**ĐẠI HỌC QUỐC GIA TP. HCM**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

****

**BÀI TẬP LỚN MÔN HỌC**

**MÔN HỌC: XỬ LÍ TÍN HIỆU SỐ VỚI FPGA**

**ĐỀ TÀI**

**NÉN DỮ LIỆU KHÔNG MẤT DỮ LIỆU BẰNG HUFFMAN CODING**

**LỚP: L01 NHÓM: 01**

**HK222**

**GVHD: Trịnh Vũ Đăng Nguyên**

**SINH VIÊN THỰC HIỆN**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **HỌ** | **TÊN** | **ĐIỂM BTL** | **GHI CHÚ** |
| 1 | 2012689 | Diệp Thái | Bình |  | Thành viên |
| 2 | 2012624 | Nguyễn Hoàng | Ái |  | Thành viên |

**­­­­­­Ngày hoàn thành: 26/04/2023**

**Ngày hạn nộp: …/…/2023**

**TP. HỒ CHÍ MINH, NĂM 2023**

**MỤC LỤC**

[**BẢNG ĐIỂM CHẤM CHÉO** 2](#_Toc133434947)

[**MÔ TẢ CÔNG VIỆC THỰC HIỆN** 2](#_Toc133434948)

[**I – PHẦN MỞ ĐẦU** 3](#_Toc133434949)

[**1. Lý do chọn đề tài** 3](#_Toc133434950)

[**2. Nhiệm vụ của đề tài** 4](#_Toc133434951)

[**II – PHẦN NỘI DUNG** 5](#_Toc133434952)

[**CHƯƠNG 1: CƠ SỞ LÝ THUYẾT** 5](#_Toc133434953)

[**1. Khái niệm về nén dữ liệu không mất mát** 5](#_Toc133434954)

[**2. Khái niệm về huffman coding** 5](#_Toc133434955)

[**3. Ứng dụng của huffman coding trong nén dữ liệu không mất mát** 6](#_Toc133434956)

[**CHƯƠNG 2: MÔ TẢ THUẬT TOÁN** 8](#_Toc133434957)

[**1. Mô hình tổng quan của thuật toán nén dữ liệu không mất mát bằng huffman coding** 8](#_Toc133434958)

[**2. Mô tả chi tiết từng bước của thuật toán** 8](#_Toc133434959)

[**3. Mô tả chi tiết thuật toán giải mã dữ liệu đã nén bằng huffman coding** 9](#_Toc133434960)

[**CHƯƠNG 3: QUY TRÌNH THIẾT KẾ** 11](#_Toc133434961)

[**1. Giới thiệu tổng quát về phần mềm và ngôn ngữ sử dụng** 11](#_Toc133434962)

[**2. Thiết kế cấp độ hệ thống (Specification)** 11](#_Toc133434963)

[**3. Thiết kế cấp độ RTL (Register Transfer Level)** 12](#_Toc133434964)

[**4. Thực hiện mô phỏng (Simulation)** 12](#_Toc133434965)

[**5. Thực hiện tổng hợp (Synthesis)** 12](#_Toc133434966)

[**CHƯƠNG 4: KẾT QUẢ THỰC TẾ** 13](#_Toc133434967)

[**III – PHẦN KẾT LUẬN** 14](#_Toc133434968)

[**IV – TÀI LIỆU THAM KHẢO** 14](#_Toc133434969)

[**V – PHỤ LỤC** 14](#_Toc133434970)

# **BẢNG ĐIỂM CHẤM CHÉO**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **HỌ** | **TÊN** | **TV1** | **TV2** |
| 1 | 2012689 | Diệp Thái | Bình | 10 | 10 |
| 2 | 2012624 | Nguyễn Hoàng | Ái | 10 | 10 |

# **MÔ TẢ CÔNG VIỆC THỰC HIỆN**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **MSSV** | **HỌ** | **TÊN** | **CÔNG VIỆC ĐÃ THỰC HIỆN** |
| 1 | 2012689 | Diệp Thái | Bình | Thực hiện báo cáo và viết chương trình |
| 2 | 2012624 | Nguyễn Hoàng | Ái | Thực hiện viết code đổ lên kit DE2 |

# **I – PHẦN MỞ ĐẦU**

## **1. Lý do chọn đề tài**

Nén dữ liệu không mất mát là một lĩnh vực quan trọng trong khoa học máy tính và kỹ thuật điện tử, có ứng dụng rộng rãi trong các lĩnh vực như truyền thông, lưu trữ, xử lý ảnh, âm thanh, video… Nén dữ liệu không mất mát là quá trình giảm dung lượng của dữ liệu mà không làm mất thông tin gốc của dữ liệu. Mục tiêu của nén dữ liệu không mất mát là tối ưu hóa tỉ lệ nén (tỉ lệ giữa dung lượng dữ liệu gốc và dữ liệu đã nén) và thời gian nén và giải nén.

