TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH

KHOA ĐIỆN- ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH- VIỄN THÔNG

HCMUTE

ĐỒ ÁN 2

THIẾT KẾ VÀ TRIỂN KHAI BUS AHB SỬ DỤNG NGÔN NGỮ VERILOG

Ngành Công Nghệ Kỹ Thuật Máy Tính

Sinh viên: **HOÀNG VÕ HOÀI NAM**

MSSV: 21119234

TP. HỒ CHÍ MINH – 05/2025

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

KHOA ĐIỆN- ĐIỆN TỬ

BỘ MÔN KỸ THUẬT MÁY TÍNH- VIỄN THÔNG

ĐỒ ÁN 2

THIẾT KẾ VÀ TRIỂN KHAI BUS AHB SỬ DỤNG NGÔN NGỮ VERILOG

Ngành Công Nghệ Kỹ Thuật Máy Tính

Sinh viên: **HOÀNG VÕ HOÀI NAM**

MSSV: 21119234

Hướng dẫn**: Ph.D TRƯƠNG QUANG PHÚC**

TP. HỒ CHÍ MINH – 05/2025

TÓM TẮT

Báo cáo này trình bày quá trình lên ý tưởng, nghiên cứu, thiết kế và triển khai một hệ thống bus AHB sử dụng ngôn ngữ Verilog. Dự án tập trung vào việc xây dựng và mô phỏng giao thức AHB với cấu hình gồm một Master và một Slave.

Hệ thống được thiết kế để thực hiện giao tiếp đọc và ghi dữ liệu giữa Master và Slave một cách hiệu quả, đảm bảo tính đồng bộ và độ chính xác cao. Các thành phần chính của giao thức, bao gồm kênh đọc, kênh ghi và các tín hiệu điều khiển, được triển khai chi tiết và kiểm tra thông qua mô phỏng trong môi trường Xilinx Vivado.

Báo cáo này có thể được sử dụng làm tài liệu tham khảo cho sinh viên các ngành Kỹ thuật máy tính, Kỹ thuật điện – điện tử, cũng như những người quan tâm đến lĩnh vực thiết kế vi mạch số và giao thức truyền thông trong hệ thống FGPA, SoC.

MỤC LỤC

Danh mục hình

Danh mục bảng

CHƯƠNG 1. TỔNG QUAN

1.1. ĐẶT VẤN ĐỀ

Tính tới hiện nay, lĩnh vực vi mạch được coi là một lĩnh vực vô cùng quan trong, có thể nói là lĩnh vực nòng cốt cho việc phát triển nhiều lĩnh vực khác. Các quốc gia lớn mạnh trên thế giới hiện tại đều có sự đóng góp quan trọng của ngành công nghiệp vi mạch. Bên cạnh đó, thách thức về nguồn nhân lực cũng rất lớn, các chuyên gia trình độ cao và tầm nhìn xa trong ngành.

Gần đây, công nghệ VLSI đã được cải thiện đáng kể và nhiều bóng bán dẫn hơn có thể được kết hợp trong một con chip. Cấu hình Hệ thống trên chip (SOC) có số khối được tích hợp trên một chip. Nhiều khối được tích hợp trong một vi mạch duy nhất, nhưng để truy cập chức năng của chúng, chúng yêu cầu một kiến trúc truyền thông mạnh mẽ. Điều này chỉ có thể đạt được bằng cách sử dụng kiến trúc bus trên chip để đáp ứng các yêu cầu của chúng. Các công ty khác nhau có các kiến trúc Bus on-Chip khác nhau nhưng một trong những kiến trúc phù hợp nhất là AMBA của ARM.

AMBA bao gồm ba bus chính, đó là Bus hệ thống nâng cao (ASB), Bus ngoại vi nâng cao (APB) và Bus hiệu suất cao nâng cao (AHB). Khi so sánh với hai bus khác AHB là hiệu suất cao, băng thông lớn và đối với các mô-đun hệ thống có tần số xung nhịp cao, các nhà thiết kế Hệ thống chọn AHB là sự lựa chọn chính của họ. AHB (Advanced High-performance Bus) là một bus cao cấp trong họ AMBA (Advanced Microcontroller Bus Architecture). Nó là một tiêu chuẩn cho việc giao tiếp giữa các mô đun trong một khuôn khổ. Các tiêu chuẩn bus AHB (Advance High performance) được đặc trưng bởi ARM hỗ trợ giao tiếp bộ nhớ trên chip, bộ xử lý và giao diện của bộ nhớ ngoài chip.

Với những lý do trên là một động lực giúp nhóm quyết định chọn đề tài Thiết kế và đánh giá giao thức Bus AMBA AHB.

1.2. MỤC TIÊU NGHIÊN CỨU

Mục tiêu nghiên cứu của đề tài "Thiết kế và đánh giá giao thức AMBA AHB":

- Xây dựng 1 hệ thống gồm 1 master và 1 slave được truyền nhận dữ liệu cho nhau qua chuẩn giao thức AMBA AHB.
- Dùng mã verilog để thiết kế hệ thống và kiểm tra mô phỏng các chế đồ truyền của hệ thống.

1.3. NỘI DUNG NGHIÊN CỨU

• NỘI DUNG 1: Tìm hiểu tổng quan về chuẩn Bus AMBA AHB.

• NỘI DUNG 2: Tổng quan về ngôn ngữ mô tả phần cứng Verilog và các môi trường làm việc, mô phỏng.

• NỘI DUNG 3: Nghiên cứu và xây dựng mô hình hệ thống tổng quát.

• NỘI DUNG 4: Thực hiện thiết kế RTL code và mô phỏng cho mô hình hệ thống.

• NỘI DUNG 5: Đánh giá kết quả thực hiện.

1.4. HẠN CHẾ ĐỀ TÀI

Vì lý do yếu tố về thời gian cũng như kiến thức, để tài này chỉ dừng lại ở việc thiết kế và kiểm chứng chuẩn bus AHB với một số chức năng cơ bản trong tài liệu kỹ thuật của AMBA AHB.

1.5. BỐ CỤC

Chương 1: Tổng quan

Trong chương này đồ án được trình bày tổng quan về lý do chọn đề tài, mục tiêu và nội dung nghiên cứu, giới hạn và bố cục đề tài.

Chương 2: Cơ sở lý thuyết

Trong chương này trình bày lý thuyết về chuẩn Bus AMBA AHB, ngôn ngữ mô tả phần cứng Verilog. Giới thiệu một số phần mềm được dùng trong quá trình thiết kế.

Chương 3: Thiết kế hệ thống

Chương này sẽ trình bày mô tả sơ đồ khối hệ thống một cách tổng quát sau đó đi sâu vào thiết kế chi tiết các khối.

Chương 4: Kết quả đánh giá

Chương này trình bày kết quả sau khi tổng hợp (synthesis), mô phỏng chức năng và đánh giá thiết kế của hệ thống.

Chương 5: Kết luận và hướng phát triển

Chương này trình bày ngắn gọn lại một lần nữa mục tiêu, hoạt động và đưa ra hướng phát triển của để tài.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. GIỚI THIỆU VỀ CHUẨN BUS AMBA AHB

2.1.1. Tổng quan về hệ thống AMBA AHB

Hệ thống bus là một thành phần không thể thiếu trong một hệ thống trên chip (SoC). Hệ thống bus đóng vai trò chính trong việc trung chuyển dữ liệu giữa các thành phần trong một hệ thống SoC. Đó cũng là một trong những thành phần chủ đạo ảnh hưởng đến hiệu năng của một chip SoC. Kiến trúc bus nâng cao dùng cho vi điều khiển (AMBA- Advanced Microcontroller Bus Architecture) là một kiến trúc bus dành cho các hệ thống SoC được công ty ARM đưa ra lần đầu tiên trên thị trường vào năm 1996. Trãi qua thời gian nghiên cứu và phát triển, đến hiện nay kiến trúc bus của AMBA đã có năm phiên bản phù hợp với nhiều mục đích truyền thông khác nhau trên một hệ thống SoC. Các phiên bản bus là sự nâng cấp tương ứng với sự phát triển của hệ thống trên chip, cũng như yêu cầu về mặt tốc độ và băng thông càng ngày càng tăng cao của các hệ thống này. Bốn phiên bản hệ thống bus của AMBA được ra mắt lần lượt là :

- Bus hệ thống nâng cao – ASB (Advanced System Bus) và bus ngoại vi nâng cao – APB (Advanced Peripheral Bus) được giới thiệu vào năm 1996.

- Bus hiệu suất cao – AHB (Advanced High-performance Bus) được ra mắt vào năm 1999 là một giao thức xung nhịp đơn.

- Giao diện mở rộng nâng cao – AXI (Advance eXtensible Interface) được giới thiệu vào năm 2003 và được nâng cấp lên AXI4 vào năm 2010.

- Giao diện trung tâm mạch lạc – CHI (Coherent Hub Interface) được ra mắt vào năm 2013 là giao thức để kết nối các bộ xử lý lại với nhau để giảm tắt nghẽn.

Bus hiệu suất cao được đánh giá qua tốc độ đường truyền (bao nhiêu bit trên một giây) và băng thông rộng (độ rộng của khung dữ liệu). Tốc độ truyền sẽ được tính dựa trên lượng dữ liệu truyền chia cho thời gian truyền xong dữ liệu đó: S = A/T. Từ công thức ta thấy khi băng thông dữ liệu càng lớn và thời gian truyền ít thì tốc độ truyền còn lớn, và đấy là mục đích bus AHB được ra mắt.

AHB bus là hệ thống bus thứ hai được ra mắt trong họ AMBA, được tạo ra với mục tiêu là nhằm đáp ứng yêu cầu của các thiết kế cần hiệu suất cao. Và đây là một hệ thống bus có hỗ trợ nhiều bus chủ với khả năng hỗ trợ băng thông rộng và tốc độ truyền dữ liệu cao. Bus AHB được tạo ra với mục đích là để đảm bảo quá trình truyền dữ liệu giữa các khối chức năng cần có độ rộng lớn và tốc độ truyền cao. Các tính năng của hệ thống bus AHB cần thiết cho các hệ thống tần số xung nhịp cao, hiệu suất cao bao gồm:

• Hoạt động tại sườn lên của xung đơn.

