JTE HCMUTE
ISSN: 1859-1272

JOURNAL OF TECHNOLOGY EDUCATION SCIENCE
Ho Chi Minh City of Technology and Education
Website: https://jte.hcmute.edu.vn/index.php/jte/index
Email: jte@hcmute.edu.vn

Application of Hamming Code for Error Control in Memory

Tran Do Hon Nhien, Vo Tan Thanh, Nguyen Thanh Khoa, Nguyen Quoc Thang, Nguyen Van Thanh Loc, Huynh Hoang Ha, Nguyen Ngo Lam, Do Duy Tan*

Faculty of Electrical and Electronics Engineering, Ho Chi Minh City of Technology and Education, Vietnam

* Corresponding author. *Email: tandd@hcmute.edu.vn*

ARTICLE INFO

Received: 18/2/2022
Revised: 5/8/2022
Accepted: 19/8/2022
Published: 30/8/2022

KEYWORDS

ECC memory;
Hamming code;
Encoder;
Decoder;
Testbench.

ABSTRACT

Error Correction Coding (Error Correction Coding - ECC) has been considered as a powerful tool to enhance the reliability in data storage by detecting and correcting corrupted data errors that may occur in memory. There are many ECC algorithms used with different advantages such as Hamming, Reed-Solomon, BCH, LDPC, etc. In particular, Hamming code is suitable for controlling errors in memory with the outstanding advantage of simple encoding/decoding algorithms with fast coding speed. In this paper, we present an ECC module design using Hamming code for memory error control. The Hamming code-based ECC design including encoder and decoder blocks is presented in detail. Then, extensive simulation results are conducted to validate the functionality and effectiveness of the designed ECC module.

Ứng Dụng Mã Hamming Trong Kiểm Soát Lỗi Bộ Nhớ

Trần Đỗ Hồn Nhiên, Võ Tấn Thanh, Nguyễn Thành Khoa, Nguyễn Quốc Thắng, Nguyễn Văn Thành Lộc, Huỳnh Hoàng Hà, Nguyễn Ngô Lâm, Đỗ Duy Tân*

Khoa Điện-Điện Tử, Trường Đại Học Sư Phạm Kỹ Thuật TPHCM, Việt Nam

** Tác giả liên hệ. Email: tandd@hcmute.edu.vn*

THÔNG TIN BÀI BÁO

Ngày nhận bài: 18/2/2022
Ngày hoàn thiện: 5/8/2022
Ngày chấp nhận đăng: 19/8/2022
Ngày đăng: 30/8/2022

TỪ KHÓA

Bộ nhớ ECC;
Mã Hamming;
Bộ mã hóa;
Bộ giải mã;
Testbench.

TÓM TẮT

Mã hóa kiểm tra lỗi (Error Correction Coding - ECC) được ứng dụng để làm tăng độ tin cậy trong lưu trữ dữ liệu nhờ khả năng phát hiện và sửa lỗi dữ liệu bị hỏng xảy ra trong bộ nhớ. Có nhiều thuật toán ECC được sử dụng với những ưu điểm khác nhau như mã Hamming, Reed-Solomon, BCH, LDPC. Trong đó, mã Hamming phù hợp với việc kiểm soát lỗi trong bộ nhớ với ưu điểm nổi bậc là thuật toán mã hóa và giải mã đơn giản nên tốc độ mã hóa và giải mã cao. Trong bài báo này, chúng tôi trình bày chi tiết thiết kế module ECC sử dụng mã Hamming trong kiểm soát lỗi bộ nhớ bằng ngôn ngữ mô tả phần cứng VHDL. Thiết kế gồm 2 khối encoder và decoder thực hiện việc mã hóa và giải mã dữ liệu. Sau đó, các kết quả mô phỏng được trình bày để đánh giá chi tiết chức năng và công suất tiêu thụ của thiết kế ECC.

Doi: https://doi.org/10.54644/jte.71B.2022.1141

1. Giới thiệu

DRAM là một loại bộ nhớ truy cập ngẫu nhiên, lưu mỗi bit dữ liệu trong một tụ điện riêng biệt và được nạp xã mỗi lần 64ms [1]. Là một loại bộ nhớ được sử dụng rộng rãi trên các hệ thống máy tính như là một bộ nhớ chính. Lỗi bộ nhớ trong DRAM dẫn đến một vài bit dữ liệu phát sinh chuyển đổi trạng thái ngược lại từ logic 1 về logic 0 hoặc ngược lại [2]. Lỗi bộ nhớ là sự cố xảy ra khi dữ liệu được truy cập để sử dụng bởi chương trình nào đó. Có hai loại lỗi xảy ra thường xuyên nhất là lỗi cứng và lỗi mềm. Lỗi cứng là lỗi lặp lại liên tục do lỗi phần cứng hoặc thiết kế vật lý trên module và một số nguyên

JTE
HCMUTE
ISSN: 1859-1272

JOURNAL OF TECHNOLOGY EDUCATION SCIENCE
Ho Chi Minh City of Technology and Education
Website: https://jte.hcmute.edu.vn/index.php/jte/index
Email: jte@hcmute.edu.vn

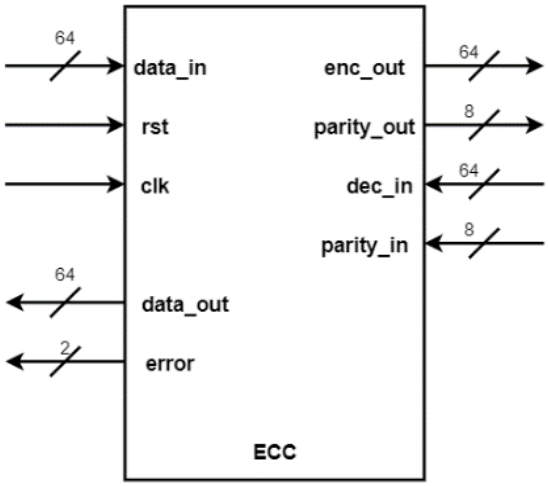
nhân khác như nhiệt độ, áp suất, điện áp, quá trình chế tạo. Lỗi mềm là lỗi làm hỏng ngẫu nhiên các bit trong bộ nhớ, nguyên nhân phổ biến nhất là do bức xạ hạt alpha và nhiễu [3]. Theo nghiên cứu chỉ ra rằng, xác suất lỗi xảy ra trong bộ nhớ rất nhỏ vào khoảng 10^{-9}. Để đảm bảo tính toàn vẹn của dữ liệu khi đọc từ bộ nhớ, dữ liệu cần phải chính xác nên cần phải phát hiện bất kỳ lỗi nào xảy ra và sửa chúng.

