

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH  
KHOA ĐIỆN ĐIỆN TỬ  
BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

ĐỒ ÁN MÔN HỌC 1

**XÂY DỰNG MÔ HÌNH ĐIỂM – ĐIỂM  
SỬ DỤNG NGÔN NGỮ VERILOG**

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ - TRUYỀN THÔNG

Sinh viên: **NGUYỄN ĐÌNH KHÁNH VY**  
MSSV: 22161043

TP. HỒ CHÍ MINH – 05/2025

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH  
KHOA ĐIỆN ĐIỆN TỬ  
BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

ĐỒ ÁN MÔN HỌC 1

**XÂY DỰNG MÔ HÌNH ĐIỂM – ĐIỂM  
SỬ DỤNG NGÔN NGỮ VERILOG**

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ TRUYỀN THÔNG

Sinh viên: **NGUYỄN ĐÌNH KHÁNH VY**  
MSSV: 22161043

Hướng dẫn: **ThS. TRƯƠNG QUANG PHÚC**

TP. HỒ CHÍ MINH – 05/2025

## LỜI CẢM ƠN

Em xin bày tỏ lòng biết ơn sâu sắc đến Thầy Trương Quang Phúc – người đã trực tiếp hướng dẫn đồ án, tận tâm hỗ trợ, luôn theo sát quá trình thực hiện, đưa ra những góp ý quý báu và tạo điều kiện tốt nhất để em có thể hoàn thành đồ án môn học một cách hiệu quả. Sự đồng hành và định hướng từ Thầy là nguồn động lực to lớn giúp em vượt qua những khó khăn trong suốt quá trình thực hiện đồ án.

Em cũng xin chân thành cảm ơn Ban Giám hiệu nhà trường và Bộ môn Kỹ thuật Máy tính – Viễn thông đã tạo điều kiện thuận lợi về cơ sở vật chất, môi trường học tập chuyên nghiệp và hỗ trợ sinh viên trong việc triển khai và thực hiện đồ án môn học.

Em xin chân thành cảm ơn!

**Trân trọng**

Sinh viên thực hiện

Nguyễn Đình Khánh Vy

# TÓM TẮT

Trong các hệ thống vi mạch hiện đại, đặc biệt là hệ thống trên chip (SoC), việc truyền nhận dữ liệu giữa các khối chức năng như bộ xử lý trung tâm, bộ nhớ và các ngoại vi đóng vai trò then chốt trong việc đảm bảo hiệu suất tổng thể và độ tin cậy của hệ thống. Để thực hiện điều này, các giao thức truyền thông nội bộ phải được sử dụng để kết nối và điều phối hoạt động giữa các thành phần cứng với nhau một cách hiệu quả.

Trong các giao thức truyền thông nội bộ hiện nay, AXI (Advanced eXtensible Interface) là một thành phần trong kiến trúc bus AMBA (Advanced Microcontroller Bus Architecture) do hãng ARM phát triển đã trở thành chuẩn giao tiếp phổ biến trong thiết kế SoC. Giao thức AXI nổi bật với khả năng truyền dữ liệu tốc độ cao, hỗ trợ truyền song song nhiều luồng dữ liệu, tối ưu hóa băng thông và khả năng hoạt động ổn định ở tần số cao.

Đề tài này tôi thực hiện xây dựng và mô phỏng mô hình giao tiếp AXI4 theo cấu trúc mô hình điểm – điểm giữa một MASTER và một SLAVE sử dụng ngôn ngữ mô tả phần cứng Verilog. Quá trình thiết kế dựa trên việc nghiên cứu lý thuyết từ đặc tả chuẩn AMBA AXI, từ đó phân tích và xây dựng khối thành phần độc lập là khối Master và khối Slave, rồi tích hợp thành một mô hình hoàn chỉnh đáp ứng đầy đủ các kênh truyền dữ liệu theo chuẩn AXI4. Toàn bộ quá trình thiết kế và mô phỏng được thực hiện trên phần mềm Xilinx Vivado phiên bản 2022.2.

Sau quá trình thiết kế và mô phỏng, mô hình giao tiếp điểm – điểm theo chuẩn giao tiếp AXI4 giữa một khối Master và một Slave đã hoạt động đúng theo đặc tả giao thức AMBA AXI4. Các testcase được xây dựng để kiểm tra quá trình truyền dữ liệu đọc và ghi với ba cơ chế truyền dữ liệu cho thấy hệ thống đáp ứng chính xác các yêu cầu về truyền dữ liệu với các kích thước và độ dài khác nhau. Kết quả dạng sóng mô phỏng xác nhận tính toàn vẹn của dữ liệu truyền qua các kênh giao tiếp, đồng thời chứng minh khả năng hoạt động ổn định và hiệu quả của hệ thống. Tuy nhiên, đề tài này chỉ mới dừng lại ở mức thiết kế và mô phỏng kiểm tra dạng sóng, chưa triển khai trên phần cứng thực tế.

# MỤC LỤC

|                                                                      |     |
|----------------------------------------------------------------------|-----|
| DANH MỤC HÌNH.....                                                   | VII |
| DANH MỤC BẢNG.....                                                   | IX  |
| CÁC TỪ VIẾT TẮT .....                                                | X   |
| CHƯƠNG 1 TỔNG QUAN.....                                              | 1   |
| 1.1 GIỚI THIỆU .....                                                 | 1   |
| 1.2 MỤC TIÊU NGHIÊN CỨU .....                                        | 2   |
| 1.3 TÌNH HÌNH NGHIÊN CỨU .....                                       | 3   |
| 1.4 BỐ CỤC CỦA ĐỀ TÀI.....                                           | 4   |
| CHƯƠNG 2 CƠ SỞ LÝ THUYẾT.....                                        | 5   |
| 2.1 TỔNG QUAN VỀ HỆ THỐNG TRÊN CHIP.....                             | 5   |
| 2.1.1 Định nghĩa.....                                                | 5   |
| 2.1.2 Ưu và nhược điểm của SoC .....                                 | 7   |
| 2.2 TỔNG QUAN VỀ HỆ THỐNG BUS TRONG SOC .....                        | 8   |
| 2.3 TỔNG QUAN VỀ GIAO THỨC AMBA AXI.....                             | 10  |
| 2.3.1 Giới thiệu chung về giao thức AMBA AXI.....                    | 10  |
| 2.3.2 Cấu trúc bus AXI.....                                          | 11  |
| 2.3.3 Các quy định về hoạt động.....                                 | 15  |
| 2.3.4 Giao thức AMBA AXI4.....                                       | 16  |
| 2.3.4.1 Giới thiệu về giao thức AMBA AXI4 .....                      | 16  |
| 2.3.4.2 Cơ chế bắt tay của AXI4 .....                                | 17  |
| 2.3.4.3 Cơ chế burst của AXI4 .....                                  | 19  |
| 2.4 TỔNG QUAN VỀ NGÔN NGỮ MÔ TẢ CỨNG VERILOG VÀ PHẦN MỀM MÔ<br>PHỎNG | 21  |
| 2.4.1 Ngôn ngữ mô tả phần cứng Verilog .....                         | 21  |
| 2.4.2 Phần mềm XILINX VIVADO 2022.2.....                             | 23  |
| CHƯƠNG 3 THIẾT KẾ HỆ THỐNG.....                                      | 24  |
| 3.1 YÊU CẦU HỆ THỐNG .....                                           | 24  |
| 3.2 THIẾT KẾ HỆ THỐNG SỬ DỤNG GIAO THỨC AMBA AXI4 .....              | 24  |

|                                                   |                                                                      |           |
|---------------------------------------------------|----------------------------------------------------------------------|-----------|
| 3.2.1                                             | Thiết kế khối AXI4 MASTER .....                                      | 29        |
| 3.2.2                                             | Thiết kế khối AXI4 SLAVE .....                                       | 30        |
| <b>CHƯƠNG 4 KẾT QUẢ MÔ PHỎNG.....</b>             |                                                                      | <b>32</b> |
| <b>4.1</b>                                        | <b>TÓM TẮT NỘI DUNG.....</b>                                         | <b>32</b> |
| <b>4.2</b>                                        | <b>MÔ PHỎNG HỆ THỐNG SỬ DỤNG PHẦN MỀM XILINX VIVADO 2022.2 .....</b> | <b>32</b> |
| 4.2.1                                             | Hoạt động ghi dữ liệu .....                                          | 36        |
| 4.2.1.1                                           | <i>Testcase 1</i> .....                                              | 36        |
| 4.2.1.2                                           | <i>Testcase 2</i> .....                                              | 41        |
| 4.2.1.3                                           | <i>Testcase 3</i> .....                                              | 45        |
| 4.2.1.4                                           | <i>Testcase 4</i> .....                                              | 50        |
| 4.2.2                                             | Hoạt động đọc dữ liệu.....                                           | 52        |
| 4.2.2.1                                           | <i>Testcase 1</i> .....                                              | 52        |
| 4.2.2.2                                           | <i>Testcase 2</i> .....                                              | 55        |
| 4.2.2.3                                           | <i>Testcase 3</i> .....                                              | 59        |
| 4.2.2.4                                           | <i>Testcase 4</i> .....                                              | 64        |
| <b>CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....</b> |                                                                      | <b>66</b> |
| <b>5.1</b>                                        | <b>KẾT LUẬN .....</b>                                                | <b>66</b> |
| <b>5.2</b>                                        | <b>HƯỚNG PHÁT TRIỂN .....</b>                                        | <b>67</b> |
| <b>TÀI LIỆU THAM KHẢO .....</b>                   |                                                                      | <b>68</b> |
| <b>PHỤ LỤC .....</b>                              |                                                                      | <b>70</b> |

# DANH MỤC HÌNH

|                   |                                                                     |    |
|-------------------|---------------------------------------------------------------------|----|
| <b>Hình 2.1.</b>  | Thành phần cơ bản của SoC.....                                      | 7  |
| <b>Hình 2.2.</b>  | Hệ thống bus trong SoC.....                                         | 9  |
| <b>Hình 2.3.</b>  | Giao diện giao tiếp và kết nối bus.....                             | 11 |
| <b>Hình 2.4.</b>  | Các kênh giao tiếp giữa Master và Slave.....                        | 12 |
| <b>Hình 2.5.</b>  | Hoạt động của giao dịch đọc.....                                    | 15 |
| <b>Hình 2.6.</b>  | Hoạt động của giao dịch ghi .....                                   | 16 |
| <b>Hình 2.7.</b>  | Tín hiệu VALID tích cực trước tín hiệu READY.....                   | 18 |
| <b>Hình 2.8.</b>  | Tín hiệu VALID tích cực sau tín hiệu READY.....                     | 18 |
| <b>Hình 2.9.</b>  | Tín hiệu VALID tích cực cùng lúc với tín hiệu READY.....            | 19 |
| <b>Hình 2.10.</b> | Minh họa cơ chế burst .....                                         | 19 |
| <b>Hình 3.1.</b>  | Mô hình kết nối điểm – điểm của AXI4 bus.....                       | 25 |
| <b>Hình 3.2.</b>  | Sơ đồ khối chi tiết của khối AXI4 MASTER .....                      | 29 |
| <b>Hình 3.3.</b>  | Sơ đồ khối chi tiết của khối AXI4 SLAVE .....                       | 31 |
| <b>Hình 4.1.</b>  | Dạng sóng trường hợp 1 testcase 1 của quá trình ghi .....           | 37 |
| <b>Hình 4.2.</b>  | Bộ nhớ trường hợp 1 testcase 1 của AXI4 SLAVE quá trình ghi .....   | 38 |
| <b>Hình 4.3.</b>  | Dạng sóng trường hợp 2 testcase 1 của quá trình ghi .....           | 38 |
| <b>Hình 4.4.</b>  | Bộ nhớ trường hợp 2 testcase 1 của AXI4 SLAVE quá trình ghi .....   | 39 |
| <b>Hình 4.5.</b>  | Dạng sóng trường hợp 3 testcase 1 của quá trình ghi .....           | 40 |
| <b>Hình 4.6.</b>  | Bộ nhớ trường hợp 3 testcase 1 của AXI SLAVE cho quá trình ghi..... | 41 |
| <b>Hình 4.7.</b>  | Dạng sóng trường hợp 1 testcase 2 của quá trình ghi .....           | 42 |
| <b>Hình 4.8.</b>  | Bộ nhớ trường hợp 1 testcase 2 của AXI4 SLAVE quá trình ghi .....   | 43 |
| <b>Hình 4.9.</b>  | Dạng sóng trường hợp 2 testcase 2 của quá trình ghi .....           | 44 |
| <b>Hình 4.10.</b> | Bộ nhớ trường hợp 2 testcase 2 của AXI4 SLAVE quá trình ghi.....    | 45 |
| <b>Hình 4.11.</b> | Dạng sóng trường hợp 1 testcase 3 của quá trình ghi .....           | 46 |
| <b>Hình 4.12.</b> | Bộ nhớ trường hợp 1 testcase 3 của AXI4 SLAVE quá trình ghi.....    | 47 |
| <b>Hình 4.13.</b> | Dạng sóng trường hợp 2 testcase 3 của quá trình ghi .....           | 48 |
| <b>Hình 4.14.</b> | Bộ nhớ trường hợp 2 testcase 3 của AXI SLAVE quá trình ghi .....    | 50 |
| <b>Hình 4.15.</b> | Dạng sóng testcase 4 của quá trình ghi.....                         | 51 |
| <b>Hình 4.16.</b> | Dạng sóng trường hợp 1 testcase 1 của quá trình đọc.....            | 53 |
| <b>Hình 4.17.</b> | Bộ nhớ trường hợp 1 testcase 1 của AXI4 MASTER quá trình đọc.....   | 54 |

|                   |                                                                   |    |
|-------------------|-------------------------------------------------------------------|----|
| <b>Hình 4.18.</b> | Dạng sóng trường hợp 2 testcase 1 của quá trình đọc.....          | 54 |
| <b>Hình 4.19.</b> | Bộ nhớ trường hợp 2 testcase 1 của AXI4 MASTER quá trình đọc..... | 55 |
| <b>Hình 4.20.</b> | Dạng sóng trường hợp 1 testcase 2 của quá trình đọc.....          | 56 |
| <b>Hình 4.21.</b> | Bộ nhớ trường hợp 1 testcase 2 của AXI4 MASTER quá trình đọc..... | 57 |
| <b>Hình 4.22.</b> | Dạng sóng trường hợp 2 testcase 2 của quá trình đọc.....          | 58 |
| <b>Hình 4.23.</b> | Bộ nhớ trường hợp 2 testcase 2 của AXI4 MASTER quá trình đọc..... | 59 |
| <b>Hình 4.24.</b> | Dạng sóng trường hợp 1 testcase 3 của quá trình đọc.....          | 60 |
| <b>Hình 4.25.</b> | Bộ nhớ trường hợp 1 testcase 3 của AXI4 MASTER quá trình đọc..... | 61 |
| <b>Hình 4.26.</b> | Dạng sóng trường hợp 2 testcase 3 của quá trình ghi .....         | 62 |
| <b>Hình 4.27.</b> | Bộ nhớ trường hợp 2 testcase 3 của AXI MASTER quá trình đọc.....  | 64 |
| <b>Hình 4.28.</b> | Dạng sóng testcase 4 của quá trình đọc .....                      | 65 |



## DANH MỤC BẢNG

|                                                                         |    |
|-------------------------------------------------------------------------|----|
| <b>Bảng 2.1.</b> Các tín hiệu của kênh địa chỉ ghi .....                | 12 |
| <b>Bảng 2.2.</b> Các tín hiệu của kênh dữ liệu ghi.....                 | 13 |
| <b>Bảng 2.3.</b> Các tín hiệu của kênh phản hồi ghi .....               | 14 |
| <b>Bảng 2.4.</b> Các tín hiệu của kênh địa chỉ đọc .....                | 14 |
| <b>Bảng 2.5.</b> Các tín hiệu của kênh dữ liệu đọc .....                | 15 |
| <b>Bảng 2.6.</b> Các cặp tín hiệu bắt tay trong giao thức AXI4 .....    | 18 |
| <b>Bảng 2.7.</b> Kích thước dữ liệu tương ứng với giá trị AxLEN .....   | 20 |
| <b>Bảng 2.8.</b> Các loại burst tương ứng với giá trị AxBURST .....     | 21 |
| <b>Bảng 3.1.</b> Các chân tín hiệu được sử dụng trong AXI4 MASTER ..... | 26 |
| <b>Bảng 3.2.</b> Các chân tín hiệu được sử dụng trong AXI4 SLAVE.....   | 27 |
| <b>Bảng 4.1.</b> Giới thiệu các testcase .....                          | 33 |
| <b>Bảng 4.2.</b> Thông số đầu vào của trường hợp 1 testcase 1 .....     | 36 |
| <b>Bảng 4.3.</b> Thông số đầu vào của trường hợp 2 testcase 1 .....     | 38 |
| <b>Bảng 4.4.</b> Thông số đầu vào của trường hợp 3 testcase 1 .....     | 40 |
| <b>Bảng 4.5.</b> Thông số đầu vào của trường hợp 1 testcase 2 .....     | 41 |
| <b>Bảng 4.6.</b> Thông số đầu vào của trường hợp 2 testcase 2 .....     | 43 |
| <b>Bảng 4.7.</b> Thông số đầu vào của trường hợp 1 testcase 3 .....     | 46 |
| <b>Bảng 4.8.</b> Thông số đầu vào của trường hợp 2 testcase 3 .....     | 48 |
| <b>Bảng 4.9.</b> Thông số đầu vào của testcase 4 .....                  | 50 |
| <b>Bảng 4.10.</b> Thông số đầu vào của trường hợp 1 testcase 1 .....    | 52 |
| <b>Bảng 4.11.</b> Thông số đầu vào của trường hợp 2 testcase 1 .....    | 54 |
| <b>Bảng 4.12.</b> Thông số đầu vào của trường hợp 1 testcase 2 .....    | 56 |
| <b>Bảng 4.13.</b> Thông số đầu vào của trường hợp 2 testcase 2 .....    | 58 |
| <b>Bảng 4.14.</b> Thông số đầu vào của trường hợp 1 testcase 3 .....    | 60 |
| <b>Bảng 4.15.</b> Thông số đầu vào của trường hợp 2 testcase 3 .....    | 62 |
| <b>Bảng 4.16.</b> Thông số đầu vào của testcase 4 .....                 | 64 |

## CÁC TỪ VIẾT TẮT

|                  |                                                         |
|------------------|---------------------------------------------------------|
| ADC              | Analog-to-Digital Converter                             |
| AHB              | Advanced High-performance Bus                           |
| AI               | Artificial Intelligence                                 |
| AMBA             | Advanced Microcontroller Bus Architecture               |
| APB              | Advanced Peripheral Bus                                 |
| ARM              | Advanced RISC Machine                                   |
| ASIC             | Application-Specific Integrated Circuit                 |
| AXI              | Advanced eXtensible Interface                           |
| CPU              | Central Processing Unit                                 |
| DMA              | Direct Memory Access                                    |
| DSP              | Digital Signal Processor                                |
| EEPROM           | Electrically Erasable Programmable Read-Only Memory     |
| FLASH            | Flash Memory                                            |
| FPGA             | Field-Programmable Gate Array                           |
| FSM              | Finite State Machine                                    |
| GPU              | Graphics Processing Unit                                |
| HDMI             | High-Definition Multimedia Interface                    |
| I/O              | Input/Output                                            |
| I <sup>2</sup> C | Inter-Integrated Circuit                                |
| IoT              | Internet of Things                                      |
| IP               | Intellectual Property                                   |
| NoC              | Network-on-Chip                                         |
| PC               | Personal Computer                                       |
| PLL              | Phase-Locked Loop                                       |
| RAM              | Random Access Memory                                    |
| ROM              | Read-Only Memory                                        |
| SoC              | System-on-Chip                                          |
| SPI              | Serial Peripheral Interface                             |
| USART            | Universal Synchronous/Asynchronous Receiver/Transmitter |
| USB              | Universal Serial Bus                                    |

# CHƯƠNG 1

## TỔNG QUAN

### 1.1 GIỚI THIỆU

Trong bối cảnh công nghệ vi mạch và hệ thống nhúng đang phát triển mạnh mẽ, các hệ thống trên một chip ngày càng trở nên phổ biến và đóng vai trò quan trọng trong các thiết bị điện tử hiện đại như điện thoại di động, thiết bị Internet vạn vật (IoT), xe tự hành, các nền tảng xử lý tín hiệu số. Những hệ thống này thường tích hợp nhiều khối chức năng khác nhau trên cùng một chip vật lý như bộ xử lý, bộ điều khiển bộ nhớ, các thiết bị ngoại vi và các khối xử lý chuyên dụng.

Để đảm bảo các khối chức năng này có thể hoạt động một cách hiệu quả và đồng bộ, việc thiết kế hệ thống truyền dữ liệu nội bộ đóng vai trò then chốt. Các phương thức truyền dữ liệu trong hệ thống trên một chip có thể được triển khai theo nhiều hình thức khác nhau như dạng điểm – điểm, bus chia sẻ hay mạng trên chip (NoC). Trong đó, kiến trúc bus đóng vai trò như một thành phần trung gian giúp kết nối và điều phối luồng dữ liệu giữa các thành phần trong hệ thống.

Hiện nay, một số các kiến trúc bus được sử dụng rộng rãi trong thiết kế SoC như AMBA (Advanced Microcontroller Bus Architecture) AHB (Advanced High-performance Bus), APB (Advanced Peripheral Bus) và AXI (Advanced eXtensible Interface) do hãng ARM phát triển, WISHBONE do cộng đồng OpenCores đề xuất, CoreConnect do IBM phát triển.

Trong đề tài này, tôi lựa chọn tìm hiểu, xây dựng và mô phỏng giao thức AMBA AXI4 đơn giản theo mô hình điểm – điểm giữa một Master và một Slave. Lý do lựa chọn mô hình điểm – điểm xuất phát từ mục tiêu làm rõ và kiểm chứng nguyên lý hoạt động cơ bản của giao thức AXI4 mà không bị ảnh hưởng bởi sự

phức tạp của các kiến trúc nhiều liên kết giữa Master và Slave như phương thức truyền dữ liệu bus chia sẻ hay mạng trên chip. Việc lựa chọn mô hình điểm – điểm giúp tôi tập trung xem xét vào từng kênh truyền dữ liệu riêng lẻ, dễ theo dõi và đánh giá các luồng tín hiệu, thuận lợi cho quá trình mô phỏng, phân tích và kiểm chứng logic. Giao thức AXI4 được tôi lựa chọn vì đây là chuẩn giao tiếp cao cấp và phổ biến nhất trong các giao thức thuộc họ AMBA do hãng ARM phát triển. Giao thức này được sử dụng rộng rãi trong các hệ thống trên một chip hiện đại nhờ kiến trúc phân tách năm kênh giao tiếp, hỗ trợ truyền dữ liệu tốc độ cao, khả năng đọc và ghi dữ liệu song song, đồng thời đảm bảo hiệu suất truyền dữ liệu cao, phù hợp với các hệ thống yêu cầu băng thông lớn và độ trễ thấp.

Trong quá trình thực hiện, đề tài đi sâu vào phân tích nguyên lý hoạt động của giao thức AXI4 theo chuẩn đặc tả của hãng ARM, từ đó xây dựng các module phần cứng bằng ngôn ngữ mô tả phần cứng Verilog, bao gồm các máy trạng thái FSM (Finite State Machine) điều khiển cho các kênh: kênh địa chỉ ghi (write address), kênh dữ liệu ghi (write data), kênh phản hồi ghi (write response), kênh địa chỉ đọc (read address), và kênh đọc dữ liệu (read data).

Việc thiết kế được thực hiện một cách hệ thống, từ phân tích tín hiệu, thiết kế mô hình máy trạng thái cho từng module cho đến tích hợp tổng thể thành một hệ thống AXI4 tối giản nhưng hoạt động đúng theo nguyên lý. Toàn bộ quá trình được mô phỏng, kiểm chứng trên phần mềm Xilinx Vivado phiên bản 2022.2, với khả năng mô phỏng thời gian thực, hỗ trợ phân tích dạng sóng của các tín hiệu, giúp đảm bảo tính đúng đắn và hiệu quả của hệ thống thiết kế.

## **1.2 MỤC TIÊU NGHIÊN CỨU**

Mục tiêu của đề tài này là thiết kế mô hình giao tiếp bus theo chuẩn AXI4 (Advance eXtensible Interface) sử dụng ngôn ngữ Verilog. Đề tài hướng tới việc xây dựng một mô hình điểm – điểm đơn giản, bao gồm một khối AXI4 MASTER và một khối AXI4 SLAVE, tuân thủ đầy đủ các đặc tả kỹ thuật của giao thức AMBA AXI4 do hãng ARM phát triển, nhằm thực hiện quá trình truyền và nhận dữ liệu một cách chính xác và hiệu quả. Trong quá trình thực hiện, đề tài tập trung xây dựng các tập kiểm thử nhằm mô phỏng quá trình truyền

dữ liệu giữa Master và Slave, kiểm tra khả năng đáp ứng của mô hình trong các cơ chế truyền dữ liệu khác nhau như FIXED, INCR và WRAP. Đồng thời, các trường hợp kiểm thử cũng bao gồm nhiều trường hợp độ dài truyền và kích thước dữ liệu, nhằm đánh giá tính linh hoạt và khả năng xử lý của mô hình trong nhiều tình huống thực tế.

### **1.3 TÌNH HÌNH NGHIÊN CỨU**

Những năm gần đây sự phát triển nhanh chóng của các ứng dụng nhúng phức tạp đã thúc đẩy mạnh mẽ xu hướng thiết kế hệ thống tích hợp cao, đặc biệt là các kiến trúc hệ thống trên chip. Tuy nhiên, sự tích hợp ngày càng cao cũng đặt ra yêu cầu khắt khe về một kiến trúc giao tiếp nội bộ đủ mạnh, có khả năng truyền dữ liệu nhanh chóng, đáng tin cậy và có thể mở rộng một cách linh hoạt.

Để giải quyết bài toán này, hãng ARM đã phát triển dòng giao thức bus AMBA, một kiến trúc chuẩn giao tiếp nội bộ cho các hệ thống SoC. Trong đó, giao thức AXI, đặc biệt là phiên bản AXI4 là giao thức nổi bật nhất và được sử dụng rộng rãi trong các thiết kế công nghiệp hiện đại nhờ khả năng hỗ trợ truy cập song song, truyền dữ liệu có độ trễ thấp, hỗ trợ nhiều chế độ burst và tương thích cao giữa các IP.

Nhiều nghiên cứu đã được thực hiện nhằm phân tích, tối ưu và triển khai giao thức AMBA AXI4. Cụ thể trong bài báo [1], các tác giả tập trung vào việc thiết kế mô hình giao thức AMBA AXI có thể tổng hợp được nhằm phục vụ cho mục tiêu tích hợp vào các hệ thống SoC. Nghiên cứu này không chỉ phân tích chi tiết về cấu trúc và các tín hiệu của giao thức mà còn đưa ra một phương pháp thiết kế hướng đến việc đơn giản hóa quá trình xác minh và kiểm tra tính đúng đắn của các giao dịch truyền dữ liệu trên chip. Đây cũng là một trong những vấn đề thường gặp trong các hệ thống phức tạp.