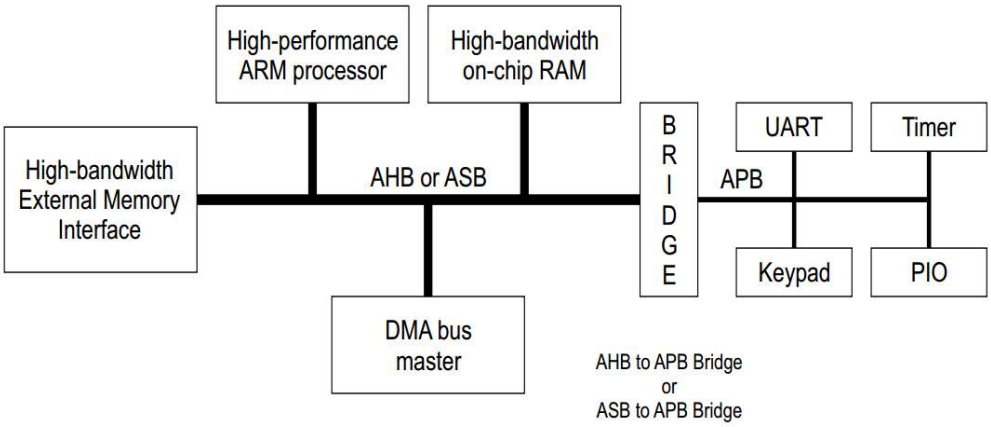
• Phân chia quá trình truyền.

• Bus chủ được chuyển giao trong một xung đơn.

• Truyền theo khối

• Không triển khai mức ba trạng thái.

• Có thể cấu hình độ rộng (8 bit => 1024 bit).



Hình 1. Hệ thống có AHB Bus

Hình 1 trình bày cấu trúc tiêu biểu của một vi điều khiển sử dụng hệ thống bus AHB theo chuẩn AMBA. Mục tiêu chính của tiêu chuẩn này là tăng tính linh hoạt và khả năng tái sử dụng của các khối chức năng phần cứng, từ đó giúp đơn giản hóa quy trình thiết kế hệ thống SoC (System-on-Chip). Kiến trúc AHB được

tối ưu hóa cho các ứng dụng yêu cầu tần số hoạt động cao và hiệu suất lớn, tiêu biểu bao gồm:

- Bộ xử lý ARM tốc độ cao.

- Bộ nhớ trong (on-chip RAM) với băng thông lớn.

- Bộ điều khiển DMA (Direct Memory Access).

- Cầu nối (bridge) giữa bus AHB/ASB và bus ngoại vi APB.

- Giao tiếp bộ nhớ ngoài có băng thông cao.

Trong khi đó, bus APB (Advanced Peripheral Bus) phù hợp hơn cho việc kết nối các thiết bị ngoại vi tốc độ thấp như: UART, bộ đếm thời gian (Timer), bàn phím (Keypad), và các khối I/O đơn giản.

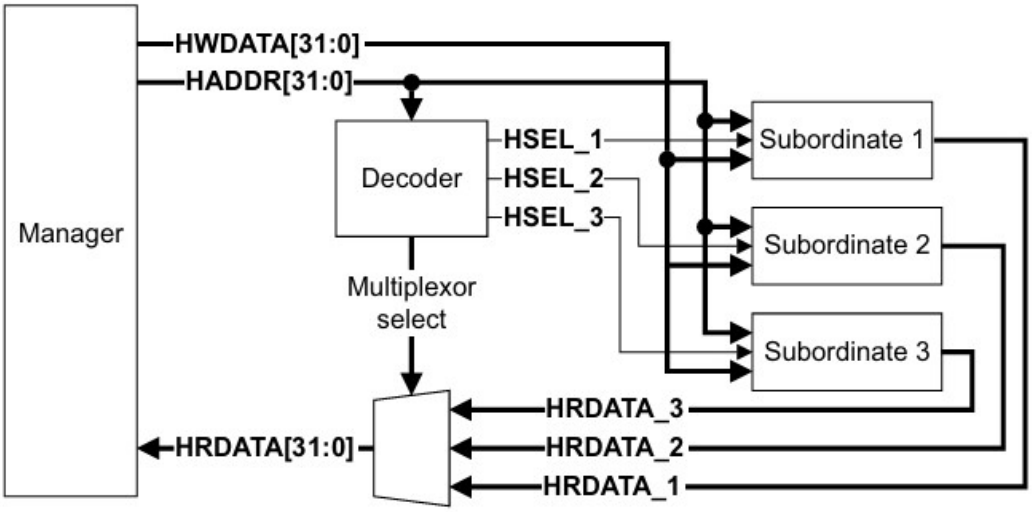
Với những tiến bộ đáng kể trong công nghệ VLSI, số lượng bóng bán dẫn có thể tích hợp trên một con chip ngày càng tăng. Điều này tạo điều kiện cho việc xây dựng các hệ thống SoC có độ tích hợp cao. Tuy nhiên, để đảm bảo khả năng trao đổi dữ liệu hiệu quả giữa các khối chức năng trong SoC, cần có một kiến trúc truyền thông nội bộ mạnh mẽ và linh hoạt. Kiến trúc bus AMBA do ARM đề xuất đã trở thành một trong những lựa chọn hàng đầu nhờ tính mở và khả năng mở rộng cao.

Trong bối cảnh thiết kế vi mạch ngày càng phức tạp, quá trình xác minh hệ thống cũng trở thành một bước quan trọng và tốn thời gian. Vì vậy, nghiên cứu này hướng tới việc thiết kế giao thức AHB sử dụng ngôn ngữ mô tả phần cứng Verilog, đồng thời áp dụng kỹ thuật kiểm chứng độ bao phủ (Coverage) để đảm bảo tính đầy đủ và chính xác của quá trình xác minh.

Ngày nay, kiến trúc AMBA không còn chỉ xuất hiện trong môi trường nghiên cứu hay thử nghiệm mà đã được ứng dụng rộng rãi trong các sản phẩm thương mại SoC. Bus AHB – một thành phần quan trọng của AMBA – có thể được tích hợp dưới nhiều hình thức khác nhau: từ các IP lõi mềm (soft IP) đến IP lõi cứng (hard IP), hoặc như một thành phần tích hợp trong toàn bộ hệ thống bus SoC. Chính vì

khả năng tương thích cao và cấu trúc mở, AHB đã trở thành một lựa chọn tiêu chuẩn trong thiết kế SoC hiện đại.

2.1.2. Mô hình hoạt động của giao thức AMBA AHB:



Hình 2. Mô hình AHB

Hình 2 biểu diễn cho mô hình hệ thống AHB trong tài liệu của AMBA, các khối hệ thống được đề xuất được mô tả như sau:

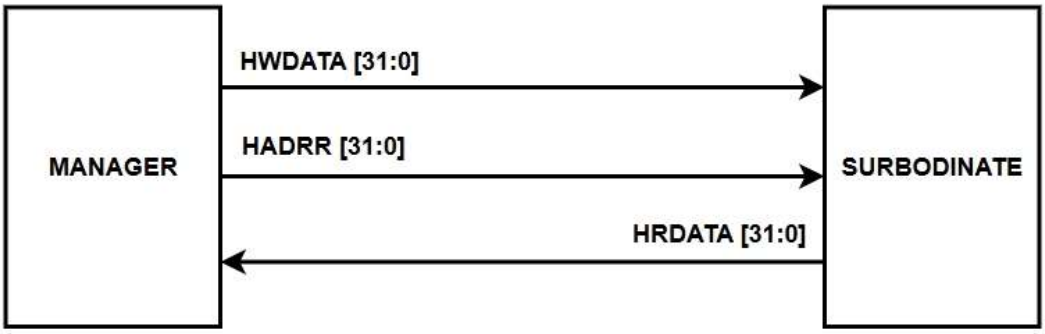
- Bus chủ (Master): cho phép hoạt động ghi và đọc thông qua cung cấp địa chỉ và các tín hiệu điều khiển. Trong một thời điểm hệ thống hoạt động thì chỉ có một bus chủ được phép tham gia hoạt động.

- Bus tớ (Slave): bus tớ sẽ thực hiện một hoạt động đọc hoặc ghi thông qua địa chỉ được cung cấp từ bus chủ trước đó. Bus tớ cũng sẽ trả về các tín hiệu cho bus chủ biết quá trình truyền có thành công, thất bại hay đang được thực thi.

Bộ giải mã địa chỉ (Address Decoder): được dùng với chức năng giải mã tín hiệu địa chỉ trong mỗi lần truyền, đưa ra tín hiệu lựa chọn bus tớ cần truyền và tín hiệu điều khiển cho khối ghép kênh dữ liệu đọc.

Các bộ ghép kênh tín hiệu (Multiplexor): là các bộ ghép kênh tín hiệu dữ liệu đọc/ghi, tín hiệu phản hồi và tín hiệu địa chỉ trong hệ thống bus, những tín hiệu này sẽ được truyền đến các bus chủ và bus tớ tương ứng.

Tuy nhiên hệ thống mà nhóm hướng đến là mô hình 1 chủ 1 tớ nên sẽ không có bộ giải mã địa chỉ (Address Decoder) và bộ ghép kênh tín hiệu (Muxtiplexor).



Hình 3. Sơ đồ AHB Bus

Theo như mô hình ở hình 3 thì hoạt động của khối được mô tả như sau: khối Master (trong hình gọi là MANAGER) đóng vai trò trung tâm điều khiển các giao dịch truyền dữ liệu giữa các thành phần. Hoạt động của Master được đồng bộ theo tín hiệu xung nhịp HCLK và có thể được khởi tạo hoặc đặt lại thông qua tín hiệu reset chủ động mức thấp HRESETn.

Khi bắt đầu một giao dịch, Master phát ra các tín hiệu điều khiển gồm: địa chỉ đích HADDR, chế độ truyền HTRANS, kiểu truy cập đọc/ghi HWRITE, và kích thước dữ liệu HSIZE.

Trong đó, HTRANS là tín hiệu quan trọng xác định loại giao dịch: IDLE (00) thể hiện không hoạt động, NONSEQ (10) dùng để bắt đầu một giao dịch mới, và SEQ (11) thể hiện giao dịch tiếp nối trong chế độ burst.

Nếu giao dịch là ghi, dữ liệu sẽ được đặt lên bus thông qua HWDATA; nếu là đọc, dữ liệu sẽ được lấy từ HRDATA. Suốt quá trình, Master theo dõi HREADY để biết Slave đã sẵn sàng tiếp nhận hoặc phản hồi chưa, đồng thời giám sát HRESP để kiểm tra phản hồi trạng thái (thành công hoặc lỗi). Cơ chế điều phối này đảm bảo hoạt động truyền dữ liệu chính xác và giúp tăng hiệu năng của bus AHB.

Quá trình ghi dữ liệu sẽ được thực hiện từ bus chủ ghi đến bus tớ và quá trình đọc sẽ đọc từ bus tớ đến bus chủ, các quá trình trên đều thực hiện trong một chu kỳ truyền tín hiệu địa chỉ, tín hiệu điều khiển và một hoặc có thể nhiều chu kỳ truyền dữ liệu. Việc truyền tín hiệu địa chỉ không cần nhiều chu kỳ do các bus tớ

luôn lấy mẫu địa chỉ trong xuyên suốt quá trình truyền. Tuy nhiêu quá trình truyền dữ liệu có thể có thêm nhiều chu kỳ thông qua tín hiệu HREADY, tín hiệu này sẽ tạo ra một trạng thái chờ trong quá trình truyền khi ở mức thấp. Trạng thái chờ này cho phép bus tớ có thêm thời gian xử lý và lấy mẫu dữ liệu.