ECC dùng để kiểm soát dữ liệu trong bộ nhớ, nghiên cứu cho thấy khoảng 98% lỗi bộ nhớ là bit đơn [4], do đó, mã Hamming được nhiều nhà nghiên cứu lựa chọn trong lĩnh vực này. Trong đó, đề tài [5] trình bày thiết kế bộ mã hóa và giải mã Hamming(7,4) bằng ngôn ngữ mô tả phần cứng VHDL, bộ mã hóa được thiết kế thông qua phép nhân ma trận giữa khối dữ liệu với ma trận tạo, bộ giải mã phát hiện lỗi bằng phương pháp liệt kê từ mã. Đối với đề tài [6] [7], tác giả thiết kế hệ thống sử dụng mã Hamming, bộ mã hóa và giải mã sử dụng phương pháp chẵn lẻ, XORing các bit trong dữ liệu để tạo thành từ mã và giải mã các từ mã đó để khôi phục dữ liệu. Bên cạnh đó, đề tài [8], [9] thực hiện việc tính toán syndrome dựa vào các bit kiểm tra để phát hiện lỗi và tạo mặt nạ sửa lỗi. Tuy nhiên, các thiết kế trong [8][9] chưa trình bày chi tiết sơ đồ khối hoặc thiết kế còn đơn giản chỉ có thể phát hiện lỗi đơn. Bên cạnh đó, các đề tài trên đánh giá số bit rất nhỏ, ở mức 4 bit không phù hợp với thực tế, hoặc 64 bit nhưng các thiết kế module ECC chưa được trình bày và đánh giá một cách chi tiết. Do đó, để cải thiện những hạn chế nêu trên, trong bài báo này, chúng tôi trình bày thiết kế và đánh giá hiệu năng của module ECC sử dụng mã Hamming có độ rộng bus dữ liệu là 64 bit với xác suất lỗi gần đúng với thực tế là 3.2×10^{-4}. Cụ thể, module mở rộng độ rộng dữ liệu lên 64 bit làm tăng hiệu suất đọc ghi nhanh hơn gấp 16 lần so với thiết kế chỉ có độ rộng 4 bit. Ngoài ra, thiết kế có khả năng phát hiện và sửa được lỗi bit đơn, phát hiện được lỗi 2 bit xảy ra trong dữ liệu nhằm tăng độ tin cậy của hệ thống.

Phần còn lại của bài báo được tổ chức như sau. Phần 2 giới thiệu thiết kế sơ đồ khối tổng quát và thiết kế chi tiết từng khối mã hóa và giải mã bên trong module ECC. Phần 3 trình bày các kết quả tổng hợp mạch và mô phỏng để đánh giá hiệu năng của module ECC được thiết kế. Cuối cùng, bài báo được kết luận trong Phần 4.

2. Thiết kế module ECC dùng mã hóa Hamming

Module ECC trong bài báo này được thiết kế sử dụng thuật toán mã hóa Hamming mở rộng $C_{Hamming}(72,64)$ bao gồm mã (71,64) Hamming và 1 bit parity mở rộng bổ sung. Việc bổ sung thêm bit parity mở rộng có tác dụng phát hiện và phân biệt lỗi 2 bit với lỗi 1 bit nhằm tăng tính chính xác cho quá trình giải mã. **Hình 1** trình bày giao diện ngõ vào/ngõ ra của một module ECC tổng quát. Khi có dữ liệu data_in được ghi vào bộ nhớ, n bit parity sẽ được tính toán tương ứng và đưa tới ngõ ra parity_out. Ở hướng ngược lại, khi đọc dữ liệu, bộ giải mã sẽ tiến hành tính toán lại giá trị parity và so sánh với parity của dữ liệu được lưu trữ. Thông qua quá trình so sánh, bộ giải mã xác định vị trí lỗi và sửa lỗi để trả về data_out là dữ liệu thông tin gốc. Trong trường hợp xảy ra lỗi không thể sửa chữa, module sẽ trả về tín hiệu error.



Hình 1. *Sơ đồ khối thể hiện giao diện ngõ vào/ngõ ra của module ECC.*

JTE
HCMUTE
ISSN: 1859-1272

JOURNAL OF TECHNOLOGY EDUCATION SCIENCE
Ho Chi Minh City of Technology and Education
Website: https://jte.hcmute.edu.vn/index.php/jte/index
Email: jte@hcmute.edu.vn

Sơ đồ khối của module ECC trong **Hình 1** bao gồm 1 bus dữ liệu vào (data_in), tín hiệu này thông qua quá trình xử lý sẽ tạo ra dữ liệu đã mã hóa gồm enc_out và parity_out. Các tín hiệu enc_out và parity_out là dữ liệu sẽ được lưu trữ tại bộ nhớ. Các chân dec_in và parity_in lần lượt là các tín hiệu dữ liệu và parity được đọc từ bộ nhớ, các tín hiệu này thông qua quá trình xử lý giải mã sẽ trả về các tín hiệu ở data_out và error. Chức năng chi tiết của các đường tín hiệu được mô tả trong **Bảng 1**.

