

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**  
**TRƯỜNG ĐIỆN – ĐIỆN TỬ**



**BÀI TẬP LỚN**  
**THIẾT KẾ VLSI**

**Đề tài:**

**THIẾT KẾ VÀ TỔNG HỢP HỆ THỐNG VITERBI  
DECODER  
TỪ RTL ĐẾN GDSII SỬ DỤNG OPENLANE FLOW**

**Giảng viên hướng dẫn:** TS. Nguyễn Vũ Thắng

**Sinh viên thực hiện:** Phạm Chí Dũng – 20200106

Võ Ngọc Vinh – 20227447

Nguyễn Văn Dương – 20241713E

**Lớp:** 163187 – ET4340

Hà Nội, 2026

# TÓM TẮT NỘI DUNG

Báo cáo này trình bày quy trình toàn diện về thiết kế, kiểm chứng và thực thi vật lý cho một hệ thống giải mã Viterbi (Viterbi Decoder SoC). Hệ thống được xây dựng hoàn chỉnh với các khối giao tiếp ngoại vi (FIFO, PISO, SIPO) để xử lý dữ liệu dòng. Điểm nổi bật của thiết kế là việc áp dụng kiến trúc **Register Exchange (RE)** trong đơn vị truy vết, giúp tối ưu hóa băng thông xử lý và giảm độ trễ so với phương pháp Traceback truyền thống.

Kết quả kiểm tra chức năng (Functional Verification) cho thấy mạch hoạt động chính xác tuyệt đối với bộ dữ liệu kiểm thử. Quy trình thiết kế vật lý (Physical Design) được thực hiện bằng công cụ mã nguồn mở **OpenLane Flow** với công nghệ Sky130, đạt kết quả khả quan về diện tích ( $0.16mm^2$ ), công suất tiêu thụ thấp và đáp ứng tốt các yêu cầu về thời gian (Timing) ở tần số 50MHz.

# Mục lục

|   |           |
|---|-----------|
| <b>TÓM TẮT NỘI DUNG</b>   | <b>1</b>  |
| <b>1 TỔNG QUAN VỀ MÃ TÍCH CHẬP VÀ GIẢI MÃ VITERBI</b>                   | <b>5</b>  |
| 1.1 Giới thiệu mã tích chập . . . . .                                   | 5         |
| 1.2 Nguyên lý giải mã Viterbi . . . . .                                 | 5         |
| <b>2 ĐẶC TẢ KỸ THUẬT VÀ THIẾT KẾ RTL</b>                                | <b>6</b>  |
| 2.1 Đặc tả kỹ thuật (Specifications) . . . . .                          | 6         |
| 2.2 Kiến trúc hệ thống và Chi tiết I/O từng khối . . . . .              | 7         |
| 2.2.1 Khối đệm dữ liệu (FIFO) . . . . .                                 | 7         |
| 2.2.2 Khối chuyển đổi song song sang nối tiếp (PISO) . . . . .          | 8         |
| 2.2.3 Khối tính toán chi phí nhánh (BMU) . . . . .                      | 8         |
| 2.2.4 Khối Cộng - So sánh - Chọn (ACSU) . . . . .                       | 9         |
| 2.2.5 Khối quản lý Path Metric (PMU) . . . . .                          | 10        |
| 2.2.6 Khối truy vết Survivor Memory (TBU - Register Exchange) . . . . . | 11        |
| 2.2.7 Khối chuyển đổi nối tiếp sang song song (SIPO) . . . . .          | 12        |
| 2.3 Triển khai RTL (RTL Implementation) . . . . .                       | 13        |
| <b>3 KIỂM TRA CHỨC NĂNG MẠCH</b>  | <b>17</b> |
| 3.1 Môi trường kiểm chứng (Testbench Environment) . . . . .             | 17        |
| 3.2 Kết quả mô phỏng (Simulation Results) . . . . .                     | 17        |
| 3.2.1 Khối đệm dữ liệu (FIFO) . . . . .                                 | 17        |
| 3.2.2 Khối chuyển đổi song song sang nối tiếp (PISO) . . . . .          | 19        |
| 3.2.3 Khối tính toán nhánh (BMU) . . . . .                              | 21        |
| 3.2.4 Khối Cộng - So sánh - Chọn (ACSU) . . . . .                       | 23        |
| 3.2.5 Khối quản lý Metric (PMU) . . . . .                               | 25        |
| 3.2.6 Khối truy vết (TBU) . . . . .                                     | 26        |
| 3.2.7 Khối chuyển đổi nối tiếp sang song song (SIPO) . . . . .          | 28        |
| 3.2.8 Kiểm chứng toàn hệ thống (System Integration) . . . . .           | 29        |
| <b>4 TỔNG HỢP VÀ THIẾT KẾ VẬT LÝ</b>                                    | <b>33</b> |
| 4.1 Quy trình OpenLane Flow . . . . .                                   | 33        |
| 4.2 Kết quả tổng hợp (Synthesis Results) . . . . .                      | 33        |
| 4.3 Kết quả thiết kế vật lý (Physical Layout) . . . . .                 | 33        |
| <b>KẾT LUẬN</b>   | <b>35</b> |

# Danh sách hình vẽ

|      |  |    |
|------|--|----|
| 2.1  | Sơ đồ khối Top-Level System . . . . .  | 6  |
| 2.2  | Sơ đồ khối FIFO . . . . .  | 7  |
| 2.3  | Sơ đồ khối PISO . . . . .  | 8  |
| 2.4  | Sơ đồ khối BMU . . . . .   | 9  |
| 2.5  | Sơ đồ khối ACSU . . . . .  | 10 |
| 2.6  | Sơ đồ khối PMU . . . . .   | 11 |
| 2.7  | Sơ đồ khối TBU . . . . .   | 12 |
| 2.8  | Sơ đồ khối SIPO . . . . .  | 13 |
| 2.9  | Sơ đồ khối chi tiết hệ thống Viterbi Decoder SoC . . . . .                             | 14 |
| 2.10 | Lưu đồ thuật toán điều khiển luồng dữ liệu hệ thống . . . . .                          | 15 |
| 3.1  | Log mô phỏng FIFO - Phần 1 . . . . .   | 17 |
| 3.2  | Log mô phỏng FIFO - Phần 2 . . . . .   | 18 |
| 3.3  | Giản đồ sóng tín hiệu FIFO . . . . .   | 18 |
| 3.4  | Log mô phỏng PISO - Phần 1 . . . . .   | 19 |
| 3.5  | Log mô phỏng PISO - Phần 2 . . . . .   | 20 |
| 3.6  | Giản đồ sóng tín hiệu PISO . . . . .   | 20 |
| 3.7  | Log mô phỏng BMU - Phần 1 . . . . .  | 21 |
| 3.8  | Log mô phỏng BMU - Phần 2 . . . . .  | 22 |
| 3.9  | Giản đồ sóng tín hiệu BMU . . . . .  | 22 |
| 3.10 | Log mô phỏng ACSU - Phần 1 . . . . .   | 23 |
| 3.11 | Log mô phỏng ACSU - Phần 2 . . . . .   | 24 |
| 3.12 | Giản đồ sóng tín hiệu ACSU . . . . .   | 24 |
| 3.13 | Log mô phỏng PMU - Phần 1 . . . . .  | 25 |
| 3.14 | Log mô phỏng PMU - Phần 2 . . . . .  | 25 |
| 3.15 | Giản đồ sóng tín hiệu PMU . . . . .  | 26 |
| 3.16 | Log mô phỏng TBU - Phần 1 . . . . .  | 26 |
| 3.17 | Log mô phỏng TBU - Phần 2 . . . . .  | 27 |
| 3.18 | Giản đồ sóng tín hiệu TBU . . . . .  | 27 |
| 3.19 | Log mô phỏng SIPO - Phần 1 . . . . .   | 28 |
| 3.20 | Log mô phỏng SIPO - Phần 2 . . . . .   | 29 |
| 3.21 | Giản đồ sóng tín hiệu SIPO . . . . .   | 29 |
| 3.22 | Bảng tổng kết (Test Summary Report) từ Simulation Log . . . . .                        | 31 |
| 3.23 | [SYS_01] Sanity Check: Hệ thống giải mã chính xác dữ liệu chuẩn . . . . .              | 31 |
| 3.24 | [SYS_03] Single Bit Error: Hệ thống tự động sửa sai khi đầu vào bị lỗi 1 bit . . . . . | 32 |
| 3.25 | [SYS_07] Continuous Stress: Hoạt động ổn định dưới tải liên tục . . . . .              | 32 |

#### 4.1 Hình ảnh Layout GDSII cuối cùng của hệ thống . . . . . 34

## Danh sách bảng

|     |   |    |
|-----|---|----|
| 2.1 | Các thông số thiết kế chính . . . . .                     | 6  |
| 2.2 | Đặc tả giao diện System Top . . . . .                     | 7  |
| 2.3 | Giao diện khối Sync FIFO . . . . .                        | 8  |
| 2.4 | Giao diện khối PISO . . . . .                             | 8  |
| 2.5 | Giao diện khối BMU . . . . .                              | 9  |
| 2.6 | Giao diện khối ACSU . . . . .                             | 10 |
| 2.7 | Giao diện khối PMU . . . . .                              | 11 |
| 2.8 | Giao diện khối TBU . . . . .                              | 12 |
| 2.9 | Giao diện khối SIPO . . . . .                             | 13 |
| 3.1 | Tổng hợp kết quả các kịch bản kiểm thử hệ thống . . . . . | 30 |
| 4.1 | Tài nguyên phần cứng sau tổng hợp . . . . .               | 33 |

# Chương 1: TỔNG QUAN VỀ MÃ TÍCH CHẬP VÀ GIẢI MÃ VITERBI

## 1.1 Giới thiệu mã tích chập

Mã tích chập (Convolutional Codes) là một lớp mã sửa lỗi quan trọng, trong đó luồng bit đầu ra được tạo ra từ việc thực hiện phép tích chập giữa luồng bit đầu vào với đáp ứng xung của các thanh ghi dịch. Một bộ mã hóa tích chập được đặc trưng bởi bộ tham số  $(n, k, m)$ , trong đó:

- $k$ : Số bit đầu vào tại mỗi nhịp thời gian.
- $n$ : Số bit đầu ra tại mỗi nhịp thời gian.
- $m$ : Độ sâu của bộ nhớ (số lượng thanh ghi dịch).

Độ dài giới hạn quan trọng  $K = m + 1$  biểu thị số lượng bit đầu vào ảnh hưởng đến một bit đầu ra. Tỷ lệ mã  $R = k/n$  biểu thị hiệu suất băng thông.

## 1.2 Nguyên lý giải mã Viterbi

Giải mã Viterbi là một giải thuật quy hoạch động dùng để tìm đường dẫn có khả năng xảy ra cao nhất (Maximum Likelihood Path) trên biểu đồ lưới (Trellis diagram). Quá trình giải mã gồm ba bước cơ bản:

1. **Branch Metric Calculation:** Tính toán độ tương đồng giữa tín hiệu nhận được và tín hiệu lý thuyết trên mỗi nhánh.
2. **Path Metric Update (ACS):** Tại mỗi trạng thái, cộng Metric nhánh vào Metric đường dẫn tích lũy, so sánh và chọn đường dẫn tốt nhất (Survivor Path).
3. **Survivor Path Memory:** Lưu trữ lịch sử quyết định để khôi phục lại chuỗi dữ liệu gốc.