Một quá trình truyền sẽ được kết thúc hoặc truyền lại được quyết định thông qua tín hiệu HRESP[1:0] của bus tớ. Bus tớ luôn thông báo cho bus chủ về trạng thái truyền trong một chu kỳ truyền, có bốn trạng thái được bus tớ phản hồi cho bus chủ là:

• OKAY: Thông báo quá trình truyền đang được diễn ra bình thường và quá trình truyền kết thúc khi tín hiệu HREADY lên mức cao.

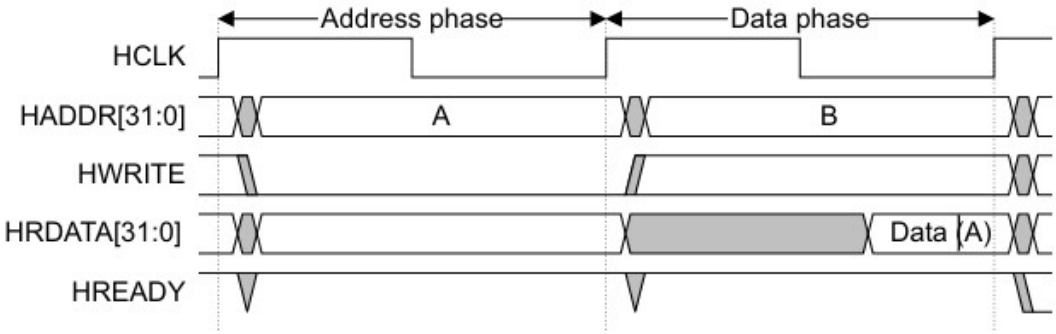
• ERROR: Thông báo đã có lỗi phát sinh trong quá trình truyền và vi vậy quá trình truyền sẽ không thành công.

• RETRY: Thông báo quá trình truyền có một địa chỉ không thực hiện truyền được và yêu cầu bus chủ gửi lại địa chỉ vừa truyền cho đến khi quá trình truyền hoàn tất.

• SPLIT: Thông báo quá trình truyền chưa thể hoàn thành ngay, nhưng bus chủ có thể tiếp tục quá trình đang bỏ dỡ vào lần truy cập bus tiếp theo.

2.1.3. Chế độ truyền dữ liệu

a. Quá trình truyền cơ bản:



Hình 4. Quá trình truyền

Hình 4 biểu diễn quá trình đọc của hệ thống gồm các bước sau:

- Giai đoạn địa chỉ (Address Phase):

+ Trong chu kỳ đầu, master đưa lên bus:

+ HADDR = A: địa chỉ muốn đọc.

+ HWRITE = 0: xác định đây là một giao dịch đọc.

+ HREADY đang ở mức cao → bus sẵn sàng nhận giao dịch mới.

+ Đây là thời điểm địa chỉ được trình bày cho slave, nhưng chưa có dữ liệu đọc được trả về ngay.

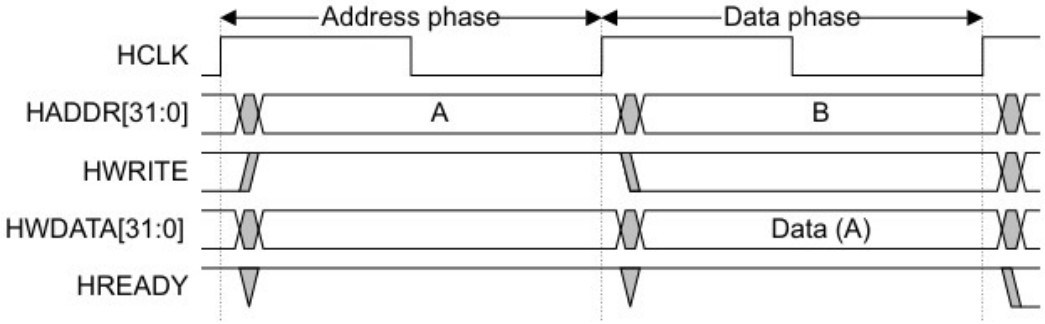
- Giai đoạn dữ liệu (Data Phase):

+ Ở chu kỳ kế tiếp, địa chỉ mới B được đưa lên HADDR (cho giao dịch kế tiếp), nhưng dữ liệu Data {A} tương ứng với địa chỉ A sẽ:

+ Xuất hiện tại HRDATA sau một chu kỳ trễ.

+ Được đặt lên bởi slave sau khi xử lý địa chỉ A ở chu kỳ trước.

+ HREADY vẫn giữ mức cao → xác nhận rằng dữ liệu từ slave đã sẵn sàng.



Hình 5. Quá trình nhận

Hình 5 biểu diễn quá trình ghi của hệ thống gồm các bước:

- Address Phase (Giai đoạn địa chỉ):

+ Tín hiệu HADDR = A: master phát địa chỉ A lên bus để ghi dữ liệu.

+ Tín hiệu HWRITE = 1: xác định đây là giao dịch ghi.

+ HREADY = 1: slave sẵn sàng tiếp nhận, nên giao dịch có thể tiến hành mà không bị chặn.

+ Tại thời điểm này, dữ liệu chưa được gửi đi — chỉ là địa chỉ và thông tin điều khiển.

- Data Phase (Giai đoạn dữ liệu):

+ Ở chu kỳ kế tiếp:

+ Master đưa dữ liệu cần ghi (tương ứng địa chỉ A) lên HWDATA.

+ HADDR đồng thời thay đổi sang giá trị B cho giao dịch kế tiếp.

+ Slave lấy Data (A) từ HWDATA và ghi vào vị trí địa chỉ A.

b. Các kiểu truyền :

Trong khi diễn ra quá trình truyền, bus chủ sẽ thông báo cho bus tớ về kiểu truyền đang được thực hiện. Việc thông báo này giúp cho bus tớ có thể biết được quá trình truyền nào đang được diễn ra. Điều này cho phép bus tớ biết 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 vào một trong bốn kiểu truyền: IDLE, BUSY, NONSEQ, SEQ như được trình bày trong bảng 2.1. Việc mã hóa các kiểu truyền này sẽ được thực hiện thông qua tín hiệu HTRANS truyền từ bus chủ sang bus tớ.

Bảng 1. Các kiểu truyền

HTRANS [1:0]	Kiểu	Mô tả
00	IDLE	Báo hiệu không có quá trình truyền dữ liệu nào được thực hiện. Kiểu IDLE được dùng khi một bus chủ mặc định được phép sử dụng bus. Các bus tớ phải luôn cung cấp một trạng thái chờ với HREADY ở mức thấp và gửi phản hồi OKAY đối với một quá trình IDLE.
01	BUSY	Kiểu truyền BUSY cho phép bus chủ thêm các chu kỳ trống vào giữa quá trình truyền khối. Kiểu này báo hiệu rằng bus chủ vẫn đang tiếp tục với một quá trình truyền khối nhưng khối truyền tiếp theo sẽ chưa được thực hiện ngay lập tức. Các bus tớ phải luôn cung cấp một trạng thái chờ và gửi phản hồi OKAY tương tự như đối với quá trình IDLE. Lúc này quá trình truyền sẽ bị bus tớ bỏ qua.
10	NONSEQ	Báo hiệu khối truyền đầu tiên của một quá trình truyền khối hoặc là một quá trình truyền đơn. Tín hiệu điều khiển và địa chỉ không liên quan đến quá trình truyền trước đó.
11	SEQ	Báo hiệu khối truyền đầu tiên của một quá trình truyền khối hoặc là một quá trình truyền đơn. Tín hiệu điều khiển và địa chỉ không liên quan đến quá trình truyền trước đó.

c. Phản hồi của bus tớ

Sau khi một bus chủ bắt đầu một quá trình truyền, bus tớ sẽ quyết định xem quá trình truyền nên tiến hành như thế nào. Trong cấu trúc của hệ thống bus AHB

không cho phép bus chủ hủy một quá trình truyền khi nó đã diễn ra. Khi một bus tớ được tham gia vào quá trình truyền, nó sẽ cung cấp các tín hiệu phản hồi để báo hiệu tình trạng hiện tại của quá trình truyền. Tín hiệu HREADY được sử dụng để mở rộng một quá trình truyền và nó hoạt động kết hợp với một tín hiệu phản hồi khác (HRESP[1:0]) để cung cấp thêm thông tin về trạng thái của quá trình truyền[5]. Có bốn loại phản hồi truyền là OAKY, ERROR, RETRY, SPLIT và được quy định bởi tín hiệu HRESP[1:0] như được mô tả ở bảng 2.3.

Bảng 2. Các kiểu phản hồi

HRESP [1:0]	Kiểu	Mô tả
00	OKAY	Cùng với tín hiệu HREADY ở mức cao thông báo một quá trình truyền đã hoàn tất. Ngoài ra, phản hồi này còn được sử dụng khi có bất kỳ chu kỳ nào được chèn thêm trong lúc HREADY ở mức thấp, trước khi gửi một trong ba phản hồi còn lại.
01	ERROR	Chỉ ra một lỗi đã phát sinh. Điều kiện lỗi cần được báo cho bus chủ để cảnh báo quá trình truyền đã thất bại. Ta cần ít nhất hai chu kỳ để gửi một điều kiện lỗi này.
10	RETRY	Chỉ ra một địa chỉ đã không thực hiện truyền được. Bus chủ cần thử gửi lại địa chỉ vừa truyền lại cho đến khi quá trình truyền hoàn tất. Phản hồi này cũng cần hai chu kỳ xung.
11	SPLIT	Quá trình truyền chưa thể hoàn tất ngay lập tức. Bus chủ sẽ tiếp tục quá trình truyền đang bỏ dở vào lần được phép truy cập bus tiếp theo. Phản hồi này cũng yêu cầu hai chu kỳ xung.

Một bus tớ có thể hoàn thành một quá trình truyền theo nhiều cách khác nhau:

- Hoàn tất quá trình truyền ngay lập tức.

- Thêm vào một hoặc nhiều trạng thái chờ nhằm thêm thời gian cần thiết để hoàn thành quá trình truyền.

- Báo hiệu lỗi để thông báo một quá trình truyền thất bại.

- Tạm hoãn sự hoàn tất của một quá trình truyền, nhưng cho phép bus chủ và bus tớ thoát khỏi bus để bus có thể sẵn sàng cho những quá trình truyền khác.

