Nội dung

[Chủ đề 1: Thiết kế và triển khai bộ mã hóa BCH trên FPGA 2](#_Toc180575923)

[Chủ đề 2: Thiết kế bộ giải mã BCH với phát hiện và sửa lỗi 3](#_Toc180575924)

[Chủ đề 3: Thiết kế và triển khai bộ mã hóa Cyclic trên FPGA 6](#_Toc180575925)

[Chủ đề 4: Thiết kế bộ giải mã Cyclic với khả năng phát hiện và sửa lỗi 7](#_Toc180575926)

[Chủ đề 5: Thiết kế và triển khai bộ mã hóa Convolutional Code trên FPGA 10](#_Toc180575927)

[Chủ đề 6: Thiết kế bộ giải mã Convolutional Code với thuật toán Viterbi 11](#_Toc180575928)

[Chủ đề 7: Thiết kế và tính toán hệ số cho bộ lọc FIR thông thấp pha tuyến tính bậc 5 14](#_Toc180575929)

[Chủ đề 8: Thiết kế và triển khai bộ lọc FIR thông thấp bậc 5 trên FPGA 15](#_Toc180575930)

[Chủ đề 9: Thiết kế và tính toán hệ số cho bộ lọc FIR thông dải pha tuyến tính bậc 5 18](#_Toc180575931)

[Chủ đề 10: Thiết kế và triển khai bộ lọc FIR thông dải bậc 5 trên FPGA 19](#_Toc180575932)

[Chủ đề 11: Thiết kế và triển khai bộ mã hóa Hamming (7,4) trên FPGA 22](#_Toc180575933)

[Câu hỏi 13: Thiết kế và triển khai bộ mã hóa Reed-Solomon trên FPGA 25](#_Toc180575934)

[Câu hỏi 14: Thiết kế bộ giải mã Reed-Solomon với khả năng phát hiện và sửa lỗi 26](#_Toc180575935)

[Câu hỏi 15: Thiết kế hệ truyền tin bảo mật tối tiểu cho mã mạng dựa trên bài toán logarit rời rạc 28](#_Toc180575936)

[Câu hỏi 16: Thiết kế và triển khai bộ mã hóa Cyclic cục bộ trên FPGA 30](#_Toc180575937)

[Câu hỏi 17: Thiết kế bộ giải mã Cyclic cục bộ với khả năng phát hiện và sửa lỗi 31](#_Toc180575938)

[Bài 18: Xây dựng thuật toán mã hóa và giải mã Haffman trên FPGA 33](#_Toc180575939)

[Bài 19: Thiết kế và mô tả đồng hồ thời gian thực (Ngày trong tuần-Giờ-Phút-Giây). 33](#_Toc180575940)

[Bài 20: Thiết kế và mô tả mạch hiển thị dòng chữ “KTDT PTIT” trên ma trận LED với một số hiệu ứng 33](#_Toc180575941)

# Chủ đề 1: Thiết kế và triển khai bộ mã hóa BCH trên FPGA

* **Yêu cầu**: Sinh viên cần thiết kế một **bộ mã hóa BCH** để mã hóa dữ liệu đầu vào với khả năng phát hiện và sửa lỗi nhiều bit. Ví dụ: thiết kế mã BCH trong đó:
  + là độ dài của chuỗi mã hóa (tổng số bit),
  + là số bit dữ liệu,
  + là số bit lỗi có thể sửa được.
* **Mục tiêu**:
  + Nghiên cứu mã BCH, cách thức mã hóa dữ liệu và tạo bit kiểm tra (parity).
  + Thiết kế bộ mã hóa BCH bằng ngôn ngữ **Verilog hoặc VHDL**.
  + Triển khai thiết kế trên FPGA và thực hiện mô phỏng để kiểm tra quá trình mã hóa.
* **Yêu cầu chi tiết**:
  + Mã BCH hoạt động như thế nào? Trình bày quy trình tạo các bit kiểm tra cho mã BCH .
  + Viết mã Verilog để thực hiện bộ mã hóa BCH và mô phỏng hoạt động của nó.
  + Thực hiện kiểm tra mã hóa với các chuỗi dữ liệu khác nhau và phân tích kết quả.
  + Sử dụng các công cụ phân tích tài nguyên FPGA (Vivado/Quartus) để đánh giá mức độ sử dụng tài nguyên của thiết kế ban đầu.
  + Đề xuất và thực hiện các giải pháp tối ưu hóa như sử dụng pipeline, song song hóa, và giảm thiểu số bước tính toán để cải thiện hiệu năng.
  + So sánh hiệu quả của hệ thống trước và sau tối ưu hóa, tập trung vào các tiêu chí: thời gian trễ, mức tiêu thụ tài nguyên, và tốc độ xử lý.

# Chủ đề 2: Thiết kế bộ giải mã BCH với phát hiện và sửa lỗi

* **Yêu cầu**: Sinh viên cần thiết kế một **bộ giải mã BCH** có khả năng phát hiện và sửa lỗi đa bit. Bộ giải mã phải sử dụng thuật toán Euclid mở rộng hoặc phương pháp khác để phát hiện và sửa lỗi.
* **Mục tiêu**:
  1. Nghiên cứu quá trình giải mã BCH và thuật toán sửa lỗi (ví dụ: sử dụng **Chien Search** hoặc **Berlekamp-Massey Algorithm** để tìm nghiệm đa thức lỗi).
  2. Viết mã Verilog/VHDL cho bộ giải mã BCH.
  3. Mô phỏng quá trình giải mã với các chuỗi dữ liệu chứa lỗi để kiểm tra hiệu quả sửa lỗi.
* **Yêu cầu chi tiết**:
  1. Trình bày chi tiết về thuật toán giải mã BCH và các bước để phát hiện, xác định vị trí và sửa lỗi.
  2. Viết mã Verilog cho bộ giải mã BCH và mô phỏng quá trình phát hiện, sửa lỗi nhiều bit.
  3. Mô phỏng hệ thống giải mã với các chuỗi dữ liệu chứa lỗi, phân tích mức độ hiệu quả của hệ thống trong việc phát hiện và sửa lỗi.
  4. Sử dụng các công cụ phân tích tài nguyên FPGA (Vivado/Quartus) để đánh giá mức độ sử dụng tài nguyên của thiết kế ban đầu.
  5. Đề xuất và thực hiện các giải pháp tối ưu hóa như sử dụng pipeline, song song hóa, và giảm thiểu số bước tính toán để cải thiện hiệu năng.
  6. So sánh hiệu quả của hệ thống trước và sau tối ưu hóa, tập trung vào các tiêu chí: thời gian trễ, mức tiêu thụ tài nguyên, và tốc độ xử lý.

**Báo cáo và thuyết trình kết quả**

* Sinh viên cần chuẩn bị một **báo cáo chi tiết** mô tả quá trình thiết kế, mô phỏng, và tối ưu hóa hệ thống mã BCH.
* **Nội dung báo cáo**:
  1. Giới thiệu về mã BCH và các tham số chính của mã.
  2. Trình bày chi tiết cách thực hiện bộ mã hóa và giải mã BCH.
  3. Phân tích kết quả mô phỏng của quá trình mã hóa và giải mã, bao gồm các trường hợp phát hiện và sửa lỗi.
  4. Đánh giá hiệu năng của hệ thống trước và sau khi tối ưu hóa (về tài nguyên FPGA và thời gian trễ).
* **Thuyết trình**: Sinh viên cần thuyết trình (~10-15 phút) về quá trình thực hiện bài tập lớn, bao gồm các kết quả mô phỏng và tối ưu hóa thiết kế.