# Chương 2: ĐẶC TẢ KỸ THUẬT VÀ THIẾT KẾ RTL

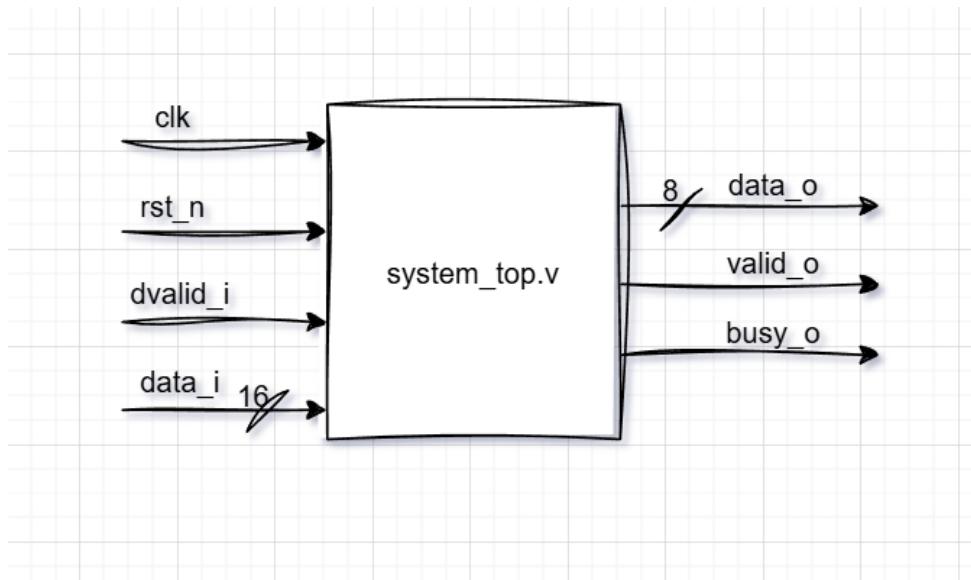
## 2.1 Đặc tả kỹ thuật (Specifications)

Dựa trên yêu cầu của đề tài, hệ thống Viterbi Decoder được thiết kế với các thông số kỹ thuật sau:

Bảng 2.1: Các thông số thiết kế chính

| Tham số                   | Giá trị                              |
|---------------------------|--------------------------------------|
| Constraint Length ( $K$ ) | 3                                    |
| Code Rate ( $R$ )         | 1/2                                  |
| Generators (Octal)        | $G_1 = 7_8(111_2), G_2 = 5_8(101_2)$ |
| Traceback Length ( $L$ )  | 15                                   |
| Architecture Type         | Register Exchange (RE)               |
| Soft/Hard Decision        | Hard Decision (1-bit quantization)   |

Bảng tín hiệu giao tiếp của khối Top-Level (`system_top`):



Hình 2.1: Sơ đồ khái niệm về khung Top-Level System

Bảng 2.2: Đặc tả giao diện System Top

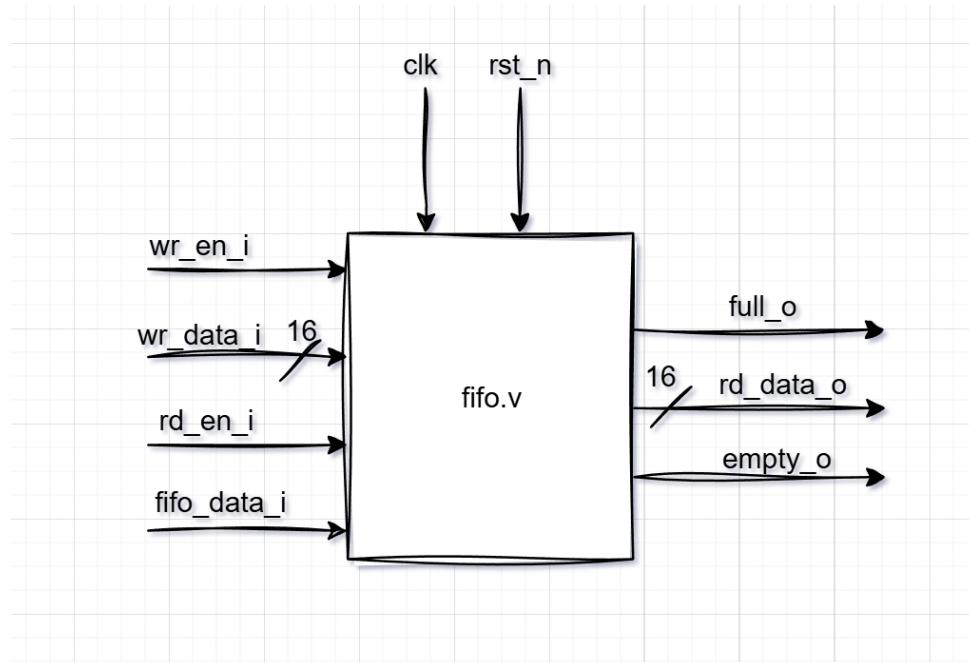
| Tên tín hiệu | Độ rộng | Hướng | Mô tả  |
|--------------|---------|-------|--|
| clk          | 1       | I     | Xung clock hệ thống (Cạnh dương).                |
| rst_n        | 1       | I     | Reset hệ thống (Tích cực thấp).                  |
| dvalid_i     | 1       | I     | Tín hiệu báo dữ liệu đầu vào hợp lệ (FIFO WrEn). |
| data_i       | 16      | I     | Gói dữ liệu đầu vào (16-bit).                    |
| data_o       | 8       | O     | Dữ liệu byte đã giải mã (8-bit).                 |
| valid_o      | 1       | O     | Tín hiệu báo dữ liệu đầu ra đã sẵn sàng.         |
| busy_o       | 1       | O     | Báo bận khi bộ đệm FIFO đầy.                     |

## 2.2 Kiến trúc hệ thống và Chi tiết I/O từng khối

Hệ thống được thiết kế theo luồng dữ liệu Pipeline để tối ưu hóa hiệu năng. Dưới đây là mô tả chi tiết các chân tín hiệu của từng thực thể chức năng.

### 2.2.1 Khối đệm dữ liệu (FIFO)

Khối này lưu trữ tạm thời dữ liệu 16-bit từ bên ngoài trước khi đưa vào bộ giải mã.



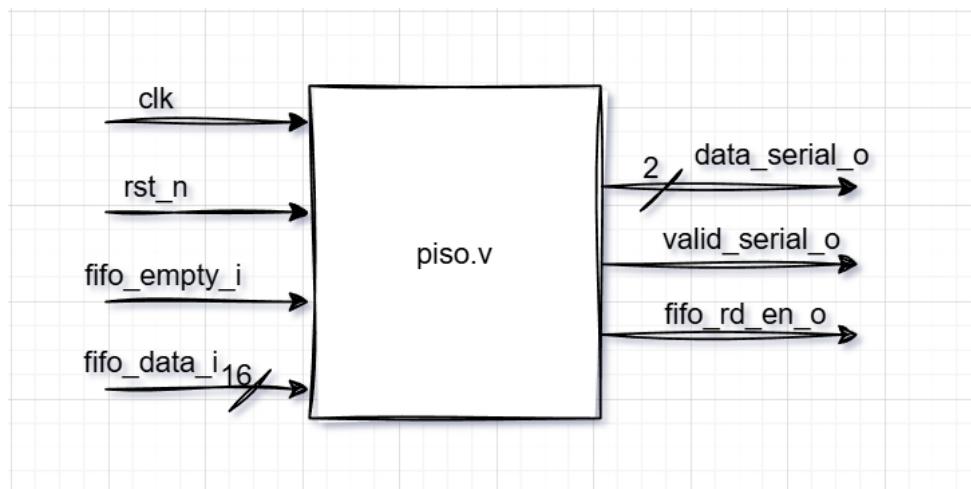
Hình 2.2: Sơ đồ khối FIFO

Bảng 2.3: Giao diện khói Sync FIFO

| Tên tín hiệu | Độ rộng | Hướng | Mô tả                           |
|--------------|---------|-------|---------------------------------|
| clk          | 1       | I     | Xung clock hệ thống.            |
| rst_n        | 1       | I     | Reset hệ thống (Tích cực thấp). |
| wr_en_i      | 1       | I     | Lệnh ghi dữ liệu vào FIFO.      |
| wr_data_i    | 16      | I     | Dữ liệu 16-bit ghi vào.         |
| full_o       | 1       | O     | Cờ báo bộ đệm đã đầy.           |
| rd_en_i      | 1       | I     | Lệnh đọc dữ liệu ra (từ PISO).  |
| rd_data_o    | 16      | O     | Dữ liệu 16-bit đọc ra.          |
| empty_o      | 1       | O     | Cờ báo bộ đệm rỗng.             |

## 2.2.2 Khối chuyển đổi song song sang nối tiếp (PISO)

Chuyển đổi gói 16-bit thành các cặp 2-bit (symbol) để nạp vào Viterbi Core.



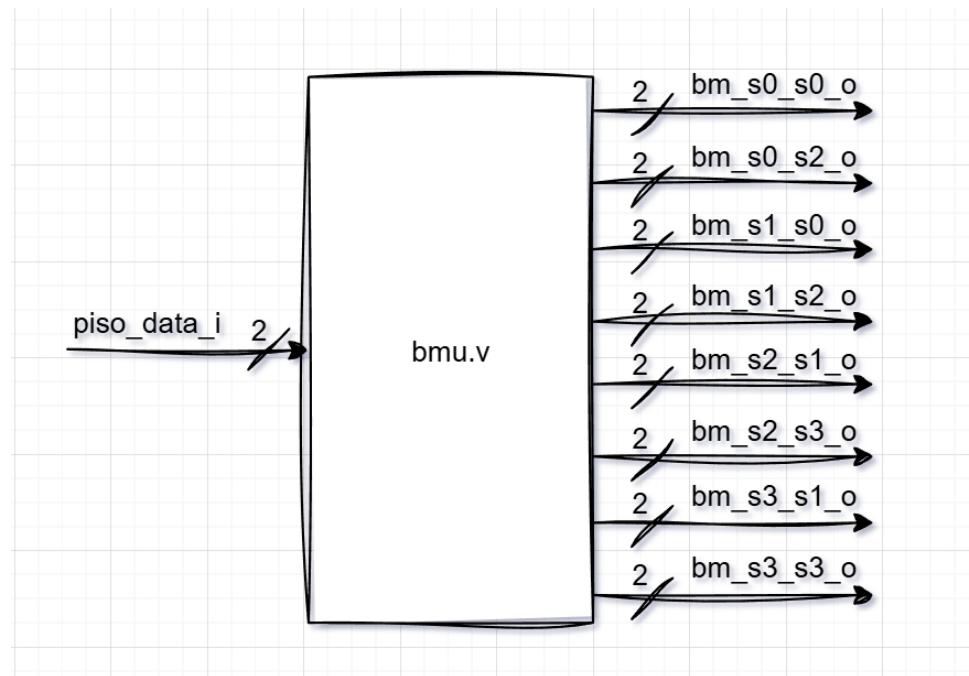
Hình 2.3: Sơ đồ khói PISO

Bảng 2.4: Giao diện khói PISO

| Tên tín hiệu   | Độ rộng | Hướng | Mô tả                           |
|----------------|---------|-------|---------------------------------|
| clk            | 1       | I     | Xung clock hệ thống.            |
| rst_n          | 1       | I     | Reset hệ thống (Tích cực thấp). |
| fifo_data_i    | 16      | I     | Dữ liệu nhận từ FIFO.           |
| fifo_empty_i   | 1       | I     | Trạng thái rỗng của FIFO.       |
| fifo_rd_en_o   | 1       | O     | Lệnh yêu cầu FIFO xuất dữ liệu. |
| data_serial_o  | 2       | O     | Cặp bit symbol gửi tới Core.    |
| valid_serial_o | 1       | O     | Báo symbol hiện tại hợp lệ.     |

## 2.2.3 Khối tính toán chi phí nhánh (BMU)

Tính khoảng cách Hamming giữa cặp bit nhận được và các nhánh kỳ vọng trong đồ thị lưới.



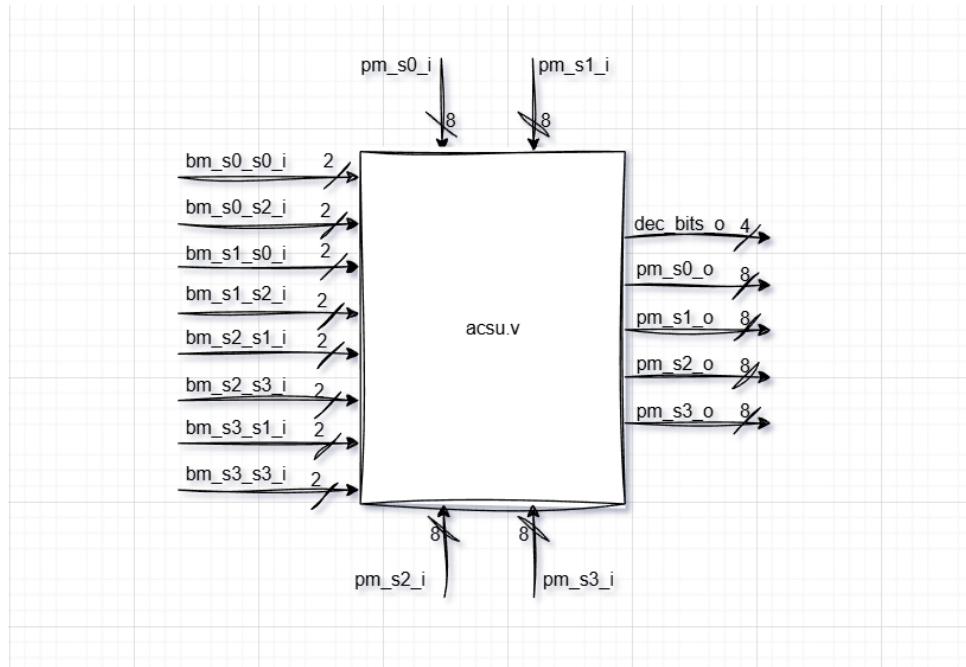
Hình 2.4: Sơ đồ khái niệm khối BMU

Bảng 2.5: Giao diện khái niệm khối BMU

| Tên tín hiệu | Độ rộng | Hướng | Mô tả                       |
|--------------|---------|-------|-----------------------------|
| piso_data_i  | 2       | I     | Dữ liệu 2-bit từ PISO.      |
| bm_s0_s0_o   | 2       | O     | Chi phí nhánh từ S0 đến S0. |
| bm_s0_s2_o   | 2       | O     | Chi phí nhánh từ S0 đến S2. |
| bm_s1_s0_o   | 2       | O     | Chi phí nhánh từ S1 đến S0. |
| bm_s1_s2_o   | 2       | O     | Chi phí nhánh từ S1 đến S2. |
| bm_s2_s1_o   | 2       | O     | Chi phí nhánh từ S2 đến S1. |
| bm_s2_s3_o   | 2       | O     | Chi phí nhánh từ S2 đến S3. |
| bm_s3_s1_o   | 2       | O     | Chi phí nhánh từ S3 đến S1. |
| bm_s3_s3_o   | 2       | O     | Chi phí nhánh từ S3 đến S3. |

#### 2.2.4 Khối Cộng - So sánh - Chọn (ACSU)

Cập nhật Path Metric cho 4 trạng thái và đưa ra quyết định chọn đường dẫn (Survivor Path).