Huffman coding là một trong những phương pháp nén dữ liệu không mất mát phổ biến và hiệu quả nhất. Huffman coding là một kỹ thuật mã hóa biến độ dài (variable-length encoding), sử dụng cây nhị phân để tạo ra các mã nhị phân cho các ký tự trong dữ liệu dựa trên tần số xuất hiện của chúng. Các ký tự có tần số xuất hiện cao sẽ có mã ngắn hơn, và ngược lại.

Huffman coding có nhiều ưu điểm so với các phương pháp nén dữ liệu không mất mát khác, như run-length encoding hay Lempel-Ziv-Welch. Một số ưu điểm của huffman coding là:

Huffman coding đạt được tỉ lệ nén cao hơn, vì nó sử dụng các mã biến độ dài thích ứng với tần số xuất hiện của các ký tự, trong khi các phương pháp khác sử dụng các mã cố định hoặc phụ thuộc vào ngữ cảnh.

Huffman coding có thể áp dụng cho nhiều loại dữ liệu khác nhau, như văn bản, hình ảnh, âm thanh, video… chỉ cần biết được tần số xuất hiện của các ký tự trong dữ liệu.

Huffman coding có thuật toán đơn giản và hiệu quả, chỉ cần xây dựng cây huffman từ tần số xuất hiện của các ký tự và mã hóa và giải mã theo cây huffman. Thuật toán có độ phức tạp O(nlogn) với n là số lượng ký tự khác nhau trong dữ liệu.

Vì những lý do trên, nhóm 1 đã chọn đề tài bài tập lớn về thuật toán nén dữ liệu không mất mát bằng huffman coding. Nhóm mong muốn khám phá thêm về nguyên lý hoạt động, thuật toán và ứng dụng của huffman coding trong thực tế.

## **2. Nhiệm vụ của đề tài**

*Mục tiêu của đề tài* là nghiên cứu và thực hiện thuật toán nén dữ liệu không mất mát bằng huffman coding trên các loại dữ liệu khác nhau (văn bản, hình ảnh, âm thanh, video…).

*Phạm vi của đề tài bao gồm*:

- Tìm hiểu cơ sở lý thuyết về nén dữ liệu không mất mát và huffman coding

- Mô phỏng thuật toán nén dữ liệu không mất mát bằng huffman coding bằng ngôn ngữ phần cứng SystemVerilog.

- Tổng hợp mạch thuật toán nén dữ liệu không mất mát bằng huffman coding bằng ngôn ngữ miêu tả phần cứng SystemVerilog.

- Thực hiện thực tế thuật toán nén dữ liệu không mất mát bằng huffman coding trên kit DE2 và DE10.

*Phương pháp nghiên cứu và thiết kế bao gồm*:

- Tìm kiếm và tổng hợp các tài liệu tham khảo liên quan đến đề tài từ các nguồn uy tín.

- Phân tích và so sánh các phương pháp nén dữ liệu không mất mát khác nhau.

- Thiết kế và cài đặt thuật toán nén dữ liệu không mất mát bằng huffman coding trên các môi trường và công cụ khác nhau.

- Kiểm tra và đánh giá kết quả nén dữ liệu không mất mát bằng huffman coding theo các tiêu chí khác nhau.

*Cấu trúc báo cáo gồm có*:

- Phần mở đầu: Giới thiệu về đề tài, lý do chọn đề tài, nhiệm vụ của đề tài.

- Phần nội dung: Trình bày chi tiết về cơ sở lý thuyết, thuật toán và kết quả thực hiện của đề tài.

- Phần kết luận: Tổng kết lại những kết quả đã đạt được, nhận xét về ưu nhược điểm của đề tài, đề xuất hướng phát triển tiếp theo của đề tài.

- Tài liệu tham khảo: Liệt kê các nguồn tham khảo đã sử dụng trong quá trình nghiên cứu và viết báo cáo.