Như vậy chỉ có quá trình phản hồi OKAY là được thực hiện trong một chu kỳ xung. Còn các quá trình phản hồi còn lại đều cần ít nhất hai chu kỳ xung. Trong chu kỳ xung đầu tiên, bus tớ đưa tín hiệu HREADY về mức thấp, đồng thời phát đi một phản hồi thích hợp để thông báo cho bus chủ biết. Ở chu kỳ tiếp theo, bus tớ vẫn giữ nguyên tín hiệu phản hồi HRESP nhưng đưa tín hiệu HREADY về lại mức cao để thông báo quá trình truyền phản hồi đã hoàn tất. Việc sử dụng hai chu kỳ xung để phát các phản hồi ERROR, RETRY, SPLIT là để bus chủ có thêm thời gian hoãn quá trình truyền vừa thực hiện và thay đổi lại quá trình truyền cho phù hợp với phản hồi được bus tớ gửi đi.

2.2. GIỚI THIỆU VỀ NGÔN NGỮ MÔ TẢ PHẦN CỨNG VERILOG

2.2.1. Verilog HDL là gì?

Verilog HDL là ngôn ngữ mô tả phần cứng (Hardware Description Language) được sử dụng nhiều trong nhiều thiết kế hệ thống số. Verilog đươc sử dụng để mô hình hóa các hệ thống điện tử cho mục đích tổng hợp, cũng như được sử dụng trong việc xác minh với mục đích để mô phỏng.

Cùng với ngôn ngữ VHDL, Verilog HDL là một trong hai ngôn ngữ mô tả phần cứng được sử dụng phổ biến nhất hiện nay. Nó đáp ứng tất cả những yêu cầu cho việc thiết kế và tổng hợp các hệ thống số. Verilog cho phép mô tả phần cứng không chỉ ở mức cổng (gate level), và mức chuyển dịch thanh ghi (register-transfer level -RTL), mà còn cho phép mô tả theo thuật toán trừu tượng. Verilog cũng hỗ trợ mạnh về các mức độ mô tả định thời cho việc mô phỏng.

Hiện nay trên thị trường có rất nhiều phần mềm và công cụ có môi trường dựa trên Verilog cung cấp khả năng tổng hợp thiết kế, kiểm tra thiết kế và mô phỏng đa dạng với những công cụ tạo dạng sóng cũng như giao diện đồ họa trước khi layout.

2.2.2. Lịch sử phát triển của Verilog

Verilog được phát triển vào giữa những năm 1980 bởi Gateway Design Automation và sau đó được chuyển giao cho IEEE (viện kỹ sư Điện - Điện tử). Ban đầu, ngôn ngữ được sử dụng như một công cụ kiểm tra và mô phỏng, và sau này năm 1987 công cụ tổng hợp này đã được xây dựng và phát triển dựa vào ngôn ngữ này.

Vào năm 1990, Gateway Design Automation Inc đã được Candence Design System mua lại, hiện là một trong những nhà cung cấp lớn nhất về các công nghệ thiết kế điện tử và dịch vụ kỹ thuật trong ngành công nghiệp tự động hóa thiết kế điện tử (EDA). Cadence nhận ra giá trị của Verilog và nhận ra rằng nếu Verilog vẫn là ngôn ngữ khép kín, áp lực của tiêu chuẩn hóa cuối cùng sẽ khiến mọi người chuyển sang dùng VHDL. Vì vậy, vào năm 1991, Verilog International (OVI) được tổ chức Candence và tài liệu của Verilog đã được chuyển sang pham vị công cộng với tên OVI, và sau đó gửi cho IEEE. Verilog được xác định chính thức bởi tiêu chuẩn IEEE 1364 và được phê duyệt vào năm 1995 (gọi tắt là Verilog -1995).

Vào năm 2001, các phần mở rộng cho Verilog-1995 đã được gửi lại cho IEEE và trở thành tiêu chuẩn IEEE 1364-2001 được gọi là Verilog-2001. Các tiện ích mở rộng bao gồm một số thiếu sót mà người dùng đã tìm thấy trong Verilog-1995. Một trong những nâng cấp quan trọng nhất là các biến có chữ ký đã được hỗ trợ. Verilog-2001 hiện là phiên bản thống trị của Verilog được hỗ trợ hầu hết các công cụ thiết kế hiện nay.

Năm 2005, Verilog-2005 (tiêu chuẩn IEEE 1364-2005) đã được xuất bản với các sửa đổi. Cũng trong năm 2005, System Verilog , một phiên bản cao cấp của Verilog 2005, với nhiều tính năng và khả năng mới để hỗ trợ xác minh thiết kế, đã được ra mắt.

Kể từ năm 2009, các tiêu chuẩn ngôn ngữ của System Verilog và Verilog đã được sáp nhập vào SystemVerrilog-2009 (tiêu chuẩn IEEE 1800-2009), là một trong những ngôn ngữ phổ biến nhất để thiết kế và xác minh IC ngày nay.

2.2.3. Các thuật ngữ trong Verilog

Có bốn thuật ngữ quan trọng thường gặp trong Verilog như sau:

HDL (Hardware Description Language):

Là ngôn ngữ mô tả phần cứng, được sử dụng để mô tả một hệ thống kỹ thuật số như bộ chuyển mạch mạng, bộ vi xử lý, bộ nhớ hoặc một flip-flop. HDL cung cấp hai phương pháp để mô hình hóa mạch số: Behavior Modeling (Mô hình hành vi), Structural Modeling (Mô hình cấu trúc).

RTL (Register Transfer Level):

Là một cấp độ trừu tượng trong thiết kế kỹ thuật số, mô hình hóa luồng dữ liệu giữa các thanh ghi phần cứng và các phép toán logic thực hiện trên các tín hiệu đó. RTL thường được sử dụng trong HDL để tạo ra các mô tả cấp cao của mạch, từ đó có thể triển khai thành các cấp thấp hơn và cuối cùng là hệ thống dây vật lý. Thiết kế ở cấp độ RTL là một phương pháp phổ biến trong thiết kế mạch kỹ thuật số hiện đại.

Structural Modeling (Mô hình cấu trúc):

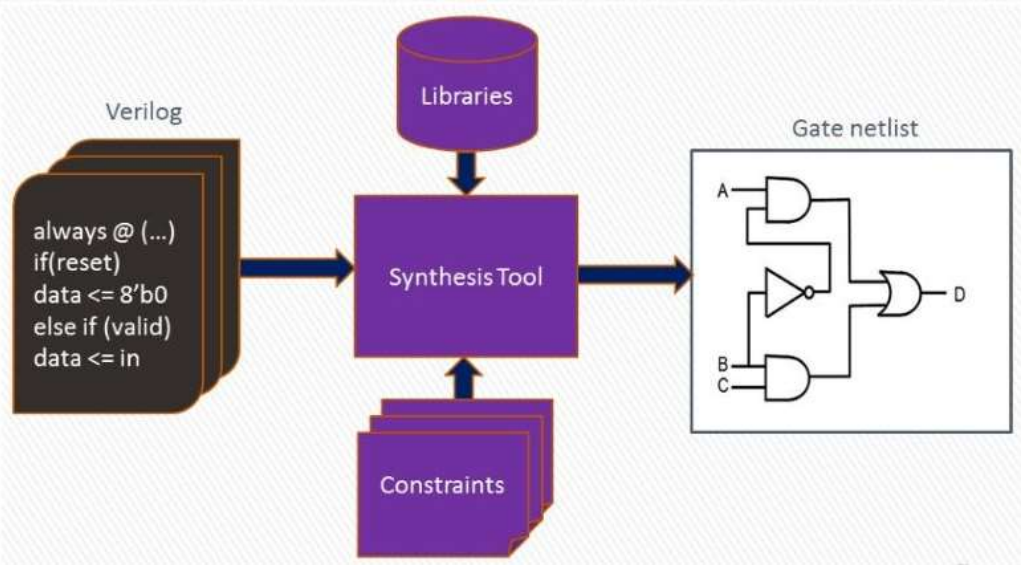
Là cách mô tả mạch bằng cách kết nối các thành phần phần cứng, các mô-đun, và các cổng logic với nhau. Mô hình này phản ánh cấu trúc phần cứng thực tế của hệ thống.

Behavior Modeling (Mô hình hành vi):

Sử dụng các câu lệnh thủ tục để điều khiển mô phỏng và thao tác với các biến kiểu dữ liệu. Các câu lệnh này thường nằm trong các thủ tục (như always, initial) và mô tả luồng hoạt động của mạch một cách trực quan và gần với thuật toán.

2.2.4. Các đặc tính của Verilog:

- Tổng hợp:



Hình 6. Mô hình Verilog

Hình 6 minh họa quy trình tổng hợp RTL. Ở bước đầu tiên, tổng hợp chuyển đổi mã quy tắc của bạn thành phần cứng bằng cách sử dụng các thành phần kiến trúc được có sẵn trong thư viện của bạn. Sau đó nó sẽ đến bước tối ưu hoá để đảm bảo mô tả mạch của bạn có thể được tạo ra một cách tốt nhất.

- Mức độ hành vi

Trong mô hình hành vi (Behavior Modeling) , bạn sẽ mô tả chức năng của mạch chứ không phải cấu trúc của mạch. Hành vi ngõ ra (output) được mô tả theo mối quan hệ với 24 các các ngõ vào (input). Trong Verilog có nhiều mức khác nhau để thiết kế mạch tích hợp như: Register Transfer Level (RTL), mức GATE và một số mức khác.

- Mức độ cổng

Mức độ này là một thiết kế được thực hiện bằng các thuật ngữ cổng logic cơ sở (NAND, NOR, AND, OR, MUX, FLIP-FLOP). Với mức độ cổng (Gate-level) thì gần như có mức độ trừu tượng thấp nhất, vì vậy mức độ cổng thường được dùng để thực hiện các mô-đun mức thấp nhất trong thiết kế như: Multiplexers, bộ

cộng,….Một hệ thống số logic chỉ có thể có các giá trị logic nhất định (0,1,x,z), sử dụng mô hình hóa mức cổng không phải là một ý tưởng hay ở bất kỳ một thiết kế logic nào. Tuy nhiên, có thể mô hình hóa mức cổng chính xác hơn thì những tín hiệu Verilog này thêm về độ mạnh của mức giá trị vào các logic nhất định trên.

2.3. GIỚI THIỆU VỀ PHẦN MỀM VIVADO

Vivado Design Suite là một công cụ thiết kế phần cứng kỹ thuật số được phát triển bởi Xilinx (nay thuộc AMD), chuyên dùng để thiết kế, mô phỏng và triển khai các hệ thống số trên các dòng FPGA hiện đại như Artix, Kintex, Virtex, và Zynq. Vivado được phát hành để thay thế phần mềm ISE Design Suite trước đây, mang lại khả năng tối ưu hóa tốt hơn, giao diện trực quan hơn và hỗ trợ sâu hơn cho các dòng thiết bị FPGA mới.

