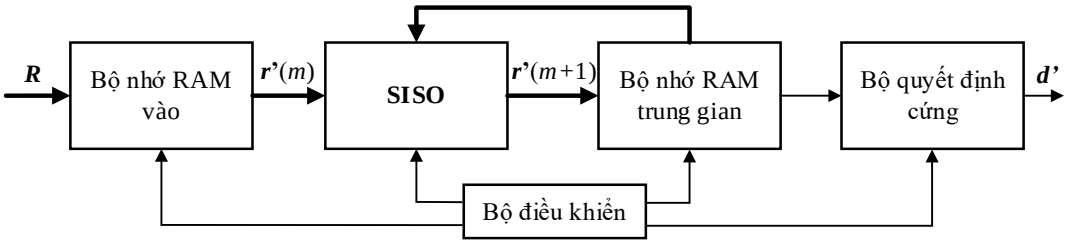
$$\mathbf{r}'(m+1) = \begin{cases} (1-\alpha(m))\mathbf{r} + \alpha(m)\left(\dfrac{\|\mathbf{r}-\mathbf{v}\|^2 - \|\mathbf{r}-\mathbf{d}\|^2}{4} \right)\mathbf{d}, \text{ khi } \mathbf{d} \neq \mathbf{v}, \\ \mathbf{r} + \alpha(m)\beta(m)\mathbf{d}, \qquad\qquad\qquad\qquad \text{khi } \mathbf{d} = \mathbf{v}. \end{cases} \tag{9}$$

Bảng 2. *Giá trị α, δ theo số vòng lặp m.*

m	1	2	3	4	5	6	7	8
α	3/16	5/16	1/2	11/16	7/8	1	1	1
δ	1/16	1/8	3/8	5/16	9/16	7/8	1	1

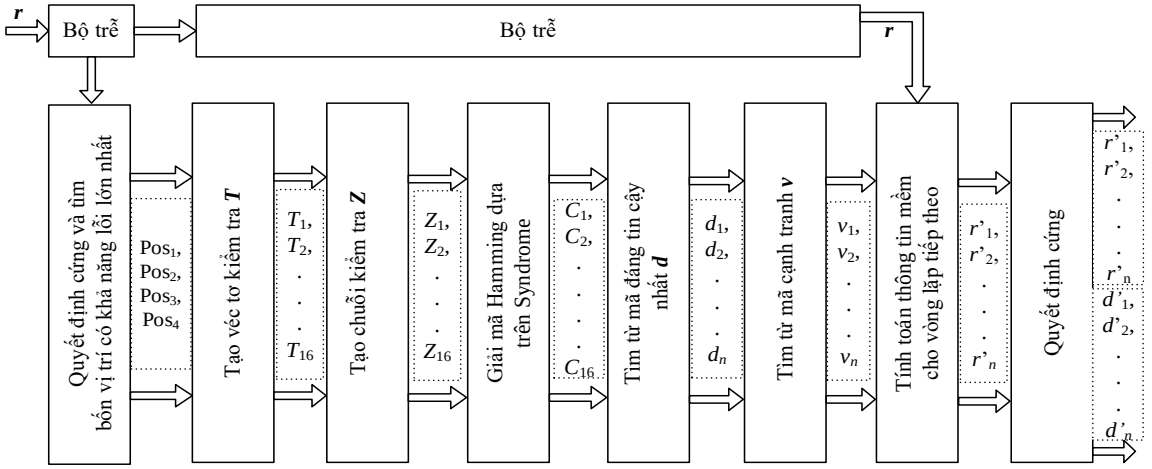
Để tính toán giá trị hợp lý vào cho bộ giải mã SISO tại vòng lặp tiếp theo theo công thức (9), chúng tôi lựa chọn các giá trị $\alpha(m)$ và $\delta(m) = \alpha(m)\beta(m)$ như liệt kê trong bảng 2. Việc lựa chọn này thực chất là phép lấy gần đúng các giá trị trong bảng 1 sao cho các giá trị $\alpha(m)$ và $\delta(m)$ có thể biểu diễn dưới dạng 2^n, với n là số nguyên. Khi đó, các phép nhân trong công thức (9) có thể thực thi trên phần cứng bằng việc dịch bit đơn giản nhằm tăng khả năng đáp ứng bộ giải mã với xung nhịp hệ thống ở tần số cao. Qua mô phỏng đánh giá rằng, việc lựa chọn các hệ số gần đúng này làm giảm hiệu năng sửa sai bộ giải mã không đáng kể.

