HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

VƯƠNG VIẾT THAO

NGHIÊN CỨU THIẾT KẾ BỘ ĐIỀU KHIỂN BỘ NHỚ NGOÀI DDRAM3 SỬ DỤNG AXI4 TRÊN HỆ THỐNG TRÊN CHIP (SOC)

ĐỀ ÁN TỐT NGHIỆP THẠC SỸ KĨ THUẬT

(Theo định hướng ứng dụng)

HÀ NỘI – 2025

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA ĐÀO TẠO SAU ĐẠI HỌC

VƯƠNG VIẾT THAO

NGHIÊN CỨU THIẾT KẾ BỘ ĐIỀU KHIỂN BỘ NHỚ NGOÀI DDRAM3 SỬ DỤNG AXI4 TRÊN HỆ THỐNG TRÊN CHIP (SOC)

Chuyên ngành: Kĩ thuật điện tử

Mã số : 8.52.02.03

ĐỀ ÁN TỐT NGHIỆP THẠC SỸ KĨ THUẬT

(Định hướng ứng dụng)

NGƯỜI HƯỚNG DẪN KHOA HỌC: TS. NGUYỄN TRUNG HIẾU

HÀ NỘI - 2025

BẢN CAM ĐOAN

Tôi xin cam đoan đề án "Nghiên cứu thiết kế bộ điều khiển bộ nhớ ngoài DDRAM3 sử dụng AXI4 trên hệ thống trên chip (SoC)" là công trình nghiên cứu khoa học của riêng tôi, được thực hiện dưới sự hướng dẫn của TS. Nguyễn Trung Hiếu. Các số liệu và trích dẫn trong đề án tốt nghiệp là trung thực. Kết quả nghiên cứu của đề án không trùng với các công trình khác.

Hà Nội, ngày 05 tháng 5 năm 2025

HỌC VIÊN CAO HỌC

(*Ký và ghi rõ họ tên*)

Vương Viết Thao

LỜI CẢM ƠN

Lời đầu tiên, tôi xin được bày tỏ lòng biết ơn sâu sắc đến TS. Nguyễn Trung Hiếu, Khoa Kĩ thuật điện tử 1, Học viện Công nghệ Bưu chính Viễn thông, người thầy đã trực tiếp hướng dẫn và đồng hành cùng tôi trong suốt quá trình thực hiện đề án *"Thiết kế bộ điều khiển DDRAM3 sử dụng giao thức AXI4 trên nền tảng SoC"*. Thầy đã tận tình định hướng, tư vấn chuyên môn cũng như luôn khuyến khích tôi vượt qua khó khăn trong quá trình nghiên cứu. Những nhận xét, góp ý và đánh giá từ thầy đã giúp tôi hoàn thiện đề án tốt nghiệp này một cách nghiêm túc và hiệu quả.

Tôi xin chân thành cảm ơn Ban lãnh đạo và các thầy cô Học viện Công nghệ Bưu chính Viễn thông nói chung, đặc biệt là các thầy cô thuộc Khoa Đào tạo Sau đại học và Khoa Kỹ thuật điện tử 1, đã truyền đạt kiến thức và tạo điều kiện thuận lợi cho tôi trong suốt thời gian học tập tại trường.

Tôi cũng xin gửi lời cảm ơn chân thành đến Khoa Kĩ thuật điện tử 1, đã tạo điều kiện thuận lợi để tôi có thể cân bằng giữa công việc và học tập.

Cuối cùng, tôi xin tri ân sâu sắc đến gia đình và bạn bè – những người luôn bên cạnh động viên, khích lệ và tiếp thêm động lực cho tôi trong suốt hành trình học tập và hoàn thành đề án này.

Một lần nữa, tôi xin chân thành cảm ơn!

Hà Nội, ngày 05 tháng 5 năm 2025

HỌC VIÊN CAO HỌC

(*Ký và ghi rõ họ tên*)

Vương Viết Thao

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT VÀ CÁC KÍ HIỆU

STT	Ký hiệu	Ý nghĩa tiếng Anh	Ý nghĩ tiếng tiếng Việt
1.	AMBA	Advanced Microcontroller Bus Architecture	Kiến trúc bus tiên tiến dành cho vi điều khiển
2.	APB	Advanced Peripheral Bus	Bus ngoại vi nâng cao
3.	ASIC	Application-Specific Integrated Circuit	Mạch tích hợp chuyên dụng cho dụng cụ cụ thể
4.	AXI / AXI4	Advanced eXtensible Interface	Giao diện mở rộng tiên tiến
5.	BRESP / RRESP	Write/Read Response	Tín hiệu phản hồi ghi / đọc
6.	CAS	Column Address Strobe	Xung điều khiển cột
7.	CLK	Clock	Xung nhịp
8.	CPU	Central Processing Unit	Bộ xử lý trung tâm
9.	DDR SDRAM	Double Data Rate Synchronous DRAM	DRAM đồng bộ tốc độ kép
10.	DDR2/DDR3/DDR4	DDR Generation 2/3/4	Các thế hệ DDR
11.	DMA	Direct Memory Access	Truy cập bộ nhớ trực tiếp

12.	DQ	Data Queue	Đường truyền dữ liệu
13.	DQS	Data Strobe	Tín hiệu chốt dữ liệu
14.	DRAM	Dynamic RAM	RAM động
15.	DSP	Digital Signal Processing	Xử lý tín hiệu số
16.	ECC	Error Correction Code	Mã sửa lỗi
17.	EDA	Electronic Design Automation	Tự động hóa thiết kế điện tử
18.	FIFO	First-In First-Out	Hàng đợi vào trước ra trước
19.	FSM	Finite State Machine	Máy trạng thái hữu hạn
20.	GPU	Graphics Processing Unit	Khối xử lý đồ họa
21.	HDL	Hardware Description Language	Ngôn ngữ mô tả phần cứng
22.	HLS	High-Level Synthesis	Tổng hợp ở cấp độ cao
23.	I2C	Inter-Integrated Circuit	Giao tiếp liên vi mạch
24.	I/O	Input/Output	Đầu vào / đầu ra
25.	IP Core	Intellectual Property Core	Lõi phần cứng sẵn có
26.	IoT	Internet of Things	Mạng lưới thiết bị kết nối mạng

27.	JEDEC	Joint Electron Device Engineering Council	Hội đồng chuẩn hóa điện tử
28.	MIG	Memory Interface Generator	Khối tạo giao tiếp bộ nhớ
29.	M / S	Master / Slave	Thành phần điều khiển / bị điều khiển
30.	MT/s	Mega Transfers per second	Triệu lần truyền mỗi giây
31.	ODT	On-Die Termination	Kết thúc tín hiệu nội mạch
32.	QoS	Quality of Service	Chất lượng dịch vụ
33.	RAM	Random Access Memory	Bộ nhớ truy cập ngẫu nhiên
34.	RAS	Row Address Strobe	Xung điều khiển hàng
35.	RTL	Register Transfer Level	Mức truyền thanh ghi
36.	SoC	System on Chip	Hệ thống tích hợp trên chip
37.	SPI	Serial Peripheral Interface	Giao diện nối tiếp ngoại vi
38.	SRAM	Static RAM	RAM tĩnh
39.	UART	Universal Asynchronous Receiver Transmitter	Bộ thu phát không đồng bộ

40.	VALID / READY	VALID / READY signals	Tín hiệu bắt tay trong AXI
41.	VHDL	VHSIC Hardware Description Language	Ngôn ngữ HDL cho mạch tích hợp
42.	WE#	Write Enable	Tín hiệu cho phép ghi
43.	WRAP / INCR / FIXED Burst	Burst Types: Wrap / Increment / Fixed	Các kiểu giao dịch dữ liệu

DANH MỤC CÁC BẢNG

DANH MỤC HÌNH ẢNH

MỞ ĐẦU

1. Lý do chọn đề tài

Trước sự phát triển bùng nổ mạnh mẽ của ngành vi mạch bán dẫn trên thế giới, Việt Nam cũng đang định hướng phát triển ngành bán dẫn và vi mạch điện tử. Thiết kế hệ thống trên chip đã trở thành một lĩnh vực nghiên cứu phổ biến trong các tổ chức giáo dục bậc đại học, các trường cao đẳng đào tạo về điện tử. Để đạt được sự phát triển bùng nổ ngành bán dẫn ở Việt Nam, các trường đào tạo ngành nghề luôn phải đi đầu trong công cuộc nghiên cứu và thiết kế, phát triển xây dựng kiến thức nền phục vụ tốt cho ngành học. Đứng trước những yêu cầu về sự phát triển vi mạch bán dẫn không chỉ ở trên thế giới, mà Việt Nam cũng ngày càng phát triển, chính vì lẽ đó đề tài "NGHIÊN CỨU THIẾT KẾ BỘ ĐIỀU KHIỂN BỘ NHỚ NGOÀI DDRAM3 SỬ DỤNG AXI4 TRÊN HỆ THỐNG TRÊN CHIP (SOC)" được tôi chọn để thực hiện đề án tốt nghiệp, đóng góp một phần kiến thức vào sự phát triển vi mạch tương lai.

Sự phát triển của công nghệ ngày càng có chiều hướng đi sâu vào nghiên cứu cải thiện phần cứng, thiết kế thu gọn giảm kích thước, tăng tốc độ đối với các vi mạch điện tử ngày nay. Vì thế việc tích hợp nhiều thành phần khác nhau lên một IC hay một con chip trở thành xu hướng của phát triển ngành bán dẫn hiện nay. Đứng trước dữ liệu khổng lồ và nhu cầu lưu trữ dữ liệu với tốc độ cao ngày càng tăng, các phương pháp lưu trữ cũ đã không còn đủ dung lượng và không đủ tốc độ cao để có thể đáp ứng được nhu cầu thực tiễn. Với việc cải thiện bộ điều khiển bộ nhớ và phát triển bộ điều khiển đa kênh DDR SDRAM hỗ trợ việc lưu dung lượng dữ liệu lớn và tăng tốc độ đọc ghi dữ liệu đặc biệt cho hệ thống trên chips SOC.

2. Tổng quan nghiên cứu

Nhiều bộ xử lý tích hợp SoC đã hỗ trợ giao tiếp DDR3 qua giao thức chuẩn như AXI [1], [2], [3]. Giao thức AXI4 (Advanced eXtensible Interface) của ARM là một phần trong hệ thống AMBA [4], được sử dụng rộng rãi để kết nối các IP trong hệ thống, hỗ trợ truyền dữ liệu theo cơ chế truyền dữ liệu liên tục, kênh truyền độc

lập và độ trễ thấp. Do đó, việc xây dựng bộ điều khiển DDR3 tương thích với AXI4 giúp tối ưu luồng dữ liệu giữa bộ xử lý và bộ nhớ ngoài, đồng thời bảo đảm khả năng mở rộng và tái sử dụng trong các hệ thống khác nhau.

Bên cạnh đó, các hướng tiếp cận khác như thiết kế RTL giảm độ trễ đọc [5], sử dụng giao tiếp quang học FPGA–DRAM để cải thiện tính ổn định [6], thiết kế bus đa cổng [7], và sử dụng các mô hình chuẩn hóa để kiểm tra việc tuân thủ giao thức AXI [4], đều khẳng định tính cấp thiết của việc nghiên cứu các bộ điều khiển DDR3 hiệu năng cao.

Từ các công trình nói trên, có thể thấy rằng việc thiết kế một bộ điều khiển DDR3 tùy biến, tuân thủ giao thức AXI4, hoạt động ổn định trong môi trường hệ thống trên chip SoC là hoàn toàn khả thi, có tính ứng dụng cao trong các hệ thống xử lý tín hiệu số, thiết kế hệ thống trên chip SoC, các hệ thống nhúng cần lưu trữ dữ liệu lớn mà phần cứng nội không đáp ứng được.

Những nghiên cứu liên quan:

Những thiết kế trước đó như [1], [3], [7], [8] đã tập trung vào việc xây dựng bộ điều khiển DDR3 với hiệu năng cao bằng cách tối ưu hóa cơ chế truy cập đa cổng, khai thác cấu trúc FSM, hoặc loại bỏ các thành phần cơ chết bắt tay không cần thiết trong giao tiếp AXI. Nghiên cứu [9] khai thác những hướng tiếp cận đặc biệt như sử dụng FIFO độ trễ thấp, hay kết nối quang học nhằm tăng tốc độ truy xuất và độ ổn định. Bên cạnh đó, các công trình [2], [8] tập trung vào kiến trúc phần cứng, ASIC hoặc IP tái sử dụng nhằm đơn giản hóa việc tích hợp bộ điều khiển vào hệ thống SoC lớn hơn, trong khi [1], [6] hướng đến việc kiểm chứng tính tuân thủ chuẩn giao thức AXI ở mức logic.

Tuy nhiên, phần lớn các thiết kế nói trên hoặc tập trung vào một khía cạnh cụ thể (thông lượng, IP tái sử dụng, giao tiếp), hoặc phục vụ cho các hệ thống ứng dụng riêng biệt như radar, xử lý ảnh hoặc truyền dẫn tốc độ cao. Trong khi đó, để kết hợp cả tính tùy biến cao, khả năng tích hợp đơn giản với SoC thông qua AXI4, và độ ổn định khi hoạt động trong nhiều miền xung nhịp khác nhau.

Nội dung đề án:

Đề án này hướng đến việc thiết kế một bộ điều khiển DDR3 tùy biến, tuân thủ giao thức AXI4, có khả năng cấu hình linh hoạt, và tương thích với các hệ thống SoC hiện đại. Thiết kế sẽ ưu tiên đảm bảo tính đúng chuẩn, hiệu suất truy xuất ổn định, đồng thời hỗ trợ mở rộng hoặc tích hợp vào các hệ thống nhúng xử lý tín hiệu số, các hệ thống IOT đòi hỏi lưu trữ lớn và các hệ thống SoC.

3. Mục đích nghiên cứu

Nghiên cứu, thiết kế, mô phỏng và thử nghiệm bộ điều khiển bộ nhớ ngoài DDR3 thành công trên công cụ thiết kế vi mạch Vivado. Nghiên cứu, làm chủ quy trình công nghệ thiết kế và chế tạo hệ thống trên chip (SoC) đồng thời đóng góp một phần cho nghiên cứu trong lĩnh vực thiết kế vi mạch số và hệ thống nhúng. Nghiên cứu tập trung xây dựng được bộ điều khiển bộ nhớ ngoài DDR3 và tích hợp trên SoC, tiến tới phát triển hệ thống truyền nhận dữ liệu theo chuẩn AXI4 được tích hợp trên hệ thống SoC, phục vụ cho các hệ thống đòi hỏi lưu trữ dữ liệu với tốc độ cao, chính xác và không gian lưu trữ lớn.

4. Phương pháp nghiên cứu

Về phương pháp nghiên cứu, tôi sử dụng giao thức AXI4 là một trong các giao thức BUS trong họ AMBA được phát triển bởi hãng ARM để thiết kế bộ điều khiển bộ nhớ ngoài cho DDR3 trên SoC. Sử dụng công cụ thiết kế logic phát triển nghiên cứu thử nghiệm, mô phỏng, kiểm tra, đánh giá tính chính xác của bộ điều khiển DDRAM cho việc điều khiển bộ nhớ ngoài. Sử dụng IP-CORE có sẵn MIG 7 Series thực hiện tích hợp DDR3 vào IP-CORE. MIG 7 Series hộ trợ giao thức AXI4 hộ trợ việc thiết kế và thử nghiệm bộ điều khiển bộ nhớ ngoài một cách thuận tiện.