Vivado tích hợp nhiều công cụ mạnh mẽ, bao gồm:

- Thiết kế RTL: Người dùng có thể viết mã mô tả phần cứng bằng Verilog hoặc VHDL để xây dựng các hệ thống số phức tạp.
- Tổng hợp (Synthesis): Quá trình chuyển mã RTL thành sơ đồ logic (netlist), chuẩn bị cho giai đoạn ánh xạ.
- Triển khai (Implementation): Gồm ba bước chính: ánh xạ (mapping), bố trí (placement) và định tuyến (routing) logic trên tài nguyên phần cứng của FPGA.
- Thiết kế sơ đồ khối (IP Integrator): Cho phép người dùng kéo thả các khối IP có sẵn (ví dụ: UART, GPIO, DDR, AXI...) và tự động kết nối bus.
- Mô phỏng (XSim): Mô phỏng hành vi của hệ thống trước khi triển khai thực tế.
- Phân tích timing: Đánh giá các vi phạm về thời gian (setup, hold) trong thiết kế.

Vivado hỗ trợ ba luồng thiết kế chính:

- RTL Design Flow: Viết Verilog/VHDL thủ công, phù hợp cho các kỹ sư có kinh nghiệm.

- Block Design Flow: Sử dụng IP Integrator để xây dựng hệ thống qua sơ đồ khối, phù hợp cho phát triển SoC với bus AXI.
- High-Level Synthesis (HLS): Thiết kế phần cứng bằng ngôn ngữ C/C++, giúp rút ngắn thời gian phát triển và tăng tính trừu tượng.

Vivado có mặt rộng rãi trong nhiều lĩnh vực công nghệ cao:

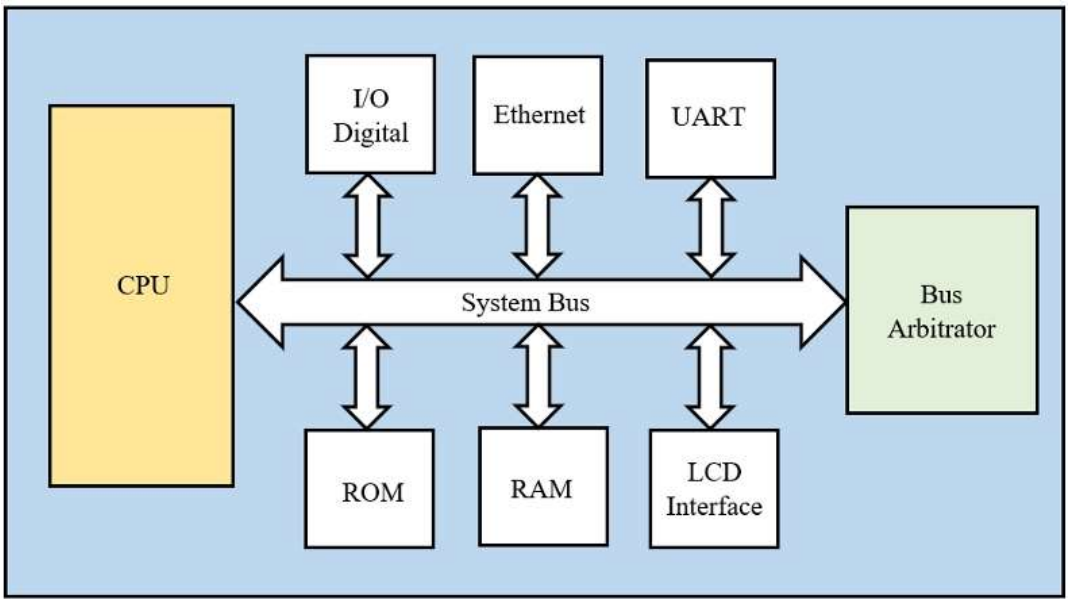
- Thiết kế các hệ thống xử lý tín hiệu số (DSP) trong truyền thông, radar, xử lý ảnh...
- Xây dựng hệ thống nhúng tùy chỉnh với bộ xử lý ARM và các khối IP phần cứng tích hợp.
- Giao tiếp ngoại vi như UART, SPI, I2C, Ethernet, DDR, PCIe...
- Ứng dụng công nghiệp và quốc phòng, như trong các thiết bị điều khiển thời gian thực, các hệ thống an toàn cao.
- Nghiên cứu và đào tạo, là công cụ phổ biến trong các trường đại học và viện nghiên cứu chuyên ngành điện – điện tử – điều khiển.

CHƯƠNG 3. THIẾT KẾ HỆ THỐNG

3.1. VẤN ĐỀ THIẾT KẾ

Việc nghiên cứu và phát triển các ứng dụng kiểu hệ thống trên chip (SoC) trong lĩnh vực đa phương tiện thì việc triển khai một hệ bus AHB tốc độ cao nhằm đảo bảo các yêu cầu về truyền thông của hệ thống là rất cần thiết. Tại phòng thí nghiệm Hệ thống tích hợp thông minh (SIS) đã tiến hành xây dựng một hệ thống trên chip với tên gọi COMOSY (Controlling Monitoring System) . Trong đó các lõi IP có hiệu năng hoạt động cao và lưu lượng truyền thông lớn được kết nối với nhau qua hệ thống bus tốc độ cao AHB nhằm đảm bảo yêu cầu truyền thông. Các lõi IP có hiệu năng hoạt động thấp hơn như GPIO, USART… được kết nối với bus APB. Hai hệ thống bus này được kết nối thông qua cầu bus. Đồ án này chỉ tập trung vào nghiên cứu, thiết kế và kiểm chứng một hệ thống bus AHB.

Hệ thống COMOSY là một hệ thống trên chip được phát triển cho các ứng dụng trong lĩnh vực đa phương tiện giám sát môi trường và có khả năng giao tiếp với mạng máy tính chuẩn Ethernet. Hệ thống COMOSY được xây dựng bao gồm một vi xử lý 32 bit đóng vai trò xử lý chính và một số lõi IP chức năng như: khối nhớ (RAM ,ROM), khối giao tiếp ethernet, khối giao tiếp LCD, khối giao tiếp vào/ra và có thể triển khai mã hóa, giải mã chuẩn ảnh JPEG hoặc tín hiệu video trong tương lai. Toàn bộ hệ thống được mô tả như trong hình.



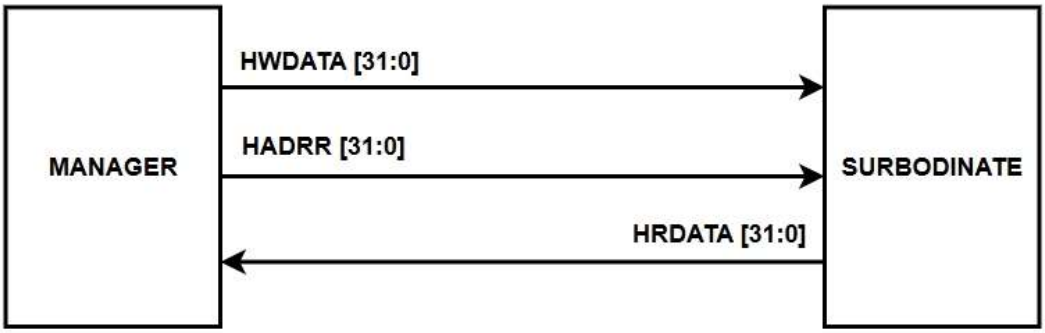
Hình 7. Hệ thống chip COMOSY

Hình 7 mô tả platform (nền tảng) hệ thống trên chip COMOSY, system bus (hệ thống bus) bao gồm: bus tốc độ cao AHB và bus ngoại vi nâng cao APB. Trong đó bus tốc độ cao AHB được dùng cho các ứng dụng ROM, RAM, LCD Interface, bus ngoại vi APB được dùng cho các ứng dụng I/O Digital, Ethernet, UART. Từ sơ đồ khối ta thấy rằng số lượng bus chủ và bus tớ cần sử dụng trong hệ thống là không nhiều. Do đó, để đáp ứng nhu cầu xử lý và truyền dữ liệu với tốc độ cao của một hệ thống xử lý đa phương tiện thì việc truyền thông giữa các loại IP cũng đặt ra yêu cầu về tốc độ truyền cao. Chính vì vậy mà hệ thống bus AHB là một lựa chọn phù hợp cho vấn đề truyền thông trên chip đã được nên ra ở trên.

3.2. THIẾT KẾ SƠ ĐỒ KHỐI HỆ THỐNG

Để có thể dễ dàng tiếp cận và tìm hiểu phương thức hoạt động của bus AHB, cũng như để dễ dàng xây dựng nên mô hình của hệ thống nên ở đồ án này chỉ hướng đến mục tiêu phát triển 1 bus chủ và 1 bus tớ trong hệ thống. Tuy nhiên hệ thống này vẫn có phát triển mở rộng hỗ trợ giao tiếp lên thêm nhiều chủ và tớ nên có thêm khối phân xử và khối giải mã, ngoài ra hệ thống bus có thể giao tiếp được với nhiều loại IP khác nhau mà không cần phải thay đổi nhiều về cấu trúc. Qua các

phân tích, đánh giá trên, đồ án đề xuất mô hình sơ đồ khối Bus AHB được xây dựng như hình.



Hình 8. Sơ đồ hệ thống

Hình 8 mô tả sơ đồ khối hệ thống sử dụng AHB bus bao gồm 1 bus chủ và 1 bus tớ. Trong đó bộ phân xử đóng vai trò phân quyền hoạt động cho 2 khối bus chủ.

- Khối bus chủ (Manager) có chức năng thực hiện việc truyền dữ liệu và chọn địa chỉ vào bus tớ.

-Khối bus tớ (Subordinate) có chức năng nhận dữ liệu và đọc dữ liệu ra từ tín hiệu của bus chủ.

3.3. THIẾT KẾ CHI TIẾT TỪNG KHỐI

3.3.1. Khối Master

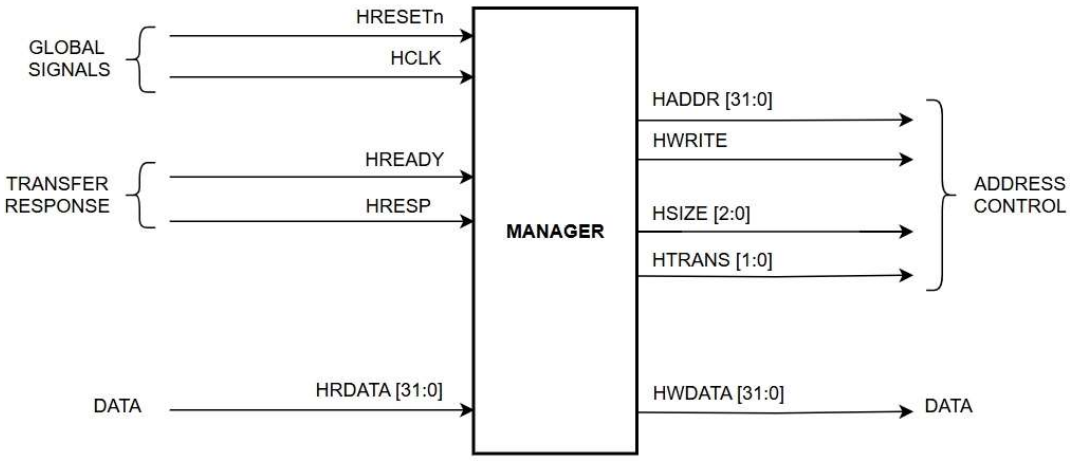
Để thực hiện việc truyền dữ liệu giữa bus chủ và bus tớ, sau khi nghiên cứu và tham khảo tài liệu mô tả kỹ thuật của AMBA AHB, đồ án này đề xuất giao diện của bus chủ được mô tả như hình 3.3.

