|  |
| --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  logo_128  BÁO CÁO BÀI TẬP LỚN  **THIẾT KẾ VLSI**  **Đề tài:**  **Thiết kế hệ thống mã hóa sử dụng mã Hamming**  Sinh viên thực hiện: Nguyễn Quang Khải 20172615  Bùi Đức Phương 20172762  Nguyễn Tiến Thành 20209574  Giảng viên hướng dẫn: TS. Phan Xuân Vũ  Hà Nội, 1-2021 |

**MỤC LỤC**

[DANH MỤC HÌNH VẼ i](#_Toc61938096)

[DANH MỤC BẢNG BIỂU i](#_Toc61938097)

[CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG 2](#_Toc61938098)

[1.1 Tổng quan về hệ thống 2](#_Toc61938099)

[1.1.1 Mã hóa Hamming 2](#_Toc61938100)

[1.2 Mô tả và phân rã hệ thống 3](#_Toc61938101)

[1.2.1 Mô tả bài toán 3](#_Toc61938102)

[1.2.2 Lưu đồ thuật toán 4](#_Toc61938103)

[1.2.3 Sơ đồ khối thiết kế 5](#_Toc61938104)

[CHƯƠNG 2. SỬ DỤNG PHẦN MỀM SYNOPSYS ĐỂ SYNTHESIZE VÀ LAYOUT 9](#_Toc61938105)

[2.1 Sử dụng phần mềm Synopsys Design Compiler 9](#_Toc61938106)

[2.2 Sử dụng phần mềm Synopsys IC Compiler 12](#_Toc61938107)

[KẾT LUẬN 17](#_Toc61938108)

[TÀI LIỆU THAM KHẢO 17](#_Toc61938109)

# DANH MỤC HÌNH VẼ

[Hình 1.1: Sơ đồ bài toán 4](#_Toc61937196)

[Hình 1.2: Lưu đồ thuật toán hệ thống 5](#_Toc61937197)

[Hình 1.3: Sơ đồ khối hệ thống 5](#_Toc61937198)

[Hình 1.4: Khối Top 6](#_Toc61937199)

[Hình 1.5: Mô phỏng khối Top bằng ModelSim 7](#_Toc61937200)

[Hình 1.6: Khối Selector In 7](#_Toc61937201)

[Hình 1.7: Khối Selector Out 8](#_Toc61937202)

[Hình 1.8: Khối mã hóa Hamming thứ N 9](#_Toc61937203)

[Hình 2.1: Sơ đồ RTL của hệ thống 10](#_Toc61937204)

[Hình 2.2: Sơ đồ sau khi Compile của hệ thống 11](#_Toc61937205)

[Hình 2.3: Lưu đồ quy trình thiết kế của ICC 12](#_Toc61937206)

[Hình 2.4: Chuẩn bị cho Floorplaning 13](#_Toc61937207)

[Hình 2.5: Floorplan sau khi đã tạo các vòng dây nguồn 14](#_Toc61937208)

[Hình 2.6: Layout cell-level của mạch 15](#_Toc61937209)

[Hình 2.7: Layout transistor-level của mạch 16](#_Toc61937210)

# DANH MỤC BẢNG BIỂU

[Bảng 1.1: Mô tả khối Top 8](#_Toc61937260)

[Bảng 1.2: Mô tả khối Selector In 9](#_Toc61937261)

[Bảng 1.3: Mô tả khối Selector Out 9](#_Toc61937262)

[Bảng 1.4: Mô tả khối Hamming 10](#_Toc61937263)

# CHƯƠNG 1: TỔNG QUAN VỀ HỆ THỐNG

## Tổng quan về hệ thống

Hệ thống sẽ có đầu vào là chuỗi bit X, cho đầu ra là chuỗi bit Y, được mã hóa theo thuật toán Hamming với 3 tỉ lệ bit mã hóa. Tỉ lệ bit này sẽ được điều chỉnh thông qua một đầu vào khác. Như vậy, ta có thể xác định rằng, hệ thống sẽ gồm 2 thành phần: khối lựa chọn tín hiệu vào/ra, và khối mã hóa tín hiệu

### Mã hóa Hamming

Trong viễn thông (telecommunication), mã Hamming là một mã sửa lỗi tuyến tính (linear error-correcting code), được đặt tên theo tên của người phát minh ra nó, Richard Hamming. Mã Hamming có thể phát hiện một bit hoặc hai bit bị lỗi (single and double-bit errors). Mã Hamming còn có thể sửa các lỗi do một bit bị sai gây ra. Ngược lại với mã của ông, mã chẵn lẻ (parity code) đơn giản vừa không có khả năng phát hiện các lỗi khi 2 bit cùng một lúc bị hoán vị (0 thành 1 và ngược lại), vừa không thể giúp để sửa được các lỗi mà nó phát hiện được.

Thuật toán cho việc sử dụng bit chẵn lẻ trong 'mã Hamming' *thông thường* cũng tương đối đơn giản:

1. Tất cả các bit ở vị trí là các số mũ của 2 (*powers of two*) được dùng làm bit chẵn lẻ. (các vị trí như 1, 2, 4, 8, 16, 32, 64 v.v. hay nói cách khác 20, 21, 22, 23, 24, 25, 26 v.v.)
2. Tất cả các vị trí bit khác được dùng cho dữ liệu sẽ được mã hóa. (các vị trí 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)
3. Mỗi bit chẵn lẻ tính giá trị chẵn lẻ cho một số bit trong từ mã (*code word*). Vị trí của bit chẵn lẻ quyết định chuỗi các bit mà nó luân phiên kiểm tra và bỏ qua (*skips*).
   * Vị trí 1 (n=1): bỏ qua 0 bit(n-1), kiểm 1 bit(n), bỏ qua 1 bit(n), kiểm 1 bit(n), bỏ qua 1 bit(n), v.v.
   * Vị trí 2(n=2): bỏ qua 1 bit(n-1), kiểm 2 bit(n), bỏ qua 2 bit(n), kiểm 2 bit(n), bỏ qua 2 bit(n), v.v.
   * Vị trí 4(n=4): bỏ qua 3 bit(n-1), kiểm 4 bit(n), bỏ qua 4 bit(n), kiểm 4 bit(n), bỏ qua 4 bit(n), v.v.
   * Vị trí 8(n=8): bỏ qua 7 bit(n-1), kiểm 8 bit(n), bỏ qua 8 bit(n), kiểm 8 bit(n), bỏ qua 8 bit(n), v.v.
   * Vị trí 16(n=16): bỏ qua 15 bit(n-1), kiểm 16 bit(n), bỏ qua 16 bit(n), kiểm 16 bit(n), bỏ qua 16 bit(n), v.v.
   * Vị trí 32(n=32): bỏ qua 31 bit(n-1), kiểm 32 bit(n), bỏ qua 32 bit(n), kiểm 32 bit(n), bỏ qua 32 bit(n), v.v.
   * và tiếp tục như trên.

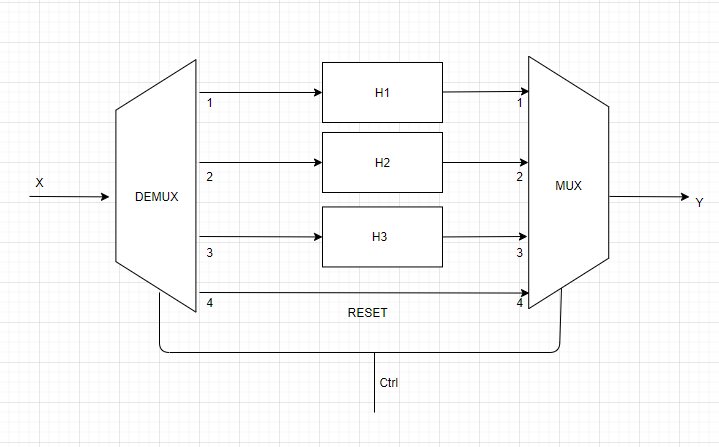
Nói cách khác, bit chẵn lẻ tại vị trí 2k kiểm các bit ở các bit ở vị trí t có giá trị logic của phép toán AND giữa k và t là khác 0

Trong phạm vi của hệ thống này, chúng em sẽ sử dụng Hamming (3,1), (7,4) và (15,11).

## Mô tả và phân rã hệ thống

### Mô tả bài toán

Xét về bài toán, đầu tiên ta sẽ có một khối mã hóa Hamming lý tưởng. Khối mã hóa này sẽ gồm 3 khối con, tương ứng với 3 tỉ lệ bit mã hóa. Tiếp theo, đầu vào sẽ qua một khối lựa chọn tín hiệu vào, để được đưa đến khối mã hóa có tỉ lệ thỏa mãn. Sau khi mã hóa xong, chuỗi bit đầu ra sẽ được đưa qua một khối lựa chọn tín hiệu ra, lúc này để nhận biết rằng chuỗi bit đầu ra là hợp lệ, và được đưa ra ngoài đầu ra. Một đầu vào khác sẽ có tác dụng để lựa chọn tỉ lệ mã hóa.



