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 TỐT NGHIỆP

THIẾT KẾ, MÔ PHỎNG VÀ SO SÁNH HIỆU NĂNG GIỮA HỆ THỐNG SỬ DỤNG KIẾN TRÚC BUS WISHBONE VÀ AMBA AXI

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG

Sinh viên: **NGUYỄN QUANG ANH TUẤN**
MSSV: 18161299
NGUYỄN VĂN TUẤN
MSSV: 18161300

TP. HỒ CHÍ MINH – 07/2022

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 TỐT NGHIỆP

THIẾT KẾ, MÔ PHỎNG VÀ SO SÁNH HIỆU NĂNG GIỮA HỆ THỐNG SỬ DỤNG KIẾN TRÚC BUS WISHBONE VÀ AMBA AXI

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG

Sinh viên: **NGUYỄN QUANG ANH TUẤN**
MSSV: 18161299
NGUYỄN VĂN TUẤN
MSSV: 18161300

Hướng dẫn: **ThS. TRƯƠNG QUANG PHÚC**

TP. HỒ CHÍ MINH – 07/2022

TRANG THÔNG TIN LUẬN VĂN

1. Thông tin sinh viên

Họ và tên sinh viên: NGUYỄN QUANG ANH TUẤN MSSV: 18161299

Email: 18161299@student.hcmute.edu.vn Điện thoại: 0944261250

Họ và tên sinh viên: NGUYỄN VĂN TUẤN MSSV: 18161300

Email: 18161300@student.hcmute.edu.vn Điện thoại: 0328584177

2. Thông tin đề tài

- Tên đề tài: THIẾT KẾ, MÔ PHỎNG VÀ SO SÁNH HIỆU NĂNG GIỮA HỆ THỐNG SỬ DỤNG KIẾN TRÚC BUS WISHBONE VÀ AMBA AXI.

- Đơn vị quản lý: Bộ môn Kỹ thuật Máy tính – Viễn thông, Khoa Điện – Điện tử, Trường Đại học Sư phạm Kỹ thuật TP. Hồ Chí Minh.

- Thời gian thực hiện: từ ngày -/-/2022 đến ngày -/-/2022.

- Thời gian bảo vệ: Ngày -/-/2022.

3. Lời cam đoan của sinh viên

Đồ án tốt nghiệp này là công trình nghiên cứu nhóm dưới sự hướng dẫn khoa học của ThS. Trương Quang Phúc. Các số liệu, những nội dung nghiên cứu được trình bày trong luận văn này hoàn toàn trung thực, không sao chép kết quả của công trình khác. Các nội dung tham khảo đã được trích dẫn đầy đủ.

TP.HCM, ngày tháng năm 2022

SV thực hiện đồ án

(Ký và ghi rõ họ tên)

Giảng viên hướng dẫn xác nhận quyển báo cáo đã được chỉnh sửa theo đề nghị ghi trong biên bản của Hội đồng đánh giá Khóa luận tốt nghiệp.

.....

Xác nhận của Bộ môn

Tp. HCM, ngày … tháng … năm 2022

Giáo viên hướng dẫn

(Ký, ghi rõ họ tên và học hàm – học vị)

BẢN NHẬN XÉT KHÓA LUẬN TỐT NGHIỆP

(Dành cho giảng viên hướng dẫn)

Đề tài: THIẾT KẾ, MÔ PHỎNG VÀ SO SÁNH HIỆU NĂNG GIỮA HỆ THỐNG SỬ DỤNG KIẾN TRÚC BUS WISHBONE VÀ AMBA AXI

Sinh viên: 1. Nguyễn Quang Anh Tuấn MSSV: 18161299

2. Nguyễn Văn Tuấn MSSV: 18161300

Hướng dẫn: ThS. Trương Quang Phúc

Nhận xét bao gồm các nội dung sau đây:

1. Tính hợp lý trong cách đặt vấn đề và giải quyết vấn đề; ý nghĩa khoa học và thực tiễn:
Đặt vấn đề rõ ràng, mục tiêu cụ thể; đề tài có tính mới, cấp thiết; đề tài có khả năng ứng dụng, tính sáng tạo.

2. Phương pháp thực hiện/ phân tích/ thiết kế:
Phương pháp hợp lý và tin cậy dựa trên cơ sở lý thuyết; có phân tích và đánh giá phù hợp; có tính mới và tính sáng tạo.

3. Kết quả thực hiện/ phân tích và đánh giá kết quả/ kiểm định thiết kế:
Phù hợp với mục tiêu đề tài; phân tích và đánh giá / kiểm thử thiết kế hợp lý; có tính sáng tạo/ kiểm định chặt chẽ và đảm bảo độ tin cậy.

4. Kết luận và đề xuất:
Kết luận phù hợp với cách đặt vấn đề, đề xuất mang tính cải tiến và thực tiễn; kết luận có đóng góp mới mẻ, đề xuất sáng tạo và thuyết phục.

5. Hình thức trình bày và bố cục báo cáo:
Văn phong nhất quán, bố cục hợp lý, cấu trúc rõ ràng, đúng định dạng mẫu; có tính hấp dẫn, thể hiện năng lực tốt, văn bản trau chuốt.

6. Kỹ năng chuyên nghiệp và tính sáng tạo:
Thể hiện các kỹ năng giao tiếp, kỹ năng làm việc nhóm, và các kỹ năng chuyên nghiệp khác trong việc thực hiện đề tài.

7. Tài liệu trích dẫn
Tính trung thực trong việc trích dẫn tài liệu tham khảo; tính phù hợp của các tài liệu trích dẫn; trích dẫn theo đúng chỉ dẫn IEEE.

8. Đánh giá về sự trùng lặp của đề tài
Cần khẳng định đề tài có trùng lặp hay không? Nếu có, đề nghị ghi rõ mức độ, tên đề tài, nơi công bố, năm công bố của đề tài đã công bố.

9. Những nhược điểm và thiếu sót, những điểm cần được bổ sung và chỉnh sửa*

10. Nhận xét tinh thần, thái độ học tập, nghiên cứu của sinh viên

Đề nghị của giảng viên hướng dẫn
Ghi rõ: "Báo cáo đạt/ không đạt yêu cầu của một khóa luận tốt nghiệp kỹ sư, và được phép/ không được phép bảo vệ khóa luận tốt nghiệp"

Tp. HCM, ngày ... tháng .... năm 2022
Người nhận xét
(*Ký và ghi rõ họ tên)*

LỜI CẢM ƠN

Đầu tiên, nhóm sinh viên thực hiện xin chân thành cảm ơn ThS. Trương Quang Phúc, giảng viên Bộ môn Kỹ thuật Máy tính – Viễn thông, Khoa Điện - Điện tử, Trường Đại học Sư Phạm Kỹ Thuật Thành phố Hồ Chí Minh, đã trực tiếp hướng dẫn, tận tình giúp đỡ và tạo điều kiện tốt nhất cho nhóm để có thể hoàn thành được đề tài này.

Nhóm sinh viên thực hiện đề tài cũng xin chân thành cảm ơn các thầy cô của Khoa Điện – Điện tử nói chung và Bộ môn Kỹ thuật Máy tính – Viễn thông nói riêng đã trang bị cho sinh viên những kiến thức nền tảng để nhóm có thể vận dụng và tiếp cận đến những kiến thức mới một các dễ dàng hơn. Nhóm sinh viên thực hiện xin cảm ơn nhà trường đã cung cấp những tài liệu bổ ích từ thư viện số để nhóm có thể hoàn thành tốt đồ án lần này.

Nhóm cũng gửi lời cảm ơn đến các bạn học đã chia sẻ và trao đổi kiến thức trong thời gian học tập ở trường và trong quá trình thực hiện đề tài.

Vì thời gian và lượng kiến thức có hạn nên việc thực hiện đồ án không thể tránh khỏi việc có một số sai sót, mong quý thầy cô thông cảm.

Xin chân thành cảm ơn !

Trân trọng

Nhóm thực hiện đề tài

Nguyễn Quang Anh Tuấn

Nguyễn Văn Tuấn

TÓM TẮT

Trong xu hướng hiện tại của cuộc Cách mạng Công nghiệp 4.0, thiết kế Hệ thống trên chip (SoC) đóng vai trò quan trọng trong các hệ thống nhúng và là một lĩnh vực mũi nhọn của nhiều nước công nghiệp trên thế giới. Nhờ sự phát triển nhanh chóng của công nghệ bán dẫn và các kỹ thuật thiết kế mạch tích hợp, chúng ta đã có thể tích hợp ngày càng nhiều thành phần hay cả một hệ thống hoàn chỉnh lên trên một con chip. Do đó, nhiệm vụ giao tiếp và kết nối các thành phần với nhau trở nên phức tạp hơn. Một trong những giải pháp hiệu quả là phát triển và sử dụng các giao thức bus làm giao diện kết nối tiêu chuẩn để kết nối các thành phần thiết kế với nhau trong SoC.

Trong đề tài này, nhóm tác giả thực hiện so sánh hai loại kiến trúc bus là bus WISHBONE và AMBA AXI nhằm có một cái nhìn tổng quát về cấu trúc và hoạt động của các bus hệ thống dùng trong bên trong một SoC. Dựa trên lý thuyết của kiến trúc các giao thức bus WISHBONE và AMBA AXI từ đó xây dựng các khối thành phần và tổng hợp thiết kế hệ thống hoàn chỉnh. Bên cạnh đó, nhóm thực hiện kiểm tra và đánh giá hoạt động của từng loại kiến trúc bus thông qua mô phỏng dạng sóng dựa trên mô hình kết nối điểm – điểm giữa một MASTER và một SLAVE. Sau đó, nhóm tiến hành so sánh hiệu năng dựa trên thông số tài nguyên, công suất tiêu thụ của mỗi hệ thống khi tổng hợp trên nền tảng ZYNQ-7000 SoC Zybo Board. Quá trình mô phỏng và tổng hợp sử dụng mô phỏng trên phần mềm Xilinx Vivado 2019.1.

Kết quả sau khi mô phỏng và thực nghiệm cho thấy ưu điểm của hệ thống sử dụng bus WISHBONE là giao diện kết nối đơn giản, yêu cầu ít tín hiệu để kết nối. Hệ thống sử dụng bus AMBA AXI yêu cầu nhiều tín hiệu để kết nối hơn và quá trình xác nhận thông tin gồm nhiều cặp tín hiệu xác nhận giúp đảm bảo cho dữ liệu truyền hợp lệ. Do đó, tài nguyên sử dụng và công suất tiêu thụ của hệ thống bus WISHBONE ít hơn so với của AMBA AXI. Thời gian để một dữ liệu được truyền nhận của bus WISHBONE nhanh hơn so với AMBA AXI. Tuy nhiên, do ưu điểm về các kênh truyền độc lập của bus AMBA AXI nên khi có nhiều hơn một dữ liệu được truyền nhận thì AMBA AXI sẽ cho thời gian và hiệu suất truyền tốt hơn bus WISHBONE.

MỤC LỤC

DANH MỤC HÌNH IX
DANH MỤC BẢNG XI
DANH MỤC CÁC TỪ VIẾT TẮT XII

CHƯƠNG 1 TỔNG QUAN **1**

1.1 GIỚI THIỆU 1
1.2 MỤC TIÊU ĐỀ TÀI 2
1.3 TÌNH HÌNH NGHIÊN CỨU 2
1.4 PHƯƠNG PHÁP NGHIÊN CỨU 3
1.5 BỐ CỤC TRÌNH BÀY CỦA ĐỀ TÀI 4

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT **5**

2.1 TỔNG QUAN VỀ SOC 5
2.1.1 Định nghĩa *5*
2.1.2 Kiến trúc của SoC *5*
2.1.3 Ưu điểm và nhược điểm của SoC *7*
2.1.4 Lõi IP *8*
2.1.4.1 Lõi IP mềm 8
2.1.4.2 Lõi IP bán cứng 8
2.1.4.3 Lõi IP cứng 8
2.2 TỔNG QUAN VỀ HỆ THỐNG BUS TRONG SOC 9
2.2.1 Hệ thống bus trong SoC *9*
2.2.2 Đặc điểm của hệ thống bus trong SoC *10*
2.3 BUS WISHBONE 11
2.3.1 Giới thiệu *11*
2.3.2 Đặc điểm bus WISHBONE *12*
2.3.3 Kết nối WISHBONE *13*
2.3.3.1 Kết nối Điểm - điểm (Point-to-Point) 14
2.3.3.2 Kết nối Luồng dữ liệu (Data Flow) 14
2.3.3.3 Kết nối Bus chia sẻ (Shared Bus) 15
2.3.3.4 Kết nối Chuyển mạch chéo (Crossbar switch) 16
2.3.4 Tín hiệu WISHBONE *16*
2.3.4.1 Tín hiệu của mô-đun SYSCON 17
2.3.4.2 Tín hiệu chung của giao diện WISHBONE MASTER và SLAVE 18
2.3.4.3 Tín hiệu của giao diện WISHBONE MASTER 18
2.3.4.4 Tín hiệu của giao diện WISHBONE SLAVE 19
2.3.5 Chu kỳ ĐỌC/GHI ĐƠN *19*
2.3.5.1 Chu kỳ ĐỌC ĐƠN 19
2.3.5.2 Chu kỳ GHI ĐƠN 20
2.3.6 Chu kỳ ĐỌC/GHI KHỐI *21*
2.3.6.1 Chu kỳ ĐỌC KHỐI 21

2.3.6.2 Chu kỳ GHI KHỐI 21
2.4 BUS AMBA AXI 22
2.4.1 Giới thiệu *22*
2.4.2 Cấu trúc bus AMBA AXI *22*
2.4.3 Các quy định về hoạt động *25*
2.4.4 AMBA AXI4 ..... *27*
2.4.4.1 Giới thiệu 27
2.4.4.2 Cơ chế bắt tay của AXI4 27
2.4.4.3 Cơ chế burst của AXI4 29
2.5 NỀN TẢNG ZYNQ-7000 SoC 29
2.5.1 Thành phần PS của ZYNQ-7000 ..... *31*
2.5.2 Thành phần PL của ZYNQ-7000 ..... *32*
2.6 PHẦN CỨNG ZYBO ZYNQ-7000 32
2.7 PHẦN MỀM XILINX VIVADO DESIGN SUITE 2019.1 33
CHƯƠNG 3 THIẾT KẾ HỆ THỐNG 34
3.1 YÊU CẦU HỆ THỐNG 34
3.2 ĐẶC TẢ HỆ THỐNG 34
3.3 THIẾT KẾ CHI TIẾT HỆ THỐNG SỬ DỤNG BUS WISHBONE 35
3.3.1 Thiết kế khối WISHBONE MASTER *37*
3.3.2 Thiết kế khối WISHBONE SLAVE *41*
3.4 THIẾT KẾ CHI TIẾT HỆ THỐNG SỬ DỤNG BUS AMBA AXI 43
3.4.1 Thiết kế khối AXI MASTER *46*
3.4.2 Thiết kế khối AXI SLAVE *47*
CHƯƠNG 4 MÔ PHỎNG VÀ KẾT QUẢ 50
4.1 SƠ ĐỒ HỆ THỐNG TRÊN PHẦN MỀM VIVADO 2019.1 50
4.2 THÔNG SỐ TÀI NGUYÊN HỆ THỐNG 52
4.3 KẾT QUẢ MÔ PHỎNG SỬ DỤNG PHẦN MỀM XILINX VIVADO 2019.1 53
4.3.1 Kết quả mô phỏng của hệ thống sử dụng kiến trúc bus AMBA AXI *53*
4.3.1.1 Hoạt động ghi 54
4.3.1.2 Hoạt động đọc 55
4.3.2 Kết quả mô phỏng của hệ thống sử dụng kiến trúc bus WISHBONE *56*
4.3.2.1 Hoạt động ghi 57
4.3.2.2 Hoạt động đọc 57
CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 59
5.1 KẾT LUẬN 59
5.2 HƯỚNG PHÁT TRIỂN 59
TÀI LIỆU THAM KHẢO 60

DANH MỤC HÌNH

Hình 2.1: Kiến trúc SoC 6
Hình 2.2: Kiến trúc SoC sử dụng nhiều lõi IP 6
Hình 2.3: Hệ thống bus trong SoC 9
Hình 2.4: Giao diện kết nối INTERCON 13
Hình 2.5: Kết nối Điểm - điểm 14
Hình 2.6: Kết nối Luồng dữ liệu 14
Hình 2.7: Kết nối Bus chia sẻ 15
Hình 2.8: Kết nối Chuyển mạch chéo 16
Hình 2.9: Tín hiệu kết nối hai lõi IP MASTER/SLAVE 17
Hình 2.10: Chu kỳ ĐỌC ĐƠN 20
Hình 2.11: Chu kỳ GHI ĐƠN 20
Hình 2.12: Chu kỳ ĐỌC KHỐI 21
Hình 2.13: Chu kỳ GHI KHỐI 22
Hình 2.14: Các kênh giao tiếp giữa AXI MASTER và SLAVE 23
Hình 2.15: Hoạt động của một quá trình đọc 26
Hình 2.16: Hoạt động của một quá trình ghi 26
Hình 2.17: Trường hợp tín hiệu VALID tích cực trước READY 28
Hình 2.18: Trường hợp tín hiệu VALID tích cực sau READY 28
Hình 2.19: Trường hợp tín hiệu VALID tích cực cùng lúc với READY 28
Hình 2.20: Cơ chế burst của AXI 29
Hình 2.21: Kiến trúc ZYNQ-7000 SoC 30
Hình 2.22: Phần cứng ZYBO ZYNQ-7000 32
Hình 3.1: Sơ đồ đặc tả hệ thống 34
Hình 3.2: Sơ đồ khối tổng quát hệ thống kết nối sử dụng bus WISHBONE 35
Hình 3.3: Sơ đồ khối SYSCON và dạng sóng tín hiệu xung CLK và RST 35
Hình 3.4: Sơ đồ phân cấp cho mô phỏng hệ thống 36
Hình 3.5: Sơ đồ khối WISHBONE MASTER 37
Hình 3.6: Sơ đồ khối chi tiết của WISHBONE MASTER 38
Hình 3.7: Sơ đồ máy trạng thái của hệ thống bus WISHBONE 40
Hình 3.8: Sơ đồ khối WISHBONE SLAVE 41
Hình 3.9: Sơ đồ khối chi tiết của WISHBONE SLAVE 42
Hình 3.10: Sơ đồ tổng quát hệ thống kết nối sử dụng bus AMBA AXI 43

Hình 3.11: Sơ đồ khối chi tiết của AXI MASTER.....46
Hình 3.12: Sơ đồ khối AXI SLAVE.....47
Hình 3.13: Sơ đồ máy trạng thái quá trình ghi của AXI RAM 48
Hình 3.14: Sơ đồ máy trạng thái máy quá trình đọc của AXI RAM.....49
Hình 4.1: Sơ đồ hệ thống sử dụng bus WISHBONE trên phần mềm Vivado 2019.1.....50
Hình 4.2: Sơ đồ hệ thống sử dụng bus AMBA AXI trên phần mềm Vivado 2019.1.....51
Hình 4.3: Kết quả mô phỏng hoạt động của bus AMBA AXI 53
Hình 4.4: Dạng sóng mô phỏng hoạt động ghi của bus AMBA AXI.....54
Hình 4.5: Dạng sóng mô phỏng hoạt động đọc của bus AMBA AXI.....55
Hình 4.6: Kết quả mô phỏng hoạt động của bus WISHBONE 56
Hình 4.7: Dạng sóng mô phỏng hoạt động ghi của bus WISHBONE.....57
Hình 4.8: Dạng sóng mô phỏng hoạt động đọc của bus WISHBONE.....57

DANH MỤC BẢNG

Bảng 2.1: Các tín hiệu trong kênh Địa Chỉ Đọc của bus AMBA AXI 23
Bảng 2.2: Các tín hiệu trong kênh Dữ Liệu Đọc của bus AMBA AXI 24
Bảng 2.3: Các tín hiệu trong kênh Địa Chỉ Ghi của bus AMBA AXI..... 24
Bảng 2.4: Các tín hiệu trong kênh Dữ Liệu Ghi của bus AMBA AXI..... 25
Bảng 2.5: Các tín hiệu trong kênh Phản Hồi Ghi của bus AMBA AXI 25
Bảng 2.6: Các cặp tín hiệu bắt tay trong năm kênh truyền của bus AMBA AXI..... 28
Bảng 3.1: Các tín hiệu sử dụng trong giao diện MASTER của bus WISHBONE 36
Bảng 3.2: Các tín hiệu sử dụng trong giao diện SLAVE của bus WISHBONE..... 37
Bảng 3.3: Các tín hiệu sử dụng trong giao diện MASTER của bus AMBA AXI 44
Bảng 3.4: Các tín hiệu sử dụng trong giao diện SLAVE của bus AMBA AXI..... 45
Bảng 4.1: Thông số tài nguyên hệ thống sử dụng bus WISHBONE và AMBA AXI 52
Bảng 4.2: Công suất tiêu thụ của hệ thống sử dụng bus WISHBONE 52
Bảng 4.3: Công suất tiêu thụ của hệ thống sử dụng bus AMBA AXI 52

DANH MỤC CÁC TỪ VIẾT TẮT

AHB	Advanced High-Performance Bus
AMBA	Advanced Microcontroller Bus Architecture
ASIC	Application-Specific Integrated Circuit
AXI	Advanced Extensible Interface
BRAM	Block Random Access Memory
CLB	Configurable Logic Blocks
CPU	Central Processing Unit
DMA	Direct Memory Access
DSP	Digital Signal Processor
FF	Flip-Flop
FPGA	Field-Programmable Gate Array
HDL	Hardware Description Language
I/O	Input/Output
IC	Integrated Circuit
IP	Intellectual Property
LUT	Look-up table
PL	Programmable Logic
PS	Processing System
RAM	Random Access Memory
ROM	Read-Only Memory
RTL	Register Transfer Level
SoC	System-On-Chip
VHDL	Very High-Speed Integrated Circuit Hardware Description Language

CHƯƠNG 1

TỔNG QUAN

1.1 GIỚI THIỆU

Thiết kế Hệ thống trên chip (SoC) là một xu hướng chung và đóng một vai trò quan trọng trong nền công nghiệp 4.0 của nhiều nước công nghiệp trên thế giới. Nhờ sự phát triển nhanh chóng của công nghệ bán dẫn và các kỹ thuật thiết kế mạch tích hợp, việc tích hợp số lượng lớn các thành phần hay cả một hệ thống hoàn chỉnh lên trên một con chip đã trở nên dễ dàng hơn. SoC có thể thực hiện hầu hết các chức năng của một hệ thống điện tử hoàn chỉnh trong một kích thước nhỏ gọn hơn, hiệu suất và tính ổn định cũng cao hơn. Chính nhờ các ưu điểm đó của SoC mà các vi điều khiển của thiết bị ít tốn không gian hơn, sử dụng ít năng lượng hơn, phù hợp với các thiết bị có thiết kế tinh gọn như điện thoại thông minh [1].

Bên cạnh đó, với nhu cầu giảm độ phức tạp của giao diện, tiêu hao năng lượng thấp, chi phí sản xuất giảm nên các nhà sản xuất ngày càng chú trọng đến việc áp dụng tái sử dụng thiết kế bằng cách tích hợp các lõi sở hữu trí tuệ (Intellectual Property - IP) lên một con chip [2]. Các lõi IP có thể được phát triển bởi các nhà sản xuất khác nhau với các chức năng riêng biệt và có thể sẽ có sự khác nhau về cách kết nối tùy vào nhà sản xuất. Do đó, nhiệm vụ giao tiếp và kết nối các thành phần với nhau trở nên phức tạp hơn. Một trong những giải pháp khả thi nằm ở việc sử dụng bus kết nối nội bộ tiêu chuẩn để kết nối các thành phần thiết kế với nhau trong các mô-đun ứng dụng SoC sao cho thuận tiện cho việc sử dụng hiện tại và tái sử dụng trong tương lai [3].

Bus là hệ thống có vai trò chính trong việc giao tiếp, trung chuyển dữ liệu giữa các thành phần trong một SoC và ảnh hưởng trực tiếp đến hiệu suất của hệ thống. Một số các kiến trúc bus hệ thống phổ biến hiện nay như: AMBA (Advanced Microcontroller Bus Architecture) AHB (Advanced High-Performance Bus) và AXI (Advanced Extensible Interface) bus phát triển bởi ARM, CoreConnect của IBM, Open Core Protocol (OCP) của OCP International Partnership, WISHBONE của OpenCores.

Trong đề tài này, nhóm tác giả lựa chọn so sánh giữa hai loại bus là WISHBONE và AMBA AXI để có thể hiểu về hoạt động của các kiến trúc bus và sự ảnh hưởng của chúng đến hiệu năng của một hệ thống trên chip.

1.2 MỤC TIÊU ĐỀ TÀI

Mục tiêu của đề tài là tìm hiểu, thiết kế, so sánh và đánh giá về hiệu năng của hai loại bus là WISHBONE và AMBA AXI thông qua mô hình kết nối điểm - điểm giữa một MASTER và một SLAVE. Phần kiểm tra hoạt động của từng hệ thống được thực hiện thông qua mô phỏng các hoạt động truyền nhận dữ liệu giữa hai giao diện. Phần so sánh và đánh giá hiệu năng được dựa trên các thông số tài nguyên, công suất tiêu thụ của mỗi hệ thống khi tổng hợp trên nền tảng ZYNQ-7000 SoC Zybo Board. Quá trình mô phỏng và tổng hợp được thực hiện trên phần mềm mô phỏng Xilinx Vivado 2019.1.

