

ĐẠI HỌC BÁCH KHOA HÀ NỘI  
TRƯỜNG ĐIỆN - ĐIỆN TỬ



ĐỒ ÁN  
**TỐT NGHIỆP ĐẠI HỌC**

Đề tài:

**NGHIÊN CỨU, THIẾT KẾ VÀ KIỂM THỬ GIAO  
THỨC BUS TRUYỀN THÔNG AHB**

Sinh viên thực hiện: TRẦN ANH TIÊN

Lớp ĐT07 – K63

Giảng viên hướng dẫn: TS. TRẦN THỊ NGỌC LAN

Hà Nội, 7-2023

ĐẠI HỌC BÁCH KHOA HÀ NỘI  
TRƯỜNG ĐIỆN - ĐIỆN TỬ



ĐỒ ÁN  
**TỐT NGHIỆP ĐẠI HỌC**

Đề tài:

**NGHIÊN CỨU, THIẾT KẾ VÀ KIỂM THỦ GIAO  
THÚC BUS TRUYỀN THÔNG AHB**

Sinh viên thực hiện: TRẦN ANH TIỀN

Lớp ĐT07 – K63

Giảng viên hướng dẫn: TS. TRẦN THỊ NGỌC LAN

Cán bộ phản biện:

Hà Nội, 7-2023

TRƯỜNG ĐIỆN – ĐIỆN TỬ**ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP****(DÀNH CHO CÁN BỘ HƯỚNG DẪN)**

Tên đề tài: Nghiên cứu, thiết kế và kiểm thử giao thức bus truyền thông AHB

Họ tên SV: Trần Anh Tiên

MSSV: 20182816

Cán bộ hướng dẫn: TS. Trần Thị Ngọc Lan

STT	Tiêu chí (Điểm tối đa)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	<b>Thái độ làm việc (2,5 điểm)</b>	Nghiêm túc, tích cực và chủ động trong quá trình làm ĐATN  Hoàn thành đầy đủ và đúng tiến độ các nội dung được GVHD giao	
2	<b>Kỹ năng viết quyển ĐATN (2 điểm)</b>	Trình bày đúng mẫu quy định, bố cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đê cập đến trong đồ án, có cẩn lè, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.  Kỹ năng diễn đạt, phân tích, giải thích, lập luận: Cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.	
3	<b>Nội dung và kết quả đạt được (5 điểm)</b>	Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.  Nội dung và kết quả được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.  Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/tính sáng tạo trong nội dung và kết quả đồ án.	
4	<b>Điểm thành tích (1 điểm)</b>	Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. <b>(1 điểm)</b>  Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. <b>(0,5 điểm)</b>	
		<b>Điểm tổng các tiêu chí:</b>	
		<b>Điểm hướng dẫn:</b>	

**Cán bộ hướng dẫn**

(Ký và ghi rõ họ tên)

**ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**TRƯỜNG ĐIỀN – ĐIỀN TỬ**

**ĐÁNH GIÁ ĐỒ ÁN TỐT NGHIỆP  
(DÀNH CHO CÁN BỘ PHẢN BIỆN)**

Tên đề tài: Nghiên cứu, thiết kế và kiểm thử giao thức bus truyền thông AHB

Họ tên SV: Trần Anh Tiên

MSSV: 20182816

Cán bộ phản biện: .....

STT	Tiêu chí (Điểm tối đa)	Hướng dẫn đánh giá tiêu chí	Điểm tiêu chí
1	<b>Trình bày quyển ĐATN (4 điểm)</b>	<p>Đồ án trình bày đúng mẫu quy định, bô cục các chương logic và hợp lý: Bảng biểu, hình ảnh rõ ràng, có tiêu đề, được đánh số thứ tự và được giải thích hay đề cập đến trong đồ án, có cẩn lè, dấu cách sau dấu chấm, dấu phẩy, có mở đầu chương và kết luận chương, có liệt kê tài liệu tham khảo và có trích dẫn, v.v.</p> <p>Kỹ năng diễn đạt, phân tích, giải thích, lập luận: cấu trúc câu rõ ràng, văn phong khoa học, lập luận logic và có cơ sở, thuật ngữ chuyên ngành phù hợp, v.v.</p>	
2	<b>Nội dung và kết quả đạt được (5,5 điểm)</b>	<p>Nêu rõ tính cấp thiết, ý nghĩa khoa học và thực tiễn của đề tài, các vấn đề và các giả thuyết, phạm vi ứng dụng của đề tài. Thực hiện đầy đủ quy trình nghiên cứu: Đặt vấn đề, mục tiêu đề ra, phương pháp nghiên cứu/ giải quyết vấn đề, kết quả đạt được, đánh giá và kết luận.</p> <p>Nội dung và kết quả đạt được trình bày một cách logic và hợp lý, được phân tích và đánh giá thỏa đáng. Biện luận phân tích kết quả mô phỏng/ phần mềm/ thực nghiệm, so sánh kết quả đạt được với kết quả trước đó có liên quan.</p> <p>Chỉ rõ phù hợp giữa kết quả đạt được và mục tiêu ban đầu đề ra đồng thời cung cấp lập luận để đề xuất hướng giải quyết có thể thực hiện trong tương lai. Hàm lượng khoa học/ độ phức tạp cao, có tính mới/ tính sáng tạo trong nội dung và kết quả đồ án.</p>	
3	<b>Điểm thành tích (1 điểm)</b>	<p>Có bài báo KH được đăng hoặc chấp nhận đăng/ đạt giải SV NCKH giải 3 cấp Trường trở lên/ Các giải thưởng khoa học trong nước, quốc tế từ giải 3 trở lên/ Có đăng ký bằng phát minh sáng chế. (1 điểm)</p> <p>Được báo cáo tại hội đồng cấp Trường trong hội nghị SV NCKH nhưng không đạt giải từ giải 3 trở lên/ Đạt giải khuyến khích trong cuộc thi khoa học trong nước, quốc tế/ Kết quả đồ án là sản phẩm ứng dụng có tính hoàn thiện cao, yêu cầu khối lượng thực hiện lớn. (0,5 điểm)</p>	
		<b>Điểm tổng các tiêu chí:</b>	
		<b>Điểm phản biện:</b>	

**Cán bộ phản biện**

(Ký và ghi rõ họ tên)

## LỜI NÓI ĐẦU

Cùng với sự phát triển của xã hội hiện đại và tiến bộ khoa học kỹ thuật, lĩnh vực Điện – Điện tử đã và đang trở nên ngày một tiến bộ và văn minh hơn trong mọi khía cạnh. Trong lĩnh vực này, vi mạch và vi mạch tích hợp (IC) ngày càng được phát triển với nhiều chức năng tích hợp trong một chip duy nhất, mang lại kích thước nhỏ gọn và tiện lợi. Tích hợp nhiều khối chức năng trên một hệ thống đơn chip đặt ra yêu cầu quan trọng về việc giải quyết vấn đề truyền thông giữa các khối chức năng này. Để đáp ứng yêu cầu này, đã có nhiều phương thức truyền thông giữa các chip được phát triển, bao gồm truyền thông theo dạng kết nối điểm tới điểm, bus truyền thông và các phương thức tiên tiến hơn như "Mạng trên chip" (Network on Chip). Tuy nhiên, phương thức truyền thông bus vẫn đang được sử dụng phổ biến do tính đơn giản và thuận tiện của nó. Nó cho phép truyền thông giữa các khối chức năng trên chip dễ dàng và hiệu quả. Hơn nữa, các thế hệ bus tiên tiến đã được phát triển, có khả năng đáp ứng yêu cầu truyền thông tốc độ cao và hỗ trợ truyền thông giữa nhiều lõi xử lý khác nhau.

Một trong những giao thức truyền thông bus phổ biến là "Advanced High-performance Bus" (AHB). Giao thức AHB là một giao thức bus tiêu chuẩn được phát triển bởi ARM (Advanced RISC Machines) và được sử dụng rộng rãi trong việc kết nối các thành phần trong hệ thống tích hợp trên chip (SoC). Giao thức AHB cung cấp môi trường truyền thông đáng tin cậy và hiệu quả cho việc trao đổi dữ liệu giữa các thành phần trong hệ thống. Nó hỗ trợ nhiều tính năng như truyền thông đa tốc độ, truyền thông ưu tiên, và kiểm soát quyền truy cập đến các tài nguyên trên bus.

Đề tài nghiên cứu về giao thức AHB sẽ tập trung vào khảo sát, phân tích và đánh giá hiệu năng của giao thức này trong việc truyền thông giữa các thành phần trên chip SoC. Nghiên cứu này sẽ giúp cải thiện và tối ưu hóa việc truyền thông trong các hệ thống tích hợp ngày càng phức tạp, đảm bảo sự liên kết mạnh mẽ và hiệu quả giữa các khối chức năng trên chip. Trong thời gian nghiên cứu, tìm hiểu và thực hiện đồ án dựa vào những kiến thức đã được học ở trường, qua một số tài liệu liên quan, cùng với sự giúp đỡ tận tình của cô Trần Thị Ngọc Lan, em đã hoàn thành thành công việc đúng tiến độ được đề ra. Em xin chân thành cảm ơn cô!



## **LỜI CAM ĐOAN**

Tôi là Trần Anh Tiên, mã số sinh viên 20182816, sinh viên lớp Điện tử 07, khóa K63. Người hướng dẫn là TS. Trần Thị Ngọc Lan. Tôi xin cam đoan toàn bộ nội dung được trình bày trong đồ án Nghiên cứu, thiết kế và kiểm thử bus truyền thông AHB là kết quả quá trình tìm hiểu và nghiên cứu của tôi. Các dữ liệu được nêu trong đồ án là hoàn toàn trung thực, phản ánh đúng kết quả đo đạc thực tế. Mọi thông tin trích dẫn đều tuân thủ các quy định về sở hữu trí tuệ; các tài liệu tham khảo được liệt kê rõ ràng. Tôi xin chịu hoàn toàn trách nhiệm với những nội dung được viết trong đồ án này.

Hà nội, ngày 10 tháng 08 năm 2023

Người cam đoan

Trần Anh Tiên

# MỤC LỤC

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT .....	i
DANH MỤC HÌNH VẼ .....	iii
DANH MỤC BẢNG BIÊU .....	v
TÓM TẮT ĐÒ ÁN .....	vi
PROJECT SUMMARY .....	vii
CHƯƠNG 1. HỆ THỐNG TRÊN VI MẠCH VÀ VẤN ĐỀ TRUYỀN THÔNG TRÊN VI MẠCH .....	1
1.1 <i>Hệ thống trên vi mạch – lịch sử phát triển</i> .....	1
1.2 <i>Quy trình thực hiện thiết kế hệ thống trên chip</i> .....	5
1.2.1    Specification .....	6
1.2.2    RTL Coding .....	7
1.2.3    RTL Simulation .....	8
1.2.4    Synthesis .....	8
1.2.5    Design For Test.....	9
1.2.6    Layout .....	9
1.2.7    Static Timing Analysis .....	10
1.2.8    Equivalence .....	10
1.2.9    Gate simulation.....	10
1.3 <i>Tổng quan phương pháp thiết kế hệ thống</i> .....	11
1.4 <i>Vấn đề truyền thông trên chip</i> .....	12
1.5 <i>Kết luận chương</i> .....	15
CHƯƠNG 2. HỆ THỐNG BUS AMBA AHB .....	16
2.1 <i>Tổng quan về hệ thống bus tốc độ cao AMBA AHB</i> .....	16
2.2 <i>Mô hình cấu trúc và hoạt động chung của bus AMBA AHB</i> .....	20
2.2.1    Mô hình một hệ thống AMBA AHB .....	20
2.2.2    Quy trình hoạt động của hệ thống AMBA AHB .....	21
2.3 <i>Các chế độ truyền cơ bản của bus AMBA AHB</i> .....	22

2.3.1 Quá trình truyền cơ bản, không có trạng thái đợi .....	22
2.3.2 Quá trình truyền cơ bản có trạng thái đợi.....	23
2.3.3 Quá trình truyền khối (burst transfer) .....	23
2.3.4 Thông báo quá trình truyền của bus chủ .....	27
<b>2.4 Kết luận chương .....</b>	<b>28</b>
<b>CHƯƠNG 3. NGÔN NGỮ VERILOG, SYSTEM VERILOG VÀ CÔNG CỤ EDA .....</b>	<b>30</b>
<b>3.1 Ngôn ngữ Verilog.....</b>	<b>30</b>
3.1.1 Khái niệm về ngôn ngữ Verilog.....	30
3.1.2 Các mức mô tả của Verilog.....	30
<b>3.2 Ngôn ngữ System Verilog .....</b>	<b>31</b>
3.2.1 Khái niệm về ngôn ngữ System Verilog.....	31
3.2.2 Điểm nâng cấp cơ bản của System Verilog so với Verilog.....	32
<b>3.3 Công cụ EDA.....</b>	<b>33</b>
3.3.1 EDA Playground .....	33
3.3.2 Ưu điểm của EDA Playground .....	33
<b>3.4 Kết luận chương .....</b>	<b>34</b>
<b>CHƯƠNG 4. XÂY DỰNG MÔ HÌNH HÓA HỆ THỐNG AHB .....</b>	<b>35</b>
<b>4.1 Bài toán thiết kế.....</b>	<b>35</b>
4.1.1 Mô hình đề xuất của bài toán.....	35
4.1.2 Đường dữ liệu và đường địa chỉ trên mô hình.....	37
<b>4.2 Mô hình giao tiếp giữa bus chủ và lõi IP.....</b>	<b>39</b>
<b>4.3 Bus chủ.....</b>	<b>41</b>
4.3.1 Các tín hiệu trên bus chủ .....	41
4.3.2 Máy trạng thái mô tả hoạt động của bus chủ.....	43
<b>4.4 Bus tớ.....</b>	<b>46</b>
4.4.1 Các tín hiệu trên bus tớ .....	46
4.4.2 Hoạt động của bus tớ.....	47
4.4.3 Phân loại bus tớ và vùng địa chỉ hoạt động các bus tớ .....	49
<b>4.5 Bộ giải mã địa chỉ và bộ phân kênh tín hiệu .....</b>	<b>50</b>
4.5.1 Bộ giải mã địa chỉ .....	50
4.5.2 Bộ phân kênh tín hiệu từ bus tớ đến bus chủ.....	52
<b>4.6 Kết luận chương .....</b>	<b>54</b>

<b>CHƯƠNG 5. MÔ PHỎNG VÀ KIỂM THỬ MÔ HÌNH.....</b>	<b>55</b>
<i>5.1 Phương pháp kiểm tra đánh giá mô hình.....</i>	<i>55</i>
<i>5.2 Mô phỏng mô hình .....</i>	<i>55</i>
5.2.1 Mô phỏng hoạt động truyền đơn không có trạng thái đợi .....	55
5.2.2 Mô phỏng hoạt động truyền đơn có trạng thái đợi .....	57
5.2.3 Mô phỏng hoạt động truyền với vùng địa chỉ bus từ mặc định .....	57
5.2.4 Mô phỏng hoạt động của mô hình khi có tín hiệu reset.....	58
5.2.5 Mô phỏng hoạt động của quá trình truyền khôi.....	58
<i>5.3 Kế hoạch kiểm thử.....</i>	<i>62</i>
<i>5.4 Kết quả thu được và đánh giá hệ thống.....</i>	<i>64</i>
5.4.1 Kết quả thực hiện testcase .....	64
5.4.2 Kết quả coverage code .....	65
<i>5.5 Đánh giá chung về kết quả thu được.....</i>	<i>66</i>
<b>KẾT LUẬN .....</b>	<b>67</b>
<i>Kết luận chung .....</i>	<i>67</i>
<i>Hướng phát triển.....</i>	<i>67</i>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>68</b>

## DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

Tùy viết tắt	Viết đầy đủ	Nghĩa
ADC	Analog - Digital Converter	Bộ Chuyển Đổi Analog - Số
AHB	Advanced High-performance Bus	Đường truyền hiệu suất cao
AMBA	Advanced Microcontroller Bus Architecture	Đường truyền vi điều khiển nâng cao
APB	Advanced Peripheral Bus	Đường truyền ngoại vi nâng cao
ASIC	Application-Specific Integrated Circuit	Mạch tích hợp ứng dụng cụ thể
AXI	Advanced eXtensible Interface	Giao diện mở rộng nâng cao
CPU	Central Processing Unit	Bộ xử lý trung tâm
CTS	Clock Tree Synthesis	Tổng hợp xung đồng hồ
DFT	Design For Test	Thiết kế để thử nghiệm
DRC	Design Rule Check	Kiểm tra quy tắc thiết kế
IC	Integrated Circuit	Mạch tích hợp
IO	Input output	Đầu vào, đầu ra
IP	Intellectual Property	Sở hữu trí tuệ
NoC	Network on Chip	Mạng trên chip
RAM	Random - Access Memory	Bộ nhớ truy cập ngẫu nhiên
ROM	Read - Only Memory	Bộ nhớ chỉ đọc
RTL	Register Transfer Level	Đăng ký mức chuyển
SDC	Synopsys Design Constraints	Các ràng buộc về thiết kế

SDF	Standard Delay Format	Định dạng trễ tiêu chuẩn
SoC	System on Chip	Hệ thống trên chip
SPEF	Standard Parasitic Extraction Format	Định dạng trích xuất tiêu chuẩn
SPI	Serial Peripheral Interface	Chuẩn truyền thông nối tiếp đồng bộ
STA	Static Timing Analysis	Phân tích thời gian tĩnh
UART	Universal Asynchronous ReceiverTransmitter	Truyền thông bất đồng bộ

## DANH MỤC HÌNH VẼ

Hình 1.1 Kiến trúc cơ bản của một hệ thống trên vi mạch (System on chip) .....	3
Hình 1.2 Cấu trúc của vi điều khiển ARM .....	4
Hình 1.3 Sơ đồ khái quát trình thiết kế chip.....	6
Hình 1.4 Mô hình 2 phương pháp thiết kế top-down và bottom-up .....	12
Hình 1.5 Mô hình kết nối điểm-tới-điểm.....	13
Hình 2.1 Các chuẩn kết nối của ABMA và sự phát triển các giao thức .....	17
Hình 2.2 Mô hình vi xử lý sử dụng bus AHB. ....	18
Hình 2.3 Sơ đồ khái niệm nền tảng IP PIP-AMBA của CAST và SOC Solutions. ....	20
Hình 2.4 Sơ đồ kết nối các thành phần trong bus AHB [11] .....	20
Hình 2.5 Quá trình truyền cơ bản không trạng thái đợi trên bus AHB [11].....	23
Hình 2.6 Quá trình truyền cơ bản có trạng thái đợi trên bus AHB [11].....	23
Hình 2.7 Quá trình truyền khói tăng với 4 nhịp [11] .....	25
Hình 2.8 Quá trình truyền khói cuộn với 4 nhịp [11].....	26
Hình 2.9 Quá trình truyền khói cuộn với 8 nhịp [11].....	27
Hình 2.10 Quá trình truyền khói tăng với 8 nhịp [11].....	27
Hình 4.1 Mô hình hệ thống bus AHB được đề xuất.....	37
Hình 4.2 Đường dẫn tín hiệu địa chỉ trong mô hình .....	38
Hình 4.3 Đường dẫn tín hiệu dữ liệu ghi.....	38
Hình 4.4 Đường dẫn tín hiệu dữ liệu đọc .....	39
Hình 4.5 Các tín hiệu giao tiếp giữa bus chủ và lõi IP .....	40
Hình 4.6 Các tín hiệu trên bus chủ .....	41
Hình 4.7 Sơ đồ máy trạng thái của bus chủ. ....	43
Hình 4.8 Biểu diễn các trạng thái tương ứng với quá trình truyền khói .....	46
Hình 4.9 Các tín hiệu giao tiếp trên bus tớ .....	46
Hình 4.10 Sơ đồ minh họa hoạt động trong bus tớ .....	48
Hình 4.11 Vùng địa chỉ của các bus tớ .....	50
Hình 4.12 Các tín hiệu trên bộ giải mã địa chỉ .....	51

Hình 4.13 Kết quả tín hiệu lựa chọn bus tớ trả về từ bộ giải mã địa chỉ .....	52
Hình 4.14 Các chân tín hiệu trên bộ phân khenh .....	53
Hình 5.1 Sơ đồ khôi quá trình mô phỏng kiểm tra hệ thống.....	55
Hình 5.2 Quá trình ghi đơn không có trạng thái đợi ở bus tớ 0 .....	56
Hình 5.3 Quá trình đọc đơn không có trạng thái đợi .....	56
Hình 5.4 Quá trình ghi đơn có trạng thái đợi vào bus tớ 2 .....	57
Hình 5.5 Quá trình ghi vào vùng địa bus tớ mặc định.....	57
Hình 5.6 Quá trình truyền khi có tín hiệu reset .....	58
Hình 5.7 Quá trình ghi khôi tăng 4 nhịp không chờ.....	59
Hình 5.8 Quá trình ghi khôi cuộn 4 nhịp có trạng thái chờ .....	60
Hình 5.9 Quá trình truyền khôi cuộn 8 nhịp .....	61
Hình 5.10 Quá trình truyền khôi cuộn 16 nhịp.....	61
Hình 5.11 Kết quả thu được trên mô phỏng.....	64
Hình 5.12 Kết quả của mô phỏng khi được thực thi.....	65
Hình 5.13 Kết quả coverage code tổng quan hệ thống .....	66

## **DANH MỤC BẢNG BIỂU**

Bảng 2.1 Các kiểu truyền khói được bus AHB hỗ trợ.....	24
Bảng 2.2 Bảng độ rộng kiểu dữ liệu truyền .....	25
Bảng 2.3 Các trạng thái của bus chủ tương với các kiểu truyền.....	28
Bảng 4.1 Bảng giá trị mã hóa các trạng thái .....	45
Bảng 5.1 Mô tả các chức năng cần kiểm thử của hệ thống .....	63

# TÓM TẮT ĐỒ ÁN

Mục tiêu của đề tài: Nghiên cứu, thiết kế và thực hiện kiểm thử bus truyền thông AHB. Công việc của đồ án là xây dựng thiết kế hệ thống bus AHB bằng ngôn ngữ mô tả phần cứng Verilog, sau đó mô phỏng hệ thống và tiến hành kiểm thử để xác minh tính đúng đắn của thiết kế.

Phạm vi và đối tượng nghiên cứu: Đề án sẽ tập trung vào việc thiết kế hệ thống bus AHB, sử dụng ngôn ngữ mô tả phần cứng Verilog để tạo ra mô hình. Sau đó, chúng ta sẽ sử dụng công cụ EDA Playground để mô phỏng hệ thống và thực hiện kiểm thử. Mục tiêu là xác minh tính chính xác và đáng tin cậy của thiết kế bus truyền thông.

Kết luận: Sự ứng dụng của hệ thống bus AHB trong việc giải quyết các vấn đề truyền thông trên chip được chứng minh là một giải pháp hiệu quả, đặc biệt cho các hệ thống trên chip có độ phức tạp vừa phải và số lượng lõi IP không quá lớn. Các kết quả từ việc thiết kế, mô phỏng và kiểm thử đã cho thấy tính thích hợp và khả năng hoạt động tốt của hệ thống bus AHB, đảm bảo tính liên thông và hiệu suất ổn định trong môi trường thực tế. Đề án gồm 5 chương:

Chương 1: Hệ thống trên vi mạch và vấn đề truyền thông trên vi mạch

Chương 2: Hệ thống bus AMBA AHB

Chương 3: Ngôn ngữ Verilog, System Verilog và công cụ EDA

Chương 4: Xây dựng mô hình hóa hệ thống AHB

Chương 5: Mô phỏng và kiểm thử mô hình

## **PROJECT SUMMARY**

**Research Objective:** Investigate, design, and perform testing of an AHB communication bus. The project aims to construct the AHB bus system using the hardware description language Verilog, followed by system simulation and testing to verify the design's correctness.

**Scope and Research Focus:** The project focuses on the design of an AHB bus system, utilizing the hardware description language Verilog to create a model. Subsequently, the EDA Playground tool is employed for system simulation and testing. The goal is to validate the accuracy and reliability of the communication bus design.

**Conclusion:** The application of the AHB bus system for addressing on-chip communication challenges proves to be an effective solution, particularly for moderately complex on-chip systems with a reasonable number of IP cores. Results from design, simulation, and testing illustrate the suitability and robust operational performance of the AHB bus system, ensuring seamless connectivity and stable performance in real-world environments. The thesis comprises five chapters:

Chapter 1: System on Chip and On-Chip Communication Issues

Chapter 2: AMBA AHB Bus System

Chapter 3: Verilog and System Verilog Languages, and EDA Tools

Chapter 4: Construction of the AHB System Model

Chapter 5: Simulation and Testing of the Model



# **CHƯƠNG 1. HỆ THỐNG TRÊN VI MẠCH VÀ VÂN ĐÈ**

## **TRUYỀN THÔNG TRÊN VI MẠCH**

Chương 1 của báo cáo tập trung vào việc trình bày quá trình phát triển của hệ thống trên vi mạch (SoC) và cung cấp cái nhìn tổng quan về quy trình tạo ra vi mạch. Chương này giới thiệu về sự tiến bộ đáng kể từ vi mạch đơn đến việc tích hợp cao và nâng cao hiệu suất trong các SoC ngày nay. Ngoài ra, chương cũng tóm lược các bước quan trọng trong quy trình thiết kế SoC.

### **1.1 Hệ thống trên vi mạch – lịch sử phát triển**

Công nghệ tích hợp vi mạch (IC) đã có một cuộc cách mạng lớn kể từ khi xuất hiện lần đầu vào năm 1958 [1]. Tiến bộ trong công nghệ vật liệu bán dẫn đã cho phép sản xuất các transistor với kích thước nano, từ đó tăng khả năng tích hợp trên vi mạch đáng kể. Ngày nay, các vi mạch tích hợp có thể chứa hàng tỷ transistor, với ví dụ như chip Xeon 7400 của Intel chứa đến 1,9 tỷ transistor [2]. Điều này không chỉ tạo ra các loại IC có độ phức tạp và hiệu năng cao, hoạt động với công suất tiêu thụ thấp, mà còn cho phép phát triển nhiều chức năng hoạt động khác nhau trên cùng một vi mạch.

Khái niệm "Hệ thống trên vi mạch" (SoC) ra đời để chỉ việc tích hợp các thành phần của một hệ thống máy tính lên một chip duy nhất. Trước đây, các hệ thống điện tử được thực hiện bởi các loại IC riêng biệt cho từng khối chức năng. Nhưng với sự phát triển của công nghệ, các khối chức năng được tích hợp trên một chip đơn không chỉ bao gồm các khối thực hiện các chức năng số (digital) mà còn có thể thực hiện các chức năng tương tự (analog) hoặc cả hai chức năng trên cùng một chip (mix-signal). thậm chí các khối có chức năng thu phát sóng vô tuyến cũng có thể được tích hợp chung với nhau trên một hệ thống đơn nhất.