Bảng 1. *Mô tả chân tín hiệu module ECC*

Tên chân tín hiệu	Độ rộng (bit)	Loại	Chức năng
data_in	64	Input	Dữ liệu
rst	1	Input	Reset hệ thống
clk	1	Input	Clock hệ thống
dec_in	64	Input	Dữ liệu đọc từ bộ nhớ đến bộ giải mã
parity_in	8	Input	Bộ bit kiểm tra đọc từ bộ nhớ đến bộ giải mã
enc_out	64	Output	Dữ liệu được mã hóa
parity_out	8	Output	Bộ bit kiểm tra được mã hóa
data_out	64	Output	Khôi phục dữ liệu ban đầu
error	2	Output	Trạng thái lỗi

Từ sơ đồ giao tiếp tổng quát ở phần trên, chúng tôi hình thành sơ đồ khối chi tiết ở **Hình 2** gồm 2 khối chính. Khối encoder có chức năng mã hóa dữ liệu bằng cách tiến hành tính toán các bit kiểm tra bằng phép toán logic XORing theo đúng thuật toán mã hóa Hamming [10]. Các chân tín hiệu và chức năng tương ứng của khối encoder được trình bày trong **Bảng 2**. Khối decoder có chức năng giải mã dữ liệu từ 2 tín hiệu dec_in và parity_in từ bộ nhớ. Cụ thể, khối decoder thực hiện tính toán tương tự như khối encoder và thực hiện phép toán logic XORing kết quả của quá trình trên cùng với các bit parity_in tương ứng, kết quả cho ra các dữ liệu Syndrome theo giải thuật được trình bày trong [10]. Dữ liệu syndrome sau khi được khối decoder tính toán và xử lý sẽ trả về kết quả ở các tín hiệu data_out và error với các trạng thái tương ứng gồm lỗi đơn, lỗi kép, và không có lỗi. Các chân tín hiệu và chức năng tương ứng của khối decoder được trình bày trong **Bảng 3**.

Bảng 2. *Mô tả chân tín hiệu khối encoder*

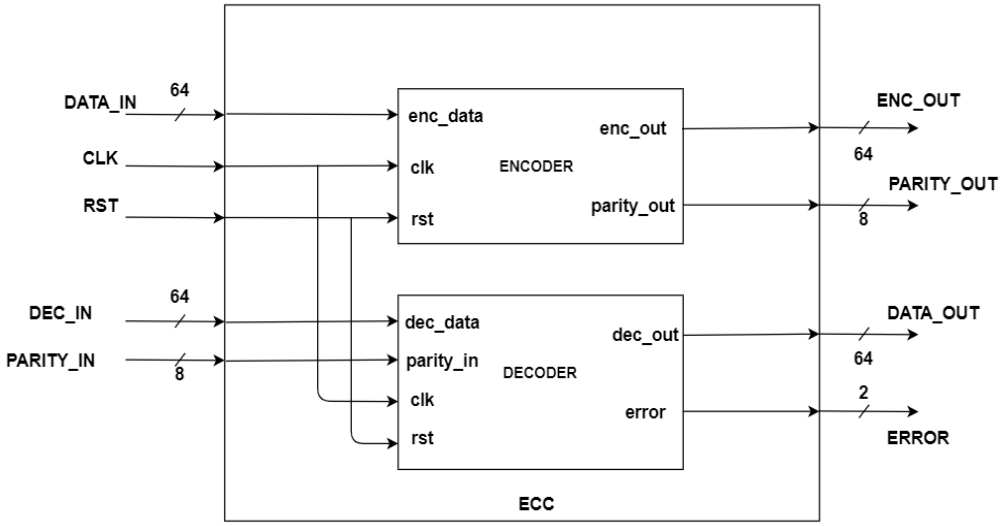
Tên chân tín hiệu	Độ rộng (bit)	Loại	Chức năng
enc_data	64	Input	Dữ liệu
rst	1	Input	Reset hệ thống
clk	1	Input	Clock hệ thống
enc_out	64	Output	Dữ liệu được mã hóa
parity_out	8	Output	Bộ bit kiểm tra

Bảng 3. *Mô tả chân tín hiệu khối decoder*

Tên chân tín hiệu	Độ rộng (bit)	Loại	Chức năng
dec_data	64	Input	Dữ liệu đọc ra từ bộ nhớ
rst	1	Input	Reset hệ thống
clk	1	Input	Clock hệ thống
dec_out	64	Output	Dữ liệu được giải mã
error	2	Output	Trạng thái lỗi của dec_data

JTE
HCMUTE
ISSN: 1859-1272

JOURNAL OF TECHNOLOGY EDUCATION SCIENCE
Ho Chi Minh City of Technology and Education
Website: https://jte.hcmute.edu.vn/index.php/jte/index
Email: jte@hcmute.edu.vn



Hình 2. *Sơ đồ khối chi tiết khối ECC*

3. Đánh giá chất lượng của thiết kế

3.1 Tài nguyên logic sử dụng

Thiết kế module ECC được tổng hợp trên phần mềm Xilinx ISE 14.7, sử dụng chip Spartan6 XC6SLX16 với tài nguyên cần sử dụng được tóm tắt trong **Bảng 4** và **Bảng 5**.