Tính toán đo đạc thời gian so sánh với các nghiên cứu về bộ điều khiển đã có. Thực hiện đọc, ghi dữ liệu, thực hiện so sánh dữ liệu ghi vào và dữ liệu đọc lại được để đảm bảo tính chính xác của bộ điều khiển thiết kế ra.

5. Bố cục đề án tốt nghiệp

Đề án được chia thành bốn chương nội dung chính, nhằm trình bày rõ ràng quá trình nghiên cứu, thiết kế và đánh giá bộ điều khiển bộ nhớ DDR3 sử dụng giao thức AXI4 trong môi trường hệ thống trên chip (SoC).

Chương I "*Cơ sở lý luận*", trình bày tổng quan về các loại bộ nhớ DRAM, đặc biệt là DDR3 với cấu trúc và nguyên lý hoạt động chi tiết. Làm nền tảng kỹ thuật cho việc thiết kế bộ điều khiển bộ nhớ trong các hệ thống SoC hiện đại.

Chương II "*Nghiên cứu giải pháp điều khiển bộ nhớ ngoài DDRAM3 trên SoC*", Phân tích kiến trúc SoC, các loại bộ nhớ tích hợp và so sánh giao thức AXI với Avalon. Khẳng định AXI4 là lựa chọn tối ưu để điều khiển bộ nhớ DDR3 nhờ hiệu suất và khả năng mở rộng cao.

Chương III "*Xây dựng hệ thống bộ điều khiển DDRAM Controller*", Thiết kế bộ điều khiển DDR3 sử dụng IP-Core MIG và giao tiếp AXI4 với các khối đọc/ghi và máy trạng thái. Giải thích chi tiết cơ chế burst, bắt tay, quản lý địa chỉ và dữ liệu trong điều khiển bộ nhớ.

Chương IV *"Mô phỏng và đánh giá mức độ hiệu quả của bộ điều khiển DDRAM"*, Thực hiện mô phỏng trên Vivado, cấu hình MIG và kiểm thử chức năng đọc/ghi thực tế. Kết quả xác nhận bộ điều khiển hoạt động chính xác, hiệu quả và phù hợp tích hợp SoC.

NỘI DUNG

CHƯƠNG I: CƠ SỞ LÝ LUẬN

Chương I xây dựng các lý thuyết, những khái niệm cơ bản về bộ nhớ, cấu trúc của bộ nhớ truy cập ngẫu nhiên, nguyên tắc hoạt động cũng như đọc ghi vào bộ nhớ truy cập ngẫu nhiên DDR SDRAM. Giới thiệu về DDR SDRAM, với trọng tâm là DDR3.

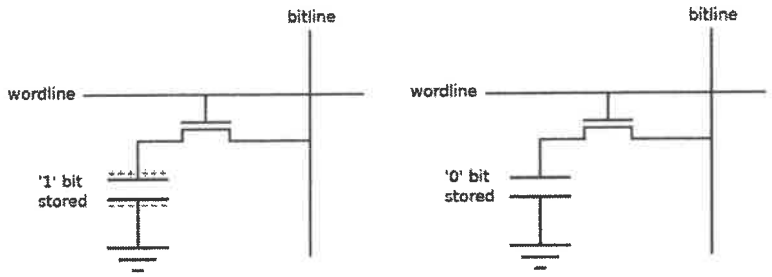
1.1 Tổng quan về bộ nhớ truy cập ngẫu nhiên DRAM

Bộ nhớ truy cập ngẫu nhiên (RAM - Random Access Memory) là một dạng bộ nhớ cho phép đọc hoặc ghi dữ liệu tại bất kỳ vị trí nào trong không gian lưu trữ mà không cần duyệt qua các ô nhớ khác theo thứ tự. Khác với bộ nhớ tuần tự, RAM cung cấp khả năng truy xuất trực tiếp, giúp tăng tốc độ xử lý trong các hệ thống máy tính. Tuy nhiên, RAM thuộc loại bộ nhớ dễ bay hơi (volatile), nghĩa là toàn bộ dữ liệu sẽ biến mất khi nguồn điện bị cắt. Đặc điểm này xuất phát từ cấu trúc vật lý của nó: các ô nhớ trong RAM, dù là DRAM hay SRAM, đều dựa trên tụ điện hoặc transistor những thành phần không thể duy trì trạng thái mà không có điện năng [1]. Khi nguồn điện ngừng, điện tích trong tụ điện tiêu tan, kéo theo sự mất mát của thông tin được lưu trữ.

1.2 DRAM và DDR SDRAM

1.2.1 DRAM

DRAM là loại bộ nhớ truy cập ngẫu nhiên động, có ưu điểm sử dụng ít mạch hơn trên mỗi IC so với các loại RAM khác như SRAM. Ô nhớ DRAM lưu trữ dữ liệu dưới dạng điện tích trong tụ điện, mỗi ô gồm một tụ điện và một transistor điều khiển truy cập. Điện tích này cần được làm mới định kỳ để tránh mất dữ liệu do rò rỉ tự nhiên.



Hình 1. 1: Cấu trúc tổng quát của DRAM

Cấu trúc: Ô nhớ được tổ chức thành mảng hai chiều với các hàng và cột. Để truy cập dữ liệu: Đọc toàn bộ hàng bằng tín hiệu RAS# (Row Address Strobe). Chọn cột bằng tín hiệu CAS# (Column Address Strobe) để đọc/ghi qua chân I/O.

Việc đọc DRAM có tính chất xóa, nên dữ liệu phải được ghi lại sau mỗi lần đọc. DRAM đồng bộ với xung nhịp hệ thống, dùng các tín hiệu như WE# (Write Enable) để phân biệt đọc/ghi, giúp tối ưu tốc độ truy cập trong các hệ thống máy tính.

1.2.2 DDR SDRAM và phân loại

1.2.2.1 DDR2 SDRAM

DDR2 SDRAM cải tiến từ DDR SDRAM với tốc độ xung nhịp cao hơn (200-533 MHz), đạt băng thông từ 400-1066 Mb/s/pin, nâng cao hiệu suất xử lý dữ liệu. Để đảm bảo tín hiệu ổn định ở tốc độ cao, DDR2 tích hợp ODT (On-Die Termination), cho phép chấm dứt tín hiệu ngay trong chip, thay vì dùng linh kiện ngoài như DDR [10].

DDR2 hoạt động ở điện áp 1.8V (thấp hơn 2.5V của DDR), giảm tiêu thụ năng lượng khoảng 28%. Số cột trong mỗi hàng giảm, giúp tiết kiệm điện khi truy cập. Tín hiệu vi sai được hỗ trợ, giảm nhiễu và tăng độ tin cậy, cấu hình qua thanh ghi EMR (Extended Mode Register).

Tính năng độ trễ bổ sung cho phép linh hoạt gửi lệnh đọc/ghi sớm hơn sau lệnh kích hoạt, tối ưu hóa băng thông. DDR2 dùng 8 bank bộ nhớ, tăng khả năng truy cập xen kẽ, và hỗ trợ burst length lên đến 8, phù hợp với ứng dụng hiệu năng cao [1].

Bảng 1: Bảng so sánh tốc độ của các dòng DDR2

DDR2 SRAM	Data Rate	Speed
DDR2-400	400 Mb/s/pin	200MHz
DDR2-533	533 Mb/s/pin	266MHz
DDR2-667	667 Mb/s/pin	333MHz
DDR2-800	800 Mb/s/pin	400MHz
DDR2-1066	1066 Mb/s/pin	533MHz

1.2.2.2 DDR3 SDRAM

DDR3 SDRAM (Double Data Rate 3 Synchronous Dynamic RAM) lưu trữ dữ liệu bằng điện tích trong tụ điện, mỗi tụ đại diện cho một bit (0: không điện tích, 1: có điện tích). Do rò ri điện tích, DDR3 cần cơ chế làm mới định kỳ để duy trì dữ liệu và chỉ lưu trữ tạm thời, mất dữ liệu khi ngắt nguồn.

So với DDR2, DDR3 cải tiến với: Tốc độ: 800-2133 Mb/s, xung nhịp 400-1066 MHz. Điện áp: 1.5V (thấp hơn 1.8V của DDR2), tiết kiệm năng lượng. Prefetch: 8n (truyền 8 bits mỗi 4 chu kỳ), tăng băng thông so với 4n của DDR2 [10].

DDR3 dùng cấu trúc fly-by trên DIMM [10] thay cho cấu trúc cây của DDR2, giảm độ trễ tín hiệu và nâng cao hiệu suất. Nó cũng khai thác 4 thanh ghi chế độ (Mode Register) để điều chỉnh độ trễ đọc/ghi, tối ưu truyền dữ liệu. Với dung lượng từ 512 Mb đến 8 Gb, DDR3 phù hợp cho các hệ thống yêu cầu xử lý dữ liệu lớn và tốc độ cao, như SoC

Bảng 2: Bảng so sánh tốc độ dữa các dòng DDR3

DDR3 SRAM	Data Rate	Speed
DDR3-800	800 Mb/s/pin	400MHz
DDR3-1066	1066 Mb/s/pin	533MHz
DDR3-1333	1333 Mb/s/pin	667MHz
DDR3-1600	1600 Mb/s/pin	800MHz
DDR3-1866	1866 Mb/s/pin	933MHz
DDR3-2133	2133 Mb/s/pin	1066MHz

Ngoài ra, DDR3 còn được thiết kế với trở kháng đầu ra cao hơn (34 Ω) so với DDR2 (18 Ω), góp phần nâng cao hiệu quả hoạt động.

DDR3 có khả năng hỗ trợ dung lượng từ 512 Mb đến 8 Gb và cấu hình đầu ra đa dạng gồm x4, x8, và x16. Nó sử dụng kiến trúc tìm nạp trước 8n, cho phép truyền 8 từ dữ liệu trong 4 chu kỳ xung nhịp, so với 4n của DDR2, vốn chỉ truyền được 4 từ trong 2 chu kỳ.

Một điểm nổi bật khác của DDR3 là sự cải tiến trong cấu trúc thanh ghi chế độ (Mode Register). Trong khi DDR2 chỉ sử dụng hai thanh ghi chế độ, DDR3 khai thác cả bốn thanh ghi để tinh chỉnh độ trễ khi đọc (CAS latency) và ghi. Điều này giúp tăng hiệu quả truyền tải dữ liệu.

Những cải tiến về hiệu suất, điện năng tiêu thụ, và dung lượng giúp DDR3 SDRAM trở thành lựa chọn hàng đầu cho các hệ thống cần xử lý dữ liệu lớn và tốc độ cao.

1.2.2.3 DDR4 SDRAM

DDR4 SDRAM, được JEDEC chuẩn hóa vào tháng 9/2012, cải thiện vượt trội so với các thế hệ DRAM trước. DDR4 đạt tốc độ từ 2133 MT/s (mục tiêu 3200 MT/s), tăng gấp đôi mật độ bộ nhớ, và giảm điện áp từ 1.5V (DDR3) xuống 1.2V, tiết kiệm khoảng 20% năng lượng. Những nâng cấp này hỗ trợ các hệ thống hiện đại cần hiệu suất cao và tiêu thụ điện thấp [10].

DDR4 thay đổi cách thiết lập điện áp tham chiếu (Vref) bằng phương pháp đo lặp tín hiệu DQ và DQS, cải thiện độ chính xác tín hiệu so với Vref cố định 750 mV của DDR3. Nó cũng áp dụng kiểm tra jitter thống kê: dưới 2133 MT/s dùng jitter xác định (DJ), trên mức này kết hợp jitter ngẫu nhiên (RJ), đảm bảo độ ổn định tín hiệu ở tốc độ cao. Các cải tiến này giúp DDR4 trở thành chuẩn mới cho ứng dụng đòi hỏi xử lý mạnh mẽ [2].

Bảng 3 : Bảng so sánh tốc độ dữa các dòng DDR4

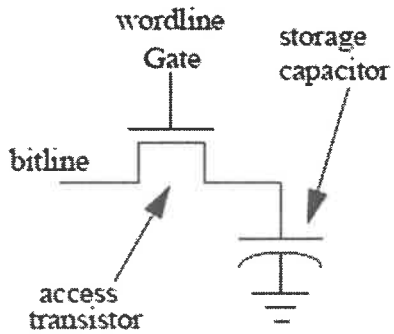
DDR4 SRAM	Data Rate	Speed
DDR4-1600	1600 Mb/s/pin	800MHz
DDR4-1866	1866 Mb/s/pin	933MHz

DDR4-2133	2133 Mb/s/pin	1066MHz
DDR4-2140	2140 Mb/s/pin	1200MHz
DDR4-2667	2667 Mb/s/pin	1333MHz
DDR4-3200	3200 Mb/s/pin	1600MHz

1.3 Cấu trúc của DDR SDRAM

1.3.1 Cell storage trong DDR SDRAM

Trong DDR SDRAM (Double Data Rate Synchoronous Random-Access Memory) các tết bào lưu trữ hay còn được gọi là cells storage là đơn vị cơ bản của bộ nhớ, nơi dữ liệu được lưu trữ dưới dạng điện tích trong các tụ điện. Dưới đây là mô tả chi tiết về cấu trúc và hoạt động của các tết bào lưu trữ trong DDRSDRAM [10]:



Hình 1. 2: Cells Strorage trong DDR SDRAM

Cell DDR SDRAM được cấu tao từ các transistor và capacitor.

Transistor: Hoạt động như một công tắc điều khiển việc đọc và ghi qua capacitor. Khi một transistor được kích hoạt, nó cho phép dòng điện chạy qua và truy cập vào capacitor.

Capacitor: Lưu trữ bit dữ liệu dưới dạng điện tích. Một capacitor khi được nạp đầy điện tích thì giá trị của nó là "1", trong khi một capacitor không có điện tích có giá trị là "0".

Ghi dữ liệu (Write): Bộ điều khiển bộ nhớ sẽ kích hoạt transistor để cho phép dữ liệu dưới dang điện tích được lưu trữ trong capacitor. Đọc dữ liệu (Read): Khi đọc dữ liệu, transistor được kích hoạt để cho phép điện tích từ capacitor truyền qua, và giá trị này được

đọc bởi mạch điện cảm biến. Làm mới (Refresh): Do capacitor có thể mất điện tích theo thời gian, các cell DDRAM cần được làm mới theo chu kì để duy trì được dữ liệu.

1.3.2 Các thành phần cơ bản của DDR SDRAM

1.3.2.1 Bank

Banks trong DDR SDRAM là các đơn vị bộ nhớ con giúp tối ưu hiệu suất bằng cách cho phép truy cập song song và quản lý hoạt động đọc/ghi hiệu quả. Một bộ nhớ DDR SDRAM thường có nhiều banks, mỗi bank hoạt động độc lập, cho phép bộ điều khiển truy cập đồng thời mà không gây xung đột.

Trong DDR3 SDRAM, số lượng banks dao động từ 4 đến 16, tùy theo loại chip. Mỗi bank có bộ điều khiển riêng, cho phép vi xử lý truy cập đồng thời vào nhiều vùng nhớ khác nhau, giảm độ trễ và tăng tốc độ xử lý dữ liệu.

Cấu trúc của một bank gồm

Hàng (Row): Mỗi bank có nhiều hàng, mỗi hàng chứa một số lượng cột nhất định.

Cột (Column): Các cột trong mỗi hàng tạo thành lưới ô nhớ.