1.3 TÌNH HÌNH NGHIÊN CỨU

Trong bài viết [1] đã mô tả tóm tắt về các giao thức bus SoC phổ biến chẳng hạn như AMBA AHB AXI, WISHBONE, OCP (Open Core Protocol), CoreConnect và giới thiệu ngắn gọn về giao thức bus SoC hiệu suất cao được gọi là MSBUS (Master-Slave Bus). Bài viết chủ yếu so sánh những đặc điểm khác nhau của mỗi loại bus và cho thấy rằng MSBUS là giao thức hiệu quả trong việc truyền các khối dữ liệu bằng cách triển khai DMA (Direct Memory Access) dựa trên MSBUS tại RTL (Register Transfer Level) trong HDL (Hardware Description Language). Sau đó so sánh tương tự với các giao thức khác bằng cách xem xét các thông số khác nhau như thời gian truyền, hiệu suất truyền, băng thông dữ liệu hợp lệ, hiệu suất năng lượng động và điện năng tiêu thụ.

Nội dung bài viết [2] và [3] trình bày một cuộc khảo sát về các kiến trúc bus phổ biến như OpenCores WISHBONE, ARM AMBA, IBM CoreConnect và Altera Avalon. Hai bài viết này đã so sánh tổng quan được các đặc điểm kỹ thuật của các loại bus với nhau và đưa ra được ưu điểm của từng loại Bus, nhằm đánh giá và lựa chọn loại bus SoC cho việc phát triển các lõi với ứng dụng khác nhau.

Trong bài viết [4], nhóm tác giả mô tả một nghiên cứu về giao thức bus SoC AMBA của ARM, chỉ ra những điểm khác nhau và so sánh hiệu suất của chúng. Nội dung bao gồm phần giới thiệu ngắn gọn về giao thức AMBA 2.0, giao thức AMBA 3.0 và giao thức AMBA 4.0. Sau đó là kết luận liên quan đến phân tích hiệu suất so sánh của cả ba giao thức bus AMBA.

Trong bài viết [5], tác giả đã trình bày các đặc điểm của giao diện bus WISHBONE và thực hiện mô phỏng sử dụng Xilinx ISE và ChipScope Pro để so sánh hai hệ thống kết nối kiểu điểm - điểm và bus chia sẻ (Shared Bus) được quan sát trên FPGA (Field-Programmable Gate Array) Virtex-II Pro và Spartan3e. Cả hai hệ thống đều sử dụng lõi DMA làm giao diện MASTER và bộ nhớ làm giao diện SLAVE.

Trong bài viết [6], tác giả đã giới thiệu tổng quan về các đặc điểm, kiến trúc và hoạt động của bus AMBA AXI, mô tả thiết kế và triển khai mô hình kết nối AXI sử dụng ngôn ngữ mô tả phần cứng Verilog trên phần cứng Xilinx Spartan 3E FPGA và mô phỏng thực hiện với phần mềm Modelsim RTL Simulation. Bên cạnh đó, trong bài viết [7] tác giả chủ yếu tập trung phân tích các tính năng quan trọng và các kết nối trong giao diện AXI. Bài viết [8] tính toán độ trễ trong các hoạt động đọc và ghi bằng cách phát triển mã hóa RTL Verilog và tổng hợp nó với công cụ của Synopsys ở tiến trình 90nm.

Trong bài viết [9] và [10], các tác giả đã trình bày và phân tích về bus WISHBONE, mô phỏng kiến trúc liên kết điểm - điểm của bus WISHBONE, sử dụng các lõi DMA, MEMORY, SYSCON thực hiện mô phỏng và quan sát kết quả dạng sóng tín hiệu khi truyền dữ liệu từ MASTER sang SLAVE.

1.4 PHƯƠNG PHÁP NGHIÊN CỨU

Đối với đề tài này, nhóm tác giả tập trung chủ yếu vào phương pháp tổng hợp và phân tích lý thuyết để tìm ra những vấn đề cần giải quyết, từ đó tìm kiếm tài liệu liên quan và lựa chọn hướng thiết kế phù hợp để giải quyết các vấn đề đặt ra. Đầu tiên, nhóm dựa trên những thiết kế sẵn có và các tài liệu kỹ thuật liên quan để có kiến trúc phù hợp cho thiết kế. Tiếp theo, nhóm xây dựng sơ đồ khối và tổng hợp các thiết kế để có một hệ thống hoàn chỉnh. Cuối cùng, dùng phương pháp thực nghiệm bằng cách thay đổi các tác nhân, tạo ra các trường hợp để kiểm tra hoạt động chung của toàn bộ thiết kế thông qua mô phỏng và nhận xét dựa trên kết quả đạt được.

1.5 BỐ CỤC TRÌNH BÀY CỦA ĐỀ TÀI

Chương 1: Giới thiệu: Trình bày tổng quan mục tiêu của đề tài, giới thiệu về hai loại bus WISHBONE và AMBA AXI.

Chương 2: Cơ sở lý thuyết: Nội dung chính bao gồm tổng quan về bus của SoC, mô tả về các khái niệm, đặc điểm, kiến trúc và kết nối của hai loại bus WISHBONE và AMBA AXI.

Chương 3: Thiết kế hệ thống: Trình bày về sơ đồ khối của hệ thống, phân tích và tiến hành mô phỏng hệ thống dựa vào các đặc tính, yêu cầu, cách kết nối và hoạt động của hệ thống.

Chương 4: Kết quả và mô phỏng: Thực hiện kiểm tra chức năng, thống kê tài nguyên sử dụng và trình bày kết quả đạt được sau quá trình mô phỏng.

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

CHƯƠNG 2

CƠ SỞ LÝ THUYẾT

2.1 TỔNG QUAN VỀ SOC

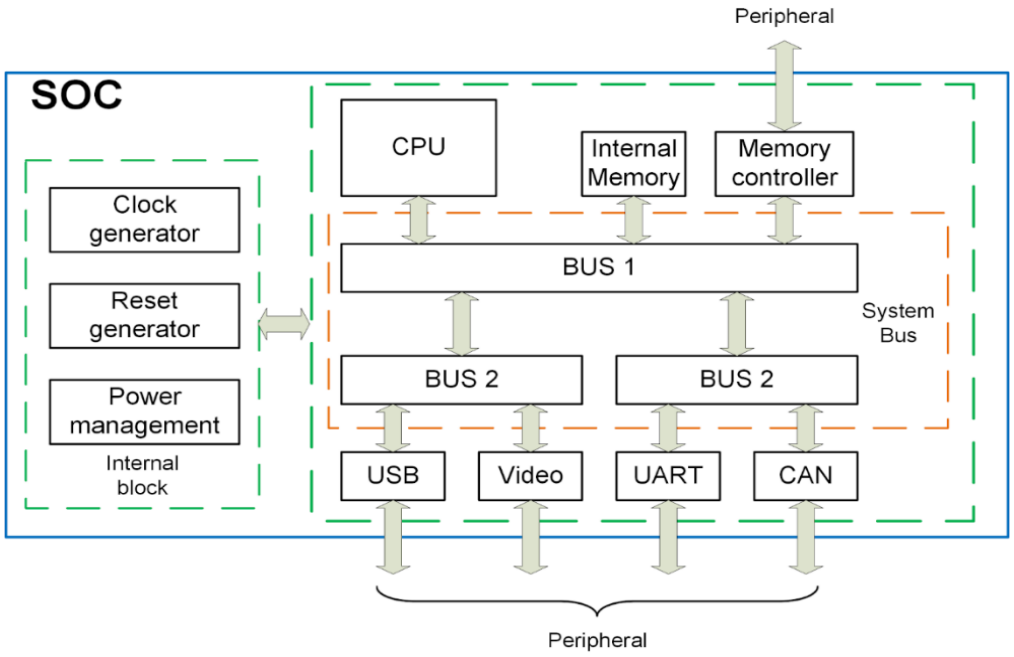
2.1.1 Định nghĩa

Hệ thống trên chip (System-on-Chip hay viết tắt là SoC) là toàn bộ một hệ thống được đóng gói hoàn chỉnh trong một con chip hay một vi mạch điện tử. SoC rất phổ biến trong ngày công nghiệp hiện đại ngày nay, là một xu hướng chung của công nghệ. SoC có thể được tích hợp các thành phần của một máy tính hoặc các hệ thống điện tử khác trên một kích thước nhỏ gọn, tiêu tốn ít năng lượng hơn mà vẫn có thể thực hiện đầy đủ chức năng của hệ thống. Một SoC có thể bao gồm các khối chức năng kỹ thuật số, tương tự, tín hiệu kết hợp và cả các khối tần số vô tuyến. Sự phát triển của công nghệ bán dẫn cho phép con người ngày càng có thể tích hợp nhiều hệ thống trên cùng một con chip với ứng dụng đa dạng hơn và hiệu năng tốt hơn [11].

2.1.2 Kiến trúc của SoC

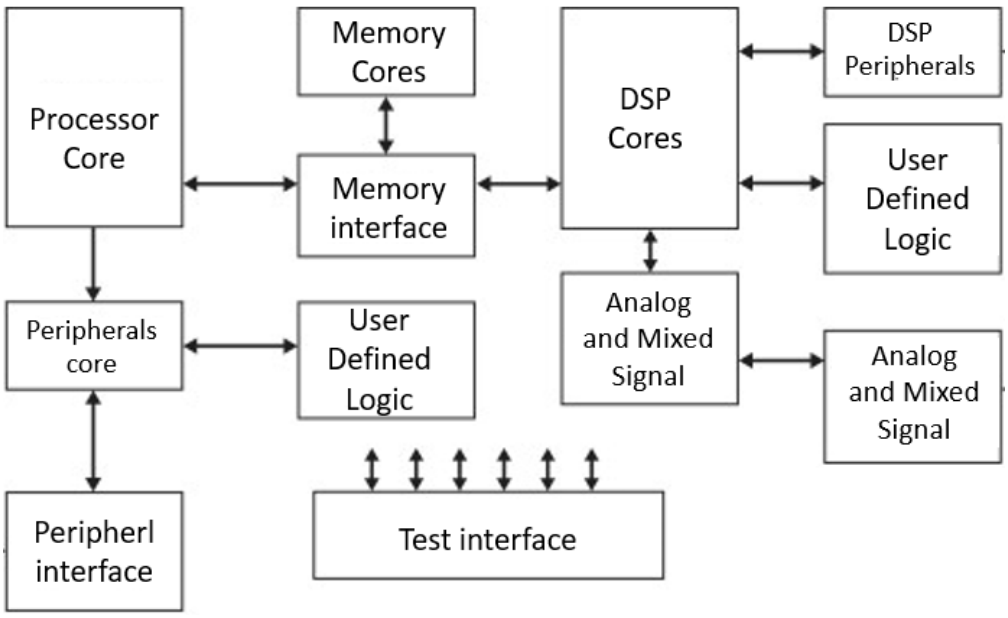
Trong một kiến trúc SoC, mỗi bộ xử lý nhúng phải thực hiện ít nhất hai nhiệm vụ. Đầu tiên, chức năng của hệ thống dưới dạng các quy trình mã nhúng sẽ được thực thi bởi các bộ xử lý và kiểm soát. Sau đó là tiến hành đồng bộ hóa việc trao đổi dữ liệu giữa các IP khác nhau của SoC. Nhiệm vụ đầu tiên mang lại tính linh hoạt cao cho các nhà thiết kế SoC, họ có thể sử dụng khả năng lập trình vốn có của bộ xử lý để cập nhật, cải thiện và sửa đổi hiệu quả chức năng của hệ thống chỉ bằng cách thêm hoặc sửa đổi phần mềm nhúng hiện có được lưu trữ trong các lõi bộ nhớ nhúng. Nhiệm vụ thứ hai cung cấp khả năng truy cập và giao tiếp hiệu quả từ bộ xử lý đến tất cả các lõi bên trong của SoC [11].

Hiện tại, cách sắp xếp phổ biến nhất là có hai bộ xử lý nhúng trong một SoC. Một bộ vi xử lý nhúng hoặc bộ xử lý tín hiệu kỹ thuật số (DSP) có thể được sử dụng cho chức năng xử lý chính, trong khi DSP có thể đảm nhận việc xử lý dữ liệu nặng hơn cho các thuật toán xử lý tín hiệu chuyên biệt. Trong các kiến trúc mà SoC giao tiếp với một số kênh dữ liệu bên ngoài, một bộ xử lý nhúng riêng biệt được liên kết với hệ thống con bộ nhớ chuyên dụng của nó có thể xử lý từng kênh giao tiếp, trong khi một kênh khác có thể được sử dụng để điều phối luồng dữ liệu.



Hình 2.1: Kiến trúc SoC

Một kiến trúc SoC điển hình chứa một số loại lõi được thể hiện trong hình 2.1 bên trên. Kiến trúc SoC bao gồm một số lõi xử lý nhúng, trong đó một lõi đóng vai trò trung tâm và các lõi khác có các tác vụ chuyên biệt. Một số lõi bộ nhớ (RAM động hoặc tĩnh, ROM) cũng có thể được sử dụng, mỗi lõi dành riêng cho một nhiệm vụ khác nhau. Các lõi khác được sử dụng để giao tiếp giữa SoC và ngõ ra của nó và các lõi khác được sử dụng để giao tiếp với các kết nối ngoại vi, chuyển đổi tương tự sang kỹ thuật số và ngược lại. Một kiến trúc SoC được tích hợp nhiều lõi IP được mô tả như trong hình 2.2 bên dưới.



Hình 2.2: Kiến trúc SoC sử dụng nhiều lõi IP

Việc kiểm tra cả bộ vi xử lý và bộ xử lý nhúng trong một SoC luôn là một thách thức vì hầu hết các kỹ thuật kiểm tra truyền thống đều không thể đáp ứng được. Điều này là do cấu trúc tuần tự phức tạp của kiến trúc bộ xử lý, bao gồm các đơn vị đường dẫn dữ liệu hiệu suất cao và logic điều khiển tinh vi để tối ưu hóa hiệu suất [11].

2.1.3 Ưu điểm và nhược điểm của SoC

Ưu điểm của SoC đó là có thể sử dụng cho các thiết bị điện tử cần kích thước nhỏ như điện thoại, máy tính bảng. SoC có mật độ tích hợp cao, nhiều chức năng nên việc làm phần cứng sản phẩm đơn giản hơn, dễ sản xuất hơn, chi phí thấp hơn, thời gian thiết kế sản phẩm nhanh hơn. Sản xuất số lượng đủ lớn, giá thành sẽ giảm đáng kể. Sản phẩm sử dụng SoC tiết kiệm năng lượng hơn so với sản phẩm cùng chức năng nhưng không dùng SoC vì số lượng linh kiện lớn hơn, bo mạch làm phức tạp hơn. Khi sản phẩm càng nhiều chức năng thì đặc điểm này càng thấy rõ [11].

Các khối được thiết kế trước thường được gọi là lõi hoặc sở hữu trí tuệ IP được lấy từ các nguồn nội bộ hoặc bên thứ ba và được kết hợp thành một chip. Các lõi này có thể bao gồm bộ xử lý nhúng, khối bộ nhớ hoặc mạch xử lý các chức năng xử lý cụ thể. Những người thiết kế SoC có thể kết hợp chúng vào một con chip để thực hiện các chức năng phức tạp. Bằng cách sử dụng kỹ thuật này, nhà thiết kế không cần phải tìm hiểu chi tiết từng lõi mà có thể tập trung vào các vấn đề cấp hệ thống cao hơn. Trong khi đó, các lõi sẽ được thiết kế bởi các chuyên gia cho một giao thức hoặc chức năng riêng biệt. Bên cạnh đó, phương pháp SoC cung cấp một cách để giảm thiểu chi phí thiết kế và xác minh các lõi giữa nhiều chip [12].

Nhược điểm của SoC là một loại SoC khó đáp ứng nhu cầu của nhiều loại sản phẩm khác nhau. Chính vì vậy, mỗi hãng thiết kế và sản xuất chip đều có nhiều dòng SoC khác nhau, mỗi dòng sẽ đáp ứng một phân khúc sản phẩm nhất định và tối ưu nhất cho phân khúc sản phẩm này. Tối ưu ở đây được hiểu là số lượng chức năng mà chip tích hợp là vừa đủ, không quá nhiều và cũng không quá ít. Quá nhiều chức năng thì giá chip sẽ tăng, giá sản phẩm tăng trong khi chức năng thừa không được dùng đến. Quá ít chức năng thì không đáp ứng được nhu cầu ứng dụng của sản phẩm và khó tiêu thụ được sản phẩm ra thị trường [11].

2.1.4 Lõi IP

Các lõi sở hữu trí tuệ (Intellectual Property – IP) là các mô-đun độc lập hay một khối logic được sử dụng trong FPGA hoặc các mạch tích hợp dành riêng cho ứng dụng ASIC (Application-Specific Integrated Circuit). Chúng được phát triển bằng các ngôn ngữ mô tả phần cứng như VHDL, Verilog, System Verilog hoặc ngôn ngữ tổng hợp cấp cao (High-level Synthesis - HLS) như C.

Lõi IP có thể thực hiện các chức năng phức tạp và có thể được tái sử dụng trong thiết kế mạch tích hợp. Có 3 loại lõi IP là IP cứng (hard IP), IP mềm (soft IP) và IP bán cứng (firm IP). Lõi IP mềm đơn giản chính là RTL code (Verilog hoặc VHDL) thực hiện một chức năng nhất định. Lõi IP bán cứng được phát triển từ lõi IP mềm, lõi IP bán cứng sau khi được tổng hợp và bố trí (layout) sẽ tạo ra lõi IP cứng. Lõi IP cứng chính là một con chip hoặc vi mạch tích hợp hoàn chỉnh được đóng gói và sản xuất [15].

2.1.4.1 Lõi IP mềm

Lõi IP mềm được phát triển bằng ngôn ngữ mô tả phần cứng như VHDL, Verilog, System Verilog. Người dùng được cung cấp mã nguồn, vì vậy các IP có thể được sửa đổi tuỳ theo ứng dụng và nhu cầu. Chúng có thể dễ dàng được tích hợp và tái sử dụng cho các FPGA. Ưu điểm của các lõi IP mềm là có thể sửa chữa nếu có lỗi hoặc thay đổi nhỏ trong ứng dụng.

2.1.4.2 Lõi IP bán cứng

Các lõi IP bán cứng là một thiết kế ở cấp độ cổng (gate-level netlist), hay nói cách khác là các dạng code RTL sau khi qua quá trình tổng hợp sẽ chuyển sang dạng các cổng logic. Các lõi IP bán cứng có thể linh hoạt đặt mô-đun trong SoC và với cấu hình có thể do người dùng lập trình. Lõi IP bán cứng. Ưu điểm của lõi này là có thể sửa chữa ở một mức độ nhất định và việc thử nghiệm các lỗi đã được hoàn tất ở IP mềm.

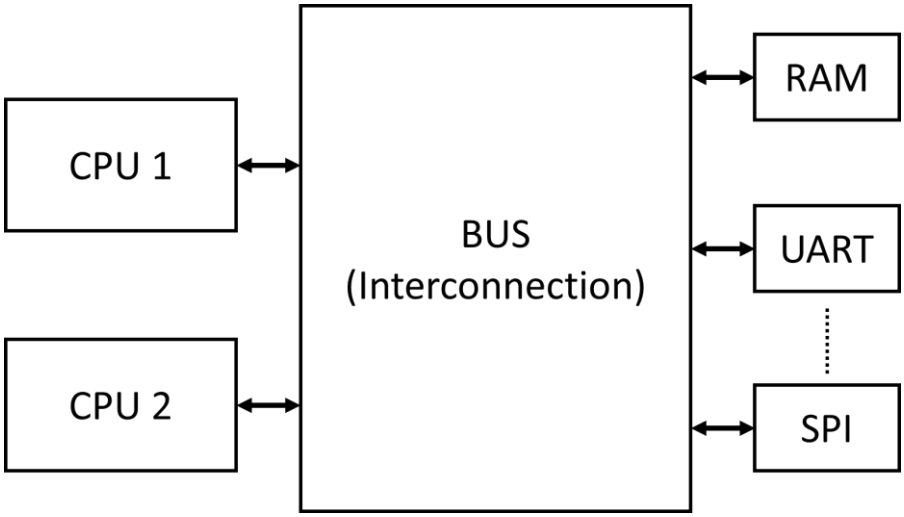
2.1.4.3 Lõi IP cứng

Lõi IP cứng là kết quả của firm IP sau khi đã hoàn tất giai đoạn kết nối các cổng logic theo một trật tự cụ thể. Các lõi cứng sau khi được đưa đi sản xuất sẽ trở thành một phần của SoC và thực hiện một chức năng cụ thể. Vì các khối này đã là một phần của thiết bị FPGA và vị trí cố định trong FPGA vì vậy các lõi này không thể được chuyển sang các FPGA khác [14].

2.2 TỔNG QUAN VỀ HỆ THỐNG BUS TRONG SOC

2.2.1 Hệ thống bus trong SoC

Hệ thống bus trong SoC được sử dụng như cầu nối phục vụ cho mục đích truy xuất dữ liệu đến một thành phần trong hệ thống. Trong một SoC phức tạp, sẽ có nhiều hệ thống bus được nối với nhau và với các module khác nhau. Kiến trúc kết nối của bus trên Soc là một trong những thách thức hàng đầu trong công nghệ SoC do tần số hoạt động tăng nhanh và kích thước chip ngày càng tăng. Hiệu suất của thiết kế SoC phụ thuộc rất nhiều vào cấu trúc bus mà thiết kế đó sử dụng. Thông thường, các lõi IP được thiết kế với nhiều giao diện và giao thức kết nối khác nhau. Việc tích hợp các lõi như vậy trong một SoC thường không tối ưu về mặt logic. Tuy nhiên các tiêu chuẩn của cấu trúc bus trên chip đã được phát triển để hạn chế vấn đề này.



Hình 2.3: Hệ thống bus trong SoC

Các giao thức bus phổ biến như kiến trúc bus vi điều khiển tiên tiến (AMBA) bus hiệu suất cao tiên tiến (AHB) và giao diện mở rộng nâng cao (AXI) từ ARM, WISHBONE từ Silicore Corporation, Open Core Protocol (OCP) của OCP International Partnership và CoreConnect của IBM. Đặc điểm chính của tất cả các bus này là chúng truyền dữ liệu một cách tuyến tính. Một số ứng dụng cụ thể như xử lý hình ảnh, thị giác máy tính và giao tiếp không dây, truyền tải dữ liệu khối và lưu trữ. Trong những trường hợp này, truyền dữ liệu theo khối hoặc ma trận được ưu tiên hơn là theo cụm tuyến tính. Ngoài ra, đối với một số cấu trúc bus nâng cao như kiến trúc đa bus và đa lớp, băng thông có thể được cải thiện khi các giao dịch tối đa xảy ra trong cùng một mức bus hoặc cùng một lớp bus.

Tuy nhiên, các bus này sử dụng một số lượng lớn các dây dẫn cho một số tập hợp các tín hiệu bus và xác định các cấu trúc phần cứng phức tạp gây tốn kém về diện tích silicon và tiêu thụ năng lượng. Các kiến trúc dựa trên bus truyền thống này cũng không phù hợp với các thiết bị di động chạy bằng pin do các tính năng của băng thông cao và tiêu thụ năng lượng cao [1] [2].

2.2.2 Đặc điểm của hệ thống bus trong SoC

Trong cách tiếp cận thiết kế dựa trên bus, các thành phần IP giao tiếp thông qua một hoặc nhiều bus thường được kết nối với nhau bằng các cầu bus. Vì đặc điểm kỹ thuật bus có thể được chuẩn hóa nên có thể phát triển các thư viện cho các thành phần có giao diện trực tiếp khớp với đặc điểm kỹ thuật này. Các công ty cung cấp các thư viện thành phần rất phong phú, có các môi trường mô phỏng và phát triển chuyên biệt để thiết kế các hệ thống xung quanh bus của họ.

Một cách tiếp cận hơi khác là thiết kế dựa trên các lõi. Trong trường hợp này, các thành phần lõi IP tuân theo giao diện chuẩn và độc lập với bus, do đó được kết nối trực tiếp với nhau. Mặc dù tiêu chuẩn có thể hỗ trợ một loạt các chức năng, mỗi thành phần có thể chỉ có một giao diện chỉ chứa các chức năng liên quan đến nó. Chúng cho phép tích hợp các thành phần IP đồng nhất tuân theo các tiêu chuẩn này được kết nối trực tiếp với nhau mà không yêu cầu phát triển các trình bao bọc phức tạp. Giải pháp được áp dụng rộng rãi này không may gặp phải các hạn chế về khả năng mở rộng sức mạnh và hiệu suất cũng như việc chia sẻ tài nguyên giữa các phần tử giao tiếp bị hạn chế.

Đối với mạng bus, bus bị chiếm bởi một giao tiếp duy nhất ngay cả khi nhiều giao tiếp có thể hoạt động đồng thời trên các phần khác nhau trên bus. Do đó, rất nhiều nỗ lực đã được dành cho sự phát triển của cấu trúc liên kết bus tiên tiến và các giao thức để hỗ trợ tốt hơn khả năng định tuyến, tính linh hoạt, độ tin cậy và khả năng cấu hình lại.

Do đó, một cách có hệ thống để thiết kế mạng với cấu trúc liên kết có thể tùy ý đang trở nên quan trọng. Đối với các nhu cầu cụ thể, SoC có thể được xây dựng dựa trên một chip mạng tinh vi và chuyên dụng có thể mang lại hiệu suất rất cao để kết nối một số lượng lớn các thành phần. Có vẻ như mô hình thiết kế này chuyển sang hướng giao tiếp trên chip được nhịp độ dựa trên các mạng vi mô kết nối với nhau hoặc mạng lưới trên chip [13].