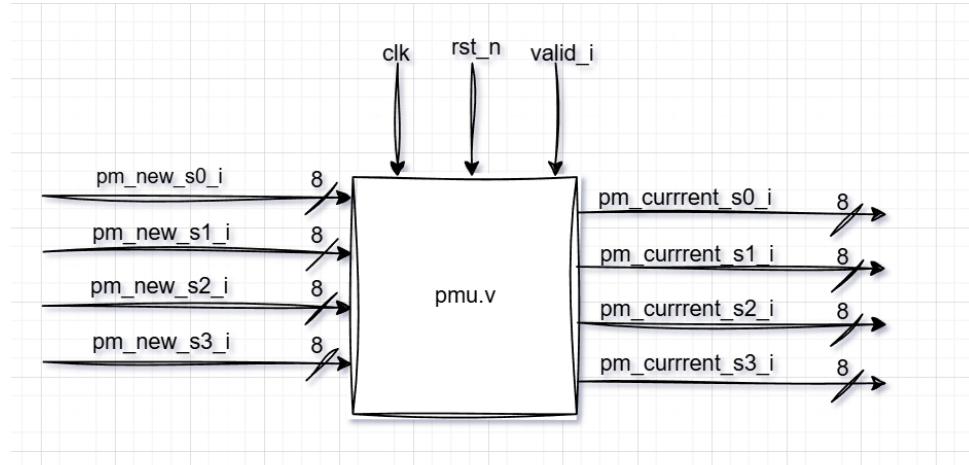
Hình 2.5: Sơ đồ khái niệmACSU

Bảng 2.6: Giao diện khái niệmACSU

| Tên tín hiệu | Độ rộng | Hướng | Mô tả   |
|--------------|---------|-------|---|
| bm_s0_s0_i   | 2       | I     | Chi phí nhánh từ S0 đến S0.                         |
| bm_s0_s2_i   | 2       | I     | Chi phí nhánh từ S0 đến S2.                         |
| bm_s1_s0_i   | 2       | I     | Chi phí nhánh từ S1 đến S0.                         |
| bm_s1_s2_i   | 2       | I     | Chi phí nhánh từ S1 đến S2.                         |
| bm_s2_s1_i   | 2       | I     | Chi phí nhánh từ S2 đến S1.                         |
| bm_s2_s3_i   | 2       | I     | Chi phí nhánh từ S2 đến S3.                         |
| bm_s3_s1_i   | 2       | I     | Chi phí nhánh từ S3 đến S1.                         |
| bm_s3_s3_i   | 2       | I     | Chi phí nhánh từ S3 đến S3.                         |
| pm_s0_i      | 8       | I     | Chi phí đường đi hiện tại của S0.                   |
| pm_s1_i      | 8       | I     | Chi phí đường đi hiện tại của S1.                   |
| pm_s2_i      | 8       | I     | Chi phí đường đi hiện tại của S2.                   |
| pm_s3_i      | 8       | I     | Chi phí đường đi hiện tại của S3.                   |
| pm_s0_o      | 8       | O     | Chi phí đường đi mới của S0.                        |
| pm_s1_o      | 8       | O     | Chi phí đường đi mới của S1.                        |
| pm_s2_o      | 8       | O     | Chi phí đường đi mới của S2.                        |
| pm_s3_o      | 8       | O     | Chi phí đường đi mới của S3.                        |
| dec_bits_o   | 4       | O     | 4 bit quyết định chọn nhánh (1 bit mỗi trạng thái). |

## 2.2.5 Khối quản lý Path Metric (PMU)

Lưu trữ và phản hồi chi phí tích lũy của các trạng thái.



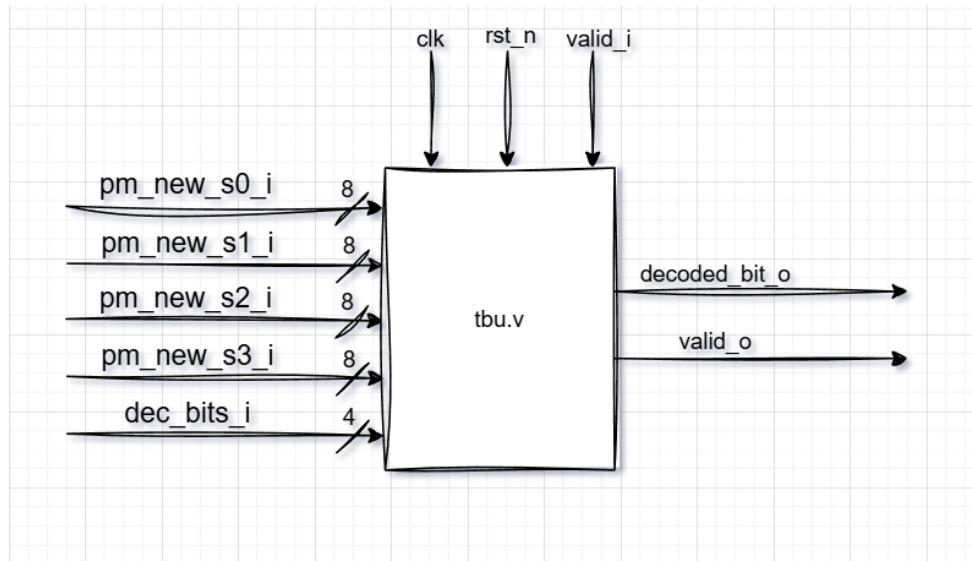
Hình 2.6: Sơ đồ khái niệm khối PMU

Bảng 2.7: Giao diện khái niệm khối PMU

| Tên tín hiệu    | Độ rộng | Hướng | Mô tả                             |
|-----------------|---------|-------|-----------------------------------|
| clk             | 1       | I     | Xung clock hệ thống.              |
| rst_n           | 1       | I     | Reset hệ thống (Tích cực thấp).   |
| valid_i         | 1       | I     | Cho phép cập nhật giá trị mới.    |
| pm_new_s0_i     | 8       | I     | Giá trị mới từ ACSU cho S0.       |
| pm_new_s1_i     | 8       | I     | Giá trị mới từ ACSU cho S1.       |
| pm_new_s2_i     | 8       | I     | Giá trị mới từ ACSU cho S2.       |
| pm_new_s3_i     | 8       | I     | Giá trị mới từ ACSU cho S3.       |
| pm_current_s0_o | 8       | O     | Giá trị hiện tại của S0 gửi ACSU. |
| pm_current_s1_o | 8       | O     | Giá trị hiện tại của S1 gửi ACSU. |
| pm_current_s2_o | 8       | O     | Giá trị hiện tại của S2 gửi ACSU. |
| pm_current_s3_o | 8       | O     | Giá trị hiện tại của S3 gửi ACSU. |

### 2.2.6 Khối truy vết Survivor Memory (TBU - Register Exchange)

Thực hiện kỹ thuật Register Exchange để khôi phục bit gốc từ các bit quyết định.



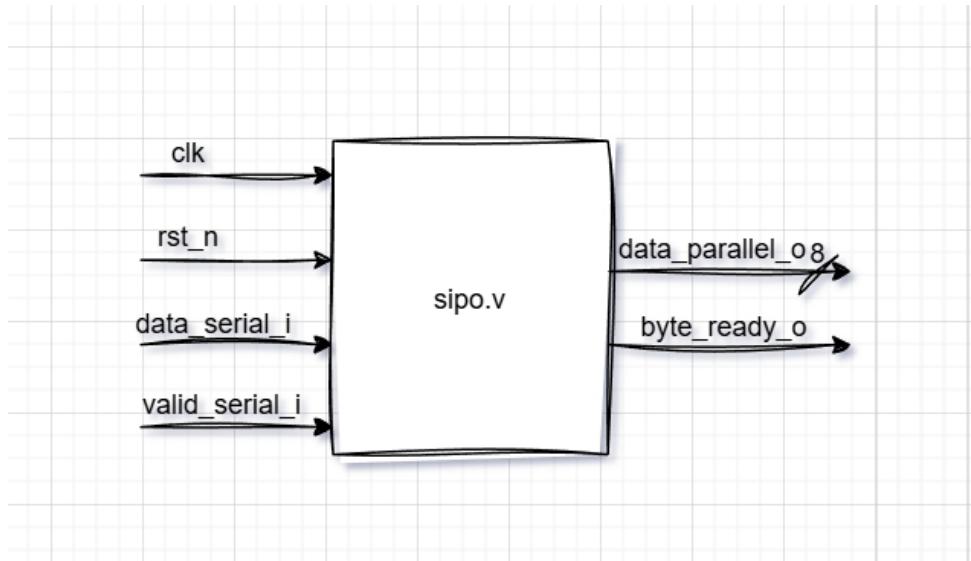
Hình 2.7: Sơ đồ khái niệm TBU

Bảng 2.8: Giao diện khái niệm TBU

| Tên tín hiệu  | Độ rộng | Hướng | Mô tả                                      |
|---------------|---------|-------|--|
| clk           | 1       | I     | Xung clock hệ thống.                       |
| rst_n         | 1       | I     | Reset hệ thống (Tích cực thấp).            |
| valid_i       | 1       | I     | Tín hiệu cho phép cập nhật.                |
| dec_bits_i    | 4       | I     | Các bit quyết định từ ACSU.                |
| pm_new_s0_i   | 8       | I     | Chi phí đường dẫn mới của S0 (để so sánh). |
| pm_new_s1_i   | 8       | I     | Chi phí đường dẫn mới của S1 (để so sánh). |
| pm_new_s2_i   | 8       | I     | Chi phí đường dẫn mới của S2 (để so sánh). |
| pm_new_s3_i   | 8       | I     | Chi phí đường dẫn mới của S3 (để so sánh). |
| decoded_bit_o | 1       | O     | Bit dữ liệu đã giải mã thành công.         |
| valid_o       | 1       | O     | Báo bit giải mã hợp lệ.                    |

### 2.2.7 Khối chuyển đổi nối tiếp sang song song (SIPO)

Gom các bit đơn lẻ từ TBU thành byte 8-bit hoàn chỉnh trước khi xuất ra ngoài hệ thống.



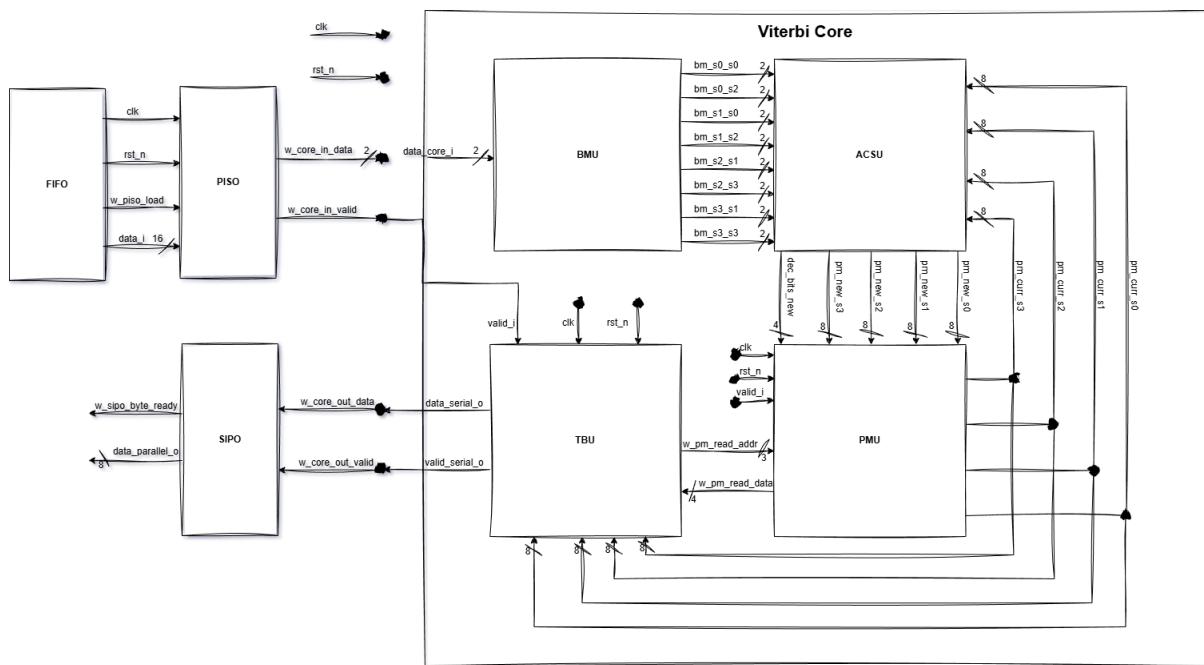
Hình 2.8: Sơ đồ khối SIPO

Bảng 2.9: Giao diện khói SIPO

| Tên tín hiệu    | Độ rộng | Hướng | Mô tả                              |
|-----------------|---------|-------|------------------------------------|
| clk             | 1       | I     | Xung clock hệ thống.               |
| rst_n           | 1       | I     | Reset hệ thống (Tích cực thấp).    |
| data_serial_i   | 1       | I     | Bit nhận từ TBU.                   |
| valid_serial_i  | 1       | I     | Cờ báo bit đầu vào hợp lệ.         |
| data_parallel_o | 8       | O     | Dữ liệu byte hoàn chỉnh.           |
| byte_ready_o    | 1       | O     | Báo hiệu đã đủ 8 bit (Byte Valid). |