Ô nhớ (Memory Cell): Nằm tại giao điểm giữa hàng và cột, mỗi ô chứa một bit dữ liệu dưới dạng điện tích trong tụ điện.

Nhờ cách tổ chức này, DDR SDRAM có thể thực thi nhiều lệnh song song, giúp cải thiện hiệu suất truy xuất dữ liệu đáng kể.

1.3.2.2 Rank

Rank là tập hợp các chip nhớ trên một module DDR SDRAM, chia sẻ chung một tập lệnh điều khiển và có thể được kích hoạt cùng lúc. Mỗi rank hoạt động như một đơn vị độc lập để đọc hoặc ghi dữ liệu.

Single Rank và Dual Rank: Single rank module bộ nhớ có một rank duy nhất. Dual Rank module có hai rank, mỗi rank có thể truy cập độc lập nhưng không đồng thời.

Cấu trúc của Rank

Chip nhớ: Một rank bao gồm nhiều chip nhớ DDR SDRAM.

Tín hiệu điều khiển: Các chip trong cùng một rank chia sẻ chung tập lệnh như command, address.

Data bus: Dùng chung bus dữ liệu, nhưng chỉ một rank có thể truyền dữ liệu tại một thời điểm.

Nhờ cách tổ chức này, DDR SDRAM có thể cải thiện hiệu suất bộ nhớ đáng kể, đặc biệt trong các hệ thống cần băng thông cao.

1.3.2.3 Row

Địa chỉ hàng được cung cập với lệnh ACTIVATE được giải mã bởi mạch giải mã hàng. Bộ mã này chịu trách nhiệm chọn đúng hàng có trong bank bộ nhớ. Bộ giải mã hàng kích hoạt tương ứng với hàng được chọn thông qua đường từ. Đường từ là một dây dẫn kết nối đến tất cả các ô nhớ (tụ điện) trong hàng được chọn. Kích hoạt đường từ làm cho điện tích được lưu trữ trong các tụ điện của các ô nhớ được truyền đến các đường bit tương ứng (cột). Việc truyền tải điện tích này rất quan trọng để đọc hoặc ghi dữ liệu.

1.3.2.4 Column

Cấu trúc cột trong DDR SDRAM chịu trách nhiệm cho việc truy xuất dữ liệu từ các ô nhớ đã được chọn bởi hàng. Bit line: mỗi cột trong ma trận bộ nhớ DDR tương ứng với một đường bit. Các đường bit chạy dọc theo cột và kết nối với các ô nhớ trong mỗi hàng.

Memory cells là các ô nhớ, tại mỗi giao điểm của hàng và côt, có một ô nhớ. Mỗi ô nhớ chứa một bit dữ liệu dưới dạng điện tích trong một tụ điện. Bộ khuếch đại cảm biến: các đường bit được kết nối với các khuếch đại cảm biến, chịu trách nhiệm khuếch đại tín hiệu điện nhỏ từ cá ô nhớ để có thể đọc được.

1.4 Nguyên tắc truy cập bộ nhớ DDR SDRAM

Lệnh activate: bộ điều khiển bộ nhớ gửi lệnh activate cùng với địa chỉ hàng (Row address) [10] đến DDR SDRAM. Lệnh này chỉ ra một hàng cụ thể trong một bank cần được kích hoạt.

Giải mã địa chỉ hàng:

Bộ giải mã hàng: địa chỉ hàng được gửi đến bộ giải mã hàng. Bộ giải mã hàng nhận địa chỉ hàng và xác định hàng nào trong banks nào cần được kích hoạt.

Kích hoạt đường: Sau khi địa chỉ hàng được giải mã, bộ giải mã hàng kích hoạt đường tương ứng với hàng được chọn. Đường từ này kết nối với các ô nhớ trong hàng với các đường bit (bit line).

Truyền điện tích: Khi đường từ được kích hoạt, điện tích lưu trữ trong các ô nhớ của hàng được truyền đến các đường bit. Các đường bit dẫn tín hiệu điện tích từ các ô nhớ đến các bộ khuếch đại cảm biến.

Các bộ khuếch đại cảm biến nhận tín hiệu điện nhỏ từ các ô nhớ và khuếch đại chúng để tạo ra tín hiệu đủ mạnh để đọc hoặc ghi. Quá trình khuếch đại này đảm bảo rằng dữ liệu từ các ô nhớ có thể được xử lý chính xác.

1.5 Kết luận chương I

Trong chương 1, đề án đã trình bày cơ sở lý thuyết về bộ nhớ DRAM, từ các khái niệm tổng quát đến phân loại chi tiết như DRAM và DDR SDRAM. Qua việc phân tích đặc điểm kỹ thuật, ưu điểm và hạn chế của từng loại, tác giả lựa chọn DDR3 SDRAM làm đối tượng nghiên cứu chính trong đề án. Việc chọn DDR3 không chỉ vì tính phổ biến và hiệu suất cao trong các hệ thống SoC hiện nay, mà còn phù hợp với mục tiêu thiết kế bộ điều khiển bộ nhớ ngoài cho SoC với yêu cầu tốc độ và độ ổn định cao. Kiến thức trong chương này là nền tảng quan trọng để triển khai thiết kế và mô phỏng bộ điều khiển DDR3 trong các chương tiếp theo.

CHƯƠNG II: NGHIÊN CỨU GIẢI PHÁP ĐIỀU KHIỂN BỘ NHỚ NGOÀI DDRAM3 TRÊN SOC

Chương II trình bày tổng quan kiến trúc SoC, các loại bộ nhớ tích hợp và các giao thức điều khiển phổ biến như AXI và Avalon. Trọng tâm chương là phân tích ưu điểm của giao thức AXI4 trong điều khiển bộ nhớ DDR3, làm cơ sở lựa chọn giải pháp thiết kế phù hợp, hướng đến tích hợp hiệu quả trên hệ thống SoC.

2.1 Giới thiệu về SoC (System on chip)

2.1.1 SoC là gì?

SoC (System on Chip) là một vi mạch tích hợp tiên tiến, nơi toàn bộ hệ thống điện tử được gói gọn trong một con chip duy nhất. Xuất phát từ nhu cầu thu nhỏ thiết bị và nâng cao hiệu suất, SoC trở thành một trong những bước tiến quan trọng trong công nghệ bán dẫn, mang lại giải pháp tối ưu cả về hiệu suất lẫn mức tiêu thụ năng lượng.

Không giống như các thiết kế truyền thống sử dụng nhiều vi mạch riêng lẻ, SoC tích hợp cả phần cứng số, tương tự và mạch hỗn hợp trên cùng một nền tảng. Chẳng hạn, một SoC xử lý âm thanh có thể bao gồm bộ thu nhận tín hiệu, bộ chuyển đổi ADC, vi xử lý trung tâm, bộ điều khiển bộ nhớ và các cổng giao tiếp ngoại vi, tất cả hoạt động đồng bộ trong một hệ thống duy nhất.

SoC không chỉ kế thừa mà còn phát triển từ kiến trúc ASIC, mở rộng khả năng ứng dụng trong các lĩnh vực từ thiết bị di động, hệ thống nhúng đến ô tô thông minh. Một SoC điển hình bao gồm: Vi xử lý (CPU) - Điều khiển toàn bộ hệ thống, có thể là đơn nhân hoặc đa nhân nhằm đáp ứng các mức hiệu suất khác nhau; Bộ xử lý đồ họa (GPU): Chuyên xử lý tác vụ đồ họa, hỗ trợ các ứng dụng về game, xử lý ảnh và video; Bộ nhớ (RAM, ROM): RAM dùng để lưu trữ dữ liệu tạm thời, trong khi ROM chứa các chương trình cố định; Bộ điều khiển ngoại vi: Đảm nhận giao tiếp với các thiết bị như USB, SPI, I2C hay UART; Bộ giao tiếp mạng: Hỗ trợ các kết nối Wi-Fi, Bluetooth, LTE, Ethernet để mở rộng khả năng truyền dữ liệu; Bộ quản lý năng lượng: Giám sát và điều chỉnh mức tiêu thụ điện nhằm tối ưu hóa hiệu suất hoạt động.

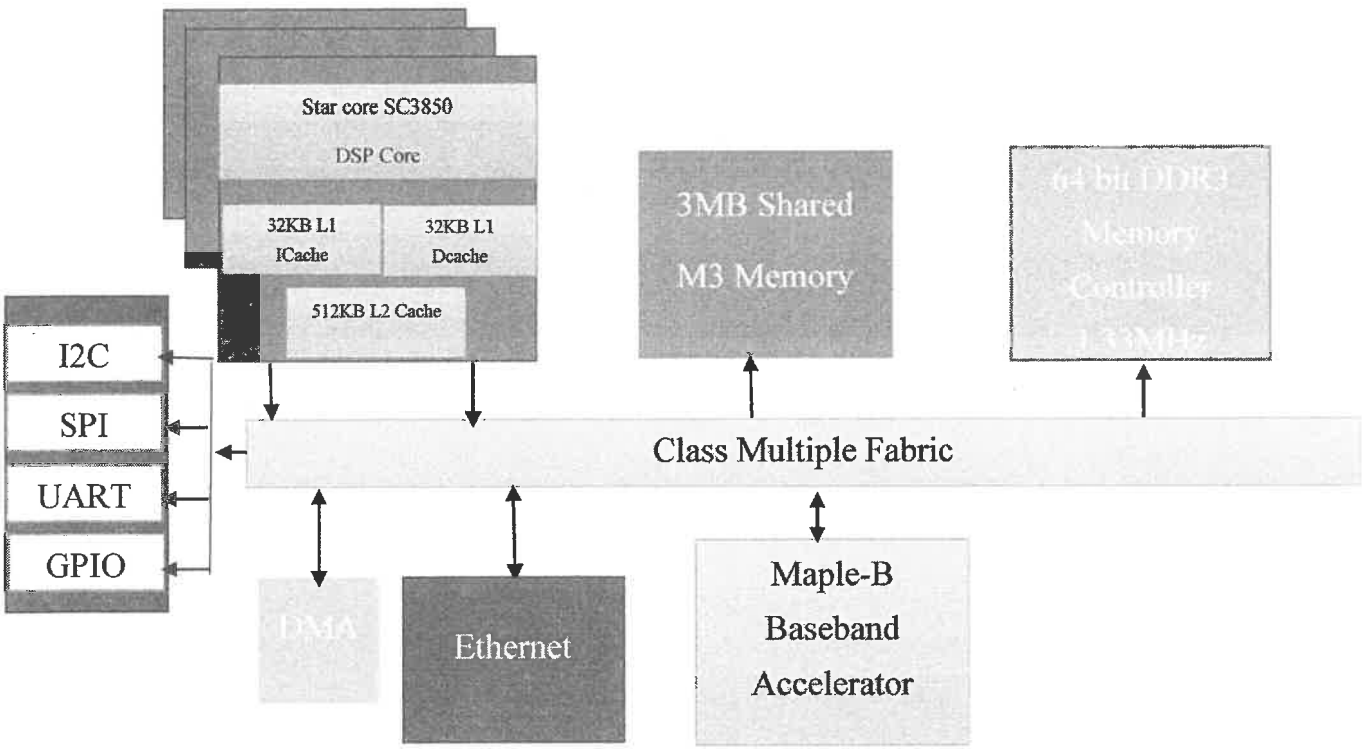
2.1.2 Các ứng dụng của SoC

Điện thoại di động: Hầu hết các điện thoại thông minh sử dụng SoC để cung cấp hiệu suất cao và tiêu thụ năng lượng thấp. Ví dụ điển hình là các SoC của Apple, QualComm, Samsung. Thiết bị IoT: Các thiết bị IoT thông minh như loa, cảm biến thông minh, các thiết bị đeo tay thông minh sử dụng SoC đảm bảo kích thước nhỏ và hiệu suất cao. Ô tô thông minh: SoC được sử dụng trong các hệ thống giải trí điều khiển động cơ và các cảm biến an toàn trong xe hơi. Thiết bị y tế: Các thiết bị y tế hiện đại như máy đo nhịp tim, máy đo huyết áp, và các thiết bị theo dõi sức khoẻ khác cũng sử dụng SoC đảm bảo độ chính xác hiệu suất cao.

2.1.3 Các giải pháp thiết kế SoC

2.1.3.1 IP và công nghệ nền cho SoC

IP core trong thiết kế SoC:



Hình 2. 1: Kiến trúc tổng quan về lõi IP Core trong SoC

Ip core (Intellectual Property Core) là các khối chức năng đã được thiết kế sẵn và có thể được tích hợp trực tiếp vào các thiết kế SoC. Các IP core này gồm:

Bộ xử lý trung tâm (CPU)

Bộ xử lý đồ hoạ (GPU)

Bộ xử lý điều khiển thiết bị ngoại vi (I/O Controller)

Bộ nhớ (ROM, RAM, DDR SDRAM, ...)

Giao tiếp không dây (Ethernet, USB, WiFi, Blutooth).

2.1.3.2 Phương pháp thiết kế theo module

Thiết kế hệ thống trên chip (SoC) liên quan chặt chẽ đến cả phần cứng và phần mềm, nhằm mục tiêu điều khiển bộ vi xử lý, các thiết bị ngoại vi cũng như các giao diện liên kết. Một quy trình thiết kế hiệu quả thường đồng thời phát triển phần cứng và phần mềm theo phương pháp đồng thiết kế kiến trúc.

Ở khía cạnh phần cứng, tương tự như các thiết kế vi mạch tích hợp khác, nhiều SoC được xây dựng dựa trên việc sử dụng các lõi IP (Intellectual Property) là các khối phần cứng đã được định nghĩa trước. Các khối này được thiết kế thông qua các công cụ tự động hóa thiết kế điện tử (EDA tools) và được phân thành ba loại chính:

Lõi IP cứng: Các bố cục cố định, tối ưu hóa cho một công nghệ chế tạo cụ thể. Mặc dù đạt hiệu suất cao, chúng kém linh hoạt do phụ thuộc vào công nghệ sản xuất.

Lõi IP mềm: Các mô tả chức năng dưới dạng mã nguồn (VHDL/Verilog), có khả năng tùy chỉnh và tái cấu hình cao. Tuy nhiên, chúng yêu cầu quá trình tổng hợp (synthesis) và xác minh (verification) trước khi triển khai.

Lõi IP bán cứng: Là dạng trung gian giữa IP cứng và IP mềm, được cung cấp dưới dạng danh sách mạng đã tối ưu hóa cho các thư viện vật lý cụ thể sau quá trình tổng hợp.

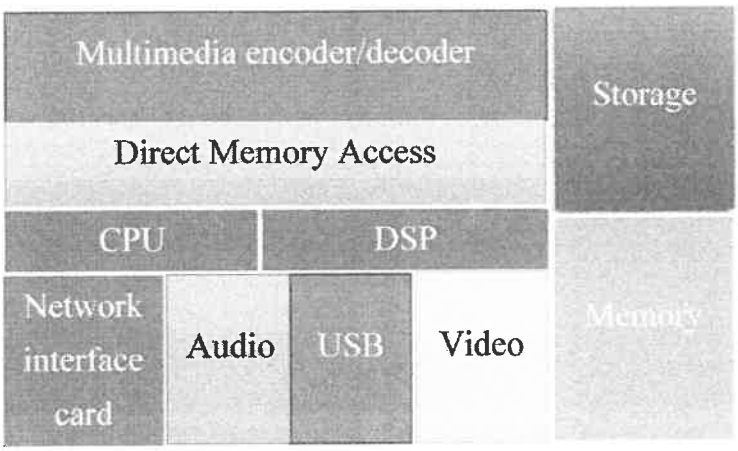
Về phần mềm, thiết kế SoC thường sử dụng các ngôn ngữ lập trình cấp cao như MATLAB hoặc C++, sau đó được chuyển đổi thành thiết kế RTL thông qua các công cụ tổng hợp mức cao (High-Level Synthesis - HLS). Các công cụ này cho phép

nhà thiết kế mô hình hóa hệ thống, mạch, phần mềm và kiểm tra xác minh ở nhiều cấp độ chỉ bằng một ngôn ngữ duy nhất.