2.3 BUS WISHBONE

2.3.1 Giới thiệu

WISHBONE là một kiến trúc bus hệ thống thường được sử dụng trong thiết kế SoC. Đây là một phương pháp thiết kế đơn giản, linh hoạt giúp cho việc kết nối các lõi IP trong một SoC trở nên hiệu quả và dễ dàng hơn. Các lõi IP có thể được phát triển bởi các nhà sản xuất khác nhau với các chức năng và cách kết nối khác nhau tùy vào nhu cầu của người dùng. Mục đích chính của bus WISHBONE là hỗ trợ người dùng chuẩn hóa các giao diện lõi IP dưới một giao diện chung, giúp cho việc kết nối các lõi IP trở nên dễ dàng và có thể tái sử dụng thiết kế này trên các hệ thống khác. Điều này cải thiện tính di động và độ tin cậy của hệ thống, đồng thời dẫn đến thời gian đưa ra thị trường nhanh hơn cho người dùng cuối. Trước đây, các lõi IP sử dụng các sơ đồ kết nối không tiêu chuẩn khiến chúng khó kết hợp với nhau. Bằng cách áp dụng sơ đồ kết nối tiêu chuẩn, người dùng cuối có thể tích hợp các lõi một cách nhanh chóng và dễ dàng hơn. Đặc điểm kỹ thuật này không yêu cầu sử dụng các công cụ phát triển cụ thể hoặc phần cứng mục tiêu. Ngoài ra, nó hoàn toàn phù hợp với hầu như tất cả các công cụ tổng hợp logic.

Kết nối WISHBONE được thiết kế như một giao diện mục đích chung. Nó giúp xác định trao đổi dữ liệu tiêu chuẩn giữa các mô-đun lõi IP nhưng không làm thay đổi các chức năng ứng dụng riêng biệt của lõi IP. Những nhà phát triển đã bị ảnh hưởng mạnh mẽ bởi ba yếu tố để phát triển nên bus WISHBONE. Đầu tiên, cần có một giải pháp tích hợp Hệ thống trên chip tốt, đáng tin cậy. Thứ hai, cần có một đặc điểm kỹ thuật giao diện chung để tạo điều kiện thuận lợi cho các phương pháp thiết kế có cấu trúc trên các nhóm dự án lớn. Thứ ba, họ bị ấn tượng bởi các giải pháp tích hợp hệ thống truyền thống được cung cấp bởi các bus máy tính vi mô như bus PCI và VMEbus.

Trên thực tế, kiến trúc WISHBONE tương tự như một bus máy tính vi mô ở chỗ cả hai đều cung cấp giải pháp tích hợp linh hoạt có thể dễ dàng phù hợp với một ứng dụng cụ thể, đưa ra nhiều chu kỳ bus và độ rộng đường dẫn dữ liệu để giải quyết các vấn đề hệ thống khác nhau và ít sản phẩm được thiết kế bởi nhiều nhà cung cấp do đó làm giảm giá trong khi cải thiện hiệu suất và chất lượng [15] [16].

2.3.2 Đặc điểm bus WISHBONE

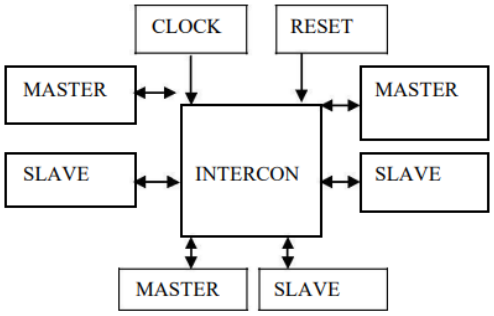
Kết nối WISHBONE giúp Hệ thống trên chip và tái sử dụng thiết kế dễ dàng bằng cách tạo giao thức trao đổi dữ liệu tiêu chuẩn. Các đặc điểm của công nghệ này bao gồm [15] [16]:

- Giao diện phần cứng lõi IP logic, nhỏ gọn, đơn giản, yêu cầu rất ít cổng logic.
- Tập hợp đầy đủ các giao thức bus truyền dữ liệu phổ biến bao gồm:
  - Chu kỳ ĐỌC/GHI
  - Chu kỳ chuyển giao BLOCK
  - Chu kỳ RMW (Read-Modify-Write)
- Giao thức bắt tay (handshaking protocol) tạo ra một giao thức đồng bộ cho phép mỗi MASTER và SLAVE có thể điều chỉnh tốc độ truyền dữ liệu.
- Tối đa một lần truyền dữ liệu cho mỗi chu kỳ Clock.
- Hỗ trợ kết thúc chu kỳ bình thường, thử kết thúc lại và kết thúc do lỗi.
- Độ rộng địa chỉ lên đến 64-bit có thể điều chỉnh.
- Lược đồ giải mã một phần địa chỉ (Partial address decoding scheme) cho SLAVE.
- Hỗ trợ thẻ do người dùng xác định (User-defined tag), dùng để phân biệt giữa truyền dữ liệu, vectơ ngắt, hoạt động điều khiển bộ đệm và các giao dịch bus khác.
- Kiến trúc MASTER / SLAVE cho các thiết kế hệ thống rất linh hoạt.
- Khả năng đa xử lý (đa MASTER đa SLAVE). Điều này cho phép tạo ra nhiều loại cấu hình SoC.
- Phương pháp phân xử (Arbitration methodology) do người dùng cuối xác định.
- Giao diện linh hoạt hỗ trợ bộ nhớ được ánh xạ (memory mapped), bộ nhớ FIFO và kết nối thanh ngang (crossbar).
- Độ rộng bus dữ liệu và kích thước toán hạng từ 8 đến 64-bit (có thể mở rộng).
- Hỗ trợ xung Clock đơn.
- Hỗ trợ cả dữ liệu BIG ENDIAN và LITTLE ENDIAN.
- Hỗ trợ đa dạng lõi IP kết nối khác nhau, bao gồm: point-to-point, shared bus, crossbar switch, kết nối luồng dữ liệu, off-chip.
- Thiết kế đồng bộ đảm bảo tính di động, đơn giản, dễ sử dụng và dễ kiểm tra.

- Đặc điểm kỹ thuật thời gian (timing) rất đơn giản.
- Các yêu cầu về tài liệu cho phép người dùng cuối nhanh chóng đánh giá từng giao diện.
- Độc lập với công nghệ phần cứng (FPGA, ASIC, lưỡng cực, MOS, v.v.).
- Không phụ thuộc vào phương thức phân phối (delivery method) (soft, firm hoặc hard core).
- Độc lập với công cụ tổng hợp (synthesis tool), bộ định tuyến (router) và công cụ bố trí (layout tool).

2.3.3 Kết nối WISHBONE

WISHBONE sử dụng kiến trúc MASTER/SLAVE. Các giao diện MASTER và SLAVE giao tiếp thông qua một giao diện kết nối được gọi là INTERCON [15] [16].



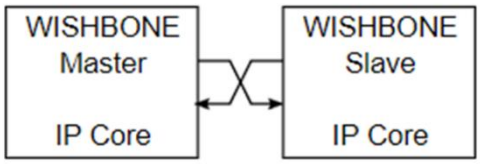
Hình 2.4: Giao diện kết nối INTERCON

Các giao diện MASTER là các lõi IP có khả năng bắt đầu các chu kỳ bus, đưa ra yêu cầu truyền hoặc nhận dữ liệu tới hoặc từ giao diện SLAVE. Trong khi các giao diện SLAVE có khả năng tiếp nhận các chu kỳ bus và thực hiện yêu cầu từ MASTER.

Kết nối WISHBONE cho phép người dùng thay đổi cách mà các lõi IP kết nối với nhau giúp cho việc triển khai phần cứng có khả năng tương thích với nhiều loại cấu trúc liên kết. Điều này rất quan trọng vì không có một cách duy nhất nào đúng để thực hiện cho mọi SoC. Có bốn loại kết nối WISHBONE được xác định và bao gồm:

- Điểm - điểm (Point-to-Point)
- Luồng dữ liệu (Data Flow)
- Bus chia sẻ (Shared Bus)
- Chuyển mạch chéo (Crossbar Switch)

2.3.3.1 Kết nối Điểm - điểm (Point-to-Point)



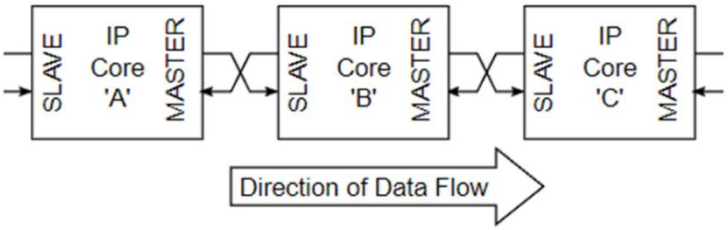
Hình 2.5: Kết nối Điểm - điểm

Kết nối điểm - điểm hỗ trợ kết nối trực tiếp hai bên tham gia truyền dữ liệu, cho phép kết nối trực tiếp từ một giao diện MASTER đến một giao diện SLAVE. Đây là cách đơn giản nhất để kết nối hai lõi IP. Ví dụ hình 2.4 bên trên: giao diện MASTER có thể nằm trên lõi IP của bộ vi xử lý và giao diện SLAVE có thể nằm trên cổng I/O nối tiếp.

Vì giao diện kết nối điểm - điểm chỉ hỗ trợ kết nối của một giao diện chính và một giao diện phụ duy nhất, hạn chế của nó không làm phù hợp với kết nối đa thiết bị SoC.

2.3.3.2 Kết nối Luồng dữ liệu (Data Flow)

Kết nối Luồng dữ liệu được sử dụng khi dữ liệu được xử lý theo cách tuần tự. Như trong hình 2.5 bên dưới, mỗi lõi IP trong kiến trúc luồng dữ liệu có cả giao diện MASTER và SLAVE. Luồng dữ liệu tuần tự từ lõi đến lõi tiếp theo. Đôi khi quá trình này có thể được gọi là kết nối ống dẫn (pipelining).

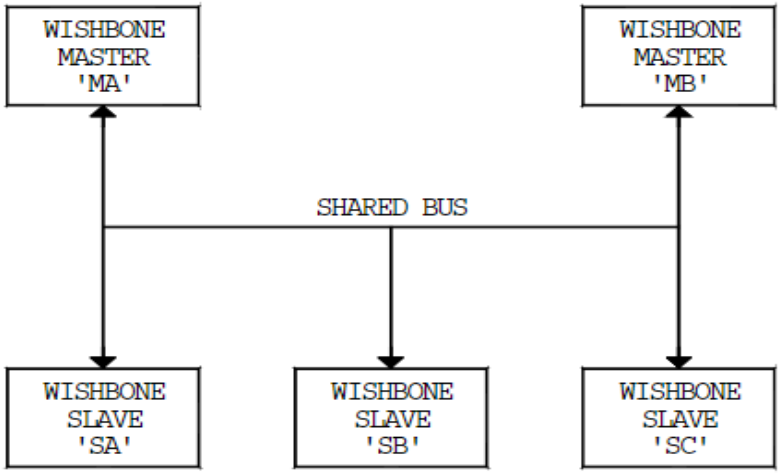


Hình 2.6: Kết nối Luồng dữ liệu

Kiến trúc Luồng dữ liệu khai thác tính song song, do đó tăng tốc thời gian thực thi. Để kiểm tra, nếu mỗi lõi IP trong đại diện cho một bộ xử lý thì hệ thống có tiềm năng xử lý số gấp ba lần so với một đơn vị. Tất nhiên, điều này giả định rằng mỗi lõi IP cần một khoảng thời gian như nhau để giải quyết vấn đề của nó và vấn đề có thể được giải quyết theo cách tuần tự. Trong thực tế, điều này có thể đúng hoặc có thể không đúng, nhưng nó minh họa cách kiến trúc Luồng dữ liệu có thể cung cấp mức độ song song cao khi giải quyết vấn đề.

2.3.3.3 Kết nối Bus chia sẻ (Shared Bus)

Kết nối Bus chia sẻ, hay Bus dùng chung, rất hữu ích để kết nối hai hoặc nhiều MASTER với một hoặc nhiều SLAVE. Sơ đồ khối được thể hiện như trong hình 2.6. Trong cấu trúc liên kết này, MASTER bắt đầu một chu kỳ bus đến một SLAVE mục tiêu đã được xác định từ trước. SLAVE tham gia vào một hoặc nhiều chu kỳ bus với MASTER đó.



Hình 2.7: Kết nối Bus chia sẻ

Một bộ phân xử (arbiter) xác định thời điểm MASTER có thể truy cập vào bus chung. Bộ phân xử hoạt động để xác định thời điểm và cách thức mỗi MASTER truy cập vào tài nguyên được chia sẻ. Ngoài ra, loại bộ phân xử hoàn toàn được xác định bởi người dùng.

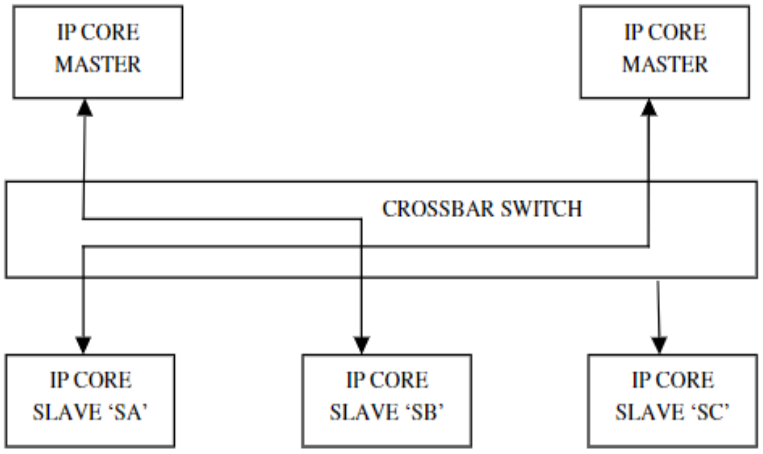
Ưu điểm chính của kỹ thuật này là các hệ thống kết nối được chia sẻ tương đối hợp lý. Nói chung, nó yêu cầu ít cổng logic và tài nguyên định tuyến hơn các cấu hình khác, đặc biệt là Chuyển mạch chéo (Crossbar switch). Nhược điểm chính của nó là các MASTER có thể phải đợi trước khi có được quyền truy cập vào kết nối. Điều này làm giảm tốc độ tổng thể mà MASTER có thể truyền dữ liệu.

Không có đặc điểm chỉ định cụ thể nào về cách triển khai Bus chia sẻ. Nó có thể được thực hiện với bộ ghép kênh (multiplexer) hoặc bus ba trạng thái (three-state bus). Điều này mang lại cho người dùng tính linh hoạt, vì một số chip logic hoạt động tốt hơn với logic ghép kênh và một số hoạt động tốt hơn với bus ba trạng thái.

Bus chia sẻ thường được sử dụng trong các bus tiêu chuẩn như PCI và VMEbus.

2.3.3.4 Kết nối Chuyển mạch chéo (Crossbar switch)

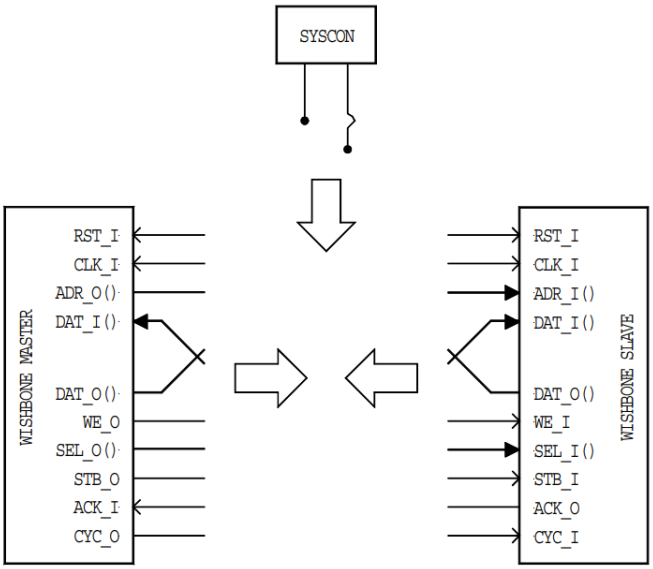
Kết nối Chuyển mạch chéo được sử dụng khi kết nối hai hoặc nhiều MASTER với nhau để mỗi cái có thể truy cập hai hoặc nhiều SLAVE. Sơ đồ kết nối Chuyển mạch chéo được thể hiện trong hình 2.7 bên dưới. Trong kết nối này, MASTER bắt đầu một chu kỳ bus có thể định địa chỉ đến một SLAVE chỉ định. Một bộ phân xử xác định khi nào mỗi MASTER có thể truy cập vào SLAVE được chỉ định. Không giống như kết nối Bus chia sẻ, kết nối Chuyển mạch chéo có nhiều hơn một MASTER để sử dụng kết nối miễn là hai MASTER không truy cập cùng một SLAVE cùng một lúc. Theo phương pháp này, mỗi MASTER sẽ phân xử cho một kênh khi truyền dữ liệu. Khi điều này được thiết lập, dữ liệu được chuyển giữa MASTER và SLAVE qua một liên kết giao tiếp riêng. hình 2.7 cho thấy hai kênh có thể xuất hiện trên kết nối Chuyển mạch chéo, kết nối MASTER 'MA' với SLAVE 'SB' và kết nối MASTER 'MB' với SLAVE 'SA'. Tốc độ truyền dữ liệu tổng thể của kết nối Chuyển mạch chéo nhanh hơn so với cơ chế Bus chia sẻ.



Hình 2.8: Kết nối Chuyển mạch chéo

2.3.4 Tín hiệu WISHBONE

Mỗi giao diện MASTER và SLAVE có một bộ tín hiệu. Hầu hết chúng là tùy chọn và có thể có hoặc không trên một giao diện cụ thể. WISHBONE sử dụng giao diện mà các lõi IP được kết nối một cách hợp lý với các công cụ phát triển của người dùng. Ví dụ, các lõi IP mềm VHDL sẽ được kết nối với nhau bằng các mô tả trong một cặp phần tử / kiến trúc.



Hình 2.9: Tín hiệu kết nối hai lõi IP MASTER/SLAVE

Hai lõi MASTER / SLAVE IP được kết nối với nhau được hiển thị trong hình 2.8 bên trên. Chúng cho thấy cách các tín hiệu kết nối với giao diện MASTER và SLAVE để tạo thành một kết nối điểm - điểm. Người dùng cuối hình thành các kết nối này bằng các công cụ phát triển của họ. Ở đây chúng tôi phân loại tín hiệu WISHBONE thành bốn loại:

- Tín hiệu của mô-đun SYSCON
- Tín hiệu chung của giao diện WISHBONE MASTER và SLAVE
- Tín hiệu của giao diện WISHBONE MASTER
- Tín hiệu của giao diện WISHBONE SLAVE

Tất cả các tín hiệu giao diện WISHBONE phải sử dụng logic tích cực mức cao [15]. Một tín hiệu được khẳng định (asserted) là tín hiệu chuyển từ mức logic thấp “0” sang mức logic cao “1” (chuyển từ trạng thái không hoạt động sang trạng thái hoạt động). Ngược lại, một tín hiệu bị phủ định (negated) là tín hiệu chuyển từ mức logic cao “1” sang mức logic thấp “0” (chuyển từ trạng thái hoạt động sang không hoạt động).

2.3.4.1 Tín hiệu của mô-đun SYSCON

SYSCON chủ yếu tạo ra hai tín hiệu, RST_O và CLK_O. RST_O được sử dụng để cấu trúc thiết lập lại hệ thống với MASTER và SLAVE. CLK_O cung cấp xung Clock cho cả MASTER và SLAVE giữ cho chúng hoạt động đồng bộ [15].

2.3.4.2 Tín hiệu chung của giao diện WISHBONE MASTER và SLAVE

Những tín hiệu sau đây sử dụng chung cho cả MASTER và SLAVE [15]:

- CLK_I: Ngõ vào đồng hồ được sử dụng để đồng bộ hóa tất cả các hoạt động logic bên trong master và slave. Tín hiệu này được kết nối với CLK_O của mô-đun SYSCON.
- RST_I: Ngõ vào thiết lập lại được sử dụng để kích hoạt thiết lập lại hệ thống. Điều này làm cho hệ thống WISHBONE trở lại trạng thái làm việc ban đầu. Tín hiệu này được kết nối với RST_O của mô-đun SYSCON.
- DAT_I: Bus ngõ vào dữ liệu được sử dụng để nhận một mảng dữ liệu. Kích thước của dữ liệu được xác định bởi kích thước cổng được xác định trong quá trình thiết kế bus WISHBONE.
- DAT_O: Bus ngõ ra dữ liệu được sử dụng để truyền một mảng dữ liệu. Kích thước của dữ liệu được xác định bởi kích thước cổng được xác định trong quá trình thiết kế bus WISHBONE.

2.3.4.3 Tín hiệu của giao diện WISHBONE MASTER

Những tín hiệu sau đây chỉ sử dụng cho MASTER [15]:

- ACK_I: Tín hiệu ngõ vào xác nhận được sử dụng bởi MASTER để kết thúc bình thường chu kỳ bus hiện tại.
- ADR_O: Ngõ ra địa chỉ được sử dụng để gửi địa chỉ dưới dạng dữ liệu nhị phân, nó thường được sử dụng để trỏ địa chỉ của vị trí mà dữ liệu phải được ghi hoặc đọc. Kích thước cổng của nó được xác định trong quá trình thiết kế bus WISHBONE.
- CYC_O: Ngõ ra Chu kỳ (Cycle) được sử dụng để chỉ ra rằng chu kỳ bus hợp lệ đang diễn ra. Tín hiệu được khẳng định miễn là hoạt động hiện tại vẫn đang diễn ra.
- SEL_O: Ngõ ra Lựa chọn (Select) giữ dữ liệu nhị phân cho biết vị trí của dữ liệu hợp lệ có trong DAT_O và DAT_I.
- STB_O: Tín hiệu ngõ ra Nhấp nháy (Strobe) được sử dụng để chỉ ra chu kỳ truyền dữ liệu hợp lệ, nó được sử dụng để xác định sự hiện diện của dữ liệu hợp lệ trên dữ liệu và lựa chọn bus.

- WE_O: Tín hiệu ngõ ra Cho phép ghi (Write Enable) được sử dụng để đánh dấu chu kỳ bus hiện tại là ghi dữ liệu.

2.3.4.4 Tín hiệu của giao diện WISHBONE SLAVE

Những tín hiệu sau đây chỉ sử dụng cho SLAVE [15]:

- ACK_O: Tín hiệu ngõ ra xác nhận được gửi bởi SLAVE để kết thúc bình thường chu kỳ bus hiện tại.
- ADR_I: Ngõ vào địa chỉ được sử dụng để chấp nhận địa chỉ ở dạng dữ liệu nhị phân, nó thường được sử dụng để trỏ địa chỉ của vị trí mà dữ liệu phải được ghi hoặc đọc bởi dữ liệu chính. Kích thước cổng của nó được xác định trong quá trình thiết kế bus WISHBONE.
- CYC_I: Ngõ vào Chu kỳ (Cycle) được sử dụng để phát hiện rằng chu kỳ bus hợp lệ đang diễn ra. Tín hiệu được khẳng định miễn là hoạt động hiện tại vẫn đang diễn ra.
- SEL_I: Ngõ vào Lựa chọn (Select) giữ dữ liệu nhị phân cho biết vị trí của dữ liệu hợp lệ có trong DAT_O và DAT_I.
- STB_I: Tín hiệu ngõ vào Nhấp nháy (Strobe) được sử dụng để phát hiện chu kỳ truyền dữ liệu hợp lệ, nó được sử dụng để xác định sự hiện diện của dữ liệu hợp lệ trên dữ liệu và chọn bus.
- WE_I: Tín hiệu ngõ vào Cho phép ghi (Write Enable) được sử dụng để chấp nhận chu kỳ bus hiện tại là ghi dữ liệu.

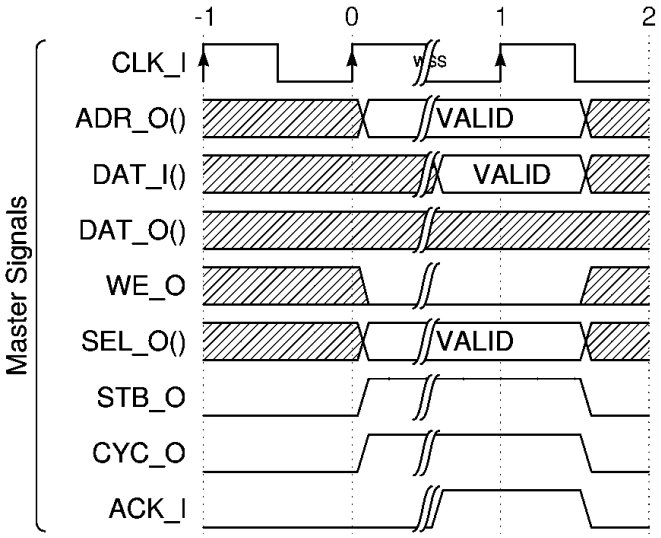
2.3.5 Chu kỳ ĐỌC/GHI ĐƠN

Các chu kỳ ĐỌC/GHI ĐƠN thực hiện một lần truyền dữ liệu tại một thời điểm. Đây là các chu trình cơ bản được sử dụng để thực hiện truyền dữ liệu trên kết nối WISHBONE [15].

2.3.5.1 Chu kỳ ĐỌC ĐƠN

Trong chu kỳ ĐỌC ĐƠN, chỉ một đoạn dữ liệu được lấy bởi MASTER từ SLAVE. Kích thước đoạn dữ liệu đó được xác định trước bởi nhà thiết kế và có thể là 8-bit, 16-bit, 32-bit hoặc 64-bit [15].

Chu kỳ bus của ĐỌC ĐƠN cổ điển được mô tả trong hình 2.9 bên dưới. STB_O và ACK_I đóng một phần quan trọng trong giao thức bắt tay và đảm bảo đồng bộ hóa cần thiết với SLAVE. WE_O luôn bị phủ định trong suốt chu kỳ bus để duy trì hoạt động đọc.