- Phụ lục: Bổ sung các thông tin phụ trợ cho báo cáo, như mã nguồn, sơ đồ mạch, hình ảnh minh họa…

# **II – PHẦN NỘI DUNG**

## **CHƯƠNG 1: CƠ SỞ LÝ THUYẾT**

### **1. Khái niệm về nén dữ liệu không mất mát**

[Nén dữ liệu không mất mát (lossless data compression) là quá trình giảm dung lượng của dữ liệu mà không làm mất thông tin gốc của dữ liệu](https://www.adobe.com/uk/creativecloud/photography/discover/lossless-compression.html). Nén dữ liệu không mất mát sử dụng một thuật toán để thu gọn dữ liệu mà không bỏ đi bất kỳ thông tin quan trọng nào. Cơ bản, nó viết lại file để làm cho nó hiệu quả hơn - dẫn đến kích thước lưu trữ hoặc truyền tải nhỏ hơn. [Một cách mà thuật toán làm điều này là bằng cách thay thế các thông tin không cần thiết và lưu trữ chúng trong một file chỉ mục](https://www.adobe.com/uk/creativecloud/photography/discover/lossless-compression.html).

Nén dữ liệu không mất mát có thể được phân loại thành hai loại chính: nén dữ liệu không mất mát theo từng bit (bit-level lossless data compression) và nén dữ liệu không mất mát theo từng byte (byte-level lossless data compression). Nén dữ liệu không mất mát theo từng bit là quá trình giảm số lượng bit cần thiết để biểu diễn dữ liệu, bằng cách sử dụng các mã biến độ dài (variable-length codes) cho các bit trong dữ liệu. Nén dữ liệu không mất mát theo từng byte là quá trình giảm số lượng byte cần thiết để biểu diễn dữ liệu, bằng cách sử dụng các kỹ thuật như mã hóa từ điển (dictionary encoding), mã hóa ngữ cảnh (context encoding), mã hóa chuyển đổi (transform encoding)…

Nén dữ liệu không mất mát có nhiều ứng dụng trong các lĩnh vực như truyền thông, lưu trữ, xử lý ảnh, âm thanh, video… Nén dữ liệu không mất mát giúp tiết kiệm băng thông, dung lượng lưu trữ, thời gian truyền tải và xử lý của dữ liệu. Nén dữ liệu không mất mát cũng đảm bảo tính toàn vẹn và chính xác của dữ liệu sau khi nén và giải nén.

### **2. Khái niệm về huffman coding**

#### 2.1. Nguyên lý hoạt động của huffman coding

[Huffman coding là một phương pháp nén dữ liệu không mất mát theo từng bit, sử dụng các mã biến độ dài (variable-length codes) cho các ký tự trong dữ liệu dựa trên tần số xuất hiện của chúng](https://www.programiz.com/dsa/huffman-coding). Huffman coding được đặt tên theo David A. Huffman, người đã đề xuất phương pháp này vào năm 1952.

Nguyên lý hoạt động của huffman coding là gán các mã ngắn hơn cho các ký tự có tần số xuất hiện cao hơn, và ngược lại, gán các mã dài hơn cho các ký tự có tần số xuất hiện thấp hơn. Như vậy, huffman coding giảm được số lượng bit cần thiết để biểu diễn dữ liệu, từ đó giảm được dung lượng của dữ liệu.

#### 2.2. Thuật toán xây dựng cây huffman

Thuật toán xây dựng cây huffman (huffman tree) là thuật toán quan trọng nhất trong huffman coding. Cây huffman là một cây nhị phân đặc biệt, mà mỗi nút lá chứa một ký tự và tần số xuất hiện của nó, và mỗi nút nội chứa tổng tần số xuất hiện của hai nút con của nó. Cây huffman được xây dựng từ dưới lên, bằng cách lặp lại các bước sau:

- Chọn hai nút có tần số xuất hiện nhỏ nhất trong tập các nút chưa được xử lý.

- Tạo một nút mới là cha của hai nút đã chọn, với tần số xuất hiện bằng tổng tần số xuất hiện của hai nút con.

- Thêm nút mới vào tập các nút chưa được xử lý và loại bỏ hai nút đã chọn khỏi tập này.

- Lặp lại cho đến khi chỉ còn duy nhất một nút trong tập các nút chưa được xử lý. Nút này là gốc của cây huffman.