Trong bài báo [2], tác giả đề xuất và hiện thực mô hình AXI4 Master sử dụng ngôn ngữ mô tả phần cứng Verilog. Thiết kế này hỗ trợ cơ chế truyền dữ liệu có độ dài lên tới 32 beats, đây là một tính năng quan trọng giúp tăng băng thông và giảm overhead khi giao tiếp truyền dữ liệu. Ngoài ra, các giao dịch bị khóa được loại bỏ nhằm tăng tính linh hoạt, đồng thời hệ thống cũng cải thiện đáng kể khả

năng phản hồi ghi. Toàn bộ thiết kế được kiểm chứng qua mô phỏng trên công cụ Xilinx Vivado, chứng minh được tính khả thi và hiệu quả của mô hình.

Trong hướng nghiên cứu khác, bài báo [3] đi sâu vào thiết kế mô hình AXI4 tổng quát, hỗ trợ tối đa 16 master và 16 slave, với khả năng cấu hình địa chỉ linh hoạt. Hệ thống được thực hiện cũng bằng ngôn ngữ mô tả phần cứng Verilog và kiểm thử thông qua Verilog Compiler Simulator (VCS) ở tần số hoạt động 100MHz. Kết quả thực nghiệm cho thấy, thời gian xử lý một giao dịch đọc đơn là 160ns, trong khi giao dịch ghi đơn mất khoảng 565ns, từ đó cho thấy hiệu năng và khả năng mở rộng của giao thức.

Trong bài báo [4], các tác giả tiến hành xây dựng hệ thống kết nối điểm – điểm sử dụng kiến trúc bus WISHBONE và AMBA AXI. Sau đó, tiến hành so sánh, đánh giá về tài nguyên sử dụng và công suất tiêu thụ của mỗi hệ thống thông qua công cụ trên phần mềm Xilinx Vivado 2019.1. Kết quả thực nghiệm cho thấy, tài nguyên sử dụng và công suất tiêu thụ của hệ thống bus WISHBONE ít hơn so với của AMBA AXI. Tuy nhiên thời gian và hiệu suất truyền của hệ thống bus AMBA AXI tốt hơn của WISHBONE.

## **1.4 BỐ CỤC CỦA ĐỀ TÀI**

**Chương 1. Giới thiệu:** Trình bày tổng quan về đề tài, mục tiêu đề tài, tình hình nghiên cứu và phương pháp nghiên cứu được sử dụng trong đồ án môn học.

**Chương 2. Cơ sở lý thuyết:** Tổng quan về bus của SoC, mô tả về khái niệm, đặc điểm, kiến trúc, nguyên lý hoạt động của giao thức AXI4.

**Chương 3. Thiết kế hệ thống:** Trình bày chi tiết về sơ đồ khối của hệ thống, nguyên lý hoạt động của các khối thành phần bên trong.

**Chương 4. Kết quả mô phỏng:** Thực hiện kiểm tra chức năng truyền dữ liệu của các kênh, kiểm tra từng testcase để đảm bảo mô hình hoạt động chính xác.

**Chương 5. Kết luận và hướng phát triển:** Kết luận, tổng kết về đề tài với những kết quả đạt được, những mặt hạn chế và đề xuất hướng phát triển đề tài.

## CHƯƠNG 2

# CƠ SỞ LÝ THUYẾT

### 2.1 TỔNG QUAN VỀ HỆ THỐNG TRÊN CHIP

#### 2.1.1 Định nghĩa

Hệ thống trên chip (SoC) là một vi mạch tích hợp (IC) chứa gần như toàn bộ các khối chức năng cần thiết để tạo thành một hệ thống điện tử hoạt động độc lập trên một chip duy nhất. Một SoC thường được thiết kế tích hợp những thành phần bao gồm: bộ xử lý trung tâm, bộ nhớ, cổng I/O, các khối xử lý tín hiệu tương tự, các bộ điều khiển giao tiếp và nhiều khối chức năng chuyên biệt khác. Hệ thống trên chip khác với các thiết bị truyền thống và kiến trúc PC thông thường ở chỗ các thành phần như CPU, GPU, RAM và các thiết bị ngoại vi thường được bố trí trên các chip rời rạc và kết nối với nhau qua bo mạch chủ. Trong khi đó, SoC hợp nhất toàn bộ các thành phần này vào một vi mạch duy nhất. Điều này giúp giảm kích thước hệ thống, tiết kiệm điện năng, tăng hiệu suất xử lý và giảm độ trễ truyền dữ liệu giữa các khối chức năng [5].

Tùy thuộc vào mục đích sử dụng, SoC có thể tích hợp thêm các thành phần có chức năng chuyên biệt nhằm đáp ứng những nhu cầu sử dụng và các yêu cầu kỹ thuật khác nhau. Ví dụ, trên điện thoại thông minh hoặc các thiết bị IoT, hệ thống trên chip thường tích hợp thêm bộ thu phát sóng di động, WiFi, Bluetooth và các module giao tiếp không dây khác. Việc tích hợp trực tiếp các thành phần này trên cùng một mạch cho phép rút ngắn khoảng cách kết nối nội bộ giữa các khối chức năng, từ đó giúp giảm tiêu thụ điện năng, cải thiện tốc độ truyền dữ liệu và tăng cường độ tin cậy của toàn bộ hệ thống.

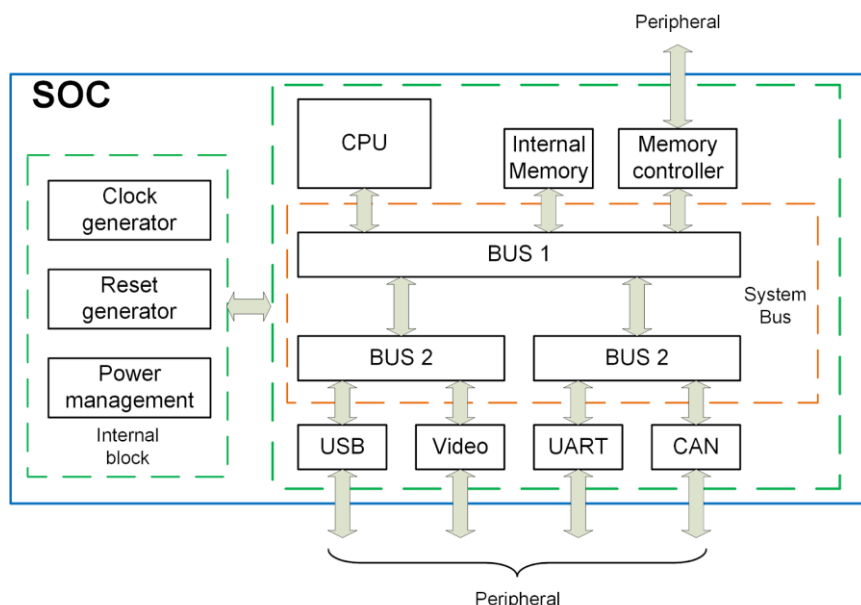
Hệ thống trên chip hiện đại thường bao gồm các khối chức năng chính sau:

- **Bộ xử lý đa lõi:** Bộ xử lý có đa lõi có thể là vi điều khiển, bộ vi xử lý, bộ xử lý tín hiệu số hoặc bộ xử lý chuyên dụng được thiết kế theo tập lệnh riêng.
- **Khả năng bộ nhớ:** Tích hợp các loại bộ nhớ như RAM, ROM, FLASH, EEPROM cùng các bộ nhớ đệm (cache) để nâng cao tốc độ truy xuất dữ liệu.
- **Giao tiếp ngoại vi:** Hỗ trợ các chuẩn truyền thông có dây như: HDMI, USB4, FireWire, USART, SPI, I<sup>2</sup>C, và Ethernet, cho phép kết nối linh hoạt với các thiết bị bên ngoài.
- **Khả năng không dây:** Bao gồm các chuẩn truyền không dây như WiFi, Bluetooth và các thành phần hỗ trợ giao tiếp tần số vô tuyến.
- **Bộ xử lý đồ họa:** Đảm nhiệm các tác vụ xử lý hình ảnh và tăng tốc đồ họa, đặc biệt quan trọng với các ứng dụng yêu cầu hiển thị phức tạp như thiết bị di động hoặc hệ thống nhúng đa phương tiện.
- **Điều chỉnh điện áp:** Gồm các thành phần như bộ điều áp, mạch điều khiển vòng khóa pha (PLL), bộ dao động nội, bộ định thời và bộ chuyển đổi tương tự - số (ADC).
- **Giao tiếp nội bộ:** Sử dụng các kiến trúc bus hoặc mạng trên chip để kết nối các khối chức năng bên trong hệ thống một cách hiệu quả và có thể mở rộng.
- **Xử lý tín hiệu số:** Bao gồm các khối xử lý tín hiệu số, tín hiệu tương tự hoặc hỗn hợp dùng cho các cảm biến, thu thập dữ liệu và phân tích tín hiệu dữ liệu theo thời gian thực.
- **Tính năng cho thế hệ mới:** Tích hợp các công nghệ tiên tiến như khả năng mở rộng phần cứng, hiệu năng cao, trí tuệ nhân tạo và các giải pháp bảo mật phần cứng nhằm đáp ứng yêu cầu ngày càng cao trong các hệ thống thông minh [5].

Một SoC cơ bản được minh họa trong hình 2.1 bên dưới, trình bày kiến trúc chứa các thành phần chức năng chính như CPU, bộ nhớ trong, bộ điều khiển bộ nhớ và các thiết bị ngoại vi. Các thành phần này được kết nối thông qua hệ thống



bus phân cấp, bao gồm BUS 1 trung tâm và các BUS 2 phụ. Mô hình sử dụng kết nối điểm – điểm đơn giản, phù hợp với các hệ thống nhúng có quy mô nhỏ, yêu cầu ít tài nguyên. Kiến trúc này giúp dễ dàng kiểm soát quá trình truyền dữ liệu và đơn giản hóa thiết kế phần cứng, đồng thời vẫn đảm bảo khả năng hoạt động ổn định trong các ứng dụng nhúng cơ bản.



**Hình 2.1.** Thành phần cơ bản của SoC

### 2.1.2 Ưu và nhược điểm của SoC

Ưu điểm của SoC là có thể sử dụng cho các thiết bị điện tử cần kích thước nhỏ như điện thoại, máy tính bảng và các thiết bị thông minh khác. SoC có mật độ tích hợp cao, nhiều chức năng nên việc làm phần cứng sản phẩm đơn giản hơn, dễ sản xuất hơn, chi phí thấp hơn, thời gian thiết kế sản phẩm nhanh hơn. Sản xuất số lượng đủ lớn, giá thành sẽ giảm đáng kể. Sản phẩm sử dụng SoC tiết kiệm năng lượng hơn so với sản phẩm cùng chức năng nhưng không dùng SoC vì số lượng linh kiện lớn hơn, bo mạch làm phức tạp hơn. Khi sản phẩm càng nhiều chức năng thì đặc điểm này càng thấy rõ [6].

Ngoài ra, việc thay thế các linh kiện điện tử rời và các bảng mạch lớn bằng SoC không chỉ giúp giảm thiểu đáng kể mức tiêu thụ điện năng mà còn cải thiện đáng kể các chỉ số kỹ thuật quan trọng như công suất, hiệu suất, mật độ tích hợp và độ tin cậy của hệ thống. Một SoC có số lượng kết nối vật lý ít hơn so với hệ

thống đa linh kiện như các thiết bị truyền thông, từ đó làm giảm khả năng xảy ra lỗi kết nối và tăng độ ổn định trong vận hành [7].

Nhược điểm của SoC là một loại SoC khó đáp ứng nhu cầu của nhiều loại nhu cầu sản phẩm khác nhau. Chính vì vậy, mỗi hãng thiết kế và sản xuất chip đều có nhiều dòng SoC khác nhau, mỗi dòng sẽ đáp ứng một phân khúc sản phẩm nhất định và tối ưu nhất cho phân khúc sản phẩm này. Tối ưu ở đây được hiểu là số lượng chức năng mà chip tích hợp là vừa đủ, không quá nhiều và cũng không quá ít. Quá nhiều chức năng thì giá chip sẽ tăng, giá sản phẩm tăng trong khi chức năng thừa không được dùng đến. Quá ít chức năng thì không đáp ứng được nhu cầu ứng dụng của sản phẩm hoặc làm sản phẩm khó, thời gian ra thị trường lâu [6].

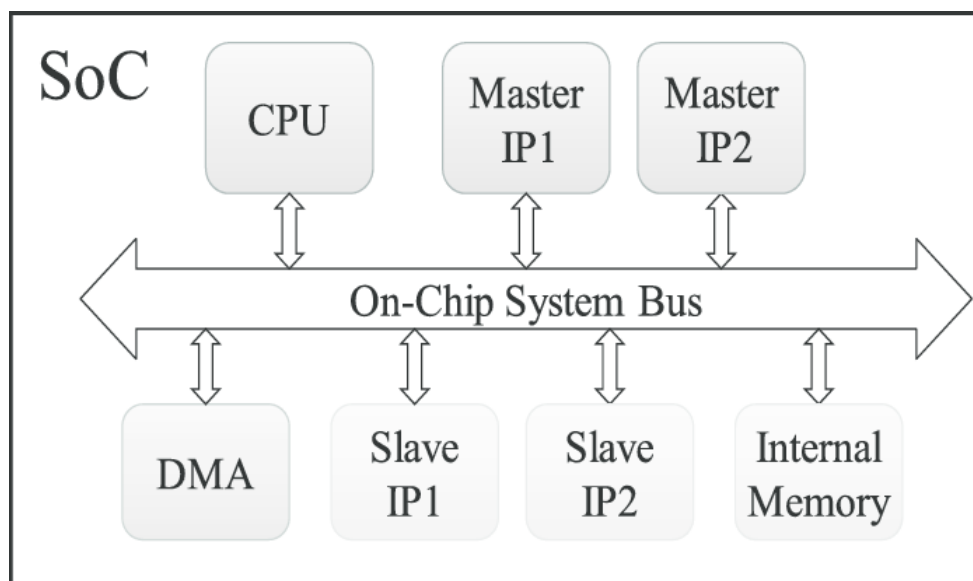
## **2.2 TỔNG QUAN VỀ HỆ THỐNG BUS TRONG SOC**

Trong kiến trúc hệ thống trên chip, hệ thống bus là cầu nối phục vụ cho mục đích truy xuất dữ liệu đến một thành phần trong hệ thống. Về bản chất, bus là một tập hợp các đường truyền tín hiệu dùng chung, cho phép nhiều thành phần có thể giao tiếp và phối hợp hoạt động một cách có tổ chức trong hệ thống [4].

Trong các thiết kế SoC hiện đại, đặc biệt là những hệ thống SoC phức tạp, nhiều loại bus khác nhau có thể cùng tồn tại và hoạt động song song. Mỗi loại bus thường phục vụ một mục đích cụ thể, kết nối đến một nhóm khối chức năng nhất định và hoạt động ở tần số phù hợp với yêu cầu hiệu suất của từng khối chức năng [4].

Hình 2.2 bên dưới minh họa một hệ thống bus trong hệ thống trên chip, trong đó các khối chức năng được tổ chức và kết nối thông qua một bus trung tâm nhằm đảm bảo khả năng truyền dữ liệu giữa các khối chức năng. Kiến trúc này thể hiện mô hình kết nối đa Master - đa Slave cùng chia sẻ dữ liệu chung một hệ thống bus. Các thành phần Master bao gồm: bộ xử lý trung tâm và hai khối IP1, IP2, có khả năng khởi tạo các giao dịch đọc và ghi dữ liệu trên bus. Trong khi đó, các Slave bao gồm bộ điều khiển truy cập trực tiếp vào bộ nhớ (DMA), hai khối IP1 và IP2, bộ nhớ trong, đóng vai trò phản hồi các yêu cầu từ Master. Với kiến trúc này phản ánh cách tổ chức các khối chức năng phổ biến trong các SoC hiện

đại, cho phép tích hợp linh hoạt nhiều khối chức năng với nhau, tối ưu hóa việc chia sẻ tài nguyên phần cứng trên một hệ thống tích hợp duy nhất.



**Hình 2.2.** Hệ thống bus trong SoC

Các giao thức bus đóng vai trò quan trọng trong việc xác định phương thức trao đổi dữ liệu giữa các khối chức năng trong hệ thống trên chip. Mỗi loại giao thức bus được thiết kế với cấu trúc kỹ thuật và tính năng chuyên biệt, nhằm phục vụ các yêu cầu khác nhau về kỹ thuật như tốc độ truyền, băng thông, mức độ phức tạp, và mức tiêu thụ năng lượng. Việc lựa chọn giao thức phù hợp phụ thuộc vào yêu cầu cụ thể của từng ứng dụng.

Các giao thức bus phổ biến trong SoC như giao thức AXI, AHB, APB là giao thức thuộc chuẩn AMBA của hãng ARM phát triển. Các giao thức này được ứng dụng rộng rãi trong các hệ thống nhúng và SoC hiệu năng cao nhờ khả năng truyền dữ liệu tuyến tính, độ trễ thấp và hỗ trợ truy cập song song. Ví dụ như giao thức AXI thường được sử dụng trong các tác vụ yêu cầu băng thông cao như xử lý ảnh, video và trí tuệ nhân tạo, những lĩnh vực yêu cầu việc truyền dữ liệu nhanh chóng và đáng tin cậy là việc quan trọng. Trong khi đó, các giao thức UART, I<sup>2</sup>C lại phù hợp hơn với các kết nối ngoại vi có tốc độ thấp, chẳng hạn như cảm biến, bộ phát tín hiệu đơn giản hoặc các tín hiệu thiết bị đầu cuối.

Việc lựa chọn giao thức bus phù hợp là yếu tố then chốt trong thiết kế hệ thống SoC, giúp đảm bảo hiệu năng tổng thể, độ tin cậy, và khả năng mở rộng

của toàn bộ kiến trúc. Trong nhiều trường hợp, các giao thức này còn có thể được tùy biến để đáp ứng nhu cầu cụ thể của hệ thống mục tiêu.

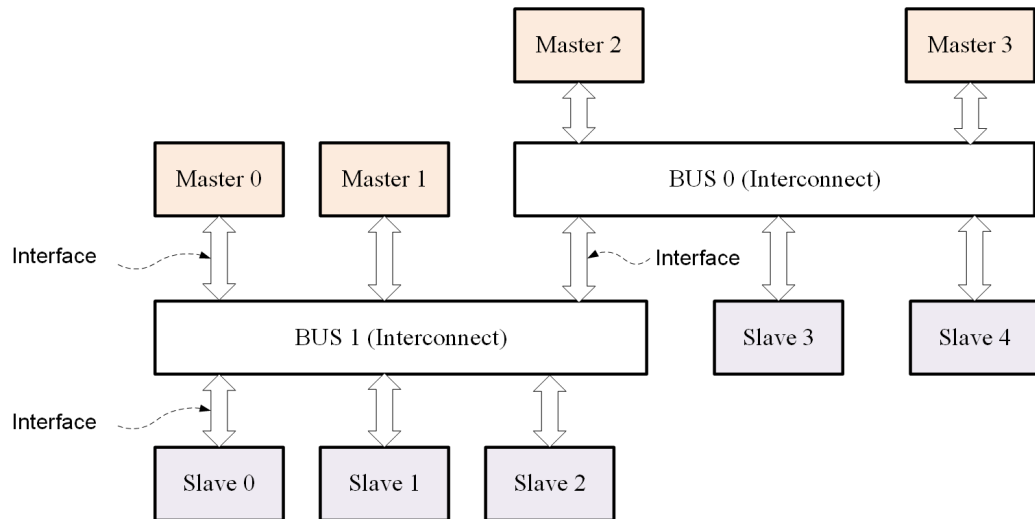
## **2.3 TỔNG QUAN VỀ GIAO THỨC AMBA AXI**

### **2.3.1 Giới thiệu chung về giao thức AMBA AXI**

Giao thức AXI là một trong những giao thức bus thuộc chuẩn AMBA, được phát triển bởi hãng ARM. Vào năm 2003, phiên bản đầu tiên của AXI ra đời, có tên gọi là AXI3, đánh dấu bước tiến quan trọng trong việc cải tiến giao tiếp nội bộ giữa các khối chức năng trong hệ thống trên chip. Trong giai đoạn 2010-2011, hãng ARM tiếp tục cải tiến AXI3 thành hai biến thể mới bao gồm AXI4 Lite và AXI4 Stream. Phiên bản AXI4 Lite là giao thức đơn giản hóa dành cho các giao dịch cấu hình có độ trễ thấp, còn AXI4 Stream thường được dùng cho việc truyền dữ liệu liên tục không địa chỉ, phù hợp với các ứng dụng xử lý tín hiệu số, hình ảnh và video. Trong giai đoạn 2013-2019, phiên bản mới nhất là AXI5 ra đời với những cải tiến về hiệu suất, hỗ trợ bảo mật và đồng bộ hóa tốt hơn, đáp ứng nhu cầu ngày càng cao trong các hệ thống SoC hiện đại.

Giao thức AXI được ứng dụng rộng rãi trong các vi điều khiển và hệ thống sử dụng kiến trúc ARM. Đây là giao thức mở, hỗ trợ ba cơ chế truyền dữ liệu, với nhiều đặc điểm nổi bật phù hợp các hệ thống yêu cầu tần số hoạt động cao, hiệu năng lớn, độ trễ thấp và băng thông cao. Do cùng thuộc chuẩn AMBA nên giao thức AXI có khả năng tương thích tốt với các giao thức bus khác như AHB và APB, giúp đơn giản hóa quá trình tích hợp hệ thống [8].

Giao thức AXI quy định về chuẩn giao tiếp giữa một Master, một Slave và một bus với nhau, hay một Master giao tiếp trực tiếp với một Slave. Trong đó, kết nối bus có thể bao gồm nhiều giao diện giao tiếp khác nhau. Với một giao diện giao tiếp giữa hai kết nối bus, vai trò Master và Slave chính là hai kết nối bus này. Hình 2.3 thể hiện một giao tiếp giữa BUS 0 và BUS 1, nếu BUS 0 khởi tạo giao dịch và gửi yêu cầu thì sẽ đảm nhận vào trò Master, trong khi BUS 1 đóng vai trò Slave hoặc ngược lại.



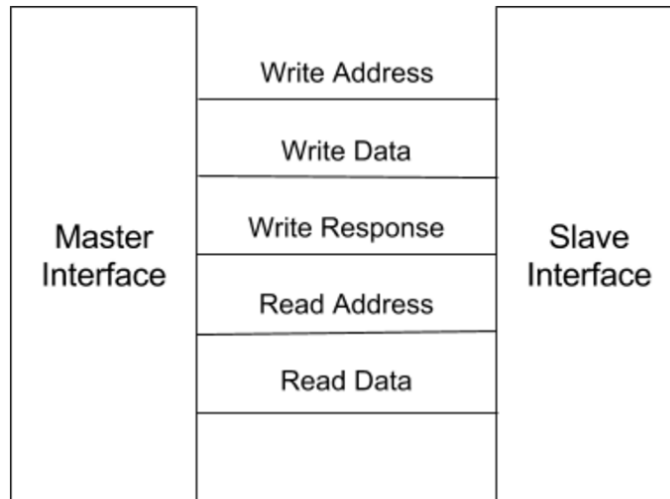
*Hình 2.3. Giao diện giao tiếp và kết nối bus*

### 2.3.2 Cấu trúc bus AXI

Giao thức quy định cơ chế, cách thức hoạt động cho quá trình đọc và ghi dữ liệu. Hai hoạt động này được gọi chung là một truy cập. Mỗi lần truy cập được hiểu là một giao dịch.

AXI hoạt động dựa trên năm kênh độc lập, hình 2.4 mô tả hai khối Master và Slave được kết nối với nhau thông qua năm kênh này. Mỗi kênh có đảm nhiệm một chức năng riêng. Mặc dù hoạt động độc lập, các kênh này vẫn phối hợp chặt chẽ để đảm bảo tính toàn vẹn và hiệu quả của giao dịch. Năm kênh này bao gồm:

- **Kênh địa chỉ ghi (Write Address Channel):** truyền thông tin địa chỉ và thông tin điều khiển liên quan đến quá trình ghi từ Master đến Slave.
- **Kênh dữ liệu ghi (Write Data Channel):** truyền dữ liệu ghi từ Master đến Slave.
- **Kênh phản hồi ghi (Write Response Channel):** truyền thông tin phản hồi cho một giao dịch ghi từ Slave đến Master sau khi hoàn thành giao dịch ghi.
- **Kênh địa chỉ đọc (Read Address Channel):** truyền thông tin địa chỉ và thông tin điều khiển của một giao dịch đọc từ Master đến Slave.
- **Kênh dữ liệu đọc (Read Data Channel):** truyền dữ liệu đọc và thông tin phản hồi của một giao dịch đọc từ Slave đến Master.



**Hình 2.4.** Các kênh giao tiếp giữa Master và Slave

Các tín hiệu trong mỗi kênh được giao thức AXI được quy định rõ ràng và được trình bày chi tiết trong các bảng từ bảng 2.1 đến bảng 2.5. Trong đó các tín hiệu ở kênh địa chỉ ghi được thể hiện ở bảng 2.1, các tín hiệu ở kênh dữ liệu ghi được thể hiện ở bảng 2.2, các tín hiệu ở kênh phản hồi ghi được thể hiện ở bảng 2.3, các tín hiệu ở kênh địa chỉ đọc được thể hiện ở bảng 2.4 và các tín hiệu ở kênh dữ liệu đọc được thể hiện ở bảng 2.5.

Bảng 2.1 bên dưới liệt kê các tín hiệu thuộc kênh địa chỉ ghi, được sử dụng bởi Master để gửi thông tin địa chỉ đến Slave, đồng thời cung cấp các thuộc tính bổ sung như độ dài của dữ liệu truyền, kích thước dữ liệu, loại truy cập, thuộc tính bảo mật, chất lượng dịch vụ và vùng định tuyến. Hầu hết các tín hiệu trong kênh này đều được xuất phát từ Master, ngoại trừ tín hiệu AWREADY được xuất phát từ Slave dùng để thông báo tình trạng sẵn sàng nhận thông tin địa chỉ được gửi từ Master.

**Bảng 2.1.** Các tín hiệu của kênh địa chỉ ghi

| TÍN HIỆU | NGUỒN  |
|----------|--------|
| AWID     | Master |
| AWADDR   | Master |
| AWLEN    | Master |
| AWSIZE   | Master |
| AWBURST  | Master |

|          |        |
|----------|--------|
| AWLOCK   | Master |
| AWCACHE  | Master |
| AWPROT   | Master |
| AWQOS    | Master |
| AWREGION | Master |
| AWUSER   | Master |
| AWVALID  | Master |
| AWREADY  | Slave  |

Bảng 2.2 bên dưới mô tả các tín hiệu thuộc kênh dữ liệu ghi, trong đó Master truyền dữ liệu đến Slave. Các tín hiệu như WDATA,WSTRB, WLAST, WVALID đóng vai trò xác định nội dung, độ dài, dữ liệu cuối cùng của một giao dịch ghi và tín hiệu thông báo Master đã sẵn sàng truyền dữ liệu. Riêng đối với tín hiệu WREADY xuất phát từ SLAVE nhằm thông báo trạng thái sẵn sàng nhận dữ liệu ghi.