Căn cứ vào đề xuất trên, sơ đồ khối bộ giải mã TPC được mô tả trên hình 5. Thông tin từ kênh truyền được lưu vào bộ nhớ vào. Bộ điều khiển sẽ điều khiển việc đọc thông tin từ bộ nhớ vào để truyền vào bộ giải mã mềm SISO, kết quả giải mã là quyết định mềm cho vòng lặp tiếp theo được lưu vào bộ nhớ trung gian. Bộ điều khiển sẽ kích hoạt bộ quyết định cứng hoạt động khi số vòng lặp đạt tới giá trị thiết lập.



Hình 5. *Sơ đồ khối bộ giải mã TPC.*

Trung tâm của bộ giải mã TPC là bộ giải mã mềm SISO. Sơ đồ khối bộ SISO được mô tả trên hình 6. Tín hiệu thu ***r*** một mặt đi qua bộ trễ để tính toán thông tin mềm cho vòng lặp tiếp theo, mặt khác đi qua các khối tính xử lý của thuật toán Chase-Pyndiah như mô tả ở trên. Thông qua mô phỏng nhận thấy rằng, với số phần tử có khả năng lỗi cao nhất của thuật toán Chase-Pyndiah $p = 4$ đảm bảo hiệu năng sửa sai là tối ưu, đồng thời mức độ phức tạp của thuật toán là chấp nhận được. Tương ứng với $p = 4$, số chuỗi kiểm tra cần tạo ra là 16.

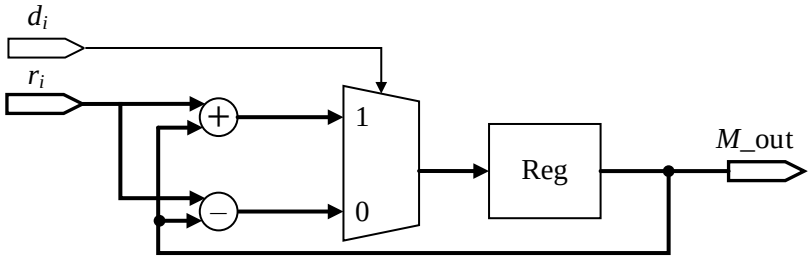


Hình 6. *Sơ đồ khối bộ SISO.*

Hiệu khoảng cách Euclidean trong công thức (6) được đơn giản hóa như sau:

$$\|\mathbf{r}-\mathbf{v}\|^2-\|\mathbf{r}-\mathbf{d}\|^2=2\sum_{i=1}^{n}r_iv_i-2\sum_{i=1}^{n}r_id_i. \quad (10)$$

Khi đó, việc tính toán khoảng cách Euclidean có thể đơn giản hóa bằng việc tính hàm tương quan. Phần tử tính hàm tương quan giữa ***r*** và ***d*** thiết kế trên FPGA được mô tả trên hình 7.