Sau khi kiến trúc hệ thống được xác lập, các thành phần phần cứng mới sẽ được mô tả bằng ngôn ngữ mô tả phần cứng (HDL) ở mức RTL, tổng hợp và kết nối lại với nhau, tạo thành thiết kế SoC hoàn chỉnh.

2.2 Các thành phần cơ bản của SoC

SoC là viết tắt của hệ thống trên chip, là một con chip nhỏ tích hợp tất cả các thành phần và mạch cần thiết của một hệ thống cụ thể. Các thành phần chính bao gồm CPU, GPU, bộ nhớ, thiết bị I/O… SoC được sử dụng trong các thiết bị khác nhau như điện thoại thông minh, các thiết bị IoT, máy tính bảng và ứng dụng hệ thống thống nhúng. Trong bài viết này chúng ta sẽ cùng tìm hiểu kiến trúc và đặc điểm kiến trúc SoC.



Hình 2. 2: Các thành phần cơ bản của SoC

2.3 Các loại bộ nhớ phổ biến dùng để tích hợp trên SoC

Một số loại bộ nhớ ngoài phổ biến thường được sử dụng trong SoC bao gồm SRAM, DRAM, và NAND Flash, mỗi loại sở hữu những đặc điểm và ứng dụng riêng biệt:

SRAM (Static RAM): Với tốc độ truy cập nhanh và mức tiêu thụ năng lượng thấp, SRAM thường được lựa chọn làm bộ nhớ cache. Đặc biệt, loại bộ nhớ này không yêu cầu làm tươi dữ liệu định kỳ như DRAM. Tuy nhiên, chi phí sản xuất của

SRAM khá cao và dung lượng lưu trữ hạn chế, điều này khiến nó ít được sử dụng làm bộ nhớ chính.

DRAM (Dynamic RAM): Là loại bộ nhớ được sử dụng phổ biến làm RAM chính trong các hệ thống máy tính nhờ dung lượng lưu trữ lớn và chi phí sản xuất thấp hơn so với SRAM. Tuy nhiên, để duy trì dữ liệu, DRAM cần được làm tươi định kỳ, điều này có thể làm giảm hiệu quả trong một số ứng dụng yêu cầu tốc độ cao.

NAND Flash: Đây là lựa chọn hàng đầu cho các thiết bị lưu trữ như SSD và thẻ nhớ nhờ khả năng lưu trữ dữ liệu ngay cả khi không có nguồn điện. NAND Flash có dung lượng lớn và chi phí hợp lý, tuy nhiên tốc độ đọc/ghi thường thấp hơn đáng kể so với SRAM và DRAM.

Ngoài ra, DDRAM (Double Data Rate RAM) được thiết kế như một sự kết hợp giữa DRAM và SRAM, nhằm tăng cường hiệu suất đọc ghi dữ liệu và giảm thiểu các hạn chế vốn có của hai loại bộ nhớ trên. DDRAM cải thiện tốc độ truy cập, đồng thời vẫn duy trì được dung lượng lớn và chi phí hợp lý, trở thành một trong những giải pháp bộ nhớ tối ưu cho SoC.

2.4 Các giải pháp điều khiển bộ nhớ

2.4.1 Điều khiển bộ nhớ dựa trên giao thức chuẩn

AXI là một phần của kiến trúc bus AMBA do ARM phát triển, được thiết kế để đáp ứng các yêu cầu cao về băng thông và độ trễ thấp trong các hệ thống SoC hiện đại. AXI hỗ trợ các tính năng như:

Burst Transfers: Cho phép truyền nhiều dữ liệu liên tiếp trong một giao dịch, giảm độ trễ khởi tạo giao dịch và tăng hiệu suất truyền dữ liệu.

Pipelining: Hỗ trợ thực hiện các giai đoạn của giao dịch song song, cải thiện thông lượng của hệ thống.

Separate Read and Write Channels: Cung cấp các kênh đọc và ghi độc lập, cho phép thực hiện đồng thời các hoạt động đọc và ghi mà không gây xung đột.

QoS Support: Hỗ trợ quản lý chất lượng dịch vụ, cho phép ưu tiên các giao dịch quan trọng và đảm bảo hiệu suất cho hệ thống.

AXI được thiết kế để hỗ trợ các hệ thống phức tạp với nhiều master và slave, cung cấp các cơ chế để quản lý truy cập và đảm bảo tính toàn vẹn của dữ liệu. Điều này làm cho AXI trở thành lựa chọn phổ biến trong các thiết kế SoC hiệu năng cao, đặc biệt là trong các hệ thống dựa trên vi xử lý ARM.

2.4.2 Bộ điều khiển bộ nhớ chuyên dùng

Bộ điều khiển bộ nhớ chuyên dụng đóng vai trò trung tâm trong việc quản lý truy cập bộ nhớ. Đối với các bộ nhớ như DDRAM3, bộ điều khiển phải xử lý các tác vụ phức tạp như làm tươi dữ liệu, quản lý hàng đợi lệnh, và sắp xếp lại thứ tự yêu cầu.

Một điểm đáng chú ý trong thiết kế bộ điều khiển DDR3 là việc tối ưu hóa quá trình burst read/write. Đây là kỹ thuật cho phép đọc/ghi một khối dữ liệu lớn trong một chu kỳ giao tiếp duy nhất, từ đó giảm đáng kể overhead và cải thiện băng thông. Các nghiên cứu gần đây cũng chỉ ra rằng việc tích hợp thuật toán lập lịch lệnh tối ưu có thể tăng hiệu suất lên tới 20% [2].

2.5 So sánh các giao thức điều khiển bộ nhớ ngoài trên SoC

2.5.1 Giao thức AXI trong SoC

AXI là một phần của kiến trúc bus AMBA do ARM phát triển, được thiết kế để đáp ứng các yêu cầu cao về băng thông và độ trễ thấp trong các hệ thống SoC hiện đại. AXI hỗ trợ các tính năng như [4]:

Burst Transfers: Cho phép truyền nhiều dữ liệu liên tiếp trong một giao dịch, giảm overhead và tăng hiệu suất truyền dữ liệu.

Pipelining: Hỗ trợ thực hiện các giai đoạn của giao dịch song song, cải thiện thông lượng của hệ thống.

Separate Read and Write Channels: Cung cấp các kênh đọc và ghi độc lập, cho phép thực hiện đồng thời các hoạt động đọc và ghi mà không gây xung đột.

QoS Support: Hỗ trợ quản lý chất lượng dịch vụ, cho phép ưu tiên các giao dịch quan trọng và đảm bảo hiệu suất cho các ứng dụng yêu cầu cao.

AXI được thiết kế để hỗ trợ các hệ thống phức tạp với nhiều master và slave, cung cấp các cơ chế để quản lý truy cập và đảm bảo tính toàn vẹn của dữ liệu. Điều này làm cho AXI trở thành lựa chọn phổ biến trong các thiết kế SoC hiệu năng cao, đặc biệt là trong các hệ thống dựa trên vi xử lý ARM [3].

2.5.2 Giao thức Avalon

Avalon là giao thức bus được sử dụng trong các thiết kế FPGA, đặc biệt là với công cụ Platform Designer. Nó cung cấp một giao diện linh hoạt cho việc kết nối các thành phần trong hệ thống trên chip (SoC). Avalon hỗ trợ nhiều loại giao diện, bao gồm [11]:

Giao tiếp ánh xạ bộ nhớ: Cho phép các master truy cập trực tiếp vào không gian địa chỉ của các slave, thuận tiện cho việc giao tiếp với bộ nhớ và các thiết bị ngoại vi.

Giao tiếp luồng: Hỗ trợ truyền dữ liệu liên tục giữa các thành phần, hữu ích trong các ứng dụng xử lý tín hiệu số hoặc truyền thông.

Avalon được thiết kế với mục tiêu đơn giản hóa quá trình tích hợp hệ thống, cung cấp các tín hiệu điều khiển và dữ liệu cần thiết để đảm bảo giao tiếp hiệu quả giữa các thành phần. Tuy nhiên, do tính linh hoạt và đơn giản, Avalon có thể thiếu một số tính năng nâng cao như quản lý chất lượng dịch vụ (QoS)[11] hoặc hỗ trợ cho các cấu trúc hệ thống phức tạp.

2.5.3 Các đặc điểm nổi bật của AXI4 trong điều khiển bộ nhớ DDR

Hỗ trợ truyền dữ liệu hiệu quả với Burst Transfers: AXI4 hỗ trợ burst transfers lên đến 256 beats [3] trên mỗi giao dịch, nơi mỗi beat có thể có độ dài từ 8-bit đến 1024-bit. Điều này giúp giảm overhead giao tiếp và tăng hiệu quả sử dụng băng thông, đặc biệt quan trọng với các ứng dụng có lưu lượng dữ liệu lớn như truyền video hoặc xử lý tín hiệu.

Đọc và ghi độc lập với các kênh riêng biệt [4]: AXI4 phân tách các giao dịch đọc và ghi thành hai kênh riêng biệt Read Address Channel và Read Data Channel, Write Address Channel, Write Data Channel, và Write Response Channel. Điều này cho phép thực hiện các giao dịch đọc và ghi đồng thời, tăng thông lượng và giảm độ trễ.

AXI4 cho phép pipelining trên cả địa chỉ và dữ liệu. Các master có thể gửi nhiều yêu cầu mà không cần chờ phản hồi từ các giao dịch trước. Điều này tối ưu hóa sử dụng băng thông và giảm độ trễ, phù hợp cho các ứng dụng yêu cầu thời gian thực.

AXI4 dễ dàng tích hợp vào các hệ thống đa master và đa slave, cung cấp cơ chế arbitration (phân xử) hiệu quả. Điều này làm cho AXI4 trở thành lựa chọn phổ biến trong các hệ thống SoC phức tạp.

AXI4 được thiết kế để tối ưu hóa việc giao tiếp với các loại bộ nhớ ngoài như DDR, SRAM, hoặc NAND Flash [3]. Nó hỗ trợ các tính năng kiểm soát truy cập bộ nhớ hiệu quả và giảm thiểu xung đột truy cập.

2.6 Kết luận chương II

Chương II đã trình bày tổng quan về SoC, các thành phần cấu tạo, cũng như các loại bộ nhớ phổ biến được tích hợp trong SoC hiện đại. Trên cơ sở đó, chương này tập trung phân tích các giải pháp điều khiển bộ nhớ ngoài, đặc biệt là các bộ điều khiển bộ nhớ DDR. Qua việc so sánh các chuẩn giao tiếp phổ biến như AXI và Avalon, luận văn làm rõ lý do lựa chọn chuẩn AXI4 làm nền tảng giao tiếp trong thiết kế bộ điều khiển. Chuẩn AXI4 với khả năng hỗ trợ truyền dữ liệu hiệu quả [3], cơ chế burst linh hoạt và khả năng mở rộng tốt, là lựa chọn phù hợp cho hệ thống yêu cầu tốc độ cao và tính tùy biến mạnh mẽ như bộ điều khiển DDR3. Những kiến thức và phân tích trong chương này tạo tiền đề cho chương tiếp theo, tiến hành thiết kế và triển khai kiến trúc điều khiển DDR3 cụ thể trên nền SoC.

CHƯƠNG III: XÂY DỰNG HỆ THỐNG BỘ ĐIỀU KHIỂN DDRAM CONTROLLER

Chương này trình bày quá trình thiết kế và xây dựng bộ điều khiển bộ nhớ DDR3 SDRAM tương thích với giao thức AXI4. Các thành phần chính bao gồm kiến trúc tổng quan, thiết kế giao diện, máy trạng thái, và cơ chế truy xuất bộ nhớ. Việc triển khai sử dụng IP-Core MIG giúp đơn giản hóa việc tích hợp vào hệ thống SoC và đảm bảo hiệu suất cao, ổn định. Đây là bước trung gian quan trọng kết nối phần lý thuyết tổng quan với thiết kế cụ thể trong chương IV.

3.1 Tổng quan về IP-Core trong thiết kế bộ điều khiển DDR SDRAM

3.1.1 Tổng quan về IP-Core

IP-Core là những mô-đun thiết kế sẵn, được tạo ra để thực hiện các nhiệm vụ cụ thể trong việc xây dựng hệ thống tích hợp trên chip (SoC). Các khối này giống như những "mảnh ghép" công nghệ, cho phép tích hợp nhanh chóng vào các dự án lớn mà không cần phát triển lại từ đầu. Chúng thường được cung cấp dưới dạng mã nguồn phần cứng (như VHDL hoặc Verilog) bởi các công ty chuyên về vi mạch (như Xilinx, Altera, hay ARM), kèm theo hướng dẫn chi tiết để người dùng dễ dàng áp dụng.

Ưu điểm lớn nhất của IP-Core nằm ở khả năng tiết kiệm thời gian. Thay vì mất hàng tháng để thiết kế một chức năng phức tạp, kỹ sư chỉ cần chọn và tùy chỉnh một IP-Core phù hợp. Ngoài ra, vì đã được kiểm thử kỹ lưỡng, các khối này mang lại độ ổn định cao, giảm rủi ro lỗi trong hệ thống. Hơn nữa, tính linh hoạt của IP-Core cho phép tái sử dụng trong nhiều dự án khác nhau, từ đó cắt giảm chi phí phát triển và đẩy nhanh tiến độ sản xuất. Trong các công cụ thiết kế như Vivado, IP-Core xuất hiện dưới dạng thư viện, hỗ trợ người dùng điều chỉnh thông số như tốc độ hoặc giao diện trước khi triển khai trên FPGA.

3.1.2 Phân tích tổng quan lõi IP truy xuất bộ nhớ

Lõi IP điều khiển bộ nhớ là một phần không thể thiếu khi thiết kế bộ điều khiển cho DDR3 SDRAM, đảm bảo kết nối trơn tru giữa bộ xử lý và bộ nhớ ngoài. Khối này được tối ưu để xử lý các thao tác đọc/ghi dữ liệu một cách hiệu quả, đồng bộ với yêu cầu tốc độ cao của hệ thống SoC.

Các đặc trưng chính của lõi IP này bao gồm:

Kết nối qua bus AXI: Sử dụng giao thức AXI4 của ARM AMBA, lõi IP hỗ trợ bus dữ liệu 32-bit hoặc 64-bit. Giao thức này cho phép truyền dữ liệu theo chuỗi (burst), tăng tốc độ và giảm thời gian chờ, rất phù hợp với DDR3 SDRAM.

Tùy chỉnh thời gian: Người dùng có thể điều chỉnh các thông số như độ trễ CAS, thời gian làm mới, hoặc thời gian kích hoạt hàng để phù hợp với đặc tính của bộ nhớ DDR3, thông qua giao diện cấu hình hoặc thanh ghi nội bộ.