**Bảng 2.2.** Các tín hiệu của kênh dữ liệu ghi

| TÍN HIỆU | NGUỒN  |
|----------|--------|
| WID      | Master |
| WDATA    | Master |
| WSTRB    | Master |
| WLAST    | Master |
| WUSER    | Master |
| WVALID   | Master |
| WREADY   | Slave  |

Bảng 2.3 trình bày các tín hiệu thuộc kênh phản hồi ghi trong giao thức AXI4. Đây là kênh phản hồi từ Slave về Master sau khi một giao dịch ghi được thực hiện, nhằm thông báo dữ liệu ghi thành công hay xuất hiện lỗi. Các tín hiệu như BID, BRESP, BUSER và BVALID được xuất phát từ Slave, trong khi tín hiệu BREADY do Master điều khiển để thông báo tình trạng sẵn sàng nhận tín hiệu từ kênh phản hồi ghi.

**Bảng 2.3.** Các tín hiệu của kênh phản hồi ghi

| TÍN HIỆU | NGUỒN  |
|----------|--------|
| BID      | Slave  |
| BRESP    | Slave  |
| BUSER    | Slave  |
| BVALID   | Slave  |
| BREADY   | Master |

Bảng 2.4 mô tả các tín hiệu của kênh địa chỉ đọc. Đây là kênh chịu trách nhiệm truyền thông tin địa chỉ địa chỉ đọc cho các yêu cầu đọc từ Master đến Slave. Tất cả các tín hiệu trong kênh này đều được điều khiển bởi Master, phản ánh vai trò chủ động của Master trong việc khởi tạo các giao dịch đọc.

**Bảng 2.4.** Các tín hiệu của kênh địa chỉ đọc

| TÍN HIỆU | NGUỒN  |
|----------|--------|
| ARID     | Master |
| ARADDR   | Master |
| ARLEN    | Master |
| ARLEN    | Master |
| ARSIZE   | Master |
| ARBURST  | Master |
| ARLOCK   | Master |
| ARCACHE  | Master |
| ARPROT   | Master |
| ARQOS    | Master |
| ARREGION | Master |
| ARUSER   | Master |
| ARVALID  | Master |
| ARREADY  | Master |

Bảng 2.5 liệt kê các tín hiệu thuộc kênh dữ liệu đọc trong giao thức AXI4. Đây là kênh chịu trách nhiệm truyền dữ liệu từ Slave về Master sau khi yêu cầu đọc được thiết lập qua kênh địa chỉ đọc. Các tín hiệu như RID, RDATA, RRESP,



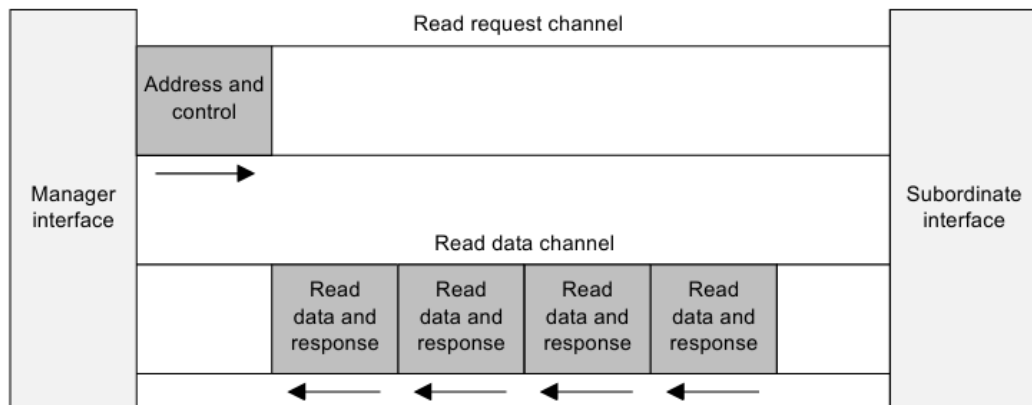
RLAST, RUSER và RVALID đều được điều khiển bởi Slave. Trong khi đó, tín hiệu RREADY xuất phát từ Master để thông báo đã sẵn sàng nhận dữ liệu.

**Bảng 2.5.** Các tín hiệu của kênh dữ liệu đọc

| TÍN HIỆU | NGUỒN  |
|----------|--------|
| RID      | Slave  |
| RDATA    | Slave  |
| RRESP    | Slave  |
| RLAST    | Slave  |
| RUSER    | Slave  |
| RVALID   | Slave  |
| RREADY   | Master |

### 2.3.3 Các quy định về hoạt động

Trong giao thức AXI, mỗi truy cập ghi hoặc đọc đều được xem là một giao dịch. Đối với một giao dịch đọc, quá trình truyền dữ liệu sẽ diễn ra thông qua hai kênh: kênh địa chỉ đọc và kênh dữ liệu đọc. Hình 2.5 minh họa quá trình hoạt động của một giao dịch đọc trong giao thức AXI.



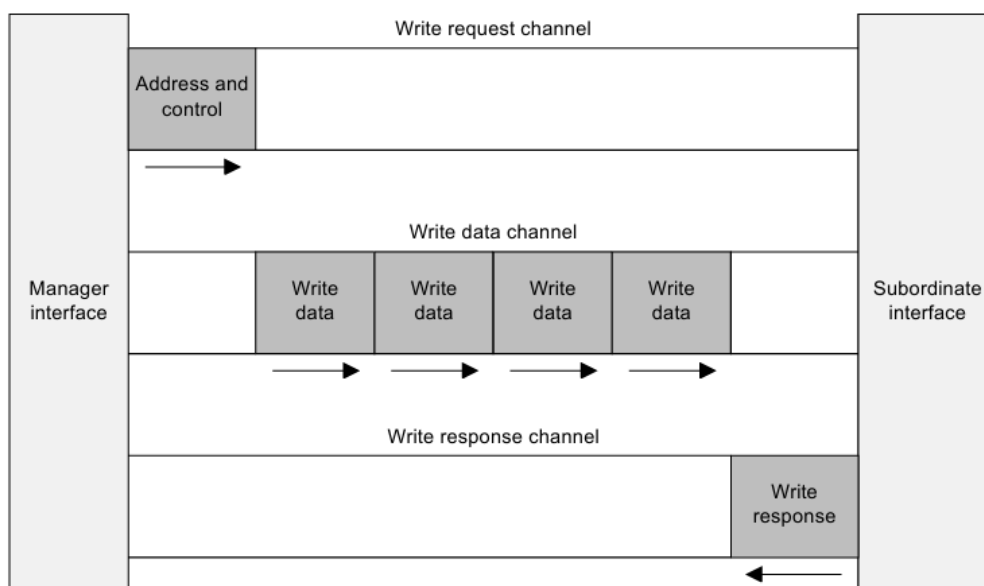
**Hình 2.5.** Hoạt động của giao dịch đọc

Một giao dịch đọc gồm hai bước chính:

- Master gửi địa chỉ cần đọc cùng với các thông tin điều khiển lên kênh địa chỉ đọc để khởi động quá trình đọc.

- Dữ liệu cùng với thông tin phản hồi sẽ được Slave truyền ngược lại cho Master thông qua kênh dữ liệu đọc. Số lượng dữ liệu được truyền sẽ phụ thuộc vào các thông tin điều khiển đã được gửi.

Một giao dịch ghi sẽ được thực hiện thông qua ba kênh: kênh địa chỉ ghi, kênh dữ liệu ghi và kênh phản hồi. Hình 2.6 minh họa trình tự hoạt động của một giao dịch ghi.



**Hình 2.6.** Hoạt động của giao dịch ghi

Một giao dịch ghi gồm ba bước:

- Master gửi địa chỉ ghi và thông tin điều khiển lên kênh địa chỉ ghi để khởi động quá trình ghi.
- Dữ liệu cần ghi được truyền từ Master đến Slave thông qua kênh dữ liệu ghi. Số lượng dữ liệu được xác định dựa trên thông tin điều khiển đã được gửi ở bước đầu.
- Sau khi hoàn tất dữ liệu ghi, Slave gửi tín hiệu phản hồi về cho Master thông qua kênh phản hồi ghi để xác nhận quá trình ghi đã hoàn tất.

## 2.3.4 Giao thức AMBA AXI4

### 2.3.4.1 Giới thiệu về giao thức AMBA AXI4

Giao thức AXI là một phần trong hệ thống chuẩn AMBA do hãng ARM phát triển từ những năm 1990, nhằm chuẩn hóa giao tiếp giữa các khối phần cứng

trong vi mạch. Phiên bản AXI4 được giới thiệu vào năm 2010, như một cải tiến lớn so với AXI3 và các chuẩn trước đó.

Các đặc trưng nổi bật của giao thức AXI4 bao gồm:

- Tách riêng kênh đọc và kênh ghi, pha truyền địa chỉ và điều khiển với pha truyền dữ liệu.
- Hỗ trợ trao đổi dữ liệu không thẳng hàng và sử dụng tín hiệu write strobe để xác định byte nào trong dữ liệu là hợp lệ, từ đó nâng cao độ linh hoạt khi truy cập bộ nhớ.
- Cho phép chèn thêm các tầng thanh ghi giúp định thời của bus và hệ thống tốt hơn.
- Các giao dịch có thể hoàn thành không theo thứ tự.
- Hỗ trợ phát hành nhiều địa chỉ truy cập đồng thời.
- Sử dụng cơ chế truyền theo burst, chỉ cần phát địa chỉ bắt đầu cho toàn bộ chuỗi truyền, giúp giảm tải overhead cho hệ thống [9].

#### 2.3.4.2 Cơ chế bắt tay của AXI4

Giao thức AXI hoạt động dựa trên cơ chế bắt tay hai chiều (two-way handshake), trong đó quá trình truyền dữ liệu được điều phối thông qua cặp tín hiệu VALID và READY trên từng kênh truyền riêng biệt.

Nguyên tắc vận hành của cơ chế này là bên phát sẽ khởi tạo tín hiệu VALID ở mức tích cực khi có dữ liệu hợp lệ cần truyền. Tín hiệu này được duy trì cho đến khi bên nhận phát ra tín hiệu READY ở mức tích cực, cho biết sẵn sàng tiếp nhận dữ liệu. Quá trình truyền dữ liệu chỉ được thực hiện tại thời điểm cả hai tín hiệu VALID và READY cùng đạt mức tích cực trên cạnh lên của xung nhịp.

Tín hiệu READY từ phía nhận có thể xuất hiện trước, đồng thời hoặc sau tín hiệu VALID từ phía truyền. Điều này dẫn đến ba trường hợp khác nhau trong cơ chế bắt tay, thể hiện sự linh hoạt của giao thức trong việc đồng bộ hóa truyền nhận dữ liệu.

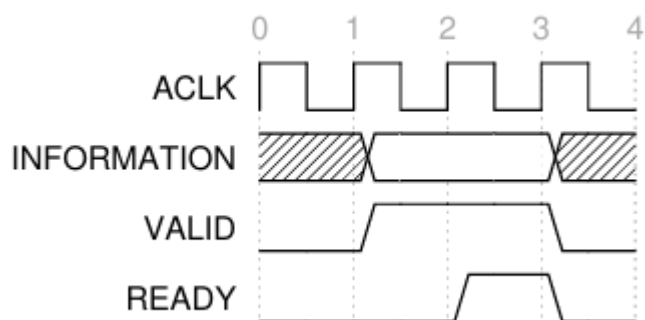
Mỗi kênh chức năng trong AXI4 bao gồm các kênh địa chỉ ghi, dữ liệu ghi, phản hồi ghi, địa chỉ đọc và dữ liệu đọc đều được trang bị cặp tín hiệu VALID,

READY độc lập, bảo đảm khả năng truyền song song và hiệu quả cao trong hệ thống. Các cặp tín hiệu bắt tay của các kênh được trình bày ở bảng 2.6.

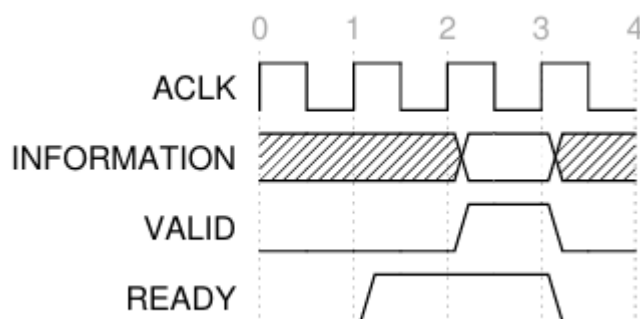
**Bảng 2.6.** Các cặp tín hiệu bắt tay trong giao thức AXI4

| KÊNH              | CẶP TÍN HIỆU BẮT TAY |
|-------------------|----------------------|
| Kênh địa chỉ ghi  | AWVALID – AWREADY    |
| Kênh dữ liệu ghi  | WVALID – WREADY      |
| Kênh phản hồi ghi | BVALID – BREADY      |
| Kênh địa chỉ đọc  | ARVALID – ARREADY    |
| Kênh dữ liệu đọc  | RVALID – RREADY      |

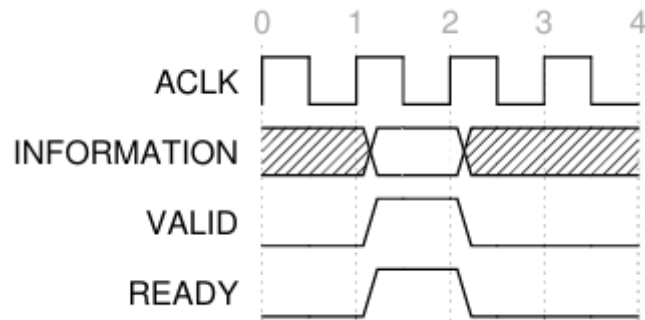
Trong quá trình truyền dữ liệu, một số tình huống phổ biến có thể xảy ra trong quá trình bắt tay: tín hiệu VALID tích cực trước tín hiệu READY được minh họa ở hình 2.7, tín hiệu VALID tích cực sau tín hiệu READY được thể hiện ở hình 2.8, và trường hợp cuối cùng là tín hiệu VALID tích cực cùng lúc với tín hiệu READY được thể hiện ở hình 2.9.



**Hình 2.7.** Tín hiệu VALID tích cực trước tín hiệu READY



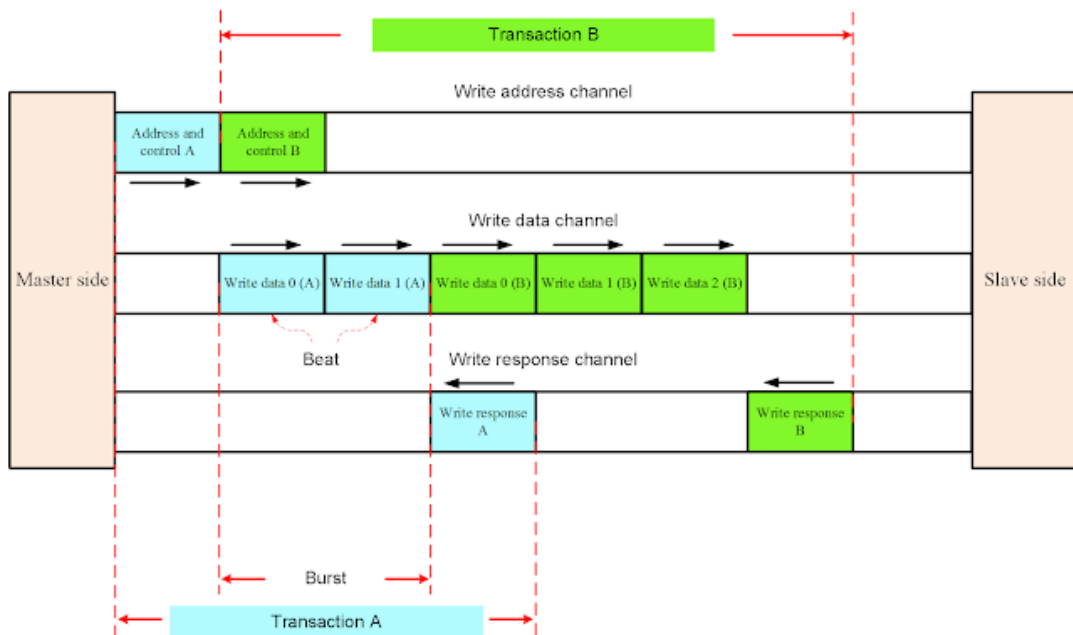
**Hình 2.8.** Tín hiệu VALID tích cực sau tín hiệu READY



**Hình 2.9.** Tín hiệu *VALID* tích cực cùng lúc với tín hiệu *READY*

### 2.3.4.3 Cơ chế burst của AXI4

Giao thức AXI4 hoạt động dựa trên cơ chế truyền dữ liệu theo kiểu burst. Cơ chế burst theo AXI4 được phân thành ba loại chính bao gồm: FIXED, INCR, và WRAP. Mỗi burst bao gồm một chuỗi các lần truyền dữ liệu (transfer) thuộc cùng một giao dịch, trong đó mỗi lần truyền dữ liệu được gọi là một beat. Một burst có thể chứa một hoặc nhiều beat, và thông tin điều khiển sẽ bao gồm các thuộc tính mô tả đặc điểm của burst trong mỗi giao dịch. Hình 2.10 minh họa cụ thể hoạt động của cơ chế burst.



**Hình 2.10.** Minh họa cơ chế burst

Độ dài một burst được quy định bởi tín hiệu *AxLEN* và được tính như sau:

$$\text{Độ dài burst (số beat)} = \text{AxLEN} + 1 \quad (2.1)$$

Số byte tối đa trong một lần truyền dữ liệu (hay còn gọi là một beat) được xác định bởi AxSIZE. Giá trị này quy định kích thước của mỗi beat và không được vượt quá độ rộng của bus dữ liệu (tức là độ rộng của các tín hiệu RDATA hoặc WDATA). Chẳng hạn, nếu bus dữ liệu có độ rộng 32 byte, thì giá trị của AxSIZE[2:0] phải nhỏ hơn hoặc bằng 5. Giá trị của AxSIZE được quy định ở bảng 2.7 dưới đây.

**Bảng 2.7.** Kích thước dữ liệu tương ứng với giá trị AxLEN

| AxSIZE | Label | Meaning                |
|--------|-------|------------------------|
| 0b000  | 1     | 1 byte per transfer    |
| 0b001  | 2     | 2 bytes per transfer   |
| 0b010  | 4     | 4 bytes per transfer   |
| 0b011  | 8     | 8 bytes per transfer   |
| 0b100  | 16    | 16 bytes per transfer  |
| 0b101  | 32    | 32 bytes per transfer  |
| 0b110  | 64    | 64 bytes per transfer  |
| 0b111  | 128   | 128 bytes per transfer |

Loại burst trong giao thức AXI4 được xác định thông qua tín hiệu AxBURST. Giao thức hỗ trợ ba loại burst chính:

- **FIXED:** Địa chỉ của mỗi lần truyền dữ liệu trong một giao dịch đều giống nhau, tức là không thay đổi theo từng beat.
- **INCR:** Địa chỉ của mỗi lần truyền dữ liệu kế tiếp được tính bằng cách cộng thêm một giá trị cố định vào địa chỉ trước đó. Giá trị cộng thêm này được xác định bởi tín hiệu AxSIZE [9]. Công thức tính địa chỉ truy cập như sau:

$$\text{Aligned\_Addr} = \text{INT}(\text{Start\_Addr} / \text{Size}) * \text{Size} \quad (2.2)$$

$$\text{Address\_N} = \text{Start\_Address} + (\text{N} - 1) * \text{Size} \quad (2.3)$$

Với:

Start\_Addr (= axaddr): Địa chỉ bắt đầu ghi dữ liệu

INT(x): Giá làm tròn xuống của x

N: Số lần truyền dữ liệu

Size: Kích thước mỗi transfer =  $2^{\text{Sizes}}$

- **WRAP:** Ban đầu địa chỉ của mỗi lần truyền dữ liệu sẽ được tính như kiểu INCR. Khi địa chỉ truy cập đạt đến địa chỉ cuộn, địa chỉ sẽ được trở về địa chỉ cuộn (wrap boundary) và tiếp tục truyền. Công thức tính địa chỉ truy cập như sau:

$$\text{Wrap\_Boundary} = \text{INT}(\text{Start\_Addr} / (\text{Size} * \text{Length})) * \text{Size} * \text{Length} \quad (2.4)$$

Nếu địa chỉ chưa vượt vùng wrap:

$$\text{Address\_N} = \text{Aligned\_Addr} + (N - 1) * \text{Size} \quad (2.5)$$

Nếu địa chỉ vượt biên và cần wrap lại:

$$\text{Address\_N} = \text{Start\_Addr} + ((N - 1) * \text{Size}) - (\text{Size} * \text{Length}) \quad (2.6)$$

Với:

Length: số lượng transfer =  $\text{AxLEN} + 1$

Bảng 2.8 dưới đây thể hiện giá trị của AxBURST tương ứng với cơ chế burst.

**Bảng 2.8.** Các loại burst tương ứng với giá trị AxBURST

| AxBURST | Label    | Meaning            |
|---------|----------|--------------------|
| 0b00    | FIXED    | Fixed burst        |
| 0b01    | INCR     | Incrementing burst |
| 0b10    | WRAP     | Wrapping burst     |
| 0b11    | RESERVED | -                  |

## 2.4 TỔNG QUAN VỀ NGÔN NGỮ MÔ TẢ CỨNG VERILOG VÀ PHẦN MỀM MÔ PHỎNG

### 2.4.1 Ngôn ngữ mô tả phần cứng Verilog

Verilog là một trong những ngôn ngữ mô tả phần cứng phổ biến nhất hiện nay, được sử dụng rộng rãi trong thiết kế, mô phỏng và triển khai các hệ thống

điện tử số. Verilog ra đời từ những năm 1980 và nhanh chóng trở thành một chuẩn công nghiệp trong lĩnh vực thiết kế vi mạch, đặc biệt trong các ứng dụng liên quan đến FPGA và ASIC [10].

Khác với các ngôn ngữ lập trình truyền thống như C hay Python, Verilog không được dùng để lập trình phần mềm mà dùng để mô tả hành vi và cấu trúc của phần cứng để thực hiện các tác vụ xảy ra song song và được điều khiển bởi tín hiệu xung nhịp. Một trong những điểm mạnh của Verilog là khả năng mô tả thiết kế ở nhiều mức độ trừu tượng, từ cấp độ cổng logic, cấp độ thanh ghi cho đến cấp độ hành vi. Điều này giúp cho người thiết kế có thể tiếp cận và kiểm soát hệ thống ở từng giai đoạn phát triển, từ khái niệm ban đầu cho đến khi tổng hợp và hiện thực hóa trên chip [11].

Verilog cung cấp cú pháp rõ ràng, gần giống với ngôn ngữ C, nên khá thân thiện với người học và kỹ sư có nền tảng lập trình. Ngôn ngữ này cho phép mô tả các phần tử logic như cổng AND, OR, bộ đếm, thanh ghi, bộ dồn kênh, và FSM, đồng thời hỗ trợ mô phỏng thời gian thực với các đơn vị thời gian linh hoạt. Với các cấu trúc như `always`, `initial`, `assign` cùng với các khối tuần tự và tổ hợp, Verilog cho phép xây dựng mô hình hoạt động của các mạch điện tử với độ chính xác cao. Các thiết kế có thể được chia thành các module riêng biệt, giúp tổ chức cấu trúc hệ thống một cách rõ ràng, dễ bảo trì và mở rộng.

Bên cạnh khả năng mô tả phần cứng, Verilog còn hỗ trợ các tính năng như khai báo các tín hiệu có nhiều trạng thái logic 0 (false), 1 (true), x (trạng thái không xác định), z (trạng thái trở kháng cao), định nghĩa thời gian trễ, mô phỏng tương tác với các thiết bị ngoại vi hoặc tệp dữ liệu, thứ rất cần thiết khi thiết kế các hệ thống phức tạp như vi xử lý, bộ điều khiển bus, hoặc bộ mã hóa, giải mã tín hiệu. Ngoài ra, ngôn ngữ này cũng hỗ trợ tạo các tệp kiểm thử để kiểm tra chức năng của các thiết kế bằng cách sinh tín hiệu đầu vào và giám sát đầu ra, giúp đảm bảo độ tin cậy trước khi triển khai thực tế.

Với sự phát triển không ngừng của hệ thống nhúng, SoC và trí tuệ nhân tạo trên phần cứng, nhu cầu mô tả và hiện thực hóa các kiến trúc phần cứng hiệu quả, linh hoạt và dễ kiểm thử ngày càng tăng cao. Ngôn ngữ Verilog nhờ vào tính phổ



biến và khả năng mở rộng, sự hỗ trợ mạnh mẽ từ các công cụ thiết kế mạnh mẽ đã tiếp tục giữ vai trò chủ chốt trong lĩnh vực thiết kế vi mạch số [10].

#### **2.4.2 Phần mềm XILINX VIVADO 2022.2**

Vivado Design Suite là bộ tổng hợp các phần mềm của hãng Xilinx, hỗ trợ quá trình thiết kế và kiểm tra trên FPGA. Phần mềm này được tạo ra bằng việc nâng cấp các thế hệ phần mềm thiết kế cũ ISE Design Suite. Vivado được dùng để phát triển các ứng dụng trên thế hệ chip và board Xilinx 7 series, ZYNQ-7000 All Programmable, UltraScale™ devices. Các thế hệ board cũ của Xilinx như Series 6 vẫn được hỗ trợ bởi ISE, Plan Ahead...

Vivado Design Suite cung cấp một môi trường để cấu hình, thực thi, kiểm tra và tích hợp các khối IP như một module độc lập trong nội dung của thiết kế cấp hệ thống. Các khối IP có thể bao gồm các logic thuần túy, bộ xử lý nhúng, các module xử lý tín hiệu số (DSP), hoặc các thuật toán DSP được thiết kế dựa trên ngôn ngữ C. Các khối IP do hãng Xilinx phát triển sử dụng chuẩn giao tiếp AXI4, cho phép tích hợp hệ thống nhanh chóng và hiệu quả hơn [12].

Phiên bản 2022.2 cung cấp hiệu năng cải thiện, dung lượng thiết kế lớn hơn và thời gian thực thi nhanh hơn, phù hợp cho các dự án từ đơn giản đến phức tạp trên nền tảng FPGA hiện đại.