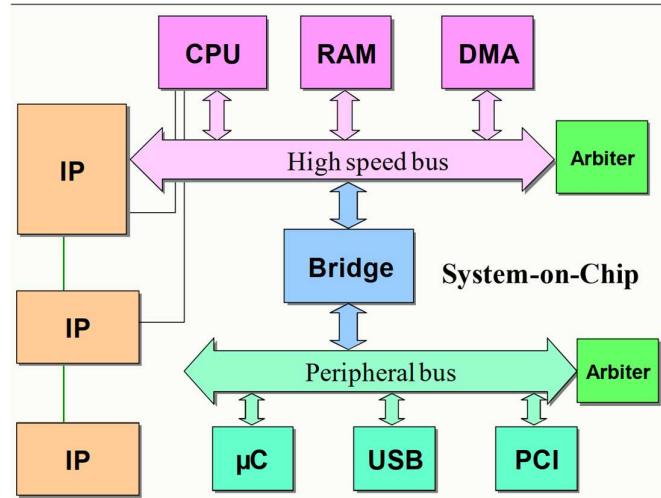
Một SoC thường bao gồm các khối số và khối tương tự, và nó tạo ra cơ hội cho việc tích hợp các khối IP (Intellectual Property) có sẵn, không chỉ giới hạn trong các khối số mà còn mở rộng đến các khối chức năng tương tự và đa dạng hơn. Từ đó, các thiết kế hệ thống trở nên linh hoạt, hiệu quả và tiết kiệm không gian.

Điều này đánh dấu một bước tiến vượt bậc trong công nghệ vi mạch, từ việc xây dựng các linh kiện riêng biệt đến việc tích hợp toàn bộ hệ thống trên một vi mạch duy nhất, tạo ra một tương lai đầy hứa hẹn cho ngành công nghệ thông tin và điện tử. Mỗi

chip SoC có mật độ tích hợp chức năng khác nhau nhưng hầu hết các SoC đều có các thành phần cơ bản sau đây:

- CPU (Central Processing Unit): Lõi vi xử lý là thành phần không thể thiếu trong một SoC làm nhiệm vụ quản lý toàn bộ hoạt động chính của một SoC. CPU đảm nhiệm luồng xử lý chính trong SoC, điều phối các hoạt động giữa các thành phần khác trong SoC, thực thi các tính toán chính. Một SoC có thể có một hoặc nhiều lõi CPU.
- Bus hệ thống (System BUS): Bus hệ thống là có nhiệm vụ kết nối thông suốt các thành phần chức năng khác nhau trong vi xử lý. Bus hệ thống giống như những con đường để vận chuyển dữ liệu giữa các thành phần trong SoC.
- Bộ nhớ (Memory): Bộ nhớ trong một SoC gọi là bộ nhớ nội để phân biệt với bộ nhớ nằm ngoài SoC và giao tiếp với SoC thông qua các chân (pin) điều khiển của SoC. Bộ nhớ nội này có thể là RAM, ROM,...
- Thành phần điều khiển nội (Internal block): là thành phần chỉ điều khiển hoạt động bên trong SoC mà không điều khiển trực tiếp port nào của SoC.
- Ngoại vi (Peripheral): là các khôi có thể lái trực tiếp các chân (pin hoặc port) của SoC để thực thi một chức năng điều khiển bên ngoài SoC.

Trong hình 1.1 dưới đây sẽ mô tả kiến trúc cơ bản của một hệ thống trên vi mạch điện tử.

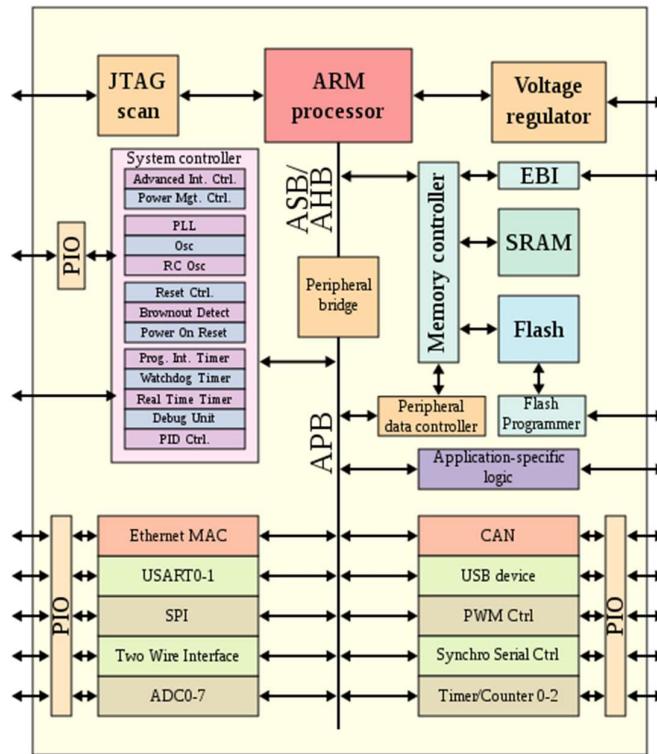


**Hình 1.1 Kiến trúc cơ bản của một hệ thống trên vi mạch**

Hình 1.2 dưới đây thể hiện cấu trúc của vi điều khiển ARM được xây dựng theo kiến trúc SoC (System on Chip). SoC là một mô hình thiết kế tích hợp toàn bộ các thành phần cần thiết của hệ thống trên một chip duy nhất, bao gồm bộ vi xử lý, bộ nhớ, bộ điều khiển giao tiếp, các giao diện I/O và các thành phần ngoại vi khác.

Trong hình ảnh, vi điều khiển ARM nằm ở trung tâm của SoC, đóng vai trò là trí tuệ chính xử lý các tác vụ và quyết định của hệ thống. Vi điều khiển ARM thường có hiệu năng mạnh mẽ, tiêu thụ năng lượng thấp và hỗ trợ nhiều giao thức và tiêu chuẩn giao tiếp.

Xung quanh vi điều khiển ARM, có các thành phần ngoại vi như bộ nhớ RAM và ROM, bộ chuyển đổi analog số (ADC), bộ điều khiển giao tiếp như UART, I2C, SPI, và các giao diện I/O như GPIO và Timer/Counter. Những thành phần này giúp vi điều khiển ARM tương tác với các phần cứng ngoại vi, cảm biến, và các thiết bị khác trong hệ thống.



**Hình 1.2 Cấu trúc của vi điều khiển ARM [7]**

Kể từ khi SoC ra đời vào năm 1974, chúng đã trải qua nhiều sự thay đổi và phát triển vượt bậc. Ngày nay, các SoC có khả năng tích hợp hàng tỷ transistor và số lõi IP (thành phần cốt lõi) trên mỗi chip, và chức năng của chúng ngày càng phong phú. Các thiết bị di động hiện đại sử dụng SoC có thể thực hiện hầu hết các chức năng về điều khiển, quản lý thiết bị, giao tiếp không dây và có dây, cũng như đa phương tiện với chất lượng cao.

Trong cuộc sống hiện đại, các thiết bị di động trở thành không thể thiếu cho mỗi người, hỗ trợ công việc, giao tiếp và giải trí. Do đó, các công ty và viện nghiên cứu đang tập trung nghiên cứu và phát triển các SoC nhằm đáp ứng nhu cầu này. Các SoC hiện nay thường tích hợp nhiều chuẩn giao tiếp không dây như WiMax, Wi-Fi 802.11b/g/n, Bluetooth, 3G và các chuẩn giao tiếp có dây tốc độ cao như USB 2.0, USB 3.0, eSATA.

Sự phát triển của thiết bị di động cũng đi đôi với nhu cầu giải trí di động. Chức năng đa phương tiện trở thành tiêu chuẩn quan trọng trong các thiết bị cá nhân hiện nay. Các SoC được phát triển với tích hợp các bộ giải mã tín hiệu âm thanh (MP3, WAV...) và hình ảnh (AVI, MP4, H.264, HDMI...) để đáp ứng nhu cầu giải trí của người dùng.

Bên cạnh đó, các SoC ngày càng tích hợp nhiều lõi vi xử lý trên cùng một chip (SoC đa lõi) nhằm tăng tốc độ và hiệu năng xử lý của hệ thống. Điều này giúp hệ thống thực hiện các tác vụ đa phương tiện nhanh chóng và mang lại sức mạnh xử lý dữ liệu ngay cả khi di chuyển, đáp ứng nhu cầu thời đại hiện đại.

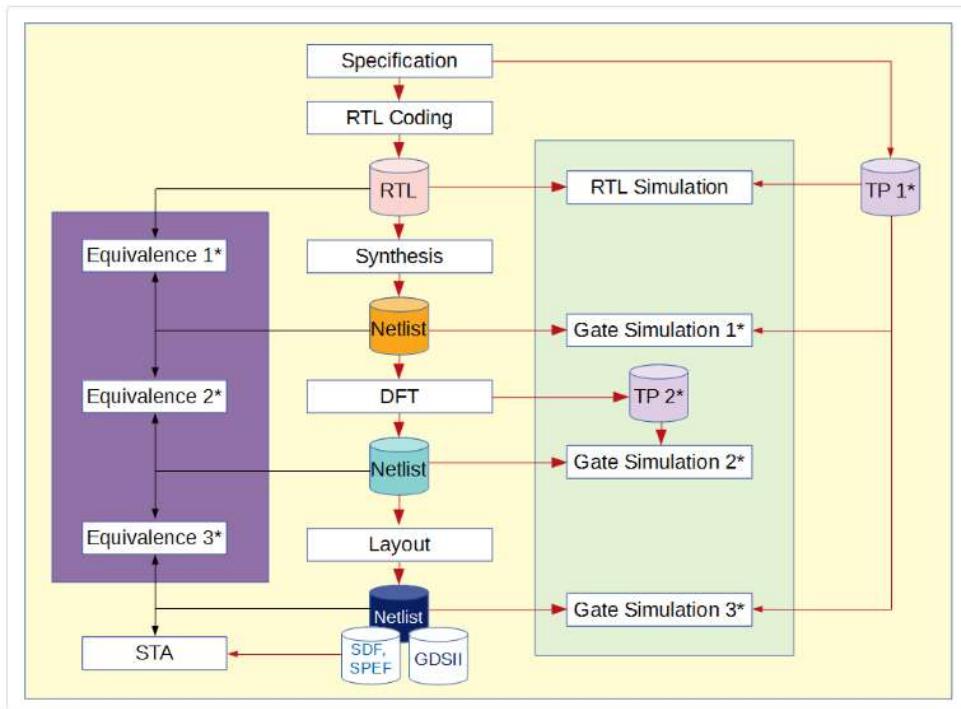
Trong lĩnh vực thiết kế SoC hiện nay, một hướng nghiên cứu thu hút sự quan tâm đáng kể là việc xây dựng các hệ thống truyền thông trên chip hiệu quả. Sự phát triển của các hệ thống trên chip đã khiến cho các hệ thống bus truyền thông trở nên cạn kiệt vì không thể đáp ứng các yêu cầu truyền thông phức tạp với tốc độ cao và dung lượng lớn. Đồng thời, việc tích hợp ngày càng nhiều lõi IP trên một chip đặt ra yêu cầu cần xây dựng các hệ thống truyền thông hỗ trợ quản lý một số lượng lõi IP lớn và có tốc độ truyền dữ liệu cao.

Các phương pháp truyền thông truyền thống như truyền thông chia sẻ bus vẫn đang được cải tiến và phát triển để đáp ứng nhu cầu truyền thông trên chip hiện nay. Các hệ thống bus tiên tiến đã xuất hiện, phần nào đã đáp ứng được những yêu cầu này. Tuy nhiên, các phương pháp truyền thông mới như "Mạng trên chip" (NoC) đang trong quá trình nghiên cứu và phát triển, được coi là phương pháp truyền thông trên chip có nhiều ưu việt hơn và có thể thay thế các phương pháp truyền bus trong các SoC thế hệ mới. Tuy nhiên, việc ứng dụng NoC vào thực tế vẫn còn hạn chế do độ phức tạp cao, chi phí thực hiện lớn và độ ổn định chưa được đảm bảo.

Việc xây dựng và phát triển các loại bus tiên tiến vẫn còn là một nhiệm vụ thách thức và cấp thiết trong lĩnh vực thiết kế SoC. Các kiến trúc bus tiên tiến đóng vai trò quan trọng trong việc thiết kế các SoC với các ứng dụng có độ phức tạp vừa phải, số lượng lõi IP không quá lớn. Điều này giúp đảm bảo tốc độ cao, băng thông lớn và độ ổn định cao của hệ thống truyền thông, đồng thời đáp ứng được nhu cầu hiện nay và tương lai của các thiết bị di động và các ứng dụng đòi hỏi hiệu năng cao.

## 1.2 Quy trình thực hiện thiết kế hệ thống trên chip

Hệ thống trên chip là một chip điện tử, do đó việc thiết kế một SoC (System-on-Chip) phải tuân thủ đúng quy trình thiết kế một chip tích hợp. Quy trình thiết kế chip thông thường bao gồm 9 giai đoạn. Mỗi giai đoạn được chia thành các bước khác nhau như được thể hiện trong Hình 1.3.



Hình 1.3 Sơ đồ khái quát quy trình thiết kế chip

### 1.2.1 Specification

Bước Specification trong quy trình thiết kế chip là giai đoạn đầu tiên và quan trọng trong quá trình phát triển sản phẩm chip. Ở bước này, các yêu cầu về chức năng, hoặc nhiệm vụ cụ thể mà chip cần thực hiện, được tập hợp và mô tả một cách tổng quan từ phía khách hàng. Điều này có thể bao gồm cả mô tả về chức năng logic cũng như các yêu cầu về khía cạnh vật lý và thời gian của chip.

Trong quá trình Specification, các kỹ sư và nhóm phát triển cần hiểu rõ các yêu cầu từ khách hàng và xác định cách thức triển khai chúng thành một sản phẩm chip cụ thể. Mục tiêu chính của bước này là xác định và mô tả rõ ràng những gì mà chip cần thực hiện, cách mà nó hoạt động, và các yêu cầu quan trọng về hiệu suất, tính toàn vẹn dữ liệu, tiêu thụ năng lượng, và nhiều khía cạnh khác.

Trong quá trình mô tả Specification, các kỹ sư thường phải làm rõ các yêu cầu bằng cách chia chúng thành các phần nhỏ hơn, đặc tả chi tiết hơn, và xác định các thông số kỹ thuật cụ thể. Điều này bao gồm việc định nghĩa các tín hiệu điều khiển, giao thức truyền tải dữ liệu, yêu cầu về độ chính xác và độ tin cậy của chip, cũng như các hạn chế về vật lý và thời gian.

Ngoài ra, ở bước này, các kỹ sư cũng bắt đầu xây dựng mô tả cho testbench và hệ thống mô phỏng. Điều này bao gồm việc định nghĩa các kịch bản kiểm tra, các dấu hiệu kích thích, và đầu ra mong đợi từ các test pattern để kiểm tra tính đúng đắn và hiệu suất của chip.

Tầm quan trọng của việc xác nhận thường xuyên với khách hàng tại bước Specification không thể thể hiện đủ. Việc này giúp đảm bảo rằng các yêu cầu được hiểu đúng và được đặt ra một cách rõ ràng từ đầu. Nếu không có sự xác nhận và đồng thuận về yêu cầu, có thể dẫn đến việc phải sửa đổi hoặc thiết kế lại từ đầu, gây lãng phí thời gian, tiền bạc và tài nguyên. Với thiết kế ASIC, việc phải thiết kế lại có thể là một quá trình tốn kém và công phu.

### 1.2.2 RTL Coding

Bước RTL Coding trong quy trình thiết kế chip đóng vai trò quan trọng trong việc triển khai các yêu cầu logic và chức năng từ mô tả ban đầu thành mã nguồn RTL (Register-Transfer Level). Trong giai đoạn này, các kỹ sư thực hiện những công việc quan trọng sau đây:

- Chia nhỏ và Xác định Khối Chức Năng: Các yêu cầu từ mô tả ban đầu được phân tích để xác định các khối chức năng cụ thể trong chip. Mỗi khối chức năng sẽ được mã hóa bằng mã RTL riêng biệt.
- RTL Coding: Các kỹ sư viết mã RTL cho từng khối chức năng. Mã RTL mô tả cách các tín hiệu được chuyển từ một bộ lưu trữ (register) sang bộ lưu trữ khác trong mỗi chu kỳ xung đồng hồ. Mã RTL thường được viết bằng các ngôn ngữ mô tả phần cứng như Verilog hoặc VHDL.
- Xác nhận và Đổi chiều với Khách hàng: Mã RTL được xác nhận và so sánh với các yêu cầu từ phía khách hàng. Điều này đảm bảo rằng mã RTL thể hiện đúng chức năng mà khách hàng yêu cầu và đồng thời đảm bảo sự thống nhất trong suy nghĩ giữa các bên.
- Lập Kế Hoạch Mô Phỏng (Simulation): Kỹ sư lập kế hoạch cho việc mô phỏng mã RTL. Mô phỏng giúp kiểm tra tính đúng đắn và hiệu suất của mã RTL trước khi thực hiện các bước tiếp theo.
- Quản Lý Thiết Kế: Kỹ sư quản lý các tệp mã nguồn RTL, theo dõi tiến độ và đảm bảo tính đúng đắn của mã nguồn.
- Đầu ra là File Code: Kết quả của giai đoạn này là các tệp mã nguồn RTL viết bằng ngôn ngữ mô tả phần cứng. Các tệp mã nguồn RTL này sẽ được sử dụng

làm đầu vào cho các bước tiếp theo trong quy trình thiết kế, như Mô Phỏng (Simulation) và Tổng Hợp (Synthesis).

Bước RTL Coding đóng vai trò quan trọng trong quy trình thiết kế chip, vì nó định hình cách thức hoạt động và chức năng của chip tại mức gần với phần cứng. Công việc chính xác và cẩn thận tại giai đoạn này đảm bảo tính đúng đắn của mã nguồn RTL và giúp tránh sai sót và vấn đề phức tạp ở các giai đoạn sau.

### 1.2.3 RTL Simulation

Bước RTL Simulation là giai đoạn quan trọng để kiểm tra tính đúng đắn của mã RTL (Register Transfer Level) – mức chuyển đổi đăng ký, thông qua việc chạy các tập lệnh kiểm chứng đã được thiết kế. Đây là nơi mà mã RTL được đặt vào môi trường thử nghiệm tạo bởi Testbench (bộ kiểm tra) để thực hiện các tương tác giữa bộ kiểm tra và thiết kế.

Các Testbench này chứa các test pattern – chuỗi các tín hiệu vào (inputs) và dự kiến (expected outputs) đã được xác định trước, tạo ra các tín hiệu điều khiển và dữ liệu để mô phỏng hoạt động của thiết kế. Quá trình này tạo ra waveform – biểu đồ thời gian của các tín hiệu – mô tả cách mà thiết kế phản ứng trong môi trường kiểm tra.

Mục tiêu của bước RTL Simulation là so sánh kết quả thực tế của thiết kế (được mô phỏng) với các kết quả dự kiến. Nếu tất cả các test pattern cho ra kết quả giống như kết quả mong đợi, thì có thể coi rằng mã RTL đã hoạt động chính xác. Tuy nhiên, nếu có sự không khớp nào đó, các lỗi có thể được xác định và sửa chữa trước khi đi vào giai đoạn tiếp theo của quy trình thiết kế chip.

### 1.2.4 Synthesis

Bước Synthesis chuyển đổi mã RTL thành dạng cổng logic bằng cách áp dụng các quy tắc cú pháp của RTL code và sử dụng thư viện cell có sẵn. Công cụ tự động ánh xạ mã RTL thành dạng Gate sử dụng thư viện công nghệ (technology library). Quá trình này liên quan đến các khía cạnh vật lý như độ trễ, tốc độ, diện tích và tiêu thụ năng lượng.

Bước Synthesis là cơ hội đầu tiên để thấy sự liên kết giữa logic và các yếu tố vật lý. Kết quả của bước này tạo ra Netlist cấp cổng logic, đóng vai trò là tiền đề cho các bước tiếp theo trong quy trình thiết kế. Để thực hiện bước này, cần sử dụng mã RTL,

thư viện cell và các yêu cầu về tần số, thời gian trễ và các ràng buộc, được mô tả trong tập tin SDC (Synopsys Design Constraints).

### 1.2.5 Design For Test

Bước DFT (Design for Testability) là quá trình chuẩn bị cho việc kiểm tra công logic trên chip sau khi sản xuất. Nó bao gồm hai phần chính: chèn các mạch kiểm tra vào Gate Level Netlist và tạo các mẫu thử (test patterns) để kiểm tra tính chính xác của công logic.

Sau khi hoàn thành bước Synthesis, ta có Gate Level Netlist của thiết kế. Kỹ sư DFT sẽ thêm các mạch kiểm tra vào Netlist để đảm bảo tính khả kiểm tra của chip. Đồng thời, dựa vào cấu trúc của thiết kế, kỹ sư DFT sử dụng các công cụ đặc thù để tạo các mẫu thử. Các mẫu thử này đảm bảo rằng các phần logic trên chip có thể được kiểm tra một cách hiệu quả.

Trước khi thực hiện kiểm tra trên chip thật tại nhà máy, các mẫu thử được chạy trên Gate Level Design để đảm bảo tính đúng đắn. Sau khi kết quả kiểm tra ổn định, các mẫu thử mới được áp dụng cho chip thực tế trong quá trình sản xuất. Bước DFT đảm bảo rằng việc kiểm tra công logic trên chip được thực hiện một cách hiệu quả và đáng tin cậy.

### 1.2.6 Layout

Bước Layout là giai đoạn tiếp theo sau khi đã chèn mạch DFT vào Netlist. Tại bước này, kỹ sư Layout có nhiệm vụ sắp xếp các cell từ Netlist theo các yêu cầu về vị trí và diện tích từ khách hàng. Các yêu cầu bao gồm vị trí của các IO, khu vực lưu trữ bộ nhớ, các thành phần lớn (macro), và nhiều yếu tố khác.

Kỹ sư Layout cũng phải giải quyết vấn đề quan trọng về Timing, đảm bảo rằng các yêu cầu về tốc độ và các ràng buộc từ khách hàng và nhà máy sản xuất được đáp ứng. Họ cũng phải xử lý các vấn đề về vận hành vật lý như áp, dòng và kiểm tra các lỗi theo quy tắc thiết kế (DRC - Design Rule Check).

Kết quả cuối cùng của bước Layout là file GDSII, một định dạng tập tin chứa thông tin về cấu trúc vật lý của chip. File GDSII này sẽ được chuyển giao cho nhà máy để tiến hành các bước sản xuất chip trên wafer. Bước Layout đảm bảo rằng thiết kế vật lý của chip được thực hiện một cách chính xác và đáp ứng đầy đủ các yêu cầu từ khách hàng và tiêu chuẩn sản xuất.

### **1.2.7 Static Timing Analysis**

Bước Static Timing Analysis (STA) liên quan mật thiết với bước số 5. STA sử dụng dữ liệu từ bước Layout (bao gồm Netlist và file dữ liệu thời gian SDF hoặc SPEF) để tính toán và đánh giáTiming của mạch. Bước này dựa vào các ràng buộc thời gian (SDC) đã được xác định trước. Mục tiêu là đảm bảo rằng tất cả các ràng buộc thời gian đều được tuân theo (MET).

Trong quá trình STA, các công cụ và phần mềm sẽ kiểm tra xem các tín hiệu trong mạch có đáp ứng được các ràng buộc về thời gian hay không. Nếu có bất kỳ tín hiệu nào không tuân theo ràng buộc, STA sẽ tạo ra các cảnh báo và báo cáo để kỹ sư có thể xem xét và điều chỉnh thiết kế để đảm bảo tính toàn vẹn thời gian.

Bước STA đóng vai trò quan trọng để đảm bảo rằng mạch hoạt động chính xác về mặt thời gian và đáp ứng được yêu cầu tốc độ từ khách hàng.

### **1.2.8 Equivalence**

Bước Equivalence kiểm tra tính tương đồng về logic giữa hai thiết kế. Mục tiêu của bước này là xác định xem hai thiết kế có logic tương đương hay không. Kết quả của bước này có thể là "Passed" (tương đồng) hoặc "Failed" (không tương đồng).

Nếu kết quả là "Failed," các bước thiết kế tiếp theo sẽ bị ảnh hưởng và có thể không còn có giá trị. Do đó, kỹ sư phải điều tra nguyên nhân của sự không tương đồng và đề xuất cách khắc phục.

Có thể thực hiện so sánh tương đồng giữa RTL và Netlist, hoặc giữa các phiên bản Netlist khác nhau trong quá trình thiết kế. Netlist có thể được chia thành các phiên bản khác nhau như Pre-DFT Netlist (sau khi synthesis nhưng chưa chèn mạch DFT), Post-DFT Netlist (sau khi chèn mạch DFT), Post-CTS Netlist (sau khi làm Clock Tree Synthesis), và Post-Route Netlist (sau khi làm Routing).

### **1.2.9 Gate simulation**

Bước Gate Simulation tương tự như bước 3, nhưng thay vì sử dụng RTL code, đối tượng đầu vào của mô phỏng là Netlist.

Gate Simulation được chia thành 2 loại:

- No Delay Simulation: Mô phỏng Gate Level mà không tính toán thông tin delay của các Cell và Net.
- Back-Annotation Simulation: Mô phỏng Gate Level kèm theo thông tin delay của Cell và Net từ SDF file của bước Layout.

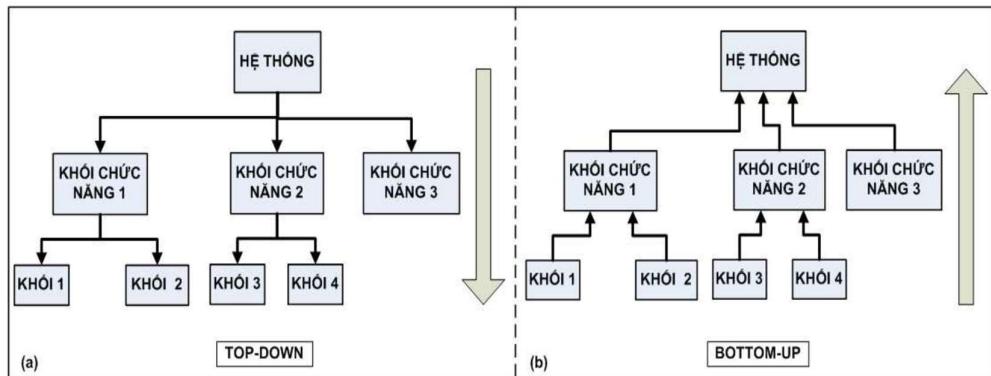
Test pattern ở bước này không khác biệt so với RTL simulation, tuy nhiên môi trường mô phỏng có thể thay đổi do sự khác biệt trong đặc tính giữa RTL code và Gate Level Netlist.