Giao tiếp vật lý: Lõi IP liên kết với DDR3 qua các tín hiệu như DQ (dữ liệu), DQS (đồng bộ dữ liệu), CLK (xung nhịp), và WDATA (dữ liệu ghi), đảm bảo truyền tải chính xác và hiệu quả.

Tính năng bổ sung: Một số lõi IP tích hợp khả năng sửa lỗi (ECC), tự động làm mới dữ liệu, và quản lý nhiều bank để nâng cao hiệu suất và độ tin cậy.

Chẳng hạn, trong Vivado của Xilinx, lõi IP Memory Interface Generator (MIG) được thiết kế để điều khiển DDR3. MIG hỗ trợ tần số lên đến 400 MHz, tạo mã nguồn tự động và dễ dàng kết nối với bus AXI, giúp đơn giản hóa quá trình thiết kế trên FPGA như Artix-7. Nhờ vậy, lõi IP không chỉ giảm công sức phát triển mà còn đảm bảo hệ thống hoạt động ổn định trong các ứng dụng thực tế như xử lý hình ảnh hoặc thiết bị nhúng.

3.2 Giao thức AXI4 trong điều khiển bộ nhớ ngoài DDR3 SDRAM

3.2.1.1 Các khái niệm quan trọng sử dụng trong AXI4

Transaction: Là một quá trình truyền dữ liệu từ master tới slaves thông qua bus. Transaction bao gồm đọc, ghi và cả phản hồi. Trong hệ thống SoC, Bus là thành phần chính của kết nối Master và slave. Nó giúp các master thực hiện các transaction đọc ghi dữ liệu bằng cách chuyển tín hiệu từ master đến slave và gửi phản hồi về từ slaves tới master thông qua bus. Bus có chức năng: Liên kết master và slave hoặc

giữa các bus trong hệ thống. Phân xử như một trọng tài về quyền truy cập. Đồng thời giúp giải mã địa chỉ.

Unaligned (không căn chỉnh): là hiện tượng địa chỉ của dữ liệu không khớp với dữ liệu. AXI4 hộ trỡ unaligned tức là hộ trợ hiện tượng địa chỉ của dữ liệu truyền đi không khớp với dữ liệu. Giao thức AXI sẽ hộ trỡ việ ghi dữ liệu ngay cả khi dữ liệu không được căn chỉnh.

Aligned: địa chỉ của dữ liệu khớp với kích thước của dữ liệu: ví dụ như một từ 32-bit được truyền tại một địa chỉ có thể chia hết cho 4.

Unaligned: Địa chỉ của dữ liệu không khớp với kích thước của dữ liệu, Ví dụ như một từ 32-bit được truyền tại một địa chỉ không chia hết cho 4 (ví dụ địa chỉ 0x03)

Burst length, burst size, burst type [3]: Trong giao thức AXI4, burst là một loại giao dịch trong đó nhiều word data được truyền liên tiếp trên bus, bắt đầu từ một địa chỉ cơ sở (base address). Burts transaction được sử dụng để tăng hiệu suất bằng cách giảm số lần cần truyền các tín hiệu điều khiển và tập trung vào truyền dữ liệu. Số lượng word dữ liệu sẽ được truyền trong một burst. Burst length được biểu diễn bằng một số nguyên từ 1 đến 256.

Kích thước của mỗi từ dữ liệu trong burst, thường được biểu diễn bằng số bit (như 8-bit, 16-bit, 32-bit, v.v.). Burst size xác định bao nhiêu byte sẽ được truyền trong mỗi chu kỳ của burst.

Incrementing Burst: Nếu master gửi một giao dịch ghi với burst length là 4 và burst size là 32-bit, nó sẽ truyền 4 từ dữ liệu liên tiếp, mỗi từ có kích thước 32-bit. Địa chỉ của các từ này sẽ tăng dần, ví dụ: 0x00, 0x04, 0x08, 0x0C

Fixed Burst: Nếu cùng một giao dịch ghi với burst length là 4 nhưng với burst type là fixed, tất cả 4 từ dữ liệu sẽ được ghi vào cùng một địa chỉ, ví dụ: 0x00, 0x00, 0x00, 0x00.

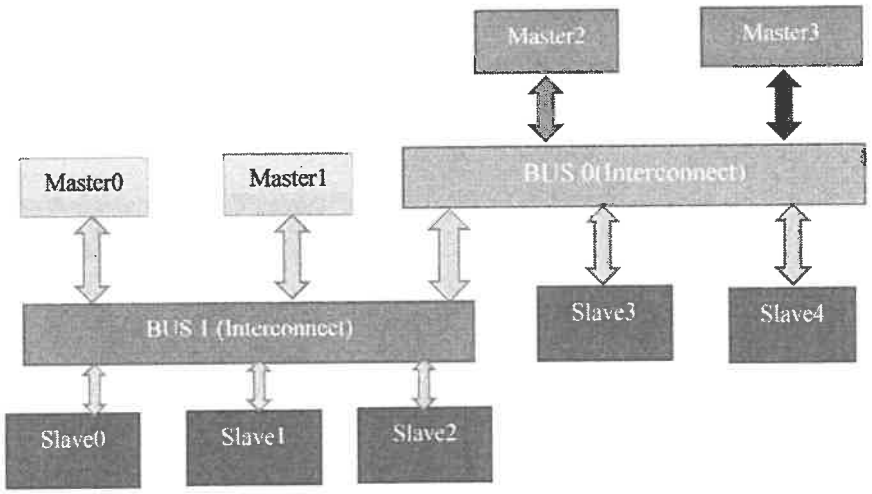
Multiple Out-standing: Trong giao thức AXI4 đề cập đến khả năng một master có thể phát hành nhiều giao dịch đọc hoặc ghi mà không cần phải đợi phản hồi từ các giao dịch trước đó. Điều này cho phép master khởi động nhiều transaction liên tiếp

trên bus, tăng cường khả năng sử dụng bus và cải thiện hiệu suất tổng thể của hệ thống.

3.2.1.2 Các quy định của giao thức AXI4

Giao thức AXI quy định về chuẩn giao tiếp (interface) giữa:

Một master và interconnect (bus), Một slave và bus, Một master và một slave (một master kết nối trực tiếp với một slave)



Hình 3. 1: Cấu trúc của giao thức AXI4

Interface là một nhóm các tín hiệu giao tiếp giữa hai thành phần. Interconnect (bus) là một thiết kế thường gồm nhiều interface, mỗi interface được kết nối với với một master, một slave hoặc một interconnect khác. Trong một interface giữa 2 interconnect, một bus sẽ đóng vai trò như master và bus còn lại có vai trò như slave. Xét Hình 3.1, giữa BUS 0 và BUS 1 có một giao tiếp (interface), nếu BUS 0 là master thì BUS 1 là slave hoặc ngược lại.

Cũng như các giao thức bus khác, AXI quy định cơ chế, cách thức xử lý cho 2 hoạt động cơ bản là:

Đọc (read hoặc read access): master lấy dữ liệu từ slave

Ghi (write hoặc write access): master phát dữ liệu cho slave

Đọc và ghi gọi chung là một truy cập (access). Một truy cập được hiểu là một transaction nên trong bài viết, đôi khi tác giả sử dụng một trong hai thuật ngữ này với cùng một ý nghĩa.

3.2.1 Quá trình đọc, ghi và cơ chế bắt tay của AXI4

3.2.2.1 Cơ chế phân kênh AXI4

AXI hoạt động dựa trên 5 loại kênh độc lập. "Độc lập" có nghĩa là mỗi kênh có vai trò, nhiệm vụ khác nhau và việc hoàn thành nhiệm vụ của mỗi kênh không bị phụ thuộc vào kênh khác chứ không có nghĩa là "không liên quan đến nhau". 5 kênh này gồm [3]:

Kênh địa chỉ đọc (Read address channel), gọi tắt là kênh AR, truyền thông tin địa chỉ và thông tin điều khiển của một transaction đọc từ master đến slave

Kênh dữ liệu đọc (Read data channel), gọi tắt là kênh R, truyền dữ liệu đọc và thông tin response của một transaction đọc từ slave đến master

Kênh địa chỉ ghi (Write address channel), gọi tắt là kênh AW, 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), gọi tắt là kênh W, truyền dữ liệu ghi từ master đến slave

Kênh đáp ứng ghi (Write response channel), gọi tắt là kênh B, truyền thông tin response của một transaction ghi từ slave đến master.

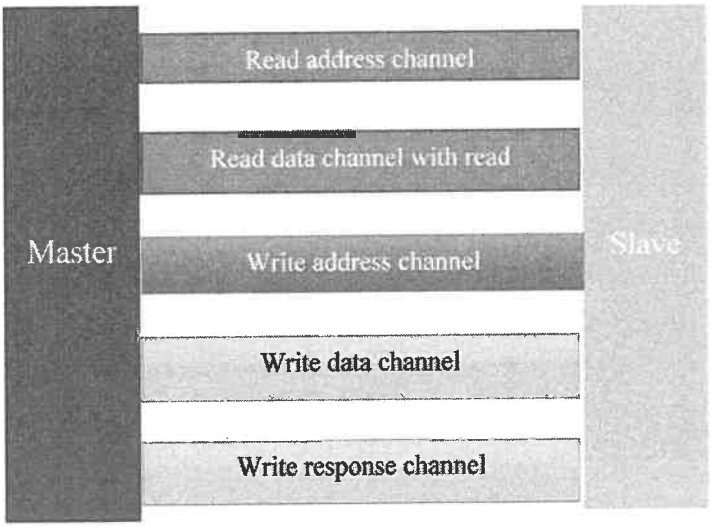
Một truy cập (access) hoặc một transaction bao gồm 4 thông tin cơ bản:

Thông tin địa chỉ (address) để xác định đích (vị trí, slave) mà transaction sẽ được xử lý.

Thông tin điều khiển (control) để xác định thuộc tính của transaction như loại burst, độ dài (length) burst, kích thước (size) burst và các thuộc tính khác.

Thông tin dữ liệu (data) của transaction

Thông tin response để xác định trạng thái của transaction có bị lỗi hay không



Hình 3. 2: Cơ chế đọc, ghi và phản hồi giữa master và slave trong AXI4

Một burst là toàn bộ các transfer dữ liệu trong một transaction. Mỗi transfer dữ liệu gọi là một beat. Mỗi burst gồm một hoặc nhiều beat. Mỗi transaction truyền quản lý một burst có thuộc tính được quy định bởi thông tin điều khiển.

Một transaction đọc được quản lý bởi 2 kênh:

Kênh AR, Kênh R

Một transaction ghi được quản lý bởi 3 kênh:

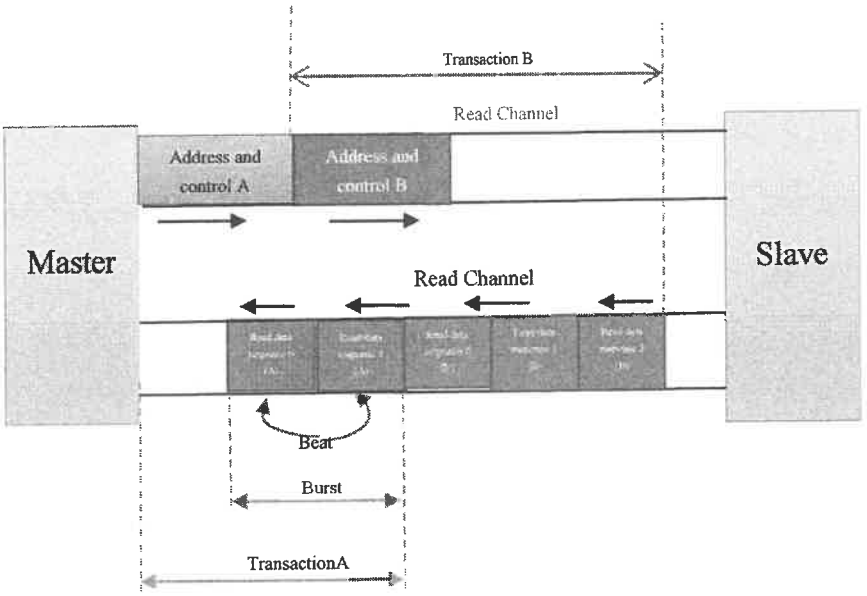
Kênh AW Kênh W Kênh B

3.2.2.2 Quá trình đọc dữ liệu

Một transaction đọc gồm hai bước xử lý:

Phía Master gửi một thông tin địa chỉ và thông tin điều khiển để khởi động một transaction đọc trên kênh AR.

Phía slave gửi các dữ liệu kèm thông tin response trên kênh R. Số lượng dữ liệu và thông tin response được quy định bởi thông tin điều khiển phát từ phía master trên kênh AR



Hình 3. 3: Ví dụ về quá trình đọc của AXI4 *[3]*

Hoạt động này thể hiện rõ đặc điểm "(1) AXI tách riêng pha truyền địa chỉ, thông tin điều khiển với pha truyền dữ liệu" đã nói phía trên.

Việc tách riêng này giúp giao dịch kế tiếp có thể được khởi tạo mà không cần phải chờ transaction trước hoàn thành. Ví dụ hình 4 thể hiện 2 giao dịch đọc được xử lý chồng lấn lên nhau. Điều này thể hiện đặc điểm "(5) Hỗ trợ phát nhiều địa chỉ chồng lấn (multiple outstanding)".

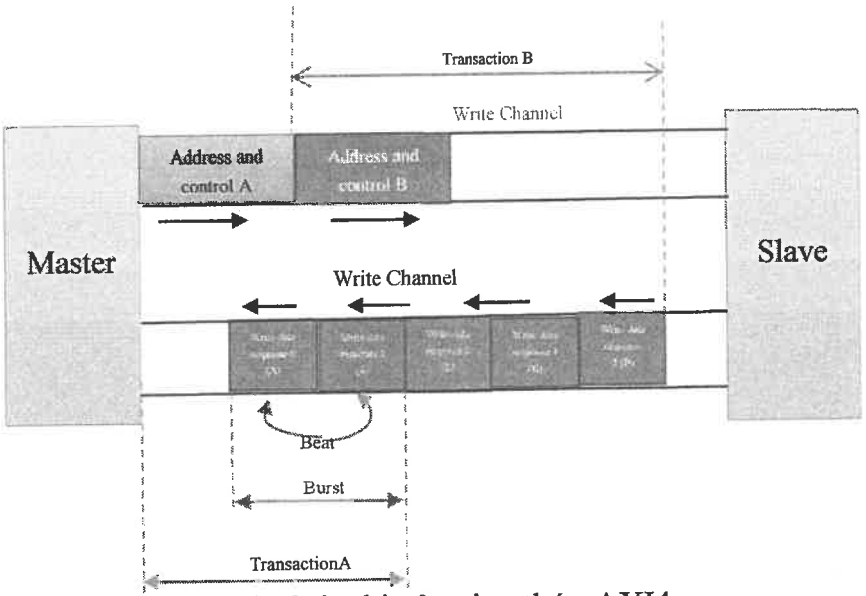
3.2.2.3 Quá trình ghi

Một transaction ghi gồm 3 bước xử lý:

Phía master gửi một thông tin địa chỉ và thông tin điều khiển để khởi động một transaction ghi trên kênh AW.

Phía master gửi các dữ liệu ghi trên kênh W. Số lượng dữ liệu ghi được quy định bởi thông tin điều khiển phát từ phía master trên kênh AW.

Phía slave gửi thông tin response khi burst ghi đã hoàn thành.