#### 2.3. Thuật toán mã hóa và giải mã huffman

Thuật toán mã hóa và giải mã huffman là thuật toán sử dụng cây huffman để tạo ra các mã nhị phân cho các ký tự trong dữ liệu và ngược lại. Thuật toán mã hóa và giải mã huffman hoạt động như sau:

- Thuật toán mã hóa: Duyệt cây huffman từ gốc đến lá, gán bit 0 cho cạnh trái và bit 1 cho cạnh phải. Khi đến một nút lá, ghi nhận mã nhị phân cho ký tự tương ứng với nút lá đó. Lặp lại cho đến khi duyệt hết cây huffman. Sau đó, thay thế các ký tự trong dữ liệu bằng mã nhị phân đã tạo ra.

- Thuật toán giải mã: Duyệt cây huffman từ gốc đến lá, theo cách ngược lại với thuật toán mã hóa. Khi gặp bit 0, đi sang cạnh trái, khi gặp bit 1, đi sang cạnh phải. Khi đến một nút lá, ghi nhận ký tự tương ứng với nút lá đó. Lặp lại cho đến khi duyệt hết dữ liệu đã mã hóa. Sau đó, ghép các ký tự đã ghi nhận lại thành dữ liệu gốc.

### **3. Ứng dụng của huffman coding trong nén dữ liệu không mất mát**

#### 3.1. Các loại dữ liệu có thể áp dụng huffman coding

Huffman coding có nhiều ứng dụng trong các lĩnh vực như truyền thông, lưu trữ, xử lý ảnh, âm thanh, video… Một số ứng dụng cụ thể của huffman coding là:

- Truyền thông: Huffman coding được sử dụng để nén dữ liệu trước khi truyền tải qua các kênh truyền thông như internet, điện thoại, vệ tinh… để tiết kiệm băng thông và thời gian truyền tải. Ví dụ, giao thức HTTP/2 sử dụng huffman coding để nén các tiêu đề HTTP.

- Lưu trữ: Huffman coding được sử dụng để nén dữ liệu trước khi lưu trữ vào các thiết bị như đĩa cứng, USB, thẻ nhớ… để tiết kiệm dung lượng lưu trữ và tăng hiệu suất đọc ghi. Ví dụ, định dạng file ZIP sử dụng huffman coding để nén các file trong nó1.

- Xử lý ảnh: Huffman coding được sử dụng để nén các ảnh số mà không làm mất chất lượng của chúng. Ví dụ, định dạng file PNG sử dụng huffman coding để nén các pixel trong ảnh.

- Xử lý âm thanh: Huffman coding được sử dụng để nén các âm thanh số mà không làm mất chất lượng của chúng. Ví dụ, định dạng file FLAC sử dụng huffman coding để nén các mẫu âm thanh trong file.

- Xử lý video: Huffman coding được sử dụng để nén các video số mà không làm mất chất lượng của chúng. Ví dụ, định dạng file MP4 sử dụng huffman coding để nén các khung hình trong video.

#### 3.2. Các ưu điểm và nhược điểm của huffman coding so với các phương pháp khác

*Các ưu điểm của huffman coding* so với các phương pháp khác là:

- [Huffman coding đạt được tỉ lệ nén cao hơn, vì nó sử dụng các mã biến độ dài thích ứng với tần số xuất hiện của các ký tự, trong khi các phương pháp khác sử dụng các mã cố định hoặc phụ thuộc vào ngữ cảnh](https://iq.opengenus.org/advantages-and-disadvantages-of-huffman-coding/).

- [Huffman coding có thể áp dụng cho nhiều loại dữ liệu khác nhau, như văn bản, hình ảnh, âm thanh, video… chỉ cần biết được tần số xuất hiện của các ký tự trong dữ liệu](https://iq.opengenus.org/advantages-and-disadvantages-of-huffman-coding/).

- Huffman coding có thuật toán đơn giản và hiệu quả, chỉ cần xây dựng cây huffman từ tần số xuất hiện của các ký tự và mã hóa và giải mã theo cây huffman. [Thuật toán có độ phức tạp O(nlogn) với n là số lượng ký tự khác nhau trong dữ liệu](https://iq.opengenus.org/advantages-and-disadvantages-of-huffman-coding/).

*Các nhược điểm của huffman coding* so với các phương pháp khác là:

- [Huffman coding yêu cầu biết trước tần số xuất hiện của các ký tự trong dữ liệu, làm cho nó ít phù hợp cho các tình huống mà phân bố của các ký tự không được biết trước hoặc thay đổi động](https://iq.opengenus.org/advantages-and-disadvantages-of-huffman-coding/).

- [Cây huffman có thể phức tạp và khó hiểu, làm cho nó khó để gỡ lỗi và bảo trì mã nguồn](https://iq.opengenus.org/advantages-and-disadvantages-of-huffman-coding/).