**Tiêu chí đánh giá**

* **Thiết kế chính xác và hoàn chỉnh** của bộ mã hóa và giải mã BCH (40%).
* **Tối ưu hóa thiết kế** và phân tích hiệu năng (30%).
* **Mô phỏng và thử nghiệm với các dữ liệu có lỗi** (20%).
* **Báo cáo và thuyết trình** (10%).

**Hướng dẫn triển khai:**

**1. Nghiên cứu lý thuyết:**

* Sinh viên cần tìm hiểu về mã BCH, bao gồm cách mã hóa, phát hiện và sửa lỗi đa bit.
* Hiểu các thuật toán như **Chien Search**, **Berlekamp-Massey Algorithm** để giải mã BCH.

**2. Viết mã Verilog/VHDL:**

* Sinh viên sử dụng ngôn ngữ **Verilog hoặc VHDL** để viết mã cho bộ mã hóa và giải mã BCH.
* Sử dụng các công cụ phát triển FPGA như **Xilinx Vivado** hoặc **Altera Quartus** để biên dịch, mô phỏng, và kiểm tra thiết kế.

**3. Tối ưu hóa:**

* Sau khi thiết kế hệ thống thành công, sinh viên sẽ tối ưu hóa hệ thống trên FPGA để giảm mức sử dụng tài nguyên và cải thiện tốc độ xử lý.

**Gợi ý tài liệu tham khảo:**

* **Sách giáo khoa**: Tìm hiểu về **Error Correction Coding** hoặc **Digital Communications** có các chương về mã BCH.
* **Tài liệu FPGA**: Sử dụng các tài liệu hướng dẫn về thiết kế phần cứng trên FPGA của **Xilinx Vivado** hoặc **Altera Quartus**.
* **Bài báo nghiên cứu**: Tham khảo các bài báo về tối ưu hóa mã sửa lỗi Reed-Solomon hoặc BCH trên FPGA.

# Chủ đề 3: Thiết kế và triển khai bộ mã hóa Cyclic trên FPGA

* **Yêu cầu**: Sinh viên cần thiết kế và triển khai một **bộ mã hóa Cyclic** để mã hóa chuỗi dữ liệu đầu vào. Ví dụ: mã Cyclic , trong đó:
  + : Độ dài mã (số bit đầu ra),
  + : Số bit dữ liệu (số bit đầu vào).
* **Mục tiêu**:
  + Nghiên cứu lý thuyết về mã Cyclic và cách tạo mã từ đa thức sinh (Generator Polynomial).
  + Viết mã **Verilog hoặc VHDL** để thiết kế bộ mã hóa Cyclic trên FPGA.
  + Mô phỏng quá trình mã hóa và kiểm tra tính đúng đắn của dữ liệu mã hóa đầu ra.
* **Yêu cầu chi tiết**:
  + Trình bày nguyên tắc hoạt động của mã Cyclic và vai trò của đa thức sinh (generator polynomial) trong quá trình mã hóa.
  + Viết mã Verilog để thiết kế bộ mã hóa Cyclic và thực hiện mô phỏng trên FPGA.
  + Thực hiện kiểm tra mã hóa với các chuỗi dữ liệu đầu vào khác nhau và phân tích kết quả mã hóa.
  + Sử dụng các công cụ phân tích tài nguyên FPGA (Vivado/Quartus) để đánh giá mức độ sử dụng tài nguyên của thiết kế ban đầu.
  + Đề xuất và thực hiện các giải pháp tối ưu hóa như sử dụng pipeline, song song hóa, và giảm thiểu số bước tính toán để cải thiện hiệu năng.
  + So sánh hiệu quả của hệ thống trước và sau tối ưu hóa, tập trung vào các tiêu chí: thời gian trễ, mức tiêu thụ tài nguyên, và tốc độ xử lý.

# Chủ đề 4: Thiết kế bộ giải mã Cyclic với khả năng phát hiện và sửa lỗi

* **Yêu cầu:** Sinh viên cần thiết kế một bộ giải mã Cyclic có khả năng phát hiện và sửa lỗi. Bộ giải mã phải phát hiện lỗi dựa trên tính chất của mã Cyclic (sử dụng đa thức kiểm tra) và sửa lỗi khi có lỗi xuất hiện.
* **Mục tiêu:**
  1. Nghiên cứu thuật toán giải mã Cyclic, bao gồm phát hiện lỗi và sửa lỗi một cách hiệu quả.
  2. Viết mã Verilog/VHDL để thiết kế bộ giải mã Cyclic.
  3. Mô phỏng quá trình giải mã với các chuỗi dữ liệu chứa lỗi và kiểm tra kết quả giải mã.
* **Yêu cầu chi tiết:**
  1. Trình bày cách phát hiện lỗi trong mã Cyclic dựa trên đa thức sinh và cách tìm ra đa thức lỗi. Làm thế nào để sửa lỗi dựa trên thông tin nhận được?
  2. Viết mã Verilog để thiết kế bộ giải mã Cyclic, bao gồm khả năng phát hiện và sửa lỗi.
  3. Thực hiện mô phỏng với các chuỗi dữ liệu có lỗi và phân tích kết quả giải mã để đánh giá hiệu quả của hệ thống.
  4. Sử dụng các công cụ phân tích tài nguyên FPGA (Vivado/Quartus) để đánh giá mức độ sử dụng tài nguyên của thiết kế ban đầu.
  5. Đề xuất và thực hiện các giải pháp tối ưu hóa như sử dụng pipeline, song song hóa, và giảm thiểu số bước tính toán để cải thiện hiệu năng.
  6. So sánh hiệu quả của hệ thống trước và sau tối ưu hóa, tập trung vào các tiêu chí: thời gian trễ, mức tiêu thụ tài nguyên, và tốc độ xử lý.

**Báo cáo và thuyết trình kết quả**

* Sinh viên cần chuẩn bị một **báo cáo chi tiết** mô tả quá trình thiết kế, mô phỏng, và tối ưu hóa hệ thống mã CYCLIC (MÃ VÒNG).
* **Nội dung báo cáo**:
  1. Giới thiệu về mã CYCLIC (MÃ VÒNG) và các tham số chính của mã.
  2. Trình bày chi tiết cách thực hiện bộ mã hóa và giải mã CYCLIC (MÃ VÒNG).
  3. Phân tích kết quả mô phỏng của quá trình mã hóa và giải mã, bao gồm các trường hợp phát hiện và sửa lỗi.
  4. Đánh giá hiệu năng của hệ thống trước và sau khi tối ưu hóa (về tài nguyên FPGA và thời gian trễ).
* **Thuyết trình**: Sinh viên cần thuyết trình (~10-15 phút) về quá trình thực hiện bài tập lớn, bao gồm các kết quả mô phỏng và tối ưu hóa thiết kế.

**Tiêu chí đánh giá**

* **Thiết kế chính xác và hoàn chỉnh** của bộ mã hóa và giải mã CYCLIC (MÃ VÒNG) (40%).
* **Tối ưu hóa thiết kế** và phân tích hiệu năng (30%).
* **Mô phỏng và thử nghiệm với các dữ liệu có lỗi** (20%).
* **Báo cáo và thuyết trình** (10%).