Bảng 4. *Tài nguyên cần thiết cho khối encoder*

Tài nguyên	Sử dụng	Có sẵn	Tỉ lệ (%)
Slice	43	35840	0.12
4 input LUTs	72	71680	0.1
Bonded IOBs	138	768	17
GCLKs	1	32	3

Bảng 5. *Tài nguyên cần thiết cho khối decoder*

Tài nguyên	Sử dụng	Có sẵn	Tỉ lệ (%)
Slice	91	35840	0.25
4 input LUTs	162	71680	0.23
Bonded IOBs	140	768	18
GCLKs	1	32	3

Qua khảo sát về tài nguyên logic cần sử dụng, có thể thấy rằng tài nguyên cần cho khối encoder và khối decoder chiếm tỉ lệ rất ít so với tài nguyên sẵn có của chip Spartan6 XC6SLX16, do đó thiết kế số này hoàn toàn phù hợp với các kit có tài nguyên thấp. Ngoài ra, tần số hoạt động tối đa cho bộ encoder và bộ decoder tương ứng là 952,38 MHz và 192,45 MHz.

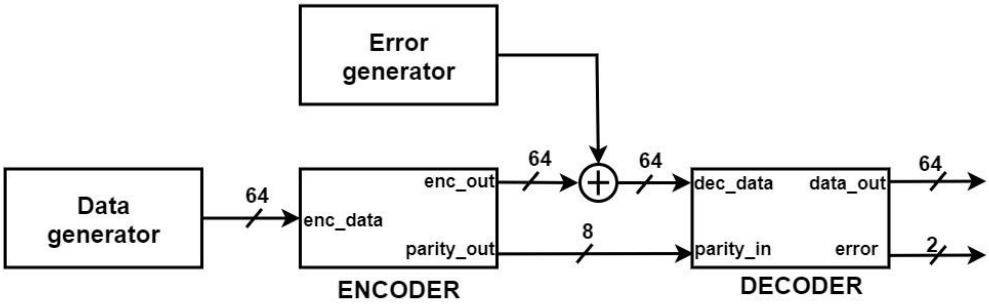
3.2 Kết quả đánh giá qua mô phỏng

3.2.1 Mô hình mô phỏng và tóm tắt testcase

Mô hình mô phỏng được minh họa trong **Hình 3** trong đó khối data generator tạo chuỗi dữ liệu kiểm tra ngẫu nhiên (bằng Matlab) sẽ đưa vào đơn vị thiết kế cần kiểm tra. Chuỗi dữ liệu kiểm tra bao gồm 32000 bit, mỗi lần ghi dữ liệu là 64 bit. Khối error generator tạo 1 lỗi ngẫu nhiên trong chuỗi 32000 bit, được XOR với ngõ ra data generator để tạo ra dữ liệu bị lỗi. **Bảng 6** và **Bảng 7** tóm tắt các testcase để đánh giá chức năng của bộ encoder và decoder được thiết kế.

JTE
HCMUTE
ISSN: 1859-1272

JOURNAL OF TECHNOLOGY EDUCATION SCIENCE
Ho Chi Minh City of Technology and Education
Website: https://jte.hcmute.edu.vn/index.php/jte/index
Email: jte@hcmute.edu.vn



Hình 3. *Sơ đồ khối mô phỏng module ECC*

Khối encoder được kiểm thử với ngõ vào có độ rộng 64 bit, ngõ ra là dữ liệu được mã hóa thành 2 đường tín hiệu: tín hiệu dữ liệu và tín hiệu bit kiểm tra. Với các bit parity_in được tạo ra, được kiểm chứng lại trên web Hamming code simulator [11].

Bảng 6. *Bảng tóm tắt các testcase bộ encoder*

Testcase	Nội dung
Testcase 1 (dữ liệu theo chuỗi)	Testcase kiểm tra 2 ngõ ra của bộ encoder là enc_out và parity_out. Chuỗi dữ liệu có độ dài 64 bit, enc_data = X"0123456789ABCDEF", lặp lại 500 lần.
Testcase 2 (dữ liệu ngẫu nhiên)	Tương tự như testcase 1 nhưng dữ liệu enc_data được thay đổi ngẫu nhiên. Chuỗi dữ liệu có độ dài 64 bit, lặp lại 500 lần.

Hệ thống khối decoder được kiểm thử với 2 ngõ vào là dữ liệu và các bit kiểm tra đã được mã hóa trước đó nhằm kiểm tra xem dữ liệu có chính xác hay không, nếu không sẽ tiến hành khắc phục lỗi.

Bảng 7. *Bảng tóm tắt các testcase bộ decoder*

Testcase	Nội dung
Testcase 3A (dữ liệu ngõ vào theo chuỗi – tương ứng với testcase 1)	Dữ liệu ngõ vào của bộ decoder: - Tương ứng với dữ liệu ngõ ra của bộ encoder như testcase 1 - Giả sử xác suất lỗi bộ nhớ là 3.2×10^{-4} có 1 bit bị lỗi
Testcase 3B (dữ liệu ngõ vào theo chuỗi – tương ứng với Testcase 1)	Dữ liệu ngõ vào của bộ decoder: - Tương ứng với dữ liệu ngõ ra của bộ encoder như testcase 1 - Giả sử xác suất lỗi bộ nhớ là 3.2×10^{-4} có 2 bit bị lỗi
Testcase 4A (dữ liệu ngõ vào ngẫu nhiên – tương ứng với Testcase 2)	Dữ liệu ngõ vào của bộ decoder: - Tương ứng với dữ liệu ngõ ra của bộ Encoder như Testcase 2 - Giả sử xác suất lỗi bộ nhớ là 3.2×10^{-4} có 1 bit bị lỗi
Testcase 4B (dữ liệu ngõ vào ngẫu nhiên – tương ứng với Testcase 2)	Dữ liệu ngõ vào của bộ decoder: - Tương ứng với dữ liệu ngõ ra của bộ Encoder như Testcase 2 - Giả sử xác suất lỗi bộ nhớ là 3.2×10^{-4} có 2 bit bị lỗi