- Huffman coding không thể đạt được entropy (tối thiểu số bit cần thiết để biểu diễn dữ liệu) khi mỗi ký tự có một xác suất xuất hiện không phải là một lũy thừa của 2. [Trong trường hợp này, các phương pháp khác như mã hóa số (arithmetic coding) có thể đạt được kết quả tốt hơn](https://iq.opengenus.org/advantages-and-disadvantages-of-huffman-coding/)

## **CHƯƠNG 2: MÔ TẢ THUẬT TOÁN**

### **1. Mô hình tổng quan của thuật toán nén dữ liệu không mất mát bằng huffman coding**

#### 1.1. Các bước chính của thuật toán

*Mô tả chi tiết từng bước của thuật toán*

- Bước 1: Tính tần số xuất hiện của các ký tự trong dữ liệu đầu vào.

- Bước 2: Xây dựng cây huffman từ các ký tự và tần số xuất hiện của chúng.

- Bước 3: Tạo bảng mã hóa từ cây huffman.

- Bước 4: Mã hóa dữ liệu đầu vào theo bảng mã hóa đã tạo.

- Bước 5: Lưu trữ dữ liệu đã mã hóa và cây huffman vào file đầu ra.

*Mô tả chi tiết thuật toán giải mã dữ liệu đã nén bằng huffman coding*

- Bước 1: Đọc file đầu vào và lấy ra cây huffman và dữ liệu đã mã hóa.

- Bước 2: Giải mã dữ liệu đã mã hóa theo cây huffman đã lấy ra.

- Bước 3: Lưu trữ dữ liệu đã giải mã vào file đầu ra.

#### 1.2. Các thành phần cơ bản của thuật toán

*Chuỗi đầu vào*: là chuỗi các ký tự hoặc bit cần nén.

*Bảng tần suất*: là bảng liệt kê số lần xuất hiện của mỗi phần tử trong chuỗi đầu vào.

*Cây Huffman*: là cây nhị phân được xây dựng từ bảng tần suất, sao cho các phần tử có tần suất cao nằm gần gốc và có mã hóa ngắn, còn các phần tử có tần suất thấp nằm xa gốc và có mã hóa dài.

*Chuỗi mã hóa*: là chuỗi các bit được tạo ra từ chuỗi đầu vào bằng cách thay thế mỗi phần tử bằng giá trị mã hóa tương ứng trên cây Huffman.

### **2.** **Mô tả chi tiết từng bước của thuật toán**

*Bước 1: Tính tần số xuất hiện của các ký tự trong dữ liệu đầu vào*

Ta có thể sử dụng một mảng hoặc một bảng để lưu trữ số lần xuất hiện của mỗi ký tự trong dữ liệu đầu vào. Ví dụ, nếu dữ liệu đầu vào là “abracadabra”, ta có thể tạo một bảng tần suất như sau:

|  |  |
| --- | --- |
| Ký tự | Tần số |
| a | 5 |
| b | 2 |
| c | 1 |
| d | 1 |
| r | 2 |

*Bước 2: Xây dựng cây huffman từ các ký tự và tần số xuất hiện của chúng*

Ta có thể sử dụng một hàng đợi ưu tiên để lưu trữ các nút của cây huffman, sao cho nút có tần suất nhỏ nhất luôn nằm đầu hàng đợi. Ban đầu, ta tạo một nút cho mỗi ký tự và đưa vào hàng đợi ưu tiên. Sau đó, ta lặp lại quá trình sau cho đến khi chỉ còn một nút trong hàng đợi:

- Lấy ra hai nút có tần suất nhỏ nhất khỏi hàng đợi.

- Tạo một nút mới có tần suất bằng tổng tần suất của hai nút vừa lấy ra, và gán hai nút này làm con trái và con phải của nút mới.

- Đưa nút mới vào hàng đợi.