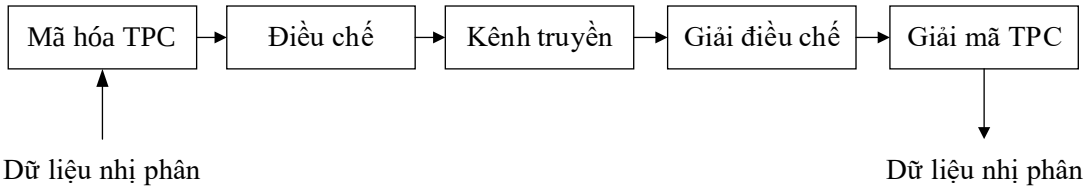


Hình 7. *Phần tử tính khoảng cách Euclid.*

3. MÔ PHỎNG LOGIC VÀ THỰC THI TRÊN FPGA

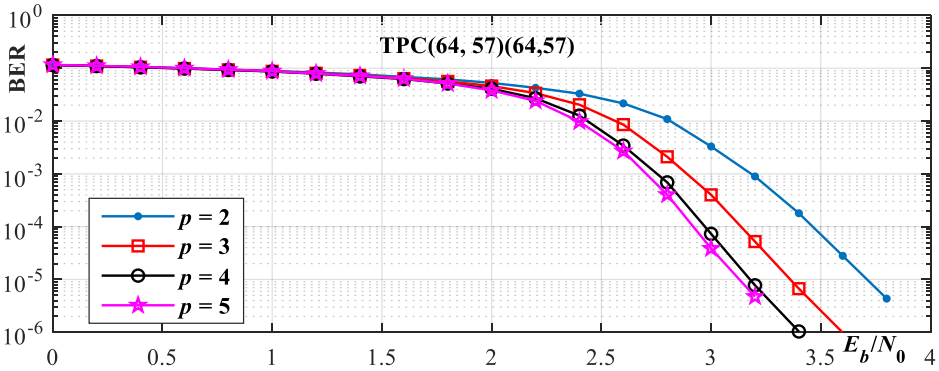
3.1. Phương pháp mô phỏng

Trong quá trình thiết kế bộ giải mã TPC, để đánh giá khả năng sửa lỗi cũng như kiểm tra chức năng của thiết kế, nhóm tác giả đã thực hiện mô phỏng đánh giá chất lượng mã này trên kênh truyền AWGN với thuật toán đã trình bày trên đây. Trong quá trình mô phỏng, các dữ liệu nhị phân được chia thành khối $k_x \times k_y$ bit, mã hóa thành $n_x \times n_y$ bit, qua khối điều chế 4-QAM rồi truyền trên kênh truyền AWGN. Quá trình thu được thực hiện tuần tự ngược lại. Sơ đồ khối hệ thống mô phỏng được biểu diễn trên hình 6. Tham số mã TPC theo chuẩn IEEE 802.16 – 2017 [3]. Tất cả các biến mô phỏng trên Matlab đều được biểu diễn dưới dạng dấu phảy tĩnh giống như trên ngôn ngữ mô tả phần cứng verilog. Sơ đồ mô phỏng hệ thống được trình bày trong hình 8.