Phương pháp thiết kế từ trên-xuống bắt đầu bằng việc dựa vào yêu cầu kỹ thuật và mô tả hệ thống để chia hệ thống thành các khối nhỏ. Mỗi khối sẽ đảm nhiệm một chức năng riêng biệt. Tiếp theo, từng khối nhỏ này sẽ được chia nhỏ tiếp thành các khối đơn vị nhỏ hơn với các chức năng cụ thể hơn. Quá trình này tiếp tục cho đến khi các khối cơ bản nhất được đạt được như trong Hình 5a. Kết quả cuối cùng của phương pháp này là kiến trúc của hệ thống được xây dựng từ các khối chức năng riêng biệt. Phương pháp này cho phép thiết kế hệ thống ở nhiều mức trừu tượng khác nhau, từ mức hành vi hệ thống đến mức RTL, thậm chí xuống đến mức cổng logic cơ bản. Chia nhỏ hệ thống giúp quản lý sai khác và xung đột, xác định thời gian thực thi hệ thống và cho phép làm việc song song và độc lập giữa các nhóm thiết kế trên từng phần của hệ thống.

### **1.3 Tổng quan phương pháp thiết kế hệ thống**

Phương pháp từ dưới-lên, ngược lại, bắt đầu bằng việc xây dựng các khối chức năng cơ bản nhất, sau đó kết hợp chúng để tạo thành các khối lớn hơn và cuối cùng hình thành hệ thống hoàn chỉnh. Phương pháp này thường áp dụng cho những hệ thống đơn giản hơn và thường dễ dàng hơn trong việc xử lý sai khác và xung đột trong quá trình thiết kế.

Việc lựa chọn phương pháp thiết kế phù hợp sẽ tùy thuộc vào yêu cầu kỹ thuật cụ thể và sự phức tạp của hệ thống cần thiết kế. Hiện nay, phương pháp từ trên-xuống được ưa chuộng và sử dụng chủ yếu trong thiết kế các SoC hiện đại.



**Hình 1.4 Mô hình 2 phương pháp thiết kế top-down và bottom-up**

Phương pháp thiết kế từ dưới-lên tiến hành ngược lại so với phương pháp từ trên-xuống. Ở phương pháp này, người thiết kế sử dụng các khói chức năng đã được thiết kế trước đó và kết hợp chúng với các yêu cầu thiết kế như hiệu năng hệ thống, thông số về điện, công suất tiêu thụ, kích thước của hệ thống, và các yêu cầu sản xuất chip.

Người thiết kế sẽ xây dựng các khói chức năng cơ bản của hệ thống, sau đó kết hợp chúng để tạo thành các khói lớn hơn và cuối cùng hình thành hệ thống hoàn chỉnh. Phương pháp này cho phép tận dụng tối đa các khói chức năng hoặc các lõi IP từ các thiết kế trước đó. Việc tái sử dụng các lõi IP giúp đảm bảo tính ổn định của lõi trong quá trình hoạt động và cho phép mở rộng tính năng để phù hợp với thiết kế hiện tại. Điều này giúp tiết kiệm chi phí tài chính và giảm thời gian phát triển hệ thống, từ lúc phát triển đến khi sản phẩm ra thị trường.

Hiện nay, khi thiết kế các SoC phức tạp, cả hai phương pháp trên thường được kết hợp nhằm tận dụng ưu điểm của từng phương pháp, tăng cường hiệu quả trong quá trình thiết kế và đáp ứng các yêu cầu ngày càng cao của công nghệ và thị trường.

#### 1.4 Vấn đề truyền thông trên chip

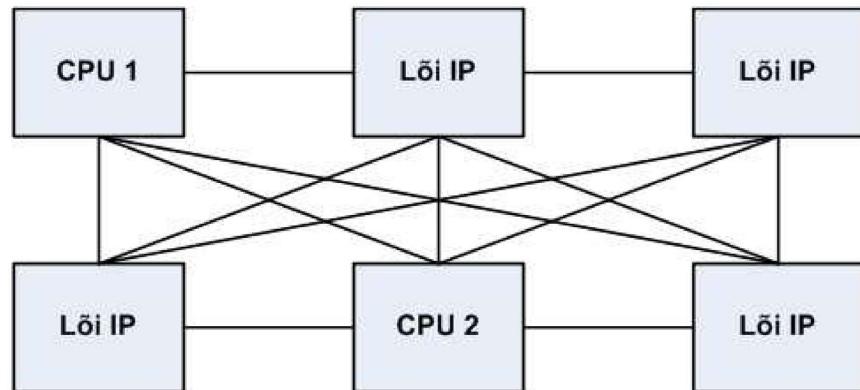
Trong các SoC hiện tại, quá trình truyền thông giữa các lõi IP chủ yếu dựa vào các kết nối điểm-tới-điểm hoặc các kiến trúc bus dùng chung. Tuy nhiên, với sự gia tăng số lượng lõi IP tích hợp trong hệ thống, các phương pháp truyền thông này đang bộc lộ nhiều hạn chế.

Một trong những hạn chế chính của kiểu kết nối điểm-tới-điểm là thông lượng truyền thông bị giới hạn. Dù có hiệu suất tối ưu giữa các lõi IP tham gia truyền thông,

nhưng việc sử dụng tối đa tài nguyên băng thông trong hệ thống là không hiệu quả. Điều này là do phải thiết kế một lượng lớn liên kết trực tiếp giữa các lõi IP, trong khi tần suất sử dụng của các liên kết này không cao. Số lượng liên kết lớn cũng gây ra vấn đề về không gian thực thi phần cứng và độ phức tạp trong quá trình đặt chỗ và định tuyến.

Các kiến trúc truyền thông trên chip cũng đối mặt với hạn chế về công suất tiêu thụ năng lượng, tính toàn vẹn của tín hiệu, và trễ đáp ứng trong quá trình truyền tín hiệu. Việc duy trì tính đồng bộ trên toàn hệ thống cũng trở thành một vấn đề phức tạp.

Nếu không có các cải tiến trong các phương pháp truyền thông trên chip, các hạn chế trên có thể tạo ra các nút thắt cổ chai và giới hạn hiệu quả của việc thiết kế và xây dựng các hệ thống tích hợp. Do đó, việc nghiên cứu và phát triển các kiểu truyền thông mới, cũng như cải tiến phương pháp truyền thông hiện có, là cần thiết để đáp ứng yêu cầu ngày càng cao của công nghệ và thị trường, và tối ưu hóa hiệu năng và tích hợp của các SoC trong tương lai.

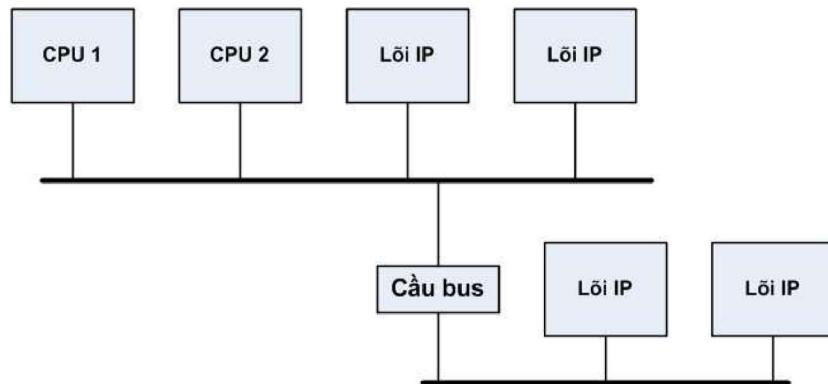


**Hình 1.5 Mô hình kết nối điểm-tới-diểm**

Để khắc phục các nhược điểm của kiến trúc truyền thông điểm-tới-diểm, kiến trúc truyền thông bus đã được đề xuất để thay thế. Hiện nay, trong các hệ thống trên chip phức tạp, kiến trúc truyền thông bus là kiểu truyền thông được sử dụng phổ biến nhất trong các ứng dụng thương mại. Kiến trúc bus cung cấp giải pháp kết nối đơn giản, hiệu quả, cho phép các lõi IP kết nối với nhau thông qua một giao thức truyền thông đã được định trước. So với kiểu kết nối điểm-tới-diểm, kiến trúc bus có nhiều ưu điểm như tính linh hoạt cao, khả năng tái sử dụng lõi IP, và giảm độ phức tạp của hệ thống.

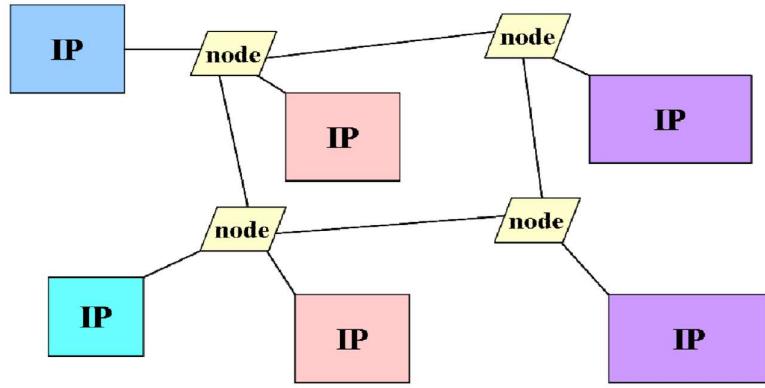
Tuy nhiên, kiến trúc truyền thông bus cũng có một số hạn chế cần được giải quyết. Một trong số đó là việc băng thông của bus phải được chia cho các lõi IP, dẫn đến hiệu suất truyền thông giảm khi số lượng lõi IP tăng lên. Điều này cũng yêu cầu việc xây dựng cơ chế phân quyền truy cập cho các lõi IP khi có nhiều yêu cầu truyền thông cùng lúc. Đồng thời, vì bus được dùng chung cho cả hệ thống, các liên kết trên bus có độ dài đáng kể, làm tăng công suất tiêu thụ toàn hệ thống. Sự tăng số lượng lõi IP cũng dẫn đến kích thước bus tăng lên, gây ra trễ truyền dẫn dữ liệu vượt quá chu kỳ xung nhịp hệ thống.

Các nhà nghiên cứu đã đưa ra nhiều cấu trúc bus khác nhau để giải quyết các vấn đề trên, như cấu trúc bus phân tầng, cấu trúc bus phân chia, cấu trúc bus dạng vòng... Tuy nhiên, các cấu trúc này vẫn không thể hoàn toàn loại bỏ các hạn chế cơ bản của kiến trúc truyền thông dạng bus.



**Hình 1.6 Kiến trúc bus chia sẻ, phân tầng**

Để giải quyết triệt để các vấn đề của kiến trúc bus truyền thông, một mô hình truyền thông mới đã được đề xuất, đó là mô hình mạng trên chip (NoC) như được mô tả trong Hình 1.7. Mô hình mạng trên chip (NoC) được coi là một giải pháp đột phá cho vấn đề truyền thông trên chip, và có thể hoàn toàn thay thế các cấu trúc bus trong các thế hệ SoC tiếp theo.



**Hình 1.7 Mô hình cơ bản của một hệ thống NoC**

Mạng trên chip (NoC) là một mô hình truyền thông phân tán, trong đó các lõi IP và các vi mạch chức năng khác được kết nối thông qua một mạng chuyển mạch (switching fabric). Mạng trên chip sử dụng giao thức định tuyến thông minh để định tuyến các gói tin dữ liệu giữa các nút trên mạng. Điều này cho phép mạng trên chip có tính mềm dẻo cao, khả năng tái sử dụng linh hoạt và khả năng mở rộng dễ dàng khi số lượng lõi IP tăng lên.

Mô hình mạng trên chip giúp giảm đáng kể sự cố chốt băng thông (congestion) và cải thiện hiệu suất truyền thông trong các hệ thống SoC phức tạp. Nó cũng giúp giảm công suất tiêu thụ và đáng tin cậy hơn trong việc truyền tải dữ liệu giữa các lõi IP, nhờ việc tối ưu hóa định tuyến và tăng khả năng phân tán lưu lượng truyền thông. Mô hình mạng trên chip đang ngày càng được ứng dụng rộng rãi trong các thiết kế SoC hiện đại, đóng vai trò quan trọng trong việc tối ưu hóa hiệu năng và hiệu quả truyền thông giữa các lõi IP, từ đó giúp cải thiện tính linh hoạt và khả năng mở rộng của các hệ thống tích hợp.

## 1.5 Kết luận chương

Chương 1 trình bày về tổng quan về SoC và vấn đề truyền thông trên SoC. Đồng thời cũng giới thiệu sơ lược về các bước để sản xuất một SoC trong công nghiệp. Để sản xuất được một Chip cần qua rất nhiều công đoạn tỉ mỉ và cần độ chính xác cao.

## **CHƯƠNG 2. HỆ THỐNG BUS AMBA AHB**

Chương 2 trong báo cáo tập trung vào tổng quan về hệ thống bus AMBA AHB và quy trình hoạt động của nó. Các khía cạnh cơ bản của giao diện AHB, bao gồm cách thức truyền thông giữa thiết bị chủ và thiết bị nô lệ cũng như cách quản lý dữ liệu trong quá trình truyền tải, sẽ được khám phá. Ngoài ra, chương này cũng đề cập đến quy trình hoạt động của hệ thống bus AHB, từ việc tạo yêu cầu cho đến quá trình truyền dữ liệu giữa các thành phần.

### **2.1 Tổng quan về hệ thống bus tốc độ cao AMBA AHB**

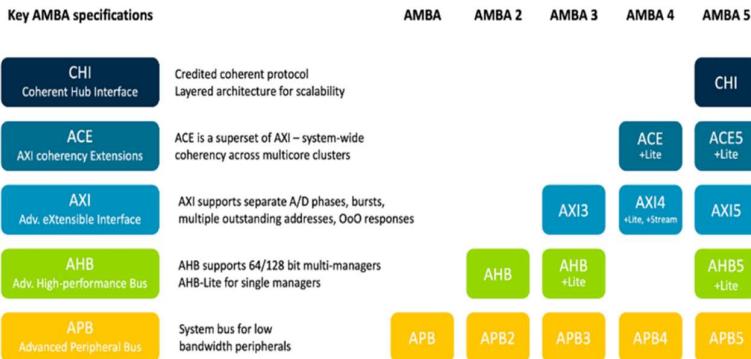
Kiến trúc bus AMBA (Advanced Microcontroller Bus Architecture [1]) là một kiến trúc bus tiên tiến dành cho các hệ thống trên chip, được công ty ARM giới thiệu lần đầu vào năm 1996. AMBA là một trong những kiến trúc bus được sử dụng rộng rãi trong việc điều khiển và hệ thống trên chip hiện đại, và sự tiến bộ và phát triển liên tục của AMBA đã đóng góp quan trọng vào sự phát triển và tăng cường hiệu suất của các hệ thống này. Qua nhiều năm phát triển, AMBA đã tiến hóa và hiện đã có ba phiên bản khác nhau, mỗi phiên bản được cải tiến tương ứng với sự phát triển của hệ thống trên chip và đáp ứng yêu cầu ngày càng cao về tốc độ và băng thông. Ba phiên bản của hệ thống bus AMBA lần lượt là:

- Bus hệ thống tiên tiến – ASB (Advanced System Bus) và bus ngoại vi tiên tiến – APB (Advanced Peripheral Bus) [2].
- Hệ thống bus tiên tiến hiệu năng cao - AHB (Advanced High-performance Bus).
- Giao tiếp mở rộng tiên tiến - AXI (Advanced eXtensible Interface).

AHB là một hệ thống bus thế hệ thứ hai của AMBA, được tạo ra để đáp ứng các thiết kế yêu cầu hiệu năng cao. Đây là một hệ thống bus hỗ trợ đa bus chủ và có khả năng cung cấp băng thông rộng với tốc độ truyền dữ liệu cao. Mục tiêu của bus AHB là để đảm bảo quá trình truyền dữ liệu giữa các lõi IP (Intellectual Property) cần độ rộng bus dữ liệu lớn và tốc độ truyền cao. Nhờ có hiệu năng cao và khả năng hỗ trợ nhiều bus chủ, AHB được sử dụng rộng rãi trong các thiết kế cần xử lý dữ liệu nhanh và hiệu quả trên hệ thống trên chip.

Một số đặc tính của bus AMBA AHB:

- Truyền thông theo khói (burst transfers).
- Phân chia quá trình truyền (split transactions).
- Chuyển giao bus chủ trong một xung đơn.
- Không dung chuyển mức ba trạng thái.
- Độ rộng bus dữ liệu có thể được thay đổi linh hoạt (từ 8 bit lên đến 1024 bit)



**Hình 2.1 Các chuẩn kết nối của ABMA và sự phát triển các giao thức [7]**

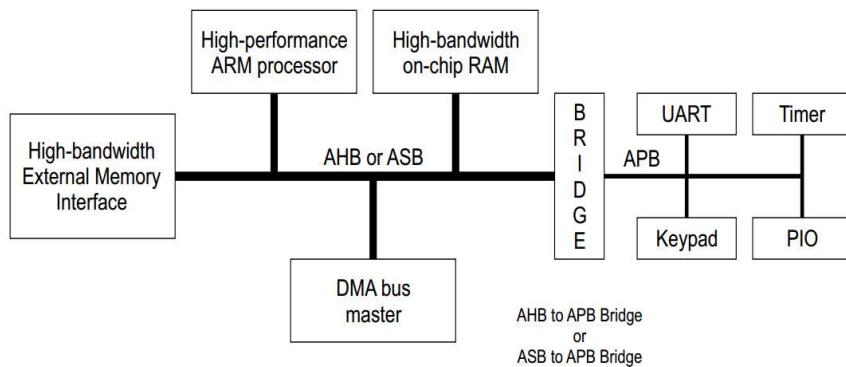
Từ khi công ty ARM giới thiệu kiến trúc bus AMBA (Advanced Microcontroller Bus Architecture) lần đầu tiên vào năm 1996, kiến trúc bus AHB (Advanced High-performance Bus) đã được phát triển và cải tiến qua nhiều phiên bản để đáp ứng các yêu cầu ngày càng cao về hiệu năng và tích hợp trong hệ thống trên chip. Dưới đây là tổng quan về sự phát triển của AHB trong AMBA:

- AHB (AMBA 2.0): AHB là bus hệ thống thế hệ thứ hai của AMBA, được giới thiệu cùng với phiên bản đầu tiên của AMBA 2.0. Phiên bản này đã cung cấp một kiến trúc bus hiệu suất cao hơn so với kiến trúc bus trước đó (APB - Advanced Peripheral Bus). AHB hỗ trợ đa bus chủ và đa bus nô lệ, cho phép nhiều lõi IP truy cập vào các tài nguyên chung trong hệ thống một cách hiệu quả.
- AHB-Lite (AMBA 3 AHB-Lite): AHB-Lite là một phiên bản tiết kiệm diện tích của AHB, được giới thiệu cùng với AMBA 3. Phiên bản này giới thiệu một tùy chọn bus nhỏ gọn và đơn giản hơn cho các hệ thống với yêu cầu diện tích thấp hoặc dễ dàng tích hợp. Mặc dù có những giới hạn về tính năng so với AHB, AHB-Lite vẫn đáp ứng được nhu cầu của nhiều ứng dụng.
- AHB3 (AMBA 5 AHB3): AHB3 là phiên bản tiếp theo của AHB và được giới thiệu cùng với AMBA 5. Phiên bản này tập trung vào việc cải thiện hiệu năng và băng thông của bus để đáp ứng các yêu cầu ngày càng cao của các hệ thống trên chip phức tạp. AHB3 giới thiệu nhiều tính năng mới và tối ưu hóa để hỗ trợ truyền

dữ liệu với tốc độ cao và đáp ứng yêu cầu của các ứng dụng như xử lý hình ảnh, truyền thông, và tích hợp hệ thống phức tạp.

Tổng quát, sự phát triển của AHB trong AMBA đã mang đến những cải tiến đáng kể về hiệu năng, tích hợp và hiệu quả trong việc truyền dữ liệu giữa các lõi IP trong hệ thống trên chip. Các phiên bản AHB khác nhau đã được điều chỉnh và tối ưu hóa để phù hợp với nhiều mục đích và ứng dụng khác nhau của các hệ thống thiết kế.

Hình 2.2 dưới đây mô tả cấu trúc cơ bản của một vi điều khiển có sử dụng hệ thống bus AMBA AHB.



**Hình 2.2 Mô hình vi xử lý sử dụng bus AHB.**

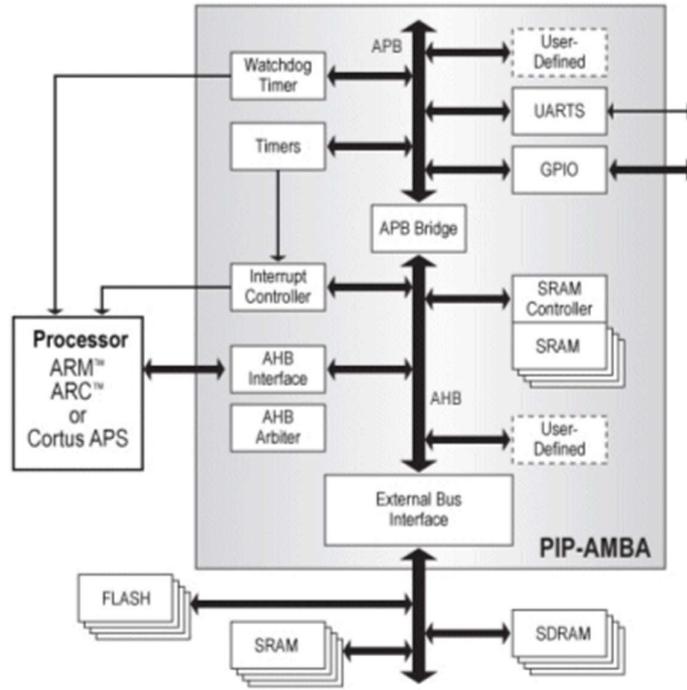
Hiện nay, các nghiên cứu về kiến trúc bus AMBA nói chung và cấu trúc bus AHB nói riêng đã đạt đến mức độ tương đối hoàn chỉnh. Các nghiên cứu hiện tại tập trung chủ yếu vào việc thiết kế các thuật toán phân xử bus nhằm tối ưu quá trình phân quyền truy cập giữa các bus chủ. Mục tiêu của các nghiên cứu này là tối ưu thông lượng truyền thông trên bus và giảm công suất tiêu thụ của bus. Theo các nghiên cứu, việc sử dụng một thuật toán phân xử bus hợp lý có thể giảm công suất tiêu thụ trên bus AHB lên đến 22%. Các nghiên cứu cũng đánh giá hiệu suất truyền thông và công suất tiêu thụ của bus AMBA AHB, đồng thời so sánh với phương thức truyền thông NoC (Network-on-Chip) mới ra đời. Mặc dù các nghiên cứu này chỉ ra rằng hiệu suất của NoC vượt trội hơn so với bus AHB, nhưng bus AHB vẫn hoàn toàn đáp ứng được yêu cầu truyền thông của hầu hết các SoC (System-on-Chip) hiện nay. Một hướng nghiên cứu khác đang được chú ý là thiết kế các hệ thống bus AMBA có khả năng tái cấu hình. Mục tiêu của các nghiên cứu này là xây dựng các hệ thống bus AMBA thông minh, có khả năng tự cấu

hình lại để thích nghi với tình trạng truyền thông hiện tại của hệ thống. Việc tái cấu hình thường tập trung vào việc thay đổi thuật toán phân xử bus hoặc cấu trúc bus để thích ứng với thông lượng truyền thông của bus. Tổng quát, nghiên cứu về kiến trúc bus AMBA trong đó có bus AHB tập trung vào tối ưu hóa quá trình phân quyền truy cập và giảm công suất tiêu thụ trên bus. Mặc dù có các phương pháp truyền thông mới như NoC cho hiệu suất vượt trội hơn, bus AHB vẫn đáp ứng tốt cho hầu hết các hệ thống SoC hiện nay. Hơn nữa, một xu hướng nghiên cứu mới là thiết kế các hệ thống bus AMBA có khả năng tái cấu hình, tạo ra các hệ thống bus thông minh và linh hoạt hơn.

Kiến trúc bus AMBA không chỉ dừng lại ở việc phát triển trong phòng thí nghiệm, mà còn là một trong những kiến trúc bus được ứng dụng rộng rãi trong nhiều SoC thương mại. Sự thành công của kiến trúc bus này là nhờ vào cấu trúc mở và tính linh hoạt, cho phép nó có thể được thay đổi để phù hợp với nhiều kiểu truyền thông khác nhau trên SoC. Do đó, các nhà phát triển hệ thống trên chip thường sử dụng kiến trúc bus AMBA khá phổ biến. Kiến trúc bus AMBA, đặc biệt là cấu trúc bus AHB, xuất hiện với đa dạng các thể loại, từ việc được tích hợp sẵn trong các lõi IP cho đến những hệ thống bus của các SoC hoàn chỉnh. Nó có thể tồn tại dưới dạng các lõi IP có chức năng truyền thông, có thể là các sản phẩm phần cứng hoặc lõi IP mềm để tích hợp vào các hệ thống khác nhau. Một số sản phẩm sử dụng kiến trúc bus AMBA là:

- Lõi IP 10-100 Ethernet - AHB và lõi IP Gigabit - AHB (Arasan)
- Các lõi IP chứa các thành phần của bus AHB (Aurora VLSI)
- Lõi PiP-AMBA (CAST)
- Các dòng vi xử lý ARM9E, ARM10E (sử dụng bus AHB) và ARM11, Cortex (sử dụng bus AXI) của công ty ARM

Những sản phẩm này đều mang tính ứng dụng cao và có thể được tích hợp vào các hệ thống SoC để cung cấp các tính năng truyền thông và giao tiếp hiệu quả.

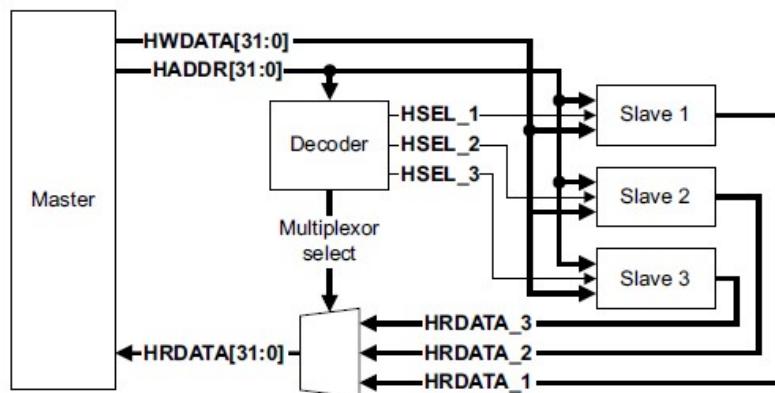


Hình 2.3 Sơ đồ khái niệm khung nền tảng IP PIP-AMBA của CAST và SOC Solutions [4]

## 2.2 Mô hình cấu trúc và hoạt động chung của bus AMBA AHB

### 2.2.1 Mô hình một hệ thống AMBA AHB

Một hệ thống AMBA AHB có những thành phần cơ bản như: bus chủ, bus tớ, bộ phân xử, bộ giải mã tín hiệu địa chỉ, bộ phân kênh... và được kết nối với nhau như hình 2.4.