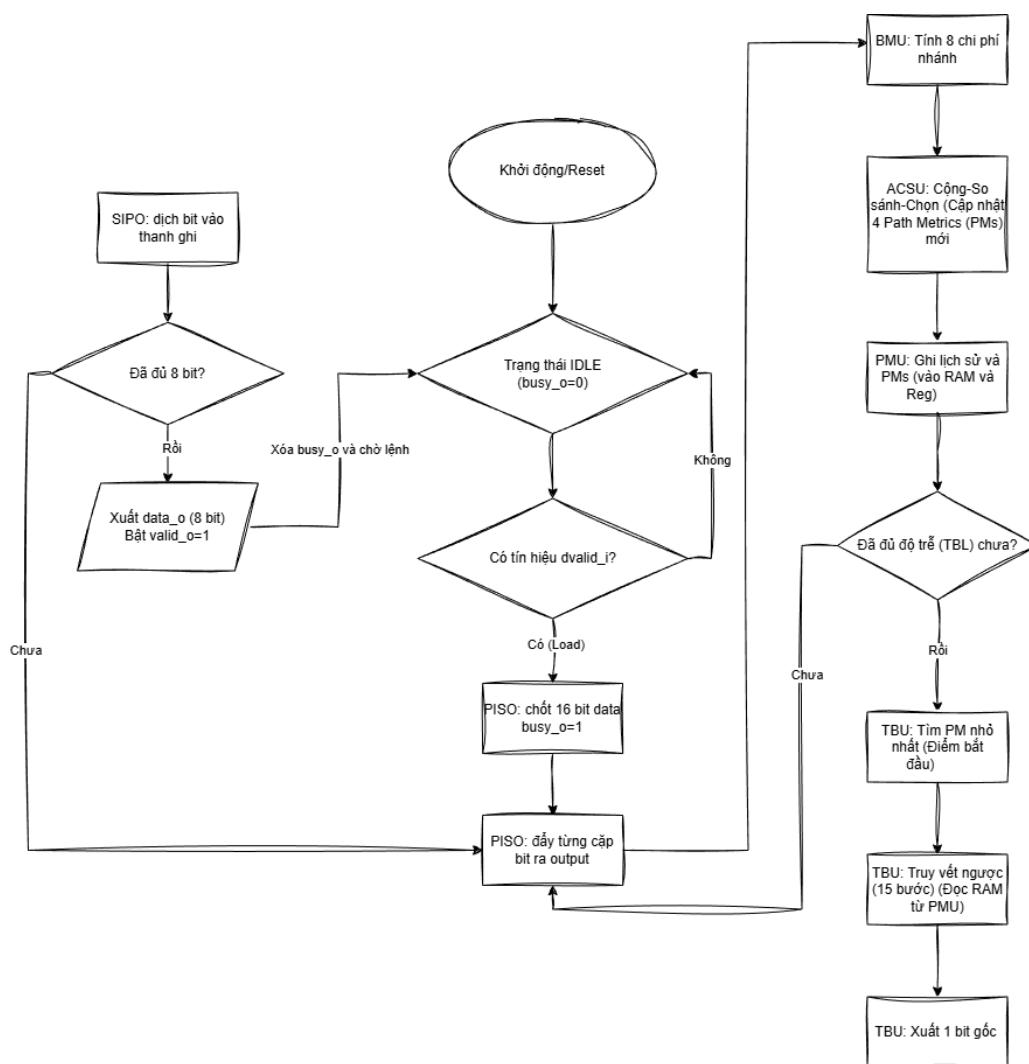
## 2.3 Triển khai RTL (RTL Implementation)

Mã nguồn RTL được viết bằng Verilog HDL. Trước khi đi vào chi tiết, sơ đồ khói hệ thống (Hình 2.9) và lưu đồ thuật toán (Hình 2.10) minh họa luồng xử lý tổng quan.



Hình 2.9: Sơ đồ khái chi tiết hệ thống Viterbi Decoder SoC

Lưu đồ thuật toán điều khiển luồng dữ liệu:



Hình 2.10: Lưu đồ thuật toán điều khiển luồng dữ liệu hệ thống

Thay vì sử dụng bộ nhớ RAM để lưu vết và quay lui (Traceback) gây độ trễ lớn, nhóm sử dụng kỹ thuật **Register Exchange (RE)** trong khối TBU.

Mỗi trạng thái ( $S_0, S_1, S_2, S_3$ ) quản lý một thanh ghi dịch lịch sử dài 15 bit.

```

1 // Update logic example for state S0
2 if (dec_bits_i[0] == 0)
3     history_s0 <= {history_s0[TBL-2:0], 1'b0}; // Select path from
4     previous S0
5 else
6     history_s0 <= {history_s1[TBL-2:0], 1'b0}; // Select path from
7     previous S1

```

Listing 2.1: Cơ chế Register Exchange trong TBU

Nhờ cơ chế này, sau khi pipeline được điền đầy (15 chu kỳ), đầu ra dữ liệu là liên tục (streaming) với năng suất (throughput) cao nhất (1 bit/cycle).

## Chương 3: KIỂM TRA CHỨC NĂNG MẠCH

### 3.1 Môi trường kiểm chứng (Testbench Environment)

Môi trường kiểm tra được xây dựng trong file `tb_system_top.sv` với các thành phần:

- **Generator:** Tạo ngẫu nhiên 1025 gói dữ liệu 16-bit.
- **Driver:** Đưa dữ liệu vào hệ thống qua giao thức bắt tay (valid/busy).
- **Monitor:** Thu thập dữ liệu đầu ra `data_o`.
- **Scoreboard:** So sánh kết quả mô phỏng với "Golden Model" (mô hình chuẩn).

### 3.2 Kết quả mô phỏng (Simulation Results)

Dưới đây là kết quả mô phỏng chi tiết cho từng khối chức năng (Unit Level) và toàn hệ thống (System Level). Các hình ảnh bao gồm log dữ liệu và giản đồ sóng xác nhận hoạt động chính xác của thiết kế.

#### 3.2.1 Khối đệm dữ liệu (FIFO)

```
lroot@LAPTOP-7Q2IL10W:/mnt/e/vlsi/vlsi_ver1$ iverilog -g2012 -o fifo.x tb_fifo.sv sync_fifo.v
lroot@LAPTOP-7Q2IL10W:/mnt/e/vlsi/vlsi_ver1$ ./fifo.x
VCD info: dumpfile fifo.vcd opened for output.
=====
START FIFO TEST: AUTO-RESET INDEX VERSION
=====

--- Round 1 ---
[210000 ns] FIFO FULL - Start Reading...
[237000 ns] [PASS] Got: 3524 | Exp: 3524
[247000 ns] [PASS] Got: 5e81 | Exp: 5e81
[257000 ns] [PASS] Got: d609 | Exp: d609
[267000 ns] [PASS] Got: 5663 | Exp: 5663
[277000 ns] [PASS] Got: 7b0d | Exp: 7b0d
[287000 ns] [PASS] Got: 998d | Exp: 998d
[297000 ns] [PASS] Got: 8465 | Exp: 8465
[307000 ns] [PASS] Got: 5212 | Exp: 5212
[317000 ns] [PASS] Got: e301 | Exp: e301
[327000 ns] [PASS] Got: cd0d | Exp: cd0d
[337000 ns] [PASS] Got: f176 | Exp: f176
[347000 ns] [PASS] Got: cd3d | Exp: cd3d
[357000 ns] [PASS] Got: 57ed | Exp: 57ed
[367000 ns] [PASS] Got: f78c | Exp: f78c
[377000 ns] [PASS] Got: e9f9 | Exp: e9f9
[387000 ns] [PASS] Got: 24c6 | Exp: 24c6

--- Round 2 ---
[610000 ns] FIFO FULL - Start Reading...
[637000 ns] [PASS] Got: d2aa | Exp: d2aa
[647000 ns] [PASS] Got: f7e5 | Exp: f7e5
[657000 ns] [PASS] Got: 7277 | Exp: 7277
[667000 ns] [PASS] Got: d612 | Exp: d612
[677000 ns] [PASS] Got: db8f | Exp: db8f
[687000 ns] [PASS] Got: 69f2 | Exp: 69f2
[697000 ns] [PASS] Got: 96ce | Exp: 96ce
[707000 ns] [PASS] Got: 7ae8 | Exp: 7ae8
[717000 ns] [PASS] Got: 4ec5 | Exp: 4ec5
[727000 ns] [PASS] Got: 495c | Exp: 495c
[737000 ns] [PASS] Got: 28bd | Exp: 28bd
[747000 ns] [PASS] Got: 582d | Exp: 582d
[757000 ns] [PASS] Got: 2665 | Exp: 2665
[767000 ns] [PASS] Got: 6263 | Exp: 6263
[777000 ns] [PASS] Got: 870a | Exp: 870a
```

Hình 3.1: Log mô phỏng FIFO - Phần 1

```

[1137000 ns] [PASS] Got: e91d | Exp: e91d
[1147000 ns] [PASS] Got: 72cf | Exp: 72cf
[1157000 ns] [PASS] Got: 4923 | Exp: 4923
[1167000 ns] [PASS] Got: 650a | Exp: 650a
[1177000 ns] [PASS] Got: 0aca | Exp: 0aca
[1187000 ns] [PASS] Got: 4c3c | Exp: 4c3c

--- Round 4 ---
[1410000 ns] FIFO FULL - Start Reading...
[1437000 ns] [PASS] Got: 618a | Exp: 618a
[1447000 ns] [PASS] Got: b341 | Exp: b341
[1457000 ns] [PASS] Got: 34d8 | Exp: 34d8
[1467000 ns] [PASS] Got: f378 | Exp: f378
[1477000 ns] [PASS] Got: 1289 | Exp: 1289
[1487000 ns] [PASS] Got: 0deb | Exp: 0deb
[1497000 ns] [PASS] Got: 65b6 | Exp: 65b6
[1507000 ns] [PASS] Got: f9c6 | Exp: f9c6
[1517000 ns] [PASS] Got: 13ae | Exp: 13ae
[1527000 ns] [PASS] Got: 02bc | Exp: 02bc
[1537000 ns] [PASS] Got: dd2a | Exp: dd2a
[1547000 ns] [PASS] Got: 9a0b | Exp: 9a0b
[1557000 ns] [PASS] Got: be71 | Exp: be71
[1567000 ns] [PASS] Got: 4185 | Exp: 4185
[1577000 ns] [PASS] Got: 554f | Exp: 554f
[1587000 ns] [PASS] Got: 603b | Exp: 603b

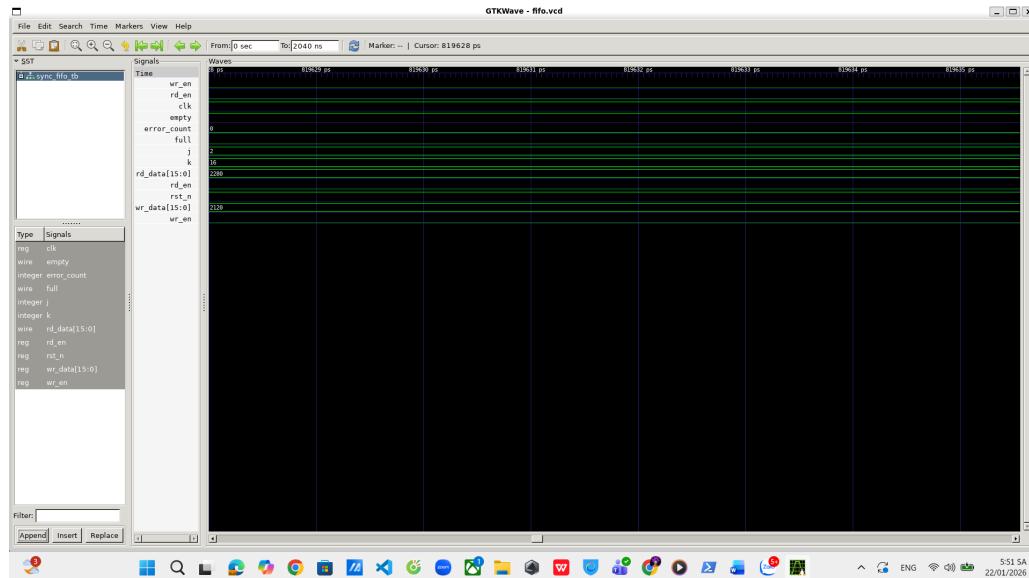
--- Round 5 ---
[1810000 ns] FIFO FULL - Start Reading...
[1837000 ns] [PASS] Got: 327e | Exp: 327e
[1847000 ns] [PASS] Got: 4b15 | Exp: 4b15
[1857000 ns] [PASS] Got: 9bf1 | Exp: 9bf1
[1867000 ns] [PASS] Got: 4bd9 | Exp: 4bd9
[1877000 ns] [PASS] Got: 0762 | Exp: 0762
[1887000 ns] [PASS] Got: fb4c | Exp: fb4c
[1897000 ns] [PASS] Got: 559f | Exp: 559f
[1907000 ns] [PASS] Got: a18f | Exp: a18f
[1917000 ns] [PASS] Got: a9f8 | Exp: a9f8
[1927000 ns] [PASS] Got: 60b7 | Exp: 60b7
[1937000 ns] [PASS] Got: 569f | Exp: 569f
[1947000 ns] [PASS] Got: 945c | Exp: 945c
[1957000 ns] [PASS] Got: c05b | Exp: c05b
[1967000 ns] [PASS] Got: 3789 | Exp: 3789
[1977000 ns] [PASS] Got: 3249 | Exp: 3249
[1987000 ns] [PASS] Got: 3ed0 | Exp: 3ed0

=====
>>> TỔNG KẾT: ALL PASS (0 ERRORS) <<<
=====

lroot@LAPTOP-7Q2ILT09:/mnt/e/vlsi/vlsi_ver1$ -