Hình 8. *Sơ đồ hệ thống đánh giá chất lượng mã TPC.*

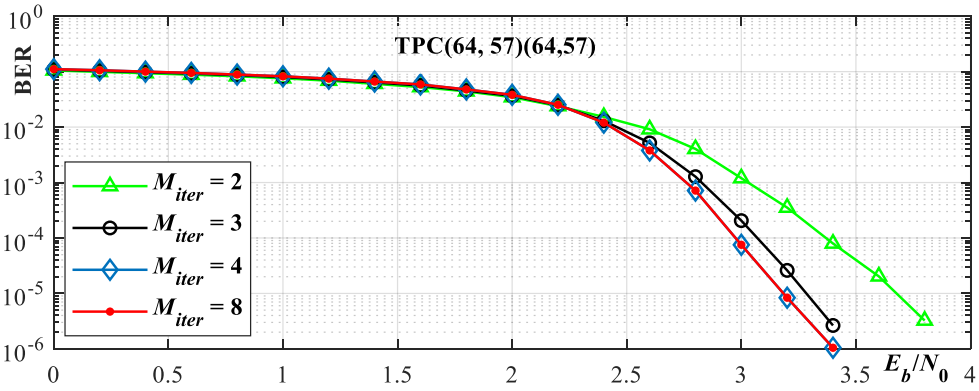
3.2. Kết quả mô phỏng và bình luận



Hình 9. *Xác suất lỗi mã TPC phụ thuộc vào p.*

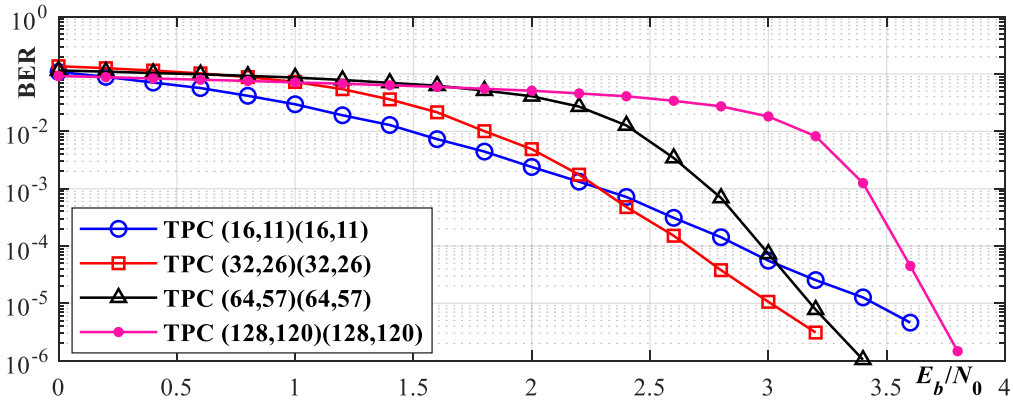
Hình 9 mô tả kết quả mô phỏng mã TPC (64, 57)(64, 57). Với mỗi điểm tính xác suất lỗi, nhằm đảm bảo giá trị BER thu được là đáng tin cậy, số bit truyền đi không nhỏ hơn 10^7, số bit lỗi

tích lũy tại đầu thu không nhỏ hơn 300. Kết quả mô phỏng cho thấy, khi tăng giá trị của p sẽ đảm bảo độ lợi về tỉ lệ E_b/N_0. Điều này được giải thích là, nếu tăng p thì số lượng các mã ứng viên sẽ tăng, qua đó sẽ tăng xác suất tìm thấy mã cạnh tranh. Bên cạnh đó, độ phức tạp của thuật toán cũng tăng lũy thừa theo p.



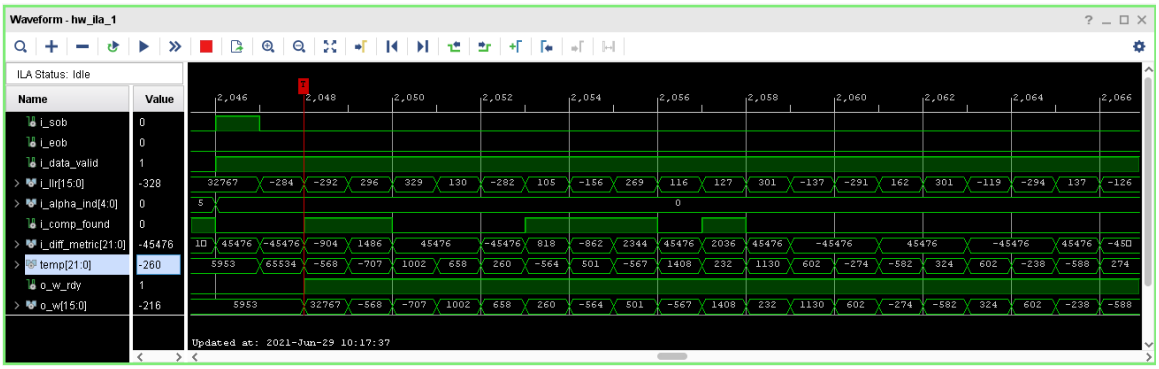
Hình 10. *Xác suất lỗi mã TPC phụ thuộc vào số vòng lặp, p = 4.*

Hình 10 biểu thị so sánh xác suất lỗi của mã TPC (64, 57)(64, 57) với số vòng lặp M_{iter} khác nhau và số phần tử có khả năng lỗi cao nhất của thuật toán Chase-Pyndiah p = 4. Nhận thấy rằng với số vòng lặp $M_{iter} \geq 3$, chất lượng giải mã gần như không thay đổi.