Hình 3.3 mô tả giao diện kết nối của khối bus chủ sau khi thông qua yêu cầu của hệ thống và thông số kỹ thuật của AMBA AHB bus. Các tín hiệu của khối được chia thành các nhóm sau:

• Global signals: nhóm tín hiệu xung nhịp và đặt lại của hệ thống.

• Transfer response: các tín hiệu phản từ khối bus tớ sau khi thực hiện xong một quá trình truyền nhận.

• Address and control signals: các tín hiệu được phát cho hệ thống sau khi nhận được từ các tín hiệu từ nhóm tín hiệu kiểm tra.



Hình 9. Khối chủ

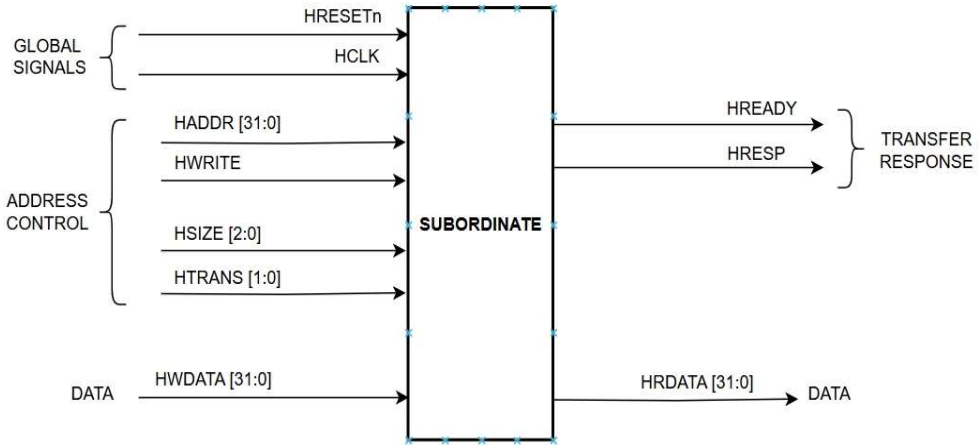
Bảng 9 trình bày chi tiết các tín hiệu của khối bus chủ, cho biết đọ rộng, ngõ vào/ra và mô tả chức năng của từng tín hiệu.

Bảng 3. Các tín hiệu của khối chủ

Tín hiệu	I/O	Độ rộng	Mô tả
Hresetn	Input	1	Tín hiệu reset hệ thống
Hclk	Input	1	Xung clock hệ thống
Hready	Input	1	Tín hiệu được bus tớ sử dụng báo kết thúc quá trình truyền
Hresp	Input	2	Tín hiệu phản hồi của bus tớ, thông báo về trạng thái truyền hiện tại của bus tớ.
Hrdata	Input	32	Tín hiệu đọc được truyền từ bus tớ
Haddr	Output	32	Tín hiệu địa chỉ
Hwrite	Output	1	Tín hiệu cho phép ghi/đọc dữ liệu
Hsize	Output	2	Tín hiệu cho biết kích thước dữ liệu truyền
Hwdata	Output	32	Tín hiệu dữ liệu đọc

3.3.2. Khối Slave

Khối bus tớ có vai trò nhận dữ liệu ghi từ bus chủ và cung cấp dữ liệu ngược lại cho bus chủ, là nơi xử lý tất cả các tín hiệu theo tiêu chuẩn kỹ thuật của bus AHB.



Hình 10. Khối tớ

Hình 10 mô tả giao diện kết nối của khối bus tớ sau khi thông qua yêu cầu của hệ thống và thông số kỹ thuật của AMBA AHB bus, tín hiệu của khối bus tớ được chia thành các nhóm tín hiệu sau:

• Global signals: nhóm tín hiệu xung nhịp và đặt lại của hệ thống.

• Transfer response: các tín hiệu phản từ khối bus tớ sau khi thực hiện xong một quá trình truyền nhận.

• Address and control signals: các tín hiệu được phát cho hệ thống sau khi nhận được từ các tín hiệu từ nhóm tín hiệu kiểm tra.

Bảng 4. Các tín hiệu của khối tớ

Tín hiệu	I/O	Độ rộng	Mô tả
Hresetn	Input	1	Tín hiệu reset hệ thống
Hclk	Input	1	Xung clock hệ thống
Hready	Output	1	Tín hiệu được bus tớ sử dụng báo kết thúc quá trình truyền

Hresp	Output	2	Tín hiệu phản hồi của bus tớ, thông báo về trạng thái truyền hiện tại của bus tớ.
Hrdata	Output	32	Tín hiệu đọc được truyền từ bus tớ
Haddr	Input	32	Tín hiệu địa chỉ
Hwrite	Input	1	Tín hiệu cho phép ghi/đọc dữ liệu
Hsize	Input	2	Tín hiệu cho biết kích thước dữ liệu truyền
Hwdata	Input	32	Tín hiệu dữ liệu đọc

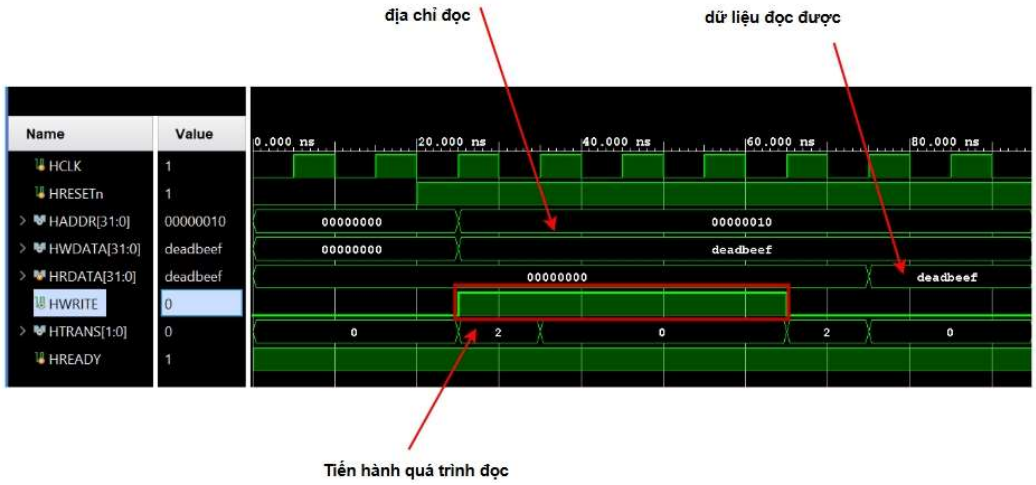
CHƯƠNG 4. KẾT QUẢ ĐÁNH GIÁ

4.1. Giới thiệu testcase:

Bảng 5. Bảng testcase

Testcase	Testname	Cấu hình
1	Test module master	HADDR = 32'h1000_0010, HWDATA = 32'h DEAD_BEEF
2	Test module slave	HADDR = 32'h1000_0020, HWDATA = 32'h A5A5_A5A5
3	Quá trình đọc	HADDR = 32'h1000_0004, HWDATA = 32'h CCCC_DDDD, HWRITE = 0
4	Quá trình ghi	HADDR = 32'h1000_0000, HWRITE = 1
5	Ghi vào địa chỉ lỗi	HADDR = 32'hFFDF_FDFF, HWDATA = 32'h 5555_5555, HWRITE = 1
6	Đọc từ địa chỉ lỗi	HADDR = 32'hFFDF_FDFF, HWRITE = 0
7	Ghi với Htrans IDLE	HADDR = 32'h1000_0004, HWDATA = 32'h 1234_5678, HWRITE = 1
8	Đọc từ Htrans IDLE	HADDR = 32'h1000_0004, HWRITE = 0
9	Ghi liên tục	HADDR = 32'h4000_0000, HADDR = 32'h4000_0004, HADDR = 32'h4000_0008, HWRITE = 1
10	Đọc liên tục	HADDR = 32'h4000_0000, HADDR = 32'h4000_0004, HADDR = 32'h4000_0008, HWRITE = 0

4.2. Testcase 1 (test module master)



Hình 11. Test module master

Hình 11 mô tả hoạt động ghi và đọc dữ liệu của khối bus chủ (master), khi nhóm tín hiệu kiểm tra được cấp dữ liệu thì khối bus chủ đã thực hiện đươc quá trình ghi và đọc dữ liệu qua tín hiệu HWRITE. Các trình tự hoạt động của các tín hiệu thuộc nhóm địa chỉ và điều khiển cũng đã thực hiện đúng theo tiêu chuẩn kỹ thuật của AMBA AHB Bus.

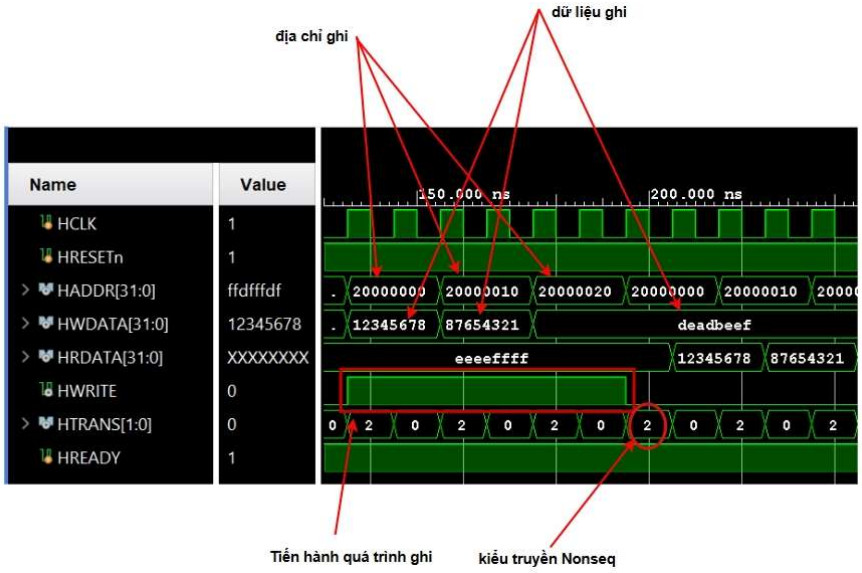
4.3. Testcase 2 (test module slave)

Hình 12. Test module slave

Hình 12 mô tả hoạt động ghi và đọc dữ liệu của khối bus tớ (slave), khi nhóm tín hiệu kiểm tra được cấp dữ liệu thì khối bus chủ đã thực hiện đươc quá trình ghi và đọc dữ liệu qua tín hiệu HWRITE. Các trình tự hoạt động của các tín hiệu thuộc

nhóm địa chỉ và điều khiển cũng đã thực hiện đúng theo tiêu chuẩn kỹ thuật của AMBA AHB Bus.