## CHƯƠNG 3

# THIẾT KẾ HỆ THỐNG

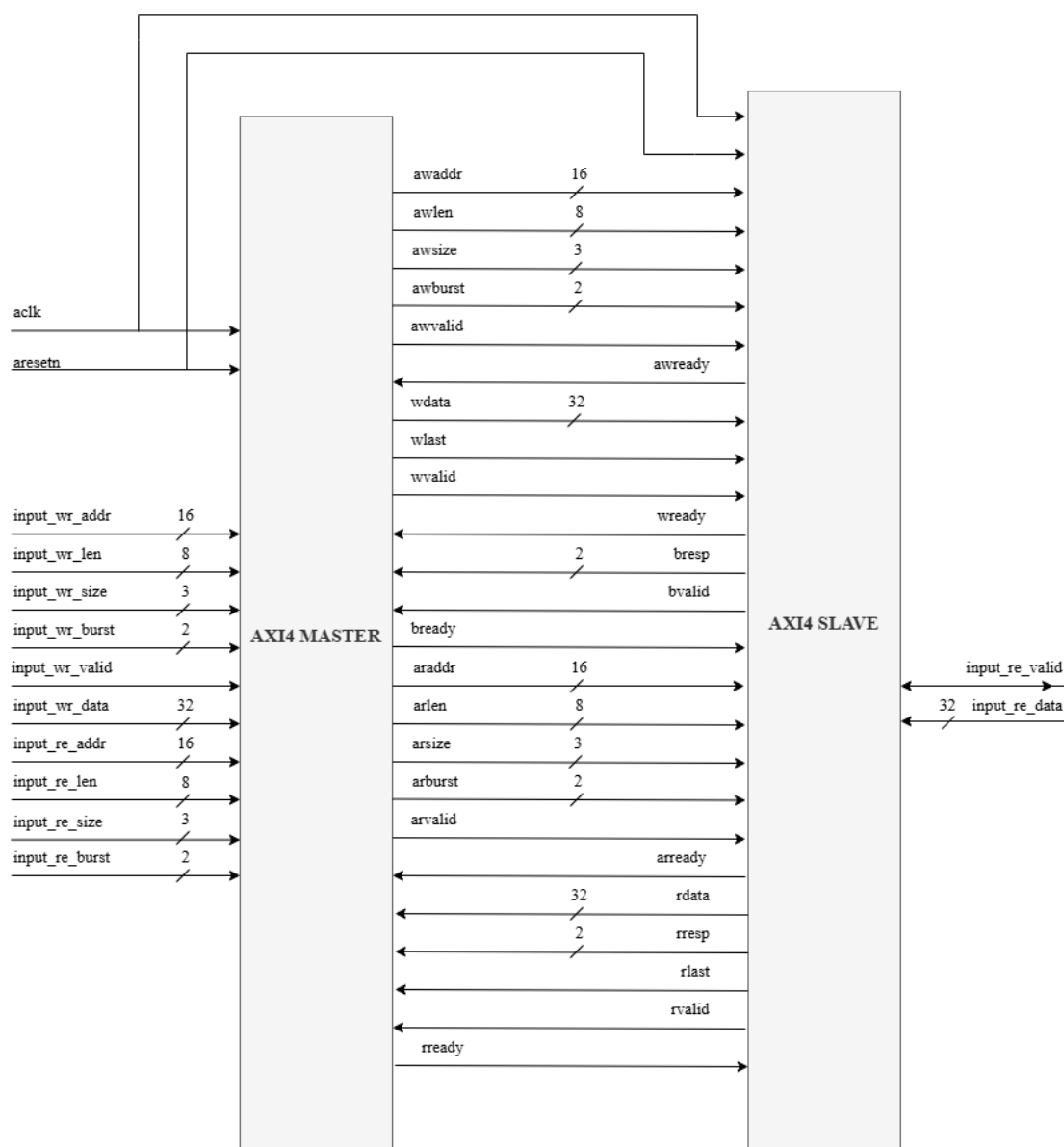
### 3.1 YÊU CẦU HỆ THỐNG

Với mục tiêu đã đề ra, mô hình điểm – điểm với một Master và một Slave phải đạt được các yêu cầu và thực hiện được các chức năng sau:

- Xây dựng mô hình điểm – điểm với một Master và một Slave sử dụng ngôn ngữ mô tả phần cứng Verilog.
- Mô hình hoạt động đúng theo giao thức AXI4 quy định với tần số hoạt động là 100MHz.
- Xây dựng các testcase để kiểm tra hoạt động của quá trình đọc và quá trình ghi theo ba cơ chế burst: FIXED, INCR, WRAP với nhiều trường hợp độ dài truyền và kích thước dữ liệu.

### 3.2 THIẾT KẾ HỆ THỐNG SỬ DỤNG GIAO THỨC AMBA AXI4

Hình 3.1 dưới mô tả hệ thống kết nối điểm – điểm sử dụng kiến trúc bus AMBA AXI4 bao gồm hai thành phần chính: một khối AXI4 MASTER và một khối AXI4 SLAVE. Hai khối AXI4 MASTER và AXI4 SLAVE được kết nối với nhau qua nhiều tín hiệu thuộc năm kênh truyền dữ liệu: kênh địa chỉ ghi, kênh dữ liệu ghi, kênh phản hồi ghi, kênh địa chỉ đọc, kênh dữ liệu đọc. Các tín hiệu này phụ thuộc vào chuẩn giao thức AXI4 do hãng ARM phát triển mà sẽ là ngõ vào hoặc ngõ ra của hai khối. Ngoài ra, hai tín hiệu aclk và aresetn là tín hiệu xung nhịp và tín hiệu xung khởi tạo lại hệ thống được dùng chung cho cả hai khối để đồng bộ hoạt động với nhau. Hệ thống sử dụng xung nhịp cạnh lên để điều khiển hoạt động và xung khởi tạo cạnh xuống để hệ thống trở về trạng thái ban đầu.



**Hình 3.1.** Mô hình kết nối điểm – điểm của AXI4 bus

Khối AXI4 MASTER đóng vai trò như một bộ nhớ AXI RAM, thực hiện việc gửi yêu cầu đọc, ghi dữ liệu, đồng thời lưu trữ dữ liệu đọc được từ khối AXI4 SLAVE. Bên cạnh đó, AXI4 MASTER cũng tiếp nhận và xử lý các tín hiệu phản hồi trong quá trình truyền nhận dữ liệu với khối AXI4 SLAVE. Khối AXI4 SLAVE cũng được thiết kế như một bộ nhớ AXI RAM, có chức năng lưu trữ dữ liệu được ghi từ AXI4 MASTER gửi đến, tiếp nhận và phản hồi lại các tín hiệu trong quá trình kết nối truyền nhận dữ liệu với khối AXI4 MASTER.

Mô tả các chân tín hiệu sử dụng trong kết nối của AXI4 bus được mô tả lần lượt trong bảng 3.1 và bảng 3.2 dưới đây.

**Bảng 3.1. Các chân tín hiệu được sử dụng trong AXI4 MASTER**

| TÍN HIỆU                                       | SỐ BIT | PHÂN LOẠI | MÔ TẢ                                                      |
|------------------------------------------------|--------|-----------|------------------------------------------------------------|
| <b>Các tín hiệu xung điều khiển</b>            |        |           |                                                            |
| aclk                                           | 1      | Ngõ vào   | Tín hiệu xung nhịp đồng bộ hệ thống                        |
| aresetn                                        | 1      | Ngõ vào   | Tín hiệu reset tích cực mức thấp để khởi tạo lại hệ thống  |
| <b>Các tín hiệu dùng để truyền dữ liệu vào</b> |        |           |                                                            |
| input_wr_addr                                  | 16     | Ngõ vào   | Khởi tạo địa chỉ bắt đầu transfer trong một giao dịch ghi  |
| input_wr_len                                   | 8      | Ngõ vào   | Khởi tạo số transfer của burst ghi                         |
| input_wr_size                                  | 3      | Ngõ vào   | Khởi tạo kích thước chung của một transfer trong burst ghi |
| input_wr_burst                                 | 2      | Ngõ vào   | Khởi tạo loại burst ghi                                    |
| input_wr_valid                                 | 1      | Ngõ vào   | Khởi tạo tín hiệu cho phép dữ liệu ghi hợp lệ              |
| input_wr_data                                  | 32     | Ngõ vào   | Khởi tạo dữ liệu ghi                                       |
| input_re_addr                                  | 16     | Ngõ vào   | Khởi tạo địa chỉ bắt đầu transfer trong một giao dịch đọc  |
| input_re_len                                   | 8      | Ngõ vào   | Khởi tạo số transfer trong một burst đọc                   |
| input_re_size                                  | 3      | Ngõ vào   | Khởi tạo kích thước chung của một transfer của burst đọc   |
| input_re_burst                                 | 2      | Ngõ vào   | Khởi tạo loại dữ liệu burst đọc                            |
| <b>Kênh địa chỉ ghi</b>                        |        |           |                                                            |
| awaddr                                         | 16     | Ngõ ra    | Địa chỉ bắt đầu transfer của giao dịch ghi                 |
| awlen                                          | 8      | Ngõ ra    | Số transfer của burst ghi                                  |
| awsiz                                          | 3      | Ngõ ra    | Kích thước chung của một transfer trong một burst ghi      |
| awburst                                        | 2      | Ngõ ra    | Loại burst ghi                                             |
| awvalid                                        | 1      | Ngõ ra    | Tín hiệu thông báo rằng địa chỉ ghi hợp lệ                 |
| awready                                        | 1      | Ngõ vào   | Cho biết Slave đã sẵn sàng nhận địa chỉ                    |
| <b>Kênh dữ liệu ghi</b>                        |        |           |                                                            |
| wdata                                          | 32     | Ngõ ra    | Dữ liệu ghi                                                |
| wlast                                          | 1      | Ngõ ra    | Tín hiệu thông báo transfer cuối cùng trong burst ghi      |
| wvalid                                         | 1      | Ngõ ra    | Tín hiệu thông báo thông tin hợp lệ                        |
| wready                                         | 1      | Ngõ vào   | Tín hiệu Slave sẵn sàng nhận dữ liệu ghi                   |

| <b>Kênh phản hồi ghi</b> |    |         |                                                               |
|--------------------------|----|---------|---------------------------------------------------------------|
| bresp                    | 2  | Ngõ vào | Tín hiệu thông báo trạng thái giao dịch ghi                   |
| bvalid                   | 1  | Ngõ vào | Thông báo một phản hồi hợp lệ                                 |
| bready                   | 1  | Ngõ ra  | Cho biết Master sẵn sàng nhận phản hồi hợp lệ                 |
| <b>Kênh địa chỉ đọc</b>  |    |         |                                                               |
| araddr                   | 16 | Ngõ ra  | Địa chỉ bắt đầu transfer trong giao dịch đọc                  |
| arlen                    | 8  | Ngõ ra  | Số transfer của burst đọc                                     |
| arsize                   | 3  | Ngõ ra  | Kích thước chung của một transfer trong một burst đọc         |
| arburst                  | 2  | Ngõ ra  | Loại burst đọc                                                |
| arvalid                  | 1  | Ngõ ra  | Tín hiệu thông báo rằng địa chỉ đọc hợp lệ                    |
| arready                  | 1  | Ngõ vào | Cho biết Slave đã sẵn sàng nhận địa chỉ và tín hiệu liên quan |
| <b>Kênh dữ liệu đọc</b>  |    |         |                                                               |
| rdata                    | 32 | Ngõ vào | Dữ liệu đọc                                                   |
| rresp                    | 2  | Ngõ vào | Tín hiệu thông báo trạng thái của transfer đọc                |
| rlast                    | 1  | Ngõ vào | Tín hiệu thông báo transfer cuối cùng trong burst đọc         |
| rvalid                   | 1  | Ngõ vào | Tín hiệu thông báo dữ liệu đọc hợp lệ                         |
| rready                   | 1  | Ngõ ra  | Tín hiệu cho biết Master sẵn sàng nhận dữ liệu đọc            |

**Bảng 3.2.** Các chân tín hiệu được sử dụng trong AXI4 SLAVE

| <b>TÍN HIỆU</b>                                | <b>SỐ BIT</b> | <b>PHÂN LOẠI</b> | <b>MÔ TẢ</b>                                              |
|------------------------------------------------|---------------|------------------|-----------------------------------------------------------|
| <b>Các tín hiệu xung điều khiển</b>            |               |                  |                                                           |
| aclk                                           | 1             | Ngõ vào          | Tín hiệu xung nhịp đồng bộ hệ thống                       |
| aresetn                                        | 1             | Ngõ vào          | Tín hiệu reset tích cực mức thấp để khởi tạo lại hệ thống |
| <b>Các tín hiệu dùng để truyền dữ liệu vào</b> |               |                  |                                                           |
| input_re_valid                                 | 1             | Ngõ vào          | Cho phép dữ liệu đọc hợp lệ                               |
| input_re_data                                  | 32            | Ngõ vào          | Khởi tạo dữ liệu đọc                                      |
| <b>Kênh địa chỉ ghi</b>                        |               |                  |                                                           |
| awaddr                                         | 16            | Ngõ vào          | Địa chỉ bắt đầu transfer của giao dịch ghi                |

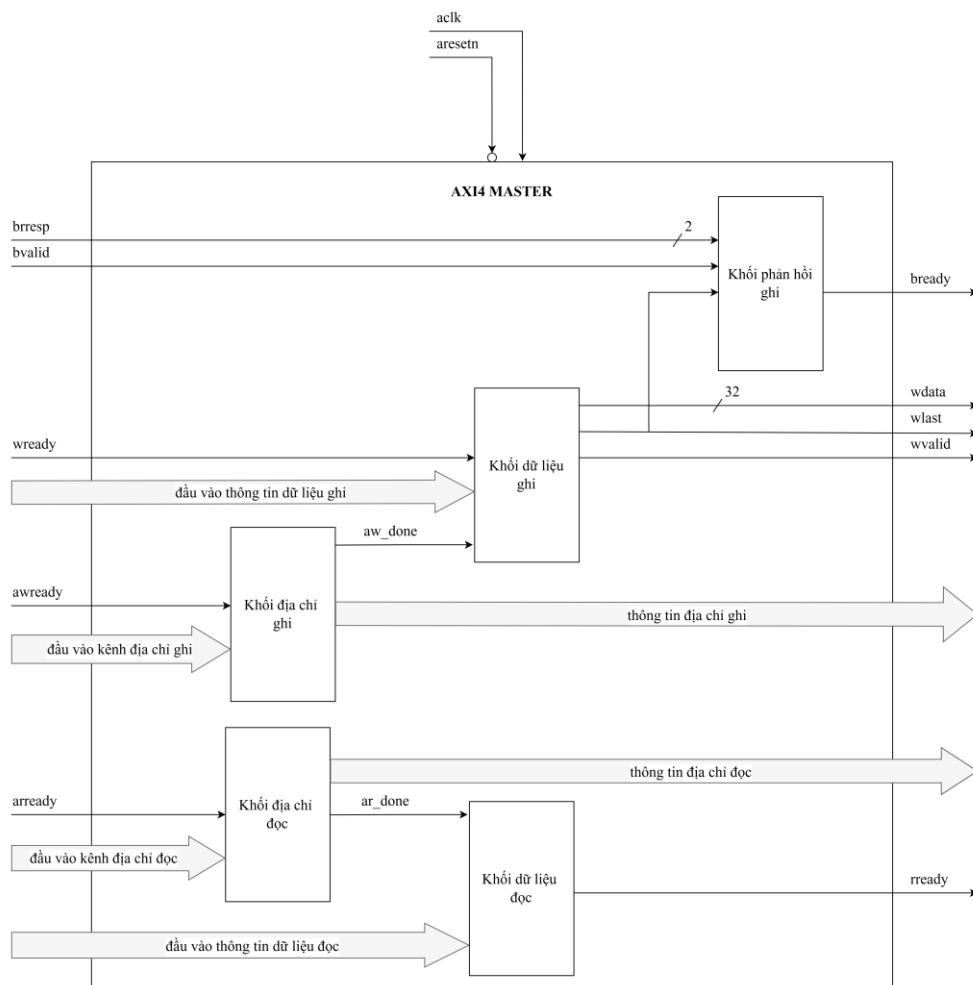
|                          |    |         |                                                               |
|--------------------------|----|---------|---------------------------------------------------------------|
| awlen                    | 8  | Ngõ vào | Số transfer của burst ghi                                     |
| awsize                   | 3  | Ngõ vào | Kích thước chung của một transfer trong một burst ghi         |
| awburst                  | 2  | Ngõ vào | Loại burst ghi                                                |
| awvalid                  | 1  | Ngõ vào | Tín hiệu thông báo rằng địa chỉ ghi hợp lệ                    |
| awready                  | 1  | Ngõ ra  | Cho biết Slave đã sẵn sàng nhận địa chỉ                       |
| <b>Kênh dữ liệu ghi</b>  |    |         |                                                               |
| wdata                    | 32 | Ngõ vào | Dữ liệu ghi                                                   |
| wlast                    | 1  | Ngõ vào | Tín hiệu thông báo transfer cuối cùng trong burst ghi         |
| wvalid                   | 1  | Ngõ vào | Tín hiệu thông báo thông tin hợp lệ                           |
| wready                   | 1  | Ngõ ra  | Tín hiệu cho biết Slave sẵn sàng nhận dữ liệu ghi             |
| <b>Kênh phản hồi ghi</b> |    |         |                                                               |
| bresp                    | 2  | Ngõ ra  | Tín hiệu thông báo trạng thái giao dịch ghi                   |
| bvalid                   | 1  | Ngõ ra  | Thông báo một phản hồi hợp lệ                                 |
| bready                   | 1  | Ngõ vào | Cho biết Master sẵn sàng nhận phản hồi hợp lệ                 |
| <b>Kênh địa chỉ đọc</b>  |    |         |                                                               |
| araddr                   | 16 | Ngõ vào | Địa chỉ bắt đầu transfer trong giao dịch đọc                  |
| arlen                    | 8  | Ngõ vào | Số transfer của burst đọc                                     |
| arsize                   | 3  | Ngõ vào | Kích thước chung của một transfer trong một burst đọc         |
| arburst                  | 2  | Ngõ vào | Loại burst đọc                                                |
| arvalid                  | 1  | Ngõ vào | Tín hiệu thông báo rằng địa chỉ đọc hợp lệ                    |
| arready                  | 1  | Ngõ ra  | Cho biết Slave đã sẵn sàng nhận địa chỉ và tín hiệu liên quan |
| <b>Kênh dữ liệu đọc</b>  |    |         |                                                               |
| rdata                    | 32 | Ngõ ra  | Dữ liệu đọc                                                   |
| rresp                    | 2  | Ngõ ra  | Tín hiệu thông báo trạng thái của transfer đọc                |
| rlast                    | 1  | Ngõ ra  | Tín hiệu thông báo transfer cuối cùng trong burst đọc         |
| rvalid                   | 1  | Ngõ ra  | Tín hiệu thông báo dữ liệu đọc hợp lệ                         |
| rready                   | 1  | Ngõ vào | Tín hiệu cho biết Master sẵn sàng nhận dữ liệu đọc            |

### 3.2.1 Thiết kế khối AXI4 MASTER

Khối AXI4 MASTER được thiết kế như bộ điều khiển truyền dữ liệu chủ động, có vai trò khởi tạo các giao dịch đọc và ghi đến AXI4 SLAVE. Trong quá trình ghi dữ liệu, AXI4 MASTER sẽ tạo các tín hiệu điều khiển và truyền dữ liệu đến khối AXI4 SLAVE, chịu trách nhiệm kiểm tra phản hồi từ AXI4 SLAVE để xác nhận quá trình ghi đã hoàn tất.

Ngược lại, trong quá trình đọc dữ liệu, AXI4 MASTER sẽ gửi yêu cầu đọc tới AXI4 SLAVE. Sau đó, AXI4 MASTER tiếp nhận dữ liệu từ AXI4 SLAVE và lưu vào ô nhớ nội bộ.

Như vậy, AXI4 MASTER đóng vai trò chủ động trong việc điều phối quá trình truyền dữ liệu từ AXI4 SLAVE. Sơ đồ khối AXI4 MASTER thể hiện ở hình 3.2.



**Hình 3.2.** Sơ đồ khối chi tiết của khối AXI4 MASTER

Khối AXI4 MASTER được thiết kế gồm 5 khối: Khối địa chỉ ghi, khối dữ liệu ghi, khối phản hồi ghi, khối địa chỉ đọc và khối dữ liệu đọc.

- Khối địa chỉ ghi: Dùng để nhận thông tin địa chỉ ghi, các tín hiệu điều khiển từ bên ngoài và gửi các dữ liệu đến khối dữ liệu ghi.
- Khối dữ liệu ghi: Nhận dữ liệu ghi, tín hiệu `wready` từ bên ngoài và tín hiệu `aw_done` từ khối địa chỉ ghi báo hiệu đã gửi xong thông tin địa chỉ ghi. Khi tín hiệu `aw_done` tích cực, khối sẽ tiến hành tích cực tín hiệu `wvalid`, xử lý dữ liệu ghi theo đúng cơ chế truyền, số beat và số byte truyền qua AXI4 SLAVE. Sau đó, truyền tín hiệu xác nhận `wlast` thể hiện beat cuối cùng được truyền qua khối phản hồi ghi.
- Khối phản hồi ghi: Nhận tín hiệu xác nhận dữ liệu truyền có hợp lệ hay không.
- Khối địa chỉ đọc: Dùng để nhận thông tin địa chỉ đọc, các tín hiệu điều khiển từ bên ngoài và gửi thông tin địa chỉ đọc đến khối dữ liệu đọc.
- Khối dữ liệu đọc: Nhận tín hiệu `ar_done` thông báo đã truyền xong thông tin địa chỉ đọc, tín hiệu điều khiển từ khối địa chỉ đọc và dữ liệu đọc từ AXI4 SLAVE.

### 3.2.2 Thiết kế khối AXI4 SLAVE

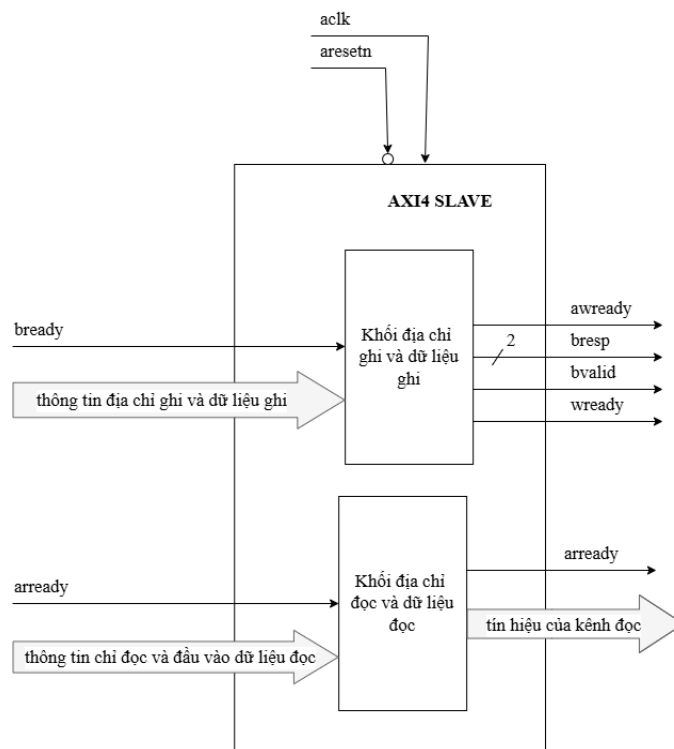
Khối AXI4 SLAVE được thiết kế như một bộ nhớ AXI RAM, đóng vai trò lưu trữ dữ liệu và phục vụ giao tiếp với khối AXI4 MASTER. Trong quá trình ghi dữ liệu, AXI4 MASTER sẽ gửi các tín hiệu điều khiển và dữ liệu cần ghi tới AXI4 SLAVE. Khi nhận được yêu cầu ghi hợp lệ, AXI4 SLAVE sẽ ghi dữ liệu và các ô nhớ nội bộ tương ứng với địa chỉ được chỉ định từ AXI4 MASTER.

Ngược lại, trong quá trình đọc dữ liệu, AXI4 MASTER gửi yêu cầu đọc đến AXI4 SLAVE. Sau khi tiếp nhận yêu cầu, AXI4 SLAVE sẽ truy xuất dữ liệu từ các ô nhớ tương ứng, sau đó gửi dữ liệu này trở lại AXI4 MASTER.

Như vậy, AXI4 SLAVE hoạt động như một vùng bộ nhớ có thể đọc và ghi được, cho phép lưu trữ dữ liệu do AXI4 MASTER gửi đến hoặc cung cấp lại dữ



liệu khi AXI4 MASTER yêu cầu. Sơ đồ khối của AXI4 SLAVE được thể hiện như hình 3.3.



**Hình 3.3.** Sơ đồ khối chi tiết của khối AXI4 SLAVE

Khối AXI4 SLAVE được thiết kế gồm 2 khối: Khối địa chỉ ghi và dữ liệu ghi, Khối địa chỉ đọc và dữ liệu đọc.

- Khối địa chỉ ghi và dữ liệu ghi: là khối tiếp nhận thông tin địa chỉ ghi và dữ liệu ghi từ AXI4 MASTER rồi lưu vào bộ nhớ nội bộ.
- Khối địa chỉ đọc và dữ liệu đọc: là khối nhận địa chỉ đọc từ AXI4 MASTER và truy xuất dữ liệu vào địa chỉ ô nhớ nội tương ứng rồi gửi dữ liệu cần đọc cho AXI4 MASTER.

## CHƯƠNG 4

# KẾT QUẢ MÔ PHỎNG

### 4.1 TÓM TẮT NỘI DUNG

Trong chương này, tôi tập trung vào việc kiểm tra hoạt động của mô hình điểm – điểm của chuẩn AXI4 đã được thiết kế ở chương trước. Các khối AXI4 MASTER và AXI4 SLAVE được mô phỏng và kiểm tra chức năng thông qua nhiều tình huống truyền dữ liệu khác nhau. Do đặc điểm hoạt động tuân theo chuẩn AXI4 với các cơ chế truyền như FIXED, INCR và WRAP, quá trình kiểm tra được thực hiện trên từng cơ chế để đảm bảo mô hình phản hồi đúng theo đặc tả giao thức. Sau quá trình mô phỏng, tôi tiến hành phân tích kết quả, đưa ra các đánh giá cụ thể về tính đúng đắn, khả năng mở rộng và hiệu năng của mô hình truyền dữ liệu qua chuẩn truyền AXI4.

### 4.2 MÔ PHỎNG HỆ THỐNG SỬ DỤNG PHẦN MỀM XILINX VIVADO 2022.2

Trong phần này, ta tiến hành mô phỏng các tệp kiểm tra của quá trình đọc và quá trình ghi với các thông số đầu vào cho từng trường hợp tệp kiểm tra khác nhau. Điểm chung duy nhất của toàn bộ tệp kiểm tra trong phần này là xung nhịp được thiết lập với chu kỳ là 10ns. Từ đó, tần số hoạt động của cả hệ thống được tính toán dưới đây:

$$f = \frac{1}{T} = \frac{1}{10 * 10^{-9}} = 100\text{MHz} \quad (2.7)$$

Bảng 4.1 dưới đây mô tả tất cả các tệp kiểm tra (testcase) của quá trình đọc và quá trình ghi với nhiều trường hợp dữ liệu, độ dài dữ liệu và số lượng beat dữ liệu truyền khác nhau đối với từng trường hợp cơ chế truyền khác nhau.