Hình 11. *Xác suất lỗi mã TPC, N_{iter} = 8, p = 4.*

Hình 11 mô tả xác suất lỗi mã TPC (16, 11)(16, 11), (32, 26)(32, 26), (64, 57)(64, 57) và (128, 120)(128, 120) với giá trị p = 4, N_{iter} = 8. Kết quả mô phỏng cho thấy rằng với giá trị tỉ lệ Eb/N0 < 2,5dB mã (16, 11)(16, 11) đảm bảo hiệu năng sửa sai tốt nhất, còn mã (128, 120)(128, 120) có hiệu năng sửa sai thấp nhất. Tuy nhiên, với giá trị tỉ lệ tín/tạp tăng lên thì khả năng sửa sai của các mã trên có sự thay đổi. Mã (16, 11)(16, 11) có hiệu năng sửa sai kém hơn so với mã (32, 26)(32, 26) với Eb/N0 > 2,5dB; còn so với mã (64, 57)(64, 57) với Eb/N0 > 3,2 dB. Xu hướng chung là với mã TPC cấu tạo từ mã Hamming thành phần ngắn hơn có hiệu năng sửa sai tốt hơn so với mã TPC cấu tạo từ mã Hamming thành phần dài hơn tại khi tỉ lệ tín/tạp nhỏ. Khi tỉ lệ tín/tạp đủ lớn thì xu hướng này là ngược lại, mã TPC cấu tạo từ mã Hamming thành phần ngắn hơn có hiệu năng sửa sai kém hơn so với mã TPC cấu tạo từ mã Hamming thành phần dài hơn. Điều này được giải thích là do mỗi mã Hamming thành phần chỉ có thể sửa sai tối đa một lỗi. Khi giá trị tỉ lệ tín/tạp nhỏ thì xác suất có một bit lỗi trong mỗi khối Hamming thành phần là nhỏ hơn với mã ngắn, khi đó quá trình giải mã lặp có khả năng sửa các lỗi sai tốt hơn. Khi tỉ lệ tín/tạp đủ lớn thì xác suất xảy ra một lỗi trong mỗi khối Hamming thành phần lớn, khi đó kết hợp với chiều dài mã lớn hơn nên quá trình giải mã lặp sẽ làm tăng hiệu năng sửa sai.



Hình 12. *Kết quả tính toán thông tin ngoài trên FPGA.*

Hình 12 thể hiện kết quả tính toán thông tin ngoài trên FPGA được trích từ FPGA thông qua công cụ phát triển Vivado của hãng Xilinx.

3.3. Đánh giá kết quả tổng hợp trên FPGA

Bảng 3. *Tốc độ giải mã TPC với clock hệ thống 200 MHz.*

Mã thành phần		Tỉ lệ mã	Tốc độ mã hóa, Mbps	Tốc độ giải mã, Mbps				
				Số vòng hồi quy				
C_x	C_y			4	5	6	7	8
(64, 57)	(64, 57)	0,793	64,0	28,7	24,4	21,2	18,7	16,8
(64, 57)	(46, 39)	0,755	62,9	27,8	23,5	20,4	18,0	16,1
(64, 57)	(32, 26)	0,724	61,9	27,0	22,9	19,8	17,5	15,6
(64, 57)	(16, 11)	0,612	57,8	24,2	20,3	17,5	15,4	13,7
(32, 26)	(32, 26)	0,660	61,9	25,4	21,4	18,5	16,3	14,6
(32, 26)	(16, 11)	0,559	57,8	22,7	19,0	16,3	14,3	12,8
(32, 26)	(8, 4)	0,406	50,0	18,1	15,0	12,8	11,2	9,9

Nhóm tác giả thiết kế bộ mã hóa và giải mã trên FPGA Xilinx Artix xc7a200tfbg484-1. Bảng 3 liệt kê tốc độ mã hóa và giải mã. trên FPGA với clock hệ thống 200 MHz. Tốc độ mã hóa và giải mã hóa được tính cho một IP-core mã hóa hoặc giải mã. Với mã (64, 57)(64, 57) có tỉ lệ mã là 0,793, tốc độ mã hóa đạt tới 64 Mbps, còn tốc độ giải mã đạt tới 28,7 Mbps.

Bảng 4. *Tài nguyên chiếm dụng bộ giải mã TPC (64, 57)(64, 57).*

Tài nguyên	Sử dụng	Có sẵn	Sử dụng %
LUT	4630	133800	3,46
LUTRAM	137	46200	0,30
FF	5529	267600	1,95
BRAM	0,50	365	0,14

Bảng 4 thể hiện tài nguyên chiếm dụng khi thực thi trên FPGA của một IP-core giải mã TPC (64, 57)(64, 57). Nhận thấy rằng, tài nguyên chiếm dụng của kiến trúc đề xuất nhỏ hơn đáng kể so với tài nguyên chiếm dụng trong các công bố [10, 11]. Hơn nữa, trong kiến trúc đề xuất không sử dụng bất cứ DSP vào để thực thi phép nhân. Với tài nguyên của FPGA Xilinx Artix xc7a200tfbg484-1 có thể thiết kế theo cấu trúc song song 28 IP-core giải mã TPC (64, 57)(64, 57). Như vậy, tốc độ giải mã có thể đạt tới 800 Mbps.