**Hướng dẫn triển khai:**

**1. Nghiên cứu lý thuyết:**

* Sinh viên cần tìm hiểu về mã CYCLIC (MÃ VÒNG), bao gồm cách mã hóa, phát hiện và sửa lỗi.
* Hiểu các thuật toán như **Euclid mở rộng,** **Chien Search**, **Berlekamp-Massey Algorithm** để giải mã CYCLIC (MÃ VÒNG).

**2. Viết mã Verilog/VHDL:**

* Sinh viên sử dụng ngôn ngữ **Verilog hoặc VHDL** để viết mã cho bộ mã hóa và giải mã CYCLIC (MÃ VÒNG).
* Sử dụng các công cụ phát triển FPGA như **Xilinx Vivado** hoặc **Altera Quartus** để biên dịch, mô phỏng, và kiểm tra thiết kế.

**3. Tối ưu hóa:**

* Sau khi thiết kế hệ thống thành công, sinh viên sẽ tối ưu hóa hệ thống trên FPGA để giảm mức sử dụng tài nguyên và cải thiện tốc độ xử lý.

**Gợi ý tài liệu tham khảo:**

* **Sách giáo khoa**: Tìm hiểu về **Error Correction Coding** hoặc **Digital Communications** có các chương về mã CYCLIC (MÃ VÒNG).
* **Tài liệu FPGA**: Sử dụng các tài liệu hướng dẫn về thiết kế phần cứng trên FPGA của **Xilinx Vivado** hoặc **Altera Quartus**.
* **Bài báo nghiên cứu**: Tham khảo các bài báo về tối ưu hóa mã sửa lỗi Reed-Solomon hoặc CYCLIC (MÃ VÒNG) trên FPGA.

# Chủ đề 5: Thiết kế và triển khai bộ mã hóa Convolutional Code trên FPGA

* **Yêu cầu**: Sinh viên cần thiết kế và triển khai một **bộ mã hóa mã chập**. Ví dụ: thiết kế một bộ mã hóa mã chập với độ dài constraint , tỷ lệ mã hóa ​, trong đó một bit dữ liệu đầu vào sẽ tạo ra hai bit đầu ra.
* **Mục tiêu**:
  1. Nghiên cứu lý thuyết về mã chập, bao gồm cấu trúc **trellis diagram** (sơ đồ lưới) và các tham số chính như độ dài ràng buộc (constraint) và tỷ lệ mã hóa.
  2. Viết mã **Verilog hoặc VHDL** để thiết kế bộ mã hóa mã chập trên FPGA.
  3. Triển khai và mô phỏng hoạt động của bộ mã hóa trên FPGA với các dữ liệu đầu vào khác nhau.
* **Yêu cầu chi tiết**:
  1. Trình bày cách thức hoạt động của mã chập. Cách tính toán các bit mã hóa từ bit dữ liệu đầu vào như thế nào?
  2. Viết mã Verilog/VHDL cho bộ mã hóa mã chập và triển khai trên FPGA.
  3. Thực hiện mô phỏng quá trình mã hóa với các chuỗi dữ liệu đầu vào khác nhau và kiểm tra kết quả đầu ra.
  4. Sử dụng các công cụ phân tích tài nguyên FPGA (Vivado/Quartus) để đánh giá mức độ sử dụng tài nguyên của thiết kế ban đầu.
  5. Đề xuất và thực hiện các giải pháp tối ưu hóa như sử dụng pipeline, song song hóa, và giảm thiểu số bước tính toán để cải thiện hiệu năng.
  6. So sánh hiệu quả của hệ thống trước và sau tối ưu hóa, tập trung vào các tiêu chí: thời gian trễ, mức tiêu thụ tài nguyên, và tốc độ xử lý.

# Chủ đề 6: Thiết kế bộ giải mã Convolutional Code với thuật toán Viterbi

* **Yêu cầu**: Sinh viên cần thiết kế và triển khai **bộ giải mã Viterbi** để giải mã mã chập dựa trên các bit đầu ra nhận được từ bộ mã hóa, và khôi phục chuỗi dữ liệu gốc.
* **Mục tiêu**:
  1. Nghiên cứu lý thuyết về giải mã mã chập bằng thuật toán Viterbi. Xây dựng cấu trúc trellis và hiểu nguyên tắc **Add-Compare-Select (ACS)**.
  2. Viết mã Verilog/VHDL cho bộ giải mã Viterbi.
  3. Mô phỏng quá trình giải mã và kiểm tra độ chính xác của dữ liệu được giải mã khi có nhiễu trong dữ liệu đầu vào.
* **Yêu cầu chi tiết**:
  1. Trình bày cách hoạt động của thuật toán Viterbi. Cách thức xây dựng **trellis diagram** và quá trình **Add-Compare-Select** để tìm ra chuỗi dữ liệu gốc là gì?
  2. Viết mã Verilog/VHDL để thực hiện bộ giải mã Viterbi trên FPGA.
  3. Mô phỏng hệ thống giải mã Viterbi với các chuỗi tín hiệu nhận được có nhiễu và phân tích kết quả giải mã.
  4. Sử dụng các công cụ phân tích tài nguyên FPGA (Vivado/Quartus) để đánh giá mức độ sử dụng tài nguyên của thiết kế ban đầu.
  5. Đề xuất và thực hiện các giải pháp tối ưu hóa như sử dụng pipeline, song song hóa, và giảm thiểu số bước tính toán để cải thiện hiệu năng.
  6. So sánh hiệu quả của hệ thống trước và sau tối ưu hóa, tập trung vào các tiêu chí: thời gian trễ, mức tiêu thụ tài nguyên, và tốc độ xử lý.

**Báo cáo và thuyết trình kết quả**

* **Báo cáo chi tiết**: Sinh viên cần chuẩn bị một **báo cáo** mô tả quá trình thiết kế, mô phỏng, và tối ưu hóa hệ thống mã chập.
  + Nội dung báo cáo gồm:
    1. Giới thiệu về mã chập và các thuật toán giải mã (Viterbi).
    2. Mô tả chi tiết về thiết kế và thực thi mã chập trên FPGA.
    3. Phân tích kết quả mô phỏng, đặc biệt là quá trình mã hóa và giải mã khi có nhiễu.
    4. Đánh giá hiệu suất trước và sau tối ưu hóa, bao gồm mức độ sử dụng tài nguyên và thời gian xử lý.
* **Thuyết trình**: Sinh viên cần chuẩn bị bài thuyết trình (~10-15 phút) về quá trình thực hiện bài tập lớn, bao gồm các kết quả mô phỏng và tối ưu hóa thiết kế.

**Tiêu chí đánh giá**

* **Thiết kế chính xác và hoàn chỉnh** của bộ mã hóa và giải mã mã chập (40%).
* **Tối ưu hóa thiết kế** và phân tích hiệu suất hệ thống (30%).
* **Mô phỏng và kiểm tra với dữ liệu có nhiễu** (20%).
* **Báo cáo và thuyết trình** (10%).

**Hướng dẫn triển khai:**

**1. Nghiên cứu lý thuyết:**

* Sinh viên cần tìm hiểu lý thuyết về mã chập và thuật toán Viterbi, bao gồm cách thức mã hóa, giải mã và các tham số liên quan như tỷ lệ mã hóa, độ dài constraint, trellis diagram.
* Tham khảo tài liệu về các thuật toán giải mã khác như **BCJR**, mặc dù trọng tâm bài tập này là thuật toán Viterbi.