*Bước 3: Tạo bảng mã hóa từ cây huffman*

Ta có thể duyệt cây huffman từ gốc đến lá, và gán giá trị mã hóa cho mỗi ký tự theo quy tắc sau: nếu đi sang con trái, thêm bit 0 vào mã hóa; nếu đi sang con phải, thêm bit 1 vào mã hóa. Ví dụ, với cây huffman trên, ta có thể tạo bảng mã hóa như sau:

|  |  |
| --- | --- |
| Ký tự | Mã hóa |
| a | 0 |
| b | 101 |
| c | 1000 |
| d | 1001 |
| r | 11 |

*Bước 4: Mã hóa dữ liệu đầu vào theo bảng mã hóa đã tạo*

Mã hóa dữ liệu đầu vào theo bảng mã hóa đã tạo. Ta có thể duyệt qua từng ký tự trong dữ liệu đầu vào, và thay thế mỗi ký tự bằng mã hóa tương ứng trong bảng mã hóa. Ví dụ, với dữ liệu đầu vào là “abracadabra”, ta có thể mã hóa thành chuỗi bit sau:

0 101 11 0 1000 0 1001 0 11 0 101

*Bước 5: Lưu trữ dữ liệu đã mã hóa và cây huffman vào file đầu ra*

Lưu trữ dữ liệu đã mã hóa và cây huffman vào file đầu ra. Ta có thể sử dụng một số phương pháp để lưu trữ cây huffman vào file, ví dụ như sử dụng mã hóa tiền tố hoặc mã hóa dài cố định. Ta cũng cần lưu trữ số bit của dữ liệu đã mã hóa để có thể giải nén chính xác.

### **3. Mô tả chi tiết thuật toán giải mã dữ liệu đã nén bằng huffman coding**

*Bước 1: Đọc file đầu vào và lấy ra cây huffman và dữ liệu đã mã hóa*

Ta có thể sử dụng phương pháp lưu trữ cây huffman đã dùng khi nén để khôi phục lại cây huffman từ file đầu vào. Ta cũng cần lấy ra số bit của dữ liệu đã mã hóa để biết khi nào kết thúc quá trình giải mã.

*Bước 2: Giải mã dữ liệu đã mã hóa theo cây huffman đã lấy ra*

Ta có thể duyệt qua từng bit trong dữ liệu đã mã hóa, và đi theo cây huffman từ gốc đến lá, theo quy tắc sau: nếu bit là 0, đi sang con trái; nếu bit là 1, đi sang con phải. Khi gặp một nút lá, ta lấy ký tự tương ứng của nút đó và thêm vào chuỗi kết quả. Sau đó, ta quay lại gốc của cây và tiếp tục duyệt cho đến khi hết số bit đã lấy ra ở bước 1. Ví dụ, với dữ liệu đã mã hóa và cây huffman trên, ta có thể giải mã thành chuỗi ký tự sau: 0 -> a 101 -> b 11 -> r 0 -> a 1000 -> c 0 -> a 1001 -> d 0 -> a 11 -> r 0 -> a 101 -> b

*Bước 3: Lưu trữ dữ liệu đã giải mã vào file đầu ra*

Ta có thể sử dụng một số phương pháp để lưu trữ chuỗi ký tự đã giải mã vào file đầu ra, ví dụ như sử dụng mã ASCII hoặc Unicode. Ví dụ, với chuỗi ký tự đã giải mã là “abracadabra”, ta có thể lưu trữ vào file như sau:

|  |  |
| --- | --- |
| **Mã ASCII** | **Ký tự** |
| 97 | a |
| 98 | b |
| 114 | r |
| 97 | a |
| 99 | c |
| 97 | a |
| 100 | d |
| 97 | a |
| 114 | r |
| 97 | a |
| 98 | b |

## **CHƯƠNG 3: QUY TRÌNH THIẾT KẾ**

### **1. Giới thiệu tổng quát về phần mềm và ngôn ngữ sử dụng**

Nhóm 1 sẽ thiết kế một hệ thống phần cứng bằng ngôn ngữ SystemVerilog. Các công cụ phần mềm được sử dụng trong quá trình thiết kế bao gồm:

- *Draw.io* để vẽ sơ đồ khối của hệ thống.