```

Hình 3.2: Log mô phỏng FIFO - Phần 2



Hình 3.3: Giản đồ sóng tín hiệu FIFO

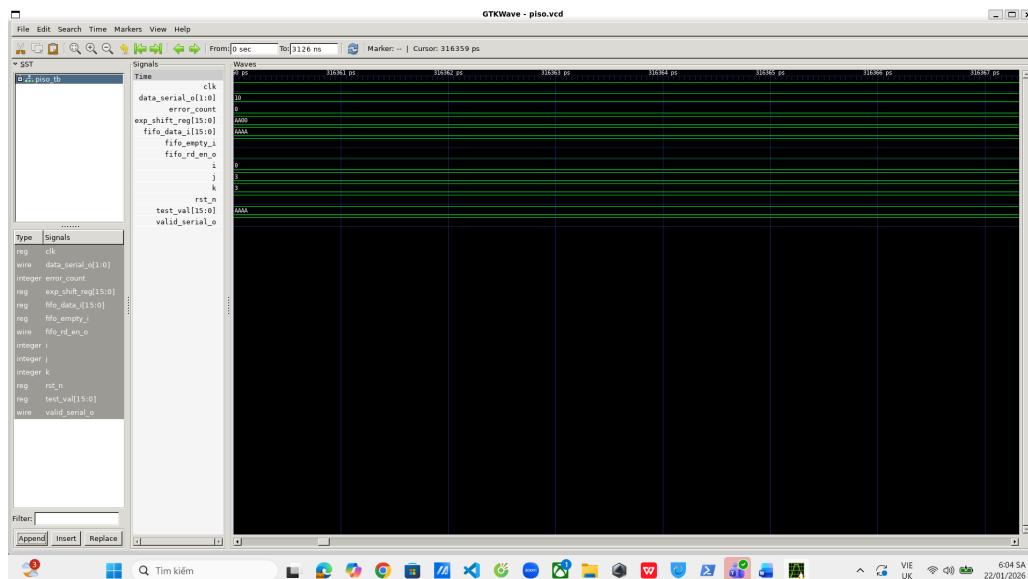
### 3.2.2 Khối chuyển đổi song song sang nối tiếp (PISO)

```
[886000 ns] [PASS] Step 0 | Got: 01 | Exp: 01
[896000 ns] [PASS] Step 1 | Got: 01 | Exp: 01
[906000 ns] [PASS] Step 2 | Got: 01 | Exp: 01
[916000 ns] [PASS] Step 3 | Got: 01 | Exp: 01
[926000 ns] [PASS] Step 4 | Got: 01 | Exp: 01
[936000 ns] [PASS] Step 5 | Got: 01 | Exp: 01
[946000 ns] [PASS] Step 6 | Got: 01 | Exp: 01
[956000 ns] [PASS] Step 7 | Got: 01 | Exp: 01
    Iteration 4:
[1006000 ns] [PASS] Step 0 | Got: 01 | Exp: 01
[1016000 ns] [PASS] Step 1 | Got: 01 | Exp: 01
[1026000 ns] [PASS] Step 2 | Got: 01 | Exp: 01
[1036000 ns] [PASS] Step 3 | Got: 01 | Exp: 01
[1046000 ns] [PASS] Step 4 | Got: 01 | Exp: 01
[1056000 ns] [PASS] Step 5 | Got: 01 | Exp: 01
[1066000 ns] [PASS] Step 6 | Got: 01 | Exp: 01
[1076000 ns] [PASS] Step 7 | Got: 01 | Exp: 01
    Iteration 5:
[1126000 ns] [PASS] Step 0 | Got: 01 | Exp: 01
[1136000 ns] [PASS] Step 1 | Got: 01 | Exp: 01
[1146000 ns] [PASS] Step 2 | Got: 01 | Exp: 01
[1156000 ns] [PASS] Step 3 | Got: 01 | Exp: 01
[1166000 ns] [PASS] Step 4 | Got: 01 | Exp: 01
[1176000 ns] [PASS] Step 5 | Got: 01 | Exp: 01
[1186000 ns] [PASS] Step 6 | Got: 01 | Exp: 01
[1196000 ns] [PASS] Step 7 | Got: 01 | Exp: 01
---> SCENARIO 3: Data 16'hffff
    Iteration 1:
[1246000 ns] [PASS] Step 0 | Got: 11 | Exp: 11
[1256000 ns] [PASS] Step 1 | Got: 11 | Exp: 11
[1266000 ns] [PASS] Step 2 | Got: 11 | Exp: 11
[1276000 ns] [PASS] Step 3 | Got: 11 | Exp: 11
[1286000 ns] [PASS] Step 4 | Got: 11 | Exp: 11
[1296000 ns] [PASS] Step 5 | Got: 11 | Exp: 11
[1306000 ns] [PASS] Step 6 | Got: 11 | Exp: 11
[1316000 ns] [PASS] Step 7 | Got: 11 | Exp: 11
    Iteration 2:
[1366000 ns] [PASS] Step 0 | Got: 11 | Exp: 11
[1376000 ns] [PASS] Step 1 | Got: 11 | Exp: 11
[1386000 ns] [PASS] Step 2 | Got: 11 | Exp: 11
[1396000 ns] [PASS] Step 3 | Got: 11 | Exp: 11
[1406000 ns] [PASS] Step 4 | Got: 11 | Exp: 11
[1416000 ns] [PASS] Step 5 | Got: 11 | Exp: 11
[1426000 ns] [PASS] Step 6 | Got: 11 | Exp: 11
[1436000 ns] [PASS] Step 7 | Got: 11 | Exp: 11
    Iteration 3:
[1486000 ns] [PASS] Step 0 | Got: 11 | Exp: 11
[1496000 ns] [PASS] Step 1 | Got: 11 | Exp: 11
```

Hình 3.4: Log mô phỏng PISO - Phần 1

```
[2446000 ns] [PASS] Step 0 | Got: 00 | Exp: 00
[2456000 ns] [PASS] Step 1 | Got: 11 | Exp: 11
[2466000 ns] [PASS] Step 2 | Got: 01 | Exp: 01
[2476000 ns] [PASS] Step 3 | Got: 01 | Exp: 01
[2486000 ns] [PASS] Step 4 | Got: 00 | Exp: 00
[2496000 ns] [PASS] Step 5 | Got: 10 | Exp: 10
[2506000 ns] [PASS] Step 6 | Got: 01 | Exp: 01
[2516000 ns] [PASS] Step 7 | Got: 00 | Exp: 00
    Iteration 2:
[2566000 ns] [PASS] Step 0 | Got: 00 | Exp: 00
[2576000 ns] [PASS] Step 1 | Got: 11 | Exp: 11
[2586000 ns] [PASS] Step 2 | Got: 01 | Exp: 01
[2596000 ns] [PASS] Step 3 | Got: 01 | Exp: 01
[2606000 ns] [PASS] Step 4 | Got: 00 | Exp: 00
[2616000 ns] [PASS] Step 5 | Got: 10 | Exp: 10
[2626000 ns] [PASS] Step 6 | Got: 01 | Exp: 01
[2636000 ns] [PASS] Step 7 | Got: 00 | Exp: 00
    Iteration 3:
[2686000 ns] [PASS] Step 0 | Got: 00 | Exp: 00
[2696000 ns] [PASS] Step 1 | Got: 11 | Exp: 11
[2706000 ns] [PASS] Step 2 | Got: 01 | Exp: 01
[2716000 ns] [PASS] Step 3 | Got: 01 | Exp: 01
[2726000 ns] [PASS] Step 4 | Got: 00 | Exp: 00
[2736000 ns] [PASS] Step 5 | Got: 10 | Exp: 10
[2746000 ns] [PASS] Step 6 | Got: 01 | Exp: 01
[2756000 ns] [PASS] Step 7 | Got: 00 | Exp: 00
    Iteration 4:
[2806000 ns] [PASS] Step 0 | Got: 00 | Exp: 00
[2816000 ns] [PASS] Step 1 | Got: 11 | Exp: 11
[2826000 ns] [PASS] Step 2 | Got: 01 | Exp: 01
[2836000 ns] [PASS] Step 3 | Got: 01 | Exp: 01
[2846000 ns] [PASS] Step 4 | Got: 00 | Exp: 00
[2856000 ns] [PASS] Step 5 | Got: 10 | Exp: 10
[2866000 ns] [PASS] Step 6 | Got: 01 | Exp: 01
[2876000 ns] [PASS] Step 7 | Got: 00 | Exp: 00
    Iteration 5:
[2926000 ns] [PASS] Step 0 | Got: 00 | Exp: 00
[2936000 ns] [PASS] Step 1 | Got: 11 | Exp: 11
[2946000 ns] [PASS] Step 2 | Got: 01 | Exp: 01
[2956000 ns] [PASS] Step 3 | Got: 01 | Exp: 01
[2966000 ns] [PASS] Step 4 | Got: 00 | Exp: 00
[2976000 ns] [PASS] Step 5 | Got: 10 | Exp: 10
[2986000 ns] [PASS] Step 6 | Got: 01 | Exp: 01
[2996000 ns] [PASS] Step 7 | Got: 00 | Exp: 00
=====
FINAL TOTAL ERRORS: 0 -> STATUS: ALL PASS
=====
[root@LAPTOP-7Q2ILTE9:/mnt/e/vlsi/vlsi_ver1$
```

Hình 3.5: Log mô phỏng PISO - Phần 2



Hình 3.6: Giản đồ sóng tín hiệu PISO

### 3.2.3 Khối tính toán nhánh (BMU)

| S0-S0   | 2   | 2        | OK     |
|---|-----|----------|--------|
| S0-S2   | 0   | 0        | OK     |
| S2-S3   | 1   | 1        | OK     |
| S3-S3   | 1   | 1        | OK     |
| <hr/>   |     |          |        |
| [99000 ns] SCENARIO: Inverse Case (11) (Input: 11)  |     |          |        |
| <hr/>   |     |          |        |
| Branch  | Got | Expected | Status |
| S0-S0   | 2   | 2        | OK     |
| S0-S2   | 0   | 0        | OK     |
| S2-S3   | 1   | 1        | OK     |
| S3-S3   | 1   | 1        | OK     |
| <hr/>   |     |          |        |
| [110000 ns] SCENARIO: Inverse Case (11) (Input: 11) |     |          |        |
| <hr/>   |     |          |        |
| Branch  | Got | Expected | Status |
| S0-S0   | 2   | 2        | OK     |
| S0-S2   | 0   | 0        | OK     |
| S2-S3   | 1   | 1        | OK     |
| S3-S3   | 1   | 1        | OK     |
| <hr/>   |     |          |        |
| [121000 ns] SCENARIO: Error 1-bit (01) (Input: 01)  |     |          |        |
| <hr/>   |     |          |        |
| Branch  | Got | Expected | Status |
| S0-S0   | 1   | 1        | OK     |
| S0-S2   | 1   | 1        | OK     |
| S2-S3   | 0   | 0        | OK     |
| S3-S3   | 2   | 2        | OK     |
| <hr/>   |     |          |        |
| [132000 ns] SCENARIO: Error 1-bit (01) (Input: 01)  |     |          |        |
| <hr/>   |     |          |        |
| Branch  | Got | Expected | Status |
| S0-S0   | 1   | 1        | OK     |
| S0-S2   | 1   | 1        | OK     |
| S2-S3   | 0   | 0        | OK     |
| S3-S3   | 2   | 2        | OK     |
| <hr/>   |     |          |        |
| [143000 ns] SCENARIO: Error 1-bit (01) (Input: 01)  |     |          |        |
| <hr/>   |     |          |        |
| Branch  | Got | Expected | Status |
| S0-S0   | 1   | 1        | OK     |
| S0-S2   | 1   | 1        | OK     |
| S2-S3   | 0   | 0        | OK     |
| S3-S3   | 2   | 2        | OK     |