Hình 2.4 Sơ đồ kết nối các thành phần trong bus AHB [11]

- Bus chủ (master): là thành phần cho phép khởi động quá trình đọc hoặc ghi bằng cách cung cấp tín hiệu địa chỉ và thông tin điều khiển đến bus tớ hoặc các khối khác để thực hiện chức năng nhất định. Tại một thời điểm thì chỉ có một bus chủ được phép hoạt động tham gia vào quá trình truyền.
- Bus tớ (bus slave): một bus tớ sẽ đáp ứng yêu cầu đọc hoặc ghi từ một bus chủ và bus tớ chỉ thực hiện quá trình trong một vùng địa chỉ đã được thiết lập từ trước. Các tín hiệu của bus tớ phản hồi lại bus chủ cho biết quá trình truyền dữ liệu thành công, thất bại hay đang được tiến hành.
- Bộ giải mã tín hiệu địa chỉ (decoder address): được sử dụng để giải mã địa chỉ của mỗi lần truyền và cung cấp tín hiệu điều khiển cho biết bus tớ nào được chọn để thực hiện quá trình truyền dữ liệu.
- Bộ phân kênh tín hiệu (multiplexor): có chức năng là như một "người trung gian" giúp chuyển đổi các tín hiệu địa chỉ và dữ liệu giữa bus chủ và bus tớ trong quá trình truyền dữ liệu.

Trong sơ đồ Hình 2.4, mỗi bus chủ sẽ lan truyền tín hiệu địa chỉ và tín hiệu điều khiển dưới dạng quảng bá để thông báo về quá trình truyền dữ liệu đang diễn ra. Nhiệm vụ của bộ phân kênh để chuyển tín hiệu từ bus chủ được chọn đến tất cả các bus tớ. Các bộ giải mã địa chỉ sẽ sử dụng tín hiệu địa chỉ để xác định đích đến là bus tớ nào tham gia vào quá trình truyền dữ liệu.

### **2.2.2 Quy trình hoạt động của hệ thống AMBA AHB**

Để bắt đầu một quá trình truyền dữ liệu, bus chủ cần gửi tín hiệu yêu cầu quyền truy cập bus đến bộ phân xử. Bộ phân xử sẽ quan sát trạng thái truy cập hiện tại của các bus chủ và gửi tín hiệu báo hiệu khi bus chủ được phép sử dụng bus. Sau khi được phép truy cập, bus chủ sẽ bắt đầu quá trình truyền dữ liệu bằng cách phát đi các tín hiệu địa chỉ và tín hiệu điều khiển. Các tín hiệu này cung cấp thông tin về địa chỉ, hướng truyền và độ rộng của quá trình truyền, cũng như báo hiệu nếu đây là một quá trình truyền khống.

Trong quá trình truyền dữ liệu, một bus ghi dữ liệu sẽ được sử dụng để chuyển dữ liệu từ bus chủ đến bus tớ, và một bus đọc dữ liệu sẽ được dùng để chuyển dữ liệu từ bus tớ đến bus chủ. Mọi quá trình truyền đều bao gồm một chu kỳ để truyền tín hiệu địa chỉ và điều khiển, và một hoặc nhiều chu kỳ để truyền dữ liệu.

Chu kỳ dành cho việc truyền địa chỉ không cần mở rộng vì tất cả các bus tớ phải lấy mẫu địa chỉ trong suốt thời gian truyền. Tuy nhiên, chu kỳ truyền dữ liệu có thể được

mở rộng thông qua tín hiệu HREADY. Khi tín hiệu HREADY ở mức thấp, nó tạo ra trạng thái chờ trong quá trình truyền, cho phép bus tớ có thêm thời gian để cung cấp hoặc lấy mẫu dữ liệu.

Trong một quá trình truyền, bus tớ sẽ thông báo các trạng thái hiện tại của quá trình truyền bằng cách sử dụng tín hiệu phản hồi HRESP[1:0]. Tín hiệu này bao gồm hai trạng thái:

- HRESP[1:0] = 00: Truyền thành công (OKAY): Bus tớ thông báo rằng quá trình truyền đã thành công và dữ liệu đã được truyền đúng.
- HRESP[1:0] = 01: Truyền thất bại (ERROR): Bus tớ thông báo rằng quá trình truyền đã thất bại do xảy ra lỗi trong quá trình truyền dữ liệu.

## 2.3 Các chế độ truyền cơ bản của bus AMBA AHB

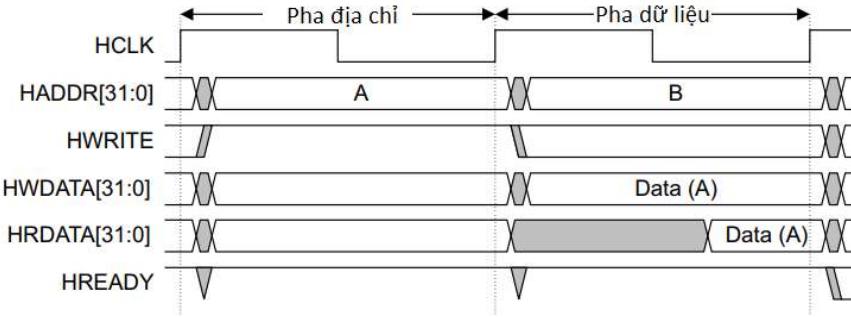
Một quá trình truyền trên bus AHB bao gồm 2 pha riêng biệt:

- Pha địa chỉ, chỉ trong một chu kỳ xung đơn duy nhất.
- Pha dữ liệu, có thể được thực hiện trong nhiều chu kỳ xung. Khi pha này hoàn thành thì sẽ được thông báo tín hiệu HREADY.

### 2.3.1 Quá trình truyền cơ bản, không có trạng thái đợi

Một quá trình truyền cơ bản, không có trạng thái đợi như hình 2.5 dưới đây bao gồm 3 bước cơ bản:

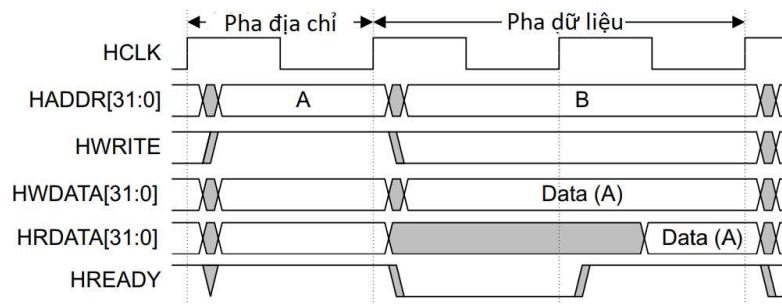
- Bus chủ phát đi tín hiệu địa chỉ kèm theo tín hiệu điều khiển tại sườn lên của xung HCLK.
- Bus tớ sẽ lấy mẫu tín hiệu địa chỉ và thông tin từ bus chủ tại sườn lên tiếp theo của xung HCLK.
- Sau khi bus tớ lấy mẫu tín hiệu địa chỉ và điều khiển, nó có thể bắt đầu xử lý và phát tín hiệu phản hồi lại cho bus chủ biết thông tin về quá trình truyền. Đồng thời bus tớ sẽ phát tín hiệu HREADY ở mức cao để biết quá trình truyền thành công.



**Hình 2.5 Quá trình truyền cơ bản không trạng thái đợi trên bus AHB [11]**

### 2.3.2 Quá trình truyền cơ bản có trạng thái đợi

Một bus tó có khả năng chèn thêm các trạng thái chờ vào trong pha dữ liệu của bất kỳ quá trình truyền nào để cho phép quá trình truyền có thêm thời gian hoàn thành, như được minh họa trong Hình 2.6. Điều này được thực hiện như sau: ở đầu pha dữ liệu, bus tó sẽ đưa tín hiệu HREADY về mức thấp để báo hiệu rằng quá trình truyền chuyển sang trạng thái delay. Khi dữ liệu cho quá trình truyền đã sẵn sàng, bus tó sẽ chuyển tín hiệu HREADY quay lại mức cao tại sườn lên của xung HCLK, báo hiệu rằng quá trình truyền đã hoàn thành thành công.



**Hình 2.6 Quá trình truyền cơ bản có trạng thái đợi trên bus AHB [11]**

Lưu ý rằng, trong quá trình ghi dữ liệu, bus chủ sẽ giữ dữ liệu ổn định trong suốt các chu kỳ mở rộng. Trong khi đó, trong quá trình đọc dữ liệu, bus tó chỉ cần cung cấp dữ liệu vào cuối chu kỳ cuối cùng của quá trình truyền.

### 2.3.3 Quá trình truyền khối (burst transfer)

Hệ thống bus AHB hỗ trợ quá trình truyền theo khối bao gồm: truyền đơn (single transfer), truyền khối với chiều dài khối bất kỳ (non-aligned burst), và truyền khối với các nhịp 4, 8 và 16 (4-beat, 8-beat, 16-beat bursts). Giao thức này cung cấp hai kiểu

truyền khói là truyền khói với địa chỉ tăng dần - truyền khói tăng (incrementing burst) và truyền khói với địa chỉ cuộn - truyền khói cuộn (wrapping burst).

Quá trình truyền khói với địa chỉ tăng dần sẽ truy cập các vị trí địa chỉ một cách tuần tự và địa chỉ của mỗi lần truyền trong truyền khói tăng theo địa chỉ của lần truyền trước. Đối với quá trình truyền khói với địa chỉ cuộn, nếu địa chỉ bắt đầu của quá trình truyền không trùng với tổng số byte trong khói (tổng byte = kích thước x số nhịp (beats)), thì địa chỉ của các quá trình truyền sẽ bị quay trở lại đến biên.

Thông tin về kiểu truyền khói sẽ được cung cấp bởi tín hiệu HBURST[2:0] và có 8 kiểu truyền được định nghĩa như trong bảng 2.1.

**Bảng 2.1 Các kiểu truyền khói được bus AHB hỗ trợ**

HBURST[2:0]	Kiểu	Mô tả
000	SINGLE	Truyền đơn
001	INCR	Truyền khói tăng với độ dài bất kỳ
010	WRAP4	Truyền khói cuộn với nhịp 4
011	INCR4	Truyền khói tăng với nhịp 4
100	WRAP8	Truyền khói cuộn với nhịp 8
101	INCR8	Truyền khói tăng với nhịp 8
110	WRAP16	Truyền khói cuộn với nhịp 16
111	INCR16	Truyền khói tăng với nhịp 16

Trong một số trường hợp, quá trình truyền đơn có thể được thực hiện bằng cách sử dụng quá trình truyền khói tăng có chiều dài bất kỳ, trong đó mỗi khói truyền sẽ có chiều dài là 1 byte. Một quá trình truyền khói tăng có thể có chiều dài bất kỳ, nhưng giới hạn là không được quá 1 KByte địa chỉ biên.

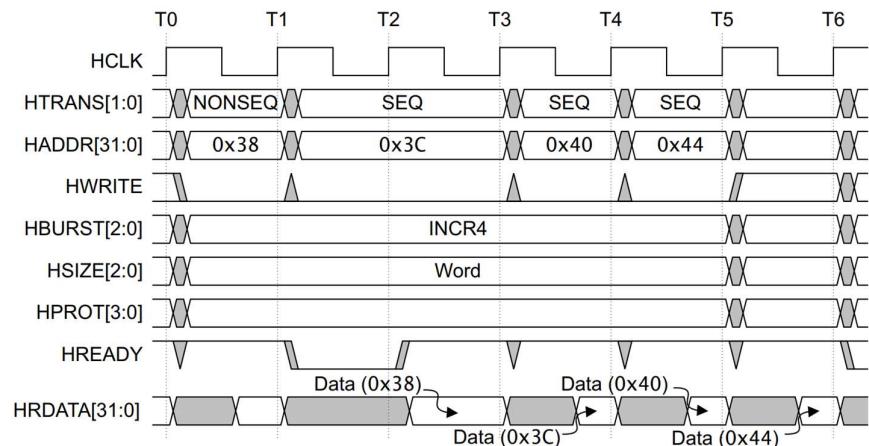
Trong quá trình truyền khói, địa chỉ của quá trình sẽ tăng phụ thuộc vào tín hiệu điều khiển HSIZE như bảng 2.2 dưới đây.

**Bảng 2.2 Bảng độ rộng kiểu dữ liệu truyền**

HSIZE[2:0]	Độ rộng(theo bit)	Mô tả
000	8	Byte
001	16	Nửa từ
010	32	Một từ
011	64	Hai từ
100	128	Bốn từ
101	256	Tám từ
110	512	-
111	1024	-

Lưu ý: Trong quá trình truyền độ rộng của dữ liệu phải nhỏ hơn hoặc bằng độ rộng được thiết lập bởi HSIZE.

Một quá trình truyền khối tăng với nhịp 4 được minh họa bằng gián đồ xung như ở Hình 2.7. Trong quá trình này, khối truyền đầu tiên được thực hiện tại địa chỉ h00000038. Các khối tiếp theo sẽ có địa chỉ tăng tuần tự với giá trị mỗi lần tăng là 4 byte (do độ rộng bus địa chỉ là 32 bit) và kết thúc ở khối h00000044.

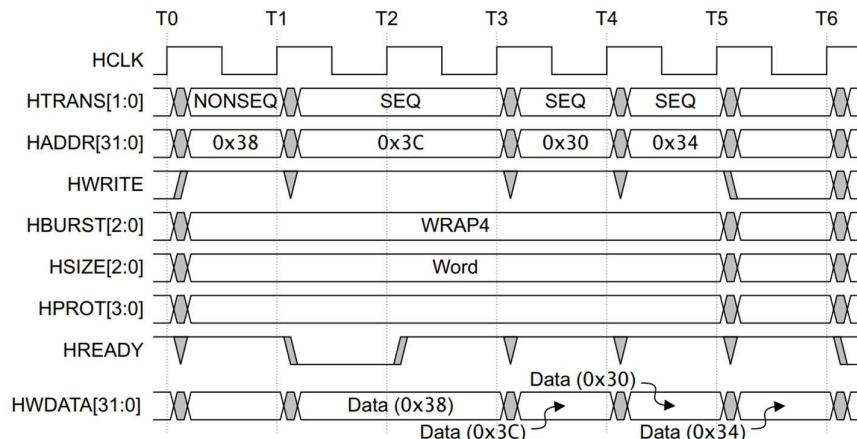


**Hình 2.7 Quá trình truyền khối tăng với 4 nhịp [11]**

Một quá trình truyền khồi cuộn với 4 nhịp được minh họa như hình 2.8 dưới đây. Trong quá trình này, ta thấy tín hiệu điều khiển HBRUST cho biết quá trình truyền là kiểu cuộn với 4 nhịp, tín hiệu HSIZE cho biết độ rộng dữ liệu mỗi nhịp là Word (tức 2 bytes).

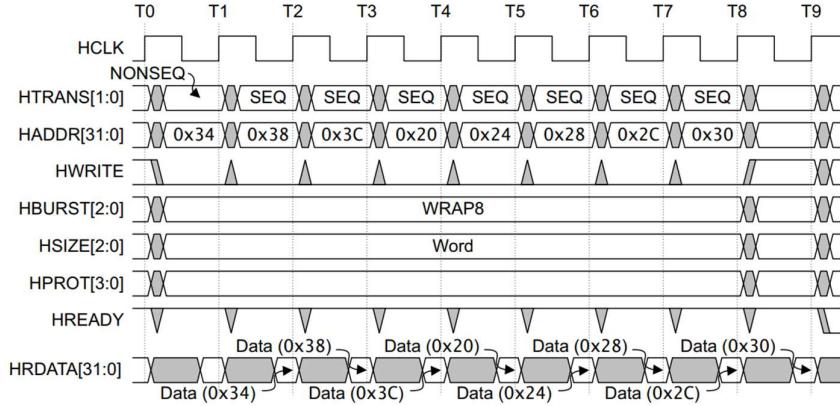
- Ta sẽ tính được địa chỉ biên bằng công thức:  $\text{địa chỉ biên} = \text{địa chỉ đầu} + (\text{độ rộng} * \text{số nhịp})$
- Địa chỉ cuộn lại được tính bằng:  $\text{địa chỉ cuộn} = (\text{địa chỉ đầu} / (\text{độ rộng} * \text{số nhịp})) * (\text{độ rộng} * \text{số nhịp})$

Quá trình truyền khồi cuộn với địa chỉ đầu tiên là 0x38 sẽ tăng mỗi nhịp thêm 4 byte, cho đến khi gặp địa chỉ biên thì sẽ quay về địa chỉ cuộn và tăng tiếp cho đến khi đủ 4 nhịp.



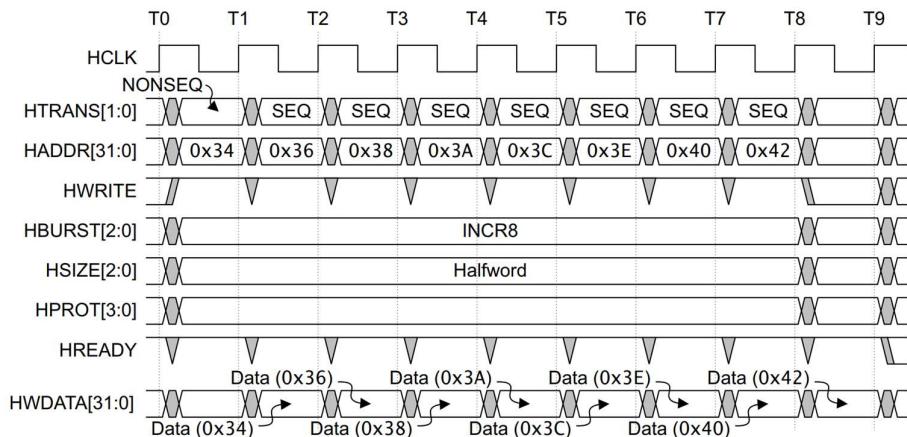
**Hình 2.8 Quá trình truyền khồi cuộn với 4 nhịp [11]**

Một quá trình truyền khồi cuộn với nhịp 8 được minh họa bởi giản đồ xung như ở Hình 2.9. Ta thấy địa chỉ của các khồi sau vẫn tăng thêm so với khồi trước là 4 byte. Tuy nhiên, tại khồi truyền có địa chỉ h0000003C, thì khồi tiếp theo thay vì có giá trị là h00000040 (là địa chỉ biên), sẽ bị cuộn lại địa chỉ h00000020. Các khồi tiếp theo vẫn tiếp tục diễn ra như bình thường.



**Hình 2.9 Quá trình truyền khôi cuộn với 8 nhịp [11]**

Một quá trình truyền khôi tăng với 8 nhịp được thể hiện như hình 2.10 dưới đây. Ta thấy địa chỉ của quá trình bắt đầu từ 0x34 và tăng 7 lần, mỗi lần tăng thêm 2 byte vì HSIZE cho biết Halfword có kích thước là 2 byte.



**Hình 2.10 Quá trình truyền khôi tăng với 8 nhịp [11]**

### 2.3.4 Thông báo quá trình truyền của bus chủ

Trong quá trình truyền, bus chủ sẽ thông báo cho bộ phân xử và bus tớ về kiểu truyền đang được thực hiện. Thông báo này giúp cho bộ phân xử và bus tớ có thể biết quá trình truyền nào đang diễn ra và kết hợp cùng tín hiệu HBURST để giám sát quá trình truyền. Điều này cho phép bộ phân xử xác định thời điểm thay đổi các tín hiệu cấp quyền truy cập bus và thông báo cho bus tớ về tình trạng sẵn sàng của bus chủ để có thể truyền dữ liệu một cách hợp lý.

Các quá trình truyền trên hệ thống bus AHB có thể được chia thành một trong bốn kiểu truyền: IDLE, BUSY, NONSEQ, SEQ như được trình bày trong Bảng 2.3. Việc mã hóa các kiểu truyền này sẽ được thực hiện thông qua tín hiệu HTRANS truyền từ bus chủ sang bus tớ.

**Bảng 2.3 Các trạng thái của bus chủ tương với các kiểu truyền.**

HTRANS[1:0]	Kiểu	Mô tả
00	IDLE	<p>Báo hiệu "IDLE" được sử dụng để thông báo rằng không có quá trình truyền dữ liệu nào được thực hiện trên hệ thống bus AHB.</p> <p>Các bus tớ luôn cung cấp một trạng thái chờ với tín hiệu HREADY ở mức thấp và gửi phản hồi "OKAY" đối với một quá trình IDLE.</p>
01	BUSY	<p>Kiểu truyền "BUSY" cho phép bus chủ thêm các chu kỳ trống vào giữa các quá trình truyền khói. Kiểu truyền này báo hiệu rằng bus chủ vẫn đang tiếp tục một quá trình truyền khói, nhưng khói truyền tiếp theo sẽ chưa được thực hiện ngay lập tức.</p> <p>Tương tự như khi gửi một quá trình "IDLE," các bus tớ phải luôn cung cấp một trạng thái chờ với tín hiệu HREADY ở mức thấp và gửi phản hồi "OKAY." Tại thời điểm này, quá trình truyền sẽ bị bỏ qua và không được thực hiện bởi bus tớ.</p>
10	NONSEQ	Báo hiệu "NONSEQ" đại diện cho khói truyền đầu tiên của một quá trình truyền khói hoặc một quá trình truyền đơn.
11	SEQ	Trong chế độ này, địa chỉ khói truyền hiện tại sẽ liên quan đến địa chỉ khói truyền kế trước đó. Thông tin điều khiển sẽ được sử dụng để tính ra địa chỉ của khói truyền hiện tại. Địa chỉ của khói truyền sẽ bằng địa chỉ của lần truyền trước cộng với kích thước truyền (tính bằng byte).

## 2.4 Kết luận chương

Chương 2 đã làm rõ tổng quan về giao thức AHB trong họ giao thức AMBA. Qua đây trình bày chi tiết về các thức hoạt động của một hệ thống AHB và phân tích cách hoạt

động truyền của giao thức này. Từ đó làm cơ sở lý thuyết để xây dựng hệ thống đề xuất của bài toán ở Chương 4.

# CHƯƠNG 3. NGÔN NGỮ VERILOG, SYSTEM VERILOG VÀ CÔNG CỤ EDA

Chương này trình bày một cách khái quát về ngôn ngữ Verilog được dùng để viết thiết kế cho RTL Design, ngôn ngữ System verilog được sử dụng cho việc kiểm thử các bản thiết kế và công cụ EDA miễn phí được sử dụng cho quá trình thực hiện đồ án.

## 3.1 Ngôn ngữ Verilog

### 3.1.1 Khái niệm về ngôn ngữ Verilog

Verilog là một loại ngôn ngữ được sử dụng để miêu tả và thiết kế các hệ thống kỹ thuật số, chẳng hạn như bộ chuyển mạch mạng, bộ vi xử lý, bộ nhớ và thậm chí các thành phần đơn giản như Flip-Flop. Bằng cách sử dụng Verilog, người thiết kế có thể tạo ra mô hình và mô tả cách hoạt động cũng như cấu trúc của các mạch điện tử ở nhiều cấp độ khác nhau.

### 3.1.2 Các mức mô tả của Verilog

#### a) Mức hành vi

Tại mức này, Verilog được sử dụng để miêu tả cách hoạt động của mạch điện tử mà không quan tâm đến cụ thể làm thế nào nó thực hiện. Điều này bao gồm mô tả các tương tác và phản ứng của mạch khi nhận đầu vào và tạo ra đầu ra. Mức mô tả này thường tập trung vào cách các tín hiệu thay đổi theo thời gian và cách chúng ảnh hưởng đến nhau.

#### b) Mức truyền thanh ghi

Mức truyền thanh ghi (Register Transfer Level - RTL) là một cấp độ mô tả trong Verilog, tập trung vào cách các thanh ghi và tín hiệu logic tương tác để thực hiện một chức năng cụ thể trong mạch điện tử. Mức RTL được sử dụng để miêu tả cách dữ liệu di chuyển giữa các thanh ghi và cách các phép toán logic được thực hiện. Điều này cho phép thiết kế và kiểm tra tương đối trừu tượng nhưng vẫn chú trọng vào cách hoạt động của mạch.

Mức RTL thường là bước tiếp theo sau khi đã hoàn thành mô tả chức năng hoặc mô tả hành vi của mạch. Nó thường được sử dụng trong quá trình thiết kế chi tiết của mạch

điện tử, và nó giúp dễ dàng tạo ra mô hình có khả năng kiểm tra và tự động tạo ra mã máy.

c) Mức công logic

Mức công logic (Gate-Level) là một cấp độ mô tả trong Verilog tập trung vào cách các công logic cụ thể (ví dụ: AND, OR, NOT) và flip-flop hoạt động để thực hiện một chức năng cụ thể trong mạch điện tử. Ở mức này, mô tả được thực hiện bằng cách sử dụng các công logic và flip-flop trong ngôn ngữ Verilog để biểu diễn mạch điện tử cụ thể.

Mức công logic thường là bước cuối cùng trong quá trình thiết kế mạch điện tử, khi đã hoàn thành mô tả cấp cao hơn (chức năng, hành vi, truyền thanh ghi). Nó giúp tạo ra một biểu diễn cụ thể và chính xác của mạch, dựa trên các công logic thực tế và các phần tử cơ bản khác.

Mức công logic (Gate-Level) trong Verilog là cấp độ mô tả chính xác nhất, tập trung vào việc sử dụng các công logic và flip-flop cụ thể để biểu diễn hoạt động của mạch điện tử.

## 3.2 Ngôn ngữ System Verilog

### 3.2.1 Khái niệm về ngôn ngữ System Verilog

SystemVerilog là một ngôn ngữ mô tả phần cứng (HDL) mở rộng từ ngôn ngữ Verilog. Nó không chỉ bao gồm tất cả các tính năng của Verilog mà còn bổ sung thêm nhiều tính năng mới mạnh mẽ, chủ yếu tập trung vào việc kiểm tra phần cứng.

SystemVerilog được sử dụng trong nhiều khía cạnh của viễn thông và kiểm tra phần cứng. Một số tính năng nổi bật của SystemVerilog bao gồm:

- Kiểm tra phần cứng (Hardware Testing): SystemVerilog hỗ trợ các tính năng kiểm tra phần cứng như kiểm tra hạng mục, kiểm tra kiểu dữ liệu, và kiểm tra tượng hình. Nó cung cấp mô hình kiểm tra mạnh mẽ và linh hoạt cho việc kiểm tra mạch.
- Kiểu dữ liệu trừu tượng (Abstract Data Types): SystemVerilog cho phép bạn định nghĩa kiểu dữ liệu trừu tượng, giúp tạo ra mã nguồn dễ đọc hơn và dễ quản lý hơn.