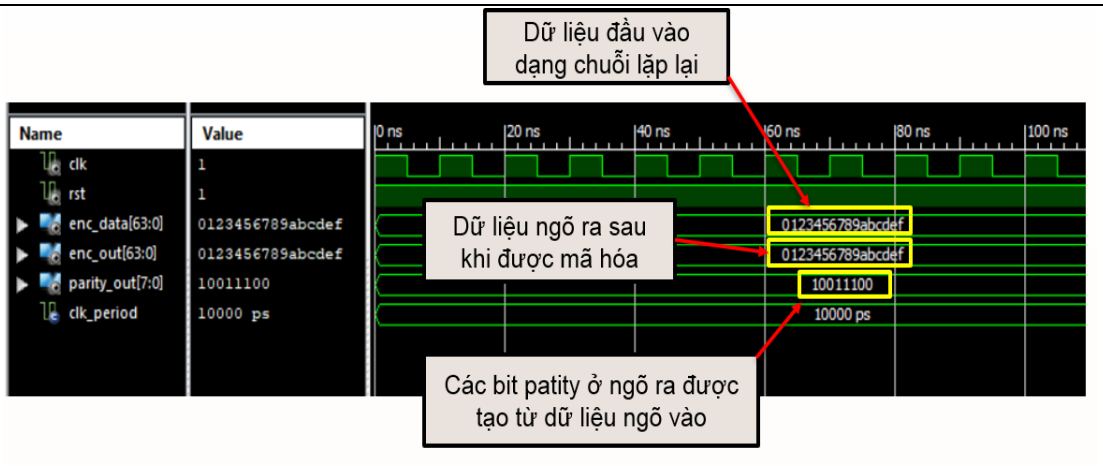
3.2.2 Phân tích kết quả mô phỏng

• Khối Encoder

Testcase 1 mô phỏng kết quả mã hóa tín hiệu đầu vào enc_data với chuỗi tuần tự, chuỗi có độ dài 64 bit và lặp lại 500 lần. Ngõ vào enc_data dạng chuỗi có độ dài 64 bit, khi có cạnh lên xung clk thì tín hiệu được mã hóa và cho ra giá trị ở ngõ ra enc_out và parity_out như thể hiện ở **Hình 4**. Giá trị parity_out hoàn toàn chính xác bằng cách kiểm tra lại trên Hamming code simulator [11].

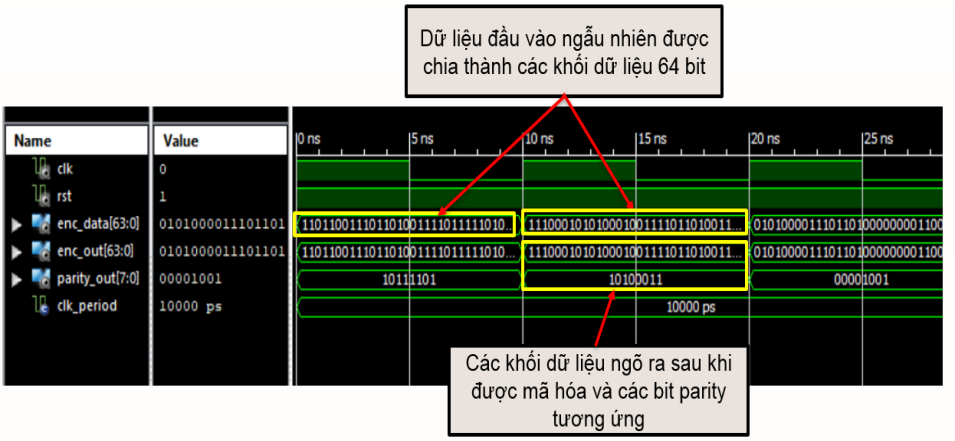
JTE
HCMUTE
ISSN: 1859-1272

JOURNAL OF TECHNOLOGY EDUCATION SCIENCE
Ho Chi Minh City of Technology and Education
Website: https://jte.hcmute.edu.vn/index.php/jte/index
Email: jte@hcmute.edu.vn



Hình 4. *Dạng sóng tín hiệu testcase 1 của bộ encoder*

Testcase 2 mô phỏng kết quả mã hóa tín hiệu đầu vào enc_data ngẫu nhiên, chuỗi có độ dài 64 bit và lặp lại 500 lần. Khi có cạnh lên xung clk thì tín hiệu được mã hóa và cho ra giá trị ở ngõ ra enc_out và parity_out như được trình bày trên **Hình 5**. Giá trị parity_out hoàn toàn chính xác bằng cách kiểm tra lại trên Hamming code simulator [11].