**2. Viết mã Verilog/VHDL:**

* Sinh viên sẽ viết mã **Verilog hoặc VHDL** để thiết kế và triển khai bộ mã hóa và giải mã mã chập trên FPGA.
* Sử dụng các công cụ phát triển FPGA như **Xilinx Vivado** hoặc **Altera Quartus** để biên dịch và mô phỏng thiết kế.

**3. Tối ưu hóa:**

* Sau khi thiết kế hệ thống thành công, sinh viên sẽ thực hiện tối ưu hóa hệ thống trên FPGA để giảm mức sử dụng tài nguyên và cải thiện tốc độ xử lý.
* Sử dụng các kỹ thuật tối ưu hóa như **pipelining**, **parallelization**, và **resource sharing**.

# Chủ đề 7: Thiết kế và tính toán hệ số cho bộ lọc FIR thông thấp pha tuyến tính bậc 5

* **Yêu cầu**: Sinh viên cần thiết kế một bộ lọc **FIR thông thấp** có pha tuyến tính bậc 5, xác định các hệ số bộ lọc Bộ lọc thông thấp có chức năng giữ lại các thành phần tần số thấp và loại bỏ các thành phần tần số cao hơn.
* **Mục tiêu**:
  1. Nghiên cứu lý thuyết về bộ lọc FIR thông thấp và pha tuyến tính, cùng các yêu cầu về đáp ứng tần số.
  2. Tính toán các hệ số của bộ lọc FIR thông thấp bậc 5 bằng các phương pháp như **phương pháp cửa sổ (window method)** hoặc sử dụng phần mềm như **MATLAB**.
  3. Phân tích đáp ứng tần số của bộ lọc để đảm bảo bộ lọc đạt được các tiêu chí thiết kế (ví dụ: tần số cắt, dải thông, độ suy giảm dải chắn).
* **Yêu cầu chi tiết**:
  1. Trình bày công thức tổng quát của bộ lọc FIR thông thấp bậc 5 và giải thích lý do phải có pha tuyến tính.
  2. Sử dụng **MATLAB** hoặc công cụ tương đương để tính toán các hệ số của bộ lọc FIR thông thấp bậc 5. Viết mã tính các hệ số bằng phương pháp cửa sổ (window method) và phân tích kết quả.
  3. Vẽ đồ thị đáp ứng tần số và pha của bộ lọc để kiểm tra xem bộ lọc có đáp ứng yêu cầu thiết kế không (ví dụ: dải thông, dải chắn).
  4. Trình bày các kỹ thuật tối ưu hóa như **pipelining**, **unrolling**, và **resource sharing** để cải thiện hiệu năng của bộ lọc FIR trên FPGA.
  5. Viết mã Verilog/VHDL tối ưu hóa thiết kế của bộ lọc FIR thông thấp, áp dụng các kỹ thuật trên. Giảm số lượng bộ nhân sử dụng bằng cách chia sẻ tài nguyên hoặc thực hiện song song các phép toán.
  6. Sử dụng công cụ phân tích hiệu suất như **Xilinx Vivado** hoặc **Altera Quartus** để so sánh mức sử dụng tài nguyên trước và sau tối ưu hóa. Đánh giá hiệu suất của hệ thống dựa trên mức sử dụng tài nguyên và tốc độ xử lý.

# Chủ đề 8: Thiết kế và triển khai bộ lọc FIR thông thấp bậc 5 trên FPGA

* **Yêu cầu**: Sinh viên cần viết mã **Verilog/VHDL** để triển khai bộ lọc FIR thông thấp bậc 5 trên FPGA, sử dụng các hệ số đã tính toán ở câu 1.
* **Mục tiêu**:
  1. Hiểu nguyên tắc thực thi bộ lọc FIR trên FPGA và cách tính toán đầu ra của bộ lọc dựa trên các hệ số .
  2. Viết mã HDL (Verilog hoặc VHDL) để triển khai bộ lọc FIR thông thấp với các hệ số đã tính toán.
  3. Thực hiện mô phỏng trên FPGA và kiểm tra đáp ứng tín hiệu đầu ra với các tín hiệu đầu vào khác nhau (tín hiệu bước, xung, sin).
* **Yêu cầu chi tiết**:
  1. Viết mã Verilog/VHDL để thực thi bộ lọc FIR thông thấp bậc 5 trên FPGA, đảm bảo mỗi đầu ra của bộ lọc được tính bằng tổ hợp tuyến tính của các đầu vào trước đó.
  2. Mô phỏng và kiểm tra đáp ứng đầu ra với các tín hiệu đầu vào khác nhau (như tín hiệu bước, tín hiệu xung và tín hiệu sin). Đánh giá hiệu quả lọc các thành phần tần số cao trong tín hiệu đầu vào.
  3. So sánh kết quả mô phỏng với đáp ứng lý thuyết của bộ lọc.
  4. Trình bày các kỹ thuật tối ưu hóa như **pipelining**, **unrolling**, và **resource sharing** để cải thiện hiệu năng của bộ lọc FIR trên FPGA.
  5. Viết mã Verilog/VHDL tối ưu hóa thiết kế của bộ lọc FIR thông thấp, áp dụng các kỹ thuật trên. Giảm số lượng bộ nhân sử dụng bằng cách chia sẻ tài nguyên hoặc thực hiện song song các phép toán.
  6. Sử dụng công cụ phân tích hiệu suất như **Xilinx Vivado** hoặc **Altera Quartus** để so sánh mức sử dụng tài nguyên trước và sau tối ưu hóa. Đánh giá hiệu suất của hệ thống dựa trên mức sử dụng tài nguyên và tốc độ xử lý.

**Yêu cầu bổ sung: Báo cáo và thuyết trình kết quả**

* **Báo cáo chi tiết**: Sinh viên cần chuẩn bị một **báo cáo** mô tả quá trình thiết kế, tính toán hệ số, mô phỏng và tối ưu hóa bộ lọc FIR thông thấp bậc 5 trên FPGA.
  + Nội dung báo cáo bao gồm:
    1. Giới thiệu về bộ lọc FIR thông thấp và lý do chọn bộ lọc pha tuyến tính.
    2. Mô tả chi tiết về tính toán hệ số lọc FIR, bao gồm công cụ hoặc phương pháp sử dụng (ví dụ: MATLAB).
    3. Phân tích quá trình triển khai bộ lọc FIR trên FPGA và đánh giá kết quả mô phỏng với các tín hiệu đầu vào.
    4. Trình bày kết quả tối ưu hóa thiết kế, so sánh mức sử dụng tài nguyên và hiệu suất trước và sau khi tối ưu hóa.
* **Thuyết trình**: Sinh viên cần chuẩn bị một bài thuyết trình (~10-15 phút) về quá trình thực hiện bài tập lớn, bao gồm các kết quả tính toán, mô phỏng và tối ưu hóa thiết kế.

**Tiêu chí đánh giá**

* **Thiết kế chính xác và hoàn chỉnh** của bộ lọc FIR thông thấp trên FPGA (40%).
* **Tính toán hệ số bộ lọc và phân tích đáp ứng tần số** (30%).
* **Tối ưu hóa thiết kế và phân tích hiệu suất hệ thống** (20%).
* **Báo cáo và thuyết trình** (10%).

**Hướng dẫn triển khai:**

**1. Nghiên cứu lý thuyết:**

* Sinh viên cần tìm hiểu về lý thuyết của bộ lọc FIR thông thấp, pha tuyến tính, và cách xác định các hệ số lọc.
* Nghiên cứu các phương pháp tính toán hệ số FIR như **phương pháp cửa sổ (window method)**.