**Bảng 4.1.** Giới thiệu các testcase

| TESTCASE             | CẤU HÌNH                                                                                                                                                               | NỘI DUNG KIỂM TRA                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Quá trình ghi</b> |                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 1                    | <b>Trường hợp 1:</b><br>input_wr_addr = 32'd0000_1235;<br>input_wr_len = 2'b0;<br>input_wr_size = 3'b000;<br>input_wr_burst = 2'b00;<br>input_wr_data = 32'h12345678;  | <b>Mục tiêu:</b> Kiểm tra việc truyền dữ liệu theo cơ chế FIXED.<br><br><b>Thông số đầu vào:</b><br>- <i>Trường hợp 1:</i><br>+ 1 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 1 bytes cho mỗi lần truyền dữ liệu<br>- <i>Trường hợp 2:</i><br>+ 4 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 2 bytes cho mỗi lần truyền dữ liệu<br>- <i>Trường hợp 3:</i><br>+ 3 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 4 bytes cho mỗi lần truyền dữ liệu |
|                      | <b>Trường hợp 2:</b><br>input_wr_addr = 32'd0000_0016;<br>input_wr_len = 8'b3;<br>input_wr_size = 3'b001;<br>input_wr_burst = 2'b00;<br>input_wr_data = 32'h12345678;  |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|                      | <b>Trường hợp 3:</b><br>input_wr_addr = 32'd0000_0106;<br>input_wr_len = 8'b2;<br>input_wr_size = 3'b010;<br>input_wr_burst = 2'b00;<br>input_wr_data = 32'h12345678;  |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 2                    | <b>Trường hợp 1:</b><br>input_wr_addr = 32'd0000_5000;<br>input_wr_len = 2'b10;<br>input_wr_size = 3'b001;<br>input_wr_burst = 2'b01;<br>input_wr_data = 32'h22161043; | <b>Mục tiêu:</b> Kiểm tra việc truyền dữ liệu theo cơ chế INCR.<br><br><b>Thông số đầu vào:</b><br>- <i>Trường hợp 1:</i><br>+ 3 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 2 bytes cho mỗi lần truyền dữ liệu<br>- <i>Trường hợp 2:</i>                                                                                                                                                                                                          |
|                      | <b>Trường hợp 2:</b><br>input_wr_addr = 32'd0000_2578;<br>input_wr_len = 2'b10;                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

|                      |                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                           |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      | input_wr_size = 3'b010;<br>input_wr_burst = 2'b01;<br>input_wr_data = 32'h22161043;                                                                                           | + 3 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 4 bytes<br>cho mỗi lần truyền dữ liệu                                                                                                                                                                                                                                                    |
| 3                    | <b>Trường hợp 1:</b><br>input_wr_addr = 32'd0000_2578;<br>input_wr_len = 4'b1101;<br>input_wr_size = 3'b000;<br>input_wr_burst = 2'b10;<br>input_wr_data = 32'h45679123;      | <b>Mục tiêu:</b> Kiểm tra việc truyền dữ liệu theo cơ chế WRAP.<br><br><b>Thông số đầu vào:</b><br>- <i>Trường hợp 1:</i><br>+ 14 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 1 bytes<br>cho mỗi lần truyền dữ liệu<br>- <i>Trường hợp 2:</i><br>+ 11 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 4 bytes<br>cho mỗi lần truyền dữ liệu |
|                      | <b>Trường hợp 2:</b><br>input_wr_addr = 32'd0001_5000;<br>input_wr_len = 8'b0000_1010;<br>input_wr_size = 3'b010;<br>input_wr_burst = 2'b10;<br>input_wr_data = 32'h45679123; |                                                                                                                                                                                                                                                                                                                                           |
| 4                    | input_wr_addr = 15'd20000;<br>input_wr_len = 8'd0;<br>input_wr_size = 3'b001;<br>input_wr_burst = 2'b00;<br>input_wr_data = 32'h12345678;                                     | <b>Mục tiêu:</b> Kiểm tra trường hợp địa chỉ ghi vượt giới hạn địa chỉ ô nhớ nội bộ.                                                                                                                                                                                                                                                      |
| <b>Quá trình đọc</b> |                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                           |
| 1                    | <b>Trường hợp 1:</b><br>input_re_addr = 32'd0001_5000;<br>input_re_len = 8'b0000_0000;<br>input_re_size = 3'b000;<br>input_re_burst = 2'b00;<br>input_re_data = 32'h45679123; | <b>Mục tiêu:</b> Kiểm tra việc truyền dữ liệu theo cơ chế FIXED.<br><br><b>Thông số đầu vào:</b><br>- <i>Trường hợp 1:</i><br>+ 1 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 1 bytes<br>cho mỗi lần truyền dữ liệu<br>- <i>Trường hợp 2:</i>                                                                                            |
|                      | <b>Trường hợp 2:</b><br>input_re_addr = 32'd0001_5000;<br>input_re_len = 8'd3;                                                                                                |                                                                                                                                                                                                                                                                                                                                           |

|   |                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                           |
|---|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|   | input_re_size = 3'b001;<br>input_re_burst = 2'b00;<br>input_re_data = 32'h45679123;                                                                                      | + 4 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 2 bytes<br>cho mỗi lần truyền dữ liệu                                                                                                                                                                                                                                                    |
| 2 | <b>Trường hợp 1:</b><br>input_re_addr = 32'd0000_5000;<br>input_re_len = 2'b10;<br>input_re_size = 3'b001;<br>input_re_burst = 2'b01;<br>input_re_data = 32'h87654321;   | <b>Mục tiêu:</b> Kiểm tra việc truyền dữ liệu theo cơ chế INCR.<br><br><b>Thông số đầu vào:</b><br>- <i>Trường hợp 1:</i><br>+ 3 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 2 bytes<br>cho mỗi lần truyền dữ liệu<br>- <i>Trường hợp 2:</i><br>+ 3 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 4 bytes<br>cho mỗi lần truyền dữ liệu   |
|   | <b>Trường hợp 2:</b><br>input_re_addr = 32'd0000_2578;<br>input_re_len = 2'b10;<br>input_re_size = 3'b010;<br>input_re_burst = 2'b01;<br>input_re_data = 32'h87654321;   |                                                                                                                                                                                                                                                                                                                                           |
| 3 | <b>Trường hợp 1:</b><br>input_re_addr = 32'd0000_2578;<br>input_re_len = 4'b1101;<br>input_re_size = 3'b000;<br>input_re_burst = 2'b10;<br>input_re_data = 32'h56217895; | <b>Mục tiêu:</b> Kiểm tra việc truyền dữ liệu theo cơ chế WRAP.<br><br><b>Thông số đầu vào:</b><br>- <i>Trường hợp 1:</i><br>+ 14 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 1 bytes<br>cho mỗi lần truyền dữ liệu<br>- <i>Trường hợp 2:</i><br>+ 11 đơn vị truyền dữ liệu<br>+ Độ dài dữ liệu là 4 bytes<br>cho mỗi lần truyền dữ liệu |
|   | <b>Trường hợp 2:</b><br>input_re_addr = 32'd0001_5000;<br>input_re_len = 8'd1010;<br>input_re_size = 3'b010;<br>input_re_burst = 2'b10;<br>input_re_data = 32'h56217895; |                                                                                                                                                                                                                                                                                                                                           |
| 4 | input_re_addr = 15'd20000;<br>input_re_len = 8'd0;<br>input_re_size = 3'b001;                                                                                            | <b>Mục tiêu:</b> Kiểm tra trường hợp địa chỉ đọc vượt giới hạn địa chỉ ô nhớ nội bộ.                                                                                                                                                                                                                                                      |

|  |                                                          |  |
|--|----------------------------------------------------------|--|
|  | input_re_burst = 2'b00;<br>input_re_data = 32'h12345678; |  |
|--|----------------------------------------------------------|--|

#### 4.2.1 Hoạt động ghi dữ liệu

Xây dựng các tập kiểm tra của quá trình ghi với mục tiêu kiểm thử các trường hợp để mô phỏng quá trình truyền dữ liệu từ AXI4 MASTER qua AXI4 SLAVE thông qua giao thức AXI4 bao gồm các trường hợp:

- Các cơ chế truyền burst: FIXED, INCR, WRAP.
- Kích thước dữ liệu và độ dài truyền khác nhau.
- Địa chỉ ghi vượt quá giới hạn bộ nhớ của AXI4 SLAVE.

Điểm chung của các tập kiểm tra này là tại thời điểm 25ns đến 45ns, là thời gian kênh địa chỉ ghi của AXI4 MASTER bắt đầu gửi dữ liệu của kênh địa chỉ ghi đến AXI4 SLAVE. Thời gian tiếp theo là thời gian các kênh dữ liệu ghi và kênh phản hồi ghi hoạt động cho đến khi kết thúc một giao dịch ghi.

##### 4.2.1.1 Testcase 1

Trong testcase 1, tôi tiến hành kiểm tra quá trình ghi dữ liệu theo cơ chế FIXED với nhiều tập kiểm tra gồm các thông số địa chỉ, kích thước, độ dài và dữ liệu khác nhau.

*Trường hợp 1:* AXI4 MASTER ghi dữ liệu qua AXI4 SLAVE theo cơ chế FIXED, truyền 1 beat và 1 bytes cho một giao dịch ghi. Các thông số đầu vào được ghi ở bảng 4.2 dưới đây

**Bảng 4.2.** Thông số đầu vào của trường hợp 1 testcase 1

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_wr_addr    | 32'd0000_1235 |
| input_wr_len     | 2'b0          |
| input_wr_size    | 3'b000        |
| input_wr_burst   | 2'b00         |
| input_wr_data    | 32'h12345678  |

Với trường hợp này, dữ liệu ghi đầu vào là 12345678 (H), số bytes truyền là 1 bytes tương ứng với dữ liệu ghi là 78. Ta quan sát kết quả mô phỏng trong hình 4.1 dưới đây.



**Hình 4.1.** Dạng sóng trường hợp 1 testcase 1 của quá trình ghi

Tại 25ns, tín hiệu awvalid tích cực, AXI4 MASTER bắt đầu gửi thông tin địa chỉ ghi qua AXI4 SLAVE, nhưng lúc này, tín hiệu awready từ bên AXI4 SLAVE không tích cực, AXI4 SLAVE chưa sẵn sàng nhận dữ liệu. Tại 35ns, tín hiệu awready tích cực, AXI4 SLAVE đã sẵn sàng nhận thông tin. Đồng thời, cơ chế bắt tay awready và awvalid được thực hiện, thông tin địa chỉ được gửi từ AXI4 MASTER sang AXI4 SLAVE được chấp nhận. Tín hiệu wready cũng được tích cực tại thời điểm này, biểu thị cho việc AXI4 SLAVE đã sẵn sàng nhận dữ liệu ghi. Tại 45ns, quá trình gửi thông tin địa chỉ của kênh ghi địa chỉ hoàn tất, tín hiệu awready và awvalid không tiếp tục tích cực. Đồng thời, AXI4 MASTER bắt đầu gửi thông tin dữ liệu qua AXI4 SLAVE, tín hiệu wvalid được tích cực, cơ chế bắt tay giữa wvalid và wready được hình thành, dữ liệu gửi AXI4 MASTER sang AXI4 SLAVE được chấp nhận. Đối với trường hợp này, số beat truyền đi là một, do đó sau khi gửi dữ liệu đi một chu kỳ xung nhịp thì đã hoàn thành việc gửi dữ liệu đi một beat. Tín hiệu wlast được tích cực, xác nhận việc gửi dữ liệu hoàn tất. Tín hiệu bready = 1, biểu thị AXI4 MASTER sẵn sàng nhận phản hồi ghi từ AXI4 SLAVE. Tại 55ns, tín hiệu wvalid và wready không tiếp tục tích cực, quá trình gửi dữ liệu của kênh dữ liệu ghi hoàn tất. Tín hiệu bvalid = 1, cơ chế bắt tay của kênh phản hồi ghi hình thành, cho phép tín hiệu bresp gửi thông tin phản hồi ghi là 0, tương ứng với 'OKAY', đồng nghĩa với việc truyền và ghi dữ liệu từ AXI4 MASTER sang AXI4 SLAVE hoàn tất, không có lỗi phát sinh.

Địa chỉ ghi của trường hợp này là 1235 (H), cơ chế truyền là kiểu FIXED, tương ứng với mỗi beat dữ liệu ghi vào cùng một ô nhớ, kiểm tra bộ nhớ của AXI4 SLAVE sau quá trình ghi, kết quả như hình 4.2.

| Name               | Value                |
|--------------------|----------------------|
| mem_slave[0:16383] | XXXXXXXXXXXXXXXXXXXX |
| > [1234][31:0]     | XXXXXXXX             |
| > [1235][31:0]     | 00000078             |
| > [1236][31:0]     | XXXXXXXX             |

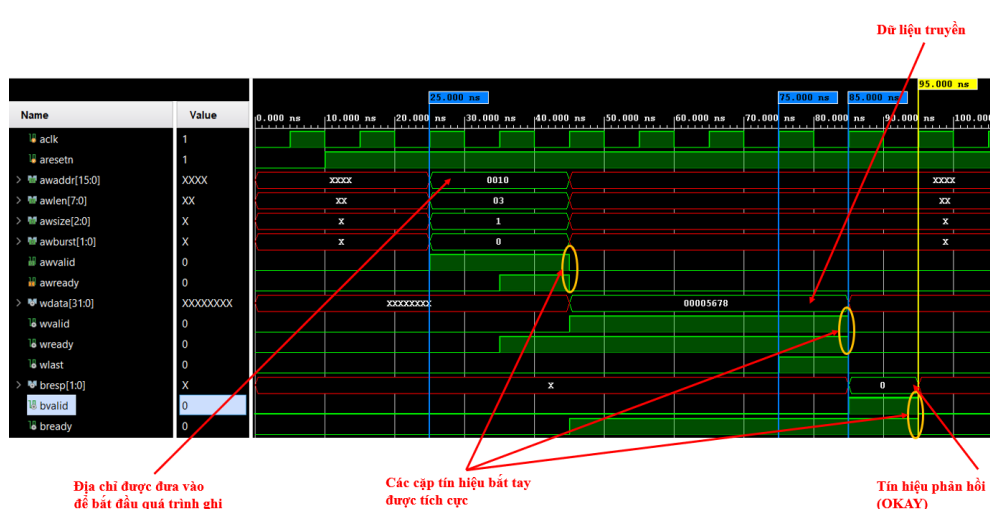
**Hình 4.2.** Bộ nhớ trường hợp 1 testcase 1 của AXI4 SLAVE quá trình ghi

Trường hợp 2: AXI4 MASTER ghi dữ liệu qua AXI4 SLAVE theo cơ chế FIXED, truyền 4 beat và 2 bytes cho một giao dịch ghi. Các thông số đầu vào được ghi ở bảng 4.3 dưới đây.

**Bảng 4.3.** Thông số đầu vào của trường hợp 2 testcase 1

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_wr_addr    | 32'd0000_0016 |
| input_wr_len     | 8'b3          |
| input_wr_size    | 8'b3          |
| input_wr_burst   | 2'b00         |
| input_wr_data    | 32'h12345678  |

Với trường hợp 2 này, dữ liệu ghi đầu vào là 12345678 (H), số bytes truyền là 2 bytes tương ứng với dữ liệu ghi là 5678. Ta quan sát kết quả mô phỏng trong hình 4.3 dưới đây.



**Hình 4.3.** Dạng sóng trường hợp 2 testcase 1 của quá trình ghi



Thời điểm từ 25ns đến 45ns là giai đoạn gửi thông tin địa chỉ ghi. Tại thời điểm này, các tín hiệu awready và awvalid tích cực và không tích cực giống với trường hợp 1. Điểm khác biệt giữa trường hợp 1 và trường hợp 2 là ở trường hợp 2, thông tin địa chỉ gửi là 10 (H), kích thước là 2 bytes và độ dài là 4 beat. Tại 45ns, quá trình gửi thông tin địa chỉ của kênh ghi địa chỉ hoàn tất, tín hiệu awready và awvalid không tiếp tục tích cực. Đồng thời, AXI4 MASTER bắt đầu gửi thông tin dữ liệu qua AXI4 SLAVE. Tín hiệu wvalid cũng được tích cực ngay tại thời điểm này, cơ chế bắt tay giữa wvalid và wready được hình thành, dữ liệu gửi AXI4 MASTER sang AXI4 SLAVE được chấp nhận. Tín hiệu bready = 1, biểu thị AXI4 MASTER sẵn sàng nhận phản hồi ghi từ AXI4 SLAVE. Đối với trường hợp này, số beat truyền đi là 4, do đó sau khi gửi dữ liệu đi 4 chu kỳ xung nhịp thì đã hoàn thành việc gửi dữ liệu đi 4 beat. Tín hiệu wvalid và wready được giữ ở mức tích cực trong suốt quá trình gửi dữ liệu. Tại 75ns, tín hiệu wlast được tích cực ở chu kỳ xung nhịp gửi dữ liệu thứ 4, xác nhận việc gửi dữ liệu hoàn tất. Tại 85ns, tín hiệu wvalid và wready không tiếp tục tích cực, quá trình gửi dữ liệu của kênh dữ liệu ghi hoàn tất. Tại thời điểm này, tín hiệu bvalid = 1, cơ chế bắt tay của kênh phản hồi ghi được hình thành, cho phép tín hiệu bresp gửi thông tin phản hồi ghi là 0, tương ứng với 'OKAY', đồng nghĩa với việc truyền và ghi dữ liệu từ AXI4 MASTER sang AXI4 SLAVE hoàn tất, không có lỗi phát sinh.

Địa chỉ ghi của tệp kiểm tra này là 16 (D), cơ chế truyền là kiểu FIXED, tương ứng với mỗi beat dữ liệu ghi vào cùng một ô nhớ, kiểm tra bộ nhớ của AXI4 SLAVE sau quá trình ghi, kết quả như hình 4.4.

| Name               | Value               |
|--------------------|---------------------|
| mem_slave[0:16383] | XXXXXXXX,XXXXXXXX,X |
| > [14][31:0]       | XXXXXXXX            |
| > [15][31:0]       | XXXXXXXX            |
| > [16][31:0]       | 00005678            |
| > [17][31:0]       | XXXXXXXX            |
| > [18][31:0]       | XXXXXXXX            |

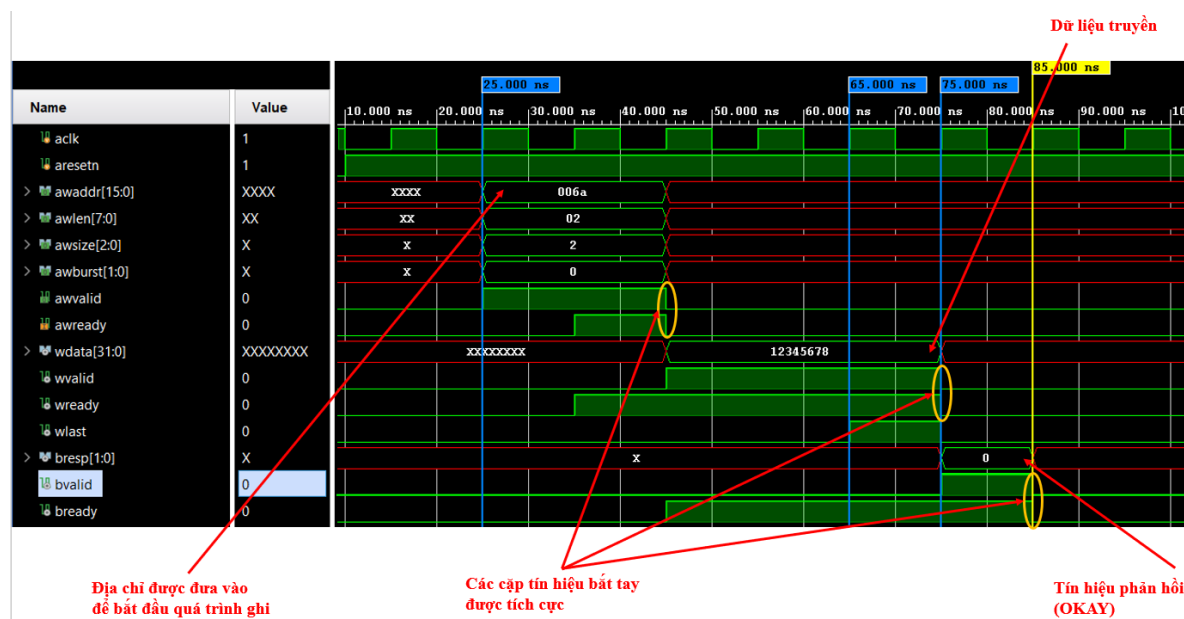
**Hình 4.4.** Bộ nhớ trường hợp 2 testcase 1 của AXI4 SLAVE quá trình ghi

*Trường hợp 3:* AXI4 MASTER ghi dữ liệu qua AXI4 SLAVE theo cơ chế FIXED, truyền 3 beat và 4 bytes cho một giao dịch ghi. Các thông số đầu vào được ghi ở bảng 4.4 dưới đây.

**Bảng 4.4.** Thông số đầu vào của trường hợp 3 testcase 1

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_wr_addr    | 32'd0000_0106 |
| input_wr_len     | 8'b2          |
| input_wr_size    | 3'b010        |
| input_wr_burst   | 2'b00         |
| input_wr_data    | 32'h12345678  |

Với tệp kiểm tra này, dữ liệu ghi đầu vào là 12345678 (H), số bytes truyền là 4 bytes tương ứng với dữ liệu ghi là 12345678. Ta quan sát kết quả mô phỏng trong hình 4.5 dưới đây.



**Hình 4.5.** Dạng sóng trường hợp 3 testcase 1 của quá trình ghi

Trong trường hợp 3 này, các thời điểm tích cực và không tích cực của các tín hiệu giống với trường hợp 2, điểm khác biệt duy nhất là số bytes dữ liệu truyền cho một beat là 4.

Địa chỉ ghi của tệp kiểm tra này là 106 (D), cơ chế truyền là kiểu FIXED, tương ứng với mỗi beat dữ liệu ghi vào cùng một ô nhớ, kiểm tra bộ nhớ của AXI4 SLAVE sau quá trình ghi, kết quả như hình 4.6.

| Name               | Value         |
|--------------------|---------------|
| mem_slave[0:16383] | XXXXXXXX,XXXX |
| > [104][31:0]      | XXXXXXXX      |
| > [105][31:0]      | XXXXXXXX      |
| > [106][31:0]      | 12345678      |
| > [107][31:0]      | XXXXXXXX      |
| > [108][31:0]      | XXXXXXXX      |

**Hình 4.6.** Bộ nhớ trường hợp 3 testcase 1 của AXI SLAVE cho quá trình ghi

Qua các trường hợp của testcase 1, ta có thể thấy được rằng đối với kiểu cơ chế truyền FIXED, dù cho kích thước và độ dài của dữ liệu có thay đổi thì dữ liệu truyền vào bộ nhớ vẫn được ghi vào cùng một địa chỉ ô nhớ được yêu cầu ghi ban đầu.

#### 4.2.1.2 Testcase 2

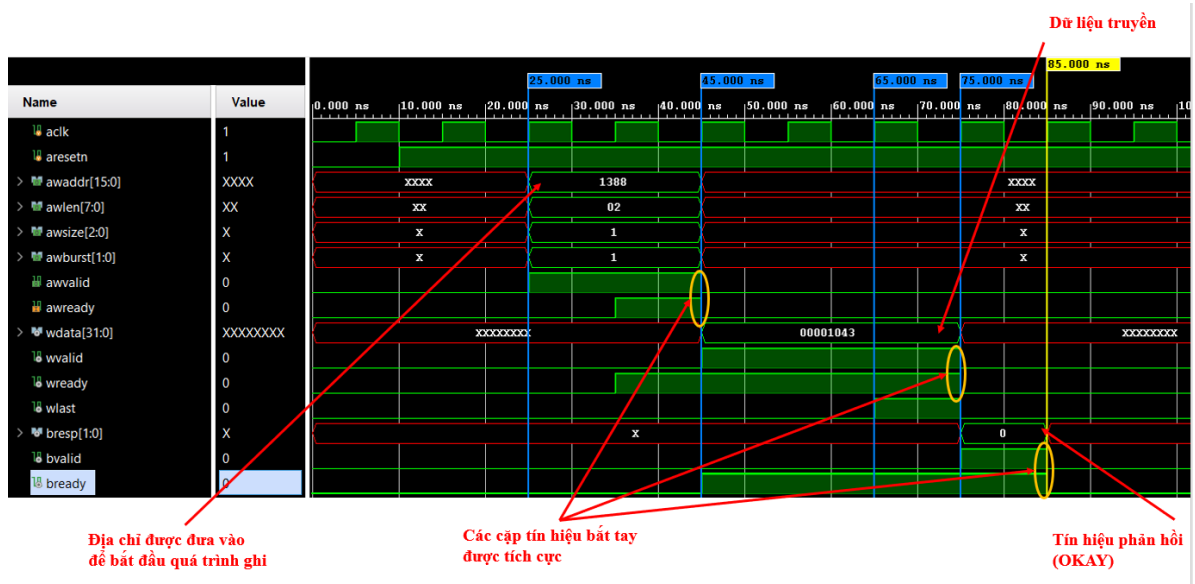
Trong testcase 2, tôi tiến hành kiểm tra quá trình ghi dữ liệu theo cơ chế INCR với nhiều tệp kiểm tra gồm các thông số địa chỉ, kích thước, độ dài và dữ liệu khác nhau.

*Trường hợp 1:* AXI4 MASTER ghi dữ liệu qua AXI4 SLAVE theo cơ chế INCR, truyền 3 beat và 2 bytes cho một giao dịch ghi. Các thông số đầu vào được ghi ở bảng 4.5 dưới đây.

**Bảng 4.5.** Thông số đầu vào của trường hợp 1 testcase 2

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_wr_addr    | 32'd0000_5000 |
| input_wr_len     | 2'b10         |
| input_wr_size    | 3'b001        |
| input_wr_burst   | 2'b01         |
| input_wr_data    | 32'h22161043  |












Với trường hợp này, dữ liệu ghi đầu vào là 22161043 (H), số bytes truyền là 2 bytes tương ứng với dữ liệu ghi là 1043. Ta quan sát kết quả mô phỏng trong hình 4.7 dưới đây.



**Hình 4.7.** Dạng sóng trường hợp 1 testcase 2 của quá trình ghi

Thời điểm từ 25ns đến 45ns là giai đoạn gửi thông tin địa chỉ ghi. Tại thời điểm này, các tín hiệu awready và awvalid, wready và wvalid tích cực và không tích cực giống với testcase 1. Tại 45ns, AXI4 MASTER bắt đầu gửi thông tin dữ liệu qua AXI4 SLAVE, cơ chế bắt tay giữa wvalid và wready được hình thành, dữ liệu gửi AXI4 MASTER sang AXI4 SLAVE được chấp nhận. Tín hiệu bready = 1, biểu thị AXI4 MASTER sẵn sàng nhận phản hồi ghi từ AXI4 SLAVE. Đối với trường hợp này, số beat truyền đi là ba, do đó sau khi gửi dữ liệu đi ba chu kỳ xung nhịp thì đã hoàn thành việc gửi dữ liệu. Tại 65ns, tín hiệu wlast được tích cực, xác nhận việc gửi dữ liệu hoàn tất. Tại 75ns, tín hiệu wvalid và wready không tiếp tục tích cực, quá trình gửi dữ liệu của kênh dữ liệu ghi hoàn tất. Tại thời điểm này, tín hiệu bvalid = 1, cơ chế bắt tay của kênh phản hồi ghi được hình thành, cho phép tín hiệu bresp gửi thông tin phản hồi ghi là 0, tương ứng với ‘OKAY’, đồng nghĩa với việc truyền và ghi dữ liệu từ AXI4 MASTER sang AXI4 SLAVE hoàn tất, không có lỗi phát sinh.