4. KẾT LUẬN

Quá trình phát triển mã TPC, nhóm tác giả đã ứng dụng mã này trên hệ thống truyền dẫn vô tuyến Vi ba băng rộng áp dụng chuẩn IEEE Standard 802.16-2017, qua đó khẳng định tính ứng dụng thực tiễn của mã này. Ngoài ra, mã này còn có triển vọng ứng dụng trong các hệ thống khác như hệ thống thông tin vệ tinh, hệ thống thông tin UAV, hệ thống truyền hình mang vác,… Tuy nhiên, kiến trúc giải mã vẫn cần nghiên cứu, cải tiến, áp dụng các kỹ thuật pipeline để tăng tốc độ giải mã lên cao hơn nữa.

TÀI LIỆU THAM KHẢO

[1]. C. Xu, Y. C. Liang, and W. S. Leon, *"Shortened turbo product codes: Encoding design and decoding algorithm,"* IEEE Trans. Veh. Technol., vol. 56, no. 6, pp. 3495–3501, (2007).

[2]. H. Mukhtar, A. Al-Dweik, M. Al-Mualla, and A. Shami, *"Adaptive hybrid ARQ system using turbo product codes with hard/soft decoding,"* IEEE Commun. Lett., vol. 17, no. 11, pp. 2132–2135, (2013).

[3]. D. M. Rankin and T. A. Gulliver, *"Single parity check product codes,"* IEEE Trans. Commun., vol. 49, no. 8, pp. 1354–1362, (2001).

[4]. IEEE Standard for Local and Metropolitan Area Networks Part 16: *"Air Interface for Broadband Wireless Access Systems"*, IEEE Standard 802.16, (2017).

[5]. IEEE Standard for Local and Metropolitan Area Networks—Part 20: *"Air Interface for Mobile Broadband Wireless Access Systems Supporting Vehicular Mobility—Physical and Media Access Control Layer Specification"*, IEEE Standard 802.20, (2008).

[6]. IEEE Standard for Broadband Over Power Line Networks: *"Medium Access Control and Physical Layer Specifications"*, IEEE Standard 1901, (2010).

[7]. R. Pyndiah, A. Glavieux, A. Picart, and S. Jacq, *"Near optimum decoding of product codes,"* in Proc. IEEE GLOBECOM, San Francisco, CA, USA, pp. 339–343, (1994).

[8]. D. Chase, *"A class of algorithms for decoding block codes with channel measurement information,"* IEEE Trans. Inform. Theory, vol IT-18, pp 170-182, (1972).

[9]. R. M. Pyndiah, *"Near-optimum decoding of product codes: Block turbo codes,"* IEEE Trans. Commun., vol. 46, no. 8, pp. 1003–1010, (1998).

[10]. W. Kuang, R. Zhao and Z. Juan, *"FPGA implementation of a modified turbo product code decoder,"* 2017 IEEE 9th International Conference on Communication Software and Networks, Guangzhou, China, pp. 71-74, (2017).

[11]. N. Nageen, Subhashini and V. Bhatia, *"An Efficient FPGA implementation of Turbo Product Code decoder with single and double error correction,"* 2020 National Conference on Communications, Kharagpur, India, pp. 1-6, (2020).

ABSTRACT

TPC decoder design for FPGA implementation

In this paper, the FPGA implementation of Turbo Product Code is presented. The design is based on the iterative decoding structure with using Chase-Pydiah algorithm for better decoding performance. In pacticular, circuit design and implementation based on the simplified algorithm without using multiplication. Simulation and implementation result show that, the decoding perfomance of this presented algorithm is not reduced. The 800 Mbps TPC deccoder was achieved for FPGA implementation on the Xilinx Artix7 xc7a200tfbg484-1 platform at the 200 MHz system clock.

Keywords: TPC; SISO; Chase-Pyndiah algorithm; QAM; FPGA.