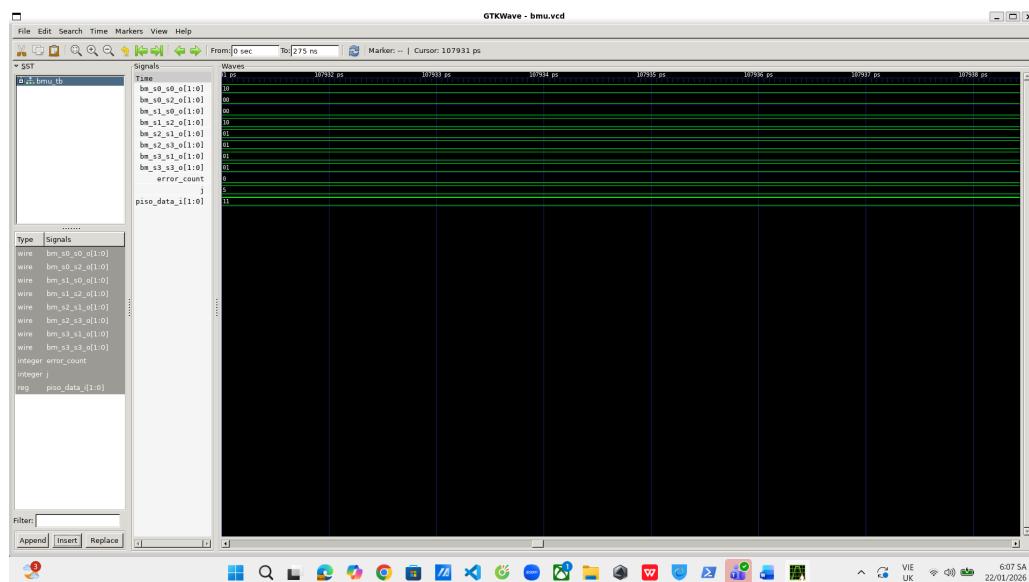
Hình 3.7: Log mô phỏng BMU - Phần 1

```

Branch | Got | Expected | Status
-----|-----|-----|-----
S0-S0 | 0 | 0 | OK
S0-S2 | 2 | 2 | OK
S2-S3 | 1 | 1 | OK
S3-S3 | 1 | 1 | OK
-----
[242000 ns] SCENARIO: Random Stress (Input: 01)
Branch | Got | Expected | Status
-----|-----|-----|-----
S0-S0 | 1 | 1 | OK
S0-S2 | 1 | 1 | OK
S2-S3 | 0 | 0 | OK
S3-S3 | 2 | 2 | OK
-----
[253000 ns] SCENARIO: Random Stress (Input: 01)
Branch | Got | Expected | Status
-----|-----|-----|-----
S0-S0 | 1 | 1 | OK
S0-S2 | 1 | 1 | OK
S2-S3 | 0 | 0 | OK
S3-S3 | 2 | 2 | OK
-----
[264000 ns] SCENARIO: Random Stress (Input: 11)
Branch | Got | Expected | Status
-----|-----|-----|-----
S0-S0 | 2 | 2 | OK
S0-S2 | 0 | 0 | OK
S2-S3 | 1 | 1 | OK
S3-S3 | 1 | 1 | OK
-----
[275000 ns] SCENARIO: Random Stress (Input: 01)
Branch | Got | Expected | Status
-----|-----|-----|-----
S0-S0 | 1 | 1 | OK
S0-S2 | 1 | 1 | OK
S2-S3 | 0 | 0 | OK
S3-S3 | 2 | 2 | OK
-----
=====
FINAL TOTAL ERRORS: 0 -> STATUS: ALL PASS
=====
1root@APTOP-7Q21LT09:/mnt/e/vlsi/vlsi_ver1$ -

```

Hình 3.8: Log mô phỏng BMU - Phần 2



Hình 3.9: Giản đồ sóng tín hiệu BMU

### 3.2.4 Khối Cộng - So sánh - Chọn (ACSU)

|     |      |      |    |
|-----|------|------|----|
| DEC | 1010 | 1010 | OK |
| PM0 | 250  | 250  | OK |
| PM1 | 76   | 76   | OK |
| PM2 | 250  | 250  | OK |
| PM3 | 75   | 75   | OK |

---

| [165000 ns] [PASS] Boundary Case |      |          |        |
|----------------------------------|------|----------|--------|
| Output                           | Got  | Expected | Status |
| DEC                              | 1011 | 1011     | OK     |
| PM0                              | 250  | 250      | OK     |
| PM1                              | 76   | 76       | OK     |
| PM2                              | 250  | 250      | OK     |
| PM3                              | 75   | 75       | OK     |

---

| [176000 ns] [PASS] Zero Case |      |          |        |
|------------------------------|------|----------|--------|
| Output                       | Got  | Expected | Status |
| DEC                          | 0000 | 0000     | OK     |
| PM0                          | 0    | 0        | OK     |
| PM1                          | 0    | 0        | OK     |
| PM2                          | 0    | 0        | OK     |
| PM3                          | 0    | 0        | OK     |

---

| [187000 ns] [PASS] Zero Case |      |          |        |
|------------------------------|------|----------|--------|
| Output                       | Got  | Expected | Status |
| DEC                          | 0000 | 0000     | OK     |
| PM0                          | 0    | 0        | OK     |
| PM1                          | 0    | 0        | OK     |
| PM2                          | 0    | 0        | OK     |
| PM3                          | 0    | 0        | OK     |

---

| [198000 ns] [PASS] Zero Case |      |          |        |
|------------------------------|------|----------|--------|
| Output                       | Got  | Expected | Status |
| DEC                          | 0000 | 0000     | OK     |
| PM0                          | 0    | 0        | OK     |
| PM1                          | 0    | 0        | OK     |
| PM2                          | 0    | 0        | OK     |
| PM3                          | 0    | 0        | OK     |

---

| [209000 ns] [PASS] Zero Case |      |          |        |
|------------------------------|------|----------|--------|
| Output                       | Got  | Expected | Status |
| DEC                          | 0000 | 0000     | OK     |
| PM0                          | 0    | 0        | OK     |
| PM1                          | 0    | 0        | OK     |
| PM2                          | 0    | 0        | OK     |
| PM3                          | 0    | 0        | OK     |

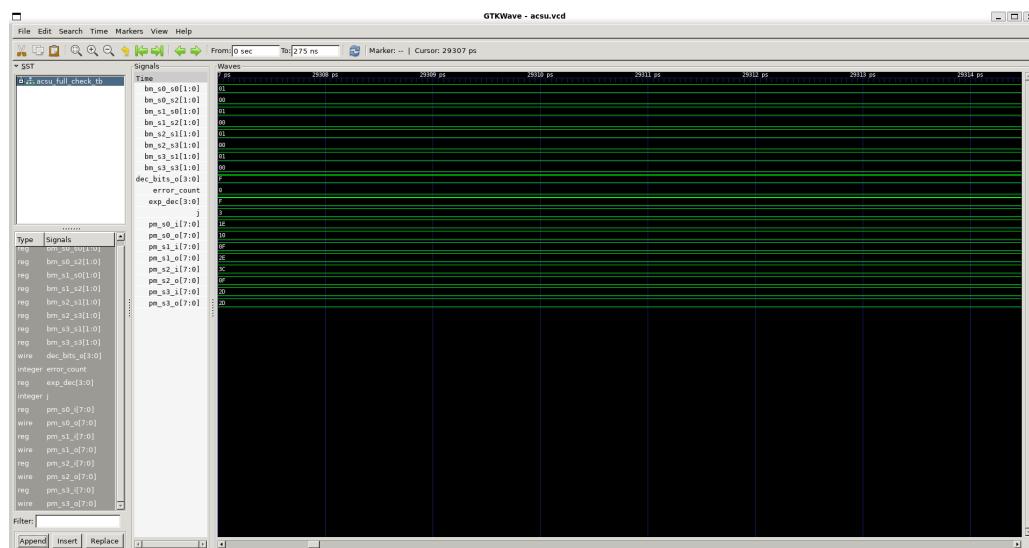
Hình 3.10: Log mô phỏng ACSU - Phần 1

```

PM1 | 10 | 10 | OK
PM2 | 37 | 37 | OK
PM3 | 11 | 11 | OK
-----
[242000 ns] [PASS] Random Stress
-----|-----|-----|-----|-----|
Output | Got | Expected | Status
-----|-----|-----|-----|
DEC | 1111 | 1111 | OK
PM0 | 142 | 142 | OK
PM1 | 201 | 201 | OK
PM2 | 143 | 143 | OK
PM3 | 200 | 200 | OK
-----
[253000 ns] [PASS] Random Stress
-----|-----|-----|-----|-----|
Output | Got | Expected | Status
-----|-----|-----|-----|
DEC | 0101 | 0101 | OK
PM0 | 198 | 198 | OK
PM1 | 95 | 95 | OK
PM2 | 197 | 197 | OK
PM3 | 94 | 94 | OK
-----
[264000 ns] [PASS] Random Stress
-----|-----|-----|-----|-----|
Output | Got | Expected | Status
-----|-----|-----|-----|
DEC | 0101 | 0101 | OK
PM0 | 20 | 20 | OK
PM1 | 15 | 15 | OK
PM2 | 22 | 22 | OK
PM3 | 16 | 16 | OK
-----
[275000 ns] [PASS] Random Stress
-----|-----|-----|-----|-----|
Output | Got | Expected | Status
-----|-----|-----|-----|
DEC | 1111 | 1111 | OK
PM0 | 60 | 60 | OK
PM1 | 139 | 139 | OK
PM2 | 62 | 62 | OK
PM3 | 140 | 140 | OK
-----
===== FINAL TOTAL ERRORS: 0 =====
[root@APTOP-7Q2TI~: /mnt/e/vlsi/vlsi_ver1$]

```

Hình 3.11: Log mô phỏng ACSU - Phần 2



Hình 3.12: Giản đồ sóng tín hiệu ACSU

### 3.2.5 Khối quản lý Metric (PMU)

```

NN Destroy
lroot@LAPTOP-7Q2ILT09:/mnt/e/vlsi/vlsi_ver1$ iverilog -g2012 -o pmu.x tb_pmu.sv pmu.v
lroot@LAPTOP-7Q2ILT09:/mnt/e/vlsi/vlsi_ver1$ ./pmu.x
=====
START PMU TEST: TIMING - GOT - EXPECTED REPORT
=====
VCD info: dumpfile pmu.vcd opened for output.
[10000 ns]      Reset Check
-----
| State | Got | Expected | Status |
-----|-----|-----|-----|
| S0   |    0 |      0 | OK   |
| S1   |  255 |    255 | OK   |
| S2   |  255 |    255 | OK   |
| S3   |  255 |    255 | OK   |
-----
[16000 ns]      Update Enable
-----
| State | Got | Expected | Status |
-----|-----|-----|-----|
| S0   |   10 |      10 | OK   |
| S1   |   20 |      20 | OK   |
| S2   |   30 |      30 | OK   |
| S3   |   40 |      40 | OK   |
-----
[26000 ns]      Update Enable
-----
| State | Got | Expected | Status |
-----|-----|-----|-----|
| S0   |   20 |      20 | OK   |
| S1   |   40 |      40 | OK   |
| S2   |   60 |      60 | OK   |
| S3   |   80 |      80 | OK   |
-----
[36000 ns]      Update Enable
-----
| State | Got | Expected | Status |
-----|-----|-----|-----|
| S0   |   30 |      30 | OK   |
| S1   |   60 |      60 | OK   |
| S2   |   90 |      90 | OK   |
| S3   |  120 |     120 | OK   |
-----
[46000 ns]      Update Enable
-----
| State | Got | Expected | Status |
-----|-----|-----|-----|
| S0   |   40 |      40 | OK   |
| S1   |   80 |      80 | OK   |