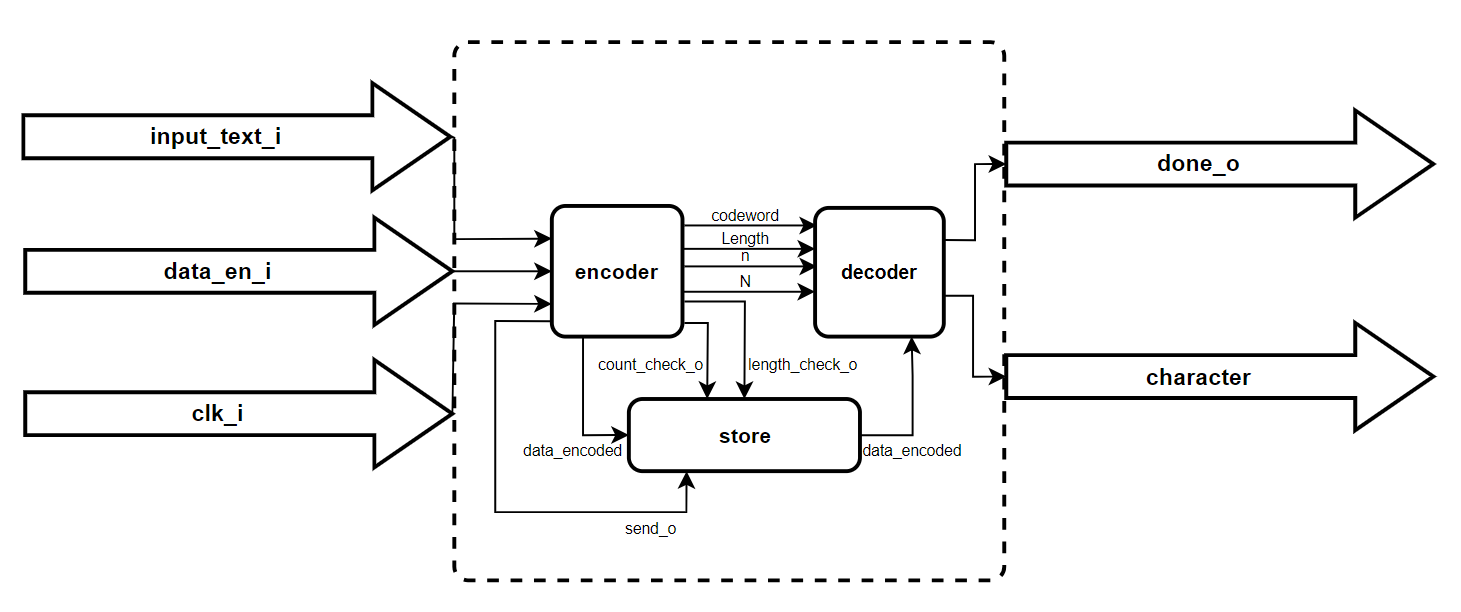
- *Visual Studio Code* để viết mã nguồn SystemVerilog.

- *Verilator* để biên dịch và mô phỏng mã nguồn SystemVerilog.

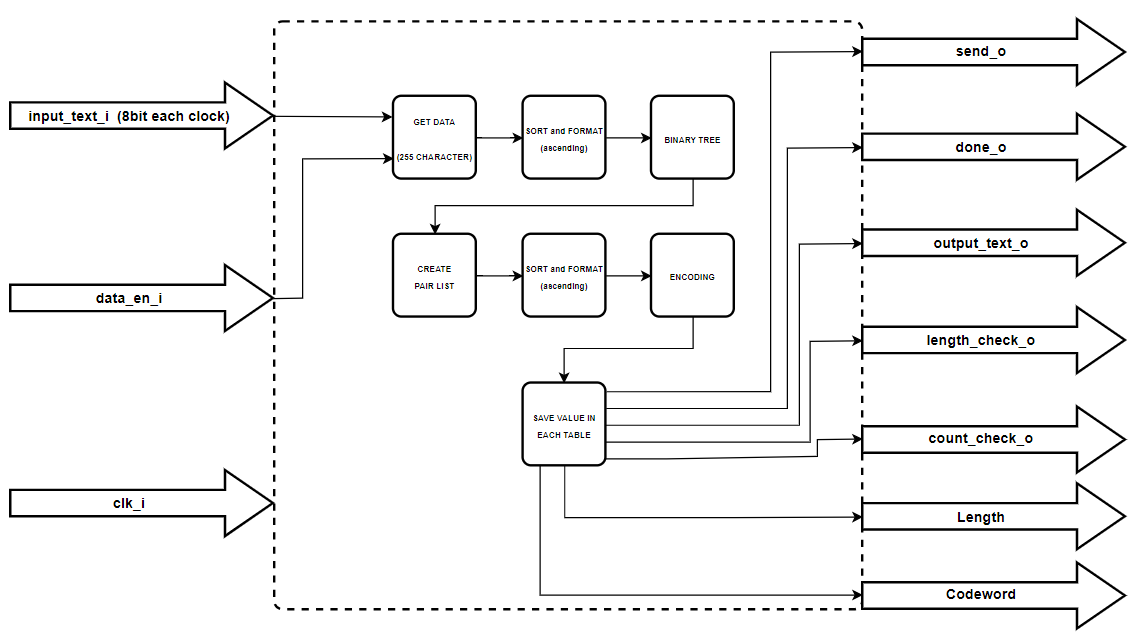
- *Quartus* để tổng hợp mã nguồn SystemVerilog thành mạch logic.