**2. Viết mã Verilog/VHDL:**

* Sinh viên sẽ viết mã **Verilog hoặc VHDL** để thiết kế bộ lọc FIR thông thấp bậc 5 và triển khai trên FPGA.
* Sử dụng công cụ phát triển FPGA như **Xilinx Vivado** hoặc **Altera Quartus** để biên dịch và mô phỏng thiết kế.

**3. Tối ưu hóa:**

* Sinh viên sẽ sử dụng các kỹ thuật tối ưu hóa như **pipeline** và **parallelization** để cải thiện hiệu năng và giảm mức sử dụng tài nguyên FPGA.

**Tài liệu tham khảo:**

* **Sách giáo khoa**: Tìm hiểu về **Digital Signal Processing** và **Digital Filters** để nắm vững lý thuyết bộ lọc FIR.
* **Tài liệu FPGA**: Sử dụng các hướng dẫn của **Xilinx Vivado** hoặc **Altera Quartus** để thực thi bộ lọc FIR trên FPGA.
* **Bài báo nghiên cứu**: Tham khảo các bài báo về tối ưu hóa bộ lọc FIR trên FPGA để nâng cao hiệu suất và giảm thiểu tài nguyên sử dụng.

# Chủ đề 9: Thiết kế và tính toán hệ số cho bộ lọc FIR thông dải pha tuyến tính bậc 5

* **Yêu cầu**: Sinh viên cần thiết kế một bộ lọc **FIR thông dải** có pha tuyến tính bậc 5, xác định các hệ số bộ lọc Bộ lọc thông dải có chức năng giữ lại các thành phần tần số thấp và loại bỏ các thành phần tần số cao hơn.
* **Mục tiêu**:
  1. Nghiên cứu lý thuyết về bộ lọc FIR thông dải và pha tuyến tính, cùng các yêu cầu về đáp ứng tần số.
  2. Tính toán các hệ số của bộ lọc FIR thông dải bậc 5 bằng các phương pháp như **phương pháp cửa sổ (window method)** hoặc sử dụng phần mềm như **MATLAB**.
  3. Phân tích đáp ứng tần số của bộ lọc để đảm bảo bộ lọc đạt được các tiêu chí thiết kế (ví dụ: tần số cắt, dải thông, độ suy giảm dải chắn).
* **Yêu cầu chi tiết**:
  1. Trình bày công thức tổng quát của bộ lọc FIR thông dải bậc 5 và giải thích lý do phải có pha tuyến tính.
  2. Sử dụng **MATLAB** hoặc công cụ tương đương để tính toán các hệ số của bộ lọc FIR thông dải bậc 5. Viết mã tính các hệ số bằng phương pháp cửa sổ (window method) và phân tích kết quả.
  3. Vẽ đồ thị đáp ứng tần số và pha của bộ lọc để kiểm tra xem bộ lọc có đáp ứng yêu cầu thiết kế không (ví dụ: dải thông, dải chắn).
  4. Trình bày các kỹ thuật tối ưu hóa như **pipelining**, **unrolling**, và **resource sharing** để cải thiện hiệu năng của bộ lọc FIR trên FPGA.
  5. Viết mã Verilog/VHDL tối ưu hóa thiết kế của bộ lọc FIR thông dải, áp dụng các kỹ thuật trên. Giảm số lượng bộ nhân sử dụng bằng cách chia sẻ tài nguyên hoặc thực hiện song song các phép toán.
  6. Sử dụng công cụ phân tích hiệu suất như **Xilinx Vivado** hoặc **Altera Quartus** để so sánh mức sử dụng tài nguyên trước và sau tối ưu hóa. Đánh giá hiệu suất của hệ thống dựa trên mức sử dụng tài nguyên và tốc độ xử lý.

# Chủ đề 10: Thiết kế và triển khai bộ lọc FIR thông dải bậc 5 trên FPGA

* **Yêu cầu**: Sinh viên cần viết mã **Verilog/VHDL** để triển khai bộ lọc FIR thông dải bậc 5 trên FPGA, sử dụng các hệ số đã tính toán ở câu 1.
* **Mục tiêu**:
  1. Hiểu nguyên tắc thực thi bộ lọc FIR trên FPGA và cách tính toán đầu ra của bộ lọc dựa trên các hệ số .
  2. Viết mã HDL (Verilog hoặc VHDL) để triển khai bộ lọc FIR thông dải với các hệ số đã tính toán.
  3. Thực hiện mô phỏng trên FPGA và kiểm tra đáp ứng tín hiệu đầu ra với các tín hiệu đầu vào khác nhau (tín hiệu bước, xung, sin).
* **Yêu cầu chi tiết**:
  1. Viết mã Verilog/VHDL để thực thi bộ lọc FIR thông dải bậc 5 trên FPGA, đảm bảo mỗi đầu ra của bộ lọc được tính bằng tổ hợp tuyến tính của các đầu vào trước đó.
  2. Mô phỏng và kiểm tra đáp ứng đầu ra với các tín hiệu đầu vào khác nhau (như tín hiệu bước, tín hiệu xung và tín hiệu sin). Đánh giá hiệu quả lọc các thành phần tần số cao trong tín hiệu đầu vào.
  3. So sánh kết quả mô phỏng với đáp ứng lý thuyết của bộ lọc.
  4. Trình bày các kỹ thuật tối ưu hóa như **pipelining**, **unrolling**, và **resource sharing** để cải thiện hiệu năng của bộ lọc FIR trên FPGA.
  5. Viết mã Verilog/VHDL tối ưu hóa thiết kế của bộ lọc FIR thông dải, áp dụng các kỹ thuật trên. Giảm số lượng bộ nhân sử dụng bằng cách chia sẻ tài nguyên hoặc thực hiện song song các phép toán.
  6. Sử dụng công cụ phân tích hiệu suất như **Xilinx Vivado** hoặc **Altera Quartus** để so sánh mức sử dụng tài nguyên trước và sau tối ưu hóa. Đánh giá hiệu suất của hệ thống dựa trên mức sử dụng tài nguyên và tốc độ xử lý.

**Yêu cầu bổ sung: Báo cáo và thuyết trình kết quả**

* **Báo cáo chi tiết**: Sinh viên cần chuẩn bị một **báo cáo** mô tả quá trình thiết kế, tính toán hệ số, mô phỏng và tối ưu hóa bộ lọc FIR thông dải bậc 5 trên FPGA.
  + Nội dung báo cáo bao gồm:
    1. Giới thiệu về bộ lọc FIR thông dải và lý do chọn bộ lọc pha tuyến tính.
    2. Mô tả chi tiết về tính toán hệ số lọc FIR, bao gồm công cụ hoặc phương pháp sử dụng (ví dụ: MATLAB).
    3. Phân tích quá trình triển khai bộ lọc FIR trên FPGA và đánh giá kết quả mô phỏng với các tín hiệu đầu vào.
    4. Trình bày kết quả tối ưu hóa thiết kế, so sánh mức sử dụng tài nguyên và hiệu suất trước và sau khi tối ưu hóa.
* **Thuyết trình**: Sinh viên cần chuẩn bị một bài thuyết trình (~10-15 phút) về quá trình thực hiện bài tập lớn, bao gồm các kết quả tính toán, mô phỏng và tối ưu hóa thiết kế.

**Tiêu chí đánh giá**

* **Thiết kế chính xác và hoàn chỉnh** của bộ lọc FIR thông dải trên FPGA (40%).
* **Tính toán hệ số bộ lọc và phân tích đáp ứng tần số** (30%).
* **Tối ưu hóa thiết kế và phân tích hiệu suất hệ thống** (20%).
* **Báo cáo và thuyết trình** (10%).