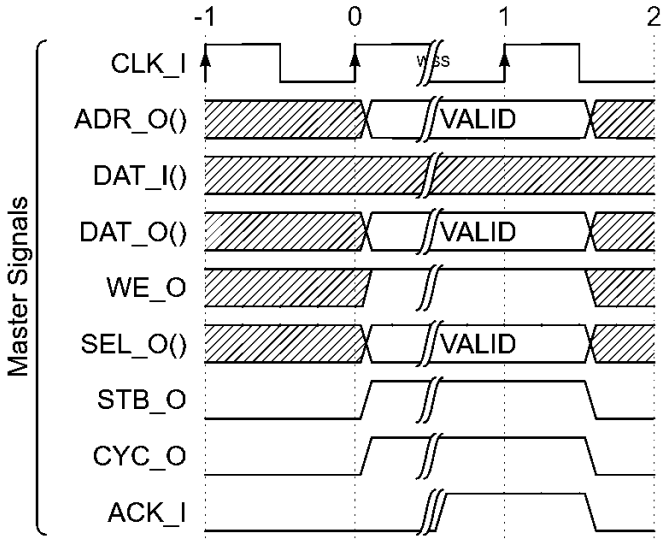


Hình 2.10: Chu kỳ ĐỌC ĐƠN

2.3.5.2 Chu kỳ GHI ĐƠN

Trong chu kỳ GHI ĐƠN, chỉ một đoạn dữ liệu được ghi bởi MASTER tới SLAVE. Kích thước đoạn dữ liệu được xác định trước bởi nhà thiết kế và có thể là 8-bit, 16-bit, 32-bit hoặc 64-bit [15].

Chu kỳ bus của GHI ĐƠN cổ điển được mô tả trong hình 2.10 bên dưới. Sự khác biệt duy nhất so với chu kỳ đọc đơn là WE_O được khẳng định vào khoảng thời gian cần thiết và do đó nó đảm bảo hoạt động ghi.



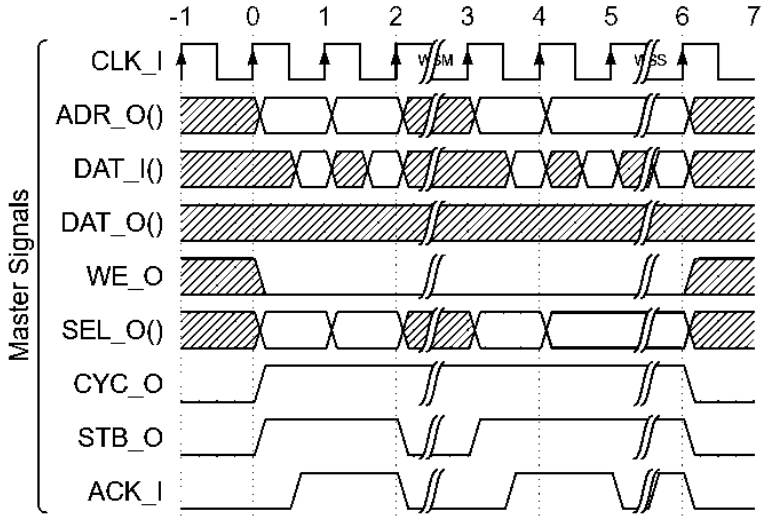
Hình 2.11: Chu kỳ GHI ĐƠN

2.3.6 Chu kỳ ĐỌC/GHI KHỐI

Các chu kỳ KHỐI thực hiện nhiều lần truyền dữ liệu. Chúng rất giống với các chu kỳ ĐỌC/GHI ĐƠN. Tuy nhiên, các chu kỳ KHỐI được sửa đổi một chút để các chu kỳ riêng lẻ này (được gọi là các pha) được kết hợp với nhau để tạo thành một chu kỳ KHỐI duy nhất. Chức năng này hữu ích nhất khi sử dụng nhiều MASTER trên kết nối. Ví dụ: nếu SLAVE là một bộ nhớ chia sẻ (cổng kép), thì một bộ phân xử cho bộ nhớ đó có thể xác định thời điểm một MASTER thực hiện xong với nó và MASTER khác có thể truy cập tiếp vào bộ nhớ [15].

2.3.6.1 Chu kỳ ĐỌC KHỐI

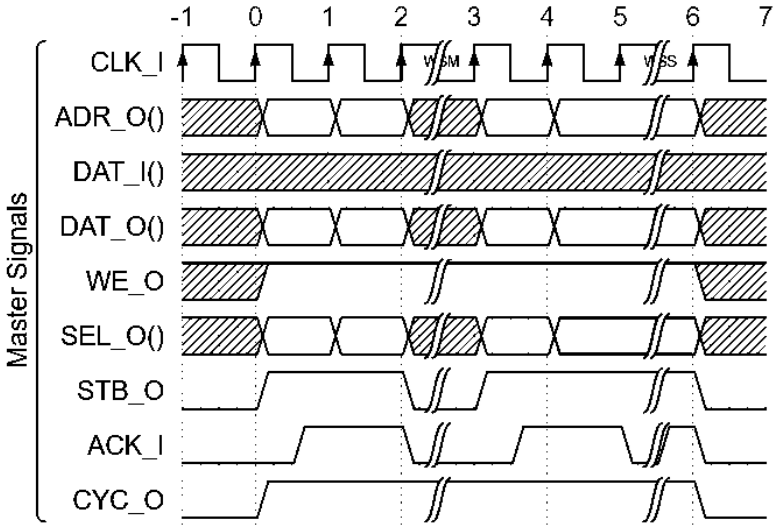
Trong chu kỳ ĐỌC KHỐI, một khối dữ liệu hoàn chỉnh được đọc bởi MASTER từ SLAVE. Một nhóm dữ liệu đơn lẻ tạo thành một khối. Chu kỳ bus của ĐỌC KHỐI cổ điển được mô tả trong hình 2.11 bên dưới. STB_O và ACK_I đóng một phần quan trọng trong giao thức bắt tay và đảm bảo đồng bộ hóa cần thiết với SLAVE. WE_O luôn được khẳng định để đảm bảo rằng chu kỳ bus hoàn chỉnh chỉ là hoạt động đọc. CYC_O được khẳng định cho đến khi khối dữ liệu hoàn chỉnh được truyền.



Hình 2.12: Chu kỳ ĐỌC KHỐI

2.3.6.2 Chu kỳ GHI KHỐI

Trong chu kỳ GHI KHỐI, chỉ một khối dữ liệu hoàn chỉnh được ghi bởi MASTER đến SLAVE. Một nhóm dữ liệu đơn lẻ tạo thành một khối. Chu kỳ bus của GHI KHỐI cổ điển được mô tả trong hình 2.12 bên dưới. Sự khác biệt duy nhất so với chu kỳ đọc khối là WE_O luôn được khẳng định vào đúng khoảng thời gian để đảm bảo rằng chu kỳ bus hoàn chỉnh chỉ dành cho hoạt động ghi.



Hình 2.13: Chu kỳ GHI KHỐI

2.4 BUS AMBA AXI

2.4.1 Giới thiệu

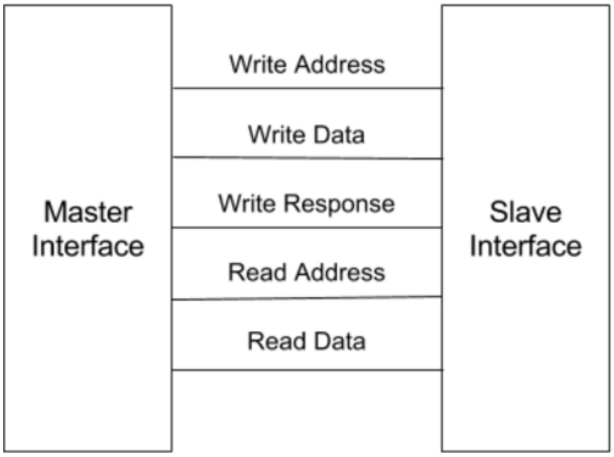
AXI (Advanced eXtensible Interface) là một trong các giao thức BUS (BUS protocol) trong bộ giao tiếp AMBA được phát triển bởi hãng ARM được ứng dụng rộng rãi trong các vi điều khiển ARM vì AXI là giao thức mở theo cơ chế burst, có nhiều đặc điểm hỗ trợ các hệ thống hiệu năng cao, tần số cao, phù hợp với các thiết kế có độ trễ thấp và băng thông cao. AXI cung cấp những hoạt động ở tần số cao mà không cần sử dụng các cầu nối phức tạp, đồng thời đáp ứng các yếu cầu giao tiếp của nhiều thành phần trên chip và cũng rất tương thích với các giao thức AHB và APB của họ AMBA [17].

Giao thức AXI quy định về chuẩn giao tiếp giữa theo 3 loại:

- Một MASTER và Interconnect bus
- Một SLAVE và Interconnect bus
- Một MASTER kết nối trực tiếp với một SLAVE

2.4.2 Cấu trúc bus AMBA AXI

Giao thức AXI quy định cơ chế, cách thức xử lý cho 2 hoạt động cơ bản là đọc và ghi. Hai hoạt động này được gọi chung là một truy cập (access). Một truy cập được hiểu là một giao dịch (transaction).



Hình 2.14: Các kênh giao tiếp giữa AXI MASTER và SLAVE

AXI hoạt động dựa trên năm loại kênh độc lập. hình 2.13 ở trên mô tả cụ thể năm kênh này. Mỗi kênh có vai trò, nhiệm vụ khác nhau và việc hoàn thành nhiệm vụ của mỗi kênh không phụ thuộc vào các kênh khác nhưng chúng vẫn liên quan đến nhau. Năm kênh này bao gồm:

- Kênh Địa Chỉ Đọc (Read Address Channel), truyền thông tin địa chỉ và thông tin điều khiển của một quá trình đọc từ MASTER đến SLAVE.
- Kênh Dữ Liệu Đọc (Read Data Channel), truyền dữ liệu đọc và thông tin phản hồi của một quá trình đọc từ SLAVE đến MASTER.
- Kênh Địa Chỉ Ghi (Write Address Channel), truyền thông tin địa chỉ và thông tin điều khiển của một transaction ghi từ MASTER đến SLAVE.
- Kênh Dữ Liệu Ghi (Write Data Channel), truyền dữ liệu ghi từ MASTER đến SLAVE.
- Kênh Đáp Ứng Ghi (Write Response Channel), truyền thông tin phản hồi của một quá trình ghi từ SLAVE đến MASTER.

Các tín hiệu sử dụng bên trong mỗi kênh được giao thức AXI quy định và được mô tả trong các bảng từ bảng 2.1 đến bảng 2.5 bên dưới.

Bảng 2.1: Các tín hiệu trong kênh Địa Chỉ Đọc của bus AMBA AXI

Tín hiệu	Nguồn
axi_arid	MASTER
axi_araddr	MASTER
axi_arlen	MASTER
axi_arsize	MASTER
axi_arburst	MASTER

axi_arlock	MASTER
axi_arcache	MASTER
axi_arprot	MASTER
axi_arqos	MASTER
axi_arregion	MASTER
axi_aruser	MASTER
axi_arvalid	MASTER
axi_arready	SLAVE

Bảng 2.2: Các tín hiệu trong kênh Dữ Liệu Đọc của bus AMBA AXI

Tín hiệu	Nguồn
axi_rid	SLAVE
axi_rdata	SLAVE
axi_rresp	SLAVE
axi_rlast	SLAVE
axi_ruser	SLAVE
axi_rvalid	SLAVE
axi_rready	MASTER

Bảng 2.3: Các tín hiệu trong kênh Địa Chỉ Ghi của bus AMBA AXI

Tín hiệu	Nguồn
axi_awid	MASTER
axi_awaddr	MASTER
axi_awlen	MASTER
axi_awsize	MASTER
axi_awburst	MASTER
axi_awlock	MASTER
axi_awcache	MASTER
axi_awprot	MASTER
axi_awqos	MASTER
axi_awregion	MASTER
axi_awuser	MASTER
axi_awvalid	MASTER
axi_awready	SLAVE

Bảng 2.4: Các tín hiệu trong kênh Dữ Liệu Ghi của bus AMBA AXI

Tín hiệu	Nguồn
axi_wid	MASTER
axi_wdata	MASTER
axi_wstrb	MASTER
axi_wlast	MASTER
axi_wuser	MASTER
axi_wvalid	MASTER
axi_wready	SLAVE

Bảng 2.5: Các tín hiệu trong kênh Phản Hồi Ghi của bus AMBA AXI

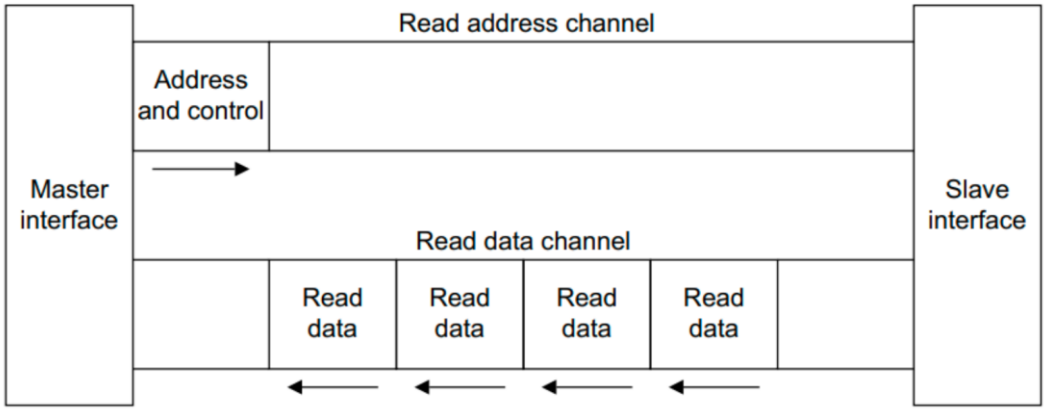
Tín hiệu	Nguồn
axi_bid	SLAVE
axi_bresp	SLAVE
axi_buser	SLAVE
axi_bvalid	SLAVE
axi_bready	MASTER

2.4.3 Các quy định về hoạt động

Một hoạt động ghi hoặc đọc trong giao thức này được xem là một transaction. Một transaction bao gồm 4 thông tin cơ bản:

- Thông tin địa chỉ để xác định đích mà transaction sẽ được xử lý.
- Thông tin điều khiển để xác định thuộc tính của transaction như loại burst, độ dài burst, kích thước burst và các thuộc tính khác.
- Thông tin dữ liệu của transaction.
- Thông tin phản hồi để xác định trạng thái của transaction có bị lỗi hay không.

Một transaction đọc được quản lý bởi 2 kênh: kênh Địa Chỉ Đọc và kênh Dữ Liệu Đọc, được mô tả như hình 2.14 bên dưới.

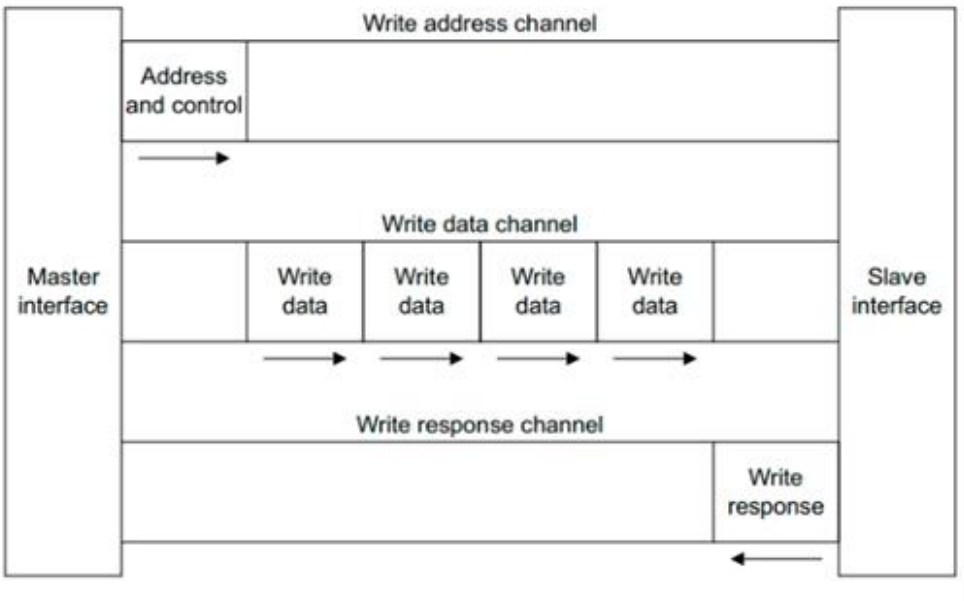


Hình 2.15: Hoạt động của một quá trình đọc

Một quá trình đọc gồm hai bước xử lý:

- Phía MASTER gửi một thông tin địa chỉ và thông tin điều khiển để khởi động một quá trình đọc trên kênh Địa Chỉ Đọc.
- Phía SLAVE gửi các dữ liệu kèm thông tin phản hồi trên kênh Phản Hồi Đọc. Số lượng dữ liệu và thông tin phản hồi được quy định bởi thông tin điều khiển phát từ phía MASTER trên kênh Địa Chỉ Đọc.

Một quá trình ghi được quản lý bởi 3 kênh: kênh Địa Chỉ Ghi, kênh Dữ Liệu Ghi và kênh Phản Hồi Ghi, được mô tả như hình 2.15 bên dưới.



Hình 2.16: Hoạt động của một quá trình ghi

Một quá trình ghi gồm 3 bước xử lý:

- Phía MASTER gửi một thông tin địa chỉ và thông tin điều khiển để khởi động một quá trình ghi trên kênh Địa Chỉ Ghi.
- Phía MASTER gửi các dữ liệu ghi trên kênh Dữ Liệu Ghi. Số lượng dữ liệu ghi được quy định bởi thông tin điều khiển phát từ phía MASTER trên kênh Địa Chỉ Ghi.
- Phía SLAVE gửi thông tin phản hồi ghi thành công trên kênh Phản Hồi Ghi khi burst ghi đã hoàn thành.

2.4.4 AMBA AXI4

2.4.4.1 Giới thiệu

Phiên bản đầu tiên của AXI được xuất hiện lần đầu trong AMBA 3.0 ra mắt vào năm 2003. Đến 2010, AMBA 4.0 ra mắt, trong đó có phiên bản thứ 2 của AXI là AXI4.

Các đặc trưng quan trọng của AXI4 là:

- Tách riêng pha truyền địa chỉ, thông tin điều khiển và pha truyền dữ liệu.
- Sử dụng các quá trình dựa trên cơ chế burst và chỉ cần cấp phát địa chỉ đầu tiên của burst.
- Tách riêng kênh dữ liệu đọc và kênh dữ liệu ghi.
- Các quá trình có thể hoàn thành không theo thứ tự (out-of-order).
- Cho phép chèn thêm các tầng thanh ghi giúp timing của BUS nói riêng và của hệ thống tốt hơn.

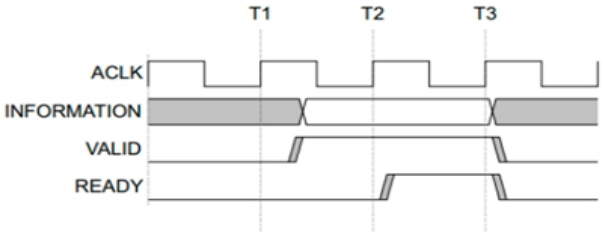
2.4.4.2 Cơ chế bắt tay của AXI4

Giao thức AXI hoạt động dựa trên cơ chế bắt tay hai chiều (two-way handshake) hay còn gọi là cơ chế kiểm soát dòng chảy hai chiều (two-way flow control) sử dụng một tín hiệu VALID và một tín hiệu READY để MASTER và SLAVE có thể biết được thông tin nào được truyền giữa MASTER và SLAVE.

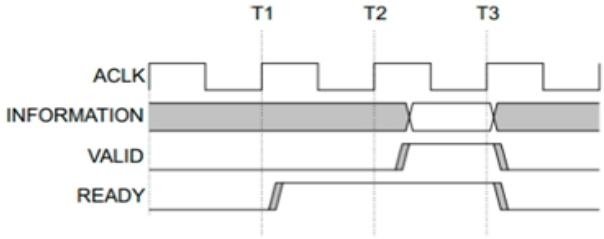
Nguyên tắc hoạt động của cơ chế bắt tay là nguồn phải tích cực tín hiệu VALID bất cứ khi nào có thông tin hợp lệ cần truyền mà không cần quan tâm READY đã tích cực hay chưa. VALID phải được duy trì cho đến khi READY tích cực và việc bắt tay xảy ra tại cạnh lên xung Clock lúc mà VALID và READY đều tích cực. Đích chỉ tích cực

READY khi sẵn sàng nhận dữ liệu, vì vậy READY có thể tích cực trước, trong hoặc sau khi VALID đã tích cực.

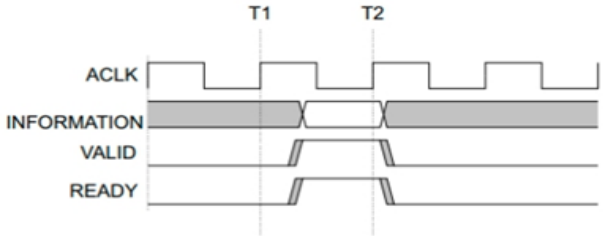
Có ba trường hợp của cơ chế bắt tay. Trường hợp tín hiệu VALID tích cực trước tín hiệu READY, trường hợp tín hiệu VALID tích cực sau READY và trường hợp tín hiệu VALID tích cực cùng lúc với READY. Các trường hợp được mô tả lần lượt như các hình 2.17, hình 2.18 và hình 2.19 bên dưới:



Hình 2.17: Trường hợp tín hiệu VALID tích cực trước READY



Hình 2.18: Trường hợp tín hiệu VALID tích cực sau READY



Hình 2.19: Trường hợp tín hiệu VALID tích cực cùng lúc với READY

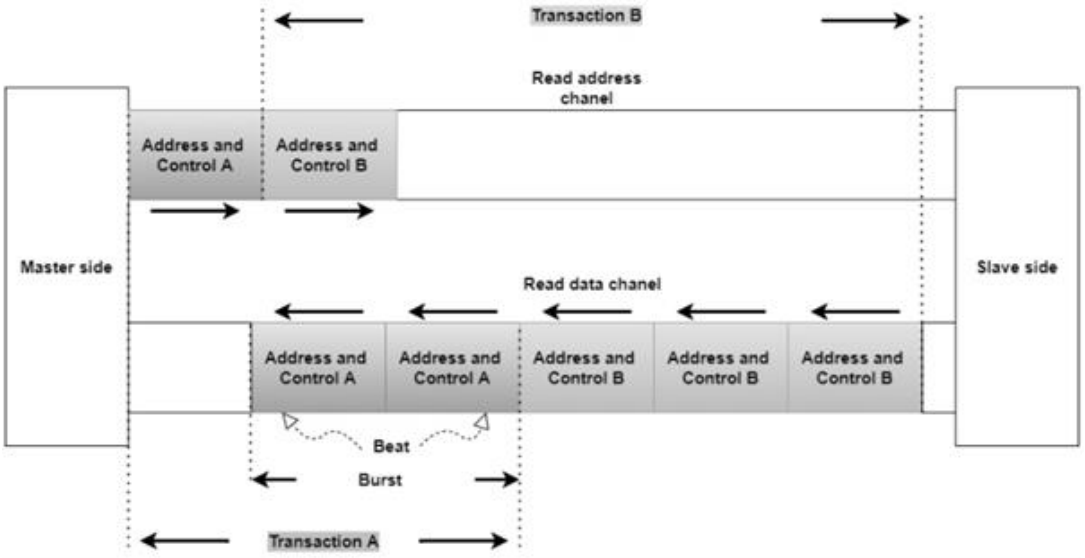
Năm kênh độc lập của AXI đều có những tín hiệu VALID và READY riêng cho từng kênh. Bảng 2.6 bên dưới liệt kê các cặp tín hiệu bắt tay của năm kênh.

Bảng 2.6: Các cặp tín hiệu bắt tay trong năm kênh truyền của bus AMBA AXI

Kênh	Cặp tín hiệu bắt tay
Write Address (Địa Chỉ Ghi)	AWVALID, AWREADY
Write Data (Dữ Liệu Ghi)	WVALID, WREADY
Write Response (Phản Hồi Ghi)	BVALID, BRREADY
Read Address (Địa Chỉ Đọc)	ARVALID, ARREADY
Read Data (Dữ Liệu Đọc)	RVALID, RREADY

2.4.4.3 Cơ chế burst của AXI4

AXI hoạt động dựa theo cơ chế burst, mỗi quá trình điều khiển một burst nên quá trình còn được gọi đầy đủ là burst transaction. Burst được chia thành 3 loại: burst FIXED, burst INCR, burst WRAP. Một burst là toàn bộ các transfer dữ liệu trong một quá trình. Mỗi transfer dữ liệu gọi là một beat. Mỗi burst gồm một hoặc nhiều beat. Mỗi quá trình truyền quản lý một burst có thuộc tính được quy định bởi thông tin điều khiển. hình 2.20 bên dưới mô tả về cơ chế burst.