Hình 3. 4: Quá trình ghi của giao thức AXI4.

Phản hồi của giao dịch ghi chỉ phát khi burst đã hoàn thành và chỉ duy nhất một phản hồi được phát để báo hiệu cho một burst. Khác với giao dịch đọc, mỗi dữ liệu đọc kèm theo một phản hồi và giá trị phản hồi có thể khác nhau. Xét lại ví dụ hình 4, trong giao dịch A, phản hồi 0 có thể báo hiệu dữ liệu đọc thứ nhất không "không lỗi" nhưng phản hồi 1 có thể báo lỗi dữ liệu đọc.

3.2.2.4 Cơ chế bắt tay

Giao thức AXI hoạt động dựa trên cơ chế bắt tay hai chiều sử dụng một tín hiệu VALID và một tín hiệu READY.

Nguồn phát (master) sẽ sử dụng tín hiệu VALID để báo hiệu một địa chỉ, thông tin điều khiển hoặc dữ liệu đã hợp lệ trên kênh truyền. VALID giống như tín hiệu yêu cầu (request) một đích (slave) nhận thông tin.

Slave sẽ sử dụng tín hiệu READY để thông báo cho master biết nó đã chấp nhận thông tin từ source. READY giống như tín hiệu ACK xác nhận yêu cầu từ master.

Ví dụ, trong một transaction đọc, trên kênh AR, phía master là nguồn phát địa chỉ và thông tin điều khiển, phía slave là đích. Vì vậy, phía master sẽ điều khiển tín

hiệu VALID, phía slave sẽ điều khiển tín hiệu READY. Đối với kênh R thì ngược lại, slave là nguồn phát dữ liệu đọc, còn master là đích nên slave sẽ điều khiển VALID còn master điều khiển READY.

Mỗi kênh sẽ có một cặp VALID/READY gọi là cặp tín hiệu bắt tay để hoạt động theo cơ chế này. Cụ thể:

Kênh AR sẽ có tín hiệu ARVALID (master) và tín hiệu ARREADY (slave)

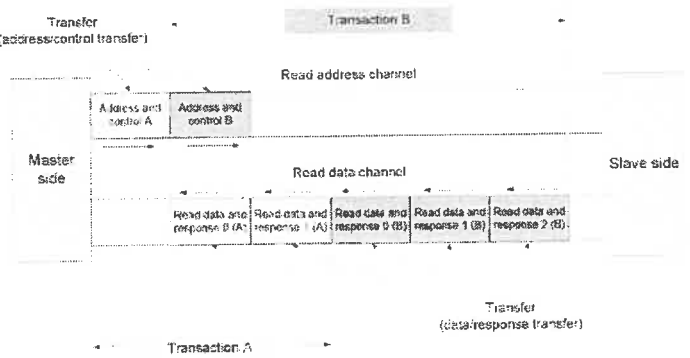
Kênh R sẽ có tín hiệu RVALID (slave) và tín hiệu RREADY (master)

Kênh AW sẽ có tín hiệu AWVALID (master) và tín hiệu AWREADY (slave)

Kênh W sẽ có tín hiệu WVALID (master) và tín hiệu WREADY (slave)

Kênh B sẽ có tín hiệu BVALID (slave) và tín hiệu BREADY (master)

Chú ý: VALID nói chung được hiểu là ARVALID, AWVALID, RVALID hoặc WVALID. READY được hiểu là ARREADY, AWREADY, RREADY hoặc WREADY.



Hình 3. 5: Quá trình yêu cầu và phản hồi trong cơ chế bắt tay của AXI4

Trên kênh AR, transfer địa chỉ và control của transaction B được phát ngay sau khi transfer địa chỉ và control của transaction A hoàn thành bất chấp các transfer dữ liệu và response của transaction A vẫn chưa hoàn thành trên kênh

3.2.2 Cơ chế burst transaction

Hỗ trợ các giao dịch theo cơ chế burst và chỉ cần phát địa chỉ đầu tiên của burst của giao thức AXI4. AXI4 là giao thức dựa trên cơ chế burst, mỗi transaction điều

khiển một burst nên transaction còn được gọi đầy đủ là burst transaction. Đặc tính của burst được quy định bởi thông tin điều khiển truyền trên kênh AR/AW.

Để khởi động một transaction, phía master sẽ phát địa chỉ và thông tin điều khiển của transaction trên kênh AR hoặc AW. Trong đó:

Địa chỉ được phát là địa chỉ của byte đầu tiên trong một transaction.

Thông tin điều khiển gồm thông tin quy định thuộc tính của burst và các thuộc tính khác của transasion.

Trong giao thức AXI, 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 transaction, 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 performance 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.

So sánh với giao thức bus APB (Advanced Peripheral Bus), một địa chỉ cụ thể chỉ ứng với một dữ liệu được đọc hoặc ghi. Điều này làm cho mạch logic xử lý giao tiếp APB đơn giản, chỉ cần giải mã địa chỉ có sẵn để sử dụng, nhưng hiệu năng bus là không cao.

Để giải thích cơ chế burst transaction, chúng ta tập trung vào các tín hiệu địa chỉ và điều khiển tại mỗi kênh như sau:

Kênh AR/AW

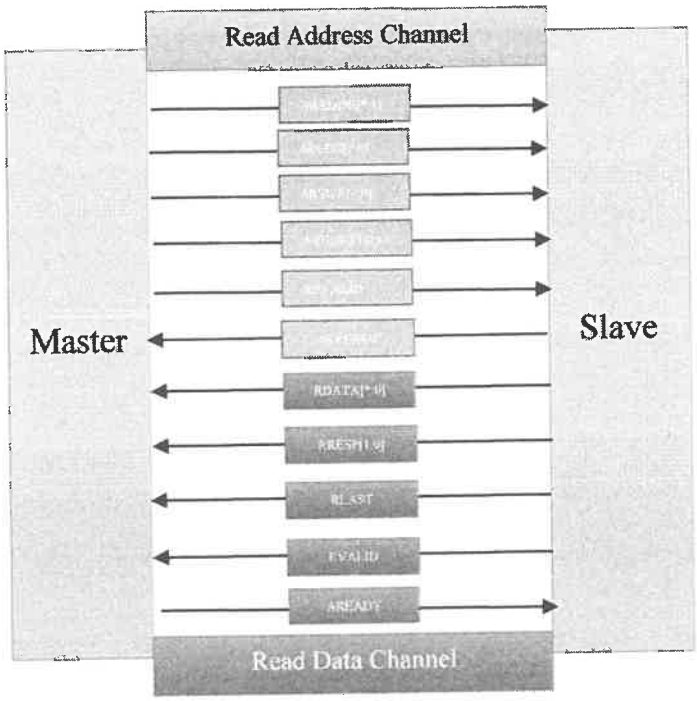
AxADDR (ARADDR/AWADDR) Địa chỉ của transfer dữ liệu đầu tiên của transaction.

AxLEN (ARLEN/AWLEN) Độ dài burst. Thông tin điều khiển này xác định số transfer dữ liệu trong burst, là số beat trong burst.

AxSIZE (ARSIZE/AWSIZE) Kích thước burst. Thông tin điều khiển này xác định kích thước chung của mỗi transfer dữ liệu trong burst.

AxBURST (ARBURST/AWBURST) Loại burst. Thông tin điều khiển này xác định phương pháp, cơ chế tính địa chỉ cho mỗi transfer dữ liệu (mỗi beat) trong burst. AxVALID (ARVALID/AWVALID) Tín hiệu VALID của kênh địa chỉ.

AxREADY (ARREADY/AWREADY) Tín hiệu READY của kênh địa ch



Hình 3. 6: Cơ chế busrt transaction

Kênh Read: Gồm RDATA Dữ liệu đọc, RLAST Tín hiệu báo transfer dữ liệu (beat) cuối cùng trong một burst đọc. RRESP Tín hiệu báo trạng thái của các transfer đọc. RVALID Tín hiệu VALID của kênh R. RREADY Tín hiệu READY của kênh R Kênh Write: WDATA Dữ liệu ghi, WLAST Tín hiệu báo transfer dữ liệu (beat) cuối cùng trong một burst ghi. WVALID Tín hiệu VALID của kênh W. WREADY Tín hiệu READY của kênh W

Kênh B: Phản hồi: BRESP Tín hiệu báo trạng thái của transaction ghi. BVALID Tín hiệu VALID của kênh B. BREADY Tín hiệu READY của kênh B Độ dài một burst (số beat) được quy định bởi AxLEN và tính như sau:

Độ dài burst (số beat) = AxLEN + 1

Số byte tối đa của một transfer dữ liệu (một beat) được quy định bởi AxSIZE. Kích thước này không được lớn hơn độ rộng bus dữ liệu (độ rộng RDATA/WDATA). Ví dụ, bus dữ liệu có độ rộng 32 bytes thì AxSIZE[2:0] phải nhỏ hơn hoặc bằng 5.

Bảng 4: Độ dài busrt của AXI4

AxSIZE[2:0]	Bytes in transfer
0b000	1
0b001	2
0b010	4
0b011	8
0b100	16
0b101	32
0b110	64
0b111	128

Loại burst được xác định bởi AxBURST. AXI hỗ trợ 3 loại burst là:

FIXED: Địa chỉ của mỗi transfer dữ liệu trong transaction là như nhau.

INCR: Địa chỉ của một transfer dữ liệu trong burst bằng địa chỉ của transfer trước đó tăng thêm một giá trị được quy định bởi AxSIZE. Ví dụ, AxSIZE[2:0] = 2, số byte của một transfer là 4 thì địa chỉ của transfer hiện tại bằng địa chỉ transfer trước tăng 4.

WRAP: Địa chỉ của một transfer dữ liệu trong burst bằng địa chỉ của transfer trước đó tăng thêm một giá trị được quy định bởi AxSIZE nhưng chỉ các bit địa chỉ LSB được tăng. Số lượng bit LSB được tăng phụ thuộc vào AxLEN và AxSIZE. Ví

dụ, khi AxLEN = 7, số transfer dữ liệu là 8, nếu AxSIZE = 2 (4 byte) thì số bit LSB được tăng là 5. Nghĩa là địa chỉ của transfer hiện tại sẽ bằng địa chỉ của transfer trước với 5 bits LSB [4:0] được cộng thêm 4. Chú ý, các bit địa chỉ [*:5] sẽ không đổi.

Địa chỉ bắt đầu phải aligned theo AxSIZE

AxLEN phải bằng 1, 3, 7, 15 ứng với độ dài burst là 2, 4, 8 và 16 beat.

Bảng 5 : Các loại Burst Type của AXI4

AxBurst[1:0]	Burst type
0b00	FIXED
0b01	INCR
0b10	WRAP
0b11	Reserved

Loại response được xác định bằng giá trị trên RRESP và BRESP. AXI hỗ trợ 4 loại response

Bảng 6: Bảng các kiểu trả về của slave

RRESP [1:0] BRESP[1:0]	Response
0b00	OKAY
0b01	EXOKAY
0b10	SLVERR
0b11	DECERR

3.2.3 Thiết kế giao diện AXI4

3.2.4.1 AXI4 tương tác với Slaves

Trong hệ thống sử dụng giao thức AXI4 (Advanced eXtensible Interface), dữ liệu và lệnh được truyền từ master đến các thiết bị ngoại vi hoặc bộ nhớ thông qua các kênh AXI4. AXI4 là một giao thức bus tiêu chuẩn trong các thiết kế hệ thống trên chip (SoC), được phát triển bởi ARM gồm các kênh sau:

Các kênh truyền của AXI4:

Write Address Channel (AW): Kênh này được sử dụng để truyền địa chỉ nơi mà dữ liệu ghi sẽ được lưu trữ, cùng với các tín hiệu điều khiển liên quan bao gồm tín hiệu clock, reset, tín hiệu cho phép ghi, …

Write Data Channel (W): Kênh này truyền dữ liệu thực tế từ master đến thiết bị slave để ghi vào địa chỉ đã được xác định trước đó trên kênh AW.

Write Response Channel (B): Sau khi hoàn tất việc ghi dữ liệu, thiết bị đích sử dụng kênh này để phản hồi lại master, thông báo rằng quá trình ghi đã hoàn tất và trạng thái của nó (thành công hoặc lỗi).

Read Address Channel (AR): Master sử dụng kênh này để truyền địa chỉ mà nó muốn đọc dữ liệu từ đó.

Read Data Channel (R): Dữ liệu được yêu cầu sẽ được thiết bị đích truyền ngược lại HOST thông qua kênh này, kèm theo tín hiệu điều khiển và trạng thái.

3.2.4.2 Phản hồi của slaves về cho master thông qua AXI4

Quá trình truyền dữ liệu và lệnh giữa HOST và thiết bị thông qua AXI4 diễn ra như sau:

Truyền Lệnh Ghi (Write Transaction):

Trước hết master gửi một lệnh ghi thông qua kênh AW, bao gồm địa chỉ đích và thông tin điều khiển như loại giao dịch, kích thước dữ liệu, v.v.

Master sau đó gửi dữ liệu cần ghi qua kênh W. Thiết bị slaves nhận dữ liệu và ghi vào vị trí bộ nhớ được xác định trước đó. Sau khi hoàn tất việc ghi, thiết bị master gửi phản hồi qua kênh B để thông báo trạng thái ghi cho master.

Truyền Lệnh Đọc (Read Transaction):

Master gửi một lệnh đọc qua kênh AR, chỉ định địa chỉ mà nó muốn đọc. Thiết bị slaves xử lý yêu cầu, đọc dữ liệu từ địa chỉ được chỉ định. Dữ liệu đọc được sẽ được gửi về master thông qua kênh R, kèm theo tín hiệu điều khiển và trạng thái.

Master nhận dữ liệu và xử lý tiếp theo

3.2.4.3 Khối điều khiển phân luồng của AXI4 (Arbiter)

Để tăng tốc độ đọc ghi dữ liệu cũng như ghi địa chỉ thì AXI4 đã tách đường đọc và đường ghi độc lập nhau. Trong quá trình ghi thì có cả ghi dữ liệu và ghi địa chỉ. Trong đọc dữ liệu thì cũng có kênh đọc độc lập nhau. Chúng hoạt động song song đồng thời nhau trong một chu kì. Chính vì thế khối arbiter ra làm một nhiệm vụ rất quan trọng trong giao thức AXI4. Giúp phân luồng và phân cấp độ ưu tiên cho quá trình đọc ghi dữ liệu

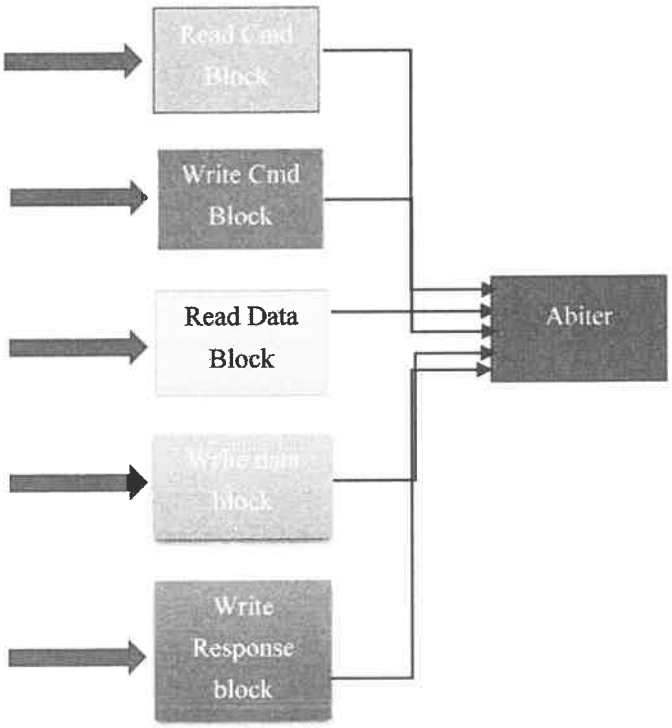
Phân luồng dữ liệu:

Trong AXI4, dữ liệu và địa chỉ được truyền trên các kênh độc lập:

Kênh Write Address (AW) và Write Data (W): Đảm nhiệm việc truyền địa chỉ và dữ liệu cần ghi từ master đến slave.