**Hướng dẫn triển khai:**

**1. Nghiên cứu lý thuyết:**

* Sinh viên cần tìm hiểu về lý thuyết của bộ lọc FIR thông dải, pha tuyến tính, và cách xác định các hệ số lọc.
* Nghiên cứu các phương pháp tính toán hệ số FIR như **phương pháp cửa sổ (window method)**.

**2. Viết mã Verilog/VHDL:**

* Sinh viên sẽ viết mã **Verilog hoặc VHDL** để thiết kế bộ lọc FIR thông dải bậc 5 và triển khai trên FPGA.
* Sử dụng công cụ phát triển FPGA như **Xilinx Vivado** hoặc **Altera Quartus** để biên dịch và mô phỏng thiết kế.

**3. Tối ưu hóa:**

* Sinh viên sẽ sử dụng các kỹ thuật tối ưu hóa như **pipeline** và **parallelization** để cải thiện hiệu năng và giảm mức sử dụng tài nguyên FPGA.

**Tài liệu tham khảo:**

* **Sách giáo khoa**: Tìm hiểu về **Digital Signal Processing** và **Digital Filters** để nắm vững lý thuyết bộ lọc FIR.
* **Tài liệu FPGA**: Sử dụng các hướng dẫn của **Xilinx Vivado** hoặc **Altera Quartus** để thực thi bộ lọc FIR trên FPGA.
* **Bài báo nghiên cứu**: Tham khảo các bài báo về tối ưu hóa bộ lọc FIR trên FPGA để nâng cao hiệu suất và giảm thiểu tài nguyên sử dụng.

# Chủ đề 11: Thiết kế và triển khai bộ mã hóa Hamming (7,4) trên FPGA

* **Yêu cầu**: Sinh viên cần thiết kế một **bộ mã hóa mã Hamming (7,4)** để mã hóa 4 bit dữ liệu thành 7 bit (4 bit dữ liệu và 3 bit kiểm tra).
* **Mục tiêu**:
  1. Thiết kế mạch số sử dụng ngôn ngữ Verilog hoặc VHDL cho bộ mã hóa Hamming.
  2. Triển khai mã trên công cụ phát triển FPGA như **Xilinx Vivado** hoặc **Altera Quartus**.
  3. Mô phỏng hoạt động của bộ mã hóa với một số dữ liệu mẫu.
* **Câu hỏi chi tiết**:
  1. Mã Hamming hoạt động như thế nào? Hãy trình bày quy trình tính các bit kiểm tra cho mã Hamming (7,4).
  2. Viết mã Verilog để thực hiện bộ mã hóa Hamming (7,4).
  3. Mô phỏng và kiểm tra kết quả của bộ mã hóa cho các đầu vào dữ liệu khác nhau.
  4. Sử dụng công cụ **timing analysis** và **resource utilization report** để đánh giá tài nguyên sử dụng của thiết kế mã Hamming.
  5. Đề xuất và triển khai các kỹ thuật tối ưu hóa (ví dụ: pipeline, giảm độ sâu logic) để cải thiện hiệu suất.
  6. Thực hiện so sánh trước và sau khi tối ưu về mức độ sử dụng tài nguyên và thời gian trễ.

**Câu hỏi 12: Thiết kế bộ giải mã Hamming với phát hiện và sửa lỗi**

* **Yêu cầu**: Sinh viên thiết kế **bộ giải mã mã Hamming** có khả năng phát hiện và sửa lỗi một bit.
* **Mục tiêu**:
  1. Thiết kế mạch số cho bộ giải mã để phát hiện và sửa lỗi đơn lẻ (Single-bit Error).
  2. Xây dựng các thuật toán phát hiện lỗi (syndrome detection) và xác định vị trí lỗi.
  3. Mô phỏng quá trình giải mã khi xuất hiện lỗi trong dữ liệu nhận được.
* **Câu hỏi chi tiết**:
  1. Trình bày quá trình giải mã Hamming (7,4), bao gồm cách tính syndrome và cách phát hiện lỗi.
  2. Viết mã Verilog cho bộ giải mã Hamming, trong đó có chức năng sửa lỗi đơn lẻ.
  3. Mô phỏng hệ thống giải mã với các trường hợp dữ liệu nhận có và không có lỗi, sau đó so sánh kết quả.
  4. Sử dụng công cụ **timing analysis** và **resource utilization report** để đánh giá tài nguyên sử dụng của thiết kế mã Hamming.
  5. Đề xuất và triển khai các kỹ thuật tối ưu hóa (ví dụ: pipeline, giảm độ sâu logic) để cải thiện hiệu suất.
  6. Thực hiện so sánh trước và sau khi tối ưu về mức độ sử dụng tài nguyên và thời gian trễ.

**Báo cáo và thuyết trình kết quả**

* Sinh viên cần chuẩn bị một **báo cáo chi tiết** về quá trình thiết kế, mô phỏng và tối ưu hóa hệ thống Hamming.
* **Nội dung báo cáo**:
  1. Giới thiệu về mã Hamming và ứng dụng của nó trong thực tiễn.
  2. Chi tiết về cách thực hiện bộ mã hóa và giải mã Hamming trên FPGA.
  3. Kết quả mô phỏng quá trình mã hóa và giải mã.
  4. Phân tích và so sánh kết quả trước và sau khi tối ưu hóa.
* **Thuyết trình**: Sinh viên cần chuẩn bị một buổi thuyết trình (~10 phút) về quá trình thực hiện bài tập lớn, bao gồm kết quả mô phỏng và tối ưu hóa.

Tiêu chí đánh giá

* **Thiết kế chính xác và hoàn chỉnh** của bộ mã hóa và giải mã Hamming (40%).
* **Tối ưu hóa thiết kế** và đánh giá hiệu suất (30%).
* **Mô phỏng và kiểm thử** với dữ liệu có lỗi (20%).
* **Báo cáo và thuyết trình** (10%).

**Hướng dẫn triển khai**:

**1. Nghiên cứu lý thuyết:**

* Sinh viên cần tìm hiểu về mã Hamming, đặc biệt là cách mã hóa, giải mã, và phát hiện/sửa lỗi.
* Tham khảo các tài liệu các thuật toán phát hiện lỗi (syndrome detection) và xác định vị trí lỗi để thực hiện giải mã.

**2. Viết mã Verilog/VHDL:**

* Sinh viên sử dụng ngôn ngữ **Verilog hoặc VHDL** để viết mã cho bộ mã hóa và giải mã.
* Sử dụng các công cụ thiết kế FPGA như **Xilinx Vivado** hoặc **Altera Quartus** để biên dịch và mô phỏng thiết kế.

**3. Tối ưu hóa:**

* Sau khi thiết kế và mô phỏng hệ thống thành công, sinh viên cần tối ưu hóa hệ thống trên FPGA để giảm tài nguyên và cải thiện hiệu suất.