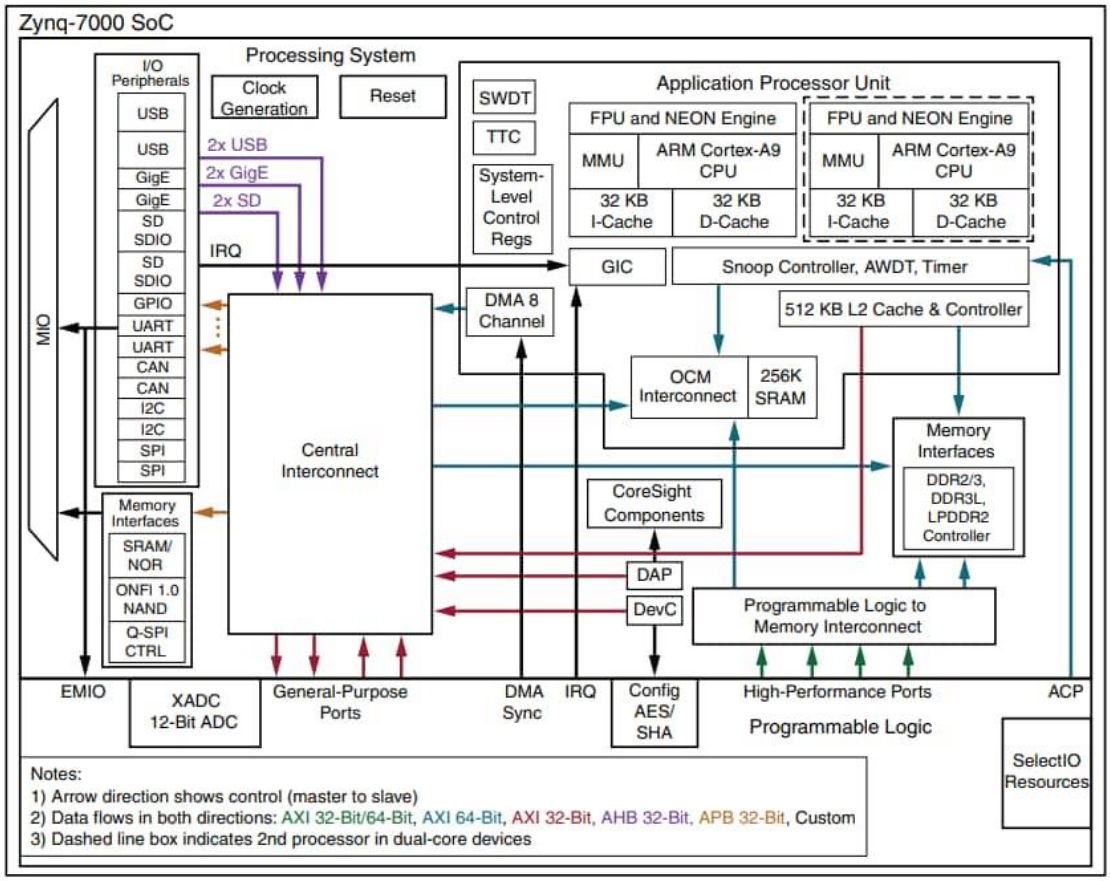


Hình 2.20: Cơ chế burst của AXI

Đặc kiểm của cơ chế burst trong AXI là phía MASTER không phát các địa chỉ trung gian của các beat trong một burst mà chỉ phát địa chỉ byte đầu tiên trong một quá trình, đây chính là địa chỉ của transfer dữ liệu đầu tiên. Phía Slave dựa trên thông tin điều khiển để xác định địa chỉ của các beat từ địa chỉ đầu tiên này. Điều này giúp tăng hiệu suất của hệ thống bus nhưng làm mạch logic xử lý giao tiếp AXI tại phía SLAVE phức tạp hơn vì phải tự tính toán các địa chỉ beat.

2.5 NỀN TẢNG ZYNQ-7000 SOC

ZYNQ – 7000 là sản phẩm được sản xuất bởi Xilinx vào năm 2012, dựa trên nền tảng cấu trúc của PSoC. ZYNQ-7000 được tích hợp một hệ thống xử lý dựa trên lõi kép ARM Cortex-A9 (PS) và nền tảng FPGA Xilinx 28 nm (PL). CPU ARM Cortex-A9 MPCore là trung tâm của PS, bao gồm các bộ nhớ on-chip, giao diện bộ nhớ ngoài và bộ thiết bị ngoại vi I/O phong phú [12]. Nền tảng ZYNQ-7000 hỗ trợ rất tốt cho các thiết kế SoC, các khối IP được kết nối thông qua giao diện bus AXI [18].



Hình 2.21: Kiến trúc ZYNQ-7000 SoC

Kiến trúc của ZYNQ-7000 bao gồm các khối sau:

- Khối Hệ thống xử lý - PS (Processing System)
  - Khối đơn vị xử lý - APU (Application processor unit)
  - Khối giao diện giao tiếp bộ nhớ - MI (Memory interfaces)
  - Khối thiết bị ngoại vi - IOP (I/O peripherals)
  - Khối kết nối
- Khối Logic lập trình – PL (Programmable Logic)
  - Các khối logic có khả năng tái cấu hình - CLB (Configurable Logic Blocks)
  - Khối RAM
  - Khối xử lý tín hiệu số - DSP48E1
  - Khối XADC (Analog-to-digital converter)

Các khối PS và PL có thể được kết hợp chặt chẽ hoặc không chặt chẽ bằng nhiều giao diện và các tín hiệu khác có tổng cộng hơn 3.000 kết nối. Điều này cho phép tích hợp hiệu quả các bộ tăng tốc phần cứng do người dùng tạo và các chức năng khác trong PL có thể truy cập tới các bộ vi xử lý và cũng có thể truy cập các tài nguyên bộ nhớ trong hệ thống xử lý [18].

2.5.1 Thành phần PS của ZYNQ-7000

Trong PS khối APU bao gồm hai lõi vi xử lý ARM Cortex-A9. Trong đó mỗi lõi đều bao gồm các khối NEON, khối xử lý dữ liệu dấu chấm động (FPU: Floating Point Unit), khối quản lý bộ nhớ (MMU: Memory Management Unit), hai bộ nhớ cache I-cache (Instruction cache) và D-cache (Data cache) cho việc lưu trữ lệnh và dữ liệu. APU cũng bao gồm một bộ nhớ cache L2 và bộ nhớ on-chip (On-Chip Memory: OCM) dùng chung cho cả hai lõi vi xử lý. Ngoài ra APU còn bao gồm các giao diện ngoại vi, giao diện ghép nối bộ nhớ, mạng truyền thông và mạch tạo tín hiệu đồng bộ. Truyền thông giữa PS với các thiết bị ngoại vi bên ngoài có thể thực hiện trực tiếp thông qua khối ghép nối MIO (Multiplexed Input/Output) gồm 54 chân. Ngoài ra có thể sử dụng các I/O là một phần của PL cho các ngoại vi của PS, việc này được thực hiện nhờ kết nối EMIO (Extended MIO). Các kết nối ngoại vi I/O trong PS gồm [18]:

- 2 kết nối SPI: cung cấp một phương thức truyền thông nối tiếp sử dụng 4 dây cho cả truyền và nhận.
- 2 kết nối I2C: cung cấp một phương thức truyền thông nối tiếp sử dụng 2 dây cho cả truyền và nhận.
- 2 kết nối CAN: chuẩn truyền thông nối tiếp được sử dụng phổ biến trong ngành công nghiệp ô-tô.
- 2 kết nối UART: chuẩn truyền thông nối tiếp thường dùng để kết nối thiết bị đầu cuối với máy tính cho mục đích gỡ rối, sửa lỗi.
- 4 bộ kết nối GPIO: bộ ghép nối song song 32-bit.
- 2 kết nối SD: dùng ghép nối với SDCard.
- 2 kết nối USB: tương thích chuẩn USB 2.0.
- 2 kết nối Ethernet: hỗ trợ tốc độ kết nối 10Mbps, 100Mbps và 1Gbps.

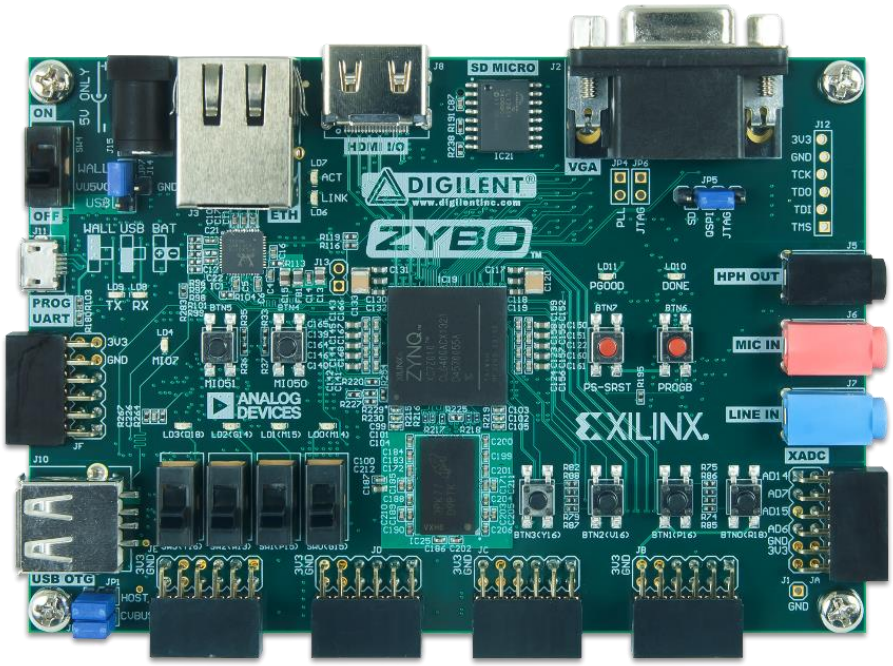
2.5.2 Thành phần PL của ZYNQ-7000

PL trên ZYNQ-7000 bao gồm các khối logic có khả năng tái cấu hình và một số tài nguyên phần cứng chuyên dụng tương thích với chuẩn FPGA Artix-7 và Kintex-7 của Xilinx:

- Các thiết bị dựa trên Artix: Z-7010 và Z-7020
- Các thiết bị dựa trên Kintex: Z-7030 và Z-7054

Cũng giống như các FPGA khác, phần PL của ZYNQ-7000 bao gồm các khối logic cấu hình (CLB), mỗi CLB gồm 2 slice. Mỗi slice chứa 4 LUT, 8 Flip-flop (FF), và một ma trận chuyển mạch (Switch Matrix) có chức năng định tuyến linh hoạt cho việc kết nối giữa các phần tử trong một CLB hoặc từ khối CLB đến các tài nguyên khác trong PL. Ngoài ra còn có khối RAM và DSP [18].

2.6 PHẦN CỨNG ZYBO ZYNQ-7000



Hình 2.22: Phần cứng ZYBO ZYNQ-7000

ZYBO (ZYnq BOard) là một nền tảng phát triển mạch kỹ thuật số và phần mềm nhúng hỗ trợ nhiều tính năng, dễ dàng sử dụng, thuộc họ Z-7010 của dòng Xilinx ZYNQ-7000. Z-7010 dựa trên kiến trúc Xilinx All Programmable System-on-Chip (AP SoC), tích hợp bộ vi xử lý ARM Cortex-A9 lõi kép thuộc dòng Xilinx 7-series FPGA [19].

ZYNQ-7000 Z-7010 AP SoC có các đặc điểm và tính năng như:

- Vi xử lý Cortex-A9 lõi kép với xung nhịp 650MHz.
- Bộ nhớ DDR3 với 8 kênh DMA.
- Bộ điều khiển thiết bị ngoại vi băng thông cao: 1G Ethernet, USB 2.0, SDIO.
- Bộ điều khiển thiết bị ngoại vi băng thông thấp: SPI, UART, CAN, I^2C.
- Logic có thể tái lập trình tương tự như Artix-7 FPGA:
    - 4,400 logic slices, với 4 LUTs 6 ngõ vào và 8 Flip-flop mỗi slice.
    - 240 KB Block RAM.
    - Hai khối quản lý xung Clock với PLL (Phase-locked Loop) và MMCM (Mixed-module Clock Manager).
    - 80 DSP Slices.
    - Xung Clock nội có tần số hơn 450MHz.
    - Bộ chuyển đổi tín hiệu Tương tự - Số (ADC) trên chip.

2.7 PHẦN MỀM XILINX VIVADO DESIGN SUITE 2019.1

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

Vivado Design Suite cung cấp các quá trình tích hợp cấp hệ thống và tập trung trên các thiết kế IP, cung cấp môi trường để cấu hình, thực thi, kiểm tra và tích hợp IP như một module độc lập hoặc trong nội dung của thiết kế cấp hệ thống. Xilinx IP sử dụng các cấu kết nối chuẩn AXI4 để cho phép tích hợp cấp hệ thống nhanh hơn [20].

Các tính năng chính trong quá trình thiết kế:

- Vivado Synthesis
- Vivado Implementation
- Vivado Timing Analysis
- Vivado Power Analysis
- Bitstream Generation

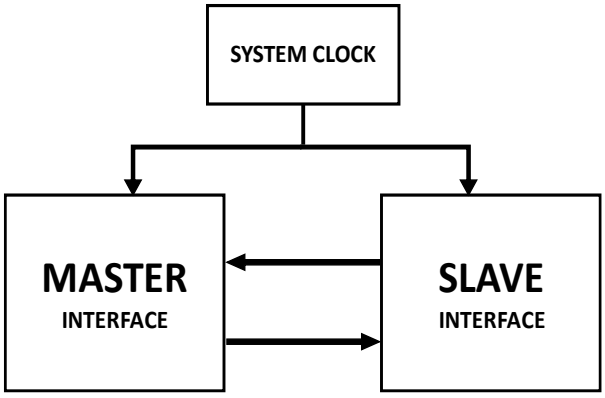
CHƯƠNG 3

THIẾT KẾ HỆ THỐNG

3.1 YÊU CẦU HỆ THỐNG

- Thiết kế hai hệ thống kết nối điểm – điểm sử dụng kiến trúc bus WISHBONE và AMBA AXI.
- Hệ thống thực hiện truyền nhận dữ liệu giữa một giao diện MASTER và một giao diện SLAVE.
- Tần số hoạt động của hệ thống là 100MHz.

3.2 ĐẶC TẢ HỆ THỐNG

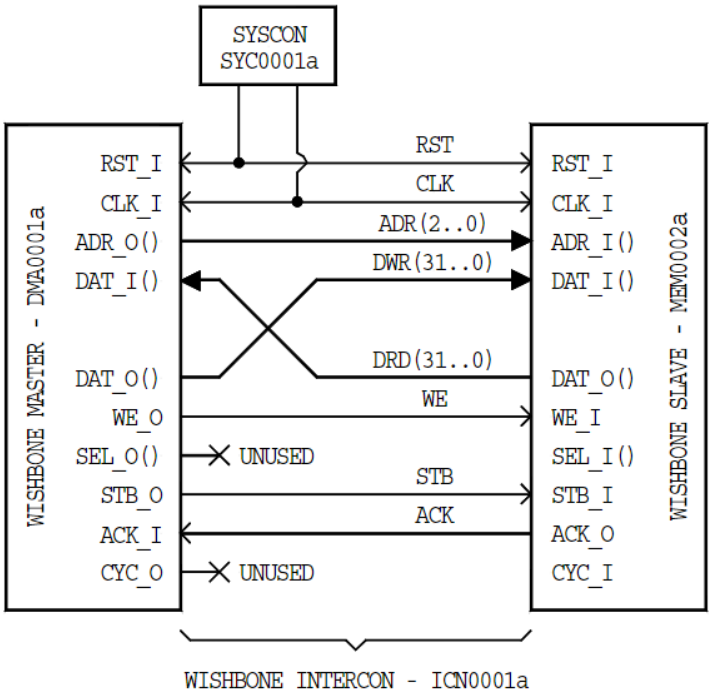


Hình 3.1: Sơ đồ đặc tả hệ thống

Sơ đồ đặc tả hệ thống gồm 3 khối được thể hiện như trong hình 3.1 bên trên, bao gồm: khối SYSTEM CLOCK, khối MASTER và khối SLAVE. Khối MASTER và khối SLAVE kết nối thông qua giao diện điểm - điểm sử dụng kiến trúc bus WISHBONE hoặc AMBA AXI.

- Khối SYSTEM CLOCK cung cấp xung Clock của hệ thống và tín hiệu Reset cho hoạt động của cả hai giao diện MASTER và SLAVE.
- Khối MASTER có chức năng khởi tạo các tín hiệu đầu vào, gửi các yêu cầu thực hiện truyền nhận dữ liệu tới SLAVE.
- Khối SLAVE là một bộ nhớ RAM 8 x 32 bit, có nhiệm vụ lưu trữ dữ liệu, tiếp nhận và phản hồi lại với các tín hiệu yêu cầu từ MASTER và thực hiện yêu cầu đó.

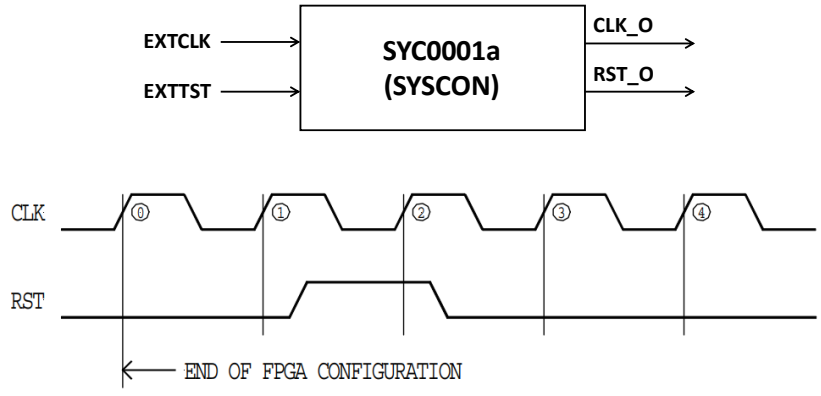
3.3 THIẾT KẾ CHI TIẾT HỆ THỐNG SỬ DỤNG BUS WISHBONE



Hình 3.2: Sơ đồ khối tổng quát hệ thống kết nối sử dụng bus WISHBONE

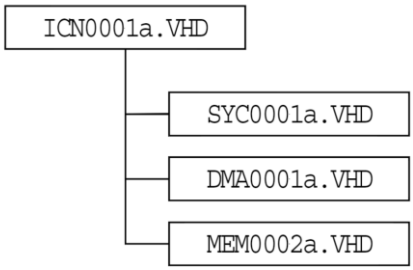
Hình 3.2 mô tả hệ thống kết nối điểm - điểm sử dụng kiến trúc bus WISHBONE. Giao diện kết nối INTERCON bao gồm một khối giao diện MASTER, một giao diện SLAVE và một khối SYSCON.

Khối SYSCON là một bộ điều khiển hệ thống đơn giản với mục đích tạo xung nhịp tương thích với WISHBONE [CLK] và tín hiệu RESET [RST] giúp điều khiển hai khối MASTER và SLAVE hoạt động đồng bộ với cùng một xung nhịp Clock. Ngõ ra xung nhịp được cấp trực tiếp từ tín hiệu bên ngoài được gọi là [EXTCLK].



Hình 3.3: Sơ đồ khối SYSCON và dạng sóng tín hiệu xung CLK và RST

Giao diện MASTER là đơn vị truy cập bộ nhớ trực tiếp DMA. Giao diện SLAVE là bộ nhớ 8 x 32 bit xây dựng dựa trên các thanh ghi dịch. Đối với mục đích mô phỏng, bộ nhớ này có thể được sử dụng bởi tính di động trên tất cả các thiết bị mục tiêu FPGA và ASIC [15]. Sơ đồ phân cấp của các khối trong hệ thống kết nối điểm – điểm của WISHBONE được mô tả như trong hình 3.4 bên dưới.



Hình 3.4: Sơ đồ phân cấp cho mô phỏng hệ thống

Các chân tín hiệu của giao diện MASTER và SLAVE của bus WISHBONE được mô tả lần lượt trong bảng 3.1 và bảng 3.2 dưới đây.

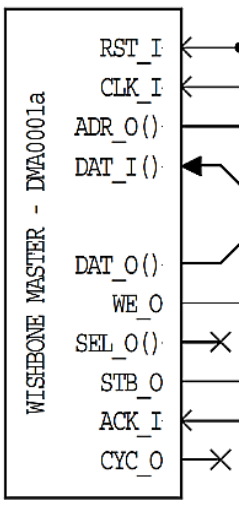
Bảng 3.1: Các tín hiệu sử dụng trong giao diện MASTER của bus WISHBONE

Tên tín hiệu	Độ rộng	Loại	Mô tả chức năng
CLK_I	1	Input	Tín hiệu xung Clock của hệ thống.
RST_I	1	Input	Tín hiệu RESET khởi động lại hệ thống.
STB_O	1	Output	Tín hiệu thông báo một chu kỳ bus truyền dữ liệu hợp lệ.
WE_O	1	Output	Tín hiệu cho phép quá trình ghi (WE = 1). Khi WE = 0 cho phép quá trình đọc.
ADR_O	5	Output	Đường địa chỉ truy xuất.
DAT_I	32	Input	Đường dữ liệu vào (dữ liệu đọc từ SLAVE).
DAT_O	32	Output	Đường dữ liệu ra (dữ liệu ghi từ MASTER).
ACK_I	1	Input	Tín hiệu bắt tay của SLAVE phản hồi lại với tín hiệu [STB] từ MASTER, cho biết SLAVE đã tiếp nhận yêu cầu từ MASTER và thực hiện truyền dữ liệu tại cạnh lên xung Clock kế tiếp.

Bảng 3.2: Các tín hiệu sử dụng trong giao diện SLAVE của bus WISHBONE

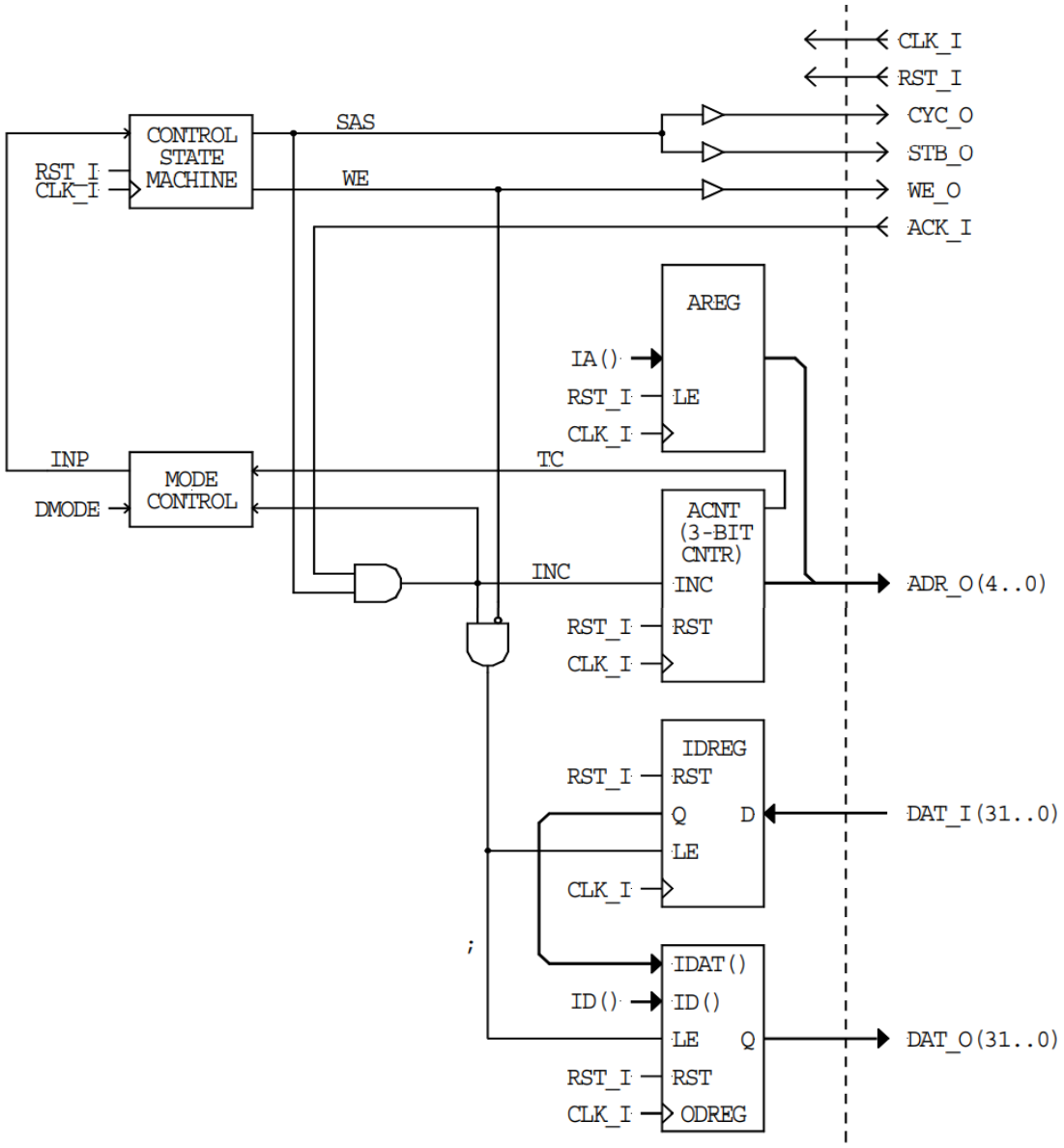
Tên tín hiệu	Độ rộng	Loại	Mô tả chức năng
CLK_I	1	Input	Tín hiệu xung Clock của hệ thống.
RST_I	1	Input	Tín hiệu RESET khởi động lại hệ thống.
STB_I	1	Input	Tín hiệu thông báo một chu kỳ bus truyền dữ liệu hợp lệ.
WE_I	1	Input	Tín hiệu cho phép quá trình ghi (WE = 1). Khi WE = 0 cho phép quá trình đọc.
ADR_I	5	Input	Đường địa chỉ truy xuất.
DAT_I	32	Input	Đường dữ liệu vào (dữ liệu ghi từ MASTER).
DAT_O	32	Output	Đường dữ liệu ra (dữ liệu đọc từ SLAVE).
ACK_O	1	Input	Tín hiệu bắt tay của SLAVE phản hồi lại với tín hiệu [STB] từ MASTER, cho biết SLAVE đã tiếp nhận yêu cầu từ MASTER và thực hiện truyền dữ liệu tại cạnh lên xung Clock kế tiếp.