Hình 5. *Dạng sóng tín hiệu testcase 2 của bộ encoder*

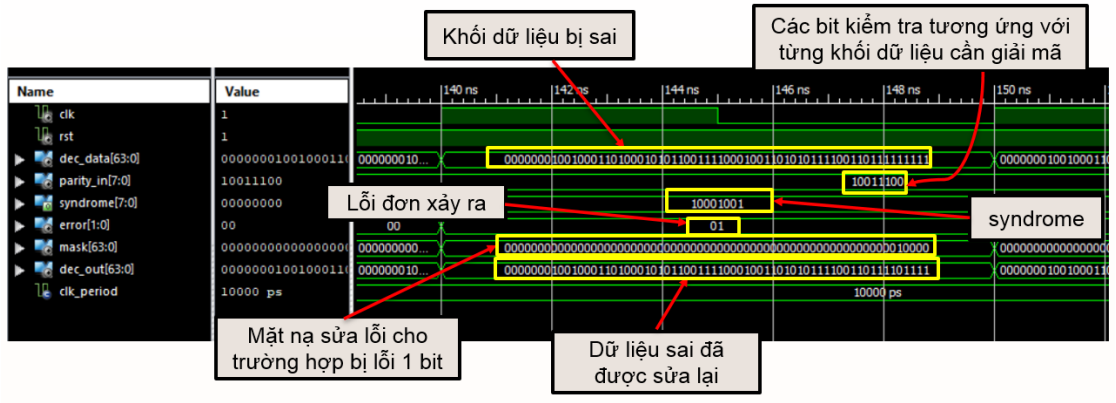
• Khối decoder

Đối với Testcase 3A, kết quả trên **Hình 6** chỉ ra tại thời điểm 140ns đến 150ns có một khối dữ liệu 64 bit đưa vào bị sai, ngõ ra dec_out đã sửa lỗi sai này và đưa tín hiệu về ban đầu. Giá trị Syndrome bao gồm 1 parity check (syndrome[7] và giá trị syndrome được tính theo mã (71,64) Hamming (syndrome[6:0]). Theo **Hình 6**, tín hiệu syndrome có giá trị "10001001", giá trị syndrome[7] ở mức 1 cho thấy xảy ra lỗi 1 bit. Bộ decoder dựa vào syndrome[6:0] để xác định vị trí xảy ra lỗi trong chuỗi dữ liệu và tạo một mặt nạ sửa lỗi cho chuỗi dữ liệu này và bộ corrector sẽ dựa vào mặt nạ này để sửa chuỗi dữ liệu. Ngõ ra error dựa vào syndrome sẽ trả về trạng thái tương ứng là '01' (xảy ra lỗi đơn).

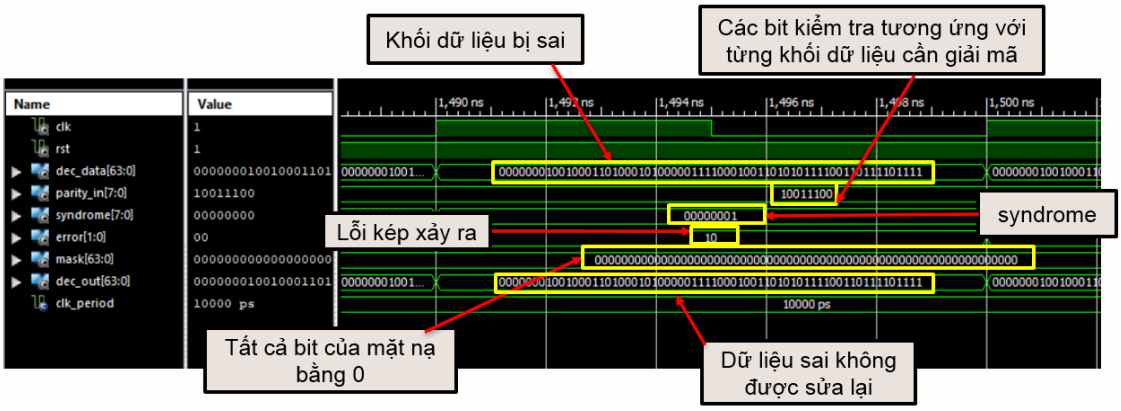
Với Testcase 3B, trong trường hợp này tín hiệu ngõ vào có 2 lỗi ngẫu nhiên trong khối dữ liệu. Quan sát kết quả trên **Hình 7**, tại thời điểm 1490ns đến 1500ns có một khối dữ liệu 64 bit đưa vào bị sai, ngõ ra dec_out không sửa được được do module chỉ có thể phát hiện và sửa lỗi 1 bit còn trường hợp lỗi 2 bit không thể sửa. Giá trị syndrome[6:0] khác 0 chứng tỏ xảy ra lỗi nhưng parity bit syndrome[7] bằng 0 do đó suy ra đã xảy ra lỗi kép. Ngõ ra error lúc này có giá trị '10' tương ứng với trạng thái xảy ra lỗi nhiều bit. Do mã Hamming được thiết kế chỉ có thể sửa được lỗi đơn, do đó bộ decoder sẽ tạo ra mặt nạ 0 và dữ liệu không được sửa chữa. Kết quả testcase 3B được trình bày ở **Hình 7** phù hợp với lý thuyết đã trình bày. Trong trường hợp testcase 3B, khối dữ liệu chỉ được phát hiện lỗi và lỗi không được sửa.

JTE
HCMUTE
ISSN: 1859-1272

JOURNAL OF TECHNOLOGY EDUCATION SCIENCE
Ho Chi Minh City of Technology and Education
Website: https://jte.hcmute.edu.vn/index.php/jte/index
Email: jte@hcmute.edu.vn



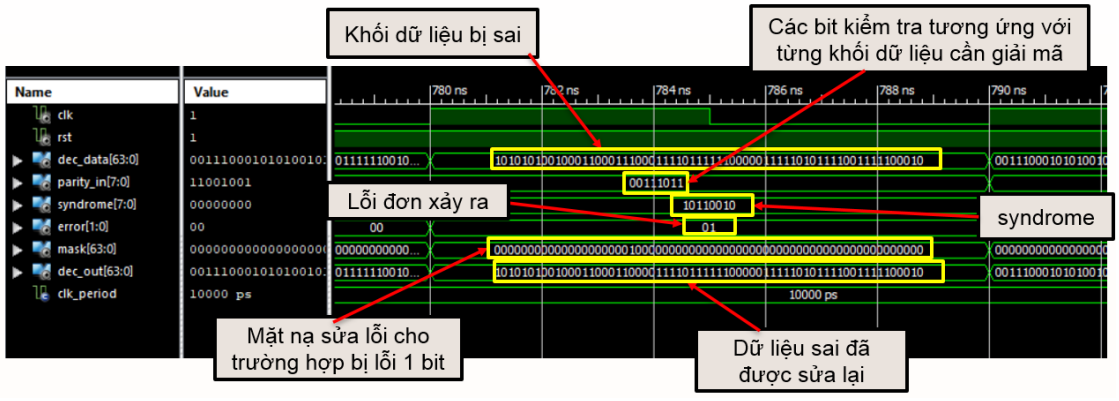
Hình 6. *Dạng sóng tín hiệu testcase 3A của bộ decoder*



Hình 7. *Dạng sóng tín hiệu testcase 3B của bộ decoder*

Trong Testcaes 4A, tín hiệu ngõ vào có 1 lỗi ngẫu nhiên trong khối dữ liệu. Quan sát kết quả trên **Hình 8** ta có thể thấy, tại thời điểm 780ns đến 790ns có một khối dữ liệu 64 bit đưa vào bị sai, ngõ ra dec_out đã sửa lỗi sai này và dữ liệu data_out đã được khôi phục như ban đầu. Tương tự như testcase 3A, ở testcase này xảy ra lỗi đơn và đồng thời được phát hiện, sửa chữa chính xác.