# Câu hỏi 13: Thiết kế và triển khai bộ mã hóa Reed-Solomon trên FPGA

* **Yêu cầu**: Sinh viên cần thiết kế và triển khai **bộ mã hóa Reed-Solomon** trên FPGA với các tham số (ví dụ ), trong đó là số ký tự mã hóa và là số ký tự dữ liệu.
* **Mục tiêu**:
  1. Nghiên cứu nguyên tắc của mã Reed-Solomon, cách tạo các bit kiểm tra (parity).
  2. Viết mã **Verilog hoặc VHDL** để thiết kế bộ mã hóa Reed-Solomon.
  3. Triển khai thiết kế trên FPGA và mô phỏng quá trình mã hóa.
* **Câu hỏi chi tiết**:
  1. Trình bày nguyên tắc hoạt động của mã Reed-Solomon và cách tính các bit kiểm tra.
  2. Viết mã Verilog để thực hiện bộ mã hóa Reed-Solomon .
  3. Thực hiện mô phỏng trên FPGA và kiểm tra kết quả mã hóa với các chuỗi dữ liệu đầu vào khác nhau.
  4. Sử dụng công cụ **timing analysis** và **resource utilization report** để đánh giá tài nguyên sử dụng của thiết kế mã Hamming.
  5. Đề xuất và triển khai các kỹ thuật tối ưu hóa (ví dụ: pipeline, giảm độ sâu logic) để cải thiện hiệu suất.
  6. Thực hiện so sánh trước và sau khi tối ưu về mức độ sử dụng tài nguyên và thời gian trễ.

# Câu hỏi 14: Thiết kế bộ giải mã Reed-Solomon với khả năng phát hiện và sửa lỗi

* **Yêu cầu**: Sinh viên cần thiết kế **bộ giải mã Reed-Solomon** có khả năng phát hiện và sửa lỗi nhiều ký tự (multi-symbol error) dựa trên kỹ thuật **Berlekamp-Massey** hoặc phương pháp **Euclid mở rộng**.
* **Mục tiêu**:
  1. Nghiên cứu thuật toán giải mã Reed-Solomon, bao gồm phát hiện lỗi và sửa lỗi.
  2. Viết mã Verilog/VHDL để thiết kế bộ giải mã Reed-Solomon.
  3. Mô phỏng quá trình giải mã khi dữ liệu đầu vào chứa lỗi và kiểm tra kết quả.
* **Câu hỏi chi tiết**:
  1. Giải thích quá trình phát hiện và sửa lỗi của bộ giải mã Reed-Solomon. Tại sao thuật toán Berlekamp-Massey được sử dụng?
  2. Viết mã Verilog để thực hiện bộ giải mã Reed-Solomon. Đảm bảo tính năng sửa lỗi nhiều ký tự.
  3. Mô phỏng và kiểm tra hệ thống giải mã với các chuỗi dữ liệu có lỗi, sau đó phân tích hiệu quả sửa lỗi của hệ thống.
  4. Sử dụng công cụ **timing analysis** và **resource utilization report** để đánh giá tài nguyên sử dụng của thiết kế mã Hamming.
  5. Đề xuất và triển khai các kỹ thuật tối ưu hóa (ví dụ: pipeline, giảm độ sâu logic) để cải thiện hiệu suất.
  6. Thực hiện so sánh trước và sau khi tối ưu về mức độ sử dụng tài nguyên và thời gian trễ.

Báo cáo và thuyết trình kết quả

* Sinh viên cần chuẩn bị một **báo cáo chi tiết** về quá trình thiết kế, mô phỏng và tối ưu hóa hệ thống Reed-Solomon.
* **Nội dung báo cáo**:
  1. Giới thiệu về mã Reed-Solomon và ứng dụng của nó trong thực tiễn.
  2. Chi tiết về cách thực hiện bộ mã hóa và giải mã Reed-Solomon trên FPGA.
  3. Kết quả mô phỏng quá trình mã hóa và giải mã.
  4. Phân tích và so sánh kết quả trước và sau khi tối ưu hóa.
* **Thuyết trình**: Sinh viên cần chuẩn bị một buổi thuyết trình (~10 phút) về quá trình thực hiện bài tập lớn, bao gồm kết quả mô phỏng và tối ưu hóa.

Tiêu chí đánh giá

* **Thiết kế chính xác và hoàn chỉnh** của bộ mã hóa và giải mã Reed-Solomon (40%).
* **Tối ưu hóa thiết kế** và đánh giá hiệu suất (30%).
* **Mô phỏng và kiểm thử** với dữ liệu có lỗi (20%).
* **Báo cáo và thuyết trình** (10%).

**Hướng dẫn triển khai**:

**1. Nghiên cứu lý thuyết:**

* Sinh viên cần tìm hiểu về mã Reed-Solomon, đặc biệt là cách mã hóa, giải mã, và phát hiện/sửa lỗi.
* Tham khảo các tài liệu về **Berlekamp-Massey Algorithm** và **Extended Euclidean Algorithm** để thực hiện giải mã.

**2. Viết mã Verilog/VHDL:**

* Sinh viên sử dụng ngôn ngữ **Verilog hoặc VHDL** để viết mã cho bộ mã hóa và giải mã.
* Sử dụng các công cụ thiết kế FPGA như **Xilinx Vivado** hoặc **Altera Quartus** để biên dịch và mô phỏng thiết kế.

**3. Tối ưu hóa:**

* Sau khi thiết kế và mô phỏng hệ thống thành công, sinh viên cần tối ưu hóa hệ thống trên FPGA để giảm tài nguyên và cải thiện hiệu suất.

# Câu hỏi 15: Thiết kế hệ truyền tin bảo mật tối tiểu cho mã mạng dựa trên bài toán logarit rời rạc

* **Yêu cầu**: Sinh viên cần thiết kế **mô hình truyền tin bảo mật sử dụng mã mạng với tối thiểu ba nút mạng**, có khả năng bảo mật tối thiểu dựa trên **hệ mật ElGamal và dịch vòng**.
* **Mục tiêu**:
  1. Nghiên cứu giải pháp bảo mật mã mạng dựa trên bài toán logarit rời rạc, tập trung vào bảo mật tối thiểu (gợi ý kết hợp ElGamanl và dịch vòng).
  2. Viết mã Verilog/VHDL để thiết kế hệ thống truyền tin có 3 thiết bị.
  3. Mô phỏng quá trình tạo khóa, giải khóa và kiểm tra kết quả.
* **Câu hỏi chi tiết**:
  1. Giải thích quá trình mật mã và giải mật mã. Tại sao sử dụng kết hợp hai hệ mật trên?
  2. Viết mã Verilog để thực hiện. Đảm bảo tính năng bảo mật tối thiểu.
  3. Mô phỏng và kiểm tra hệ thống với các chuỗi dữ liệu bị tấn công (nếu có), sau đó phân tích hiệu quả bảo mật của hệ thống.
  4. Thực hiện so sánh trước và sau khi tối ưu về mức độ sử dụng tài nguyên và thời gian trễ (nếu có).

Báo cáo và thuyết trình kết quả

* Sinh viên cần chuẩn bị một **báo cáo chi tiết** về quá trình thiết kế, mô phỏng và tối ưu hóa mô hình.
* **Nội dung báo cáo**:
  1. Giới thiệu về bài toán logarit rời rạc và ứng dụng của nó trong thực tiễn.
  2. Giới thiệu về mã mạng và ứng dụng.
  3. Giới thiệu về các hệ mật sử dụng và ứng dụng của nó trong thực tiễn.
  4. Chi tiết về cách thực hiện mô hình (bộ mật mã và giải mật mã) trên FPGA.
  5. Kết quả mô phỏng quá trình mật mã và giải mật mã.
  6. Phân tích và so sánh kết quả trước và sau khi tối ưu hóa (nếu có).
* **Thuyết trình**: Sinh viên cần chuẩn bị một buổi thuyết trình (~10 phút) về quá trình thực hiện bài tập lớn, bao gồm kết quả mô phỏng và tối ưu hóa.