- Lập trình hướng đối tượng (Object-Oriented Programming): SystemVerilog hỗ trợ lập trình hướng đối tượng với khái niệm lớp và đối tượng, giúp mô phỏng mạch phức tạp và kiểm tra mạch dễ dàng hơn.
- Mô phỏng và xác thực (Simulation and Verification): SystemVerilog có khả năng mô phỏng mạch và kiểm tra tốt hơn so với Verilog truyền thống, với sự hỗ trợ bổ sung cho kiểm tra tốt hơn.

### **3.2.2 Điểm nâng cấp cơ bản của System Verilog so với Verilog**

SystemVerilog hỗ trợ nhiều cách mạnh mẽ để người dùng tùy chỉnh và quản lý kiểu dữ liệu:

- TypeDef (typedef): Người dùng có thể sử dụng từ khóa "typedef" để định nghĩa các kiểu dữ liệu mới riêng của họ. Điều này giúp làm cho mã nguồn dễ đọc hơn và quản lý dữ liệu hiệu quả hơn.
- Con trỏ truy xuất bộ nhớ động: SystemVerilog cho phép sử dụng con trỏ để truy cập và quản lý động bộ nhớ, giúp trong việc làm việc với dữ liệu động và linh hoạt.
- Kiểu Enum: Hỗ trợ kiểu dữ liệu đặc biệt là kiểu "enum". Mỗi phần tử trong kiểu "enum" phải được gán một giá trị riêng biệt, giúp phát hiện lỗi gán giá trị trùng trong máy trạng thái FSM sớm hơn.
- Kiểu Struct/Union: Người dùng có thể sử dụng kiểu "struct" hoặc "union" để tổ chức và lưu trữ dữ liệu phức tạp.
- Mảng Liên kết (Associative Array): SystemVerilog hỗ trợ mảng liên kết để lưu trữ và quản lý các dữ liệu không cố định với không gian dữ liệu không liền mạch.
- Class và OOP (Object-Oriented Programming): SystemVerilog hỗ trợ lập trình hướng đối tượng với khái niệm lớp, đối tượng, kế thừa, thuộc tính (data), và các phương thức (method). Điều này cho phép mô hình hóa các giao thức phức tạp và xử lý dữ liệu hiệu quả hơn.
- Các cấu trúc khác (program, clocking, interface): Hỗ trợ các cấu trúc lặp và các từ khóa điều khiển giúp tổ chức và quản lý mã nguồn dễ dàng hơn.
- Tạo giá trị ngẫu nhiên có ràng buộc: SystemVerilog cho phép tạo giá trị ngẫu nhiên và áp dụng ràng buộc cho quá trình tạo giúp đảm bảo rằng giá trị ngẫu nhiên nằm trong phạm vi mong muốn và tuân theo các điều kiện đã định.

Tóm lại, SystemVerilog cung cấp nhiều tính năng cải tiến so với Verilog, giúp người dùng quản lý và mô hình dữ liệu cũng như thực hiện kiểm tra và thiết kế phần cứng một cách hiệu quả hơn.

### 3.3 Công cụ EDA

#### 3.3.1 *EDA Playground*

Công cụ EDA Playground là một môi trường trực tuyến mạnh mẽ và tiện lợi, được sử dụng rộng rãi trong lĩnh vực thiết kế và mô phỏng hệ thống điện tử. EDA Playground cung cấp một nền tảng đáng tin cậy để thực hiện các hoạt động liên quan đến thiết kế mạch điện tử, mô phỏng và kiểm tra. Đặc biệt, công cụ này cho phép các kỹ sư, nhà nghiên cứu và sinh viên tạo, chia sẻ và thử nghiệm các đoạn mã mô phỏng một cách dễ dàng và hiệu quả.

Với EDA Playground, người dùng có khả năng viết mã nguồn, mô phỏng, và kiểm tra các mạch điện tử ngay trên trình duyệt web, mà không cần phải cài đặt hoặc cấu hình các công cụ phức tạp trên máy tính cá nhân. Công cụ này hỗ trợ nhiều ngôn ngữ mô phỏng, bao gồm SystemVerilog, Verilog, VHDL và nhiều ngôn ngữ khác.

EDA Playground cung cấp một giao diện trực quan và dễ sử dụng, cho phép người dùng tạo các chương trình mô phỏng phức tạp từ các khối cơ bản, kiểm tra chức năng, và đối chiếu kết quả với các đặc tả kỹ thuật. Khả năng chia sẻ các mã mô phỏng và kết quả cho phép cộng đồng thiết kế điện tử tương tác và học hỏi lẫn nhau.

Với tất cả những ưu điểm vượt trội của mình, EDA Playground đã trở thành một công cụ không thể thiếu đối với quá trình thiết kế và kiểm tra hệ thống điện tử, giúp tiết kiệm thời gian và tăng hiệu suất trong quá trình phát triển sản phẩm điện tử.

#### 3.3.2 *Ưu điểm của EDA Playground*

Đồ án này đã áp dụng một trong những công cụ tiên tiến và mạnh mẽ trong lĩnh vực này, đó là EDA Playground, để thực hiện việc mô phỏng, kiểm tra và đánh giá hiệu suất của một hệ thống bus AHB.

Cùng với tính tiện lợi và khả năng mô phỏng trực tuyến, EDA Playground có một loạt các ưu điểm hấp dẫn khác, giúp làm tăng sự hấp dẫn và hiệu quả của công cụ này trong việc thiết kế và kiểm tra hệ thống điện tử:

- **Khả năng Trực Tuyến:** EDA Playground hoạt động trực tiếp trên trình duyệt web, không yêu cầu cài đặt hay cấu hình phức tạp. Người dùng có thể truy cập và sử dụng công cụ mọi lúc, mọi nơi, trên bất kỳ thiết bị nào có kết nối internet.
- **Tiết Kiệm Thời Gian:** Việc không cần cài đặt và cấu hình các phần mềm phức tạp trên máy tính cá nhân giúp tiết kiệm thời gian. Người dùng có thể bắt đầu viết mã và mô phỏng ngay lập tức mà không phải đổi mặt với các thủ tục khởi đầu phức tạp.
- **Dễ Dàng Chia Sẻ:** EDA Playground cho phép người dùng chia sẻ mã nguồn và kết quả mô phỏng một cách dễ dàng thông qua liên kết chia sẻ. Điều này giúp tạo ra một môi trường học tập và hợp tác hiệu quả trong cộng đồng thiết kế điện tử.
- **Hỗ Trợ Nhiều Ngôn Ngữ:** Công cụ này hỗ trợ nhiều ngôn ngữ mô phỏng phổ biến như SystemVerilog, Verilog, VHDL và nhiều ngôn ngữ khác. Điều này cho phép người dùng chọn ngôn ngữ phù hợp với nhu cầu cụ thể của họ.
- **Giao Diện Trực Quan:** Giao diện đồ họa thân thiện của EDA Playground giúp người dùng dễ dàng tạo, chỉnh sửa và quản lý các mô phỏng. Các chức năng được thể hiện một cách rõ ràng và dễ hiểu.
- **Kiểm Tra Nhanh Chóng:** Người dùng có thể nhanh chóng thực hiện kiểm tra chức năng và xác minh tính đúng đắn của mã nguồn mô phỏng thông qua việc tạo các testbench và assertion. Điều này giúp tiết kiệm thời gian trong quá trình phát triển.
- **Duyệt Lại Lịch Sử:** EDA Playground cho phép người dùng duyệt lại lịch sử các phiên làm việc trước đó, giúp theo dõi sự phát triển và điều chỉnh theo nhu cầu.
- **Tài Nguyên Tương Thích:** Công cụ hỗ trợ tất cả các tài liệu, ví dụ và hướng dẫn từ cộng đồng người dùng. Người dùng có thể tìm kiếm và chia sẻ kiến thức dễ dàng.

EDA Playground là một công cụ ưu việt và đáng giá để thực hiện việc mô phỏng, kiểm tra và học tập trong lĩnh vực thiết kế điện tử, mang lại lợi ích vượt xa việc sử dụng các phần mềm truyền thống.

### **3.4 Kết luận chương**

Chương 3 đã giới thiệu tổng quan về ngôn ngữ Verilog và System Veilog được sử dụng trong thiết kế số. Đồng thời cũng giới thiệu công cụ EDA Playground trực tuyến được sử dụng cho quá trình thực hiện đồ án.

## CHƯƠNG 4. XÂY DỰNG MÔ HÌNH HÓA HỆ THỐNG AHB

Chương 4 tập trung vào xây dựng và mô hình hóa hệ thống bus AHB. Trình bày mô hình dự án, từ cấu trúc tổng quan đến cách các thành phần tương tác. Tiếp theo, tập trung vào phân tích chi tiết từng phần, bao gồm bộ giải mã địa chỉ, bộ phân kênh tín hiệu và các bus. Mục tiêu là giúp độc giả hiểu rõ hơn về cấu trúc và hoạt động của hệ thống bus AHB trong dự án.

### 4.1 Bài toán thiết kế

#### 4.1.1 Mô hình đề xuất của bài toán

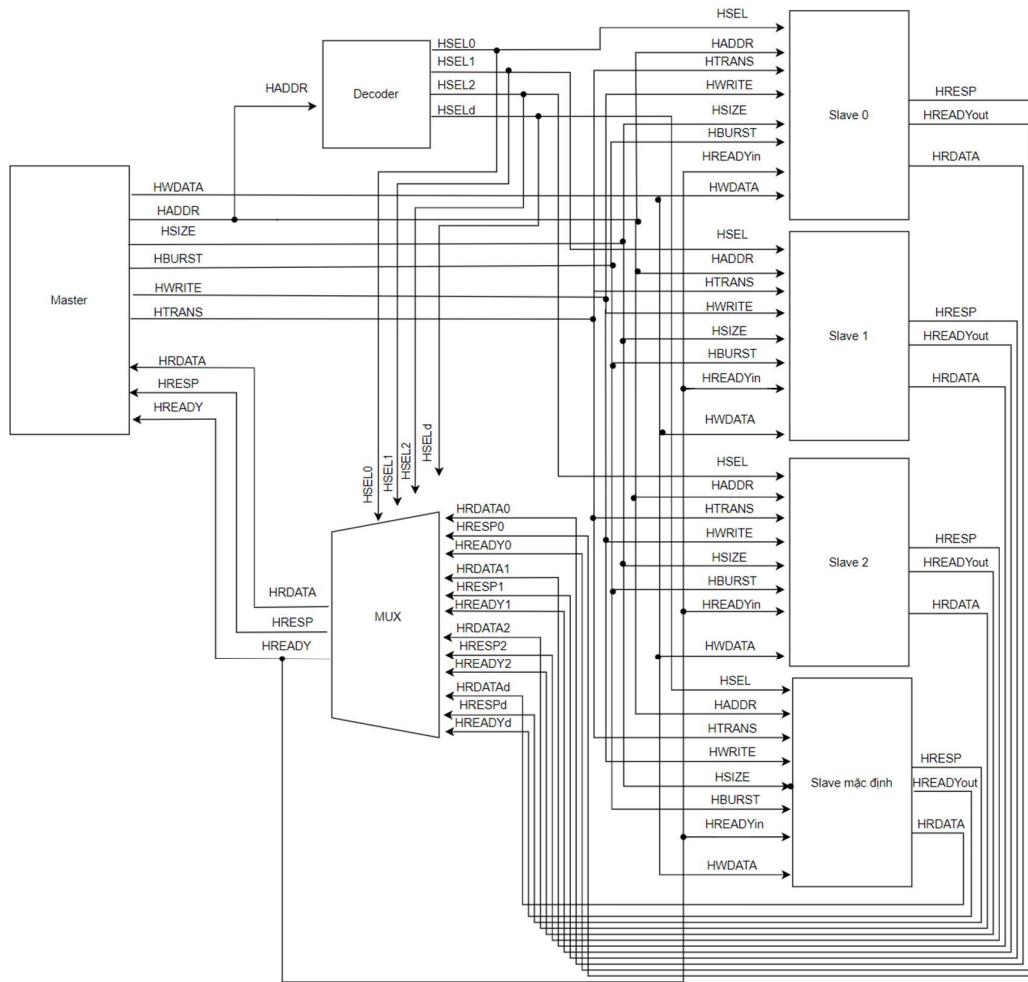
Để nghiên cứu và phát triển các ứng dụng kiểu hệ thống trên chip, việc xây dựng một nền tảng SoC (SoC platform) là vô cùng cần thiết. Phòng thí nghiệm SIS đang tập trung vào việc xây dựng một SoC có tên gọi là COMOSY (Controling Monitoring System) với mục tiêu phát triển các ứng dụng liên quan đến đa phương tiện, truyền thông và giám sát môi trường.

Nội dung đồ án tốt nghiệp tập trung vào việc nghiên cứu, thiết kế và triển khai một hệ thống bus AHB (Advanced High-performance Bus) nhằm đảm bảo yêu cầu về truyền thông trên chip cho hệ thống COMOSY. Đồng thời, hệ thống bus này cũng dễ dàng phát triển, mở rộng cho những mục đích truyền thông trên chip của các dự án tương tự. COMOSY là một hệ thống trên chip được phát triển để áp dụng trong các ứng dụng đa phương tiện, giám sát môi trường và có khả năng giao tiếp với mạng máy tính chuẩn Ethernet. Hệ thống COMOSY bao gồm một vi xử lý 32 bit đóng vai trò là trung tâm xử lý chính, cùng với một số lõi IP thực hiện các chức năng như quản lý bộ nhớ, giao tiếp Ethernet, giao tiếp LCD, giao tiếp vào/ra và trong tương lai có thể triển khai một khối mã hóa, giải mã chuẩn ảnh JPEG hoặc tín hiệu video.

Sau khi hoàn thành thiết kế và mô hình hóa, hệ thống sẽ được xây dựng bằng ngôn ngữ mô tả phần cứng Verilog ở mức truyền thanh ghi (RTL). Việc xây dựng mô hình hệ thống bus ở mức trừu tượng này không chỉ nhằm kiểm tra và đánh giá hoạt động của hệ thống, mà còn hỗ trợ quá trình tổng hợp phần cứng một cách thuận tiện hơn trong tương lai.

Qua các phân tích, đánh giá như trên, đồ án đề xuất mô hình kiến trúc của hệ thống bus AHB cần được xây dựng như sau (xem Hình 4.1):

- Hệ thống có một bus chủ: bus chủ sẽ nhận những yêu cầu thực hiện quá trình từ bên ngoài và gửi qua bus tớ để yêu cầu thực hiện các quá trình.
- Hệ thống gồm 4 bus tớ (ký hiệu lần lượt là: S0, S1, S2, Sd), trong đó có 3 bus tớ được phân vùng địa chỉ khác nhau thực hiện quá trình đọc ghi, và một bus tớ mặc định là Sd với vụng địa chỉ còn lại của 32 bit.
- Bus tớ được bổ sung thêm một tín hiệu mới có tên là HREADY\_in, ngoài các tín hiệu mặc định theo mô tả kỹ thuật của giao diện bus AMBA AHB. Tín hiệu HREADY\_in này được lấy từ đầu ra của bộ phân kênh tín hiệu từ bus tớ (bus slave) tới bus chủ (bus master). Mục đích của tín hiệu này là cho phép bus tớ giám sát quá trình truyền dữ liệu đang diễn ra và biết chính xác lúc nào nó được tham gia vào quá trình truyền dữ liệu.
- Việc triển khai các bộ phân kênh tín hiệu và bộ giải mã địa chỉ độc lập với nhau có những ưu điểm và nhược điểm riêng biệt. Mặc dù việc này có thể tăng số lượng tín hiệu trong hệ thống, nhưng nó giúp quản lý từng thành phần trong hệ thống trở nên dễ dàng hơn và đơn giản hóa việc mở rộng số lượng bus chủ và bus tớ mà không cần thay đổi cấu trúc của bộ xử lý bus và bộ giải mã địa chỉ. Ưu điểm của việc triển khai độc lập này là giảm đáng kể độ phức tạp so với việc tích hợp chung các thành phần này với nhau. Nhưng việc này cũng có nhược điểm là tăng số lượng tín hiệu trong hệ thống. Tuy nhiên, những ưu điểm về dễ quản lý và khả năng mở rộng linh hoạt có thể có giá trị quan trọng hơn so với nhược điểm này.
- Đò án này đề xuất thêm các tín hiệu giao tiếp bắt tay vào hệ thống bus để dễ dàng giao tiếp giữa các lõi IP và hệ thống bus, đồng thời đáp ứng yêu cầu hệ thống bus có khả năng tương thích với nhiều loại lõi IP khác nhau. Việc thêm các tín hiệu giao tiếp này giúp hệ thống bus AHB trở nên độc lập với lõi IP và khi tích hợp một lõi IP vào hệ thống bus, không cần phải thay đổi cấu trúc của hệ thống. Bus chủ sẽ giao tiếp với lõi IP bằng những tín hiệu giao tiếp ngoài và thực hiện quá trình truyền đọc ghi vào bus tớ.

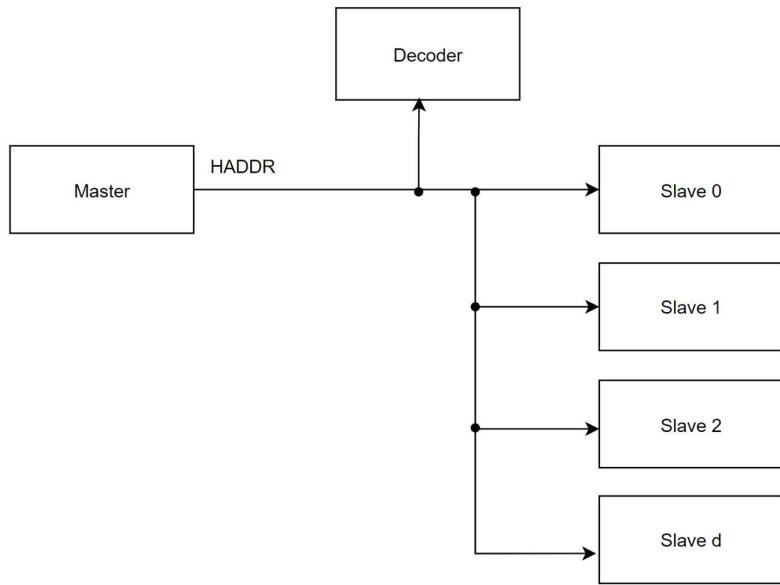


**Hình 4.1 Mô hình hệ thống bus AHB được đề xuất.**

Hình 4.1 thể hiện một mô hình đề xuất của hệ thống AHB, bao gồm một bus chủ, một bộ giải mã địa chỉ (decoder), một bộ phân kênh tín hiệu (mux) và bốn bus tớ. Mô hình này mô tả cách các thành phần trong hệ thống AHB tương tác với nhau để thực hiện việc truyền dữ liệu và điều khiển giữa bus chủ và các bus tớ thông qua một giao diện chung. Bus tớ chịu trách nhiệm gửi yêu cầu truyền dữ liệu, bộ giải mã địa chỉ quản lý việc chọn bus tớ tương ứng dựa trên địa chỉ, và multiplexer định tuyến tín hiệu giữa bus chủ và bus tớ tương ứng. Điều này tạo ra một cơ chế hoạt động cơ bản cho hệ thống AHB, cho phép truyền thông tin và điều khiển hiệu quả giữa các thành phần khác nhau.

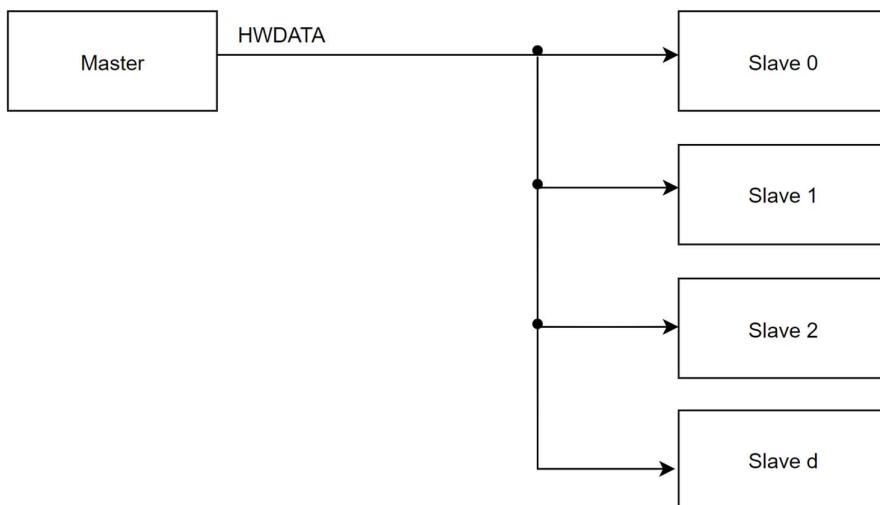
#### 4.1.2 Đường dữ liệu và đường địa chỉ trên mô hình

Trong mô hình đề xuất trên hình 4.1, bài toán giải quyết vấn đề truyền dữ liệu kết nối giữa 1 bus chủ và 4 bus tớ. Từ mô hình đề xuất ở trên, ta có thể phân chia thành kết nối đường địa chỉ và đường dữ liệu như sau:



**Hình 4.2 Đường dẫn tín hiệu địa chỉ trong mô hình**

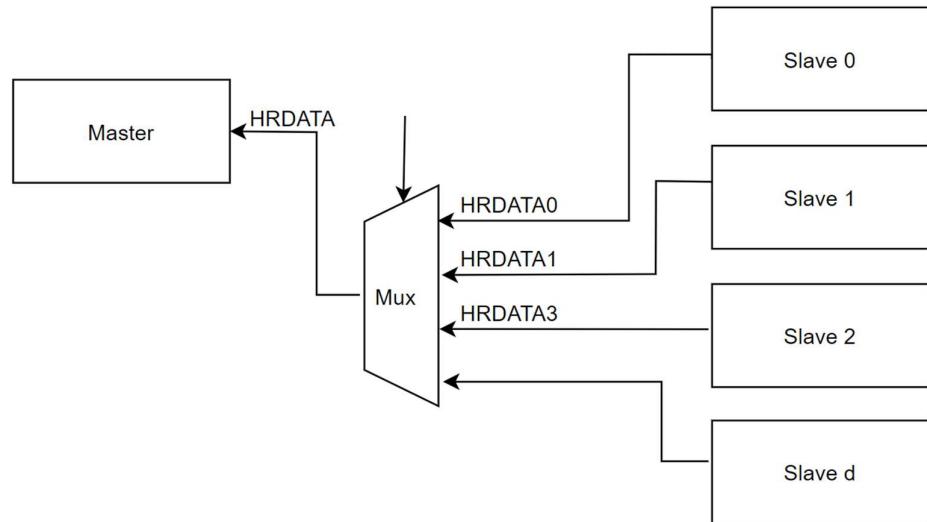
Theo hình 4.2 ở trên, nhìn chung tín hiệu địa chỉ sẽ được gửi từ bus tớ theo 2 đường khác nhau. Một đường tín hiệu địa chỉ sẽ được gửi từ bus chủ đến bộ giải mã địa chỉ, ở đây bộ giải mã sẽ quyết định xem bus nào được tham gia vào quá trình đọc ghi. Đường còn lại, tín hiệu địa chỉ sẽ được gửi từ bus chủ đến 4 bus tớ để cho bus tớ biết địa chỉ của quá trình truyền dữ liệu.



**Hình 4.3 Đường dẫn tín hiệu dữ liệu ghi**

Khác với tín hiệu địa chỉ, tín hiệu dữ liệu đọc này sẽ được gửi thẳng từ bus chủ đến 4 bus tớ để truyền dữ liệu ghi cho quá trình ghi. Ở đây chỉ có bus nào được bộ

giải mã gửi tín hiệu HSEL cho biết là bus tớ được chọn mới thực hiện quá trình ghi vào trong bộ nhớ của bus tớ tương ứng đó. Đường dẫn dữ liệu đọc sẽ được thể hiện như hình 4.3 ở trên kia.

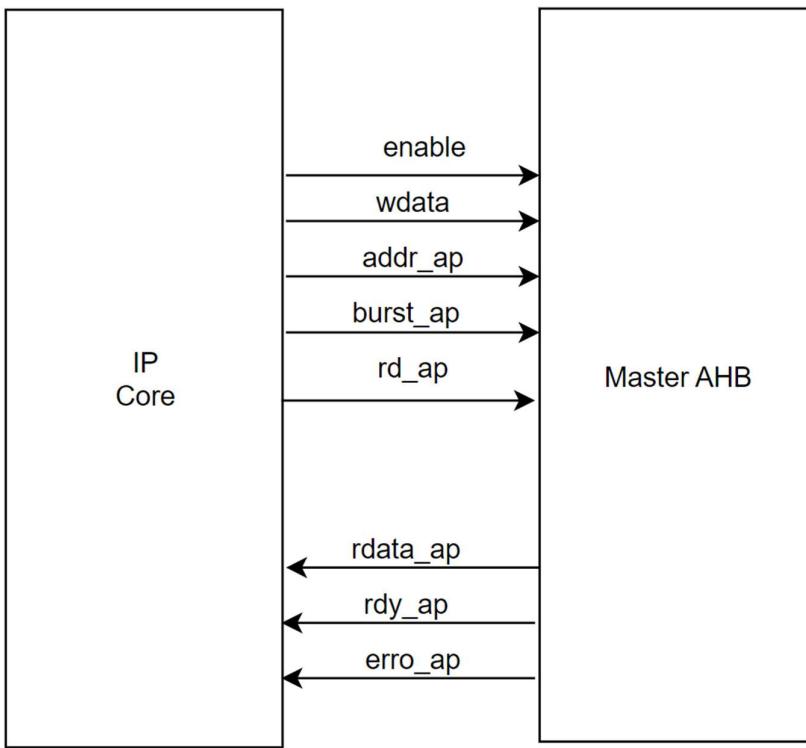


**Hình 4.4 Đường dẫn tín hiệu dữ liệu đọc**

Khác với tín hiệu dữ liệu ghi, tín hiệu dữ liệu đọc sẽ được gửi từ bus tớ đến một bộ phân kênh tín hiệu (Mux). Ở đây bộ phân kênh tín hiệu sẽ sử dụng tín hiệu điều khiển gửi từ bộ giải mã địa chỉ để quyết định xem tín hiệu đọc của bus tớ nào sẽ được gửi về bus chủ. Hình 4.4 bên trên đã thể hiện được đường dẫn tín hiệu dữ liệu đọc được đi như thế nào.

#### 4.2 Mô hình giao tiếp giữa bus chủ và lõi IP

Để thực hiện việc giao tiếp giữa bus chủ và lõi IP, mục đích là để lõi IP đưa các tín hiệu điều khiển từ bên ngoài vào bus chủ để thực thi các quá trình đọc ghi. Trong đề án này, sau khi tham khảo việc giao tiếp giữa bus chủ và lõi IP qua các tài liệu có liên quan. Đề án đề xuất ngoài các tín hiệu điều khiển và phản hồi kết nối trong của bus chủ, có thêm một tín hiệu enable cho biết quá trình được bắt đầu. Khi tín hiệu này ở mức cao, sẽ đồng ý cho bus chủ tiếp nhận các tín hiệu điều khiển đưa vào để thực thi quá trình. Khi tín hiệu này xuống mức thấp sẽ bus tớ sẽ thực hiện nốt quá trình đang tiến hành và sau đó sẽ không cho phép bus chủ nhận thêm tín hiệu điều khiển từ bên ngoài.