Kênh Read Address (AR) và Read Data (R): Đảm nhiệm việc truyền địa chỉ và dữ liệu cần đọc từ slave về master.

Khối phân luồng đóng vai trò trong việc phân xử các giao dịch này, đảm bảo rằng các yêu cầu đọc và ghi từ các master được xử lý một cách hiệu quả và không gây xung đột. Điều này đặc biệt quan trọng trong các hệ thống phức tạp, nơi nhiều master có thể yêu cầu truy cập đồng thời đến nhiều slave khác nhau.



Hình 3. 7: Khối điều khiển phân luồng của Arbiter

Phân cấp mức độ ưu tiên cho dữ liệu:

Arbiter không chỉ phân luồng các giao dịch mà còn quản lý độ ưu tiên của chúng. Dựa trên cấu hình hệ thống, arbiter có thể:

Ưu tiên các giao dịch Read: Trong một số trường hợp, việc ưu tiên giao dịch đọc là cần thiết, đặc biệt khi các dữ liệu cần đọc là đầu vào quan trọng cho các quy trình tính toán hoặc điều khiển.

Ưu tiên các giao dịch Write: Ngược lại, trong các hệ thống yêu cầu ghi dữ liệu liên tục hoặc ghi vào các thiết bị ngoại vi, giao dịch ghi có thể được ưu tiên hơn. Arbiter sử dụng các thuật toán như Round Robin hoặc Fixed Priority để quyết định thứ tự xử lý các yêu cầu, đảm bảo rằng các master với các mức độ ưu tiên khác nhau đều được phục vụ một cách hợp lý.

Việc tách biệt các kênh đọc và ghi cho phép khối arbiter tối ưu hóa băng thông của hệ thống. Các giao dịch đọc và ghi không phải chờ đợi nhau, mà có thể tiến hành đồng thời, do đó giảm thiểu độ trễ trong truyền tải dữ liệu.

Pipelining được hỗ trợ bởi arbiter cho phép nhiều giao dịch ở các giai đoạn khác nhau được xử lý đồng thời, từ đó tăng cường khả năng xử lý của hệ thống.

Quản lý burst và giảm xung đột:

Khối phân xử cũng quản lý các giao dịch burst, trong đó một loạt dữ liệu liên tiếp được đọc hoặc ghi. Khối phân xử phải đảm bảo rằng các giao dịch burst được xử lý liền mạch và không bị gián đoạn, đồng thời tối ưu hóa việc sử dụng băng thông. Burst transactions: là một loại giao dịch đặc biệt cho phép truyền nhiều dữ liệu liên tiếp trong một lần khởi tạo giao dịch duy nhất. Thay vì khởi tạo một giao dịch riêng biệt cho mỗi dữ liệu, burst transactions cho phép master gửi hoặc nhận một loạt dữ liệu liên tiếp với chi phí overhead thấp hơn, từ đó tăng hiệu suất truyền tải.

Trong trường hợp nhiều master yêu cầu truy cập đến cùng một slave đồng thời, khối phân xử sẽ quyết định thứ tự xử lý dựa trên độ ưu tiên và các quy tắc được cấu hình trước. Nếu xảy ra xung đột, khối phân xử có thể tạm dừng hoặc hoãn một yêu cầu để xử lý yêu cầu khác, sau đó tiếp tục với các yêu cầu còn lại

3.2.4.4 Quá trình lưu dữ liệu FIFO

Kiến truc dữ liệu first in, first out.

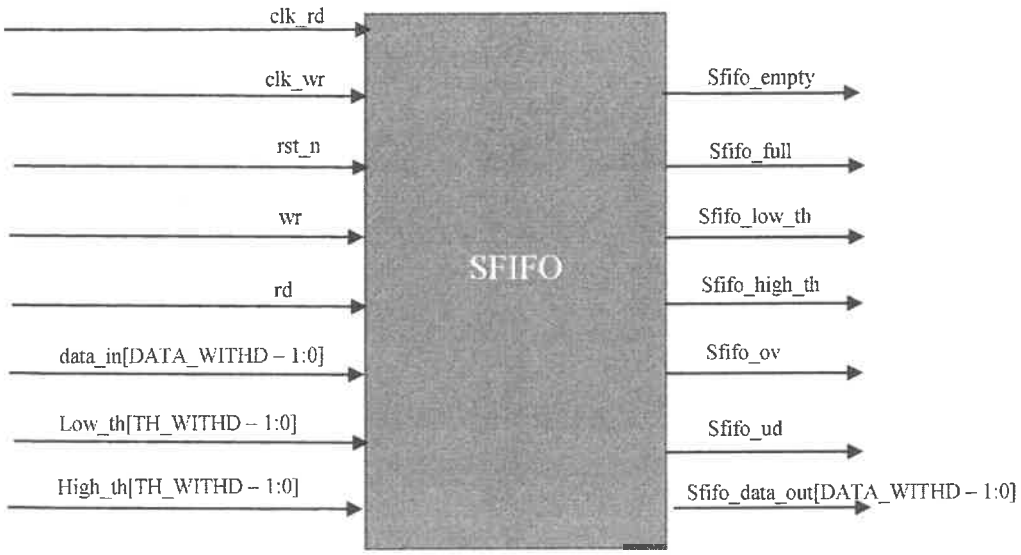
FIFO hoạt động theo nguyên tắc dữ liệu được lưu vào trước sẽ là dữ liệu được lấy ra trước. Nghĩa là, thứ tự dữ liệu được đọc ra giống như thứ tự dữ liệu được ghi vào. Hai thông số đặc trưng của FIFO là:

Số lượng ô nhớ hay còn gọi là độ sâu của FIFO

Độ rộng ô nhớ tương ứng với độ rộng dữ liệu được ghi vào và đọc ra.

Hai thông số trên sẽ cho biết dung lượng của FIFO. Ví dụ, FIFO có 4 ô nhớ, mỗi ô nhớ lưu 8 bit dữ liệu thì dung lượng FIFO là 4*8 = 32 bit hoặc 4 byte. Để định địa chỉ các ô nhớ của FIFO, một phương pháp thường dùng là sử dụng con trỏ ghi và con trỏ đọc. Con trỏ ở đây thực chất là bộ đếm tuần tự

Sơ đồ giao tiếp tín hiệu của fifo:



Hình 3. 8: Sơ đồ giao tiếp tín hiệu của FIFO

Đầu vào (input)

clk_rd: xung clock đọc dữ liệu

clk_wr: xung clock ghi dữ liệu, đồng bộ với xung clock đọc clk_rd

rst_n: tín hiệu reset tích cực mức thấp

wr: tín hiệu ghi dữ liệu vào FIFO

rd: tín hiệu đọc dữ liệu từ FIFO

data_in[DATA_WIDTH-1:0]: bus dữ liệu ghi có số bit là DATA_WIDTH

low_th[TH_WIDTH-1:0]: tín hiệu cấu hình mức ngưỡng dưới có số bit là TH_WIDTH.

high_th[TH_WIDTH-1:0]: tín hiệu cấu hình mức ngưỡng trên có số bit là TH_WIDTH.

Đầu ra (output)

sfifo_empty: tín hiệu báo FIFO rỗngsfifo_full: tín hiệu báo FIFO đầy

sfifo_low_th: tín hiệu báo FIFO thấp hơn mức ngưỡng dưới

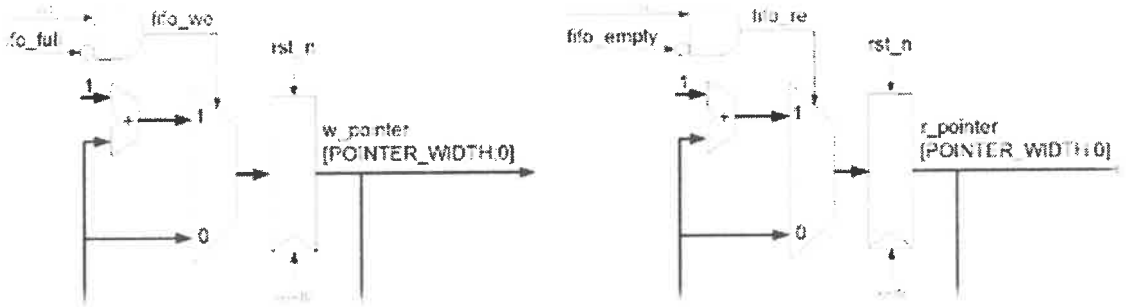
sfifo_high_th: tín hiệu báo FIFO cao hơn hoặc bằng mức ngưỡng trên

sfifo_ov: tín hiệu báo FIFO bị overflow

sfifo_ud: tín hiệu báo FIFO bị underflow

sfifo_data_out[DATA_WIDTH-1:0]: bus dữ liệu đọc có số bit là DATA_WIDTH[12]

Quá trình đọc/ghi dữ liệu vào FIFO



Hình 3. 9: Quá trình đọc dữ liệu vào FIFO

Hai tín hiệu báo hiệu đầy đó là:

Ghi khi FIFO đã đầy, sfifo_full=1

Đọc khi FIFO đã rỗng, sfifo_empty=1

Hai điều trên có thể thực hiện được trong một trường hợp đặc biệt khi cả tín hiệu đọc (rd) và tín hiệu ghi (wr) cùng tích cực khi FIFO đang đầy hoặc đang rỗng. Cụ thể:

Khi FIFO đầy (sfifo_full=1), nếu wr=rd=1 thì dữ liệu cần đọc sẽ được lấy ra khỏi FIFO và dữ liệu cần ghi sẽ được ghi vào FIFO ở cùng 1 cạch xung clock. Lúc này, cả hai con trỏ đọc và ghi cùng tăng. Trạng thái FIFO đầy vẫn được duy trì vì FIFO được lấy ra 1 dữ liệu nhưng đồng thời cũng nạp thêm 1 dữ liệu mới.

Khi FIFO rỗng (sfifo_empty=1), nếu wr=rd=1 thì dữ liệu cần đọc sẽ được truyền trực tiếp từ ngõ vào FIFO đến ngõ ra FIFO mà không cần lưu vào bộ nhớ trung gian mem_array. Trạng thái FIFO rỗng vẫn được duy trì vì FIFO không lưu giá trị dữ liệu mới. Lúc này, cả hai con trỏ đọc và ghi không thay đổi giá trị.

Việc đồng bộ và quản lý dữ liệu nhờ FIFO

Đồng Bộ Hóa Dữ Liệu Giữa Các Tốc Độ Khác Nhau

Tốc Độ Khác Nhau: FIFO cho phép các hệ thống hoặc thành phần hoạt động với các tốc độ khác nhau để giao tiếp hiệu quả. Ví dụ, một thành phần có thể hoạt

động với tốc độ cao hơn hoặc thấp hơn so với thành phần khác. FIFO lưu trữ dữ liệu tạm thời giúp bù đắp sự khác biệt về tốc độ giữa các thành phần.

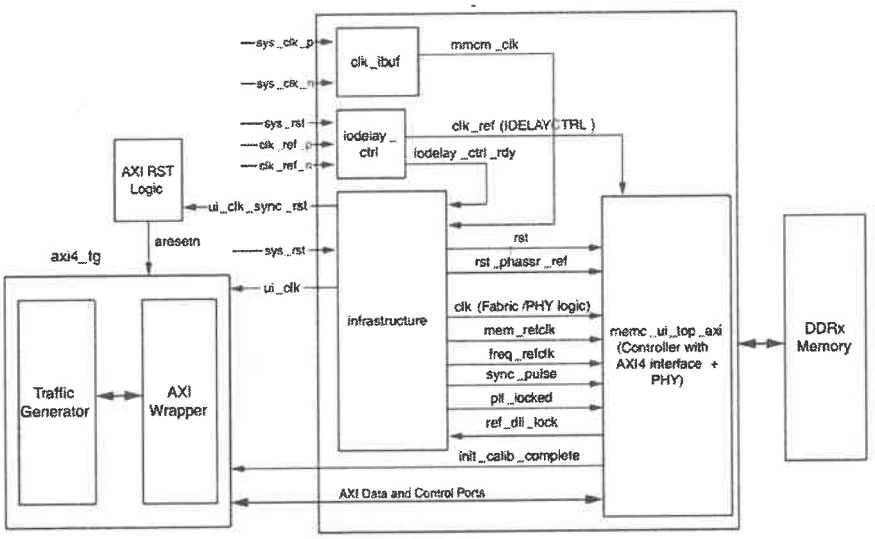
Bộ Đệm Tạm Thời: FIFO hoạt động như một bộ đệm tạm thời, giữ dữ liệu giữa các giai đoạn của hệ thống, giúp đồng bộ hóa dữ liệu khi tốc độ đọc và ghi không khớp nhau. Điều này giúp hệ thống duy trì sự liên tục và tránh tình trạng mất mát dữ liệu.

Bộ Đệm Giao Dịch: Trong hệ thống sử dụng giao thức AXI4, FIFO được sử dụng để làm bộ đệm giữa các giao dịch đọc và ghi. FIFO giúp quản lý dữ liệu giữa các master và slave có thể có tốc độ khác nhau hoặc xử lý đồng thời nhiều giao dịch. Giảm Thiểu Độ Trễ: FIFO giúp giảm độ trễ trong việc truyền dữ liệu giữa các phần của hệ thống. Khi dữ liệu được lưu trữ trong FIFO, nó có thể được xử lý hoặc truyền tiếp mà không cần phải chờ đợi quá lâu.