Địa chỉ ghi của trường hợp này là 5000 (D), cơ chế truyền là kiểu INCR, tương ứng với mỗi beat dữ liệu ghi vào địa chỉ ô nhớ tăng dần. Địa chỉ ghi vào ô nhớ bắt đầu giống với địa chỉ AXI4 MASTER gửi ban đầu, sau mỗi beat, địa chỉ ghi tăng thêm một giá trị tương ứng với số bytes được gửi. Kiểm tra bộ nhớ của AXI4 SLAVE sau quá trình ghi, kết quả như hình 4.8.

| Name                                                                                                   | Value          |
|--------------------------------------------------------------------------------------------------------|----------------|
| ▼  mem_slave[0:16383] | XXXXXXXXXX,XXX |
| >  [4997][31:0]       | XXXXXXXXXX     |
| >  [4998][31:0]       | XXXXXXXXXX     |
| >  [4999][31:0]       | XXXXXXXXXX     |
| >  [5000][31:0]       | XXXX1043       |
| >  [5001][31:0]       | XXXXXXXXXX     |
| >  [5002][31:0]       | XXXX1043       |
| >  [5003][31:0]       | XXXXXXXXXX     |
| >  [5004][31:0]       | XXXX1043       |
| >  [5005][31:0]       | XXXXXXXXXX     |
| >  [5006][31:0]      | XXXXXXXXXX     |

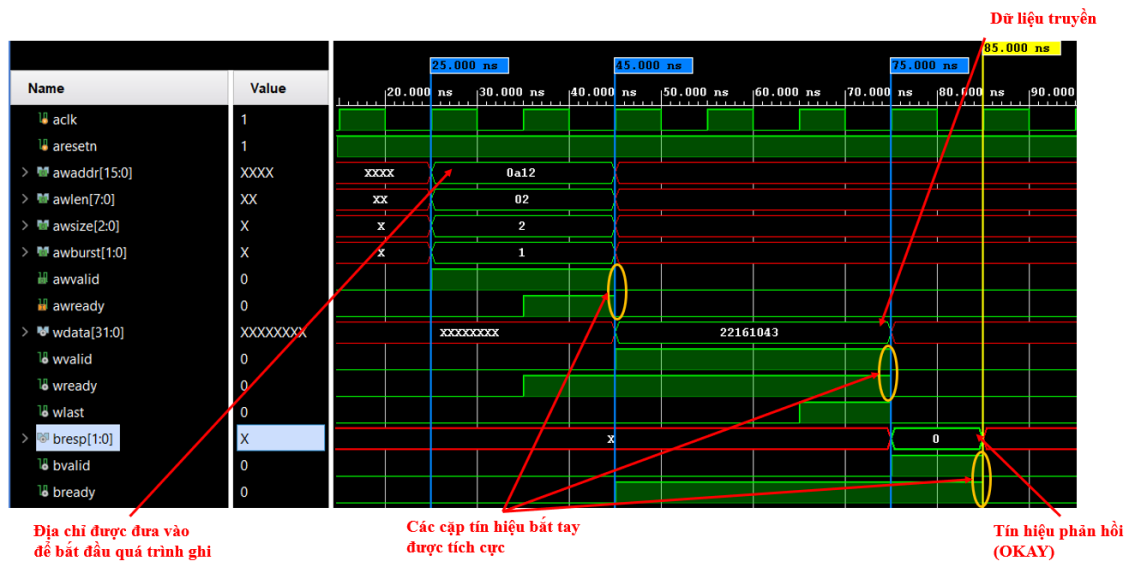
**Hình 4.8.** Bộ nhớ trường hợp 1 testcase 2 của AXI4 SLAVE quá trình ghi

Trường hợp 2: AXI MASTER ghi dữ liệu qua AXI4 SLAVE theo cơ chế INCR, truyền 3 beat và 4 bytes cho một giao dịch ghi. Các thông số đầu vào được ghi ở bảng 4.6 dưới đây.

**Bảng 4.6.** Thông số đầu vào của trường hợp 2 testcase 2

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_wr_addr    | 32'd0000_2578 |
| input_wr_len     | 2'b10         |
| input_wr_size    | 3'b010        |
| input_wr_burst   | 2'b01         |
| input_wr_data    | 32'h22161043  |

Với trường hợp 2 này, dữ liệu ghi đầu vào là 22161043 (D), số bytes truyền là 4 bytes tương ứng với dữ liệu ghi là 22161043. Ta quan sát kết quả mô phỏng trong hình 4.9 dưới đây.



**Hình 4.9.** Dạng sóng trường hợp 2 testcase 2 của quá trình ghi

Thời điểm từ 25ns đến 45ns là giai đoạn gửi thông tin địa chỉ ghi. Tại thời điểm này, các tín hiệu awready và awvalid tích cực và không tích cực giống với trường hợp 1. Tại 45ns, quá trình gửi thông tin địa chỉ của kênh ghi địa chỉ hoàn tất, tín hiệu awready và awvalid không tiếp tục tích cực. Đồng thời, AXI4 MASTER bắt đầu gửi thông tin dữ liệu qua AXI4 SLAVE. Tín hiệu wvalid cũng được tích cực ngay tại thời điểm này, cơ chế bắt tay giữa wvalid và wready được hình thành, dữ liệu gửi AXI4 MASTER sang AXI4 SLAVE được chấp nhận. Tín hiệu bready = 1, biểu thị AXI4 MASTER sẵn sàng nhận phản hồi ghi từ AXI4 SLAVE. Đối với trường hợp này, số beat truyền đi là 3, do đó sau khi gửi dữ liệu đi 3 chu kỳ xung nhịp thì đã hoàn thành việc gửi dữ liệu. Tín hiệu wvalid và wready được giữ ở mức tích cực trong suốt quá trình gửi dữ liệu. Tại 65ns, tín hiệu wlast được tích cực ở chu kỳ xung nhịp gửi dữ liệu thứ 4, xác nhận việc gửi dữ liệu hoàn tất. Tại 75ns, tín hiệu wvalid và wready không tiếp tục tích cực, quá trình gửi dữ liệu của kênh dữ liệu ghi hoàn tất. Tại thời điểm này, tín hiệu bvalid = 1, cơ chế bắt tay của kênh phản hồi ghi được hình thành, cho phép tín hiệu bresp gửi thông tin phản hồi ghi là 0, tương ứng với 'OKAY', đồng nghĩa với việc truyền và ghi dữ liệu từ AXI4 MASTER sang AXI4 SLAVE hoàn tất, không có lỗi phát sinh.

Địa chỉ ghi của trường hợp này là 2578 (D), cơ chế truyền là kiểu INCR, tương ứng với mỗi beat dữ liệu ghi vào địa chỉ ô nhớ tăng dần. Địa chỉ ghi vào ô



nhớ bắt đầu giống với địa chỉ AXI4 MASTER gửi ban đầu, sau mỗi beat, địa chỉ ghi tăng thêm một giá trị tương ứng với số bytes được gửi. Kiểm tra bộ nhớ của AXI4 SLAVE sau quá trình ghi, kết quả như hình 4.10.

| Name                 | Value      |
|----------------------|------------|
| ▼ mem_slave[0:16383] | XXXXXXXXXX |
| > [2575][31:0]       | XXXXXXXXXX |
| > [2576][31:0]       | XXXXXXXXXX |
| > [2577][31:0]       | XXXXXXXXXX |
| > [2578][31:0]       | 22161043   |
| > [2579][31:0]       | XXXXXXXXXX |
| > [2580][31:0]       | XXXXXXXXXX |
| > [2581][31:0]       | XXXXXXXXXX |
| > [2582][31:0]       | 22161043   |
| > [2583][31:0]       | XXXXXXXXXX |
| > [2584][31:0]       | XXXXXXXXXX |
| > [2585][31:0]       | XXXXXXXXXX |
| > [2586][31:0]       | 22161043   |
| > [2587][31:0]       | XXXXXXXXXX |
| > [2588][31:0]       | XXXXXXXXXX |
| > [2589][31:0]       | XXXXXXXXXX |
| > [2590][31:0]       | XXXXXXXXXX |

**Hình 4.10.** Bộ nhớ trường hợp 2 testcase 2 của AXI4 SLAVE quá trình ghi

Qua các trường hợp của testcase 2, ta có thể thấy được rằng đối với kiểu cơ chế truyền INCR, địa chỉ ô nhớ đầu tiên là địa chỉ được gửi từ AXI4 MASTER, sau mỗi beat thì địa chỉ ghi vào ô nhớ tăng dần và phụ thuộc vào kích thước và độ dài của dữ liệu gửi.

#### 4.2.1.3 Testcase 3

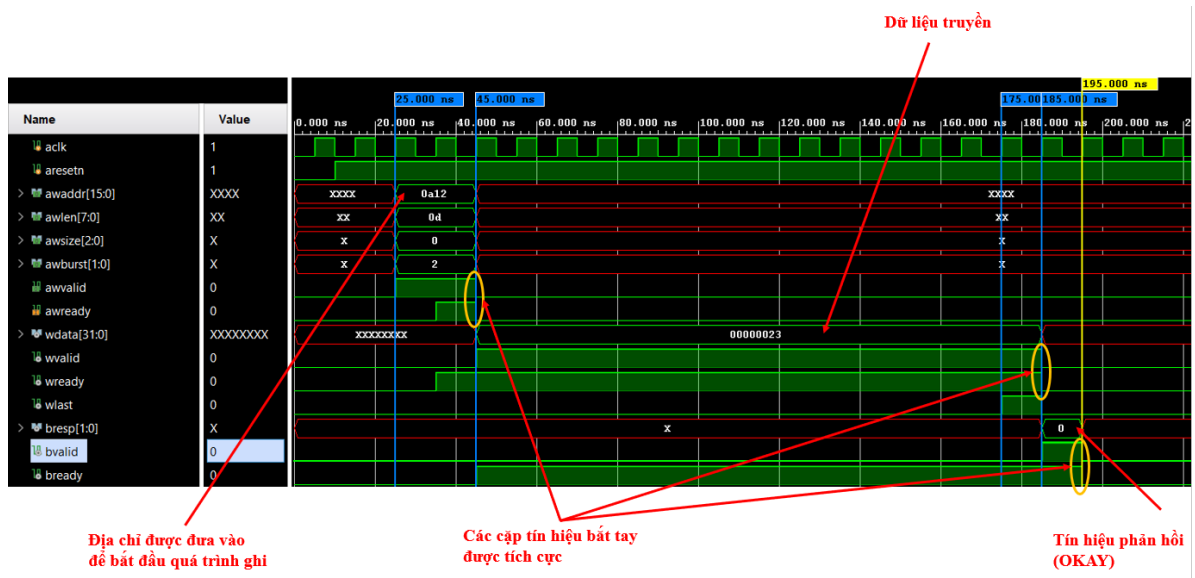
Trong testcase 3, tôi tiến hành kiểm tra quá trình ghi dữ liệu theo cơ chế WRAP với nhiều tệp kiểm tra gồm các thông số địa chỉ, kích thước, độ dài và dữ liệu khác nhau.

*Trường hợp 1:* AXI4 MASTER ghi dữ liệu qua AXI4 SLAVE theo cơ chế WRAP, truyền 14 beat và 1 bytes cho một giao dịch ghi. Các thông số đầu vào được ghi ở bảng 4.7 dưới đây.

**Bảng 4.7.** Thông số đầu vào của trường hợp 1 testcase 3

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_wr_addr    | 32'd0000_2578 |
| input_wr_len     | 4'b1101       |
| input_wr_size    | 3'b000        |
| input_wr_burst   | 2'b10         |
| input_wr_data    | 32'h45679123  |

Với trường hợp này, dữ liệu ghi đầu vào là 456789123 (H), số bytes truyền là 1 bytes tương ứng với dữ liệu ghi là 23. Ta quan sát kết quả mô phỏng trong hình 4.11 dưới đây.



**Hình 4.11.** Dạng sóng trường hợp 1 testcase 3 của quá trình ghi

Thời điểm từ 25ns đến 45ns là giai đoạn gửi thông tin địa chỉ ghi. Tại thời điểm này, các tín hiệu awready và awvalid, wready và wvalid tích cực và không tích cực giống với testcase 1. Tại 45ns, AXI4 MASTER bắt đầu gửi thông tin dữ liệu qua AXI4 SLAVE, cơ chế bắt tay giữa wvalid và wready được hình thành, dữ liệu gửi AXI4 MASTER sang AXI4 SLAVE được chấp nhận. Tín hiệu bready = 1, biểu thị AXI4 MASTER sẵn sàng nhận phản hồi ghi từ AXI4 SLAVE. Đối với trường hợp này, số beat truyền đi là 14, nên sau 14 xung nhịp thì quá trình gửi dữ liệu từ AXI4 MASTER sang AXI4 SLAVE hoàn tất. Tại 175ns, tín hiệu wlast được tích cực, xác nhận việc gửi dữ liệu hoàn tất. Tại 185ns, tín hiệu



wvalid và wready không tiếp tục tích cực, quá trình gửi dữ liệu của kênh dữ liệu ghi hoàn tất. Tại thời điểm này, tín hiệu bvalid = 1, cơ chế bắt tay của kênh phản hồi ghi được hình thành, cho phép tín hiệu bresp gửi thông tin phản hồi ghi là 0, tương ứng với ‘OKAY’, đồng nghĩa với việc truyền và ghi dữ liệu từ AXI4 MASTER sang AXI4 SLAVE hoàn tất, không có lỗi phát sinh. Quá trình ghi dữ liệu từ AXI4 MASTER sang AXI4 SLAVE hoàn tất vào thời điểm 195ns.

Địa chỉ ghi của trường hợp này là 2578 (D) và cơ chế truyền là kiểu WRAP. Địa chỉ ô nhớ bắt đầu ghi chính là địa chỉ ghi được gửi từ AXI4 MASTER tương ứng với giá trị 2578 (D) trong trường hợp này. Khi ghi đến trước địa chỉ cuộn, địa chỉ ghi sẽ được cuộn về ô nhớ có địa chỉ bằng với giá trị của wrap\_boundary và ghi tiếp dữ liệu cho đến khi đủ số beat yêu cầu. Trong trường hợp này, ta dùng công thức (2.4), (2.5) để tính toán được địa chỉ cuộn và địa chỉ tới hạn có giá trị như sau:

$$\text{wrap\_boundary} = \text{INT} \left[ \frac{2578}{(13+1) \times 2^0} \right] \times (13+1) \times 2^0 = 2576 \quad (2.8)$$

$$\text{address\_n} = 2576 + (13+1) \times 2^0 = 2590 \quad (2.9)$$

Kiểm tra bộ nhớ của AXI4 SLAVE sau quá trình ghi, kết quả như hình 4.12.

| Name               | Value         |
|--------------------|---------------|
| mem_slave[0:16383] | XXXXXXXX,XXXX |
| > [2574][31:0]     | XXXXXXXX      |
| > [2575][31:0]     | XXXXXXXX      |
| > [2576][31:0]     | XXXXXX23      |
| > [2577][31:0]     | XXXXXX23      |
| > [2578][31:0]     | XXXXXX23      |
| > [2579][31:0]     | XXXXXX23      |
| > [2580][31:0]     | XXXXXX23      |
| > [2581][31:0]     | XXXXXX23      |
| > [2582][31:0]     | XXXXXX23      |
| > [2583][31:0]     | XXXXXX23      |
| > [2584][31:0]     | XXXXXX23      |
| > [2585][31:0]     | XXXXXX23      |
| > [2586][31:0]     | XXXXXX23      |
| > [2587][31:0]     | XXXXXX23      |
| > [2588][31:0]     | XXXXXX23      |
| > [2589][31:0]     | XXXXXX23      |
| > [2590][31:0]     | XXXXXXXX      |
| > [2591][31:0]     | XXXXXXXX      |

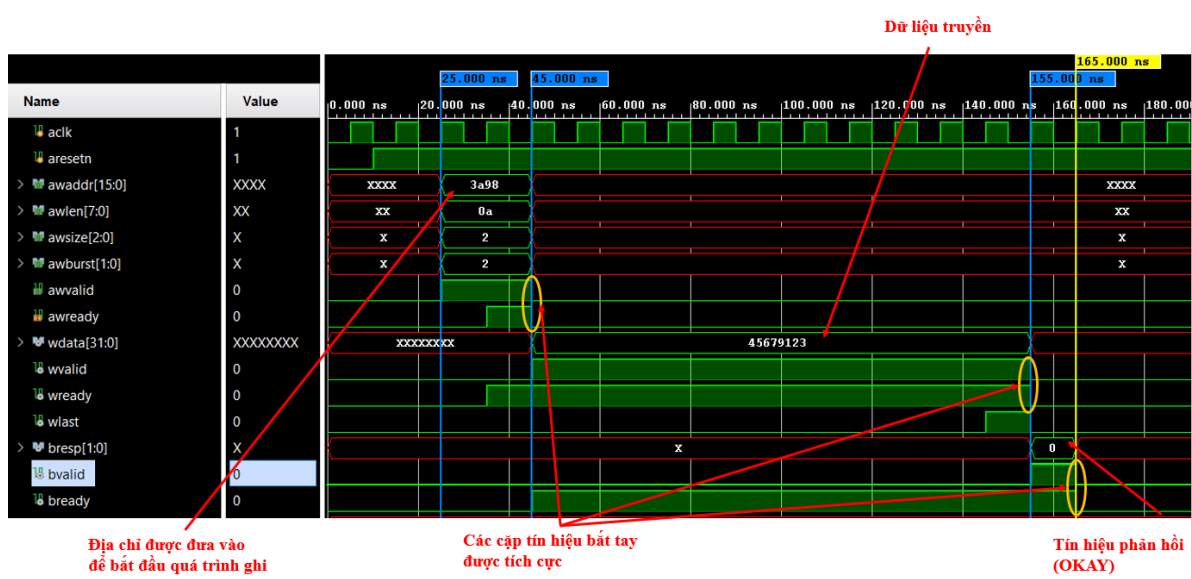
**Hình 4.12.** Bộ nhớ trường hợp 1 testcase 3 của AXI4 SLAVE quá trình ghi

Trường hợp 2: AXI4 MASTER ghi dữ liệu qua AXI4 SLAVE theo cơ chế WRAP, truyền 11 beat và 4 bytes cho một giao dịch ghi. Các thông số đầu vào được ghi ở bảng 4.8 dưới đây.

**Bảng 4.8.** Thông số đầu vào của trường hợp 2 testcase 3

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_wr_addr    | 32'd0001_5000 |
| input_wr_len     | 8'b0000_1010  |
| input_wr_size    | 3'b010        |
| input_wr_burst   | 2'b10         |
| input_wr_data    | 32'h45679123  |

Với trường hợp 2 này, dữ liệu ghi đầu vào là 456789123 (D), số bytes truyền là 4 bytes tương ứng với dữ liệu ghi là 456789123. Ta quan sát kết quả mô phỏng trong hình 4.13 dưới đây.



**Hình 4.13.** Dạng sóng trường hợp 2 testcase 3 của quá trình ghi

Tại 25ns, tín hiệu awvalid tích cực, AXI4 MASTER bắt đầu gửi thông tin địa chỉ ghi qua AXI4 SLAVE, nhưng lúc này, tín hiệu awready từ bên AXI4 SLAVE không tích cực, AXI4 SLAVE chưa sẵn sàng nhận dữ liệu. Tại 35ns, tín hiệu awready tích cực, AXI4 SLAVE đã sẵn sàng nhận thông tin. Đồng thời, cơ chế bắt tay awready và awvalid được thực hiện, thông tin địa chỉ được gửi từ AXI4 MASTER sang AXI4 SLAVE được chấp nhận. Tín hiệu wready cũng được tích

cực tại thời điểm này, biểu thị cho việc AXI4 SLAVE đã sẵn sàng nhận dữ liệu ghi. Tại 45ns, quá trình gửi thông tin địa chỉ của kênh ghi địa chỉ hoàn tất, tín hiệu awready và awvalid không tiếp tục tích cực. Đồng thời, AXI4 MASTER bắt đầu gửi thông tin dữ liệu qua AXI4 SLAVE, tín hiệu wvalid được tích cực, cơ chế bắt tay giữa wvalid và wready được hình thành, dữ liệu gửi AXI4 MASTER sang AXI4 SLAVE được chấp nhận. Đối với trường hợp này, số beat truyền đi là 11, do đó sau khi gửi dữ liệu đi 11 chu kỳ xung nhịp thì đã hoàn thành việc gửi dữ liệu. Tín hiệu wvalid và wready được giữ ở mức tích cực trong suốt quá trình gửi dữ liệu. Tại 145ns, tín hiệu wlast được tích cực xác nhận việc gửi dữ liệu hoàn tất. Tại 155ns, tín hiệu wvalid và wready không tiếp tục tích cực, quá trình gửi dữ liệu của kênh dữ liệu ghi hoàn tất. Tại thời điểm này, tín hiệu bvalid = 1, cơ chế bắt tay của kênh phản hồi ghi được hình thành, cho phép tín hiệu bresp gửi thông tin phản hồi ghi là 0, tương ứng với 'OKAY', đồng nghĩa với việc truyền và ghi dữ liệu từ AXI4 MASTER sang AXI4 SLAVE hoàn tất, không có lỗi phát sinh.

Địa chỉ ghi của trường hợp này là 15000 (D) và cơ chế truyền là kiểu WRAP. Địa chỉ ô nhớ bắt đầu ghi chính là địa chỉ ghi được gửi từ AXI4 MASTER tương ứng với giá trị 15000 (D) trong trường hợp này. Khi ghi đến trước địa chỉ cuộn, địa chỉ ghi sẽ được cuộn về ô nhớ có địa chỉ bằng với giá trị của wrap\_boundary và ghi tiếp dữ liệu cho đến khi đủ số beat yêu cầu. Nhưng địa chỉ ghi sẽ không được ghi liền mạch như ở trường hợp 1 mà sẽ có khoảng tăng với nguyên tắc tăng địa chỉ giống với cơ chế INCR. Trong trường hợp này, ta dùng công thức (2.4), (2.5) để tính toán được địa chỉ cuộn và địa chỉ tới hạn có giá trị như sau:

$$\text{wrap\_boundary} = \text{INT} \left[ \frac{15000}{(10+1) \cdot 2^2} \right] \times (10+1) \times 2^2 = 14960 \quad (2.10)$$

$$\text{address\_n} = 14960 + (10+1) \times 2^2 = 15004 \quad (2.11)$$

Kiểm tra bộ nhớ của AXI4 SLAVE sau quá trình ghi, ta nhận thấy được địa chỉ ghi ban đầu là 15000, địa chỉ ghi kế tiếp là 15004 bằng với địa chỉ tới hạn. Do đó, sau khi ghi dữ liệu vào địa chỉ ô nhớ ban đầu, địa chỉ ghi sẽ được cuộn lại địa chỉ cuộn và tiếp tục ghi dữ liệu cho đến khi ghi đầy đủ số lượng beat yêu cầu. Kết quả được dải ô nhớ như hình 4.14.

| Name               | Value              | Name               | Value              |
|--------------------|--------------------|--------------------|--------------------|
| mem_slave[0:16383] | XXXXXXXXXXXXXXXXXX | mem_slave[0:16383] | XXXXXXXXXXXXXXXXXX |
| > [14958][31:0]    | XXXXXXXXXX         | > [14988][31:0]    | 45679123           |
| > [14959][31:0]    | XXXXXXXXXX         | > [14989][31:0]    | XXXXXXXXXX         |
| > [14960][31:0]    | 45679123           | > [14990][31:0]    | XXXXXXXXXX         |
| > [14961][31:0]    | XXXXXXXXXX         | > [14991][31:0]    | XXXXXXXXXX         |
| > [14962][31:0]    | XXXXXXXXXX         | > [14992][31:0]    | 45679123           |
| > [14963][31:0]    | XXXXXXXXXX         | > [14993][31:0]    | XXXXXXXXXX         |
| > [14964][31:0]    | 45679123           | > [14994][31:0]    | XXXXXXXXXX         |
| > [14965][31:0]    | XXXXXXXXXX         | > [14995][31:0]    | XXXXXXXXXX         |
| > [14966][31:0]    | XXXXXXXXXX         | > [14996][31:0]    | 45679123           |
| > [14967][31:0]    | XXXXXXXXXX         | > [14997][31:0]    | XXXXXXXXXX         |
| > [14968][31:0]    | 45679123           | > [14998][31:0]    | XXXXXXXXXX         |
| > [14969][31:0]    | XXXXXXXXXX         | > [14999][31:0]    | XXXXXXXXXX         |
| > [14970][31:0]    | XXXXXXXXXX         | > [15000][31:0]    | 45679123           |
| > [14971][31:0]    | XXXXXXXXXX         | > [15001][31:0]    | XXXXXXXXXX         |
| > [14972][31:0]    | 45679123           | > [15002][31:0]    | XXXXXXXXXX         |
| > [14973][31:0]    | XXXXXXXXXX         | > [15003][31:0]    | XXXXXXXXXX         |
| > [14974][31:0]    | XXXXXXXXXX         | > [15004][31:0]    | XXXXXXXXXX         |
| > [14975][31:0]    | XXXXXXXXXX         |                    |                    |
| > [14976][31:0]    | 45679123           |                    |                    |
| > [14977][31:0]    | XXXXXXXXXX         |                    |                    |
| > [14978][31:0]    | XXXXXXXXXX         |                    |                    |
| > [14979][31:0]    | XXXXXXXXXX         |                    |                    |
| > [14980][31:0]    | 45679123           |                    |                    |
| > [14981][31:0]    | XXXXXXXXXX         |                    |                    |
| > [14982][31:0]    | XXXXXXXXXX         |                    |                    |
| > [14983][31:0]    | XXXXXXXXXX         |                    |                    |
| > [14984][31:0]    | 45679123           |                    |                    |
| > [14985][31:0]    | XXXXXXXXXX         |                    |                    |
| > [14986][31:0]    | XXXXXXXXXX         |                    |                    |
| > [14987][31:0]    | XXXXXXXXXX         |                    |                    |

**Hình 4.14.** Bộ nhớ trường hợp 2 testcase 3 của AXI SLAVE quá trình ghi

Qua các trường hợp của testcase 3, ta có thể thấy được rằng đối với kiểu cơ chế truyền WRAP, địa chỉ ô nhớ đầu tiên là địa chỉ được gửi từ AXI4 MASTER, sau mỗi beat thì địa chỉ ghi vào ô nhớ tăng dần giống với kiểu cơ chế INCR. Đến khi địa chỉ ghi kế tiếp đạt đến địa chỉ tới hạn, địa chỉ ghi sẽ được cuộn về địa chỉ cuộn và tiếp tục ghi dữ liệu cho đến khi đủ số beat yêu cầu.