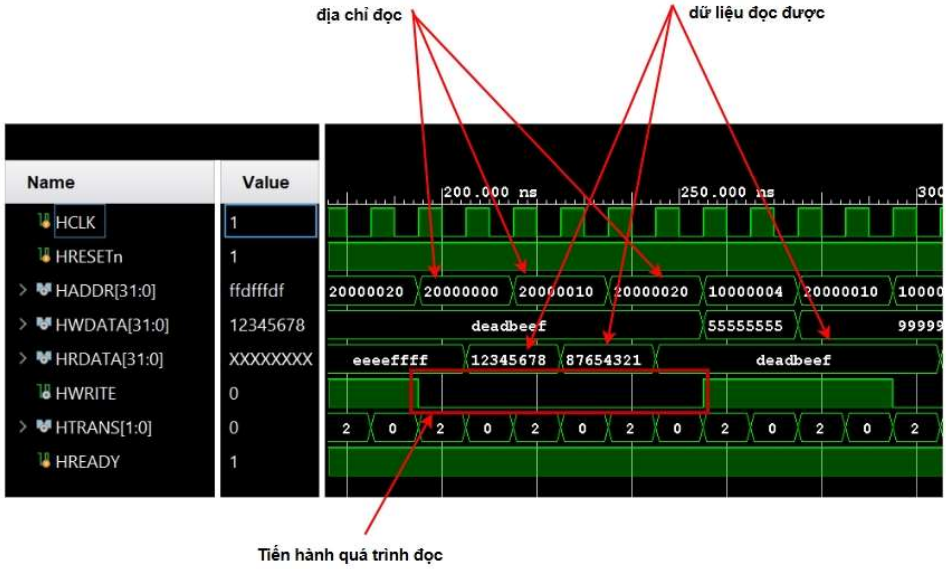
4.4. Testcase 3 (quá trình ghi)



Hình 13. Quá trình ghi

Hình 13 mô tả hoạt động ghi dữ liệu của toàn bộ hệ thống AHB, khi nhóm tín hiệu kiểm tra được cấp dữ liệu thì khối bus chủ sẽ gửi tín hiệu đọc hoặc ghi qua HWRITE, đồng thời bus chủ cũng cung cấp địa chỉ và dữ liệu muốn ghi, cùng với đó là kiểu truyền cho mỗi lần truyền (Nonseq - 01). Sau đó bus tớ sẽ nhận các tín hiệu đó và phản hồi lại qua HRDATA. Các trình tự hoạt động của các tín hiệu thuộc nhóm địa chỉ và điều khiển cũng đã thực hiện đúng theo tiêu chuẩn kỹ thuật của AMBA AHB Bus.

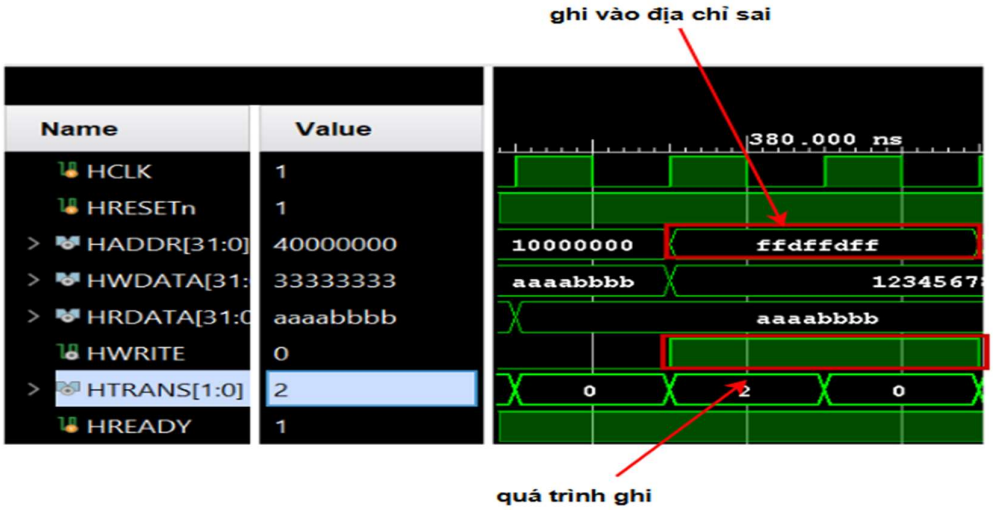
4.5. Testcase 4 (quá trình đọc)



Hình 14. Quá trình đọc

Hình 14 mô tả hoạt động đọc dữ liệu của toàn bộ hệ thống AHB, khi nhóm tín hiệu kiểm tra được cấp dữ liệu thì khối bus chủ sẽ gửi tín hiệu đọc qua HWRITE = 0, Sau đó bus tớ sẽ nhận các tín hiệu đó và phản hồi lại qua HRDATA đúng với giá trị đã ghi trước đó. Các trình tự hoạt động của các tín hiệu thuộc nhóm địa chỉ và điều khiển cũng đã thực hiện đúng theo tiêu chuẩn kỹ thuật của AMBA AHB Bus.

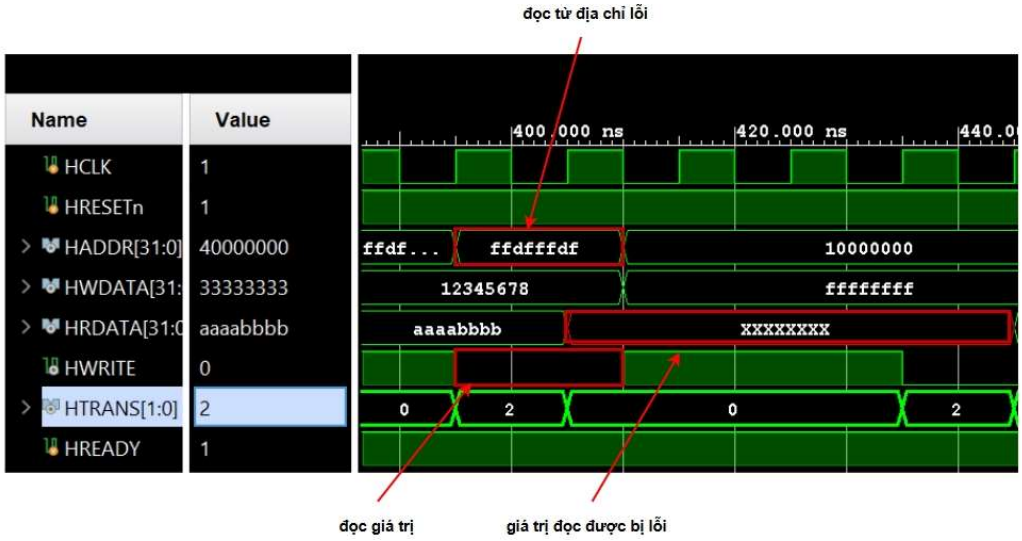
4.6. Testcase 5 (ghi địa chỉ lỗi)



Hình 15. Ghi địa chỉ lỗi

Hình 15 mô tả hoạt động của bus khi ghi vào địa chỉ không xác định (ffdffdff), với giá trị ghi là '12345678'.

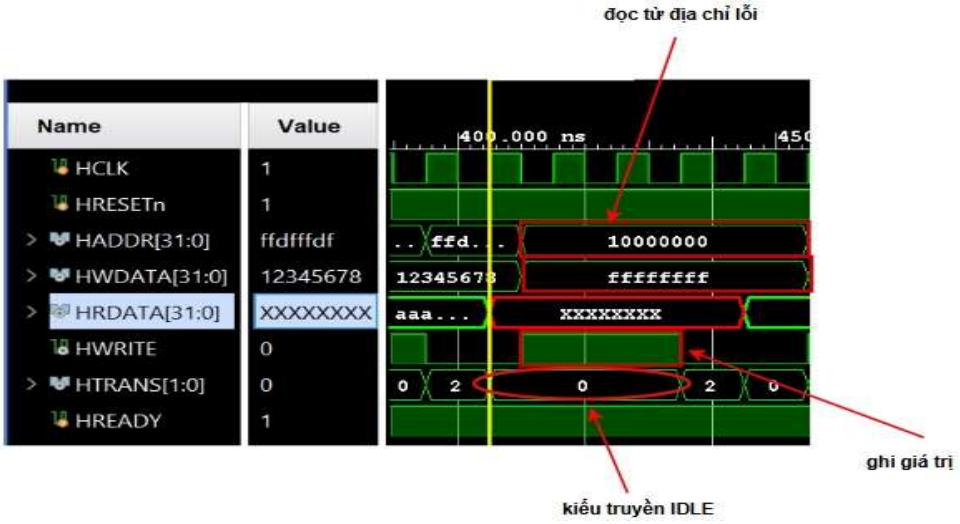
4.7. Testcase 6 (đọc địa chỉ lỗi)



Hình 16. Đọc địa chỉ lỗi

Hình 16 mô tả hoạt động khi đọc giá trị ra từ địa chỉ không xác định, khi đó với giá trị ghi vào là '12345678' thì giá trị được HRDATA đọc ra lại là 'xxxxxxxx' thể hiện giá trị không xác định.

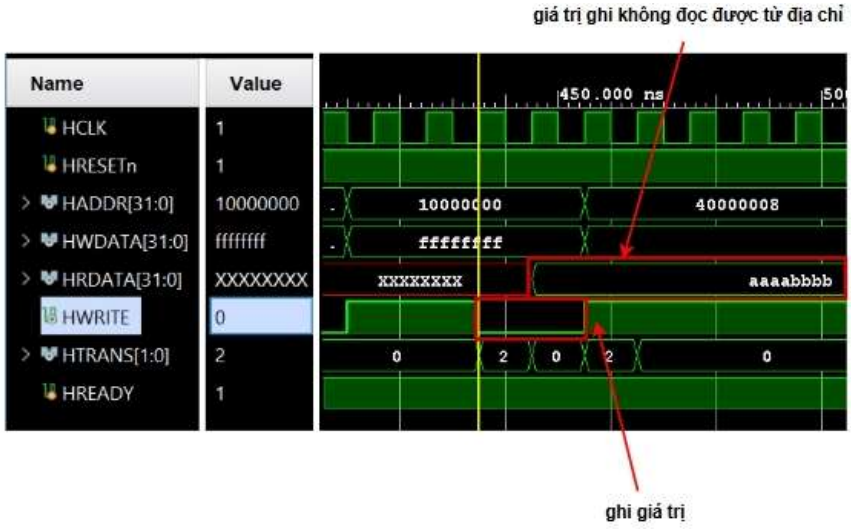
4.8. Testcase 7 (ghi với Htrans IDLE)



Hình 17. Ghi Htrans IDLE

Hình 17 mô tả hoạt động của bus khi ta tiến hành ghi giá trị ‘ffffffff’ vào địa chỉ ‘10000000’ xác định, nhưng với kiểu truyền là IDLE – trạng thái không có giao dịch, bus không được sử dụng.