3.3.1 Thiết kế khối WISHBONE MASTER



Hình 3.5: Sơ đồ khối WISHBONE MASTER

Khối MASTER sử dụng ở đây là một đơn vị DMA 32-bit đơn giản thường sử dụng cho việc mô phỏng và đánh giá. Nó hoạt động như một giao diện WISHBONE MASTER với chức năng chính là khởi tạo chu kỳ truyền nhận dữ liệu với SLAVE theo các chu kỳ ĐỌC/GHI ĐƠN (SINGLE) hoặc các chu kỳ ĐỌC/GHI KHỐI (BLOCK). Loại chu kỳ được chọn bởi ngõ vào tín hiệu [DMODE] [15]. Sơ đồ khối chi tiết của MASTER được thể hiện trong hình 3.6 bên dưới.



Hình 3.6: Sơ đồ khối chi tiết của WISHBONE MASTER

Khối MASTER được thiết kế gồm 6 khối nhỏ hơn gồm AREG (Address Register), ACNT (Address Counter), CSM (CONTROL STATE MACHINE), IDREG (Input Data Register), MODE CONTROL, ODREG (Output Data Register).

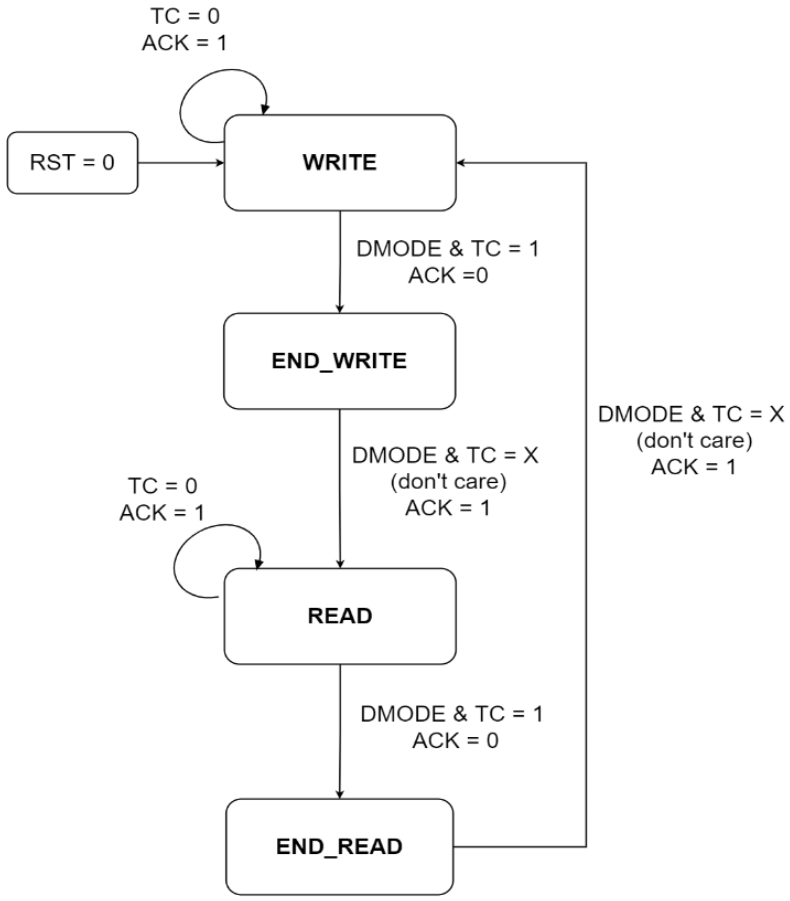
- AREG là khối thanh ghi địa chỉ chiếm 2-bit cao của ngõ ra [ADR_O] để MASTER chọn một SLAVE mục tiêu duy nhất. Ngõ vào của AREG là các tín hiệu [CLK], [RST] và [IA], trong đó [IA] là đường giá trị địa chỉ khởi tạo của SLAVE mục tiêu mà MASTER muốn nhắm đến.
- ACNT là khối đếm từ 0x0 tới 0x7 và ngược lại 0x7 đến 0x0, đếm sau mỗi chu kỳ bus. Khối này chiếm 3-bit thấp của ngõ ra [ADR_O], có chức năng là xác định giai đoạn (phase) truyền dữ liệu hiện tại. Ngõ vào của ACNT là các tín hiệu [CLK], [RST] và [INC]. Khối ACNT hoạt động khi tín hiệu [INC] được khẳng định.
- IDREG là một flip-flop D có chức năng dịch dữ liệu nhận được từ SLAVE tới MASTER. Ngõ vào của IDREG là các tín hiệu [CLK], [RST] và [DAT_I] là ngõ vào dữ liệu đọc lấy từ SLAVE.
- ODREG là một flip-flop D có chức năng lấy dữ liệu từ [ID] là đường giá trị dữ liệu khởi tạo và dịch dữ liệu cần truyền tới ngõ ra của MASTER. Ngõ vào của ODREG là các tín hiệu [CLK], [RST] và [ID].
- MODE CONTROL có ngõ vào là tín hiệu [TC], [INC] và [DMODE] để xác định chu kỳ là ĐỌC/GHI ĐƠN hoặc ĐỌC/GHI KHỐI.
- CSM là khối xử lý các tín hiệu chính trong MASTER có chức năng điều khiển cho bắt đầu một chu kỳ và cho phép các khối trên hoạt động. Ngõ vào của CSM là các tín hiệu [CLK], [RST] và [INP]. Ngõ ra của CSM là [SAS] và [WE].

Khi tín hiệu [RST_I] được khẳng định, DMA đặt lại máy trạng thái điều khiển của nó cùng với tất cả các thanh ghi và bộ đếm liên quan. Hơn nữa, các trạng thái địa chỉ ban đầu [IA] và dữ liệu ban đầu [ID] được chốt. Điều này cung cấp cho người dùng một số quyền kiểm soát đối với địa chỉ đích và các giá trị dữ liệu ghi ban đầu [15].

Nếu ngõ vào tín hiệu [DMODE] bị phủ định, DMA sẽ tạo ra các chu kỳ ĐỌC/GHI ĐƠN. Trong chế độ này, DMA bắt đầu một chu kỳ GHI ĐƠN bắt đầu ở cạnh lên [CLK_I] ngay sau khi ngõ vào đặt lại [RST_I] bị phủ định. Sau khi hoàn thành chu kỳ ghi, DMA bắt đầu một chu kỳ ĐỌC ĐƠN. Các chu kỳ ghi/đọc diễn ra xen kẽ nhau và quá trình lặp lại tương tự [15].

Sau khi hoàn thành mỗi chu kỳ bus, DMA tăng địa chỉ của nó. Hai bit địa chỉ cao nhất do DMA tạo ra được chốt để đáp ứng với việc đặt lại. Điều này cho phép mỗi DMA nhắm mục tiêu một SLAVE duy nhất. Ba bit địa chỉ thấp hơn được tạo ra bởi một bộ đếm đếm từ 0x0 đến 0x7 và lặp lại từ 0x7 đến 0x0 [15].

Nếu ngõ vào [DMODE] được khẳng định, DMA sẽ tạo các chu kỳ ĐỌC/GHI KHỐI. Trong chế độ này, DMA bắt đầu chu kỳ GHI KHỐI với tám pha. Sau khi hoàn thành chu kỳ GHI KHỐI, DMA tạo ra một chu kỳ ĐỌC KHỐI tương tự [10]. Sơ đồ máy trạng thái của quá trình truyền nhận dữ liệu trong hệ thống bus WISHBONE được thể hiện như hình 3.7 bên dưới.



Hình 3.7: Sơ đồ máy trạng thái của hệ thống bus WISHBONE

Máy trạng thái của quá trình truyền nhận dữ liệu của hệ thống bus WISHBONE gồm 4 trạng thái chính là WRITE, END_WRITE, READ và END_READ. Các trạng thái này thay đổi phụ thuộc vào các tín hiệu [DMODE], [TC] và [ACK]. Tín hiệu [DMODE] luôn tích cực mức cao do đây là quá trình ĐỌC/GHI KHỐI.

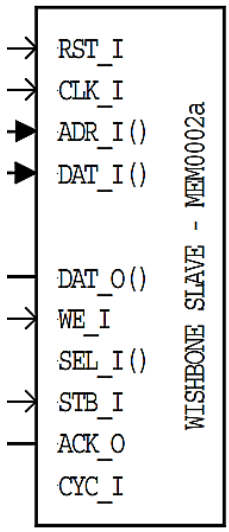
Trạng thái WRITE là trạng thái ghi dữ liệu, trạng thái này kéo dài trong 8 chu kỳ CLK. Tín hiệu [ACK] tích cực ở mức cao khi SLAVE xác nhận và phản hồi yêu cầu truyền dữ liệu từ MASTER. Dữ liệu ghi từ MASTER truyền liên tục tới SLAVE trong 8 pha. Tín hiệu [TC] = 1 khi địa chỉ ở pha cuối cùng ([ADR] = 3'b111). Khi đó [DMODE]&[TC] = 1 và tích cực mức thấp cho tín hiệu [ACK] báo hiệu kết thúc quá trình ghi.

Trạng thái END_WRITE là trạng thái kế tiếp của WRITE, đây có xem là 1 trạng thái nghỉ của hệ thống để bắt đầu chuyển qua trạng thái tiếp theo là READ. Ở trạng thái này sẽ không có dữ liệu được ghi và kết thúc trạng thái này sẽ lập tức chuyển qua trạng thái tiếp theo.

Trạng thái READ là trạng thái đọc dữ liệu, trạng thái này kéo dài trong 8 chu kỳ CLK. SLAVE tích cực tín hiệu [ACK] ở mức cao để phản hồi lại yêu cầu đọc dữ liệu từ MASTER. Dữ liệu đọc từ SLAVE được truyền liên tục tới MASTER trong 8 pha. Tín hiệu [TC] = 1 khi địa chỉ ở pha cuối cùng ([ADR] = 3'b111). Khi đó [DMODE]&[TC] = 1 và tích cực mức thấp cho tín hiệu [ACK] báo hiệu kết thúc quá trình đọc.

Cuối cùng là ở trạng thái END_READ, tại đây là trạng thái chờ trước khi chuyển sang trạng thái tiếp theo WRITE và thực hiện tương tự như mô tả trạng thái ở trên.

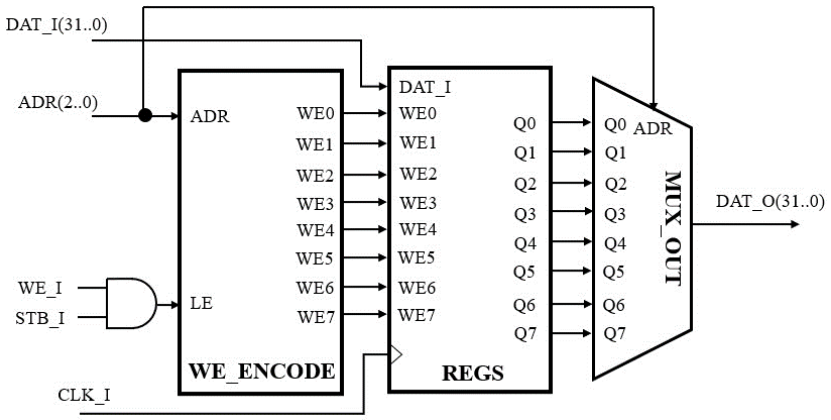
3.3.2 Thiết kế khối WISHBONE SLAVE



Hình 3.8: Sơ đồ khối WISHBONE SLAVE

Sơ đồ khối của WISHBONE SLAVE được mô tả như trong hình 3.8 bên trên. Khối SLAVE được sử dụng là một bộ nhớ dựa trên thanh ghi có dạng flip-flop D dùng cho việc kiểm tra mô phỏng các kết nối WISHBONE. Mặc dù bộ nhớ này có thể được tổng hợp trong phần cứng, nhưng nó thường không được khuyến khích vì nó sẽ tổng hợp dưới dạng các flip-flop rời rạc, bộ ghép kênh và các logic khác. Tuy nhiên, nếu tốc độ chậm hơn và kích thước lớn hơn có thể chấp nhận được thì hoàn toàn có thể sử dụng bộ nhớ này [15].

Bộ nhớ này được thiết kế để sử dụng cho mô phỏng di động của các kết nối WISHBONE. Nó có thể di động trên tất cả các thiết bị mục tiêu FPGA và ASIC. Bộ nhớ này hoạt động như một RAM đồng bộ tương thích FASM (FPGA and ASIC Subset Model) [15].



Hình 3.9: Sơ đồ khối chi tiết của WISHBONE SLAVE

Khối SLAVE được thiết kế từ 3 khối nhỏ hơn là WE_ENCODE, REGS và MUX_OUT:

- WE_ENCODE là khối gồm các chân [WE0] tới [WE7] để lựa chọn đường dữ liệu hoạt động.
- REGS là khối thanh ghi dịch khi chân cho phép [WE0] đến [WE7] được khẳng định thì các thanh ghi tương ứng được phép hoạt động.
- MUX_OUT là khối lựa chọn ngõ ra từ tín hiệu [ADR_I], [ADR_I] có giá trị từ 3'b111 đến 3'b000 cho biết đang ở giai đoạn nào của chu kỳ truyền nhận dữ liệu.

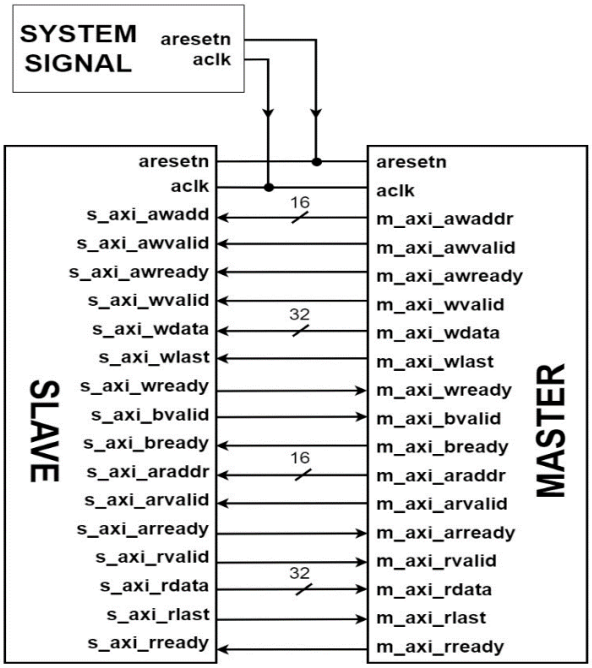
Khối SLAVE xác nhận chu kỳ truyền dữ liệu khi nhận tín hiệu STB và giá trị địa chỉ [ADR_I] được gửi từ MASTER. Tín hiệu [WE_I] xác định quá trình đọc (WE_I = 0) hoặc ghi (WE_I = 1). Dữ liệu truyền từ MASTER tới SLAVE được lưu lại trong các

thanh ghi dịch ở quá trình ghi và truyền tới MASTER ở quá trình đọc, sau đó bắt đầu một giai đoạn truyền nhận dữ liệu khác.

Ngõ vào [DAT_I] và ngõ ra [Q0] đến [Q7], khi dữ liệu được truyền từ MASTER ở giai đoạn từ 0 tới 7 của chu kỳ thì SLAVE sẽ nhận dữ liệu ở đường tương ứng, ở ngõ ra sẽ có một khối MUX để chọn ngõ ra ứng với địa chỉ hiện tại.

3.4 THIẾT KẾ CHI TIẾT HỆ THỐNG SỬ DỤNG BUS AMBA AXI

Sơ đồ khối tổng quát của hệ thống được mô tả trong hình dưới đây:



Hình 3.10: Sơ đồ tổng quát hệ thống kết nối sử dụng bus AMBA AXI

Hình 3.10 trên mô tả hệ thống kết nối điểm – điểm sử dụng kiến trúc bus AMBA AXI. Nó bao gồm một giao diện AXI MASTER, một giao diện AXI SLAVE và SYSTEM SIGNAL.

Giao diện AXI MASTER là khối được xây dựng cho việc mô phỏng và kiểm tra, đây không phải là một thiết kế RTL mà chỉ là một mô hình hành vi. AXI MASTER mô phỏng lại các hành vi chính của một MASTER thực hiện như khởi tạo các tín hiệu để kết nối, yêu cầu các chu kỳ truyền nhận dữ liệu với SLAVE, thực hiện đọc/ghi dữ liệu. Giao diện AXI SLAVE là một khối AXI RAM, có chức năng lưu dữ liệu ghi được gửi từ MASTER, tiếp nhận và phản hồi lại các tín hiệu trong quá trình kết nối truyền nhận dữ liệu với MASTER. SYSTEM SIGNAL là khối cấp xung Clock từ hệ thống và tạo tín hiệu Reset cho hoạt động của các giao diện AXI.

Mô tả các chân tín hiệu sử dụng trong giao diện kết nối của bus AMBA AXI được thể hiện lần lượt trong bảng 3.3 và bảng 3.4 dưới đây.

Bảng 3.3: Các tín hiệu sử dụng trong giao diện MASTER của bus AMBA AXI

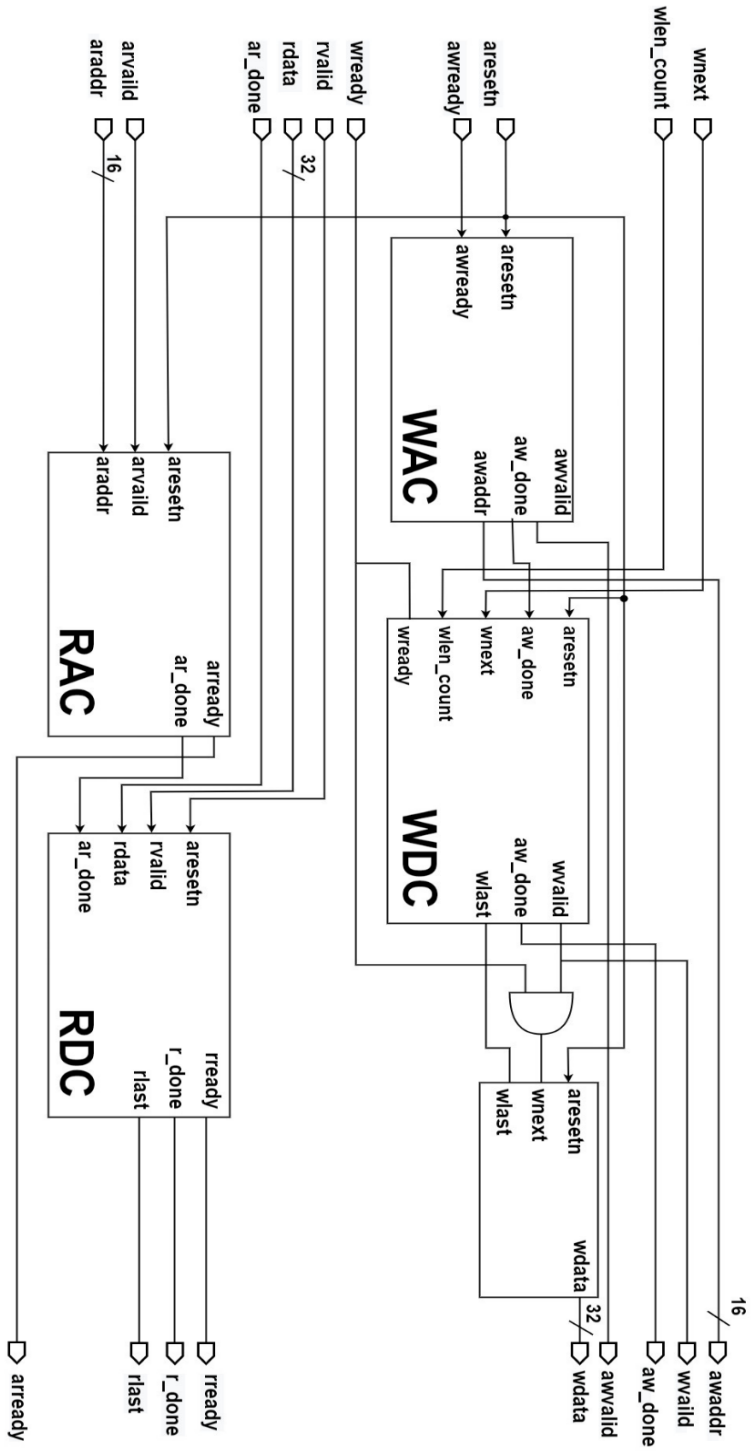
Tên tín hiệu	Độ rộng	Loại	Mô tả chức năng
aclk	1	Input	Xung Clock của toàn hệ thống.
aresetn	1	Input	Tín hiệu RESET tích cực mức THẤP khởi tạo lại cho toàn hệ thống.
m_axi_awaddr	16	Output	Địa chỉ ghi.
m_axi_awvalid	1	Output	Báo hiệu địa chỉ ghi hợp lệ.
m_axi_awready	1	Input	Báo hiệu SLAVE sẵn sàng để chấp nhận địa chỉ ghi và các tín hiệu điều khiển liên quan.
m_axi_wvalid	1	Output	Cho biết dữ liệu ghi hợp lệ.
m_axi_wdata	32	Output	Dữ liệu ghi.
m_axi_wlast	1	Output	Báo hiệu transfer cuối của của burst ghi.
m_axi_wready	1	Input	Báo hiệu SLAVE sẵn sàng nhận dữ liệu ghi.
m_axi_bvalid	1	Input	Thông báo một phản hồi ghi hợp lệ.
m_axi_bready	1	Output	Báo hiệu MASTER sẵn sàng nhận phản hồi ghi.
m_axi_araddr	16	Output	Địa chỉ đọc.
m_axi_arvalid	1	Output	Báo hiệu địa chỉ đọc hợp lệ.
m_axi_arready	1	Input	Cho biết SLAVE sẵn sàng nhận một địa chỉ đọc và các tín hiệu điều khiển liên quan.
m_axi_rvalid	1	Input	Cho biết dữ liệu đọc hợp lệ.
m_axi_rdata	32	Input	Dữ liệu đọc.
m_axi_rlast	1	Input	Báo hiệu transfer cuối cùng trong burst đọc.
m_axi_rready	1	Output	Cho biết MASTER sẵn sàng nhận dữ liệu đọc.

Bảng 3.4: Các tín hiệu sử dụng trong giao diện SLAVE của bus AMBA AXI

Tên tín hiệu	Độ rộng	Loại	Mô tả chức năng
aclk	1	Input	Xung C của toàn hệ thống.
aresetn	1	Input	Tín hiệu RESET tích cực mức THẤP khởi tạo lại cho toàn hệ thống.
s_axi_awaddr	16	Input	Địa chỉ ghi.
s_axi_awvalid	1	Input	Báo hiệu địa chỉ ghi hợp lệ.
s_axi_awready	1	Output	Báo hiệu SLAVE sẵn sàng để chấp nhận địa chỉ ghi và các tín hiệu điều khiển liên quan.
s_axi_wvalid	1	Input	Cho biết dữ liệu ghi hợp lệ.
s_axi_wdata	32	Input	Dữ liệu ghi.
s_axi_wlast	1	Input	Báo hiệu transfer cuối của của burst ghi.
s_axi_wready	1	Output	Báo hiệu SLAVE sẵn sàng nhận dữ liệu ghi.
s_axi_bvalid	1	Output	Thông báo một phản hồi ghi hợp lệ.
s_axi_bready	1	Input	Báo hiệu MASTER sẵn sàng nhận phản hồi ghi.
s_axi_araddr	16	Input	Địa chỉ đọc.
s_axi_arvalid	1	Input	Báo hiệu địa chỉ đọc hợp lệ.
s_axi_arready	1	Output	Cho biết SLAVE sẵn sàng nhận một địa chỉ đọc và các tín hiệu điều khiển liên quan.
s_axi_rvalid	1	Output	Cho biết dữ liệu đọc hợp lệ.
s_axi_rdata	32	Output	Dữ liệu đọc.
s_axi_rlast	1	Output	Báo hiệu transfer cuối cùng trong burst đọc.
s_axi_rready	1	Input	Cho biết MASTER sẵn sàng nhận dữ liệu đọc.

3.4.1 Thiết kế khối AXI MASTER

Khối AXI MASTER được sử dụng là một khối mô hình hành vi được xây dựng cho việc mô phỏng và kiểm tra. Khối MASTER thực hiện khởi tạo các tín hiệu đầu vào, các yêu cầu kết nối và truyền nhận dữ liệu với khối SLAVE. Sơ đồ khối chi tiết của khối AXI MASTER được thể hiện như hình 3.11 bên dưới.



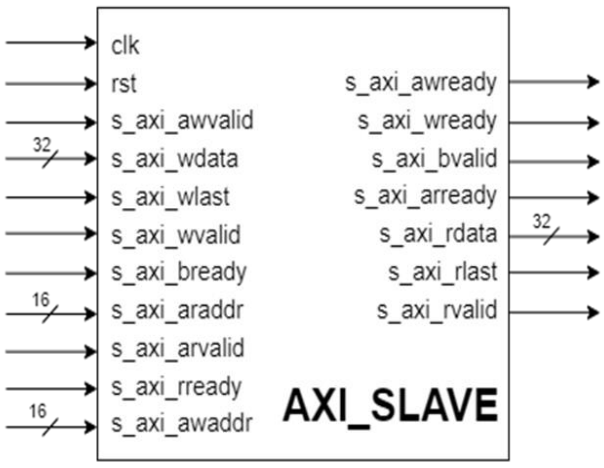
Hình 3.11: Sơ đồ khối chi tiết của AXI MASTER

Khối AXI MASTER được thiết kế gồm 5 khối như sau: Khối WAC (Write Address Channel), khối WDC (Write Data Channel), khối WRC (Write Response Channel), khối RAC (Read Address Channel) và khối RDRC (Read Data & Response Channel).