**Hình 4.5 Các tín hiệu giao tiếp giữa bus chủ và lõi IP**

Để yêu cầu một quá trình truyền trên hệ thống bus, lõi IP cần đẩy tín hiệu enable từ thấp lên cao. Sau đó dữ liệu và các tín hiệu điều khiển sẽ được bus chủ tiếp nhận. Bus chủ sẽ xử lý các tín hiệu điều khiển được đưa vào và gửi các hiệu sang phía bus tò để yêu thực hiện quá trình theo thứ tự và tiêu chuẩn được đề cập trong bản thiết kế đặc tả kỹ thuật của giao thức AHB.

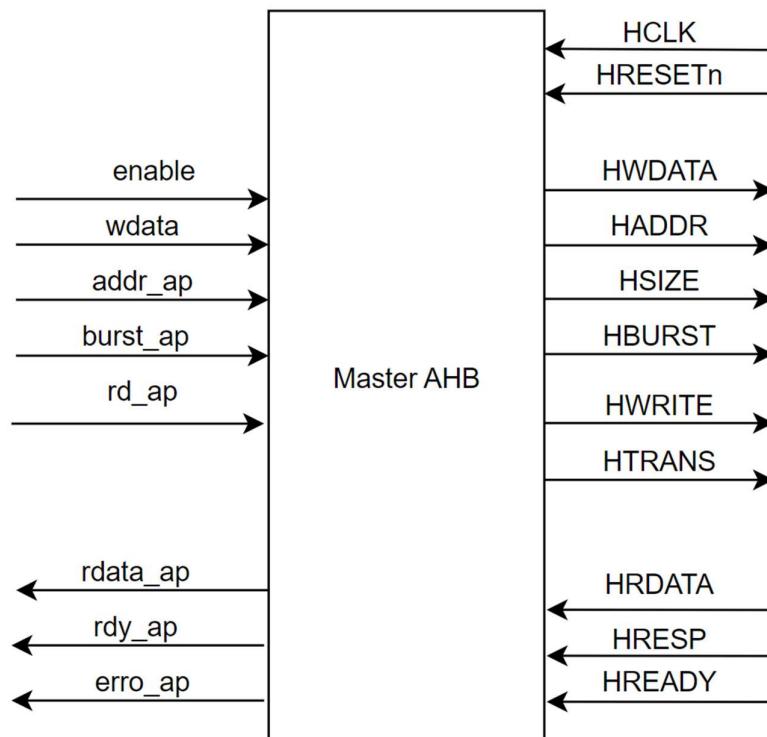
Lõi IP sẽ cung cấp hai tín hiệu quan trọng: tín hiệu địa chỉ `addr_ap` và tín hiệu thông báo loại quá trình, tức là quá trình đọc (được biểu thị bằng tín hiệu `RW_AP` có giá trị 0) hoặc quá trình ghi (được biểu thị bằng tín hiệu `RW_AP` có giá trị 1).

Khi thực hiện quá trình đọc hoặc ghi, lõi IP sẽ truyền dữ liệu tương ứng cho bus chủ thông qua hai tín hiệu trên: nếu là quá trình đọc, dữ liệu đọc sẽ được đưa lên bus chủ qua tín hiệu `rdata_ap`, và nếu là quá trình ghi, dữ liệu ghi sẽ được lõi IP cung cấp cho bus chủ qua tín hiệu `wdata`. Các tín hiệu này, cùng với tín hiệu địa chỉ `ADDR_AP`, sẽ được gửi từ lõi IP tới bus chủ để thực hiện các hoạt động trên bus. Tuy nhiên, việc xử lý quá trình truyền dữ liệu sau đó sẽ do bus chủ đảm trách, bao gồm việc điều khiển các tín hiệu và đồng bộ hóa quá trình truyền giữa lõi IP và bus chủ.

## 4.3 Bus chủ

### 4.3.1 Các tín hiệu trên bus chủ

Sau khi thêm các tín hiệu bắt tay với lõi IP, bus chủ sẽ bao gồm một số tín hiệu được biểu diễn như trong Hình 4.6. Các tín hiệu này có thể được chia thành hai nhóm chính: nhóm tín hiệu giao tiếp ngoài (các tín hiệu kết nối với lõi IP) và nhóm tín hiệu kết nối trong (giao tiếp giữa bus chủ và các thành phần khác trong hệ thống bus). Cấu trúc, chức năng và hoạt động của các tín hiệu này được mô tả cụ thể như sau:



Hình 4.6 Các tín hiệu trên bus chủ

a) Nhóm tín hiệu giao tiếp ngoài:

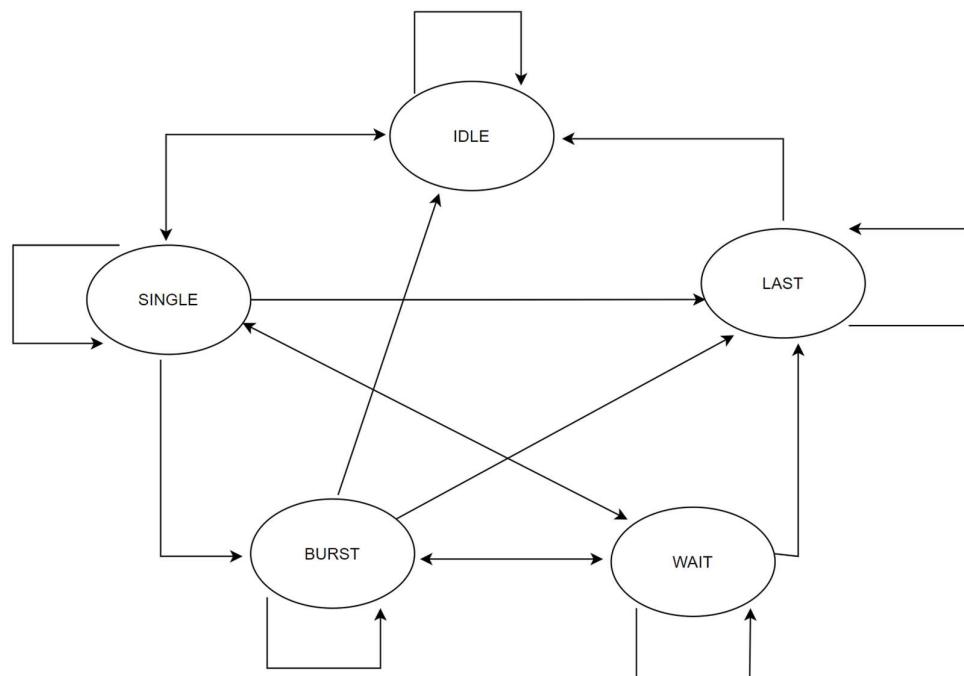
- ENABLE (lõi vào): Tín hiệu có độ rộng 1 bit, được lõi IP sử dụng để thông báo cho phép một quá trình truyền. Tín hiệu này ở mức cao khi lõi IP yêu cầu một quá trình truyền và ở mức thấp khi lõi IP hết quá trình thực thi. Sau một quá trình truyền, nếu lõi IP vẫn muốn thực hiện tiếp quá trình truyền sẽ để tín hiệu này ở mức cao cho đến khi không muốn thực hiện thêm quá trình nào nữa.
- WDATA\_AP (lõi vào): Tín hiệu này có độ rộng 32 bit, được lõi IP sử dụng để truyền đi dữ liệu cần được ghi.
- ADDR\_AP (lõi vào): Tín hiệu này độ rộng 32 bit, được lõi IP sử dụng để gửi đi tín hiệu địa chỉ của quá trình cần thực hiện.

- BURST\_AP (lối vào): Tín hiệu này có độ rộng 3 bit, được lối IP sử dụng để thông báo kiểu truyền cần thực hiện.
  - RW\_AP (lối vào): Tín hiệu này có độ rộng 1 bit, được lối IP sử dụng để thông báo quá trình truyền sẽ là đọc hay ghi.
  - RDATA\_AP (lối ra): Tín hiệu này có độ rộng 32 bit, được bus chủ sử dụng để trả lại lối IP dữ liệu của quá trình đọc.
  - RDY\_AP (lối ra): Tín hiệu này có độ rộng 1 bit, được bus chủ sử dụng để thông báo cho lối IP biết khi nào tiếp nhận dữ liệu cho quá trình ghi.
  - ERRO\_AP (lối ra): Tín hiệu này có độ rộng 1 bit, được bus chủ sử dụng để thông báo lỗi quá trình truyền. Tín hiệu này ở mức cao thông báo quá trình truyền có lỗi, và ở mức thấp nếu quá trình không có lỗi.
- b) Nhóm tín hiệu kết nối trong:
- HCLK (lối vào): Tín hiệu xung đồng bộ cho mọi thành phần bên trong hệ thống bus.
  - HRESETn (lối vào): Tín hiệu reset có độ rộng 1 bit, được sử dụng để đưa hệ thống về trạng thái khởi đầu ban đầu. Đây là tín hiệu duy nhất trong hệ thống có mức tín hiệu tích cực thấp (active low).
  - HWDATA (lối ra): Tín hiệu có độ rộng 32 bit, được bus chủ sử dụng để truyền tín hiệu dữ liệu ghi đến bus tó.
  - HADDR[31:0] (lối ra): Tín hiệu có độ rộng 32 bit, được bus chủ sử dụng để truyền tín hiệu địa chỉ lên bus.
  - HRDATA (lối vào): Tín hiệu có độ rộng 32 bit, được bus chủ sử dụng để đọc tín hiệu dữ liệu đọc từ bus tó truyền đến.
  - HSIZE (lối ra): Tín hiệu độ rộng 3 bit, được bus chủ sử dụng để thông báo cho bus tó độ rộng dữ liệu trong quá trình truyền.
  - HBURST[2:0] (lối ra): Tín hiệu có độ rộng 3 bit. Tín hiệu này được bus chủ sử dụng để thông báo về loại truyền khói (burst) được thực hiện.
  - HWRITE (lối ra): Tín hiệu có độ rộng 1 bit, được sử dụng để thông báo cho bus tó biết quá trình truyền là một quá trình đọc (nếu HWRITE ở mức thấp) hoặc là một quá trình ghi (nếu HWRITE ở mức cao). Giá trị của HWRITE được xác định từ tín hiệu RW\_AP và từ loại truyền khói đang được thực hiện, và nó được lấy mẫu tại pha địa chỉ của quá trình truyền.
  - HTRANS[1:0] (lối ra): Tín hiệu có độ rộng 2 bit và được bus chủ sử dụng để thông báo về các kiểu truyền (transfer types) đang được thực hiện trên bus.

- HRESP[1:0] (lối vào): Tín hiệu có độ rộng 2 bit. Đây là tín hiệu phản hồi truyền của bus tớ, thông báo về trạng thái truyền hiện tại của bus tớ.
- HREADY (lối vào): Tín hiệu có độ rộng 1 bit, được bus tớ sử dụng để thông báo về trạng thái của quá trình truyền hiện tại. Khi một quá trình truyền cần mở rộng pha dữ liệu, bus tớ sẽ chuyển HREADY về mức thấp. Khi pha dữ liệu của quá trình truyền đó được hoàn tất, HREADY sẽ được chuyển lại về mức cao để tiếp tục quá trình truyền tiếp theo.

#### 4.3.2 Máy trạng thái mô tả hoạt động của bus chủ

Để dễ dàng mô tả cấu trúc của bus chủ ở mức truyền thanh ghi và có thể điều khiển hoạt động của hệ thống trong từng chu kỳ xung nhịp, hoạt động của bus chủ được mô tả dưới dạng một máy trạng thái bao gồm 5 trạng thái sau: IDLE, SINGLE, BURST, WAIT, LAST. Việc chuyển các trạng thái sẽ được xác định bởi các tín hiệu lối vào trên bus chủ và bởi chính trạng thái hiện tại của máy trạng thái. Mối liên hệ giữa các trạng thái được minh họa bởi lưu đồ như trong Hình 4.7.



Hình 4.7 Sơ đồ máy trạng thái của bus chủ.

Chức năng và hoạt động của mỗi trạng thái có thể được mô tả như sau:

- IDLE: trạng thái này được kích hoạt khi tín hiệu HRESETn ở mức thấp hoặc khi các quá trình truyền kết thúc. Đây là trạng thái khởi đầu cho một hoạt động mới của bus. Trạng thái này đặt lại các tín hiệu về giá trị mặc định khi hệ thống được khởi động.
- SINGLE: trạng thái này tương pha địa chỉ khi bắt đầu một quá trình truyền nếu trạng thái trước đó là IDLE. Trạng thái này cũng tương ứng với pha dữ liệu của quá trình mới và pha địa chỉ của quá trình cũ, nếu trạng thái trước đó là trạng thái SINGLE. Ở trạng thái này, địa chỉ sẽ được lấy mẫu từ ADDR\_AP và sau đó truyền đi cùng các tín hiệu điều khiển. Dữ liệu ghi cũng sẽ được gửi đi và dữ liệu đọc cũng sẽ được nhận về nếu trước đó đã thực hiện một pha địa chỉ.
- BURST: trạng thái này tương ứng với một pha của quá trình truyền khối, địa chỉ sẽ không được lấy mẫu tại trạng thái này. Ở trạng thái này, địa chỉ mới sẽ được sinh ra phụ thuộc vào địa chỉ trước đó và được gửi đi cùng các tín hiệu điều khiển. Dữ liệu ghi cũng sẽ được gửi đi nếu là quá trình ghi khối, dữ liệu đọc về sẽ được lấy về nếu là quá trình đọc khối.
- WAIT: trạng thái này là trạng thái chờ trong quá trình truyền, trạng thái này sẽ xuất hiện khi tín hiệu HREADY bằng 0, và trở lại các trạng thái tương ứng nếu HREADY bằng 1.
- LAST: trạng thái này là trạng thái tương ứng với pha dữ liệu cuối cùng nếu sau nó không còn thêm quá trình nào nữa. Trạng thái này cũng là trạng thái pha dữ liệu cuối cùng của một quá trình truyền khối.

Điều kiện chuyển các trạng thái trong máy trạng thái của bus chủ được mô tả chi tiết dưới đây:

- Khi trạng thái hiện tại đang ở IDLE, nếu có tín hiệu enable=1 và các tín hiệu điều khiển khác được truyền từ lõi IP vào bus chủ thì trạng thái tiếp theo sẽ là SINGLE. Nếu không có các tín hiệu trên thì trạng thái vẫn giữ nguyên tại IDLE.
- Khi trạng thái hiện tại đang ở SINGLE, sẽ có 2 trường hợp xảy ra với enable tương ứng với điều sau vẫn còn quá trình truyền hoặc không:
  - o Nếu điều sau vẫn còn quá trình truyền, trạng thái tiếp theo sẽ là SINGLE hoặc BURST phụ thuộc vào tín hiệu điều khiển IN\_BURST. Trạng thái tiếp theo cũng có thể là WAIT nếu như tín hiệu HREADY ở mức thấp.
  - o Nếu sau đó không còn quá trình, thì trạng thái tiếp theo sẽ là trạng thái LAST.
- Khi trạng thái hiện tại ở BURST, trạng thái tiếp theo cũng sẽ được giữ nguyên ở BURST nếu chưa thực hiện xong số nhịp của truyền khối, và sẽ được gửi đến

trạng thái LAST nếu quá trình truyền khói đã thực hiện xong. Trạng thái này cũng sẽ được gửi đến trạng thái WAIT nếu tín hiệu HREADY ở mức thấp.

- Khi trạng thái hiện tại ở WAIT, trạng thái tiếp theo sẽ là BURST nếu quá trình truyền khói đang chờ thực hiện nốt, trạng thái tiếp theo sẽ là SINGLE nếu trạng thái trước đó là SINGLE và còn quá trình cần thực hiện ở tiếp theo. Trạng thái tiếp theo cũng có thể là LAST nếu không còn quá trình nào cần thực hiện ở sau đó.
- Khi trạng thái hiện tại ở LAST, sau khi pha địa chỉ được thực hiện xong, trạng thái tiếp theo sẽ là IDLE. Kết thúc một vòng quá trình truyền và đợi yêu cầu mới cần thực thi.

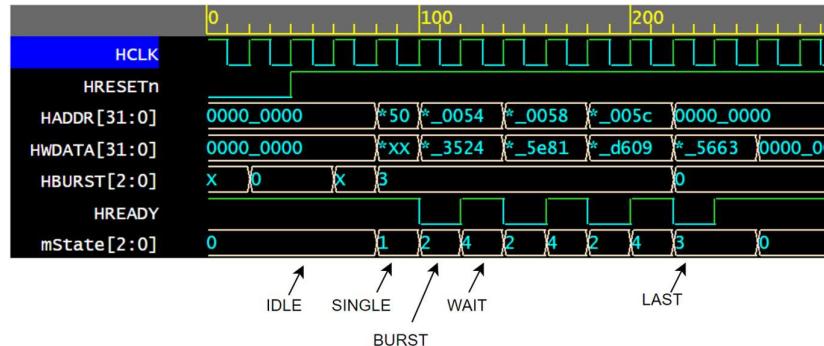
Trong quá trình thực hiện triển khai chuyển trạng thái của bus chủ, cần sử dụng 3 bit để mã hóa 5 trạng thái IDLE, SINGLE, BURST, LAST, WAIT như bảng 4.1 dưới đây:

**Bảng 4.1 Bảng giá trị mã hóa các trạng thái**

Trạng thái	Giá trị mã hóa hệ nhị phân	Giá trị mã hóa hệ hexa
IDLE	000	0
SINGLE	001	1
BURST	010	2
LAST	011	3
WAIT	100	4

Trong quá trình truyền khói có trạng thái đợi, sự chuyển trạng thái của bus chủ được thể hiện như hình 4.8 dưới đây. Trạng thái đầu tiên mặc định của bus chủ sẽ là trạng thái IDLE. Sau khi có thông tin yêu cầu một quá trình truyền dữ liệu, trạng thái của bus chủ sẽ được chuyển tới trạng thái SINGLE, tại đây tín hiệu địa chỉ sẽ được lấy mẫu. Sau đó sẽ được chuyển sang trạng thái BURST. Ở trạng thái này gắp tín hiệu HREADY ở mức thấp nên chu kỳ sau trạng thái sẽ được chuyển đến trạng thái WAIT và khi gắp HREADY ở mức cao sẽ trở về trạng thái BURST để tiếp tục quá trình truyền khói. Cho đến khi quá trình truyền khói kết thúc, trạng thái của bus chủ sẽ chuyển đến

trạng thái LAST để thực hiện truyền dữ liệu ghi lần cuối và sau đó sẽ được chuyển về trạng thái IDLE để chờ quá trình ghi tiếp theo.

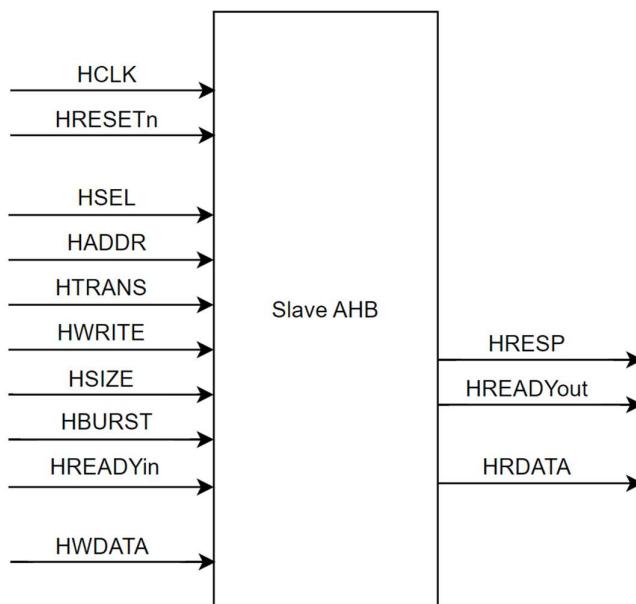


**Hình 4.8 Biểu diễn các trạng thái tương ứng với quá trình truyền khối**

## 4.4 Bus tớ

### 4.4.1 Các tín hiệu trên bus tớ

Khác với bus chủ, đồ án này đề xuất mô hình ở bus tớ không có giao tiếp tín hiệu với lõi IP, mà chỉ giao tiếp với các thành phần bên trong hệ thống. Cấu trúc, chức năng và hoạt động của các tín hiệu này cụ thể như sau.



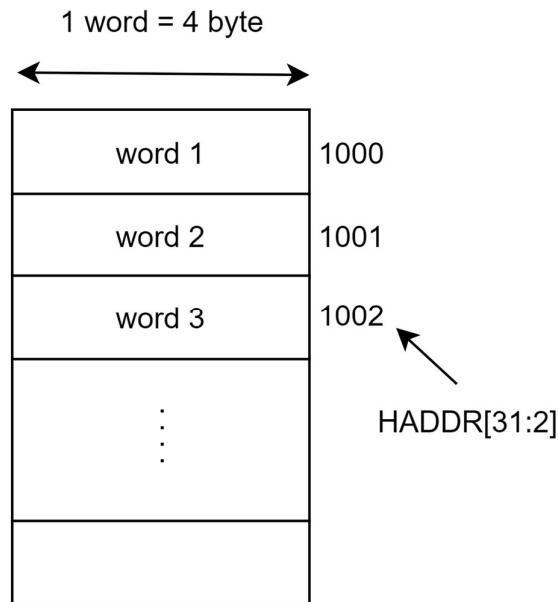
**Hình 4.9 Các tín hiệu giao tiếp trên bus tớ**

- HSELx (lối vào): Tín hiệu lựa chọn bus tớ có độ rộng 1 bit. Bộ giải mã sử dụng tín hiệu này để chọn bus tớ tương ứng tham gia vào quá trình truyền. Tín hiệu này ở mức tích cực cao.
- HWRITE (lối vào): Tín hiệu có độ rộng 1 bit, thông báo về quá trình đọc hoặc ghi dữ liệu. Bus chủ sử dụng tín hiệu này để thông báo cho bus tớ biết quá trình hiện đang thực hiện là quá trình đọc hay quá trình ghi dữ liệu.
- HRESETn (lối vào): Tín hiệu có mức tích cực thấp và có độ rộng bằng 1 bit. Tín hiệu này được sử dụng để đưa hệ thống về trạng thái khởi đầu.
- HCLK (lối vào): Tín hiệu xung nhịp đồng bộ cho toàn hệ thống.
- HTRANS[1:0] (lối vào): Tín hiệu có độ rộng 2 bit, thông báo kiểu truyền từ bus chủ.
- HADDR[31:0] (lối vào): Tín hiệu địa chỉ từ bus chủ, có độ rộng 32 bit.
- HWDATA[31:0] (lối vào): Tín hiệu dữ liệu ghi từ bus chủ tới bus tớ, có độ rộng 32 bit.
- HREADY (lối ra): Tín hiệu có độ rộng 1 bit. Tín hiệu này được sử dụng để thông báo cho bus tớ cần mở rộng pha dữ liệu để xử lý. Quá trình truyền chỉ kết thúc khi tín hiệu này ở mức cao. Tín hiệu này được xác định từ tín hiệu tương ứng IP\_OKAY của bus tớ tham gia vào quá trình truyền dữ liệu.
- HRESP[1:0] (lối vào): Tín hiệu phản hồi truyền từ bus tớ, có độ rộng 2 bit. Tín hiệu này được nối với bus chủ để yêu cầu các quá trình truyền OKAY và ERROR khi cần thiết.
- HRDATA[31:0] (lối ra): Tín hiệu dữ liệu đọc từ bus tớ tới bus chủ, có độ rộng 32 bit.
- HREADY\_in (lối vào): Tín hiệu báo tình trạng sẵn sàng truyền từ các bus tớ khác. Tín hiệu này giúp bus tớ biết được lúc nào quá trình truyền của các bus tớ khác đã hoàn tất để có thể tham gia vào quá trình truyền của mình.
- HBURST (lối vào): Tín hiệu có độ rộng 3 bit, được sử dụng để thông báo cho bus tớ biết loại quá trình truyền.
- HSIZE (lối vào): Tín hiệu có độ rộng 3 bit, được sử dụng để thông báo cho bus tớ biết độ rộng kiểu dữ liệu được ghi và đọc.

#### **4.4.2 Hoạt động của bus tớ**

Trong mô hình được mô tả trong đồ án, công việc đối diện với việc tạo liên kết giao tiếp giữa một bus chủ và bốn bus tớ khác nhau. Trong số này, có ba bus tớ (bus tớ 0, 1, 3) đặc trưng bởi việc chúng chứa các mảng dữ liệu để lưu trữ thông tin, trong khi

bus tó thứ tư là bus tó mặc định không có mảng dữ liệu. Chức năng chính của các bus tó là nhận và xử lý tín hiệu yêu cầu từ bus chủ và sau đó trả về dữ liệu đã được yêu cầu hoặc thực hiện ghi dữ liệu vào mảng tương ứng.



**Hình 4.10 Sơ đồ minh họa mảng trong bus tó**

Dựa vào hình 4.10, chúng ta có một khái niệm quan trọng về cách dữ liệu được tổ chức và truy cập trong bus tó. Trong mô hình này, bus tó được thiết kế để chứa một mảng dữ liệu, và mỗi phần tử trong mảng này được gọi là "word". Mỗi word có độ rộng là 32 bit, tương ứng với 4 byte thông tin. Khi chúng ta đề cập đến việc truy cập dữ liệu trong bus tó, việc xác định địa chỉ của một word là vô cùng quan trọng. Trong trường hợp này, địa chỉ của một word được xác định bởi các bit từ thứ 3 đến thứ 32 của tín hiệu HADDR từ bus chủ. Khi bus chủ có yêu cầu truyền dữ liệu đến một bus tó cụ thể, tín hiệu yêu cầu sẽ được gửi đến bus tó đó. Bus tó nhận tín hiệu yêu cầu và xác định xem liệu nó có thể xử lý yêu cầu đó hay không. Nếu có thể, bus tó sẽ thực hiện các thao tác cần thiết để đọc dữ liệu từ mảng (nếu yêu cầu là đọc) hoặc ghi dữ liệu vào mảng (nếu yêu cầu là ghi).

Mỗi bus tó có một vùng địa chỉ riêng biệt, được quản lý bởi bộ giải mã địa chỉ. Bộ giải mã địa chỉ sẽ xác định xem địa chỉ nào thuộc về bus tó nào. Khi bus tó nhận được

tín hiệu yêu cầu, nó sẽ kiểm tra xem địa chỉ có nằm trong vùng địa chỉ của mình hay không. Nếu đúng, bus tớ sẽ tiếp tục xử lý yêu cầu; nếu sai, nó sẽ bỏ qua yêu cầu đó.