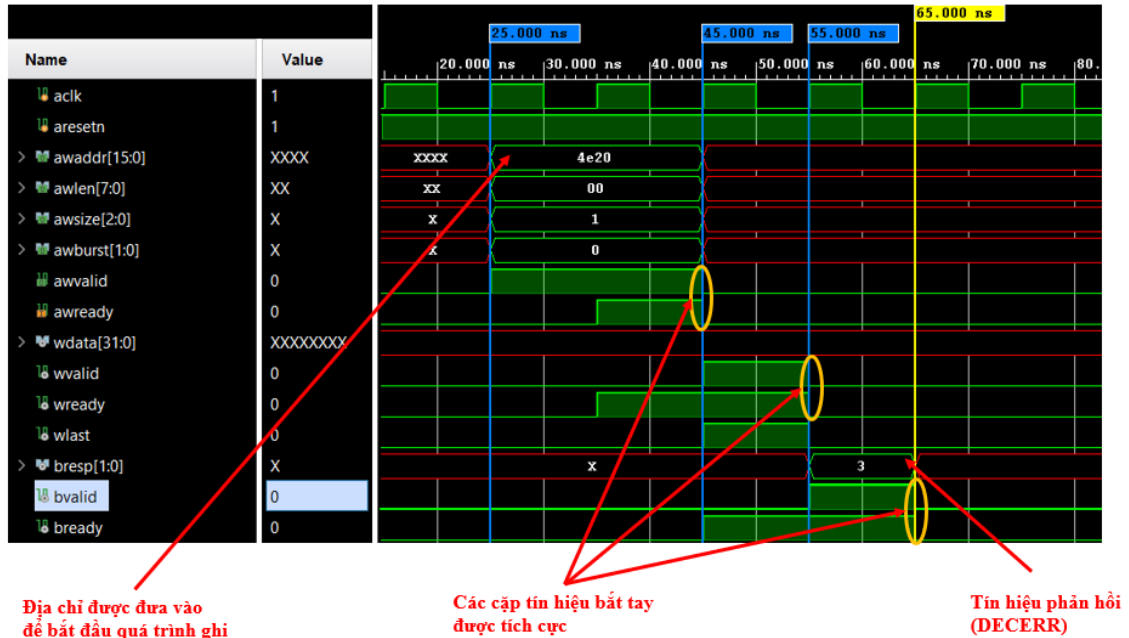
#### 4.2.1.4 Testcase 4

Trong testcase 4, tôi tiến hành kiểm tra quá trình ghi dữ liệu vượt quá giới hạn bộ nhớ cho phép. Bộ nhớ của AXI4 SLAVE là 16383 ô nhớ, ta cho địa chỉ bắt đầu ghi dữ liệu là 20000, vượt qua số lượng ô nhớ của AXI4 SLAVE. Các thông số đầu vào được ghi ở bảng 4.9 dưới đây.

**Bảng 4.9.** Thông số đầu vào của testcase 4

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ      |
|------------------|--------------|
| input_wr_addr    | 15'd20000    |
| input_wr_len     | 8'd0         |
| input_wr_size    | 3'b001       |
| input_wr_burst   | 2'b00        |
| input_wr_data    | 32'h12345678 |

Với trường hợp này, dữ liệu ghi đầu vào là 12345678 (H), số bytes truyền là 1 bytes tương ứng với dữ liệu ghi là 23. Ta quan sát kết quả mô phỏng trong hình 4.15 dưới đây.



**Hình 4.15.** Dạng sóng testcase 4 của quá trình ghi

Thời điểm từ 25ns đến 45ns là giai đoạn gửi thông tin địa chỉ ghi. Tại thời điểm này, các tín hiệu awready và awvalid, wready và wvalid tích cực và không tích cực giống với testcase 1. Tại 45ns, AXI4 MASTER bắt đầu gửi thông tin dữ liệu qua AXI4 SLAVE, cơ chế bắt tay giữa wvalid và wready được hình thành, dữ liệu gửi AXI4 MASTER sang AXI4 SLAVE được chấp nhận. Tín hiệu bready = 1, biểu thị AXI4 MASTER sẵn sàng nhận phản hồi ghi từ AXI4 SLAVE. Đối với trường hợp này, địa chỉ ghi vượt quá giới hạn bộ nhớ của AXI4 SLAVE nên dữ liệu ghi sẽ không được truyền đi mà tín hiệu wlast được tích cực ngay lập tức. Tại 55ns, tín hiệu wvalid và wready không tiếp tục tích cực, quá trình gửi dữ liệu của kênh dữ liệu ghi hoàn tất. Tại thời điểm này, tín hiệu bvalid = 1, cơ chế bắt tay của kênh phản hồi ghi được hình thành, cho phép tín hiệu bresp gửi thông tin phản hồi ghi là 3, tương ứng với 'DECERR', đồng nghĩa với việc truyền và ghi dữ liệu từ AXI4 MASTER sang AXI SLAVE phát sinh lỗi địa chỉ.

### 4.2.2 Hoạt động đọc dữ liệu

Xây dựng các tệp kiểm tra của quá trình đọc với mục tiêu kiểm thử các trường hợp để mô phỏng quá trình truyền dữ liệu từ AXI4 SLAVE sang AXI4 MASTER thông qua giao thức AXI4 bao gồm các trường hợp:

- Các chế độ truyền burst: FIXED, INCR, WRAP.
- Kích thước dữ liệu và độ dài truyền khác nhau.
- Địa chỉ đọc vượt quá bộ nhớ của AXI4 SLAVE.

Điểm chung của các tệp kiểm tra này là tại thời điểm  $t = 25\text{ns}$ , AXI4 MASTER bắt đầu gửi dữ liệu của kênh địa chỉ đọc nhưng tín hiệu `awready` chưa tích cực nên AXI4 SLAVE chưa sẵn sàng nhận thông tin về địa chỉ đọc. Sau khi cặp tín hiệu `awvalid` và `awready` được tích cực và gửi địa chỉ qua AXI4 SLAVE, tín hiệu `rvalid` và `rready` được tích cực, giữ mức tích cực trong suốt quá trình đọc dữ liệu. Lúc này, dữ liệu đọc đã được truyền qua AXI4 MASTER. Khi beat cuối cùng của dữ liệu đọc được truyền, tín hiệu `rlast` sẽ được tích cực trong một chu kỳ xung nhịp. Trong cùng thời điểm đó, tín hiệu `rresp` gửi phản hồi dữ liệu đọc có hợp lệ hay không.

#### 4.2.2.1 Testcase 1

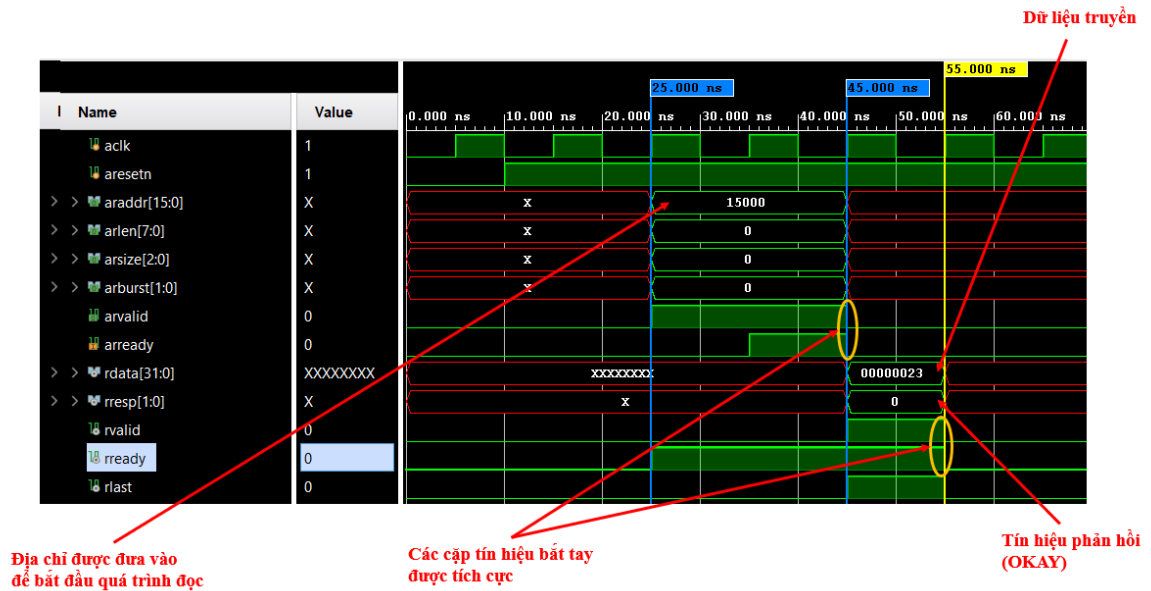
Trong testcase 1, tôi tiến hành kiểm tra quá trình đọc dữ liệu theo cơ chế FIXED với nhiều tệp kiểm tra gồm các thông số địa chỉ, kích thước, độ dài và dữ liệu khác nhau.

*Trường hợp 1:* AXI4 MASTER đọc dữ liệu từ AXI4 SLAVE theo cơ chế FIXED, truyền 1 beat và 1 bytes cho một giao dịch đọc. Các thông số đầu vào được ghi ở bảng 4.10 dưới đây.

**Bảng 4.10.** Thông số đầu vào của trường hợp 1 testcase 1

| TÍN HIỆU ĐẦU VÀO            | GIÁ TRỊ       |
|-----------------------------|---------------|
| <code>input_re_addr</code>  | 32'd0001_5000 |
| <code>input_re_len</code>   | 8'b0000_0000  |
| <code>input_re_size</code>  | 3'b000        |
| <code>input_re_burst</code> | 2'b00         |
| <code>input_re_data</code>  | 32'h45679123  |

Với trường hợp này, dữ liệu đọc đầu vào là 45679123 (H), số bytes truyền là một bytes tương ứng với dữ liệu đọc là 23. Ta quan sát kết quả mô phỏng trong hình 4.16 dưới đây.



**Hình 4.16.** Dạng sóng trường hợp 1 testcase 1 của quá trình đọc

Tại 25ns, tín hiệu `arvalid` được tích cực mức cao cho phép AXI4 MASTER gửi thông tin của các tín hiệu thuộc kênh địa chỉ đọc. Nhưng tại thời điểm này, tín hiệu `arready` chưa được tích cực, nên AXI4 SLAVE chưa sẵn sàng nhận dữ liệu được gửi qua, thông tin của kênh địa chỉ đọc gửi từ AXI4 MASTER chưa được chấp nhận. Ngay tại thời điểm này, tín hiệu `rready` được tích cực, biểu thị AXI4 MASTER sẵn sàng nhận dữ liệu đọc gửi từ AXI4 SLAVE. Tại 35ns, tín hiệu `arready` tích cực, cơ chế bắt tay của kênh địa chỉ đọc được thực hiện, dữ liệu địa chỉ đọc được gửi từ AXI4 MASTER sang AXI4 SLAVE được chấp nhận. Tại 45ns, tín hiệu `arvalid` và `arready` không tích cực, biểu thị quá trình truyền thông tin của kênh địa chỉ đọc hoàn tất. Đồng thời, tín hiệu `rvalid` = 1, cơ chế bắt tay của kênh dữ liệu đọc được hình thành, dữ liệu đọc được gửi từ AXI4 SLAVE sang AXI4 MASTER được chấp nhận và tín hiệu `rresp` = '0', tương ứng với giá trị "OKAY" biểu thị cho việc gửi dữ liệu thành công. Trong trường hợp này chỉ gửi một beat dữ liệu đọc, do đó tín hiệu `rlast` = 1 biểu thị đã gửi dữ liệu đọc cuối cùng. Tại 55ns, các tín hiệu `rready`, `rvalid` và `rlast` cùng tích cực mức thấp, biểu thị quá trình đọc dữ liệu từ AXI4 SLAVE kết thúc.



Địa chỉ đọc của trường hợp này là 15000 (H), cơ chế truyền là kiểu FIXED, tương ứng với mỗi beat dữ liệu đọc vào cùng một ô nhớ, kiểm tra bộ nhớ của AXI4 MASTER sau quá trình đọc, kết quả như hình 4.17.

| Name                | Value      |
|---------------------|------------|
| mem_master[0:16383] | XXXXXXXXXX |
| [14999][31:0]       | XXXXXXXXXX |
| [15000][31:0]       | 00000023   |
| [15001][31:0]       | XXXXXXXXXX |

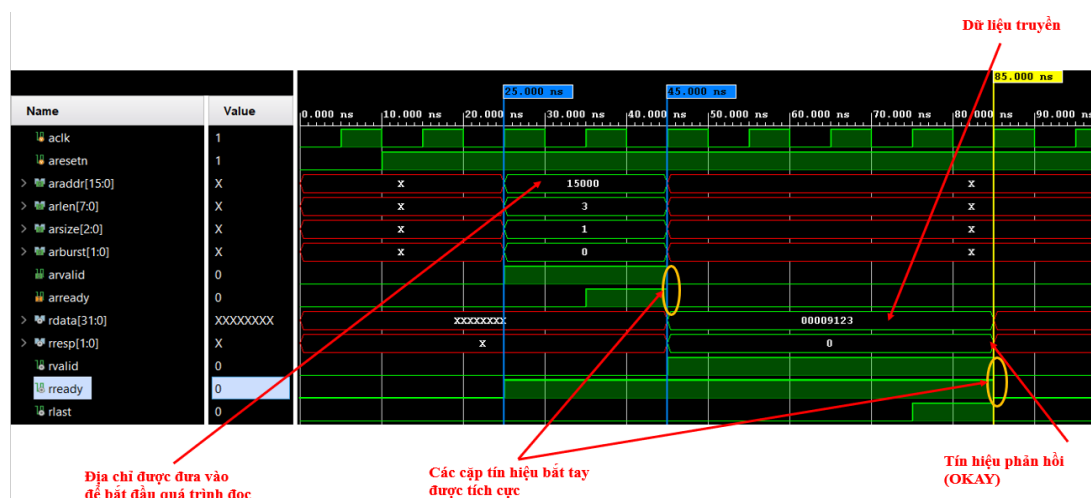
**Hình 4.17.** Bộ nhớ trường hợp 1 testcase 1 của AXI4 MASTER quá trình đọc

Trường hợp 2: AXI4 MASTER đọc dữ liệu từ AXI4 SLAVE theo cơ chế FIXED, truyền 4 beat và 2 bytes cho một giao dịch đọc. Các thông số đầu vào được ghi ở bảng 4.11 dưới đây.

**Bảng 4.11.** Thông số đầu vào của trường hợp 2 testcase 1

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_re_addr    | 32'd0001_5000 |
| input_re_len     | 8'd3          |
| input_re_size    | 3'b001        |
| input_re_burst   | 2'b00         |
| input_re_data    | 32'h45679123  |

Với trường hợp 2 này, dữ liệu đọc đầu vào là 45679123 (H), số bytes truyền là 2 bytes tương ứng với dữ liệu đọc là 9123. Ta quan sát kết quả mô phỏng trong hình 4.18 dưới đây.



**Hình 4.18.** Dạng sóng trường hợp 2 testcase 1 của quá trình đọc



Tại thời điểm 25ns, tín hiệu arvalid = 1, AXI4 MASTER gửi thông tin của kênh địa chỉ đọc. Tín hiệu rready tích cực biểu thị cho việc AXI4 MASTER sẵn sàng nhận dữ liệu đọc. Tại 35ns, tín hiệu aready tích cực, cơ chế bắt tay của kênh địa chỉ đọc được thực hiện, dữ liệu của kênh địa chỉ đọc được AXI4 SLAVE chấp thuận và bắt đầu gửi dữ liệu. Tại 45ns, tín hiệu aready và avalid không tiếp tích cực, thông tin địa chỉ được gửi hoàn tất. Tín hiệu rvalid = 1, cơ chế bắt tay của kênh dữ liệu đọc hình thành AXI4 SLAVE bắt đầu gửi dữ liệu đọc và AXI4 MASTER chấp nhận dữ liệu đọc. Tín hiệu rvalid và rready được giữ mức tích cực trong suốt quá trình đọc dữ liệu. Tại 75ns, tín hiệu rlast = 1, biểu thị beat cuối cùng của dữ liệu đọc được gửi đi. Tại 85ns, các tín hiệu rvalid, rready và rlast không tiếp tục tích cực, quá trình đọc dữ liệu hoàn tất.

Địa chỉ đọc của tệp kiểm tra này là 15000 (D), cơ chế truyền là kiểu FIXED, tương ứng với mỗi beat dữ liệu đọc vào cùng một ô nhớ, kiểm tra bộ nhớ của AXI4 MASTER sau quá trình đọc, kết quả như hình 4.19.

| Name                | Value         |
|---------------------|---------------|
| mem_master[0:16383] | XXXXXXXX,XXXX |
| > [14998][31:0]     | XXXXXXXX      |
| > [14999][31:0]     | XXXXXXXX      |
| > [15000][31:0]     | 00009123      |
| > [15001][31:0]     | XXXXXXXX      |

**Hình 4.19.** Bộ nhớ trường hợp 2 testcase 1 của AXI4 MASTER quá trình đọc

Qua các trường hợp của testcase 1, ta có thể thấy được rằng đối với kiểu cơ chế truyền FIXED, dù cho kích thước và độ dài của dữ liệu có thay đổi thì dữ liệu truyền vào bộ nhớ vẫn được đọc vào cùng một địa chỉ ô nhớ được yêu cầu đọc ban đầu.

#### 4.2.2.2 Testcase 2

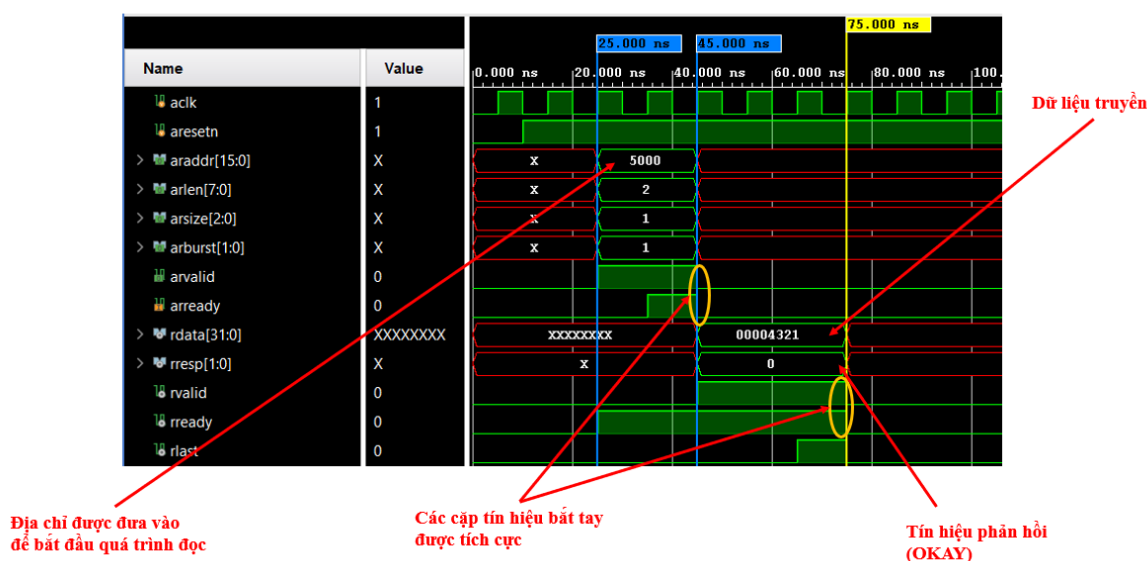
Trong testcase 2, tôi tiến hành kiểm tra quá trình đọc dữ liệu theo cơ chế INCR với nhiều tệp kiểm tra gồm các thông số địa chỉ, kích thước, độ dài và dữ liệu khác nhau.

*Trường hợp 1:* AXI4 MASTER đọc dữ liệu từ AXI4 SLAVE theo cơ chế FIXED, truyền 3 beat và 2 bytes cho một giao dịch đọc. Các thông số đầu vào được ghi ở bảng 4.12 dưới đây.

**Bảng 4.12.** Thông số đầu vào của trường hợp 1 testcase 2

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_re_addr    | 32'd0000_5000 |
| input_re_len     | 2'b10         |
| input_re_size    | 3'b001        |
| input_re_burst   | 2'b01         |
| input_re_data    | 32'h87654321  |

Với trường hợp này, dữ liệu đọc đầu vào là 87654321 (H), số bytes truyền là 2 bytes tương ứng với dữ liệu đọc là 4321. Ta quan sát kết quả mô phỏng trong hình 4.20 dưới đây.



**Hình 4.20.** Dạng sóng trường hợp 1 testcase 2 của quá trình đọc

Tại 25ns, tín hiệu `arvalid` được tích cực mức cao cho phép AXI4 MASTER gửi thông tin của các tín hiệu thuộc kênh địa chỉ đọc. Nhưng tại thời điểm này, tín hiệu `arready` chưa được tích cực, nên AXI4 SLAVE chưa sẵn sàng nhận dữ liệu được gửi qua, thông tin của kênh địa chỉ đọc gửi từ AXI4 MASTER chưa được chấp nhận. Ngay tại thời điểm này, tín hiệu `rready` được tích cực, biểu thị AXI4 MASTER sẵn sàng nhận dữ liệu đọc gửi từ AXI4 SLAVE. Tại 35ns, tín

hiệu *arready* tích cực, cơ chế bắt tay của kênh địa chỉ đọc được thực hiện, dữ liệu địa chỉ đọc được gửi từ AXI4 MASTER sang AXI4 SLAVE được chấp nhận. Tại 45ns, tín hiệu *arvalid* và *arready* không tích cực, biểu thị quá trình truyền thông tin của kênh địa chỉ đọc hoàn tất. Đồng thời, tín hiệu *rvalid* = 1, cơ chế bắt tay của kênh dữ liệu đọc được hình thành, dữ liệu đọc được gửi từ AXI4 SLAVE sang AXI4 MASTER được chấp nhận và tín hiệu *rresp* = '0', tương ứng với giá trị "OKAY" biểu thị cho việc gửi dữ liệu thành công. Trong trường hợp này chỉ gửi một beat dữ liệu đọc, do đó tín hiệu *rlast* = 1 biểu thị đã gửi dữ liệu đọc cuối cùng. Tại 75ns, các tín hiệu *rready*, *rvalid* và *rlast* cùng tích cực mức thấp, biểu thị quá trình đọc dữ liệu từ AXI4 SLAVE kết thúc.

Địa chỉ đọc của trường hợp này là 5000 (H), cơ chế truyền là kiểu INCR, tương ứng với mỗi beat dữ liệu đọc vào cùng một ô nhớ, kiểm tra bộ nhớ của AXI4 MASTER sau quá trình đọc, kết quả như hình 4.21.

| Name                | Value          |
|---------------------|----------------|
| mem_master[0:16383] | XXXXXXXX,XXXXX |
| > [4998][31:0]      | XXXXXXXX       |
| > [4999][31:0]      | XXXXXXXX       |
| > [5000][31:0]      | 00004321       |
| > [5001][31:0]      | XXXXXXXX       |
| > [5002][31:0]      | 00004321       |
| > [5003][31:0]      | XXXXXXXX       |
| > [5004][31:0]      | 00004321       |
| > [5005][31:0]      | XXXXXXXX       |
| > [5006][31:0]      | XXXXXXXX       |

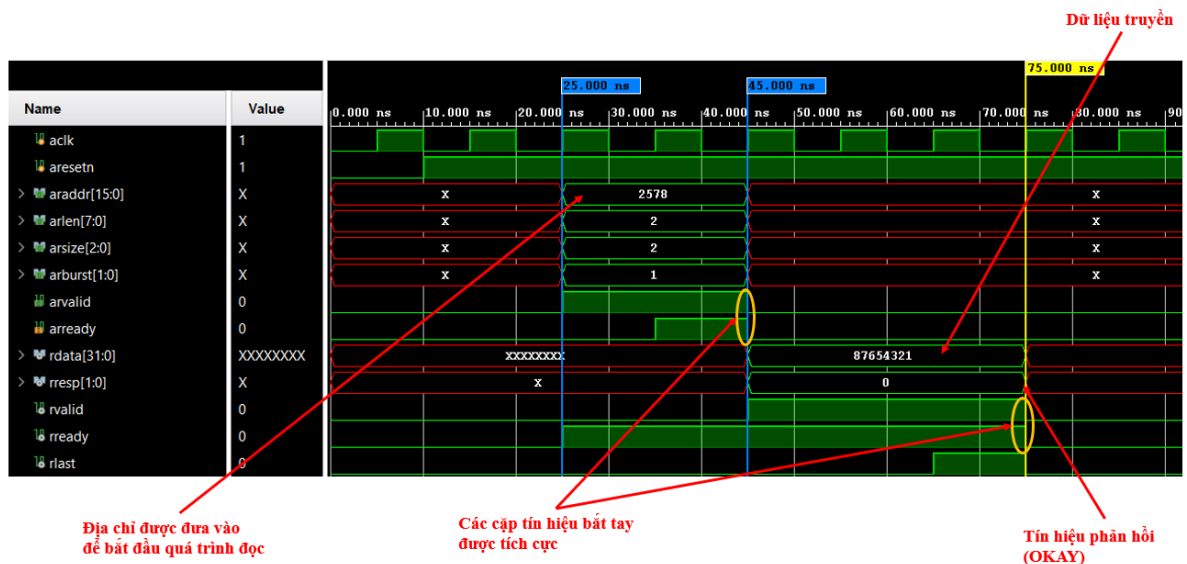
**Hình 4.21.** Bộ nhớ trường hợp 1 testcase 2 của AXI4 MASTER quá trình đọc

*Trường hợp 2:* AXI4 MASTER đọc dữ liệu từ AXI4 SLAVE theo cơ chế FIXED, truyền 3 beat và 4 bytes cho một giao dịch đọc. Các thông số đầu vào được ghi ở bảng 4.13 dưới đây.

**Bảng 4.13.** Thông số đầu vào của trường hợp 2 testcase 2

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_re_addr    | 32'd0000_2578 |
| input_re_len     | 2'b10         |
| input_re_size    | 3'b010        |
| input_re_burst   | 2'b01         |
| input_re_data    | 32'h87654321  |

Với trường hợp 2 này, dữ liệu đọc đầu vào là 2578 (H), số bytes truyền là 4 bytes tương ứng với dữ liệu ghi là 87654321. Ta quan sát kết quả mô phỏng trong hình 4.22 dưới đây.



**Hình 4.22.** Dạng sóng trường hợp 2 testcase 2 của quá trình đọc

Tại thời điểm 25ns, tín hiệu arvalid = 1, AXI4 MASTER gửi thông tin của kênh địa chỉ đọc. Tín hiệu rready tích cực biểu thị cho việc AXI4 MASTER sẵn sàng nhận dữ liệu đọc. Tại 35ns, tín hiệu arready tích cực, cơ chế bắt tay của kênh địa chỉ đọc được thực hiện, dữ liệu của kênh địa chỉ đọc được AXI4 SLAVE chấp thuận và bắt đầu gửi dữ liệu. Tại 45ns, tín hiệu aready và avalid không tiếp tích cực, thông tin địa chỉ được gửi hoàn tất. Tín hiệu rvalid = 1, cơ chế bắt tay của kênh dữ liệu đọc hình thành AXI4 SLAVE bắt đầu gửi dữ liệu đọc và AXI4 MASTER chấp nhận dữ liệu đọc. Tín hiệu rvalid và rready được giữ mức tích cực trong suốt quá trình đọc dữ liệu. Tại 65ns, tín hiệu rlast = 1, biểu thị beat

cuối cùng của dữ liệu đọc được gửi đi. Tại 75ns, các tín hiệu rvalid, rready và rlast không tiếp tục tích cực, quá trình đọc dữ liệu hoàn tất.