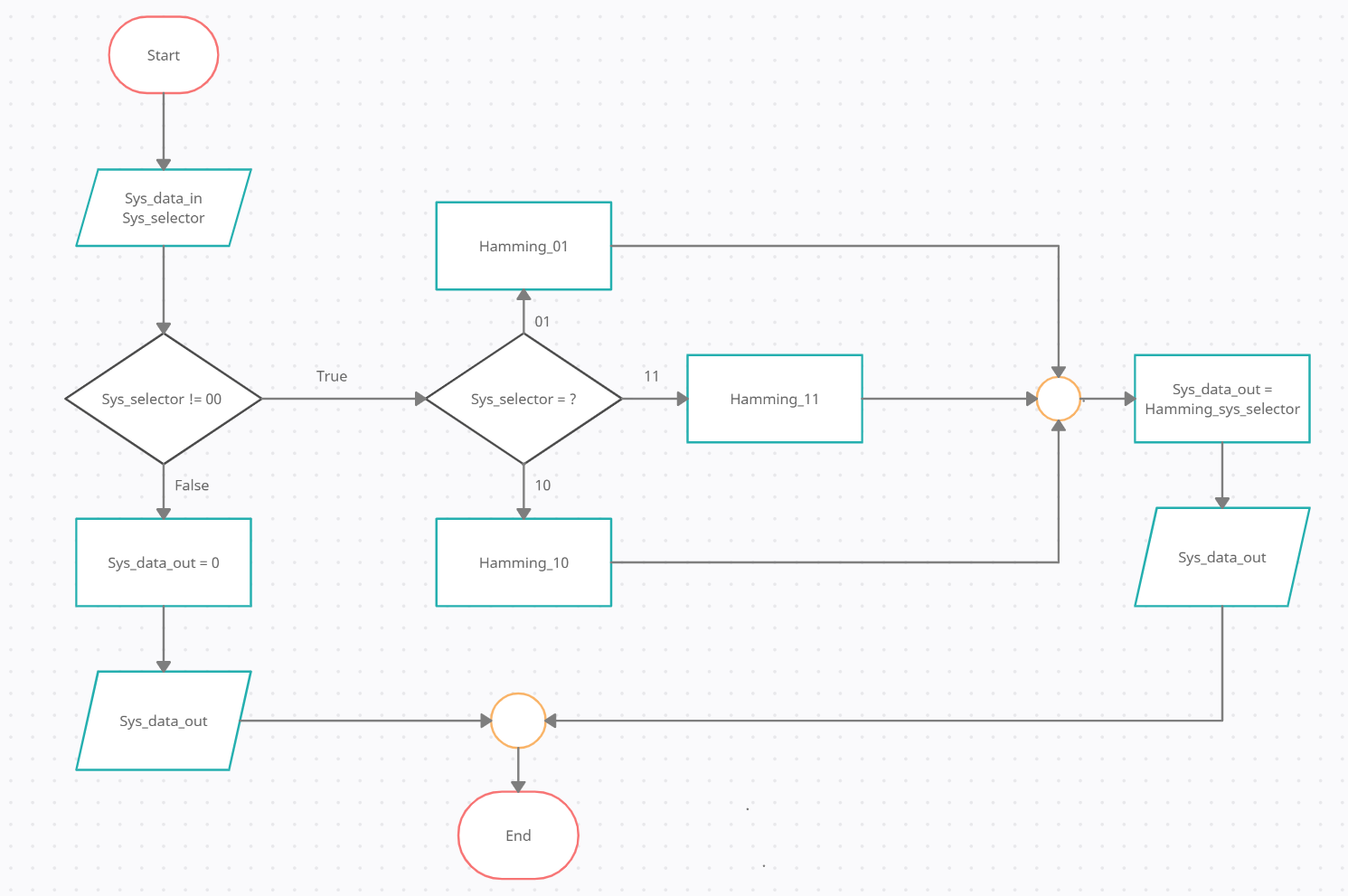
Hình 1.1: Sơ đồ bài toán

Như vậy, ta có thể xác định được tổng quan hệ thống thông qua sơ đồ sau:

* Bộ Demux 1-4 sẽ có tác dụng đưa chuỗi đầu vào đến một trong ba khối giải mã H1, H2, H3 tương ứng với ba tỉ lệ mã hóa, phụ thuộc vào tín hiệu của chân điều khiển CTRL.
* Sau khi mã hóa xong, chuỗi đầu ra sẽ được đưa qua bộ Mux 4-1, cũng được điều khiển bởi CTRL để xuất ra đầu ra.
* Reset sẽ có tác dụng set trực tiếp bit đầu ra của chuỗi bằng 0, không thông qua bộ giải mã.
* Giá trị của tín hiệu CTRL được mô tả như sau:
  + 00: RESET
  + 01: bộ mã hóa Hamming 1
  + 10: bộ mã hóa Hamming 2
  + 11: bộ mã hóa Hamming 3

### Lưu đồ thuật toán

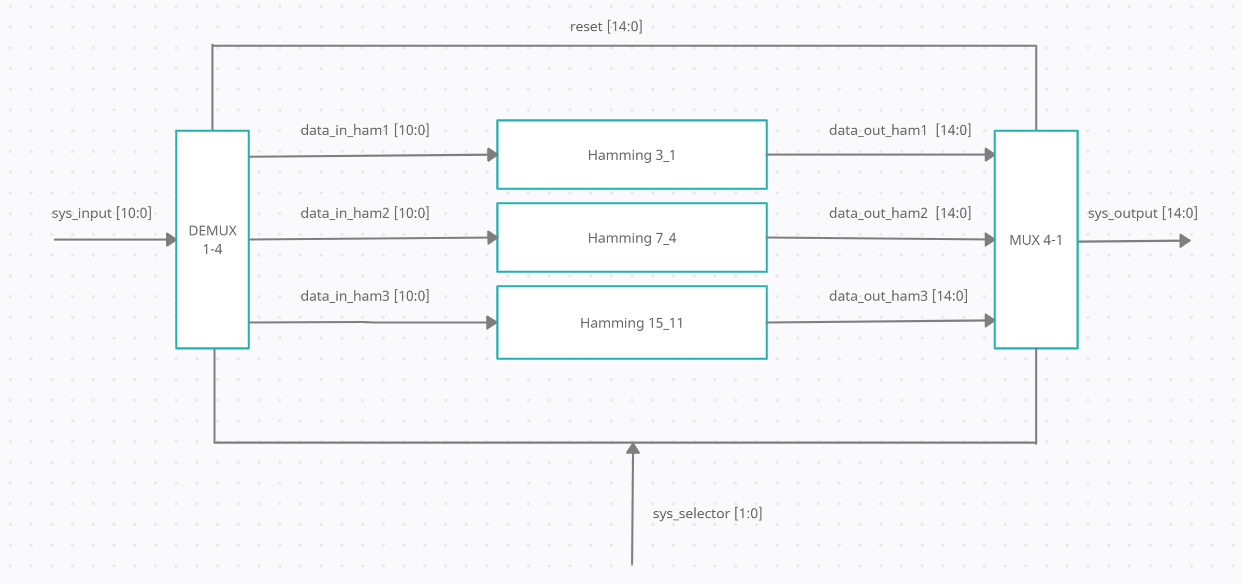
Dựa vào mô tả bài toán ở mục trên, ta có thể dựng lưu đồ thuật toán hệ thống của ta như sau:



Hình 1.2: Lưu đồ thuật toán hệ thống

### Sơ đồ khối thiết kế

Từ lưu đồ, ta triển khai thuật toán trên bằng mô hình FSMD, nhận được kiến trúc của bộ mã hóa Hamming như sau:

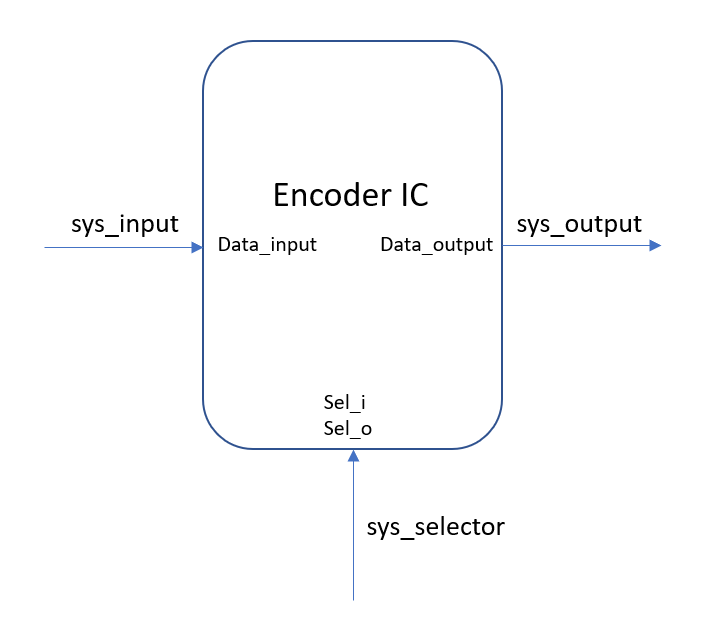


Hình 1.3: Sơ đồ khối hệ thống

Sơ đồ trên bao gồm các module:

1. Demux – Input Selector: Tác dụng lựa chọn khối mã hóa phù hợp cho chuỗi đầu vào
2. Hamming N: Tương úng với khối mã hóa Hamming có tỉ lệ bit thứ N
3. Mux – Output Selector: Tác dụng lựa chọn đầu ra phù hợp từ các khối mã hóa Hamming

#### Top Module

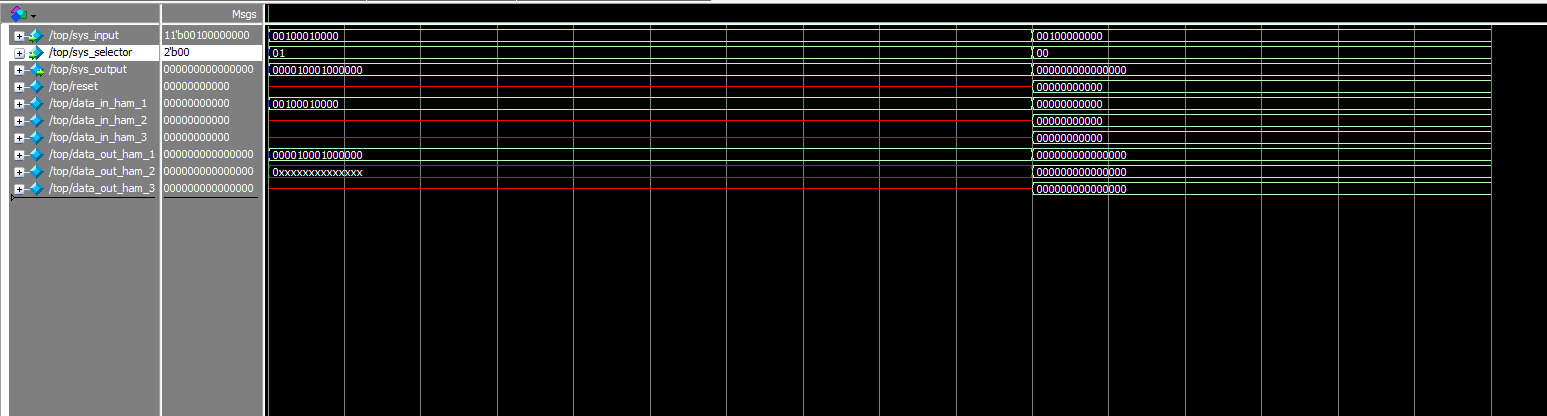


Hình 1.4: Khối Top

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên** | **Độ rộng** | **Vị trí vào** | **Vị trí ra** | **Mô tả** |
| sys\_input | 10 | Bên ngoài hệ thống | Khối selector input | Đường tín hiệu đầu vào |
| sys\_selector | 2 | Bên ngoài hệ thống | Khối selector input và output | Đường tín hiệu lựa chọn tỉ lệ mã hóa / Reset |
| sys\_output | 14 | Khối selector output | Bên ngoài hệ thống | Đường tín hiệu đầu ra |

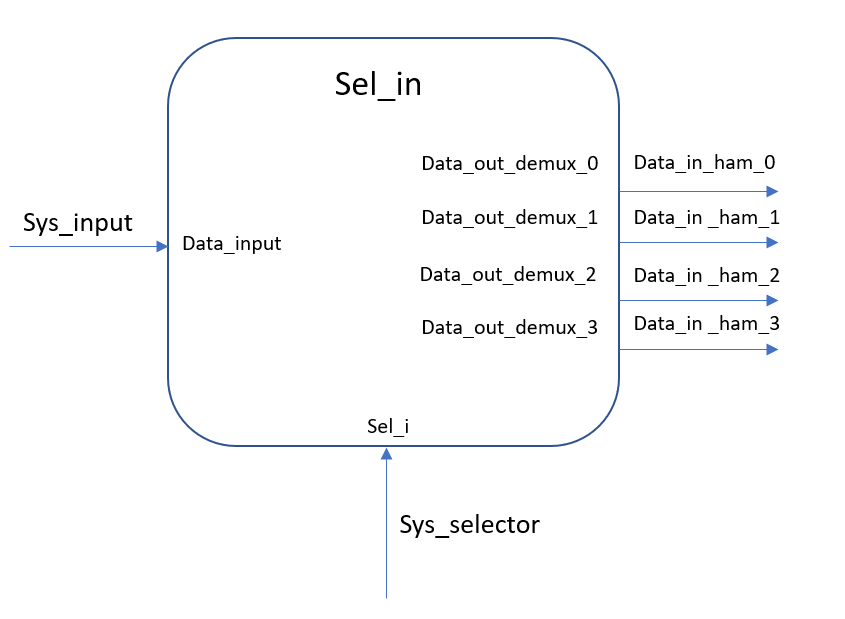
Bảng 1.1: Mô tả khối Top

Sau khi hoàn thành mạch, ta chạy mô phỏng trên ModelSim, và được kết quả như sau:



Hình 1.5: Mô phỏng khối Top bằng ModelSim

#### Selector\_in Module

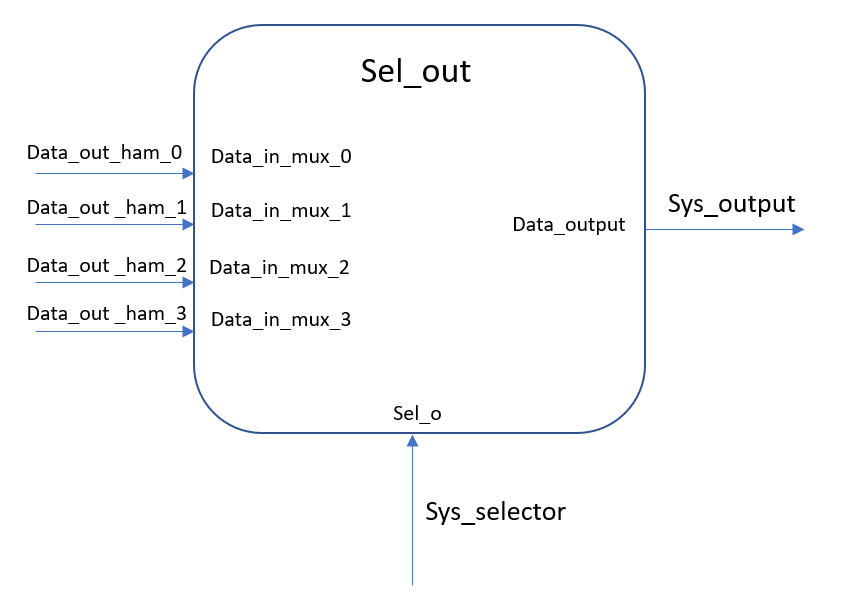


Hình 1.6: Khối Selector In

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Tên chân*** | ***I/O*** | ***Độ rộng (bit)*** | ***Tích cực*** | ***Mô tả*** |
| data\_input | I | 11 | H | Tín hiệu đầu vào |
| sel\_i | I | 1 | H | Đường tín hiệu lựa chọn tỉ lệ mã hóa / Reset |
| data\_out\_demux [N] | I | 11 | H | Tín hiệu đầu ra thứ N của selector |

Bảng 1.2: Mô tả khối Selector In

#### Selector\_out Module

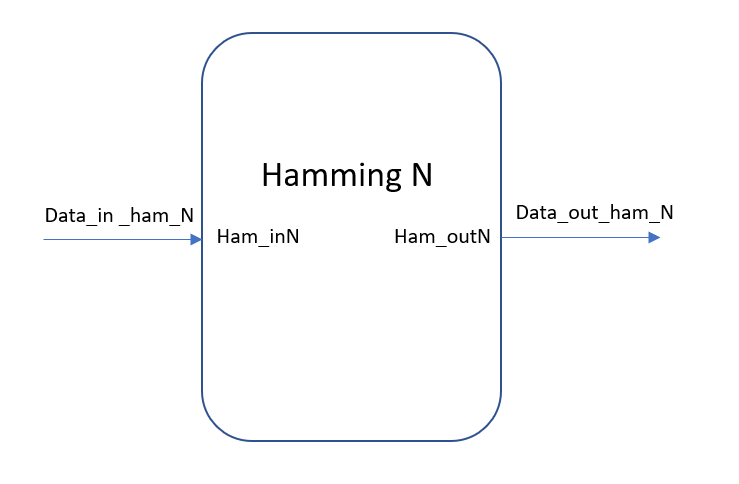


Hình 1.7: Khối Selector Out

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Tên chân*** | ***I/O*** | ***Độ rộng (bit)*** | ***Tích cực*** | ***Mô tả*** |
| data\_output | I | 11 | H | Tín hiệu đầu ra |
| sel\_o | I | 1 | H | Đường tín hiệu lựa chọn tỉ lệ mã hóa / Reset |
| data\_in\_mux [N] | I | 15 | H | Tín hiệu đầu vào thứ N của selector |

Bảng 1.3: Mô tả khối Selector Out

#### Hamming N Module



Hình 1.8: Khối mã hóa Hamming thứ N

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Tên chân*** | ***I/O*** | ***Độ rộng (bit)*** | ***Tích cực*** | ***Mô tả*** |
| Ham\_inN | I | 11 | H | Đầu vào của khối mã hóa Hamming số N |
| Ham\_outN | I | 15 | H | Đầu ra của khối mã hóa Hamming số N |

Bảng 1.4: Mô tả khối Hamming

# SỬ DỤNG PHẦN MỀM SYNOPSYS ĐỂ SYNTHESIZE VÀ LAYOUT

## Sử dụng phần mềm Synopsys Design Compiler

Tạo thư mục “dc” trong project, vào terminal gõ lệnh:

*design\_vision*

Các bước sử dụng phần mềm Synopsys có thể được tóm tắt như sau:

***Bước 1: Cài đặt thư viện***

***Bước 2 : Tạo file theo dõi quá trình có đuôi mở rộng “.svf”.***

***Bước 3: đọc file \*.v***

File → Read

***Bước 4 : Analyze***

File → Analyze (trừ file testbench)