Trong trường hợp bus tớ có chứa mảng dữ liệu, khi có yêu cầu đọc từ bus chủ, bus tớ sẽ truy xuất dữ liệu từ mảng tương ứng và gửi lại dữ liệu này cho bus chủ. Nếu có yêu cầu ghi từ bus chủ, bus tớ sẽ thực hiện các thao tác cần thiết để ghi dữ liệu vào mảng. Điều này có thể bao gồm việc xác định vị trí trong mảng cần ghi dữ liệu, kiểm tra các tín hiệu điều khiển như 'HSEL' và 'HREADY' để đảm bảo tính đồng bộ và toàn vẹn của dữ liệu.

Bus tớ cũng phải quản lý việc truyền các tín hiệu phản hồi như 'HRESP' và 'HREADY' trở lại cho bus chủ. Tín hiệu 'HRESP' sẽ chỉ ra kết quả của yêu cầu truyền, ví dụ như thành công hoặc thất bại. Tín hiệu 'HREADY' sẽ cho biết bus tớ đã sẵn sàng để truyền dữ liệu hoặc nhận dữ liệu.

Tóm lại, mỗi bus tớ trong mô hình này hoạt động như một đơn vị giao tiếp độc lập với khả năng nhận và xử lý các yêu cầu từ bus chủ, truy xuất dữ liệu từ mảng hoặc ghi dữ liệu vào mảng, và gửi lại các tín hiệu phản hồi cho bus chủ để báo cáo kết quả của yêu cầu. Qua đó, mô hình đảm bảo tính toàn vẹn, đồng bộ và hiệu suất cao trong việc truyền tải dữ liệu giữa các thành phần khác nhau của hệ thống.

#### **4.4.3 Phân loại bus tớ và vùng địa chỉ hoạt động các bus tớ**

Theo mô hình đề xuất, đồ án này đã phân loại ra bus tớ mặc định và các bus tớ khác. Trong đó, bus tớ 0 và bus tớ 1 sẽ thể hiện truyền dữ liệu với không trạng thái chờ, còn bus tớ 2 sẽ truyền dữ liệu có trạng thái chờ được thiết lập qua tín hiệu HREADY\_out. Sự khác biệt của bus tớ mặc định so với các bus tớ khác sẽ thể hiện ở các điểm sau:

- Bus tớ mặc định sẽ không có mảng dữ liệu được thiết lập sẵn.
- Thiết kế đã phân vùng địa chỉ cho bus tớ 0, bus tớ 1 và bus tớ 2 trong vùng 32 bit địa chỉ. Vùng địa chỉ của bus tớ mặc định sẽ là vùng còn lại của 3 bus tớ đã được thiết lập.
- Khi bus chủ yêu cầu quá trình truyền với địa chỉ thuộc bus tớ mặc định, sau đó bus chủ vẫn muốn thực hiện tiếp quá trình truyền dữ liệu thì bus tớ mặc định sẽ gửi lại phản hồi lỗi, yêu cầu kết thúc quá trình.

Trong thiết kế, vùng địa chỉ hoạt động của các bus tớ đã được phân chia theo hình 4.11 dưới đây.

```
HSEL0: from 0x0000_0000 to 0x00ff_FFFF  
HSEL1: from 0x1000_0000 to 0x10ff_FFFF  
HSEL2: from 0x2000_0000 to 0x20ff_FFFF
```

### Hình 4.11 Vùng địa chỉ của các bus tớ

Hình ảnh 4.11 thể hiện việc phân chia vùng địa chỉ giữa ba bus tớ khác nhau (bus tớ 0, bus tớ 1 và bus tớ 2) cùng với vùng địa chỉ của bus tớ mặc định. Mỗi bus tớ có một phạm vi địa chỉ riêng mà nó có thể truy cập và quản lý dữ liệu.

- Bus tớ 0:
  - Địa chỉ bắt đầu: 0x00000000
  - Địa chỉ kết thúc: 0x00ffff
- Bus tớ 1:
  - Địa chỉ bắt đầu: 0x10000000
  - Địa chỉ kết thúc: 0x10ffff
- Bus tớ 2:
  - Địa chỉ bắt đầu: 0x20000000
  - Địa chỉ kết thúc: 0x20ffff

Vùng địa chỉ của bus tớ mặc định sẽ là phần còn lại của không gian địa chỉ 32 bit sau khi đã trừ đi vùng địa chỉ của ba bus tớ trên. Điều này có nghĩa là vùng địa chỉ của bus tớ mặc định sẽ nằm trong khoảng từ không chứa địa chỉ của 3 bus tớ trên.

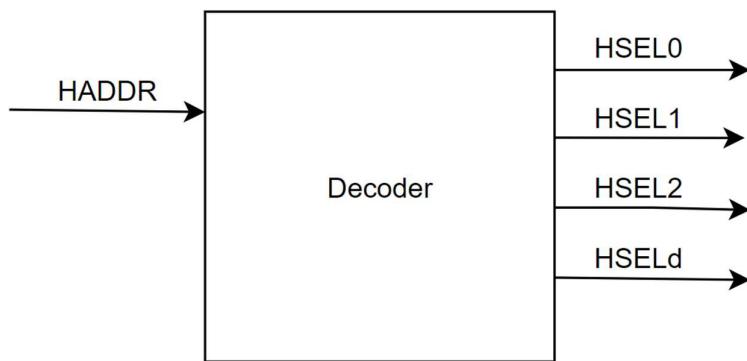
Sự phân chia chính xác này cho phép mỗi bus tớ có quyền truy cập vào một phạm vi địa chỉ cụ thể, và các vùng địa chỉ này không giao nhau, giúp đảm bảo tính toàn vẹn và chính xác khi truyền tải dữ liệu giữa các thành phần trong hệ thống.

## 4.5 Bộ giải mã địa chỉ và bộ phân kenh tín hiệu

### 4.5.1 Bộ giải mã địa chỉ

Bộ giải mã địa (Decoder) chỉ có trách nhiệm nhận địa chỉ từ bus chủ được cấp quyền truy cập bus và thực hiện giải mã để xác định các bus tớ nào sẽ cùng tham gia vào quá trình truyền với bus chủ. Mỗi bus tớ tương ứng sẽ được lựa chọn thông qua tín hiệu HSELx tương ứng. Khi tín hiệu HSELx có giá trị logic cao, bus tớ tương ứng bắt đầu hoạt động và thực hiện quá trình truyền dữ liệu.

Cấu trúc của bộ giải mã địa chỉ được mô tả như trong Hình 3.6. Nó nhận địa chỉ từ bus chủ và sau đó thông qua quá trình giải mã, xác định các bus tớ tham gia vào quá trình truyền dữ liệu. Mỗi tín hiệu HSEL<sub>x</sub> sẽ tương ứng với một bus tớ và chỉ khi nào tín hiệu HSEL<sub>x</sub> có giá trị logic cao, thì bus tớ tương ứng sẽ được chọn tham gia vào quá trình truyền. Các tín hiệu này chơi vai trò quan trọng trong việc đồng bộ hóa hoạt động của các bus tớ và đảm bảo rằng chỉ có các bus tớ được lựa chọn mới tham gia vào quá trình truyền dữ liệu.



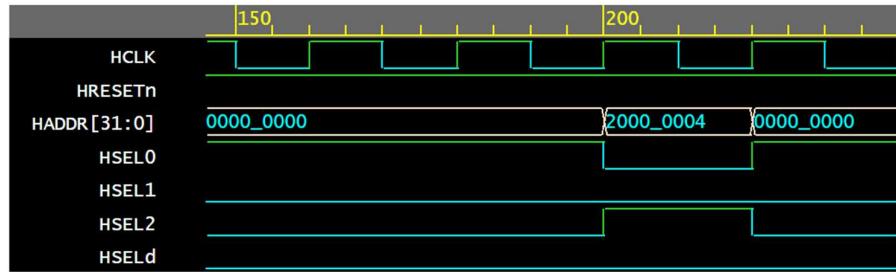
**Hình 4.12 Các tín hiệu trên bộ giải mã địa chỉ**

Trong hình 4.12, bộ giải mã địa chỉ có nhiệm vụ xử lý địa chỉ (HADDR) được gửi từ bus chủ tới các bus tớ, trong quá trình thực hiện các hoạt động trao đổi dữ liệu. Với việc đã trước phân chia các vùng địa chỉ riêng biệt cho từng bus tớ, bộ giải mã sẽ thực hiện quá trình quyết định xem bus tớ nào sẽ tham gia vào hoạt động truyền tải dữ liệu, thông qua việc đặt tín hiệu HSEL tương ứng lên mức cao, và đồng thời duy trì các tín hiệu HSEL của các bus tớ khác ở mức thấp. Những tín hiệu HSEL này đóng vai trò quan trọng trong việc điều phối quá trình truyền dữ liệu giữa các bus tớ và bus chủ.

Khi bộ giải mã địa chỉ quyết định chọn một bus tớ cụ thể, tín hiệu HSEL tương ứng sẽ được đưa lên mức cao, và thông qua cơ chế truyền tín hiệu, nó sẽ được gửi đến bus tớ tương ứng. Điều này làm cho bus tớ biết rằng nó đã được chọn để tham gia vào quá trình truyền dữ liệu và tương tác với bus chủ. Đồng thời, các tín hiệu HSEL của các bus tớ khác sẽ được giữ ở mức thấp, chỉ định rằng chúng không được chọn trong thời điểm hiện tại.

Tín hiệu HSEL không chỉ thông báo việc chọn bus tớ mà còn góp phần quản lý và điều phối các hoạt động truyền tải dữ liệu, đảm bảo tính toàn vẹn và đồng bộ của thông

tín giữa các thành phần trong hệ thống. Quá trình này giúp đảm bảo rằng các bus tớ tham gia vào hoạt động một cách hiệu quả và đồng bộ, từ đó đảm bảo sự chính xác và tin cậy của truyền thông dữ liệu trong hệ thống. Các tín hiệu HSEL này sẽ được gửi tới bus tớ để thông báo đã được chọn và được gửi đến bộ phân kênh tín hiệu làm tín hiệu điều khiển như ở dưới đây.



**Hình 4.13 Kết quả tín hiệu lựa chọn bus tớ trả về từ bộ giải mã địa chỉ**

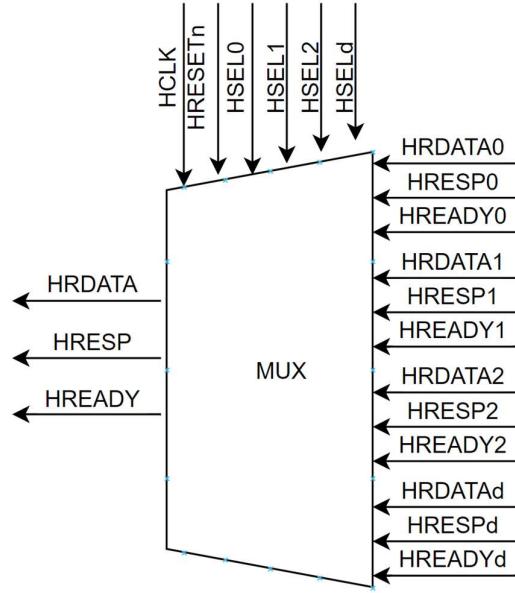
Trên hình 4.13 thể hiện giá trị HSEL thông báo cho bus tớ biết được chọn từ bộ giải mã địa chỉ gửi về. Tại thời điểm 150ns trước đó, tín hiệu HSEL 0 của bus tớ 0 có giá trị ở mức cao, còn các tín hiệu HSEL của các bus tớ khác có giá trị ở mức thấp. Tại thời điểm 200ns, có giá trị địa chỉ được truyền vào, bộ giải mã địa chỉ sẽ quyết định xem bus tớ nào sẽ được chọn tương ứng với vùng địa chỉ đó. Trong chu kỳ xung tín hiệu đồng hồ của địa chỉ 32'h20000004, tín hiệu HSEL2 ở mức cao còn các tín hiệu HSEL khác ở mức thấp, thể hiện rằng bus tớ 2 đang được chọn cho quá trình truyền dữ liệu.

#### 4.5.2 Bộ phân kênh tín hiệu từ bus tớ đến bus chủ

Với các tín hiệu truyền từ bus tớ đến bus chủ, chỉ có bus tớ đang hoạt động mới được phân kênh dữ liệu. Do đó, không cần phải chia ra hai loại tín hiệu điều khiển như bộ phân kênh MUX. Thay vào đó, việc điều khiển phân kênh sẽ được thực hiện bởi hai tín hiệu: tín hiệu HSELx (để xác định bus tớ nào đang hoạt động) và tín hiệu MUX (để biết khi nào pha dữ liệu bắt đầu).

Khi một bus tớ đang hoạt động (được xác định bởi tín hiệu HSELx), dữ liệu từ bus tớ này sẽ được phân kênh tới bus chủ dựa trên pha dữ liệu tương ứng. Việc này đảm bảo rằng chỉ có các bus tớ đang hoạt động mới tham gia vào quá trình truyền dữ liệu. Tín hiệu MUX sẽ giúp xác định thời điểm pha dữ liệu bắt đầu, từ đó điều khiển quá trình phân kênh dữ liệu.

Tóm lại, việc điều khiển phân kênh dữ liệu sẽ dựa vào tín hiệu HSELx để xác định bus tớ nào đang hoạt động và tín hiệu MUX để biết khi nào pha dữ liệu bắt đầu. Nhờ vào hai tín hiệu này, quá trình truyền dữ liệu sẽ được tổ chức một cách hiệu quả và đáng tin cậy giữa bus chủ và các bus tớ.



**Hình 4.14 Các chân tín hiệu trên bộ phân kênh**

Trong hình 4.14, đầu vào của bộ phân kênh tín hiệu từ bus tớ gửi sang bus chủ được tổ chức thành 4 nhóm tín hiệu, mỗi nhóm tương ứng với tín hiệu phản hồi từ 4 bus con khác nhau. Mỗi nhóm tín hiệu này bao gồm các tín hiệu quan trọng để thực hiện quá trình truyền tải dữ liệu từ các bus tớ về bus chủ. Cụ thể, mỗi nhóm tín hiệu bao gồm:

- Dữ liệu đọc (HRDATA): Đây là tín hiệu mang thông tin dữ liệu đã được đọc từ bus tớ và sẽ được truyền về bus chủ. Tín hiệu này chứa dữ liệu cụ thể mà bus chủ đã yêu cầu từ bus tớ. Quá trình truyền dữ liệu đọc này giúp bus chủ nhận được dữ liệu cần thiết từ các thiết bị bus tớ.
- Tín hiệu phản hồi (HRESP): Tín hiệu này chứa thông tin về phản hồi của giao dịch từ bus tớ. Nó định rõ liệu giao dịch đã thành công, bị lỗi hay cần thử lại. Tín hiệu phản hồi đóng vai trò quan trọng trong việc xác định trạng thái của giao dịch và đảm bảo tính toàn vẹn của dữ liệu truyền tải.
- Tín hiệu sẵn sàng (HREADY): Tín hiệu này được tạo ra từ bus tớ để thông báo rằng nó đã sẵn sàng thực hiện giao dịch. Khi tín hiệu sẵn sàng được đặt lên cao, nghĩa là bus tớ đã hoàn tất việc chuẩn bị và dữ liệu có thể được truyền tải.

Sự lựa chọn của nhóm tín hiệu nào sẽ được gửi về bus chủ được quyết định bởi các tín hiệu điều khiển HSEL0, HSEL1, HSEL2, HSELd từ bộ giải mã địa chỉ. Các tín hiệu này có nhiệm vụ quản lý việc chọn và điều phối dữ liệu từ các bus con về bus chủ. Bằng cách này, bộ phân kênh tín hiệu từ bus tớ gửi sang bus chủ đảm bảo rằng quá trình truyền tải dữ liệu diễn ra một cách hiệu quả và đồng bộ, đảm bảo tính toàn vẹn và chính xác của thông tin giữa các thành phần khác nhau trong hệ thống.

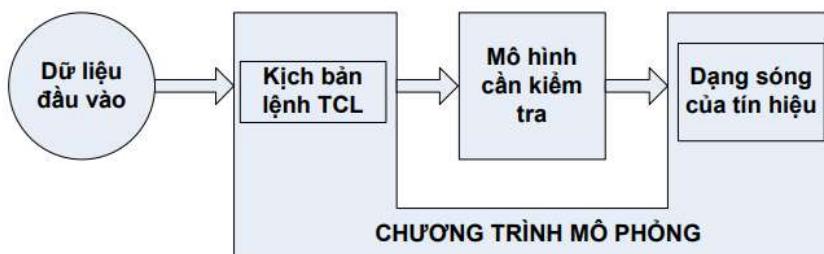
#### **4.6 Kết luận chương**

Chương 4 giải quyết vấn đề bài toán thiết kế hệ thống. Trong chương này đã đưa ra mô hình đề xuất của hệ thống giao tiếp giữa 1 bus chủ và 3 bus tớ. Qua đó chuẩn kết nối của từng module được giải thích và nhiệm vụ hoạt động của từng module đã được trình bày. Chương 5 tiếp theo sẽ thực hiện triển khai mô phỏng và kiểm thử mô hình đề xuất đã được thiết kế.

# CHƯƠNG 5. MÔ PHỎNG VÀ KIỂM THỬ MÔ HÌNH

## 5.1 Phương pháp kiểm tra đánh giá mô hình

Giai đoạn xác minh tính chính xác của mô hình hệ thống bus và đánh giá hiệu suất phần cứng được thực hiện thông qua sử dụng một nền tảng EDA Playground trực tuyến miễn phí. Được dựa trên các thông số kỹ thuật đã được nêu ở trước, ta tạo ra một loạt dữ liệu điều kiện ban đầu để tạo nên một môi trường kiểm tra cho hệ thống bus. Những dữ liệu này, thường gọi là dữ liệu kiểm tra hoặc tín hiệu kiểm tra, được đưa vào các chân lối vào theo các kịch bản đã định sẵn. Tại các chân lối ra, chúng ta thu thập dữ liệu đầu ra dưới dạng biểu đồ thời gian. Bằng cách so sánh những biểu đồ thời gian này với các biểu đồ từ các thông số kỹ thuật đã xác định trước đó, ta có thể đánh giá tính chính xác của quá trình hoạt động của hệ thống. Tất cả các bước mô phỏng và kiểm tra này được trình bày dưới dạng sơ đồ khái trong Hình 5.1.

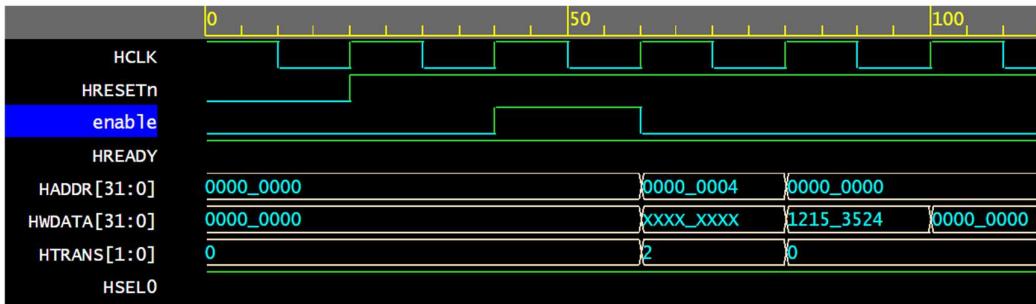


Hình 5.1 Sơ đồ khái quát quá trình mô phỏng kiểm tra hệ thống

## 5.2 Mô phỏng mô hình

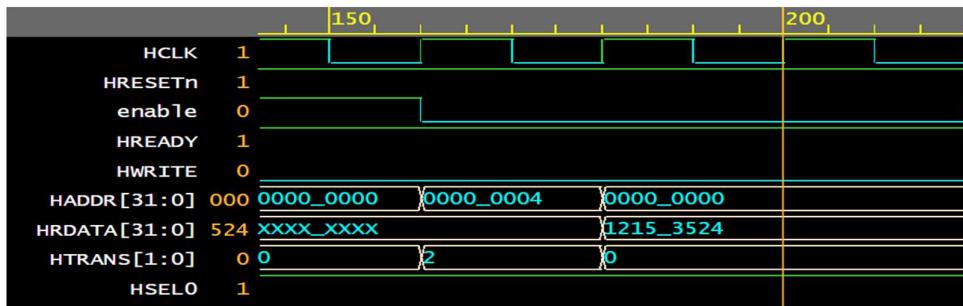
### 5.2.1 Mô phỏng hoạt động truyền đơn không có trạng thái đợi

Để đánh giá hiệu suất truyền thông của mô hình, chúng ta sẽ tạo ra các dữ liệu đầu vào ngẫu nhiên để yêu cầu hệ thống thực hiện các hoạt động đọc và ghi mà không có trạng thái chờ đợi. Các tín hiệu đầu vào như tín hiệu địa chỉ và tín hiệu dữ liệu sẽ được lựa chọn ngẫu nhiên và đưa vào bên trong thiết kế DUT. Qua việc này, chúng ta sẽ có cơ hội đánh giá khả năng hoạt động của hệ thống trong các tình huống thực tế và đo lường tốc độ và độ tin cậy của quá trình truyền thông.



**Hình 5.2 Quá trình ghi đơn không có trạng thái đợi ở bus tó 0**

Từ thời điểm 0ns đến 20ns, tín hiệu RESETn duy trì ở mức thấp, làm cho hệ thống được thiết lập lại vào trạng thái ban đầu. Khi đến thời điểm 40ns, tín hiệu enable được đưa lên mức cao để báo hiệu rằng một quá trình truyền thông sẽ được khởi tạo. Tại thời điểm 60ns, tín hiệu HADDR được ghi nhận, và tại thời điểm này, trạng thái của quá trình truyền thông được xác định là NONSEQ thông qua tín hiệu HTRANS với giá trị là 2. Tín hiệu HADDR cũng được lấy mẫu với giá trị là 32'h4; đây là vùng địa chỉ của bus tó số 0, do đó tín hiệu HSEL0 được đưa lên mức cao để báo hiệu rằng bus tó số 0 đang được chọn. Tiếp tục đến thời điểm 80ns, dữ liệu ghi được lấy mẫu cho quá trình ghi tương ứng với địa chỉ 32'h4 đã được xác định trước đó.

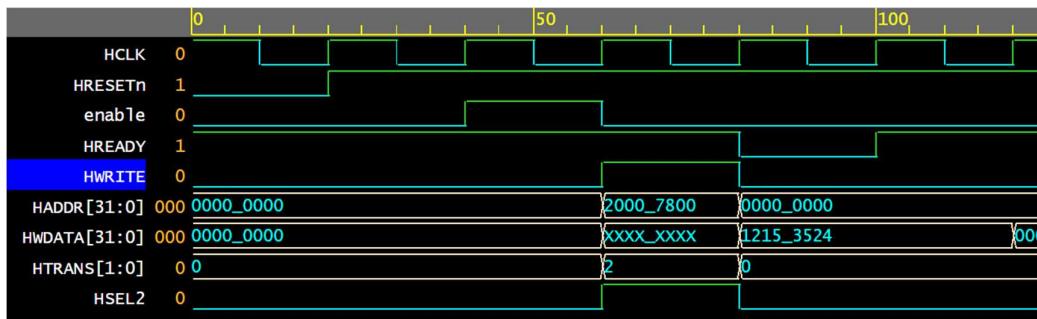


**Hình 5.3 Quá trình đọc đơn không có trạng thái đợi**

Tương tự với quá trình ghi đơn, tại thời điểm 160ns tín hiệu HADDR được lấy mẫu giá trị 32'h4. Chỉ khác quá trình ghi là tại thời điểm này tín hiệu HWRITE ở mức thấp để biểu thị đây là quá trình đọc. Tại thời điểm 180ns, dữ liệu của quá trình đọc sẽ được bus tó 0 gửi về tại tín hiệu HRDATA.

Kiểm

### 5.2.2 Mô phỏng hoạt động truyền đơn có trạng thái đợi



Hình 5.4 Quá trình ghi đơn có trạng thái đợi vào bus tớ 2

Trong khoảng thời gian từ 0ns đến 20ns, tín hiệu RESETn duy trì ở mức thấp, đưa hệ thống về trạng thái khởi đầu. Khi tiến đến 40ns, tín hiệu enable được đưa lên mức cao để thông báo rằng một quá trình truyền thông sắp diễn ra. Đến thời điểm 60ns, tín hiệu địa chỉ HADDR được lấy mẫu với giá trị là 32'h20007800, đại diện cho vùng địa chỉ của bus tớ 2 mà đã được cấu hình từ trước. Như kết quả, tín hiệu HSEL2 được nâng từ mức thấp lên mức cao, và tín hiệu HWRITE cũng ở mức cao, tạo điều kiện cho quá trình ghi vào bus tớ 2. Bus tớ 2 đã được thiết lập sẵn với trạng thái chờ sẵn sàng. Đến thời điểm 80ns, địa chỉ của quá trình ghi được lấy mẫu. Ở thời điểm này, tín hiệu HREADY ở mức thấp, báo hiệu rằng bus tớ 2 cần thêm chu kỳ để hoàn tất quá trình ghi. Tín hiệu ghi tiếp tục kéo dài cho đến khi tín hiệu HREADY quay trở lại mức cao, được kết thúc và được ghi nhận tại thời điểm 120ns.

### 5.2.3 Mô phỏng hoạt động truyền với vùng địa chỉ bus tớ mặc định

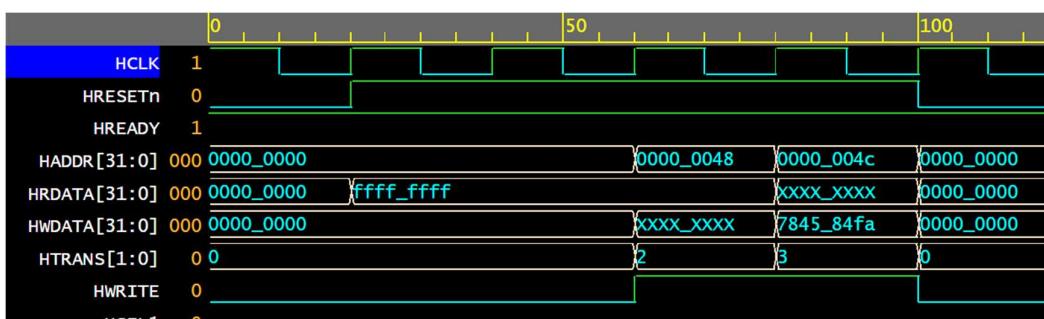


Hình 5.5 Quá trình ghi vào vùng địa bus tớ mặc định

Tương tự như quá trình ghi đơn khác, tại thời điểm 60ns, tín hiệu địa chỉ HADDR được lấy mẫu với giá trị 32'h30000650. Vùng địa chỉ này nằm ngoài phạm vi của các bus từ 0, 1, và 2, do đó nó sẽ là vùng địa chỉ của bus từ mặc định. Tại thời điểm này, tín hiệu HSELd cũng chuyển từ mức thấp lên mức cao, thể hiện việc bus từ mặc định đang được chọn. Đồng thời, tín hiệu HTRANS có giá trị 2, báo hiệu rằng quá trình truyền thông đang diễn ra ở trạng thái NONSEQ (không tuần tự). Sau đó, đến thời điểm 80ns, tín hiệu HRESP phản hồi từ bus từ về bus chủ để thông báo rằng quá trình truyền thông gặp lỗi. Tín hiệu lỗi này kéo dài trong 2 chu kỳ xung đồng hồ. Bus chủ nhận được tín hiệu báo lỗi và sau đó chuyển trạng thái của mình về IDLE (trạng thái không hoạt động) và chờ cho quá trình tiếp theo để thực hiện.