```

Hình 3.13: Log mô phỏng PMU - Phần 1

```

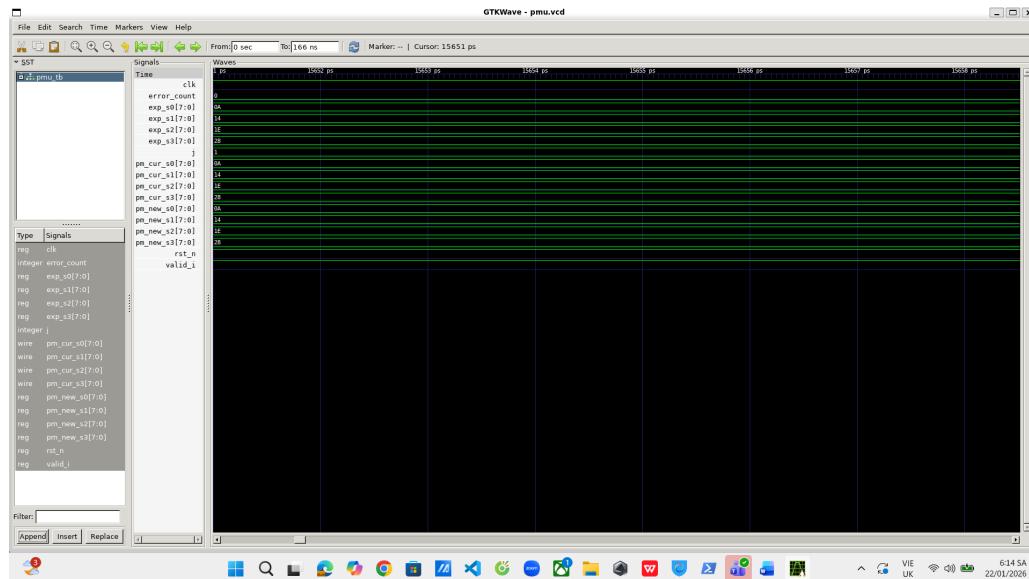
[136000 ns]      Random Stress
-----
| State | Got | Expected | Status |
-----|-----|-----|-----|
| S0   |   36 |      36 | OK   |
| S1   |  129 |     129 | OK   |
| S2   |    9 |      9 | OK   |
| S3   |   99 |      99 | OK   |
-----
[146000 ns]      Random Stress
-----
| State | Got | Expected | Status |
-----|-----|-----|-----|
| S0   |   13 |      13 | OK   |
| S1   |  141 |     141 | OK   |
| S2   |  101 |     101 | OK   |
| S3   |   18 |      18 | OK   |
-----
[156000 ns]      Random Stress
-----
| State | Got | Expected | Status |
-----|-----|-----|-----|
| S0   |   237 |     237 | OK   |
| S1   |  140 |     140 | OK   |
| S2   |  249 |     249 | OK   |
| S3   |  198 |     198 | OK   |
-----
[166000 ns]      Random Stress
-----
| State | Got | Expected | Status |
-----|-----|-----|-----|
| S0   |  197 |     197 | OK   |
| S1   |  170 |     170 | OK   |
| S2   |  229 |     229 | OK   |
| S3   |  119 |     119 | OK   |
-----
=====

FINAL TOTAL ERRORS: 0
>>> STATUS: ALL PASS <<
=====

lroot@LAPTOP-7Q2ILT09:/mnt/e/vlsi/vlsi_ver1$ 

```

Hình 3.14: Log mô phỏng PMU - Phần 2



Hình 3.15: Giản đồ sóng tín hiệu PMU

### 3.2.6 Khôi truy vết (TBU)

```
(gtkwave:1617): dconf-WARNING **: 06:14:03.023: failed to commit changes to dconf: Could not connect: No such file or directory
(gtkwave:1617): dconf-WARNING **: 06:14:03.024: failed to commit changes to dconf: Could not connect: No such file or directory
W: Destroy
1:root@LAPTOP-Q2QILTB9:~/mnt/e/vlsi/vlsi_ver1$ iverilog -g2012 -o tbu.x tb_tbu.sv tbu.v
1:root@LAPTOP-Q2QILTB9:~/mnt/e/vlsi/vlsi_ver1$ ./tbu.x
-----  
BAT DAU TBU TEST: THOI GIAN - NHAN - KY VONG  
=====  
VCD info: dumpfile tbu.vcd opened for output.  
[20000 ns] Reset Check
-----  
Signal | Got | Expected | Status  
----|---|---|---  
Valid_Out | 0 | 0 | OK  
Decoded_Bit | 0 | 0 | OK  
  
--> Scenario 1: Filling Pipeline (Latency Test)  
[166000 ns] line Processing Last Fill
-----  
Signal | Got | Expected | Status  
----|---|---|---  
Valid_Out | 0 | 0 | OK  
Decoded_Bit | 0 | 0 | OK  
  
[176000 ns] pipeline Full - Start Valid
-----  
Signal | Got | Expected | Status  
----|---|---|---  
Valid_Out | 1 | 1 | OK  
Decoded_Bit | 0 | 0 | OK  
  
--> Scenario 2: Data Traceback (S2 Input)  
[336000 ns] Traceback Bit 1 check
-----  
Signal | Got | Expected | Status  
----|---|---|---  
Valid_Out | 1 | 1 | OK  
Decoded_Bit | 1 | 1 | OK  
  
--> Scenario 3: Valid_i Gating  
[346000 ns] Valid_i Off
-----  
Signal | Got | Expected | Status  
----|---|---|---
```

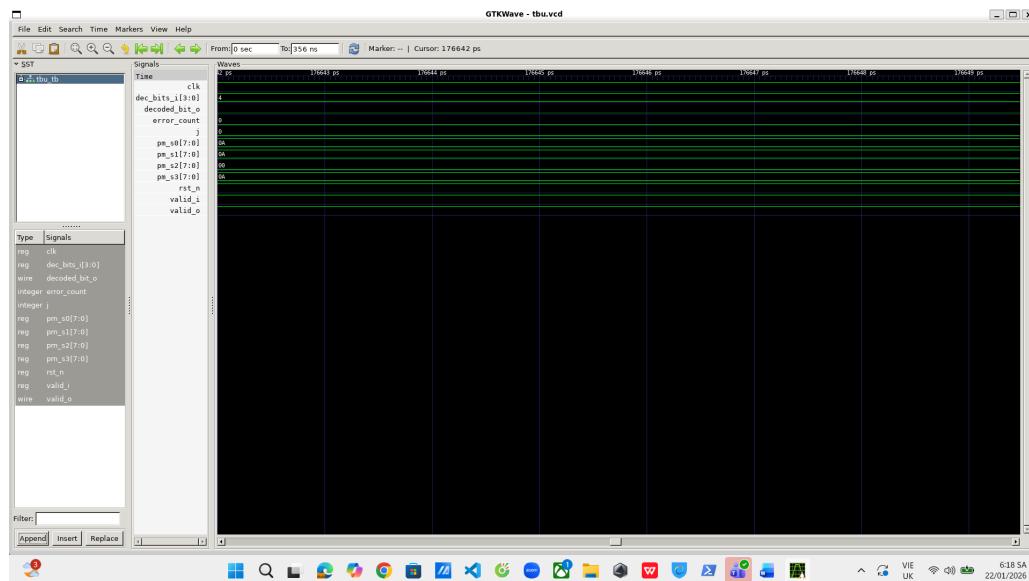
Hình 3.16: Log mô phỏng TBU - Phần 1

```

-----
| Signal | Got | Expected | Status |
-----|-----|-----|-----|
| Valid_Out | 0 | 0 | OK |
| Decoded_Bit | 0 | 0 | OK |
-----
[1760000 ns] pipeline Full - Start Valid
-----
| Signal | Got | Expected | Status |
-----|-----|-----|-----|
| Valid_Out | 1 | 1 | OK |
| Decoded_Bit | 0 | 0 | OK |
-----
--> Scenario 2: Data Traceback (S2 Input)
[3360000 ns] Traceback Bit 1 Check
-----
| Signal | Got | Expected | Status |
-----|-----|-----|-----|
| Valid_Out | 1 | 1 | OK |
| Decoded_Bit | 1 | 1 | OK |
-----
--> Scenario 3: Valid_i Gating
[3460000 ns] Valid_i Off
-----
| Signal | Got | Expected | Status |
-----|-----|-----|-----|
| Valid_Out | 0 | 0 | OK |
| Decoded_Bit | 1 | 0 | OK |
-----
--> Scenario 4: Winner Switching
[3560000 ns] Winner switched to S3
-----
| Signal | Got | Expected | Status |
-----|-----|-----|-----|
| Valid_Out | 1 | 1 | OK |
| Decoded_Bit | 0 | 0 | OK |
-----
=====
FINAL TOTAL ERRORS: 0
>>> STATUS: ALL PASS <<<
=====
root@LAPTOP-7Q2ILT09:/mnt/e/vlsi/vlsi_ver1$ 

```

Hình 3.17: Log mô phỏng TBU - Phần 2



Hình 3.18: Giản đồ sóng tín hiệu TBU

### 3.2.7 Khối chuyển đổi nối tiếp sang song song (SIPO)

```

root@LAPTOP-7Q2LT09:/mnt/e/vlsi/vlsi_ver1$ ./sipo.x
=====
START SIPO TEST: EVENT-DRIVEN VERSION (NO TIMING ERRORS)
=====
VCD info: dumpfile sipo.vcd opened for output.
[40000 ns] Reset Check: OK
[121000 ns] Scenario 1: 0xA5 Received
-----
Signal | Got | Expected | Status
-----|---|---|---|
Byte Ready | 1 | 1 | OK
Parallel Out | 10100101 | 10100101 | OK
-----
[211000 ns] Scenario 2: 0xFF Received
-----
Signal | Got | Expected | Status
-----|---|---|---|
Byte Ready | 1 | 1 | OK
Parallel Out | 11111111 | 11111111 | OK
-----
[301000 ns] Scenario 3: 0x00 Received
-----
Signal | Got | Expected | Status
-----|---|---|---|
Byte Ready | 1 | 1 | OK
Parallel Out | 00000000 | 00000000 | OK
-----
[391000 ns] Scenario 4: Random Byte
-----
Signal | Got | Expected | Status
-----|---|---|---|
Byte Ready | 1 | 1 | OK
Parallel Out | 00100100 | 00100100 | OK
-----
[501000 ns] Scenario 4: Random Byte
-----
Signal | Got | Expected | Status
-----|---|---|---|
Byte Ready | 1 | 1 | OK
Parallel Out | 10000001 | 10000001 | OK
-----
[611000 ns] Scenario 4: Random Byte
-----
Signal | Got | Expected | Status
-----|---|---|---|
Byte Ready | 1 | 1 | OK
Parallel Out | 00001001 | 00001001 | OK
-----
[721000 ns] Scenario 4: Random Byte

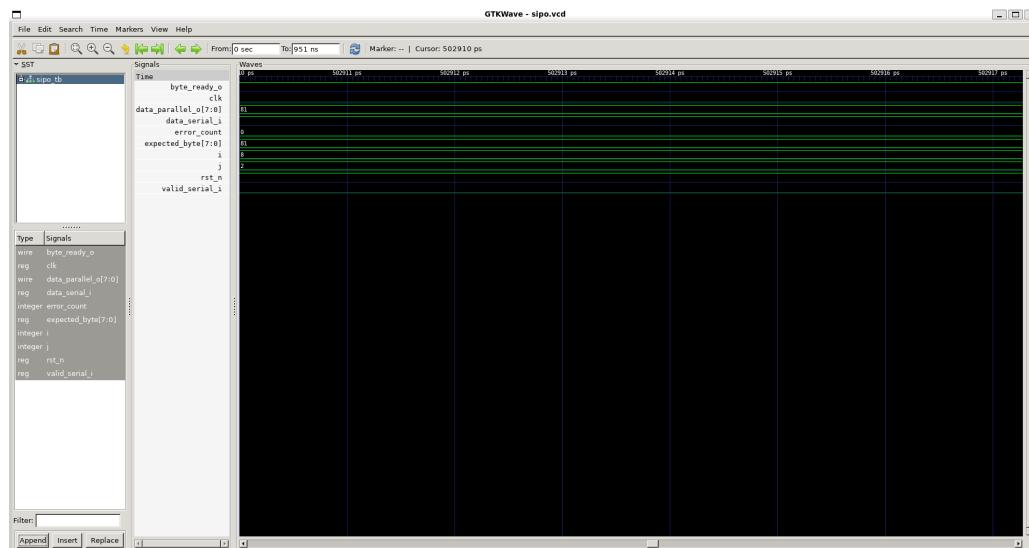
```

Hình 3.19: Log mô phỏng SIPO - Phần 1

```

Parallel Out| 11111111 | 11111111 | OK
-----
[301000 ns] Scenario 3: 0x00 Received
Signal | Got | Expected | Status
-----
Byte Ready | 1 | 1 | OK
Parallel Out| 00000000 | 00000000 | OK
-----
[391000 ns] Scenario 4: Random Byte
Signal | Got | Expected | Status
-----
Byte Ready | 1 | 1 | OK
Parallel Out| 00100100 | 00100100 | OK
-----
[501000 ns] Scenario 4: Random Byte
Signal | Got | Expected | Status
-----
Byte Ready | 1 | 1 | OK
Parallel Out| 10000001 | 10000001 | OK
-----
[611000 ns] Scenario 4: Random Byte
Signal | Got | Expected | Status
-----
Byte Ready | 1 | 1 | OK
Parallel Out| 00001001 | 00001001 | OK
-----
[721000 ns] Scenario 4: Random Byte
Signal | Got | Expected | Status
-----
Byte Ready | 1 | 1 | OK
Parallel Out| 01100011 | 01100011 | OK
-----
[831000 ns] Scenario 4: Random Byte
Signal | Got | Expected | Status
-----
Byte Ready | 1 | 1 | OK
Parallel Out| 00001101 | 00001101 | OK
-----
=====
TỔNG KẾT: TẤT CA PASS (0 LỖI)
=====
lroot@LAPTOP-7Q2ILT09:/mnt/e/vlsi/vlsi_ver1$
```

Hình 3.20: Log mô phỏng SIPO - Phần 2



Hình 3.21: Giản đồ sóng tín hiệu SIPO

### 3.2.8 Kiểm chứng toàn hệ thống (System Integration)

Sau khi các khối đơn lẻ (Unit Level) đã hoạt động đúng, nhóm thực hiện tích hợp toàn bộ hệ thống và tiến hành kiểm thử mức hệ thống (System Level Verification). Mục tiêu là đảm bảo luồng dữ liệu từ FIFO → PISO → Viterbi Core → SIPO hoạt động đồng bộ, đúng timing và

quan trọng nhất là khả năng sửa lỗi của thuật toán.