Địa chỉ đọc của trường hợp này là 5000 (D), cơ chế truyền là kiểu INCR, tương ứng với mỗi beat dữ liệu đọc vào địa chỉ ô nhớ tăng dần. Địa chỉ đọc vào ô nhớ bắt đầu giống với địa chỉ AXI4 MASTER gửi ban đầu, sau mỗi beat, địa chỉ đọc tăng thêm một giá trị tương ứng với số bytes được gửi. Kiểm tra bộ nhớ của AXI4 MASTER sau quá trình đọc, kết quả như hình 4.23.

| Name                | Value         |
|---------------------|---------------|
| mem_master[0:16383] | XXXXXXXX,XXXX |
| > [2577][31:0]      | XXXXXXXX      |
| > [2578][31:0]      | 87654321      |
| > [2579][31:0]      | XXXXXXXX      |
| > [2580][31:0]      | XXXXXXXX      |
| > [2581][31:0]      | XXXXXXXX      |
| > [2582][31:0]      | 87654321      |
| > [2583][31:0]      | XXXXXXXX      |
| > [2584][31:0]      | XXXXXXXX      |
| > [2585][31:0]      | XXXXXXXX      |
| > [2586][31:0]      | 87654321      |
| > [2587][31:0]      | XXXXXXXX      |
| > [2588][31:0]      | XXXXXXXX      |

**Hình 4.23.** Bộ nhớ trường hợp 2 testcase 2 của AXI4 MASTER quá trình đọc

Qua các trường hợp của testcase 2, ta có thể thấy được rằng đối với kiểu cơ chế truyền INCR, địa chỉ ô nhớ đầu tiên là địa chỉ được gửi từ AXI4 MASTER, sau mỗi beat thì địa chỉ đọc vào ô nhớ tăng dần và phụ thuộc vào kích thước và độ dài của dữ liệu gửi.

#### 4.2.2.3 Testcase 3

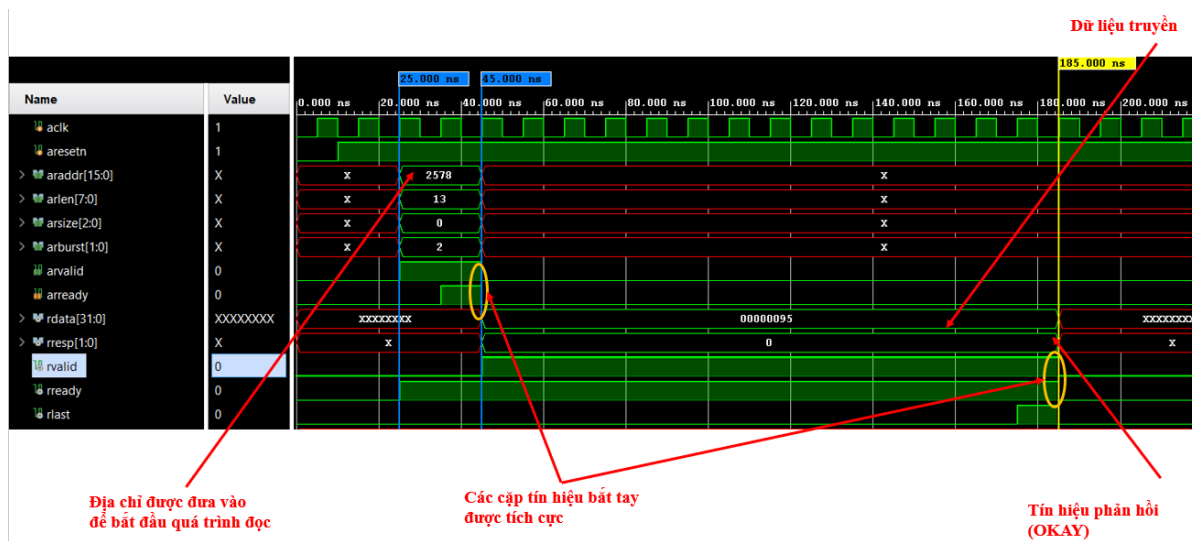
Trong testcase 3, tôi tiến hành kiểm tra quá trình đọc dữ liệu theo cơ chế WRAP với nhiều tệp kiểm tra gồm các thông số địa chỉ, kích thước, độ dài và dữ liệu khác nhau.

*Trường hợp 1:* AXI4 MASTER đọc dữ liệu từ AXI4 SLAVE theo cơ chế WRAP, truyền 14 beat và 1 bytes cho một giao dịch đọc. Các thông số đầu vào được ghi ở bảng 4.14 dưới đây.

**Bảng 4.14.** Thông số đầu vào của trường hợp 1 testcase 3

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_re_addr    | 32'd0000_2578 |
| input_re_len     | 4'b1101       |
| input_re_size    | 3'b000        |
| input_re_burst   | 2'b10         |
| input_re_data    | 32'h56217895  |

Với trường hợp này, dữ liệu đọc đầu vào là 56217895 (H), số bytes truyền là 1 bytes tương ứng với dữ liệu ghi là 95. Ta quan sát kết quả mô phỏng trong hình 4.24 dưới đây.



**Hình 4.24.** Dạng sóng trường hợp 1 testcase 3 của quá trình đọc

Tại thời điểm 25ns, tín hiệu arvalid = 1, AXI4 MASTER gửi thông tin của kênh địa chỉ đọc. Tín hiệu rready tích cực biểu thị cho việc AXI4 MASTER sẵn sàng nhận dữ liệu đọc. Tại 35ns, tín hiệu aready tích cực, cơ chế bắt tay của kênh địa chỉ đọc được thực hiện, dữ liệu của kênh địa chỉ đọc được AXI4 SLAVE chấp thuận và bắt đầu gửi dữ liệu. Tại 45ns, tín hiệu aready và avalid không tiếp tích cực, thông tin địa chỉ được gửi hoàn tất. Tín hiệu rvalid = 1, cơ chế bắt tay của kênh dữ liệu đọc hình thành AXI4 SLAVE bắt đầu gửi dữ liệu đọc và AXI4

MASTER chấp nhận dữ liệu đọc. Tín hiệu rvalid và rready được giữ mức tích cực trong suốt quá trình đọc dữ liệu. Tại 175ns, tín hiệu rlast = 1, biểu thị beat cuối cùng của dữ liệu đọc được gửi đi. Tại 185ns, các tín hiệu rvalid, rready và rlast không tiếp tục tích cực, quá trình đọc dữ liệu hoàn tất.

Địa chỉ đọc của trường hợp này là 2578 (D) và cơ chế truyền là kiểu WRAP. Địa chỉ ô nhớ bắt đầu đọc chính là địa chỉ đọc được gửi từ AXI4 MASTER tương ứng với giá trị 2578 (D) trong trường hợp này. Khi đọc đến trước địa chỉ cuộn, địa chỉ đọc sẽ được cuộn về ô nhớ có địa chỉ bằng với giá trị của wrap\_boundary và đọc tiếp dữ liệu cho đến khi đủ số beat yêu cầu. Trong trường hợp này, ta dùng công thức (2.4), (2.5) để tính toán được địa chỉ cuộn và địa chỉ tới hạn có giá trị như sau:

$$\text{wrap\_boundary} = \text{INT} \left[ \frac{2578}{(13+1) \times 2^0} \right] \times (13+1) \times 2^0 = 2576 \quad (2.12)$$

$$\text{address\_n} = 2576 + (13+1) \times 2^0 = 2590 \quad (2.13)$$

Kiểm tra bộ nhớ của AXI4 MASTER sau quá trình đọc, ta được dải ô nhớ lưu trữ các giá trị như hình 4.25.

| Name                | Value      |
|---------------------|------------|
| mem_master[0:16383] | XXXXXXXXXX |
| > [2573][31:0]      | XXXXXXXXXX |
| > [2574][31:0]      | XXXXXXXXXX |
| > [2575][31:0]      | XXXXXXXXXX |
| > [2576][31:0]      | 00000095   |
| > [2577][31:0]      | 00000095   |
| > [2578][31:0]      | 00000095   |
| > [2579][31:0]      | 00000095   |
| > [2580][31:0]      | 00000095   |
| > [2581][31:0]      | 00000095   |
| > [2582][31:0]      | 00000095   |
| > [2583][31:0]      | 00000095   |
| > [2584][31:0]      | 00000095   |
| > [2585][31:0]      | 00000095   |
| > [2586][31:0]      | 00000095   |
| > [2587][31:0]      | 00000095   |
| > [2588][31:0]      | 00000095   |
| > [2589][31:0]      | 00000095   |
| > [2590][31:0]      | XXXXXXXXXX |
| > [2591][31:0]      | XXXXXXXXXX |

Địa chỉ cuộn → [2576][31:0]

Địa chỉ ban đầu → [2578][31:0]

Vùng địa chỉ ghi dữ liệu đọc được → [2578][31:0] đến [2589][31:0]

Địa chỉ tới hạn → [2590][31:0]

**Hình 4.25.** Bộ nhớ trường hợp 1 testcase 3 của AXI4 MASTER quá trình đọc



Trường hợp 2: AXI4 MASTER đọc dữ liệu từ AXI4 SLAVE theo cơ chế WRAP, truyền 11 beat và 4 bytes cho một giao dịch đọc. Các thông số đầu vào được ghi ở bảng 4.15 dưới đây.

**Bảng 4.15.** Thông số đầu vào của trường hợp 2 testcase 3

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ       |
|------------------|---------------|
| input_re_addr    | 32'd0001_5000 |
| input_re_len     | 8'd1010       |
| input_re_size    | 3'b010        |
| input_re_burst   | 2'b10         |
| input_re_data    | 32'h56217895  |

Với trường hợp 2 này, dữ liệu ghi đầu vào là 56217895 (D), số bytes truyền là 4 bytes tương ứng với dữ liệu ghi là 56217895. Ta quan sát kết quả mô phỏng trong hình 4.26 dưới đây.



**Hình 4.26.** Dạng sóng trường hợp 2 testcase 3 của quá trình ghi

Tại thời điểm 25ns, tín hiệu `arvalid` = 1, AXI4 MASTER bắt đầu gửi thông tin của kênh địa chỉ đọc qua AXI4 SLAVE, nhưng lúc này, tín hiệu `arready` từ bên AXI4 SLAVE không tích cực, AXI4 SLAVE chưa sẵn sàng nhận dữ liệu. Tín hiệu `rready` tích cực biểu thị cho việc AXI4 MASTER sẵn sàng nhận dữ liệu đọc. Tại 35ns, tín hiệu `arready` tích cực, AXI4 SLAVE đã sẵn sàng nhận thông tin,



cơ chế bắt tay của kênh địa chỉ đọc được thực hiện, dữ liệu của kênh địa chỉ đọc được AXI4 SLAVE chấp thuận và bắt đầu gửi dữ liệu. Tại 45ns, tín hiệu *aready* và *avalid* không tiếp tích cực, thông tin địa chỉ được gửi hoàn tất. Đồng thời, AXI4 SLAVE bắt đầu gửi thông tin dữ liệu qua AXI4 MASTER, tín hiệu *rvalid* = 1, cơ chế bắt tay của kênh dữ liệu đọc hình thành AXI4 SLAVE bắt đầu gửi dữ liệu đọc và AXI4 MASTER chấp nhận dữ liệu đọc. Tín hiệu *rvalid* và *rready* được giữ mức tích cực trong suốt quá trình đọc dữ liệu. Tín hiệu *rresp* phản hồi dữ liệu đọc là 0, tương ứng với 'OKAY', đồng nghĩa với việc truyền và đọc dữ liệu từ AXI4 SLAVE sang AXI4 MASTER không có lỗi xảy ra đối với từng beat dữ liệu. Tại 145ns, tín hiệu *rlast* = 1, biểu thị beat cuối cùng của dữ liệu đọc được gửi đi. Tại 155ns, các tín hiệu *rvalid*, *rready* và *rlast* không tiếp tục tích cực, quá trình đọc dữ liệu hoàn tất. Trong suốt quá trình đọc và truyền dữ liệu, tín hiệu *rresp* vẫn luôn là 0, báo hiệu không có dữ liệu đọc nào bị lỗi.

Địa chỉ đọc của trường hợp này là 15000 (D) và cơ chế truyền là kiểu WRAP. Địa chỉ ô nhớ bắt đầu đọc chính là địa chỉ đọc được gửi từ AXI4 MASTER tương ứng với giá trị 15000 (D). Khi đọc đến trước địa chỉ cuộn, địa chỉ đọc sẽ được cuộn về ô nhớ có địa chỉ bằng với giá trị của *wrap\_boundary* và đọc tiếp dữ liệu cho đến khi đủ số beat yêu cầu. Nhưng địa chỉ đọc sẽ không được đọc liền mạch như ở trường hợp 1 mà sẽ có khoảng tăng với nguyên tắc tăng địa chỉ giống với cơ chế INCR. Trong trường hợp này, ta dùng công thức (2.4), (2.5) để tính toán được địa chỉ cuộn và địa chỉ tới hạn có giá trị như sau:

$$\text{wrap\_boundary} = \text{INT} \left[ \frac{15000}{(10+1) \times 2^2} \right] \times (10+1) \times 2^2 = 14960 \quad (2.14)$$

$$\text{address\_n} = 14960 + (10+1) \times 2^2 = 15004 \quad (2.15)$$

Kiểm tra bộ nhớ của AXI4 MASTER sau quá trình đọc, ta nhận thấy được địa chỉ ghi ban đầu là 15000, địa chỉ ghi kế tiếp là 15004 bằng với địa chỉ tới hạn. Do đó, sau khi ghi dữ liệu vào địa chỉ ô nhớ ban đầu, địa chỉ ghi sẽ được cuộn lại địa chỉ cuộn và tiếp tục ghi dữ liệu cho đến khi ghi đầy đủ số lượng beat yêu cầu. Sau khi hoàn tất quá trình đọc dữ liệu, ta được dải ô nhớ chứa các giá trị như hình 4.27.

| Name                | Value      |
|---------------------|------------|
| mem_master[0:16383] | XXXXXXXXXX |
| [14958][31:0]       | XXXXXXXXXX |
| [14959][31:0]       | XXXXXXXXXX |
| [14960][31:0]       | 56217895   |
| [14961][31:0]       | XXXXXXXXXX |
| [14962][31:0]       | XXXXXXXXXX |
| [14963][31:0]       | XXXXXXXXXX |
| [14964][31:0]       | 56217895   |
| [14965][31:0]       | XXXXXXXXXX |
| [14966][31:0]       | XXXXXXXXXX |
| [14967][31:0]       | XXXXXXXXXX |
| [14968][31:0]       | 56217895   |
| [14969][31:0]       | XXXXXXXXXX |
| [14970][31:0]       | XXXXXXXXXX |
| [14971][31:0]       | XXXXXXXXXX |
| [14972][31:0]       | 56217895   |
| [14973][31:0]       | XXXXXXXXXX |
| [14974][31:0]       | XXXXXXXXXX |
| [14975][31:0]       | XXXXXXXXXX |
| [14976][31:0]       | 56217895   |
| [14977][31:0]       | XXXXXXXXXX |
| [14978][31:0]       | XXXXXXXXXX |
| [14979][31:0]       | XXXXXXXXXX |
| [14980][31:0]       | 56217895   |
| [14981][31:0]       | XXXXXXXXXX |
| [14982][31:0]       | XXXXXXXXXX |
| [14983][31:0]       | XXXXXXXXXX |
| [14984][31:0]       | 56217895   |
| [14985][31:0]       | XXXXXXXXXX |
| [14986][31:0]       | XXXXXXXXXX |
| [14987][31:0]       | XXXXXXXXXX |

| Name                | Value      |
|---------------------|------------|
| mem_master[0:16383] | XXXXXXXXXX |
| [14988][31:0]       | 56217895   |
| [14989][31:0]       | XXXXXXXXXX |
| [14990][31:0]       | XXXXXXXXXX |
| [14991][31:0]       | XXXXXXXXXX |
| [14992][31:0]       | 56217895   |
| [14993][31:0]       | XXXXXXXXXX |
| [14994][31:0]       | XXXXXXXXXX |
| [14995][31:0]       | XXXXXXXXXX |
| [14996][31:0]       | 56217895   |
| [14997][31:0]       | XXXXXXXXXX |
| [14998][31:0]       | XXXXXXXXXX |
| [14999][31:0]       | XXXXXXXXXX |
| [15000][31:0]       | 56217895   |
| [15001][31:0]       | XXXXXXXXXX |
| [15002][31:0]       | XXXXXXXXXX |
| [15003][31:0]       | XXXXXXXXXX |
| [15004][31:0]       | XXXXXXXXXX |

**Hình 4.27.** Bộ nhớ trường hợp 2 testcase 3 của AXI MASTER quá trình đọc

Qua các trường hợp của testcase 3, ta có thể thấy được rằng đối với kiểu cơ chế truyền WRAP, địa chỉ ô nhớ đầu tiên là địa chỉ được gửi từ AXI4 MASTER, sau mỗi beat thì địa chỉ đọc vào ô nhớ tăng dần giống với kiểu cơ chế INCR. Đến khi địa chỉ đọc kế tiếp đạt đến địa chỉ tới hạn, địa chỉ đọc sẽ được cuộn về địa chỉ cuộn và tiếp tục đọc dữ liệu cho đến khi đủ số beat yêu cầu.

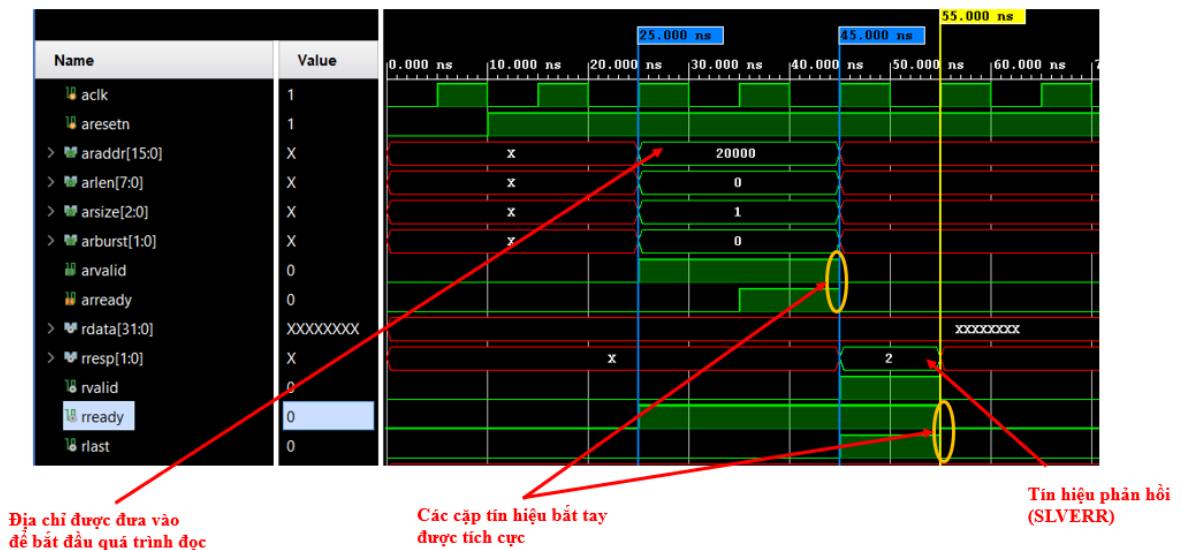
#### 4.2.2.4 Testcase 4

Trong testcase 4, tôi tiến hành kiểm tra quá trình đọc dữ liệu vượt quá giới hạn bộ nhớ cho phép. Bộ nhớ của AXI4 SLAVE là 16383 ô nhớ, ta cho địa chỉ bắt đầu đọc dữ liệu là 20000, vượt qua số lượng ô nhớ của AXI4 SLAVE. Các thông số đầu vào được ghi ở bảng 4.16 dưới đây.

**Bảng 4.16.** Thông số đầu vào của testcase 4

| TÍN HIỆU ĐẦU VÀO | GIÁ TRỊ      |
|------------------|--------------|
| input_re_addr    | 15'd20000    |
| input_re_len     | 8'd0         |
| input_re_size    | 3'b001       |
| input_re_burst   | 2'b00        |
| input_re_data    | 32'h12345678 |

Với trường hợp này, dữ liệu đọc đầu vào là 12345678 (H), số bytes truyền là 1 bytes tương ứng với dữ liệu đọc là 23. Ta quan sát kết quả mô phỏng trong hình 4.28 dưới đây.



**Hình 4.28.** Dạng sóng testcase 4 của quá trình đọc

Tại 25ns, tín hiệu arvalid tích cực, AXI4 MASTER gửi thông tin địa chỉ đọc. Đồng thời, tín hiệu rready = 1, cho phép AXI4 MASTER nhận thông tin dữ liệu đọc từ AXI4 SLAVE. Tại 35ns, tín hiệu arready tích cực, cơ chế bắt tay của kênh địa chỉ đọc hình thành, AXI4 MASTER gửi thông tin địa chỉ đọc và AXI4 SLAVE chấp nhận thông tin. Tại 45ns, tín hiệu arready và arvalid không tiếp tục tích cực, quá trình gửi và nhận thông tin kênh địa chỉ đọc hoàn tất. Trong tại thời điểm đó, AXI4 SLAVE gửi tín hiệu phản hồi rresp là 2, tương ứng với ‘SLVERR’, báo hiệu phát sinh lỗi. Tín hiệu rvalid được tích cực, hình thành cơ chế bắt tay của kênh dữ liệu đọc, nhưng do địa chỉ đọc vượt quá ô nhớ cho phép, do đó tín hiệu rlast = 1 để báo hiệu rằng đã đọc xong dữ liệu đọc cuối cùng. Tại 55ns, các tín hiệu rvalid, rready và rlast không tiếp tục tích cực, quá trình đọc dữ liệu hoàn tất.

## CHƯƠNG 5

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 5.1 KẾT LUẬN

Đề tài “Thiết kế mô hình AXI bus sử dụng ngôn ngữ Verilog” đã được thực hiện với mục tiêu xây dựng mô hình giao tiếp AXI4 bus theo cấu trúc điểm – điểm giữa một Master và một Slave. Thông qua kết quả đạt được từ mô phỏng, thiết kế đã thực thi tốt các yêu cầu:

- Xây dựng thành công mô hình điểm – điểm với một khối AXI4 MASTER và một khối AXI4 SLAVE, tuân thủ đầy đủ đặc tả giao thức AMBA AXI4 của hãng ARM.
- Mô hình hoạt động đúng đặc tả AXI4 với tần số hoạt động là 100MHz, hỗ trợ các cơ chế truyền dữ liệu khác nhau. Các testcase đa dạng đã được triển khai, bao gồm các trường hợp với kích thước và độ dài dữ liệu khác nhau, cũng như kiểm tra lỗi khi địa chỉ vượt giới hạn bộ nhớ.

Bên cạnh đó, thiết kế vẫn còn nhiều hạn chế:

- Các testcasae chỉ xoay quanh việc kiểm tra các hoạt động trao đổi dữ liệu đơn giản.
- Thiết kế chỉ dừng lại ở mô phỏng trên phần mềm, chưa thể áp dụng lên các mạch thực tế.
- Mô hình chỉ tập trung vào cấu trúc điểm – điểm đơn giản, chưa hỗ trợ nhiều Master và Slave hoặc các tính năng nâng cao của AXI4 như giao dịch không theo thứ tự hay các cơ chế bảo mật.

## 5.2 HƯỚNG PHÁT TRIỂN

Dựa trên kết quả đạt được và các hạn chế của đề tài, người thực hiện đề xuất một số hướng phát triển trong tương lai cho đề tài này như sau:

- Thiết kế một hệ thống AXI4 với mô hình đa điểm hỗ trợ nhiều Master và Slave, cho phép cấu hình địa chỉ linh hoạt và kiểm tra hiệu năng trong các hệ thống SoC phức tạp hơn.
- Tích hợp thêm các tính năng nâng cao của AXI4, hỗ trợ cho các giao dịch không theo thứ tự, chèn thêm các tầng thanh ghi để tối ưu hóa định thời và triển khai các cơ chế bảo mật phần cứng phù hợp với các ứng dụng SoC hiện đại.

## TÀI LIỆU THAM KHẢO

- [1] B. B. R. a. T. G. P. M. S. P. Reddy, "A Synthesizable Design of AMBA-AXI Protocol for SoC Integration," *International Journal of Engineering Inventions*, tập 1, số 3, pp. 1-6, 2021.
- [2] S. S. Math, M. R. Bharamagoudra, "Design of AMBA AXI4 protocol for System-on-Chip communication," *International Journal of Communication Networks and Security*, vol. 1, no. 4, pp. 1-6, 2012.
- [3] B. Gadgay, "Design and Implementation of AMBA AXI 4.0 Master for High Performance SoCs," *International Journal of Innovative Science, Engineering & Technology (IJISSET)*, vol. 3, no. 7, pp. 52-56, 2016.
- [4] N. Q. A. Tuan, N. V. Tuan, V. P. M. Dat, V. B. Huy, T. Q. Phuc, N. N. Lam, "A Comparison of the Wishbone and Amba Axi Bus Architecture," *Journal of Technology and Education Science*, vol. 17, no. Special Issue 02, Aug 2022.
- [5] Synopsys, ""What is a System on Chip (SoC)?," Synopsys Blog, [Online]. Available: <https://www.synopsys.com/blogs/chip-design/system-on-chip.html>. [Accessed 18 May 2025].
- [6] N. Quan, "Background: SoC (System on Chip)," Nguyen Quan ICD, Aug 2018. [Online]. Available: <https://nguyenquanica.blogspot.com/2018/05/background-soc-system-on-chip.html>. [Accessed 2025 May 18].
- [7] Ansys, "What is a system on a chip (SoC)?," Ansys Blog, 31 Oct 2023. [Online]. Available: <https://www.ansys.com/blog/what-is-system-on-a-chip>. [Accessed 15 Jun 2025].
- [8] N. Quan, "Bus bài 1: Giao thức AMBA AXI," Nguyen Quan ICD, Aug 2018. [Online]. Available: <https://nguyenquanica.blogspot.com/>. [Accessed 18 May 2018].
- [9] ARM Limited , "AMBA® AXI Protocol Specification," *ARM IHI 0051B*, Sep 2023.

- [10] N. T. Anh, "Giáo trình ngôn ngữ mô tả phần cứng," Academia.edu, 2017. [Online]. Available: [https://www.academia.edu/34332903/Giáo\\_trình\\_ngôn\\_ngữ\\_mô\\_tả\\_phần\\_cứng](https://www.academia.edu/34332903/Giáo_trình_ngôn_ngữ_mô_tả_phần_cứng). [Accessed 18 May 2025].
- [11] T. Dung, "Các mức mô tả trừu tượng của Verilog," Semicon Việt Nam, Dec 2015. [Online]. Available: <https://www.semiconvn.com/home/hoc-thiet-ke-vi-mach/bai-hc-vi-mch/9782-cac-muc-mo-ta-truu-tuong-cua-verilog.html>. [Accessed 30 May 2025].
- [12] Xilinx, "Vivado Design Suite User Guide: Design Flows Overview," *UG892 (v2022.1)*, 20 Apr 2022.

## PHỤ LỤC

Các mã nguồn của hệ thống và các tài liệu liên quan được đặt trong đường dẫn sau: [https://github.com/ngdkvy/DA1\\_AXI4](https://github.com/ngdkvy/DA1_AXI4)

Link video mô phỏng: <https://youtu.be/jLCSUQPqZ38>

Link github cá nhân: <https://github.com/ngdkvy>