Tiêu chí đánh giá

* **Thiết kế chính xác và hoàn chỉnh** của bộ mã hóa và giải mã Reed-Solomon (40%).
* **Tối ưu hóa thiết kế** và đánh giá hiệu suất (30%).
* **Mô phỏng và kiểm thử** với dữ liệu có lỗi (20%).
* **Báo cáo và thuyết trình** (10%).

**Hướng dẫn triển khai**:

**1. Nghiên cứu lý thuyết:**

* Sinh viên cần tìm hiểu về mật mã, mã mạng, bài toán logarit rời rạc, đặc biệt là cách mật mã, giải mật mã sử dụng trong mô hình.
* Tham khảo các tài liệu về **Elgamal** và **Dịch vòng**.
* Tham khảo bài báo “**minimum security for network coding based on the discrete logarithm problem**”.

**2. Viết mã Verilog/VHDL:**

* Sinh viên sử dụng ngôn ngữ **Verilog hoặc VHDL** để viết mã cho mô hình bảo mật.
* Sử dụng các công cụ thiết kế FPGA như **Xilinx Vivado** hoặc **Altera Quartus** để biên dịch và mô phỏng thiết kế.

**3. Tối ưu hóa:**

* Sau khi thiết kế và mô phỏng hệ thống thành công, sinh viên cần tối ưu hóa hệ thống trên FPGA để giảm tài nguyên và cải thiện hiệu suất.

# Câu hỏi 16: Thiết kế và triển khai bộ mã hóa Cyclic cục bộ trên FPGA

* **Yêu cầu**: Sinh viên cần thiết kế và triển khai **bộ mã hóa Cyclic cục bộ** trên FPGA với các tham số (ví dụ ), trong đó là số ký tự mã hóa và là số ký tự dữ liệu, là khoảng cách hamming.
* **Mục tiêu**:
  1. Nghiên cứu nguyên lý của mã Cyclic cục bộ, cách tạo các bit kiểm tra (parity).
  2. Viết mã **Verilog hoặc VHDL** để thiết kế bộ mã hóa Cyclic cục bộ.
  3. Triển khai thiết kế trên FPGA và mô phỏng quá trình mã hóa.
* **Câu hỏi chi tiết**:
  1. Trình bày nguyên tắc hoạt động của mã Cyclic cục bộ.
  2. Viết mã Verilog để thực hiện bộ mã hóa Cyclic cục bộ .
  3. Thực hiện mô phỏng trên FPGA và kiểm tra kết quả mã hóa với các chuỗi dữ liệu đầu vào khác nhau.
  4. Sử dụng công cụ **timing analysis** và **resource utilization report** để đánh giá tài nguyên sử dụng của thiết kế mã.
  5. Đề xuất và triển khai các kỹ thuật tối ưu hóa (ví dụ: pipeline, giảm độ sâu logic) để cải thiện hiệu suất.
  6. Thực hiện so sánh trước và sau khi tối ưu về mức độ sử dụng tài nguyên và thời gian trễ.

# Câu hỏi 17: Thiết kế bộ giải mã Cyclic cục bộ với khả năng phát hiện và sửa lỗi

* **Yêu cầu**: Sinh viên cần thiết kế **bộ giải mã Cyclic cục bộ** có khả năng phát hiện và sửa lỗi nhiều ký tự (multi-symbol error) dựa trên kỹ thuật **Giải mã ngưỡng**.
* **Mục tiêu**:
  1. Nghiên cứu kỹ thuật giải mã Cyclic cục bộ sử dụng giải mã ngưỡng.
  2. Viết mã Verilog/VHDL để thiết kế bộ giải mã cyclic cục bộ.
  3. Mô phỏng quá trình giải mã khi dữ liệu đầu vào chứa lỗi và kiểm tra kết quả.
* **Câu hỏi chi tiết**:
  1. Giải thích quá trình sửa lỗi của bộ giải mã Cyclic cục bộ. Tại sao kỹ thuật Giải mã ngưỡng được sử dụng?
  2. Viết mã Verilog để thực hiện bộ giải mã Cyclic cục bộ. Đảm bảo tính năng sửa lỗi nhiều bit.
  3. Mô phỏng và kiểm tra hệ thống giải mã với các chuỗi dữ liệu có lỗi, sau đó phân tích hiệu quả sửa lỗi của hệ thống.
  4. Sử dụng công cụ **timing analysis** và **resource utilization report** để đánh giá tài nguyên sử dụng của thiết kế bộ giải mã.
  5. Đề xuất và triển khai các kỹ thuật tối ưu hóa (ví dụ: pipeline, giảm độ sâu logic) để cải thiện hiệu suất.
  6. Thực hiện so sánh trước và sau khi tối ưu về mức độ sử dụng tài nguyên và thời gian trễ.

Báo cáo và thuyết trình kết quả

* Sinh viên cần chuẩn bị một **báo cáo chi tiết** về quá trình thiết kế, mô phỏng và tối ưu hóa hệ thống Cyclic cục bộ.
* **Nội dung báo cáo**:
  1. Giới thiệu về mã Cyclic cục bộ và ứng dụng của nó trong thực tiễn.
  2. Chi tiết về cách thực hiện bộ mã hóa và giải mã Cyclic cục bộ trên FPGA.
  3. Kết quả mô phỏng quá trình mã hóa và giải mã.
  4. Phân tích và so sánh kết quả trước và sau khi tối ưu hóa.
* **Thuyết trình**: Sinh viên cần chuẩn bị một buổi thuyết trình (~10 phút) về quá trình thực hiện bài tập lớn, bao gồm kết quả mô phỏng và tối ưu hóa.

Tiêu chí đánh giá

* **Thiết kế chính xác và hoàn chỉnh** của bộ mã hóa và giải mã Reed-Solomon (40%).
* **Tối ưu hóa thiết kế** và đánh giá hiệu suất (30%).
* **Mô phỏng và kiểm thử** với dữ liệu có lỗi (20%).
* **Báo cáo và thuyết trình** (10%).

**Hướng dẫn triển khai**:

**1. Nghiên cứu lý thuyết:**

* Sinh viên cần tìm hiểu về mã Cyclic cục bộ, đặc biệt là cách mã hóa, giải mã, và phát hiện/sửa lỗi.
* Tham khảo các tài liệu về **Giải mã ngưỡng** để thực hiện giải mã.

**2. Viết mã Verilog/VHDL:**

* Sinh viên sử dụng ngôn ngữ **Verilog hoặc VHDL** để viết mã cho bộ mã hóa và giải mã.
* Sử dụng các công cụ thiết kế FPGA như **Xilinx Vivado** hoặc **Altera Quartus** để biên dịch và mô phỏng thiết kế.

**3. Tối ưu hóa:**

* Sau khi thiết kế và mô phỏng hệ thống thành công, sinh viên cần tối ưu hóa hệ thống trên FPGA để giảm tài nguyên và cải thiện hiệu suất.

# Bài 18: Xây dựng thuật toán mã hóa và giải mã Haffman trên FPGA

Xem tài liệu từ giáo trình Lý thuyết thông tin.

# Bài 19: Thiết kế và mô tả đồng hồ thời gian thực (Ngày trong tuần-Giờ-Phút-Giây).

# Bài 20: Thiết kế và mô tả mạch hiển thị dòng chữ “KTDT PTIT” trên ma trận LED với một số hiệu ứng