### Kịch bản kiểm thử (Testcases)

Hệ thống được kiểm tra với bộ Test Suite gồm 8 kịch bản chính, bao phủ từ hoạt động bình thường, khả năng sửa lỗi bit đến các trường hợp chịu tải cao (Stress test).

Bảng 3.1: Tổng hợp kết quả các kịch bản kiểm thử hệ thống

| ID     | Tên Testcase      | Mô tả kịch bản   | Kết quả         |
|--------|-------------------|--|-----------------|
| SYS_01 | Sanity Check      | Kiểm tra cơ bản: Gửi gói tin chuẩn, không lỗi.                 | Pass            |
| SYS_02 | Full Range Sweep  | Quét toàn bộ dải giá trị đầu vào (0x0000 - 0xFFFF).            | Pass            |
| SYS_03 | Single Bit Error  | Gây nhiễu sai 1 bit ngẫu nhiên. Hệ thống phải tự sửa lại đúng. | Pass            |
| SYS_04 | Multi-bit Error   | Gây nhiễu sai 2-3 bit (Vượt quá khả năng sửa lỗi lý thuyết).   | Mixed           |
| SYS_05 | Burst Error       | Kiểm tra khả năng chịu lỗi chùm (sai 4 bit liên tiếp).         | Fail (Expected) |
| SYS_06 | Busy Ignore       | Gửi dữ liệu khi cờ Busy=1 (FIFO đầy).                          | Pass            |
| SYS_07 | Continuous Stress | Gửi dữ liệu liên tục tốc độ cao (Back-to-back transaction).    | Pass            |
| SYS_08 | Reset Recovery    | Reset hệ thống giữa chừng khi đang xử lý.                      | Pass            |

### Mã nguồn Testbench (Trích đoạn)

Module kiểm thử tự động (`tb_system_top.sv`) được xây dựng với cơ chế "Self-checking". Dưới đây là trích đoạn Task gửi dữ liệu có hỗ trợ tiêm lỗi (Error Injection):

```

1 // Task: Send 16-bit data packet
2 task send_packet(input [15:0] data, input [2:0] error_count);
3     logic [15:0] corrupted_data;
4 begin
5     // Inject artificial error to test error correction
6     // capability
7     corrupted_data = inject_error(data, error_count);
8
9     // Handshake protocol with the system
10    @(posedge clk);           // Wait for system ready
11    dvalid_i = 1'b1;          // Assert Valid
12    data_i   = corrupted_data;
13    @(posedge clk);
14    dvalid_i = 1'b0;

```

```

15      end
16  endtask

```

Listing 3.1: Task gửi dữ liệu và tiêm lỗi trong Testbench

### Kết quả mô phỏng

Hình 3.22 dưới đây là báo cáo tổng hợp tự động từ Testbench. Hệ thống đạt tỷ lệ thành công tuyệt đối (100%) với các trường hợp dữ liệu sạch (Clean) và lỗi 1 bit (Error 1b).

| TEST SUMMARY REPORT |       |      |      |              |  |
|---------------------|-------|------|------|--------------|--|
| Error Type          | Total | PASS | FAIL | Success Rate |  |
| Clean (0b)          | 392   | 392  | 0    | 100.0%       |  |
| Error 1b            | 64    | 64   | 0    | 100.0%       |  |
| Error 2b            | 50    | 40   | 10   | 80.0%        |  |
| Error 3b            | 20    | 6    | 14   | 30.0%        |  |
| Burst (4+)          | 20    | 5    | 15   | 25.0%        |  |

TOTAL RUNS : 546

Hình 3.22: Bảng tổng kết (Test Summary Report) từ Simulation Log

Chi tiết log mô phỏng của một số trường hợp tiêu biểu:

| [SYS_01] SANITY CHECK |         |      |        |        |        |
|-----------------------|---------|------|--------|--------|--------|
| Time (ns)             | In(16b) | Recv | Expect | Errors | Status |
| 525000                | ----    | 00   | --     | --     | WARMUP |
| 625000                | 0000    | 00   | 00     | 0      | PASS   |
| 725000                | daaa    | ff   | ff     | 0      | PASS   |
| 825000                | 4888    | aa   | aa     | 0      | PASS   |
| 925000                | 5222    | 55   | 55     | 0      | PASS   |
| 1025000               | ----    | 00   | --     | --     | FLUSH  |
| 1125000               | ----    | 00   | --     | --     | FLUSH  |
| 1225000               | ----    | 00   | --     | --     | FLUSH  |
| 1325000               | ----    | 00   | --     | --     | FLUSH  |
| 1425000               | ----    | 00   | --     | --     | FLUSH  |

Hình 3.23: [SYS\_01] Sanity Check: Hệ thống giải mã chính xác dữ liệu chuẩn

## [SYS\_03] SINGLE BIT ERROR CORRECTION (Deterministic)

| Time (ns)                                | In(16b) | Recv | Expect | Errors | Status |
|--|---------|------|--------|--------|--------|
| [INFO] Bit Sweep: 44 (With Zero Tailing) |         |      |        |        |        |
| 28835000                                 | ---     | 00   | --     | --     | WARMUP |
| 28935000                                 | 0ecf    | 44   | 44     | 1      | PASS   |
| 29035000                                 | c000    | 00   | 00     | 0      | PASS   |
| 29135000                                 | ---     | 00   | --     | --     | FLUSH  |
| 29235000                                 | ---     | 00   | --     | --     | FLUSH  |
| 29335000                                 | ---     | 00   | --     | --     | FLUSH  |
| 29435000                                 | ---     | 00   | --     | --     | FLUSH  |
| 29535000                                 | ---     | 00   | --     | --     | FLUSH  |
| 29635000                                 | ---     | 00   | --     | --     | FLUSH  |
| 29735000                                 | ---     | 00   | --     | --     | FLUSH  |
| 29835000                                 | ---     | 00   | --     | --     | FLUSH  |
| 29935000                                 | ---     | 00   | --     | --     | FLUSH  |
| 30035000                                 | ---     | 00   | --     | --     | FLUSH  |
| 30135000                                 | ---     | 00   | --     | --     | FLUSH  |
| 30235000                                 | ---     | 00   | --     | --     | FLUSH  |
| 5029205000                               | ---     | 00   | --     | --     | WARMUP |
| 5029305000                               | 0ecc    | 44   | 44     | 1      | PASS   |
| 5029405000                               | c000    | 00   | 00     | 0      | PASS   |
| 5029505000                               | ---     | 00   | --     | --     | FLUSH  |
| 5029605000                               | ---     | 00   | --     | --     | FLUSH  |
| 5029705000                               | ---     | 00   | --     | --     | FLUSH  |
| 5029805000                               | ---     | 00   | --     | --     | FLUSH  |
| 5029905000                               | ---     | 00   | --     | --     | FLUSH  |
| 5030005000                               | ---     | 00   | --     | --     | FLUSH  |
| 5030105000                               | ---     | 00   | --     | --     | FLUSH  |
| 5030205000                               | ---     | 00   | --     | --     | FLUSH  |

Hình 3.24: [SYS\_03] Single Bit Error: Hệ thống tự động sửa sai khi đầu vào bị lỗi 1 bit

## [SYS\_07] CONTINUOUS PROCESSING (Stress)

|              |      |    |    |    |        |
|--------------|------|----|----|----|--------|
| 320164705000 | ---  | 00 | -- | -- | WARMUP |
| 320164805000 | 0ef6 | e4 | e4 | 0  | PASS   |
| 320164905000 | 7388 | a8 | a8 | 0  | PASS   |
| 320165005000 | b0e2 | 50 | 50 | 0  | PASS   |
| 320165105000 | 222f | 95 | 95 | 0  | PASS   |
| 320165205000 | 51aa | fd | fd | 0  | PASS   |
| 320165305000 | 4b3b | 22 | 22 | 0  | PASS   |
| 320165405000 | ec0d | c1 | c1 | 0  | PASS   |
| 320165505000 | 46a7 | 3e | 3e | 0  | PASS   |
| 320165605000 | ef67 | 39 | 39 | 0  | PASS   |
| 320165705000 | daa9 | 7f | 7f | 0  | PASS   |
| 320165805000 | ce17 | 34 | 34 | 0  | PASS   |
| 320165905000 | d917 | 37 | 37 | 0  | PASS   |
| 320166005000 | d70d | c3 | c3 | 0  | PASS   |
| 320166105000 | 736a | f8 | f8 | 0  | PASS   |
| 320166205000 | 7d9f | 9c | 9c | 0  | PASS   |
| 320166305000 | b038 | a0 | a0 | 0  | PASS   |
| 320166405000 | be2c | 14 | 14 | 0  | PASS   |
| 320166505000 | d46a | fb | fb | 0  | PASS   |
| 320166605000 | 4b35 | 62 | 62 | 0  | PASS   |
| 320166705000 | c036 | e0 | e0 | 0  | PASS   |
| 320166805000 | 4864 | ba | ba | 0  | PASS   |
| 320166905000 | bd9f | 9c | 9c | 0  | PASS   |

Hình 3.25: [SYS\_07] Continuous Stress: Hoạt động ổn định dưới tải liên tục

## Chương 4: TỔNG HỢP VÀ THIẾT KẾ VẬT LÝ

### 4.1 Quy trình OpenLane Flow

Thiết kế được đưa vào quy trình OpenLane để chuyển đổi từ RTL sang GDSII. Các bước thực hiện bao gồm:

1. **Synthesis (Yosys)**: Tổng hợp logic và map vào thư viện cells Sky130.
2. **Floorplan**: Định nghĩa kích thước die, tạo lưới nguồn (PDN).
3. **Placement (OpenROAD)**: Đặt các cell vào khung layout, tối ưu hóa độ dài dây dẫn.
4. **CTS**: Cân bằng cây xung clock để giảm Clock Skew.
5. **Routing**: Đi dây chi tiết và kiểm tra luật thiết kế (DRC).

### 4.2 Kết quả tổng hợp (Synthesis Results)

Báo cáo tổng hợp cho thấy tài nguyên phần cứng sử dụng rất hiệu quả:

Bảng 4.1: Tài nguyên phần cứng sau tổng hợp

| Tài nguyên                          | Số lượng       |
|-------------------------------------|----------------|
| Số lượng cổng logic (Combinational) | $\approx 1400$ |
| Số lượng Flip-Flops (Sequential)    | $\approx 600$  |
| Tổng số Cells                       | 2032           |
| Fanout tối đa                       | 10             |

### 4.3 Kết quả thiết kế vật lý (Physical Layout)

Sau khi hoàn tất Routing và Sign-off, các thông số vật lý cuối cùng ghi nhận được từ file `metrics.csv`:

- **Diện tích (Die Area)**:  $0.16mm^2$ .
- **Mật độ (Core Utilization)**: 40% (Đảm bảo không gian cho routing và tránh tắc nghẽn).
- **Công suất tiêu thụ ước tính**:  $0.007mW$  (Tại điều kiện hoạt động điển hình).
- **Thời gian (Timing)**:
  - Worst Negative Slack (WNS): 4.15 ns.

- Tần số hoạt động tối đa: 50 MHz.
- **Kiểm tra vật lý:** Không có vi phạm LVS (Layout Vs Schematic) và DRC (Design Rule Check).

Hình 4.1: Hình ảnh Layout GDSII cuối cùng của hệ thống

## KẾT LUẬN

Đồ án đã thiết kế và hiện thực hóa thành công một hệ thống Viterbi Decoder SoC từ mức ý tưởng đến Layout vật lý. Việc áp dụng thành công kỹ thuật **Register Exchange** đã chứng minh hiệu quả vượt trội trong việc xử lý dòng bit liên tục. Quy trình thiết kế sử dụng hoàn toàn các công cụ nguồn mở (OpenLane, Sky130, iVerilog) mở ra hướng đi khả thi và tiết kiệm chi phí cho việc thiết kế và sản xuất chip VLSI tại Việt Nam.

## Tài liệu tham khảo

- [1] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," IEEE Transactions on Information Theory.
- [2] OpenLane Documentation, "The Open-Source Digital ASIC Implementation Flow".
- [3] SkyWater SKY130 PDK, Google Open Source Silicon.