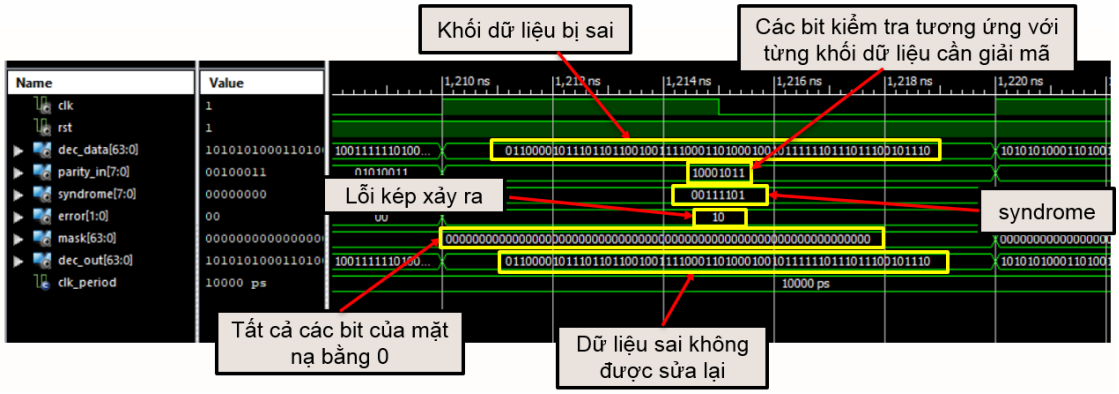
Trong Testcase 4B, tín hiệu ngõ vào có 2 lỗi ngẫu nhiên trong khối dữ liệu. Như thể hiện trên **Hình 9**, tại thời điểm 1210ns đến 1220ns có một khối dữ liệu 64 bit đưa vào bị sai, ngõ ra dec_out không sửa được do module chỉ có thể phát hiện và sửa lỗi 1 bit còn trường hợp lỗi 2 bit không sửa được. Kết quả mô phỏng cho thấy module giải mã hoạt động theo đúng thiết kế đã xác định.



Hình 8. *Dạng sóng tín hiệu testcase 4A của bộ decoder*

JTE
HCMUTE
ISSN: 1859-1272

JOURNAL OF TECHNOLOGY EDUCATION SCIENCE
Ho Chi Minh City of Technology and Education
Website: https://jte.hcmute.edu.vn/index.php/jte/index
Email: jte@hcmute.edu.vn



Hình 9. *Dạng sóng tín hiệu testcase 4B của bộ decoder*

Kết quả mô phỏng so với lý thuyết chỉ ra rằng module ECC dựa trên mã Hamming được thiết kế hoạt động chính xác, đáp ứng được các yêu cầu thiết kế. Bảng so sánh kết quả giữa mô phỏng và lý thuyết được tóm tắt trong **Bảng 8**.

Bảng 8. *Bảng so sánh kết quả giữa mô phỏng và lý thuyết*

Testcase	Lý thuyết	Thực tế	Đánh giá
3A	Phát hiện lỗi đơn và sửa lỗi đơn	Phát hiện lỗi đơn và sửa lỗi đơn	Chính xác
3B	Phát hiện lỗi kép nhưng không sửa được	Phát hiện lỗi kép nhưng không sửa được	Chính xác
4A	Phát hiện lỗi đơn và sửa lỗi đơn	Phát hiện lỗi đơn và sửa lỗi đơn	Chính xác
4B	Phát hiện lỗi kép nhưng không sửa được	Phát hiện lỗi kép nhưng không sửa được	Chính xác

Bằng cách thực hiện lặp lại mỗi Testcase 10 lần với lỗi bit được tạo ra tại các vị trí ngẫu nhiên với xác suất lỗi nhất định như được tóm tắt ở Bảng 7, chúng tôi rút ra được tỉ lệ phát hiện lỗi của module ECC được thiết kế ở **Bảng 9** dưới đây.

Bảng 9. *Bảng đánh giá kết quả mô phỏng*

Testcase	Tỉ lệ phát hiện lỗi	Tỉ lệ sửa lỗi
3A (1 lỗi)	100%	100%
3B (2 lỗi)	100%	0%
4A (1 lỗi)	100%	100%
4B (2 lỗi)	100%	0%

3.3 Đánh giá công suất tiêu thụ

Để đánh giá công suất tiêu thụ, chúng tôi đã thực hiện tổng hợp công suất tiêu thụ với nhiều tần số khác nhau trên phần mềm Xilinx. **Bảng 10** dưới đây mô tả công suất tiêu thụ của bộ encoder với các tần số lần lượt là 50, 100 và 150Mhz. Công suất tiêu thụ của mạch bằng tổng công suất động dynamic và công suất tĩnh quiescent.

Bảng 10. *Công suất tiêu thụ của bộ encoder*

Tần số hoạt động (MHz)	Total (W)	Dynamic (W)	Quiescent (W)
50	1.015	0.027	0.988
100	1.019	0.031	0.988
150	1.024	0.036	0.988

Bảng 11 dưới đây mô tả công suất tiêu thụ của bộ decoder với các tần số lần lượt là 50, 100 và 150Mhz.