4.9. Testcase 8 (đọc với Htrans IDLE)

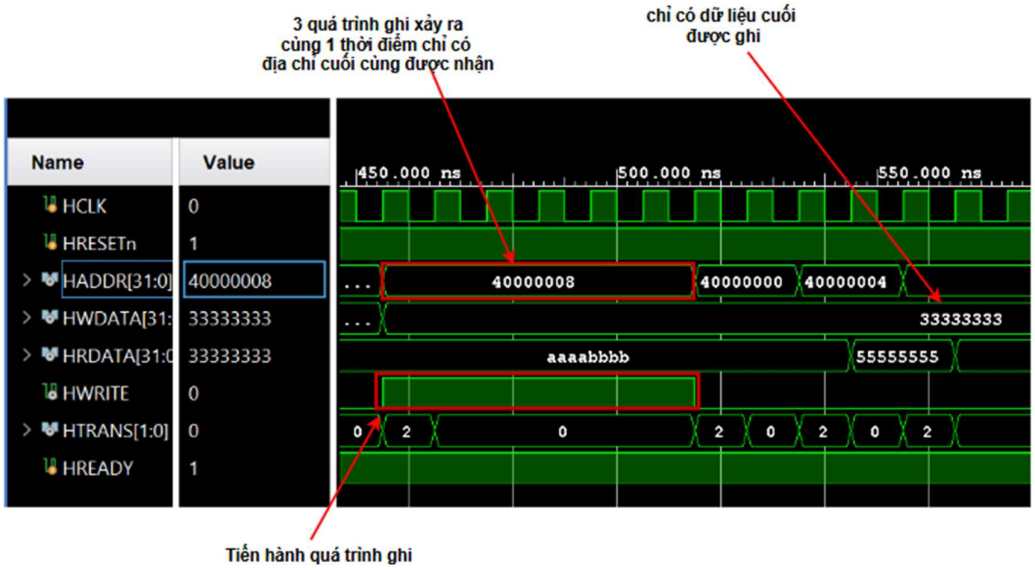


Hình 18.Đọc Htrans IDLE

Hình 18 mô tả hoạt động khi đọc giá trị ra từ địa chỉ ‘10000000’, khi đó với giá trị ghi vào là ‘ffffffff’ thì giá trị được HRDATA đọc ra lại là ‘aaaabbbb’ – giá trị đã được ghi trước đó chứ không phải giá trị đang ghi vào vì khi kiểu truyền là

IDLE sẽ không thể xày ra giao dịch nên HRDATA không nhận được giá trị ghi. Các trình tự hoạt động của các tín hiệu thuộc nhóm địa chỉ và điều khiển cũng đã thực hiện đúng theo tiêu chuẩn kỹ thuật của AMBA AHB Bus.

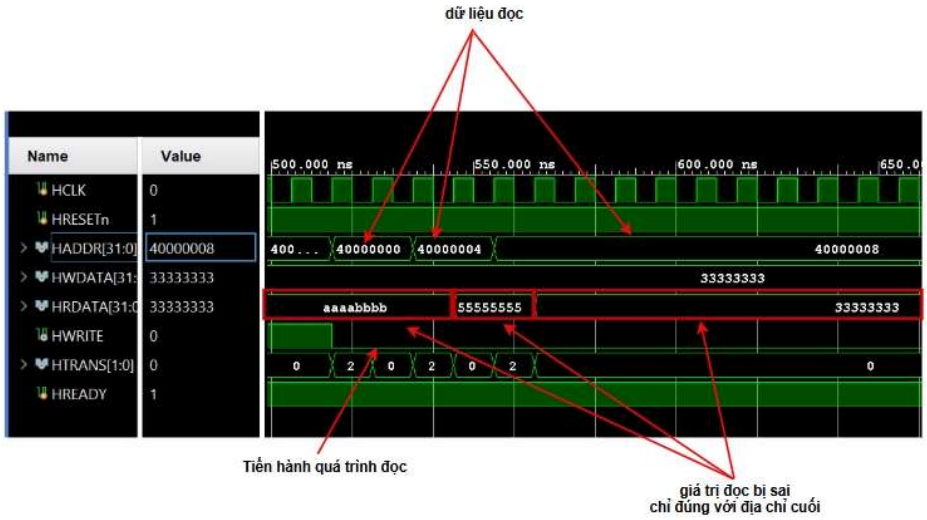
4.10. Testcase 9 (ghi liên tục không chờ)



Hình 19. Ghi liên tục

Hình 19 mô tả hoạt động khi ghi giá trị liên tục không có thời gian chờ giao dịch hoàn thành, khi đó với 3 giá trị ghi vào từ 3 địa chỉ khác nhau, bus chỉ hiện thị được giá trị của dữ liệu cuối cùng.

4.11. Testcase 10 (đọc liên tục không chờ)

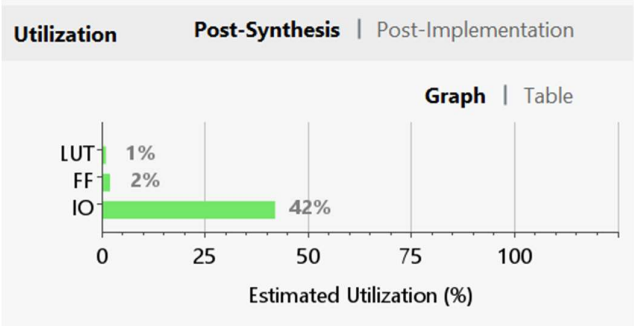


Hình 20. Đọc liên tục

Hình 20 mô tả hoạt động khi đọc giá trị liên tục không có thời gian chờ giao dịch hoàn thành, khi đó với 3 giá trị ghi vào từ 3 địa chỉ khác nhau, bus chỉ đọc được được giá trị của dữ liệu cuối cùng là '33333333' từ địa chỉ '40000008'.

4.12. Tài nguyên sử dụng của hệ thống

Tài nguyên sử dụng của thiết kế sau được đánh giá qua dạng sóng thì thiết kế được tổng hợp trên kit Zynq UltraScale+ của hãng Xillinx trên phần mềm Vivado 2024.2.



Hình 21. Tỉ lệ thiết kế

Resource	Estimation	Available	Utilization...
LUT	675	70560	0.96
FF	2122	141120	1.50
IO	107	252	42.46

Hình 22. Số lượng thiết kế

Kết quả từ hình 21 và 22 tổng hợp trên cho thấy hệ thống bus được thiết kế đã sử dụng 675 số lượng bản tìm kiếm lối vào (Look Up Table – LUT) chỉ chiếm 1% tài nguyên trong kit và phần lớn được sử dụng cho các phép tính logic. Số lượng flip flop đã sử dụng 2122 slice chiếm 2% tài nguyên, số lượng chân được sử dụng là 107 đơn vị, chiếm tới 42% tài nguyên trên chip.

CHƯƠNG 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. KẾT LUẬN

Sau khoảng 15 tuần nghiên cứu tìm hiểu, thiết kế và đánh giá đề tài, về cơ bản nhóm đã thực hiện thiết kế đánh giá thành công đề tài "Thiết kế và đánh giá giao thức AMBA AHB". Và nhóm cũng đã đạt được những mục tiêu ban đầu mà nhóm đã đề ra, qua đó nhóm cũng đã đúc kết được nhiều kinh nghiệm trong quá trình thực hiện đề tài này.

• Ưu điểm:

Sử dụng hệ thống bus này cho cho vấn đề truyền thông trên chip là một giải pháp thích hợp đối với đối với những hệ thống trên chip không quá phức tạp và có số lượng IP vừa phải.

Hệ thống bus này là cơ sở để phát triển mở rộng số lượng giao tiếp trên các hệ thống trên chip, ngoài ra hệ thống còn có thể giao tiếp với nhiều loại IP khác nhau mà không cần phải thay đổi nhiều về cấu trúc.

• Nhược điểm:

Hệ thống bus này đã đơn giản hóa một số tính năng trước khi thiết kế, và chỉ sử dụng một số tính năng cơ bản.

Hệ thống chỉ phù hợp với các hệ thống trên chip đơn giản.

5.2. HƯỚNG PHÁT TRIỂN

Nhóm sẽ tiếp tục phát triển hệ thống bus đã xây dựng với khả năng hỗ trợ giao tiếp đầy đủ các chức năng của burst và phát triển lên giao tiếp giữa nhiều khối chủ với nhiều khối tớ cùng với cầu nối tới các giao thức bus khác.

Ngoài ra nhóm sẽ xem xét việc phát triển hệ thống với khả năng tái cấu hình theo chế độ phân quyền bus khác nhau nhằm nâng cao hiệu suất truyền thông của bus, cho phép cân đối giữa tính năng hoạt động và công suất tiêu thụ của hệ thống.

TÀI LIỆU THAM KHẢO

[1] D. K, "*Design and Verification of AHB Protocol Using System Verilog*," International Journal of Engineering Research and Applications, vol. 11, no. 7, 2021.

[2] P. H. Phong, "*Nghiên cứu, thiết kế và thực hiện Bus truyền thông tốc độ cao AMBA AHB*," Đại học quốc gia Hà Nội, Hà Nội, 2009.

[3] T. X. Tú, "*Thiết kế và mô hình hóa bus truyền thông tốc độ cao dùng cho các hệ thống trên vi mạch*," Đại học quốc gia Hà Nội, Hà Nội, 2010.

[4] A. L. (1999), AMBA Specification (Rev 2.0).

[5] A. L. Copyright © 2001, ARM® AMBA® 5 AHB Protocol, ARM IHI 0033B.b.

[6] P. P. Chu, RTL Hardware Design using VHDL, Hoboken: John Wiley & Sons, Inc., 2006.

[7] T. Đ. Lung, NGÔN NGỮ MÔ TẢ PHẦN CỨNG VERILOG, Đại Học Quốc gia thành phố HCM, 2012.

[8] S. B. D. L, "*Efficient Design and Implementation of AMBA AHB Bus Protocol using Verilog,*" International Conference on Intelligent Sustainable System, no. (ICISS) in IEEE, 2019.

[9] M. C. D. M. P.Harishankar, "*Design and Synthesis of Efficient FSM for Master and Slave Interface in AMBA AHB*," International Journal of Engineering Development and Research, vol. 2, no. 3, 2014.