***Bước 5: Elaborate***

File → Elaborate

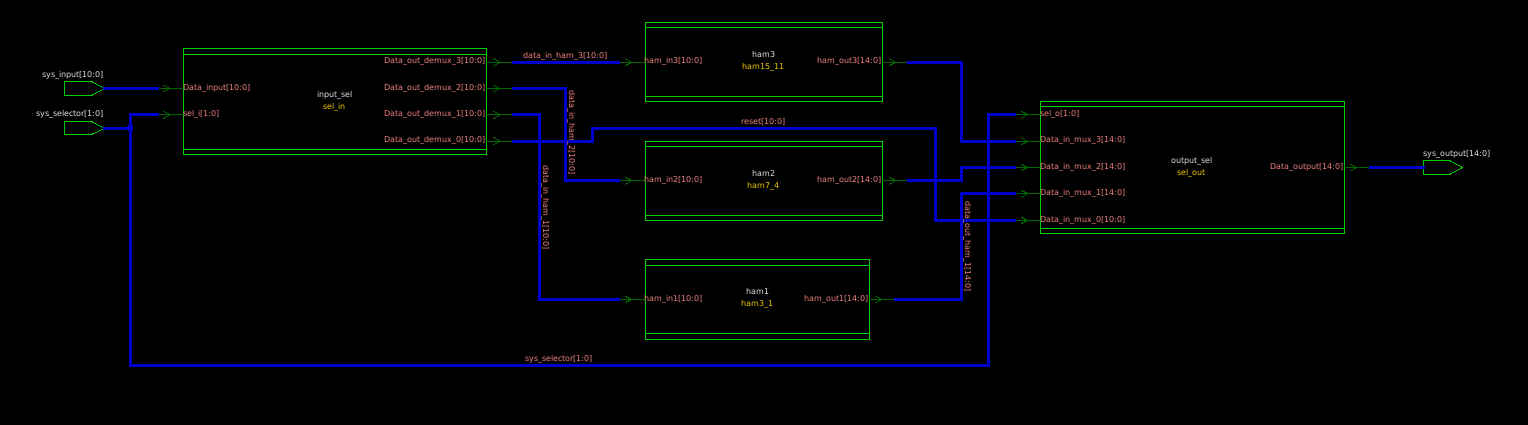
Trong mục Design chọn file top module

***Bước 6: Kiểm tra thiết kế***

Design → Check Design → OK

Bước này sẽ kiểm tra thiết kế. Trên cửa sổ terminal sẽ có các cảnh báo nên đọc kỹ.

Lúc này, một sơ đồ mạch RTL của hệ thống sẽ được tạo ra:



Hình 2.1: Sơ đồ RTL của hệ thống

***Bước 7: Thiết lập các ràng buộc thiết kế (constraint)***

* Trễ đầu vào: Chọn các tín hiệu đầu vào trên sơ đồ (trừ tín hiệu clock)

Trên thanh Menu vào “Atributes” chọn Operating Environment → Input delay

* Trễ đầu ra : Chọn các tín hiệu ra trên sơ đồ

Trên thanh Menu vào “Atributes” chọn Operating Environment → Output delay

* Điều kiện hoạt động (Operating condition)

Trên thanh Menu vào “Atributes” chọn Operating Environment →Operating conditions

* Wire load:

Trên thanh Menu vào “Atributes” chọn Operating Environment →Wire load

* Thiết lập điện trở, điện dung đầu vào, đầu ra cho IC:

Khi các cổng kích thích đầu vào và tải đầu ra chưa biết thì ta dùng Load Budget, xác định dự phòng cho thiết kế.

* Design Constraints

Trên thanh Menu vào “Atributes” chọn Operating Environment →Design constraints

***Bước 9 : Compile Design***

Trên thanh Menu vào Design → Compile Design → OK

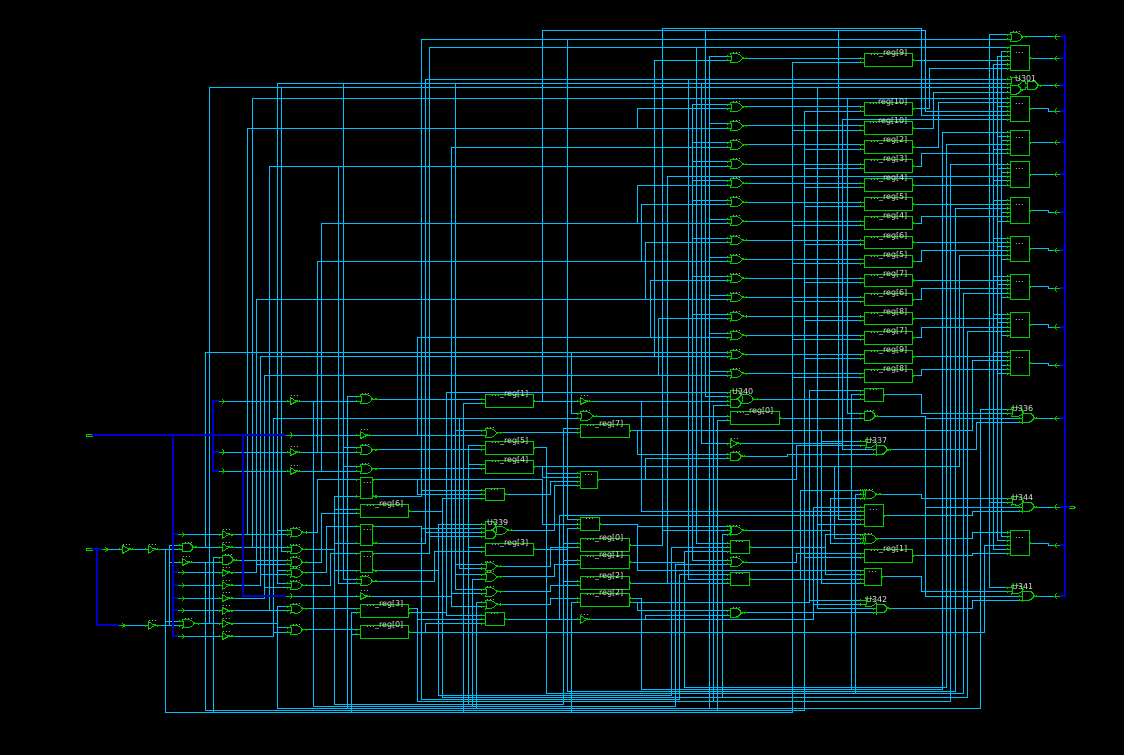
Ngoài ra bạn có thêm các lựa chọn sao cho sau quá trình tổng hợp thiết kế của mình tối ưu nhất.