Bảng 11. *Công suất tiêu thụ của bộ decoder*

Tần số hoạt động (MHz)	Total (W)	Dynamic (W)	Quiescent (W)
50	1.063	0.030	1.034
100	1.068	0.034	1.034
150	1.073	0.039	1.034

Qua khảo sát có thể thấy rằng, tần số hoạt động đã tác động đến công suất tiêu thụ của mạch. Khi tần số hoạt động tăng lên, làm cho công suất tiêu thụ cũng tăng lên.

4. Kết luận

Trong bài báo này, chúng tôi đã trình bày chi tiết thiết kế module ECC có độ rộng bus dữ liệu 64 bit nhằm tăng độ tin cậy cho truy suất bộ nhớ. Module ECC được thiết kế có thể phát hiện và sửa được lỗi bit đơn, phát hiện được lỗi 2 bit. Thiết kế module ECC này có thể được phát triển thành một IP để có thể tích hợp vào các hệ thống số lớn hơn trong quá trình thiết kế vi mạch. Bên cạnh đó, hầu hết các thiết kế vi mạch ít được trình bày chi tiết và công khai, thiết kế module ECC trong bài báo này có thể được xem như tài liệu tham khảo cho việc học tập các môn liên quan tới thiết kế vi mạch cũng như lý thuyết mã hóa trong truyền dữ liệu.

Lời cám ơn

Bài báo này được tài trợ kinh phí nghiên cứu năm 2022 bởi trường Đại Học Sư phạm Kỹ thuật Thành Phố Hồ Chí Minh (mã số đề tài SV2022-28).

TÀI LIỆU THAM KHẢO

1. H. Kwon, K. Kim, D. Jeon and K.-S. Chung, “Reducing Refresh Overhead with In-DRAM Error Correction Codes”, 18th International SoC Design Conference (ISOCC), 2021, pp. 211-214.
2. S. LIU, P. Reviriego, J. Guo, J. HAN and F. Lombardi, “Exploiting Asymmetry in eDRAM Errors for Redundancy-Free Error-Tolerant Design”, IEEE Transactions on Emerging Topics in Computing, vol. 9, no. 4, pp. 2064-2075, 1 Oct.-Dec. 2021.
3. K. Lavery, “Discriminating Between Soft Errors and Hard Errors in RAM”, SPNA109, 2008.
4. S. Mueller, “Upgrading and Repairing PCs: Upgrading and Repairing”, Que Publishing, 2015.
5. U. S. Sani and I. H. Shanono, “Design of (7, 4) Hamming Encoder and Decoder Using VHDL”, 1st International Engineering Conference, 2015.
6. D. Mokara, S. Naidu and A. K. Gupta, “Design and Implementation of Hamming Code using VHDL & DSCH”, International Journal of Latest Engineering Research and Applications, vol. 02, pp. 33-40, 2017.
7. A. H. Saleh, “Design of Hamming Code for 64 bit single Error Detection and Correction using VHDL”, Diyala Journal of Engineering Sciences, vol. 08, no. 03, pp. 22-37, 2015.
8. H. Sharma and A. Kumar, “Hamming Code for Error Detection and Corection using VHDL”, International Journal Of Engineering Research & Management Technology, vol. 01, 2014.
9. T. Zhang and Q. Ding, Design of (15, 11) Hamming Code Encoding and Decoding System Based on FPGA, Heilongjiang University, 2011.
10. A. B. Forouzan, A. C. Coombs and S. C. Fegan, Data Communications and Networking, McGraw-Hill, 2001.
11. J.Doyle, “Hamming Code Simulator”, University of Massachusetts, 1986. [Online]. Available: http://www.ecs.umass.edu/ece/koren/FaultTolerantSystems/simulator/Hamming/HammingCodes.html.

Tran Do Hon Nhien is currently a student at the Ho Chi Minh City University of Technology and Education (HCMUTE), Vietnam. His main research interests include wireless communication networks and FPGA-based designs for DSP applications.

Vo Tan Thanh received his B.S degree from Ho Chi Minh City University of Technology and Education (HCMUTE), Vietnam, in 2022. His main research interests include wireless communication networks and FPGA-based designs for DSP applications.

Nguyen Thanh Khoa received his B.S degree from Ho Chi Minh City University of Technology and Education (HCMUTE), Vietnam, in 2022. His main research interests include wireless communication networks and FPGA-based designs for DSP applications.

Nguyen Quoc Thang is currently a student at the Ho Chi Minh City University of Technology and Education (HCMUTE), Vietnam. His main research interests include wireless communication networks and FPGA-based designs for DSP applications.

Nguyen Van Thanh Loc received his B.S degree from Ho Chi Minh City University of Technology and Education (HCMUTE), Vietnam, in 2020. His main research interests include communication networks and applications of error-control coding for wireless communications.

Huynh Hoang Ha received his MEng degree from Ho Chi Minh City University of Technology and Education (HCMUTE), Vietnam, in 2015. His main research interests include Embedded system and IoT system.

Nguyen Ngo Lam is currently a lecturer at the Faculty For High Quality Training, Ho Chi Minh City University of Technology and Education . He received his Bachelor and Master degree in radio and electronics engineering from the Ho Chi Minh City University of Technology, Vietnam in 2000 and 2004 respectively. His research interests include wireless communication, data communication, digital signal processing.

Do Duy Tan received his B.S. degree from Ho Chi Minh City University of Technology (HCMUT), Vietnam, and M.S. degree from Kumoh National Institute of Technology, Korea, in 2010 and 2013, respectively. He received his Ph.D. degree from Autonomous University of Barcelona, Spain, in 2019. He is currently with the Department of Computer and Communication Engineering, Ho Chi Minh City University of Technology and Education (HCMUTE) in Vietnam as an Assistant Professor. His main research interests include real-time optimisation for resource allocation in wireless networks and coding applications for wireless communications.