### **2. Thiết kế cấp độ hệ thống (Specification)**

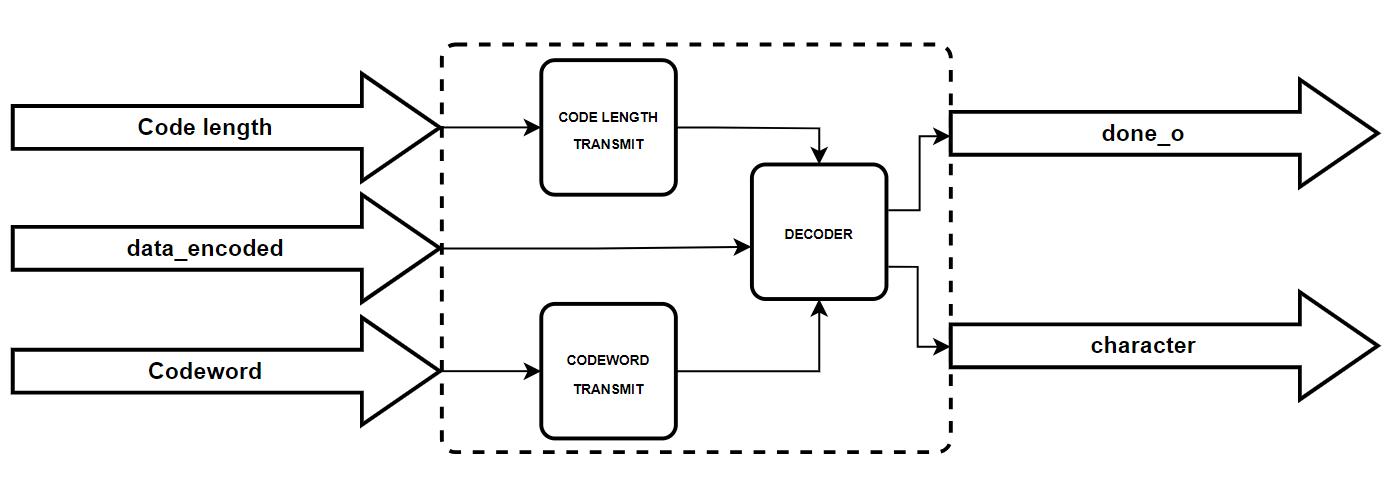
#### 2.1. Sơ đồ khối tổng quát



#### 2.2. Sơ đồ khối bộ mã hóa



#### 2.3. Sơ đồ khối bộ giải mã



### **3. Thiết kế cấp độ RTL (Register Transfer Level)**

…

### **4. Thực hiện mô phỏng (Simulation)**

…

### **5. Thực hiện tổng hợp (Synthesis)**

…

## **CHƯƠNG 4: KẾT QUẢ THỰC TẾ**

1. Môi trường và công cụ thực hiện thực tế thuật toán nén dữ liệu không mất mát bằng huffman coding

* Giới thiệu về vi điều khiển và phần cứng sử dụng để thực hiện thực tế thuật toán
* Giới thiệu về các tham số đầu vào và đầu ra của thuật toán khi thực hiện thực tế

1. Kết quả thực hiện thực tế thuật toán nén dữ liệu không mất mát bằng huffman coding

* Trình bày kết quả thực hiện thực tế với các loại dữ liệu khác nhau (văn bản, hình ảnh, âm thanh, video…)
* So sánh kết quả thực hiện thực tế với các phương pháp nén dữ liệu không mất mát khác (run-length encoding, Lempel-Ziv-Welch…)
* Đánh giá hiệu quả nén dữ liệu không mất mát bằng huffman coding theo các tiêu chí (tỉ lệ nén, thời gian nén và giải nén, độ phức tạp thuật toán…)

# **III – PHẦN KẾT LUẬN**

# **IV – TÀI LIỆU THAM KHẢO**

Lossless Compression: A Complete Guide | Adobe : <https://www.geeksforgeeks.org/lossless-and-lossy-compression/>

Advantages and Disadvantages of Huffman Coding - OpenGenus IQ : <https://iq.opengenus.org/advantages-and-disadvantages-of-huffman-coding/>

# **V – PHỤ LỤC**