#### **5.2.4 Mô phỏng hoạt động của mô hình khi có tín hiệu reset**

Trong quá trình hoạt động của bất kỳ hệ thống nào, một tình huống phổ biến là hệ thống nhận được tín hiệu reset để được đưa về trạng thái khởi đầu. Trong hệ thống bus AHB, khi tín hiệu HRESETn được kích hoạt xuống mức thấp, ngay lập tức các tín hiệu đầu ra sẽ được đưa về trạng thái ban đầu, được thiết lập là giá trị 0.



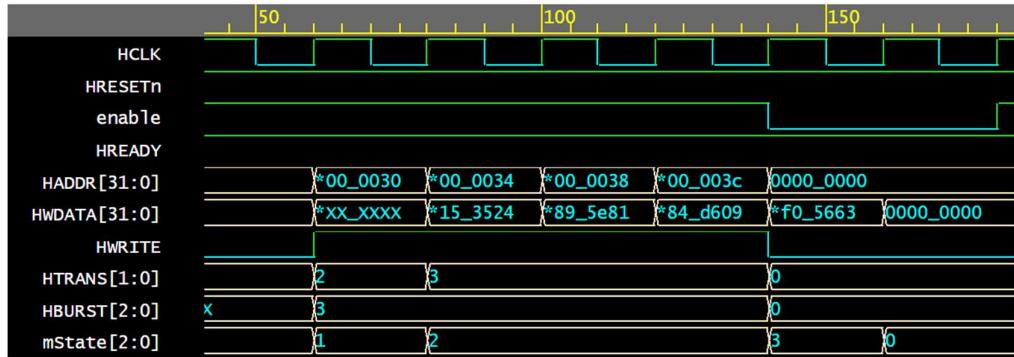
**Hình 5.6 Quá trình truyền khi có tín hiệu reset**

Trên hình 5.6, tại thời điểm 100ns tín hiệu HRESETn từ mức cao trở về mức thấp, báo hiệu mô hình đặt lại trạng thái ban đầu. Ngay lập tức các tín hiệu như HADDR, HRDATA, HWDATA, HTRANS, HWRITE, ... được đặt về trạng thái ban đầu.

#### **5.2.5 Mô phỏng hoạt động của quá trình truyền khôi**

Quá trình truyền khôi trong hệ thống truyền thông đem lại một số ưu điểm quan trọng. Thay vì truyền từng phần tử dữ liệu một, quá trình truyền khôi cho phép truyền một lượng lớn dữ liệu cùng một lúc, giúp tối ưu hóa băng thông truyền và giảm thiểu

thời gian truyền thông. Điều này đặc biệt hữu ích trong các ứng dụng yêu cầu xử lý dữ liệu nhanh chóng và hiệu quả. Quá trình truyền khói cũng giảm thiểu số lần truyền thông, giúp giảm độ trễ và tăng hiệu suất chung của hệ thống.

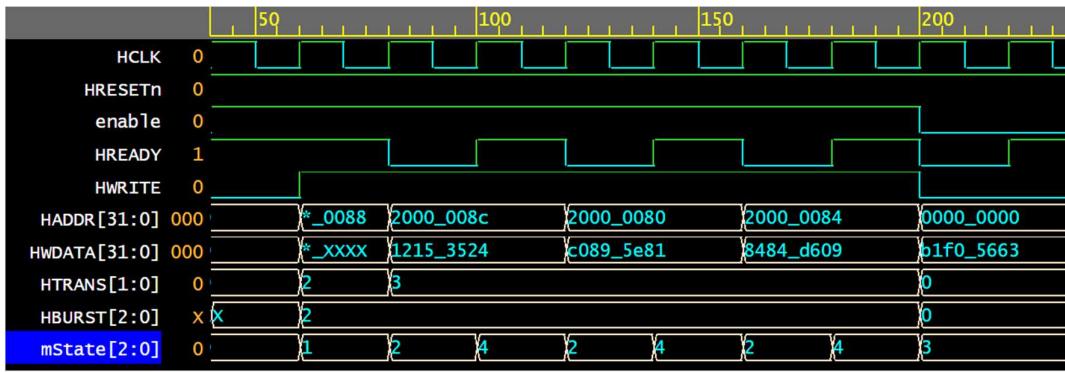


**Hình 5.7 Quá trình ghi khói tăng 4 nhịp không chờ**

Hình 5.7 minh họa quá trình truyền khói tăng địa chỉ với bốn nhịp không có trạng thái chờ. Chúng ta sẽ mô tả sự diễn ra của các tín hiệu tại các thời điểm cụ thể:

- Tại thời điểm 60ns, tín hiệu địa chỉ HADDR được lấy mẫu. Đồng thời, tín hiệu HBURST = 3 để thông báo rằng quá trình truyền khói tăng địa chỉ bốn nhịp đang bắt đầu và sẽ duy trì ổn định đến khi lần truyền cuối cùng kết thúc. Tín hiệu HTRANS = 2 cho biết trạng thái của quá trình là NONSEQ.
- Tại thời điểm 80ns, tín hiệu địa chỉ HADDR không được lấy mẫu, mà thay vào đó, nó tăng thêm 4 đơn vị so với địa chỉ trước đó. Do dữ liệu truyền có độ rộng 32 bit tương đương 4 bytes, vì vậy địa chỉ sẽ tăng 4. Đồng thời, tín hiệu HTRANS = 3 để thông báo rằng quá trình truyền đang trong trạng thái SEQ, và tín hiệu ghi cũng sẽ được truyền trong thời gian này.
- Đến thời điểm 140ns, sau khi tín hiệu địa chỉ đã được tạo ra lần thứ 4, dữ liệu ghi cuối cùng cũng sẽ được truyền. Tại thời điểm này, tín hiệu HTRANS = 0 để báo hiệu rằng quá trình truyền đã kết thúc và trở về trạng thái IDLE.

Quá trình truyền khói tăng địa chỉ này có thể được tóm tắt như sau: Đầu tiên, tín hiệu địa chỉ được lấy mẫu và tín hiệu HBURST được cài đặt để xác định số lượng nhịp truyền. Sau đó, trong mỗi nhịp truyền, địa chỉ được cập nhật theo số lượng dữ liệu truyền, và tín hiệu HTRANS xác định trạng thái của quá trình truyền. Cuối cùng, khi tất cả các dữ liệu đã được truyền, tín hiệu HTRANS được đặt lại về IDLE để kết thúc quá trình truyền khói tăng địa chỉ.



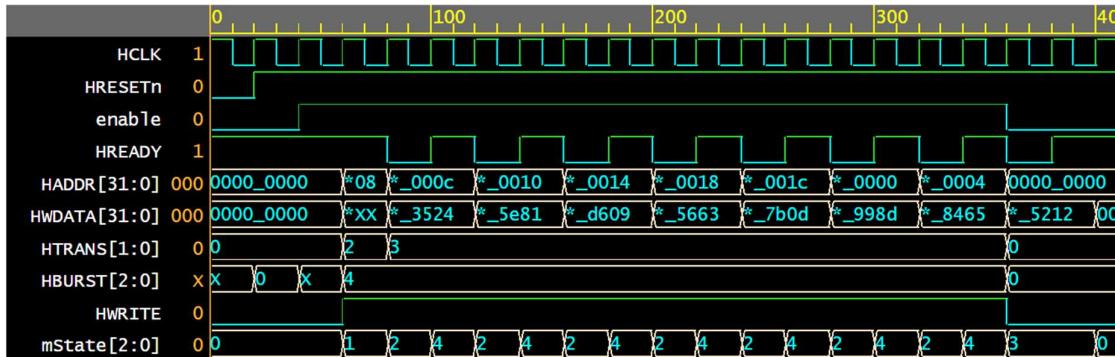
**Hình 5.8 Quá trình ghi khối cuộn 4 nhịp có trạng thái chờ**

Hình 5.8 thể hiện quá trình truyền khối cuộn 4 nhịp có trạng thái chờ. Dưới đây là mô tả chi tiết về các tín hiệu và sự kiện tại các thời điểm quan trọng:

- Tại thời điểm 60ns, tín hiệu địa chỉ HADDR được lấy mẫu với giá trị 32'b2000\_0088. Đồng thời, tín hiệu HBURST = 2 để báo hiệu một quá trình truyền khối cuộn 4 nhịp đang bắt đầu. Tín hiệu HTRANS = 2 cho biết trạng thái truyền là NONSEQ và trạng thái của bus chủ là SINGLE.
- Đến thời điểm 80ns, địa chỉ không được lấy mẫu mà được tăng thêm 4 so với địa chỉ trước đó. Tín hiệu HTRANS lúc này có giá trị là 3 để báo hiệu trạng thái truyền là SEQ. Dữ liệu ghi cũng được truyền tại thời điểm này và trạng thái của bus chủ là BURST.
- Tại thời điểm 100ns, do tín hiệu HREADY trước đó ở mức thấp, quá trình truyền dữ liệu ghi được mở rộng thêm 1 chu kỳ. Trạng thái của bus chủ lúc này là WAIT và chờ tín hiệu HREADY lên mức cao để kết thúc giai đoạn truyền dữ liệu này.
- Đến thời điểm 120ns, tín hiệu địa chỉ là 32'h2000\_0090, tuy nhiên nó trùng với địa chỉ biên của quá trình cuộn (theo công thức ở Chương 2). Do đó, địa chỉ tiếp theo sẽ quay trở lại 32'h2000\_0080.
- Cuối cùng, đến thời điểm 200ns, quá trình ghi dữ liệu sẽ được thực hiện lần cuối. Trạng thái của bus chủ tại thời điểm này là LAST, báo hiệu cho kết thúc quá trình truyền khối cuộn.

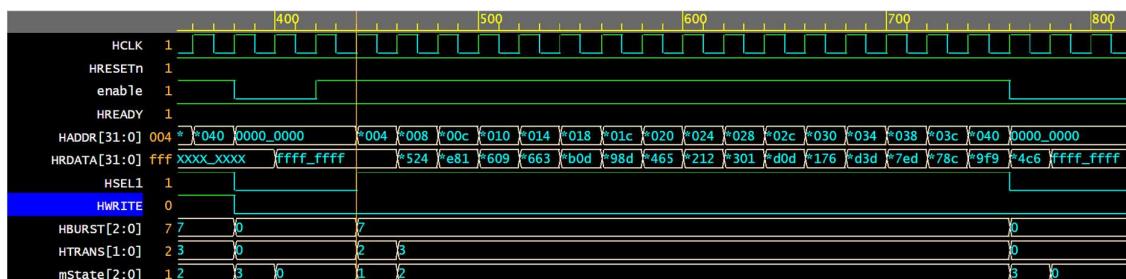
Toàn bộ quá trình truyền khối cuộn này có thể tóm tắt như sau: Đầu tiên, tín hiệu địa chỉ được lấy mẫu và tín hiệu HBURST xác định số lượng nhịp truyền. Trong mỗi nhịp truyền, địa chỉ được cập nhật và tín hiệu HTRANS xác định trạng thái của quá trình

truyền. Khi gắp tín hiệu HREADY ở mức thấp, quá trình truyền dữ liệu được mở rộng. Nếu địa chỉ cuối cùng gắp biên, nó sẽ quay trở lại địa chỉ đầu tiên của cuộn. Cuối cùng, khi tất cả dữ liệu đã được truyền, trạng thái của bus chủ trở về LAST để kết thúc quá trình truyền khối cuộn.



Hình 5.9 Quá trình truyền khối cuộn 8 nhịp

Tương tự như các quá trình truyền khối trên, hình 5.9 thể hiện quá trình ghi khối với tín hiệu HWRITE báo hiệu quá trình ghi, tín hiệu HBURST báo hiệu quá trình khối cuộn 8 nhịp. Tại thời điểm 60ns, tín hiệu địa chỉ HADDR được lấy mẫu, sau đó các tín hiệu khối khác sẽ tăng thêm 4 đơn vị. Tại thời điểm 280ns, giá trị địa chỉ được thể hiện quay về lại giá trị địa chỉ cuộn.



Hình 5.10 Quá trình truyền khối cuộn 16 nhịp

Hình 5.10 thể hiện quá trình đọc khối tăng 16 nhịp. Tại thời điểm 440ns, quá trình đọc được bắt đầu với địa chỉ khối đầu tiên. Tín hiệu HWRITE ở mức thấp thể hiện một quá trình đọc, tín hiệu HSEL1 ở mức cao báo hiệu bus tớ 1 đang thực hiện quá trình đọc. Bus tớ 1 được thiếp lập không có trạng thái đợi nên mỗi giai đoạn dữ liệu chỉ thực hiện trong một chu kỳ.

### **5.3 Kế hoạch kiểm thử**

Kiểm thử là quá trình xác định xem các chức năng của của hệ thống được đề xuất đúng như tài liệu đặc tả kỹ thuật. Kế hoạch kiểm thử sẽ liệt kê ra các tính năng của thiết kế và các kịch bản để có thể kiểm tra các chức năng đó. Trong đó các tính năng của hệ thống cần kiểm thử bao gồm:

- Tính năng reset của hệ thống
- Đọc ghi đơn
- Đọc ghi khối tăng nhịp 4
- Đọc ghi khối cuộn nhịp 4
- Đọc ghi khối tăng nhịp 8
- Đọc ghi khối cuộn nhịp 8
- Đọc ghi khối tăng nhịp 16
- Đọc ghi khối cuộn nhịp 16
- Đọc ghi vào bus tóm tắt định

Các chức năng của hệ thống cần kiểm thử được mô tả như bảng 5.1 dưới đây.

**Bảng 5.1 Mô tả các chức năng cần kiểm thử của hệ thống**

STT	Chức năng	Mô tả testcase	Mục tiêu
1	Tính năng reset của hệ thống	Cấu hình quá trình đọc hoặc ghi hệ thống với các tín hiệu điều khiển của quá trình ghi đơn. Sau đó thực hiện đưa giá trị HRESETn xuống mức thấp.	Xác nhận hoạt động của waveform đúng như mô tả của timing diagram trong tài liệu đặc tả.
2	Đọc ghi đơn	Thực hiện quá trình ghi đơn của hệ thống với tín hiệu địa chỉ HADDR được đưa vào và tín hiệu dữ liệu HWDATA được sinh ra ngẫu nhiên.  Sau đó thực hiện quá trình ghi đơn với tín hiệu HADDR như trên.	Xác nhận hoạt động của waveform đúng như mô tả của timing diagram trong tài liệu đặc tả.  So sánh dữ liệu đưa vào quá trình ghi và dữ liệu được đưa ra và thông báo ra màn hình kết quả khớp hay không khớp.
3	Đọc ghi khối	Thực hiện quá trình ghi khối với tín hiệu điều khiển HBURST tùy chọn các kiểu ghi khối khác nhau cho từng testcase. Đưa tín hiệu địa chỉ HADDR đầu tiên của khối vào quá trình và đưa ngẫu nhiên các tín hiệu dữ liệu ghi HWDATA vào quá trình.  Sau đó thực hiện quá trình đọc khối với tín hiệu điều khiển như quá trình ghi. Tín hiệu địa chỉ đầu tiên của khối cũng là tín hiệu địa chỉ HADDR của quá trình ghi.	Xác nhận hoạt động của waveform đúng như mô tả của timing diagram trong tài liệu đặc tả.  So sánh dữ liệu được đọc ra với từng dữ liệu được ghi vào theo từng địa chỉ khác nhau trong khối và thông báo ra màn hình kết quả khớp hay không khớp.
4	Đọc ghi vào bus từ mặc định	Thực hiện quá trình đọc hoặc ghi với tín hiệu địa chỉ HADDR là địa chỉ thuộc vùng bus từ mặc định.	Xác nhận hoạt động của waveform đúng như mô tả của timing diagram trong tài liệu đặc tả.

## 5.4 Kết quả thu được và đánh giá hệ thống

### 5.4.1 Kết quả thực hiện testcase

Với việc triển khai các testcase cần thực hiện để đánh giá sự đúng đắn của chức năng của hệ thống. Hình 5.11 mô tả kết quả so sánh dữ liệu được ghi vào và dữ liệu được đọc ra của khối truyền cuộn 4 nhịp.

```
----->PASS!
time = 920, -----BURST WRITE WRAP4-----
Addr write = 00000084 Data write = 114806029
Addr write = 00000088 Data write = 992211318
Addr write = 0000008c Data write = 512609597
Addr write = 00000080 Data write = 1993627629
time = 1080, -----BURST READ WRAP4-----
>>>>>Addr read= 00000084
>>>>>Expected data = 06d7cd0d --- Actual data = 06d7cd0d
----->PASS!
>>>>>Addr read= 00000088
>>>>>Expected data = 3b23f176 --- Actual data = 3b23f176
----->PASS!
>>>>>Addr read= 0000008c
>>>>>Expected data = 1e8dc3d3d --- Actual data = 1e8dc3d3d
----->PASS!
>>>>>Addr read= 00000080
>>>>>Expected data = 76d457ed --- Actual data = 76d457ed
----->PASS!
time = 1200 -----RIIDCT WDTTC WDAD4-----
```

Hình 5.11 Kết quả thu được trên mô phỏng

Theo số liệu trên hình 5.9, tại thời điểm 920ns quá trình ghi cuộn 4 nhịp được thực hiện. Với địa chỉ của từng khối và dữ liệu được thông báo ra màn hình. Sau đó, tại thời điểm 1080ns có quá trình đọc khối cuộn 4 nhịp được thực thi. Địa chỉ đầu tiên của khối là 32'h84, đi kèm theo đó có dữ liệu mong muốn Expected data là 32'h06d7cd0d và dữ liệu được đọc ra Actual data là 32'h06d7cd0d. Dữ liệu mong muốn và dữ liệu được đọc ra khớp nhau nên thông báo được in ra là PASS thể hiện khối đọc đúng dữ liệu mong muốn. Tương tự thế với các địa chỉ 32'h88, 32'h8c, 32'h80 đều có các dữ liệu đọc ra khớp với dữ liệu được ghi vào bên trên.

```

PASS =      1188 times
FAIL =        0 times
          v c s   s i m u l a t i o n   R e p o r t
Time: 90000 ns

```

### Hình 5.12 Kết quả của mô phỏng khi được thực thi

Trên hình 5.12, số lần so sánh khớp và không khớp dữ liệu được thống kê lại. Với số lần so sánh khớp dữ liệu là 1188 lần và không có lần nào không khớp dữ liệu mong muốn.

#### 5.4.2 Kết quả coverage code

Coverage code đo lường mức độ bao phủ của mã nguồn (source code) bằng cách theo dõi và đếm các thành phần cụ thể trong mã nguồn đã được thực thi trong quá trình kiểm thử. Mục tiêu của việc đo code coverage là xác định phần trăm mã nguồn đã được thực thi bởi các ca kiểm thử, giúp đảm bảo rằng các đoạn mã không được thực thi (uncovered code) không bị bỏ sót trong quá trình kiểm thử.

Các loại coverage code bao gồm:

- Branch Coverage: Đo lường tỷ lệ các nhánh hoặc điều kiện trong mã nguồn đã được thực thi. Một nhánh hoặc điều kiện được coi là covered khi nó đã được thực thi ít nhất một lần.
- Condition Coverage: Đo lường mức độ mà tất cả các điều kiện logic trong mã nguồn đã được kiểm tra. Một điều kiện logic thường là một biểu thức có chứa các phép so sánh như `==`, `!=`, `<`, `>`, `<=`, `>=`, hoặc các phép logic như `&&`, `||`. Coverage code sẽ kiểm tra xem tất cả các nhánh điều kiện trong mã nguồn đã được thử nghiệm hay chưa.
- Expression Coverage: Đo lường mức độ mà các biểu thức toán học hoặc logic đã được kiểm tra. Điều này đảm bảo rằng các tính toán và phép toán trong mã nguồn đã được thử nghiệm. Coverage code sẽ kiểm tra xem tất cả các biểu thức đã được sử dụng trong mã nguồn đã được kiểm tra hay chưa.
- FSM State: Đo lường số trạng thái của máy trạng thái đã được thực thi hay chưa. Đảm bảo rằng mỗi trạng thái sẽ được thực thi ít nhất một lần.
- FSM Transitions: Đo lường các đường chuyển trạng thái trong máy trạng thái. Một đường chuyển trạng thái được gọi là covered khi được thực hiện ít nhất một lần.

- Statement Coverage: Đo lường tỷ lệ các câu lệnh trong mã nguồn đã được thực thi. Một câu lệnh được coi là covered khi nó đã được thực thi ít nhất một lần.



**Hình 5.13 Kết quả coverage code tổng quan hệ thống**

Trên hình 5.13 thể hiện kết quả coverage code của hệ thống. Tỉ lệ bao phủ của hệ thống là 99.67% với các nội dung chi tiết được thể hiện trên hình. Trong đó có từng mục về độ bao phủ về bus chủ, 3 bus tớ và ahb\_bus bao gồm bộ giải mã, bộ phân kênh, bus tớ mặc định được thể hiện.

Tỉ lệ 99.67% thể hiện mức độ bao phủ của code gần như hoàn toàn nhưng còn một số trường hợp điều kiện case nhỏ chưa thể thực hiện. Điều đó khiến tỉ lệ này không được hoàn chỉnh 100%.

## 5.5 Đánh giá chung về kết quả thu được

Dựa vào những kết quả mô phỏng chi tiết đã được trình bày và phân tích ở phần trước, cùng với việc so sánh với các đặc tả kỹ thuật và yêu cầu đã được đặt ra trong phần mô hình hoá hệ thống ở Chương 2. Ta có thể kết luận sơ lược như sau:

- Với từng testcase đọc ghi của các chức năng truyền đã được thực hiện mô phỏng so sánh đúng dữ liệu truyền đi và đọc về.
- Wave form thực hiện mô phỏng giống với tài liệu đặc tả kỹ thuật của AHB.
- Tỉ lệ coverage code 99.67% thể hiện code thiết kế đã được bao phủ hoàn toàn.

Các chức năng hỗ trợ của hệ thống đã được kiểm thử và cho thấy kết quả đáng kể. Điều này chứng tỏ hệ thống đã hoạt động ổn định, đảm bảo quá trình truyền dữ liệu không bị mất mát.

## KẾT LUẬN

### Kết luận chung

AMBA AHB (Advanced High-performance Bus) là một trong những chuẩn bus hệ thống tiên tiến của AMBA, được phát triển bởi ARM. Mục tiêu chính của AMBA AHB là cung cấp một cơ chế truyền thông hiệu suất cao và đa năng giữa các thành phần trong hệ thống chip (SoC). Mô hình hệ thống bus AMBA AHB đã được thiết kế và xây dựng trong đồ án này nhằm cung cấp sự linh hoạt và khả năng tương thích với các thiết bị phức tạp và yêu cầu truyền thông nhanh.

Mô hình hệ thống bus AMBA AHB được mô tả bằng ngôn ngữ phần cứng Verilog và đã trải qua quá trình kiểm thử kỹ thuật. Kết quả mô phỏng đã chứng minh rằng mô hình hệ thống bus AMBA AHB hoạt động chính xác và tuân theo các đặc tả kỹ thuật của chuẩn AMBA AHB.

Tuy nhiên, do hạn chế về thời gian và phạm vi của đồ án, một số tính năng có thể đã được đơn giản hóa trong quá trình thiết kế. Mặc dù vậy, qua quá trình nghiên cứu và xây dựng, đồ án này đã giúp em hiểu rõ hơn về cấu trúc và hoạt động của hệ thống bus AMBA AHB, cũng như học hỏi được nhiều kiến thức và kinh nghiệm thiết kế trong lĩnh vực thiết kế VLSI. Việc nắm vững quy trình thiết kế VLSI và làm chủ ngôn ngữ mô tả phần cứng Verilog là những kỹ năng quan trọng mà em đã đạt được qua quá trình thực hiện đồ án này.

### Hướng phát triển

Trong thời gian tới, mục tiêu phát triển của đề tài là tiến hành triển khai giao tiếp thiết kế nhiều bus chủ kết nối với nhiều bus tớ. Sau đó là tiến hành kiểm thử giao thức với những phương pháp tối ưu hơn như UVM.

## TÀI LIỆU THAM KHẢO

- [1] Integrated\_circuit, <http://en.wikipedia.org/>, truy nhập cuối cùng ngày 06/07/2023.
- [2] System-on-chip, <http://en.wikipedia.org/>, truy nhập cuối cùng ngày 08/07/2023.
- [3] 1974 digital watch is first system on chip integrated circuit, <http://www.computerhistory.org/semiconductor/timeline/1974-digitalwatch-is-first-system-on-chip-integrated-circuit-52.html>, truy cập cuối cùng ngày 15/07/2023.
- [4] Rishad A. Shafik and Paul Rosinger and Bashir M. Al-Hashimi, “MPEG-based Performance Comparison between Network-on-Chip and AMBA MPSoC”, *Design and Diagnostics of Electronic Circuits and Systems* , pp1-6, 2008.
- [5] Douglas L.Perry, *VHDL Programming by Example*, McGraw Hill, 2002.
- [6] Sudeep Pasricha and Nikil Dutt, *On-Chip Communication Architectures-System on Chip Interconnect*, Elsevier Inc, 2008.
- [7] ARM Limited (1999), AMBA™ Specification (Rev 2.0), <https://developer.arm.com/documentation/>, truy cập lần cuối 10/07/2023.
- [8] Mounir Maaref, *Architecting and Building High-Speed SoCs: Design, develop, and debug complex FPGA-based systems-on-chip*, [Online]. Available: <https://books.google.com.vn/>.
- [9] Kanishka Lahiri and Anand Raghunathan , “Power Analysis of System-Level On-Chip Communication Architectures, Hardware/software codesign and system synthesis”, *International conference on*, pp236-241, 2004.
- [10] AMBA® AHB Protocol Specification, <https://developer.arm.com/documentation/>, truy cập lần cuối 15/06/2023.
- [11] Samir Palnitkar, *Verilog HDL A guide to Digital Design and Synthesis*, SunSoft Press, 1996.
- [12] Chris Spear, *SystemVerilog for Verification: A Guide to Learning the Testbench Language Features 2nd*, Springer, 2008.