***Bước 10: Compile Ultra***

Sau quá trình tổng hợp thiết kế, bước này sẽ vừa tổng hợp vừa tối ưu hóa thiết kế những cổng nào thừa hoặc không cần thiết trong quá trình tổng hợp sẽ bị xóa.

Trên thanh Menu vào Design → Compile Ultra → OK

Sau quá trình tổng hợp khi xem lại sơ đồ mạch sẽ trở nên phức tạp hơn rất nhiều:



Hình 2.2: Sơ đồ sau khi Compile của hệ thống

## Sử dụng phần mềm Synopsys IC Compiler

Phần IC Complier là phần mềm có chức năng layout tạo ra IC dựa vào file nestlist (.ddc), (.sdc) và các thư viện vật lý (milkyway techfile). Quy trình thiết kế của IC Complier được biểu diễn như sau :



Hình 2.3: Lưu đồ quy trình thiết kế của ICC

***Bước 1: Cài đặt dữ liệu***

* Mở cửa sổ giao diện người dung bằng cách gõ lệnh sau:

icc\_shell -gui

* Cài đặt thư viện

File → Setup → Application Setup

Thực hiện tương tự như bước cài đặt thư viện trong Design Compiler.

* Tạo thư viện milkyway

File → Creat Library

* Đọc file “filter.ddc”

File → Import → Read DDC

* Đọc file “file.sdc”

File SDC là file ràng buộc thiết kế. Nó được đưa vào để ràng buộc thiết kế sao cho thiết kế của mình tối ưu nhất.

File → Import → Read SDC

Sau khi đọc file “file.sdc”, chọn Window → New layout window, sẽ xuất hiện cửa sổ giao diện hiển thị các cell được xếp chồng lên nhau.



Hình 2.4: Chuẩn bị cho Floorplaning

***Bước 2: Floor Planning***

* Thực hiện khởi tạo nền

Floorplan → Initialize Floorplan

* Thực hiện tạo chân nối nguồn và nối đất VDD và VSS

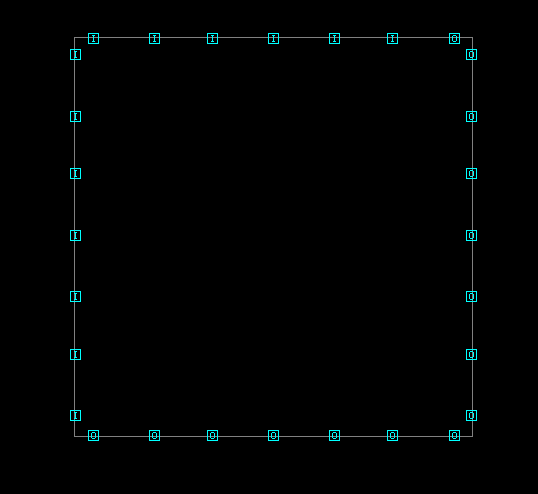
PreRoute → Derive PG Conection

* Tạo các vòng dây VDD và VSS

Preroute → Creating Rings → Net (chọn VDD)

* Tiếp theo chọn VSS.

Sau khi tạo được các vòng dây VDD và VSS trên hình sẽ xuất hiện như sau:



Hình 2.5: Floorplan sau khi đã tạo các vòng dây nguồn

* Thiết lập độ rộng cho dây nguồn dây đất cho từng lớp kim loại:

Preroute → Creat power strap

* Tạo vòng đệm:

Preroute → Creat Pad ring

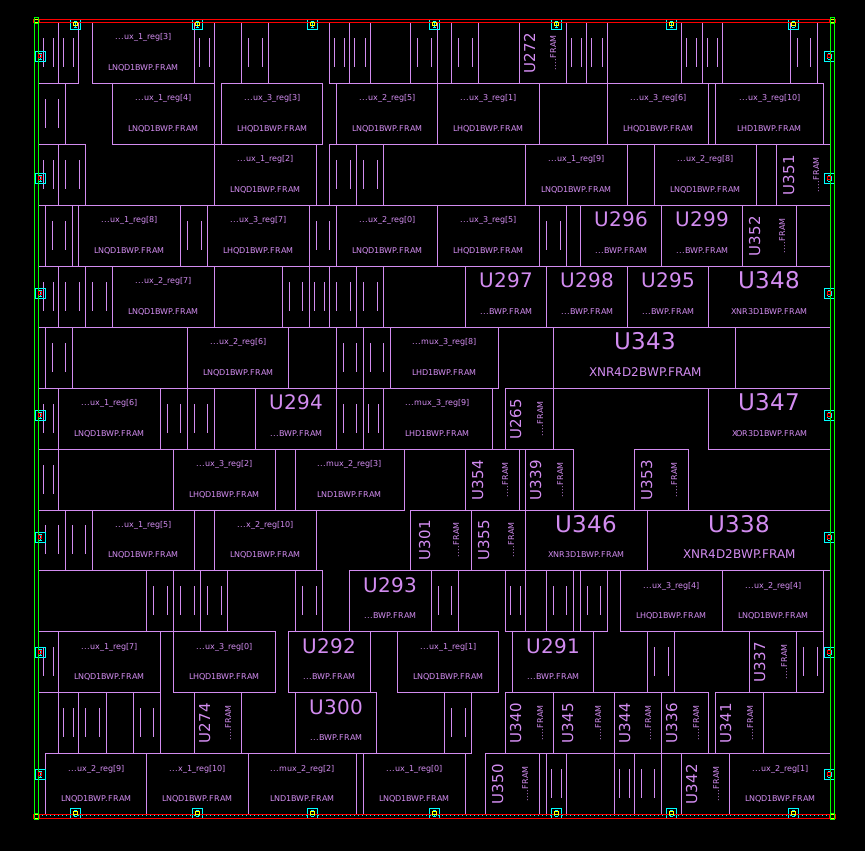
Sau khi xong bước Floor Planning ta cần lưu lại sử dụng câu lệnh

*save\_mw\_cel -as floorplanned*

***Bước 3: Placement (sắp xếp cell) và kiểm tra tắc nghẽn***

* Placement → Core Placement and Optimization

Sau khi sắp xếp các cell xong ta có:



Hình 2.6: Layout cell-level của mạch

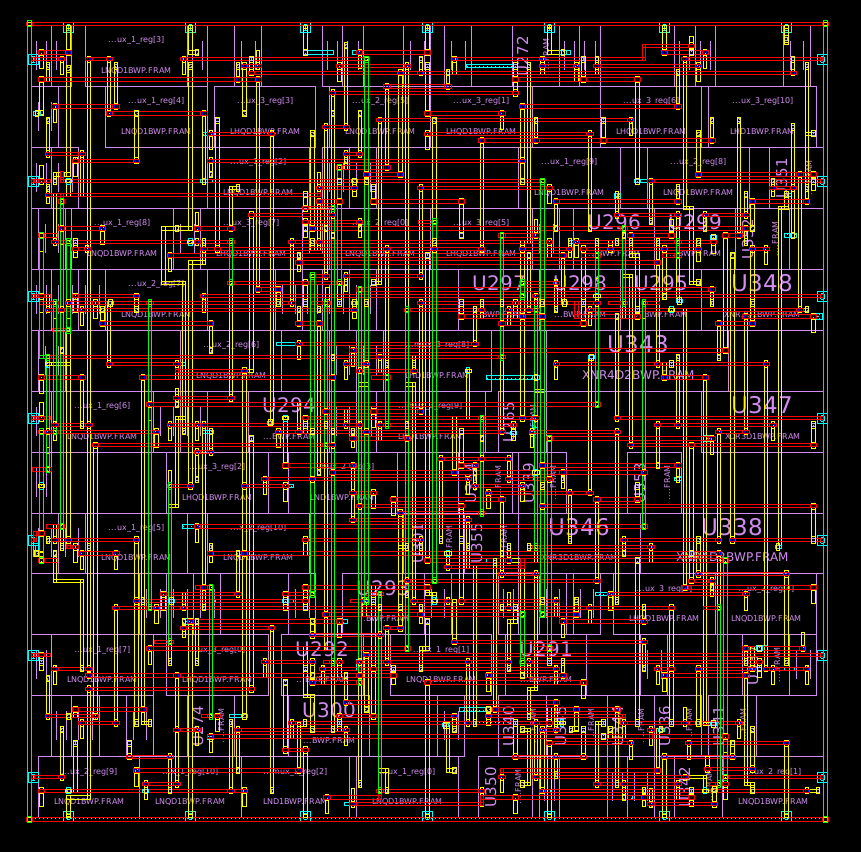
***Bước 4: Thực hiện đi dây***

* Route → Core Routing and Optimization

Lưu lại kết quả sau khi đi dây tự động

*save\_mw\_cel -as routed*

* Route → Extract RC
* Kết quả sau khi ICC



Hình 2.7: Layout transistor-level của mạch

# KẾT LUẬN

Qua quá trình tìm hiểu, thực hiện đề tài, nhóm chúng em đã thu được các kết quả chính như sau:

* Tìm hiểu, thực hành và mô phỏng hệ thống số bằng ngôn ngữ Verilog
* Tìm hiểu, thiết kế hệ thống số dựa trên thuật toán cho trước
* Tìm hiểu, làm quen và thực hành với các phần mềm thiết kế IC ASIC của Synopsys (Design Compiler, IC Compiler)

Tuy nhiên, vì gặp giới hạn về mặt thời gian nên trong phần chính của báo cáo chính em chỉ xoay quanh thiết kế hệ thống ở mức schematic level, một phần layout đơn giản, và chưa kiểm thử thiết kế bằng các phần mềm chuyên dụng (Leda, Primetime, VCS, …)

Nhóm em rất mong nhận được những nhận xét đánh giá từ thầy để có được những kiến thức mà nhóm em chưa chú trọng hoặc bỏ qua, hay là có nhầm lẫn.

*Chúng em xin chân thành cảm ơn*

# TÀI LIỆU THAM KHẢO

1. N. Weste, *CMOS VLSI Design: A Circuits and Systems Perspective (4th Edition)*. Prentice-Hall, 2008.
2. J.Baker, *CMOS: Circuit Design, Layout, and Simulation (4th Edition).* (2012)