- Khối WAC (Write Address Channel): có chức năng yêu cầu thông tin địa chỉ cho quá trình truyền dữ liệu. Giá trị địa chỉ sẽ tăng lên mỗi khi nhận một giao dịch địa chỉ.
- Khối WDC (Write Data Channel): có chức năng truyền dữ liệu ghi vào địa chỉ ghi. Lượng dữ liệu được truyền tùy thuộc vào cấu hình của AXI SLAVE.
- Khối WRC (Write Response Channel): có chức năng truyền thông tin phản hồi của quá trình ghi. Khi quá trình ghi dữ liệu kết thúc sẽ có tín hiệu phản hồi để thông báo quá trình ghi dữ liệu hợp lệ hoàn thành và bắt đầu tiếp nhận một quá trình ghi dữ liệu mới.
- Khối RAC (Read Address Channel): có chức năng tiếp nhận và truyền thông tin địa chỉ của quá trình đọc.
- Khối RDRC (Read Data & Response Channel): có chức năng truyền dữ liệu đọc và thông tin phản hồi của một quá trình đọc.

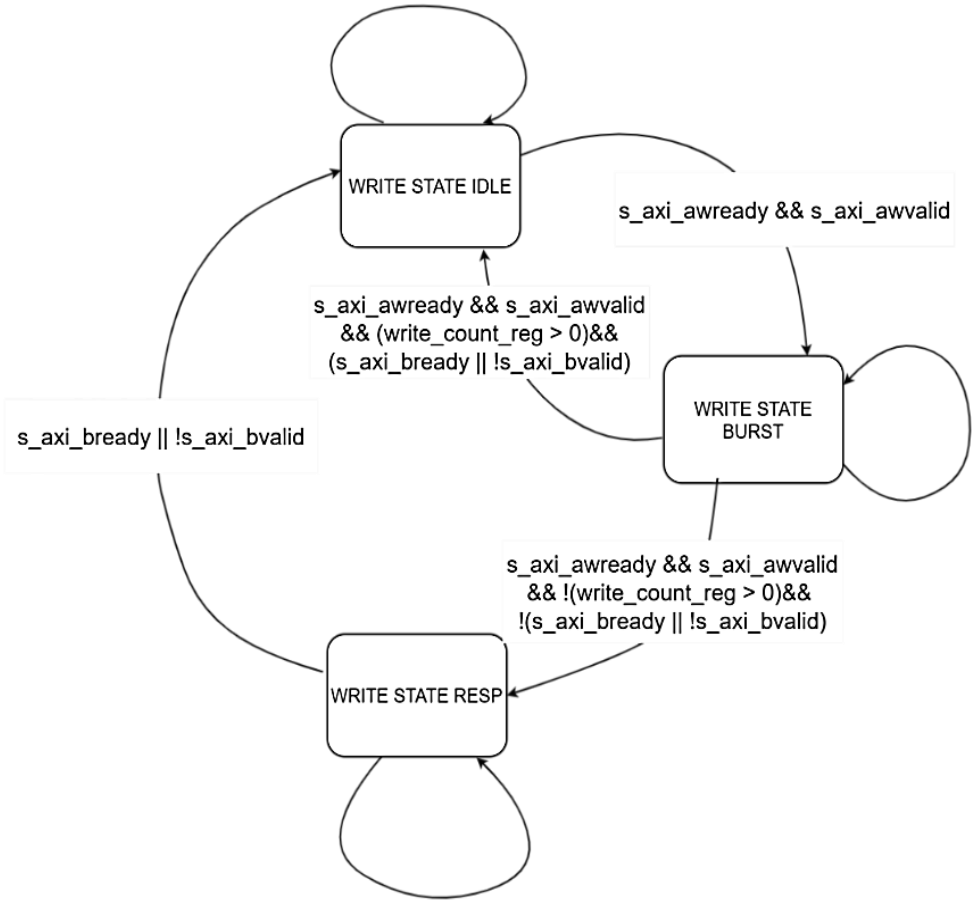
3.4.2 Thiết kế khối AXI SLAVE

Khối AXI SLAVE được sử dụng là một AXI RAM, đóng vai trò là một bộ nhớ có chức năng lưu dữ liệu. MASTER truyền các dữ liệu trong quá trình ghi tới SLAVE và được lưu lại trong các ô nhớ, sau đó các dữ liệu này sẽ được truyền lại về cho MASTER trong quá trình đọc. Sơ đồ khối của AXI SLAVE được thể hiện như hình 3.12 bên dưới.



Hình 3.12: Sơ đồ khối AXI SLAVE

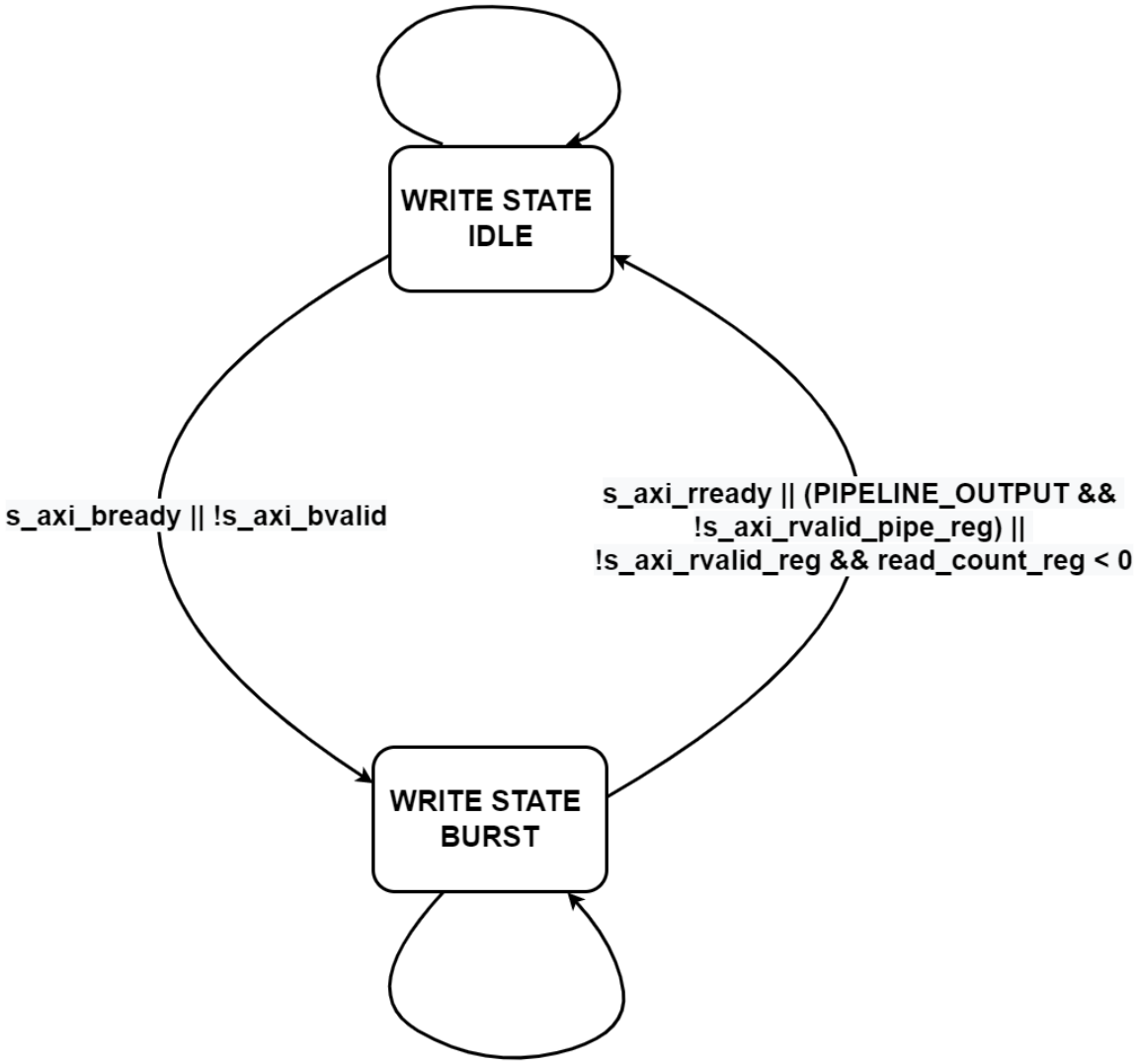
Khối AXI SLAVE có hai hoạt động chính là đọc và ghi. Đối với giao thức AXI, hoạt động đọc và ghi có thể diễn ra đồng thời miễn là không cùng trên một địa chỉ. Hoạt động đọc và ghi của AXI SLAVE được mô tả theo sơ đồ máy trạng thái lần lượt như trong hình 3.13 và hình 3.14 bên dưới.



Hình 3.13: Sơ đồ máy trạng thái quá trình ghi của AXI RAM

Hoạt động ghi của AXI RAM gồm 3 trạng thái chính: WRITE_STATE_IDLE, WRITE_STATE_BURST và WRITE_STATE_RESP.

- WRITE_STATE_IDLE: đây là trạng thái rỗi và sẽ chuyển sang trạng thái WRITE_STATE_BURST khi cặp tín hiệu awready - awvalid được tích cực mức cao.
- WRITE_STATE_BURST: đây là trạng thái bắt đầu quá trình ghi dữ liệu và dữ liệu sẽ được truyền liên tục vào các ô nhớ cho đến khi hết toàn bộ dữ liệu ghi và sẽ chuyển sang trạng thái WRITE_STATE_RESP.
- WRITE_STATE_RESP: đây là trạng thái thực hiện phản hồi của quá trình ghi. Nếu dữ liệu ghi trong quá trình ghi không hợp lệ sẽ chuyển lại về trạng thái WRITE_STATE_IDLE.



Hình 3.14: Sơ đồ máy trạng thái máy quá trình đọc của AXI RAM

Hoạt động đọc của AXI RAM gồm 2 trạng thái chính: READ_STATE_IDLE và READ_STATE_BURST.

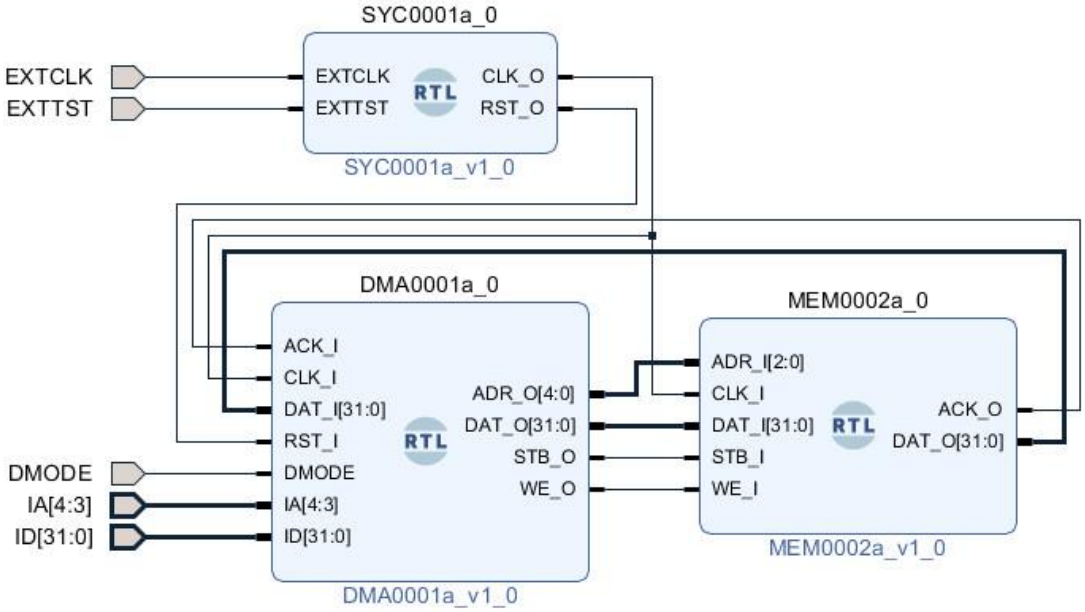
- READ_STATE_IDLE: đây là trạng thái rỗi và sẽ chuyển sang trạng thái READ_STATE_BURST khi cặp tín hiệu arready - arvalid được tích cực mức cao.
- READ_STATE_BURST: đây là trạng thái bắt đầu quá trình đọc dữ liệu. Dữ liệu đọc sẽ được truyền liên tục về MASTER cho đến khi hết toàn bộ dữ liệu đọc và sẽ chuyển sang trạng thái READ_STATE_IDLE.

CHƯƠNG 4

MÔ PHỎNG VÀ KẾT QUẢ

4.1 SƠ ĐỒ HỆ THỐNG TRÊN PHẦN MỀM VIVADO 2019.1

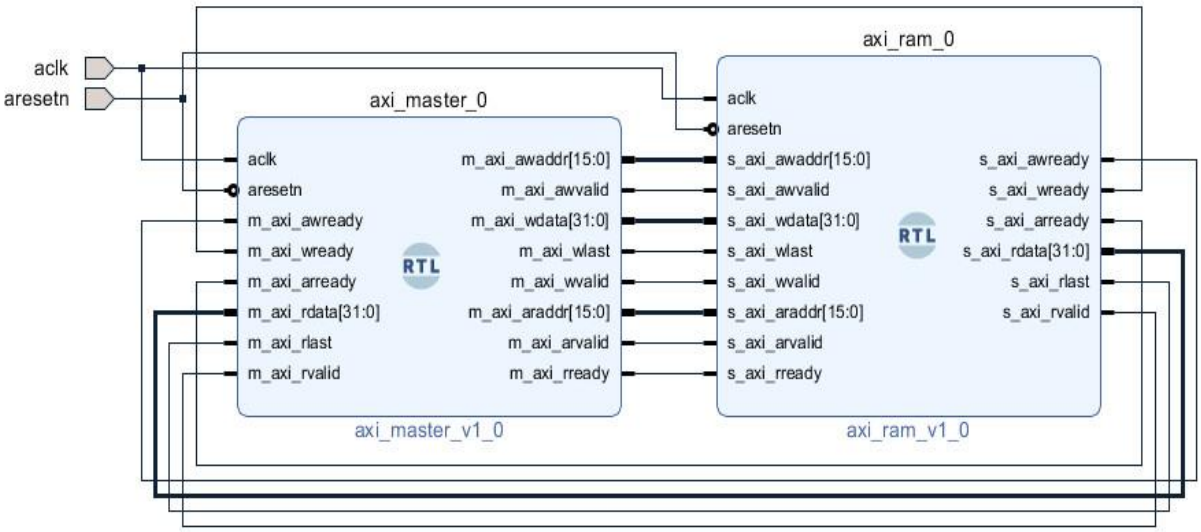
Sơ đồ toàn hệ thống sử dụng bus WISHBONE và AMBA AXI được thể hiện lần lượt như trong hình 4.1 và hình 4.2 bên dưới:



Hình 4.1: Sơ đồ hệ thống sử dụng bus WISHBONE trên phần mềm Vivado 2019.1

Hình 4.1 mô tả về sơ đồ hệ thống của bus WISHBONE, bao gồm 3 khối chính là khối SYSCON (SYC0001a), MASTER (DMA0001a) và SLAVE (MEM0002a). Khối SYSCON cung cấp xung nhịp Clock [EXTCLK] và tín hiệu Reset cho hoạt động của hệ thống, kết nối với các chân [CLK_I] và [RST_I] của khối MASTER và khối SLAVE.

Khối MASTER gửi yêu cầu truyền nhận dữ liệu với SLAVE thông qua tín hiệu [STB_O], khối SLAVE tiếp nhận yêu cầu với tín hiệu [STB_I]. Địa chỉ khởi tạo tại [IA] được MASTER truyền tới SLAVE mục tiêu thông qua ngõ ra [ADR_O], SLAVE tiếp nhận địa chỉ tại ngõ vào [ADR_I]. Tín hiệu [WE_O] và [WE_I] cho phép một quá trình đọc hoặc ghi dữ liệu. Tín hiệu [DMODE] cho biết quá trình ĐỌC/GHI ĐƠN hoặc ĐỌC/GHI KHỐI. SLAVE xác nhận thông tin và gửi tín hiệu phản hồi [ACK_O] tới [ACK_I] của MASTER. Dữ liệu ghi được MASTER truyền từ ngõ ra [DAT_O] đến ngõ vào [DAT_I] của SLAVE.



Hình 4.2: Sơ đồ hệ thống sử dụng bus AMBA AXI trên phần mềm Vivado 2019.1

Hình 4.2 mô tả về sơ đồ hệ thống của bus AMBA AXI, bao gồm 2 khối chính đó là khối MASTER (axi_master) và SLAVE (axi_ram). Xung nhịp Clock cung cấp cho hoạt động của hệ thống kết nối với chân [aclk] của khối MASTER và SLAVE. Tín hiệu [aresetn] là tín hiệu Reset cho toàn hệ thống. Hoạt động truyền nhận dữ liệu của bus AXI được thực hiện trên các kênh độc lập nhau.

Quá trình ghi của AMBA AXI diễn ra trên 3 kênh: kênh Địa Chỉ Ghi, kênh Dữ Liệu Ghi và kênh Phản Hồi Ghi. Kênh Địa Chỉ Ghi bao gồm: [axi_awaddr] truyền địa chỉ ghi, cặp tín hiệu bắt tay [axi_awvalid] và [axi_awready] thông báo địa chỉ ghi hợp lệ. Kênh Dữ Liệu Ghi bao gồm: cặp tín hiệu bắt tay [axi_wvalid] và [axi_wready] cho biết có dữ liệu ghi hợp lệ và SLAVE sẵn sàng nhận dữ liệu ghi từ MASTER, dữ liệu ghi được truyền trên đường [axi_wdata]. Tín hiệu [axi_wlast] báo hiệu khối dữ liệu cuối được ghi. Kênh Phản Hồi Ghi bao gồm: cặp tín hiệu bắt tay [axi_bvalid] và [axi_bready] phản hồi một quá trình ghi hợp lệ và sẵn sàng nhận một quá trình ghi mới.

Quá trình đọc của AMBA AXI diễn ra trên 2 kênh: kênh Địa Chỉ Đọc và kênh Dữ Liệu Đọc. Kênh Địa Chỉ Đọc bao gồm: [axi_araddr] truyền địa chỉ đọc, cặp tín hiệu bắt tay [axi_arvalid] và [axi_arready] thông báo địa chỉ đọc hợp lệ. Kênh Dữ Liệu Đọc bao gồm: cặp tín hiệu bắt tay [axi_rvalid] và [axi_rready] cho biết có dữ liệu đọc hợp lệ và MASTER sẵn sàng nhận dữ liệu đọc từ SLAVE, dữ liệu đọc được truyền trên đường [axi_rdata]. Tín hiệu [axi_rlast] báo hiệu khối dữ liệu cuối được đọc.

4.2 THÔNG SỐ TÀI NGUYÊN HỆ THỐNG

Thông số tài nguyên và công suất tiêu thụ trong hệ thống bus WISHBONE và AMBA AXI sử dụng cho việc tổng hợp trên phần cứng Zybo ZYNQ-7000 SoC mô phỏng sử dụng phần mềm Xilinx Vivado 2019.1, với tần số hoạt động là 100MHz (chu kỳ 10 ns) và điện áp là 3.3V, được mô tả lần lượt như trong bảng 4.1, bảng 4.2 và bảng 4.3 dưới đây.

Bảng 4.1: Thông số tài nguyên hệ thống sử dụng bus WISHBONE và AMBA AXI

Có sẵn [20] / Sử dụng	**LUT (17600)**	**FF (35200)**	**BRAM (60)**	**IO (100)**
WISHBONE	38 *(0.22%)*	34 *(0.10%)*	0 *(0%)*	72 *(72.0%)*
AMBA AXI	85 *(0.48%)*	131 *(0.37%)*	16 *(26.7%)*	98 *(98.0%)*

Bảng 4.2: Công suất tiêu thụ của hệ thống sử dụng bus WISHBONE

Thành phần		**Công suất (W)**	**Tỷ lệ**
Công suất động	Signals	0.001	4%
	Clock	0.001	14%
	Logic	<0.001	4%
	IO	0.006	78%
	Tổng	**0.008**	**8%**
Công suất tĩnh		**0.092**	**92%**
Tổng		**0.99**	**100%**

Bảng 4.3: Công suất tiêu thụ của hệ thống sử dụng bus AMBA AXI

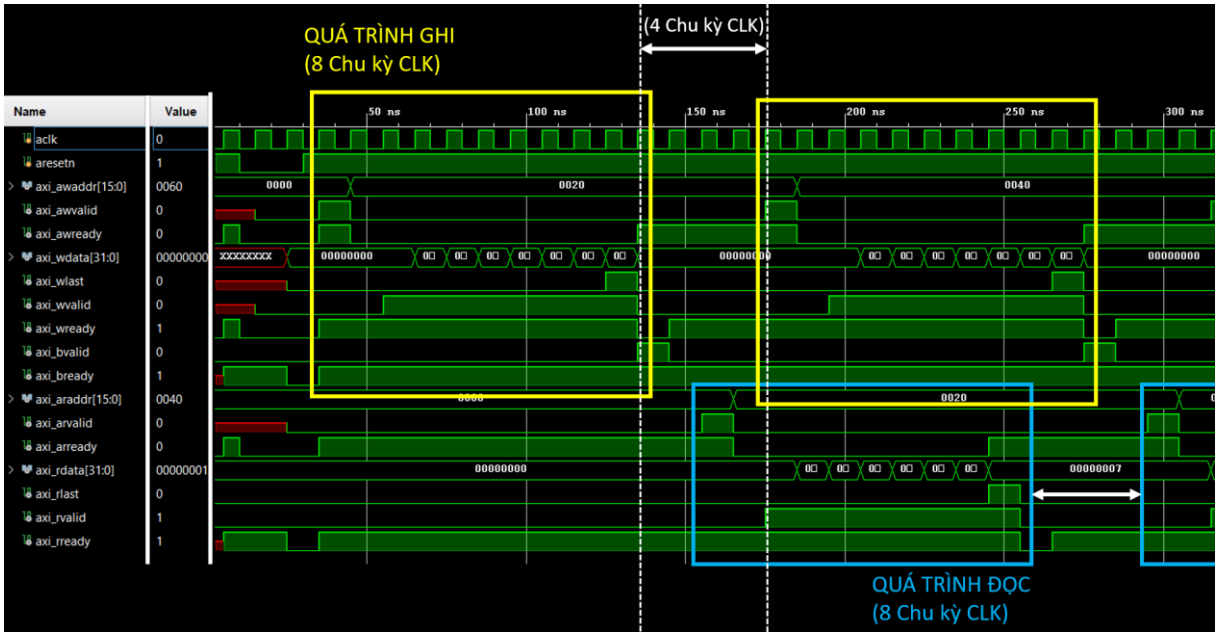
Thành phần		**Công suất (W)**	**Tỷ lệ**
Công suất động	Signals	0.001	2%
	Clock	0.002	3%
	Logic	<0.001	1%
	IO	0.027	38%
	BRAM	0.039	56%
	Tổng	**0.070**	**43%**
Công suất tĩnh		**0.093**	**57%**
Tổng		**0.163**	**100%**

Qua bảng thống kê thông số tài nguyên sử dụng ở trên, có thể thấy hệ thống bus WISHBONE chỉ sử dụng 38 LUT, 34 FF và 72 I/O khi tổng hợp trên phần cứng Zybo. Trong khi đó, hệ thống sử dụng bus AMBA AXI sử dụng 85 LUT, 131 FF, 16 BRAM và 98 I/O, nhiều hơn so với hệ thống bus WISHBONE. Có thể thấy hệ thống sử dụng bus WISHBONE yêu cầu rất ít tài nguyên để triển khai toàn bộ hệ thống và có thể đáp ứng được tài nguyên có sẵn trong hầu hết các thiết bị FPGA và ASIC. Hệ thống sử dụng bus AMBA AXI yêu cầu nhiều tín hiệu trong giao diện kết nối hơn, do đó cũng sử dụng nhiều tài nguyên hơn so với hệ thống WISHBONE. Công suất tiêu thụ của hệ thống AMBA AXI cũng lớn hơn so với công suất tiêu thụ của hệ thống WISHBONE.

4.3 KẾT QUẢ MÔ PHỎNG SỬ DỤNG PHẦN MỀM XILINX VIVADO 2019.1

4.3.1 Kết quả mô phỏng của hệ thống sử dụng kiến trúc bus AMBA AXI

Kết quả dạng sóng mô phỏng cho hoạt động của hệ thống kết nối điểm – điểm sử dụng kiến trúc bus AMBA AXI với chu kỳ xung CLK là 10 ns (tần số 100MHz) được thể hiện trong hình 4.3 bên dưới.



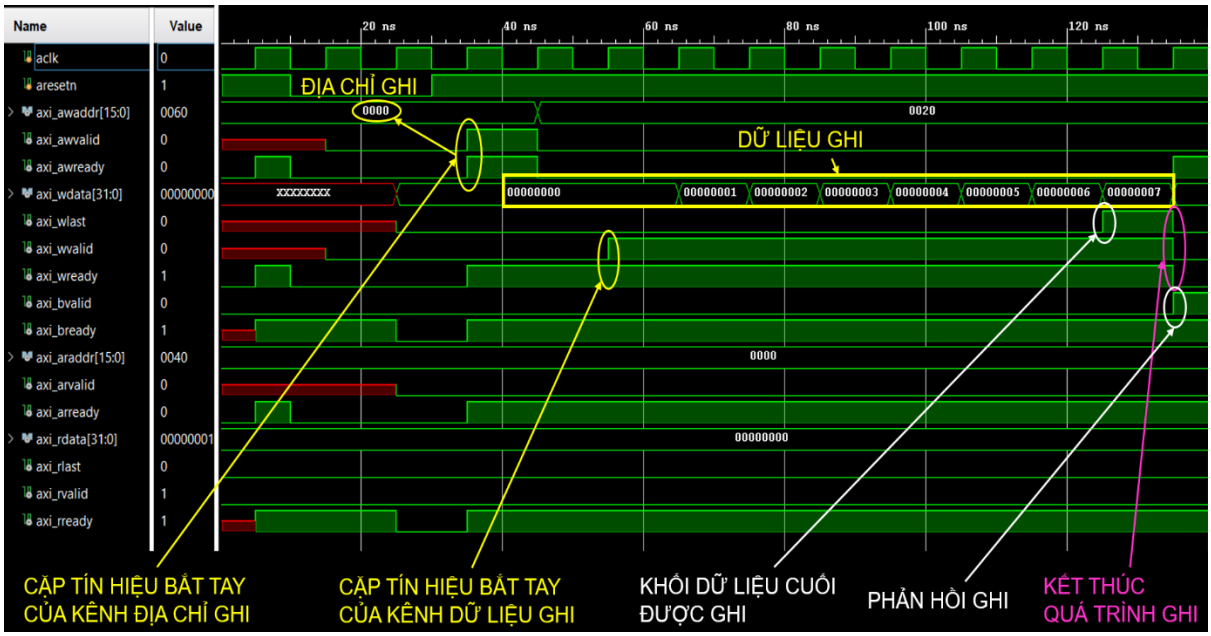
Hình 4.3: Kết quả mô phỏng hoạt động của bus AMBA AXI

Hệ thống bus AMBA AXI thực hiện quá trình đọc và ghi dữ liệu theo cơ chế truyền liên tục (burst). Phía MASTER cấp phát địa chỉ đầu tiên của giao dịch với [axi_awaddr] trong quá trình ghi hay [axi_araddr] trong quá trình đọc.

Các tín hiệu [axi_awready] và [axi_awvalid], [axi_arready] và [axi_arvalid] được tích cực mức cao để báo hiệu sẵn sàng nhận địa chỉ và thông báo nhận thông tin địa chỉ hợp lệ. Tín hiệu [axi_bready] tích cực mức cao cho biết MASTER sẵn sàng nhận phản hồi ghi. Các tín hiệu [axi_wvalid] và [axi_wready], [axi_rvalid] và [axi_rready] tích cực mức cao cho biết kênh sẵn sàng nhận các dữ liệu đọc/ghi và thông báo dữ liệu truyền hợp lệ. Dữ liệu ghi được truyền trên [axi_wdata] và dữ liệu đọc được truyền trên [axi_rdata]. Tín hiệu [axi_wlast] hay [axi_rlast] tích cực mức cao báo hiệu khối dữ liệu cuối được truyền.

Quá trình đọc và ghi của bus AXI diễn ra trong 8 chu kỳ xung Clock cho mỗi quá trình. Do AXI sử dụng các kênh độc lập cho quá trình đọc và ghi, vì vậy quá trình đọc và ghi có thể diễn ra cùng thời điểm, nhưng không cùng trên một địa chỉ.

4.3.1.1 Hoạt động ghi



Hình 4.4: Dạng sóng mô phỏng hoạt động ghi của bus AMBA AXI

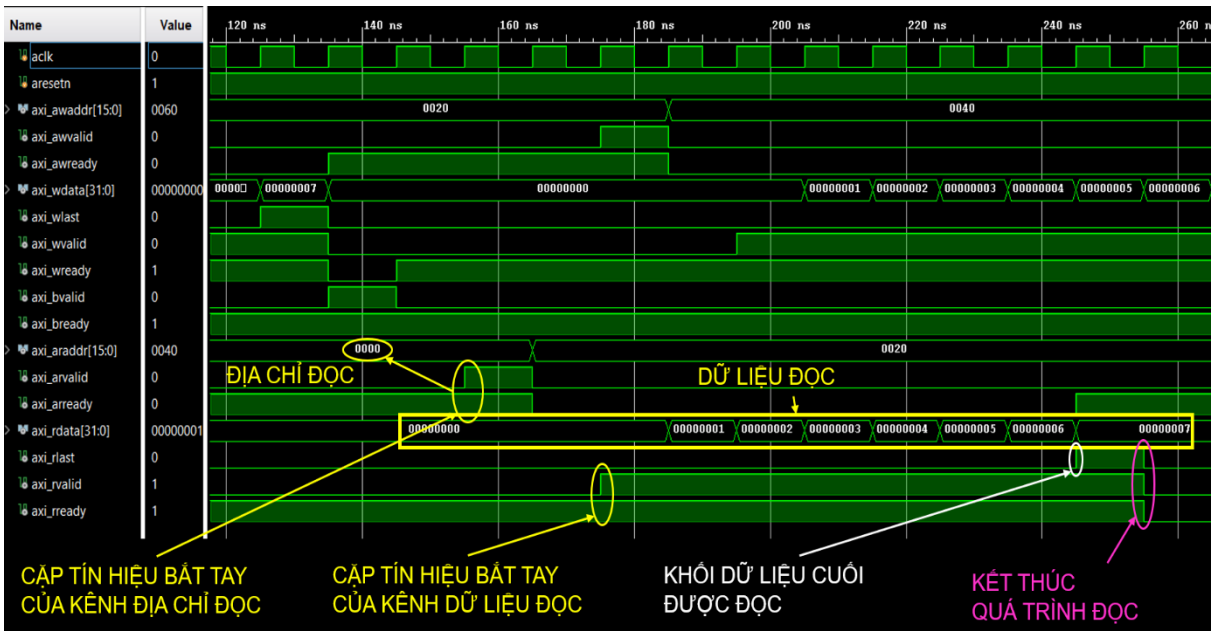
Quá trình ghi của AMBA AXI bao gồm 3 giai đoạn: cấp địa chỉ ghi, ghi dữ liệu và phản hồi ghi. AXI MASTER cấp địa chỉ đầu tiên của quá trình ghi [axi_awaddr] = 16'h0000. Tín hiệu [axi_awready] tích cực mức cao cho biết kênh sẵn sàng nhận địa chỉ ghi. Tín hiệu [axi_awvalid] tích cực mức cao báo hiệu địa chỉ ghi hợp lệ.

Sau khi cặp tín hiệu [axi_awready] và [axi_awvalid] tích cực, tín hiệu [axi_wready] tích cực cho biết SLAVE sẵn sàng nhận dữ liệu ghi và tín hiệu này tích cực trong suốt quá trình ghi dữ liệu. Tín hiệu [axi_wvalid] cho biết có dữ liệu ghi hợp lệ và SLAVE sẵn sàng nhận dữ liệu ghi.

Dữ liệu ghi được truyền liên tục trên đường dữ liệu ghi [axi_wdata] trong 8 chu kỳ xung CLK, mỗi chu kỳ ghi một khối dữ liệu có giá trị từ 32'h00000000, 32'h00000001, 32'h00000002 đến 32'h00000007. Tín hiệu [axi_wlast] tích cực mức cao báo hiệu khối dữ liệu cuối cùng được truyền của quá trình ghi. Sau cùng là cặp tín hiệu [axi_wvalid] và [axi_wready] tích cực mức thấp cho biết kết thúc quá trình ghi.

Cặp tín hiệu [axi_bvalid] và [axi_bready] tích cực mức cao cho biết một quá trình ghi đã hoàn tất và sẵn sàng tiếp nhận yêu cầu mới. Quá trình ghi diễn ra tương tự cho các địa chỉ tiếp theo như 16'h0020, 16'h0040.

4.3.1.2 Hoạt động đọc



Hình 4.5: Dạng sóng mô phỏng hoạt động đọc của bus AMBA AXI

Quá trình đọc của AMBA AXI bao gồm 2 giai đoạn: cấp địa chỉ đọc và đọc dữ liệu. AXI MASTER cấp địa chỉ đầu tiên của quá trình đọc [axi_araddr] = 16'h0000. Tín hiệu [axi_arready] tích cực mức cao cho biết kênh sẵn sàng nhận địa chỉ đọc. Tín hiệu [axi_arvalid] tích cực mức cao báo hiệu địa chỉ đọc hợp lệ.

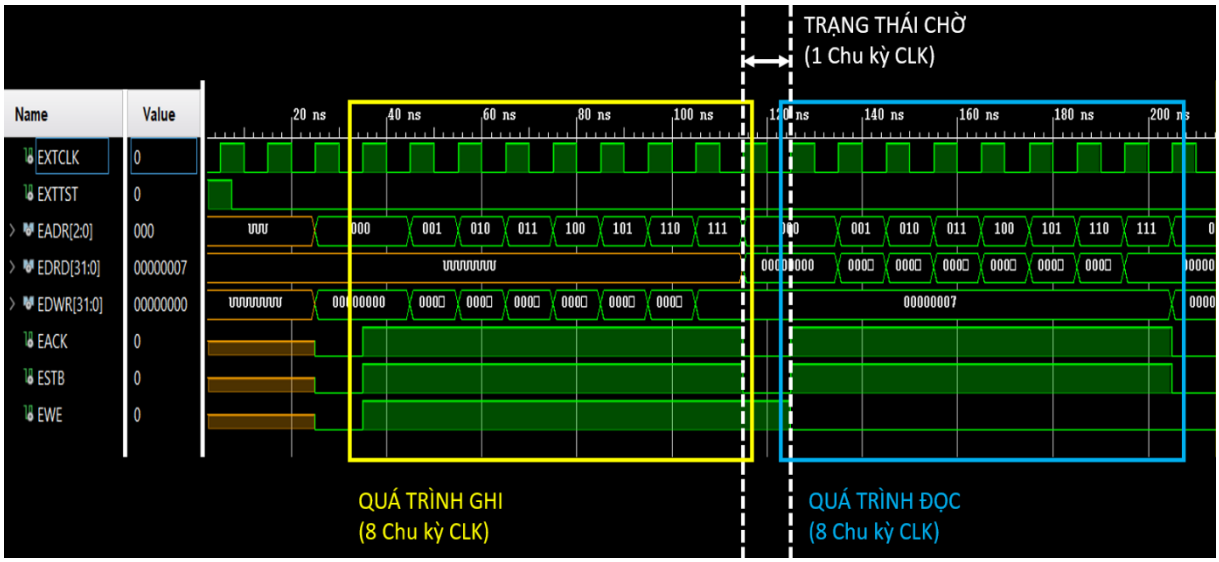
Sau khi cặp tín hiệu [axi_arready] và [axi_arvalid] tích cực, tín hiệu [axi_rready] tích cực cho biết MASTER sẵn sàng nhận dữ liệu đọc và tín hiệu này tích cực trong suốt quá trình đọc dữ liệu. Tín hiệu [axi_rvalid] cho biết có dữ liệu đọc hợp lệ và MASTER sẵn sàng nhận dữ liệu này. Dữ liệu đọc được truyền liên tục trên đường dữ liệu đọc [axi_rdata] trong 8 chu kỳ xung CLK, mỗi chu kỳ đọc một khối dữ liệu có giá trị từ 32'h00000000, 32'h00000001, 32'h00000002 đến 32'h00000007 tương ứng với các giá trị dữ liệu được ghi trong quá trình ghi trước đó tại địa chỉ 16'h0000.

Tín hiệu [axi_rlast] tích cực mức cao báo hiệu khối dữ liệu cuối cùng được truyền của quá trình đọc. Sau cùng là cặp tín hiệu [axi_rvalid] và [axi_rready] tích cực mức thấp cho biết kết thúc quá trình đọc. Quá trình đọc diễn ra tương tự cho các địa chỉ tiếp theo như 16'h0020, 16'h0040.

Thời gian thực hiện quá trình đọc/ghi một dữ liệu 32'h01234567 của bus AMBA AXI là 8 chu kỳ xung Clock cho mỗi quá trình. Thời gian để dữ liệu 32'h01234567 sau khi được ghi xong và bắt đầu được đọc là 4 chu kỳ xung Clock. Bus AMBA AXI sử dụng các kênh truyền độc lập nhau cho hoạt động đọc và ghi nên quá trình ghi một dữ liệu mới có thể được tiếp nhận trong lúc quá trình đọc đang diễn ra miễn là không cùng trên một địa chỉ. Do đó, thời gian sau khi dữ liệu 32'h01234567 được ghi xong cho đến lúc một dữ liệu mới bắt đầu được ghi cần thời gian là 6 chu kỳ xung Clock.

4.3.2 Kết quả mô phỏng của hệ thống sử dụng kiến trúc bus WISHBONE

Kết quả dạng sóng mô phỏng cho hoạt động của hệ thống kết nối điểm – điểm sử dụng kiến trúc bus WISHBONE với chu kỳ xung CLK là 10 ns (tần số 100MHz) được thể hiện trong hình 4.6 bên dưới.



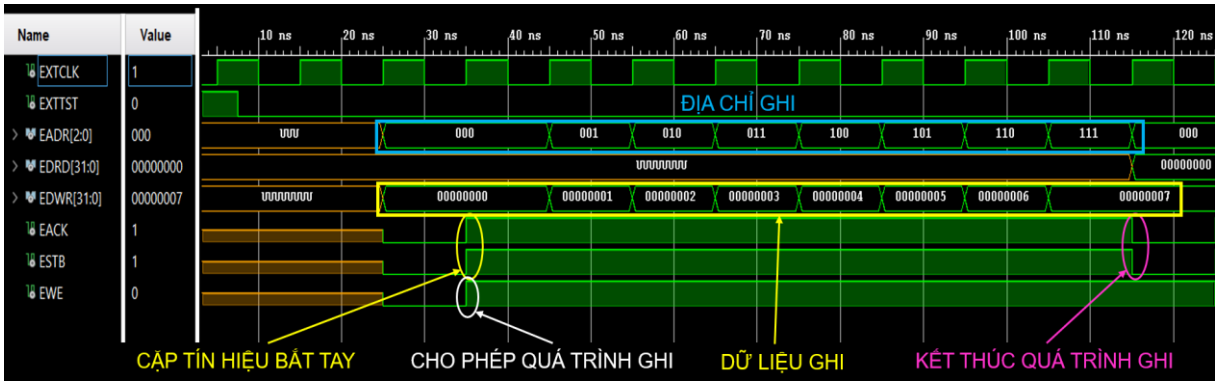
Hình 4.6: Kết quả mô phỏng hoạt động của bus WISHBONE

Hệ thống bus WISHBONE thực hiện quá trình đọc và ghi khối theo cơ chế truyền liên tục tương tự bus AMBA AXI. Tuy nhiên, phía WISHBONE MASTER phải cấp các đoạn (các pha) địa chỉ cho mỗi khối dữ liệu được truyền tới SLAVE mục tiêu.

Trong mô phỏng này, chỉ có một SLAVE mục tiêu có địa chỉ mặc định là 0x0. Đường địa chỉ [EADR] gồm các pha địa chỉ có giá trị từ 3'b000 đến 3'b111 tương ứng cho một quá trình truyền dữ liệu với 8 chu kỳ xung CLK.

Cặp tín hiệu [ESTB] và [EACK] cho biết có quá trình truyền dữ liệu truyền hợp lệ và phản hồi cho yêu cầu truyền nhận dữ liệu. Tín hiệu [EWE] cho phép quá trình đọc khi tích cực mức thấp và cho phép quá trình ghi khi tích cực mức cao. Dữ liệu ghi được truyền trên [EDWR] và dữ liệu đọc được truyền trên [EDRD].

4.3.2.1 Hoạt động ghi

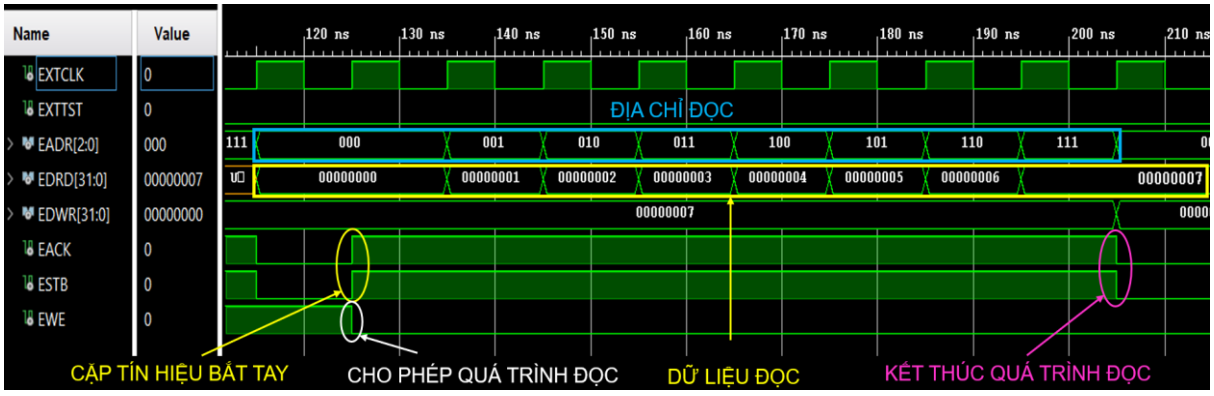


Hình 4.7: Dạng sóng mô phỏng hoạt động ghi của bus WISHBONE

Tín hiệu [EWE] tích cực mức cao cho phép quá trình ghi dữ liệu. WISHBONE MASTER gửi tín hiệu [ESTB] thông báo quá trình ghi hợp lệ. Khi tín hiệu [ESTB] tích cực mức cao thì đồng thời tín hiệu [EACK] tích cực mức cao để báo hiệu SLAVE phản hồi lại một yêu cầu hợp lệ và thực hiện quá trình ghi dữ liệu.

Khi cả 3 tín hiệu [EACK], [ESTB] và [EWE] đều tích cực mức cao, dữ liệu ghi sẽ được truyền liên tục trên [EDWR] trong 8 chu kỳ xung CLK, mỗi chu kỳ tương ứng một pha địa chỉ [EADR], có giá trị từ 3'b000 đến 3'b111. Mỗi pha ghi một khối dữ liệu có giá trị từ 32'h00000000, 32'h00000001, 32'h00000002 đến 32'h00000007. Sau khi đã ghi hết dữ liệu ở pha cuối, cặp tín hiệu [EACK] và [ESTB] tích cực mức thấp để kết thúc quá trình ghi.

4.3.2.2 Hoạt động đọc



Hình 4.8: Dạng sóng mô phỏng hoạt động đọc của bus WISHBONE

Tín hiệu [EWE] tích cực mức thấp cho phép quá trình đọc dữ liệu. WISHBONE MASTER gửi tín hiệu [ESTB] thông báo quá trình đọc hợp lệ. Khi tín hiệu [ESTB] tích cực mức cao thì đồng thời tín hiệu [EACK] tích cực mức cao để báo hiệu SLAVE phản hồi lại một yêu cầu hợp lệ và thực hiện quá trình đọc dữ liệu.

Khi 2 tín hiệu [EACK], [ESTB] tích cực mức cao và [EWE] tích cực mức thấp, dữ liệu đọc sẽ được truyền liên tục trên [EDRD] trong 8 chu kỳ xung CLK, mỗi chu kỳ tương ứng một pha địa chỉ [EADR], có giá trị từ 3'b000 đến 3'b111. Mỗi pha đọc một khối dữ liệu có giá trị từ 32'h00000000, 32'h00000001, 32'h00000002 đến 32'h00000007. Sau khi đã đọc hết dữ liệu ở pha cuối, cặp tín hiệu [EACK] và [ESTB] tích cực mức thấp để kết thúc quá trình đọc.

Thời gian thực hiện quá trình đọc/ghi một dữ liệu 32'h01234567 của bus WISHBONE là 8 chu kỳ xung Clock cho mỗi quá trình. Thời gian để dữ liệu 32'h01234567 sau khi được ghi xong và bắt đầu được đọc là 1 chu kỳ xung Clock. Bus WISHBONE sử dụng chung kênh truyền cho cả hoạt động đọc và ghi nên sau khi hoàn tất quá trình đọc thì mới có thể tiếp nhận một quá trình ghi dữ liệu mới. Do đó, thời gian sau khi dữ liệu 32'h01234567 được ghi xong cho đến lúc một dữ liệu mới bắt đầu được ghi cần thời gian là 10 chu kỳ xung Clock (1 chu kỳ chờ sau khi ghi xong, 8 chu kỳ của quá trình đọc và 1 chu kỳ chờ sau khi đọc xong).

CHƯƠNG 5
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 KẾT LUẬN

Trong đề tài này, nhóm tác giả đã thiết kế được hệ thống kết nối điểm - điểm sử dụng kiến trúc bus WISHBONE và AMBA AXI. Thông qua những kết quả đã đạt được, đề tài đã đáp ứng được những mục tiêu ban đầu đề ra. Những kết quả cụ thể mà nhóm đã đạt được như sau:

- Tìm hiểu và trình bày tổng quan về hai chuẩn bus là WISHBONE của OpenCores và AMBA AXI của ARM.
- Thiết kế được hệ thống kết nối điểm - điểm sử dụng kiến trúc bus WISHBONE và AMBA AXI.
- Thực hiện mô phỏng quá trình truyền nhận dữ liệu của hai hệ thống bus WISHBONE và AMBA AXI đúng với nội dung lý thuyết đã trình bày.
- So sánh và đánh giá về tài nguyên sử dụng và công suất tiêu thụ của mỗi hệ thống thông qua sử dụng công cụ trên phần mềm Xilinx Vivado 2019.1.

Bên cạnh những kết quả đạt được như mục tiêu đã đề ra, đề tài vẫn còn nhiều điểm hạn chế như sau:

- Thiết kế chỉ dừng lại ở mức độ mô phỏng trên phần mềm, chưa thực hiện và kiểm tra trên phần cứng thực tế.
- Testcase chỉ được xây dựng cho quá trình kết nối và truyền nhận dữ liệu đơn giản, chưa thực hiện kiểm tra cho trường hợp truyền nhận với dữ liệu lỗi.

5.2 HƯỚNG PHÁT TRIỂN

Qua các kết quả thu được sau khi thực hiện đồ án này, nhóm tác giả đề xuất hướng phát triển để hệ thống hoàn thiện hơn:

- Xây dựng các mô hình sử dụng các kiểu kết nối khác cho thiết kế với nhiều MASTER và nhiều SLAVE.
- Thực hiện trên các phần cứng thực tế để có thể đánh giá chính xác và khách quan hơn.
- Thực hiện so sánh giữa nhiều hệ thống bus hơn. Các loại bus phổ biến trên thị trường như: AMBA AHB, AMBA APB, CoreConect, OCP.

TÀI LIỆU THAM KHẢO

[1] Rohita P. Patil and Pratima V. Sangamkar, “A Review of System-On-Chip Bus Protocols”, *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, Vol. 4, Issue 1, January 2015.

[2] Mohandeep Sharma and Dilip Kumar, “WISHBONE BUS ARCHITECTURE – A SURVEY AND COMPARISON”, *International Journal of VLSI design & Communication Systems (VLSICS)*, Vol.3, No.2, April 2012.

[3] R. Usselmann, “OpenCores SoC Bus Review Rev. 1.0”, p. 12, January 9, 2001.

[4] A. Shrivastav, G. S. Tomar and A. K. Singh, "Performance Comparison of AMBA Bus-Based System-On-Chip Communication Protocol," *2011 International Conference on Communication Systems and Network Technologies*, 2011.

[5] Ayas Kanta Swain and Kamala Kanta Mahapatra, “Design and Verification of WISHBONE Bus Interface for System-on-Chip Integration”, *2010 Annual IEEE India Conference (INDICON)*, 2010.

[6] B. Sasi Rekha, G. Ananta Divya, V. Usha Sai Jyothi, "Design of AXI bus interface modules on FPGA", *International Conference on Advanced Communication Control and Computing Technologies*, 2016.

[7] M Prasanna Deepu, Prof. R. Dhanabal, "Validation of Transactions in AXI Protocol," 2017.

[8] Shubhi Sharma, Vidyadhar Jambhale, Abhijeet Shinde and S. Ravi, "Transaction based AMBA AXI bus interconnect in Verilog”, *International Research Journal of Engineering and Technology*, 2017.

[9] Chandrala Brijesh A. and Mahesh T. Kolte, “Design and Verification Point-to-Point Architecture of WISHBONE Bus for System-on-Chip”, *International Journal of Emerging Engineering Research and Technology*, Volume 2, Issue 2, PP 155-159, May 2014.

[10] Swati R. Mishra, Pramod Patil and Sudhir Shelke, “Design and Implementation of Wishbone Bus Interface Architecture for SoC Integration USING VHDL ON FPGA”, *International Journal on Recent and Innovation Trends in Computing and Communication,* July 2014.

[11] Peng Zhang, “Industrial control system simulation routines”, *Advanced Industrial Control Technology*, p. 781–810, 2010.

[12] Steven J.E. Wilton and Resve Saleh, “Programmable Logic IP Cores in SoC Design: Opportunities and Challenges”, 2001.

[13] Vojin G. Oklobdzija, Digital Systems and Applications, 2008

[14] Qiang Liu and Haie Li, “A Hierarchical IP Protection Approach for Hard IP Cores”, *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015.

[15] OpenCores, WISHBONE System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores, Revision B.3, September 7, 2002.

[16] Silicore Corporation, WISHBONE Public Domain Library for VHDL, October 8, 2001.

[17] ARM, AMBA AXI and ACE Protocol Specification, 2011.

[18] Xilinx, ZYNQ-7000 SoC Technical Reference Manual, UG585 (v1.13), April 2, 2021.

[19] Digilent, ZYBO™ FPGA Board Reference Manual, Rev. B, February 27, 2017.

[20] Xilinx, Vivado Design Suite User Guide, UG973 (v2019.1), June 7, 2019.