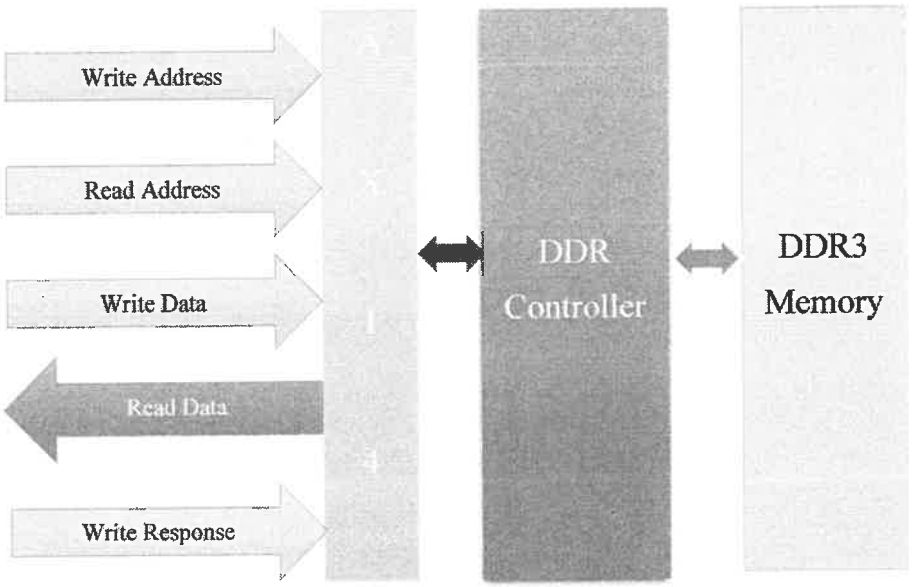
3.3 Xây dựng bộ điều khiển bộ nhớ DDR3 SRAM

3.2.1 Kiến trúc tổng quan bộ điều khiển DDR3 SDRAM

Bộ điều khiển DDR3 hoạt động như một cầu nối quan trọng giữa Master AXI và DDR3 SDRAM. Nó đảm nhiệm việc khởi tạo DDR3 và các yêu cầu về thời gian khác nhau của bộ nhớ DDR3 [10]. Bộ điều khiển DDR3 thực hiện nhiều sơ đồ để tăng thông lượng bộ nhớ hiệu quả. Để đạt được mức tối đa thông lượng từ bộ nhớ, nó vận hành tất cả banks bộ nhớ song song và giảm thiểu ảnh hưởng nạp lại/làm mới và các hoạt động nội bộ khác của DDR3. Giao diện giữa giao thức AXI và Bộ nhớ DDR3 dành cho kiến trúc SoC. Thiết kế bao gồm các khối sau: Giao diện AXI, trình quản lý truy cập AXI, bộ điều khiển DDR3 [4].



Hình 3. 10: Bộ điều khiển bộ nhớ DDR3 SDRAM được gen ra từ giao diện *[1]*



Hình 3. 11: Sơ đồ khối thiết kế bộ điều khiển DDR3 SDRAM dùng giao diện AXI4

Giao diện AXI4

Giao diện AXI4 đóng vai trò là cầu nối giữa các thiết bị master và slave, cho phép truyền dữ liệu hiệu quả thông qua năm kênh tín hiệu độc lập: địa chỉ ghi (AW), địa chỉ đọc (AR), dữ liệu ghi (W), dữ liệu đọc (R), và phản hồi ghi (B). Mỗi kênh được thiết kế hoạt động song song nhằm tối ưu hóa thông lượng và giảm độ trễ hệ thống.

Đặc biệt, AXI4 hỗ trợ nhiều kiểu truyền burst với độ dài linh hoạt, từ các giao dịch đơn lẻ (single beat) đến các burst dài hơn. Giao diện cũng yêu cầu tuân thủ nghiêm ngặt các đặc tả về định dạng và thời gian, đảm bảo dữ liệu được truyền đúng địa chỉ và thứ tự.

Bộ quản lý truy cập AXI4.

Bộ quản lý truy cập AXI4 đóng vai trò trung gian, chuyển đổi các lệnh từ giao thức AXI4 linh hoạt thành các giao dịch tương thích với cấu trúc truy cập cố định của bộ nhớ DDR3. Sự khác biệt về cơ chế burst giữa hai giao thức – AXI4 cho phép burst linh hoạt, trong khi DDR3 giới hạn ở burst 4 hoặc burst 8 – đặt ra yêu cầu thiết kế hệ thống gom nhóm và tái cấu trúc các lệnh nhằm tối ưu hóa băng thông và giảm thiểu độ trễ truy cập. Bộ quản lý truy cập AXI4 gồm:

AXI-IF: Giao diện tiếp nhận và phân tích các lệnh từ bus AXI4.

Khối điều khiển lưu trữ: Làm nhiệm vụ đệm tạm và tổ chức lệnh trước khi xử lý.

Burst Manager: Tái cấu trúc và điều chỉnh độ dài burst phù hợp với yêu cầu DDR3, đảm bảo tính hiệu quả trong truy cập bộ nhớ.

Khối điều khiển địa chỉ: Tạo ra địa chỉ bộ nhớ chính xác, duy trì tính liên tục và tránh xung đột truy cập.

Control Logic: Điều phối hoạt động giữa các khối chức năng, đảm bảo đồng bộ toàn hệ thống.

Bộ điều khiển bộ nhớ DDR3 SDRAM

Bộ điều khiển bộ nhớ DDR3

Là thành phần cốt lõi giao tiếp trực tiếp với bộ nhớ vật lý, bộ điều khiển DDR3 thực hiện các thao tác đọc/ghi dữ liệu cũng như quản lý các chức năng đặc thù như: làm mới (Refresh), tự làm mới (Self-refresh), và tiết kiệm điện (Power-down). Các chức năng này được thực hiện theo cấu hình, cho phép tối ưu giữa hiệu suất và năng lượng tiêu thụ.

Bộ điều khiển có khả năng chuyển đổi lệnh từ AXI4 thành các chuỗi lệnh DDR3 như: kích hoạt hàng (ACT), truy cập cột (READ/WRITE), và đóng hàng

(PRECHARGE). Thiết kế hỗ trợ truy cập đa kênh (multi-channel), giúp tăng cường băng thông trong các ứng dụng yêu cầu cao như xử lý tín hiệu số hay hệ thống nhúng thời gian thực.

3.2.2 Thiết kế khối điều khiển truy cập địa chỉ

3.2.2.1 Cấu trúc địa chỉ của AXI4 và DDR3 SDRAM

AXI4 là một giao thức bus tiên tiến, hỗ trợ định dạng địa chỉ linh hoạt với độ dài 32-bit hoặc 64-bit tùy cấu hình hệ thống. Mỗi giao dịch (transaction) trong AXI4 bao gồm:

Địa chỉ cơ sở (Base Address): Xác định điểm bắt đầu truyền dữ liệu.

Kích thước dữ liệu (Data Size): Thường từ 1 đến 128 byte cho mỗi beat truy cập.

Chế độ burst: Cho phép truyền nhiều beat trong một giao dịch, bao gồm:

INCR: địa chỉ tăng dần sau mỗi beat.

FIXED: địa chỉ giữ nguyên.

WRAP: địa chỉ quay vòng trong phạm vi giới hạn nhất định.

Transaction ID (ID Tag): Cho phép phân biệt các giao dịch đồng thời từ nhiều master, hỗ trợ kiến trúc đa luồng không xung đột.

DDR3, ngược lại, tổ chức bộ nhớ theo cấu trúc phân cấp gồm:

Bank Address (BA): Chọn vùng bộ nhớ tương ứng.

Row Address: Chỉ định hàng trong bank.

Column Address: Chỉ định cột trong hàng.

Chip Select (CS#): Dùng khi có nhiều chip DDR3 trên hệ thống.

Khối điều khiển truy cập địa chỉ thực hiện phân tích cú pháp địa chỉ AXI4, sau đó chuyển đổi và tái cấu trúc địa chỉ thành các thành phần tương ứng trong không gian địa chỉ DDR3 (Bank, Row, Column). Quá trình chuyển đổi này được thực hiện dựa trên các yếu tố như: Kích thước giao dịch. Độ dài và kiểu burst. Căn chỉnh địa chỉ theo hàng và bank.

3.2.2.2 Cơ chế chuyển đổi địa chỉ từ AXI4 sang DDR3

Quá trình chuyển đổi địa chỉ được thực hiện bằng cách trích xuất các bit địa chỉ từ AXI4 và ánh xạ tương ứng theo định dạng của DDR3, dựa trên cấu hình kích thước của từng vùng địa chỉ (số lượng bank, số hàng, số cột). Ví dụ: nếu hệ thống DDR3 có 8 bank, 16K hàng và 1K cột, thì địa chỉ AXI4 cần được chia thành:

3 bit cho Bank ($\log_2(8) = 3$),

14 bit cho Row ($\log_2(16K) = 14$),

10 bit cho Column ($\log_2(1K) = 10$),

Các bit còn lại được dùng để căn chỉnh dữ liệu theo burst hoặc phục vụ các chức năng điều khiển khác Giao thức AXI4 hỗ trợ nhiều chế độ truyền dữ liệu theo burst (burst transfer), mỗi chế độ yêu cầu chiến lược ánh xạ địa chỉ tương ứng: INCR Burst (Tăng dần), WRAP Burst (Quay vòng), FIXED Burst (Giữ nguyên)

3.2.3 Thiết kế khối điều khiển dữ liệu

3.2.3.1 Thành phần chính của khối điều khiển

Khối điều khiển dữ liệu chịu trách nhiệm đảm bảo việc truyền tải dữ liệu giữa bus AXI4 và bộ nhớ DDRAM3. Đảm bảo việc dữ liệu được đọc và ghi chính xác, kịp thời và đáp ứng được yêu cầu về băng thông, đột rễ.

Thành phần chính của khối điều khiển dữ liệu:

Bộ đệm dữ liệu vô cùng quan trọng được sử dụng để lưu trữ tạm thời dữ liệu đọc/ghi nhằm tối ưu hoá hiệu suất bus. Khi ghi dữ liệu, bộ đệm giúp lưu trữ dữ liệu để giảm độ trễ chờ từ bộ nhớ DDRAM3. Khi đọc dữ liệu, bộ đệm giúp lưu trữ dữ liệu để giảm độ trễ chờ từ bộ nhớ DDRAM3.

Bộ điều phối dữ liệu chịu trách nhiệm lên lịch truyền tải dữ liệu dựa trên trạng thái của bộ điều khiển. Cho phép dữ liệu được gửi theo đúng thứ tự và không vi phạm về quy tắt refresh hoặc latency của DDR3.

Bộ kiểm soát độ tin cậy để hộ trợ các cơ chế kiểm soát lỗi ECC (Error Correction Code) để phát hiện và sửa lỗi dữ liệu khi cần thiết. Kiểm tra và xác nhận tính toàn vẹn của dữ liệu trước khi gửi đến AXI4 và DDRAM3.

3.2.3.2 Cơ chế truyền tải dữ liệu từ AXI4 với DDR3

Quá trình ghi bắt đầu khi AXI4 gửi yêu cầu ghi qua các kênh địa chỉ và dữ liệu. Bộ điều khiển tiếp nhận, giải mã và lưu dữ liệu tạm thời vào bộ đệm nội trong khi chờ DDR3 sẵn sàng. Khi bộ nhớ đã được kích hoạt và không bận, lệnh ghi được phát đến DDR3 và dữ liệu được truyền theo burst – thường là burst 4 hoặc 8. Sau khi ghi xong, tín hiệu xác nhận được gửi về thiết bị AXI4 qua kênh phản hồi ghi để báo hoàn tất giao dịch, giúp tiếp tục các yêu cầu tiếp theo mà không bị gián đoạn.

Đối với quá trình đọc, thiết bị AXI4 khởi tạo giao dịch thông qua kênh địa chỉ đọc. Nếu dữ liệu đã có sẵn trong bộ đệm (nhờ truy cập trước đó hoặc cơ chế prefetch), bộ điều khiển sẽ phản hồi ngay lập tức qua kênh dữ liệu đọc, rút ngắn độ trễ đáng kể. Nếu không, lệnh đọc được gửi tới DDR3, dữ liệu sau đó được lấy về, lưu tạm trong bộ đệm rồi mới truyền qua AXI4 kèm theo tín hiệu kết thúc.

Cơ chế bộ đệm và burst không chỉ giúp giảm độ trễ và tăng thông lượng, mà còn hỗ trợ xử lý các yêu cầu đồng thời từ nhiều thiết bị chủ dựa trên Transaction ID.

Hệ thống còn có khả năng thích ứng với các kiểu truy cập đặc biệt như FIXED hoặc WRAP, đảm bảo tính linh hoạt. Tuy nhiên, cũng có những thách thức như nguy cơ tràn bộ đệm hoặc độ trễ cao khi DDR3 chưa sẵn sàng. Do đó, việc cấu hình hợp lý – như mở rộng bộ đệm hoặc tối ưu hóa thuật toán prefetch – là cần thiết để đảm bảo hiệu quả tối đa cho từng ứng dụng cụ thể.

3.2.3.3 Cải thiện hiệu suất truyền tải dữ liệu

Prefetching (dự đoán dữ liệu) là một kỹ thuật tối ưu hóa nhằm giảm độ trễ truy cập bộ nhớ DDR3 bằng cách chủ động nạp trước dữ liệu vào bộ đệm nội bộ, dựa trên các mẫu truy cập được phát hiện từ các giao dịch AXI4. Thay vì chờ đến khi thiết bị chủ gửi yêu cầu cụ thể, bộ điều khiển sử dụng cơ chế phân tích mẫu truy cập (access pattern recognition) để phát hiện các chuỗi truy cập tuần tự hoặc định kỳ, từ đó đưa ra quyết định thực hiện các lệnh đọc trước một cách chủ động.

Cụ thể, trong các chuỗi đọc tuyến tính với burst INCR hoặc các truy cập lặp lại tại địa chỉ cố định (FIXED), bộ điều khiển có thể nhận diện xu hướng truy cập tiếp theo dựa trên lịch sử giao dịch gần nhất. Ví dụ, trong một chuỗi đọc burst 4 từ AXI4,

bộ điều khiển có thể chủ động gửi lệnh đọc burst 8 đến DDR3, bao gồm cả dữ liệu hiện tại và các từ dữ liệu kế tiếp. Dữ liệu này sau đó được lưu trữ trong bộ đệm để sẵn sàng phục vụ các yêu cầu tiếp theo mà không cần phải chờ chu kỳ truy cập DDR3, vốn có độ trễ cao do các bước khởi tạo, read và precharge.

Lợi ích chính của prefetching là giảm thiểu độ trễ truy cập DDR3 – vốn là điểm nghẽn hiệu năng chủ yếu trong các hệ thống bộ nhớ DRAM – đồng thời tăng khả năng đáp ứng và throughput của hệ thống. Tuy nhiên, một điểm hạn chế cần lưu ý là trong trường hợp dự đoán sai, dữ liệu được nạp trước có thể không được sử dụng, gây lãng phí tài nguyên bộ đệm và băng thông DDR3. Do đó, việc thiết kế chiến lược prefetch phải cân nhắc giữa độ chính xác của dự đoán và chi phí phần cứng liên quan.

Out-of-order Execution (thực thi không theo thứ tự) là một cơ chế tối ưu hóa cho phép bộ điều khiển bộ nhớ sắp xếp lại thứ tự thực thi các yêu cầu đọc và ghi từ giao thức AXI4, từ đó khai thác hiệu quả hơn băng thông DDR3 và giảm độ trễ trung bình trong truy cập bộ nhớ. Trong môi trường DDR3, việc chuyển đổi giữa các hàng khác nhau trong cùng một bank gây ra độ trễ đáng kể do yêu cầu đóng hàng hiện tại và kích hoạt hàng mới. Nếu các yêu cầu được xử lý tuần tự theo thứ tự nhận được từ AXI4 mà không tính đến trạng thái hiện tại của DDR3, hệ thống có thể gặp phải các chu kỳ chờ không cần thiết, ảnh hưởng đến hiệu năng tổng thể.

Để khắc phục điểm yếu này, cơ chế Out-of-order Execution sử dụng một hàng đợi lệnh (command queue) kết hợp với bảng trạng thái hàng và bank (status table) để phân tích và sắp xếp lại các yêu cầu sao cho phù hợp với trạng thái hiện tại của DDR3. Cụ thể, hệ thống ưu tiên các lệnh nhắm đến hàng đang được mở (open row) hoặc các bank đang ở trạng thái rảnh (idle), nhằm tránh các thao tác precharge và activate không cần thiết. Ví dụ, nếu trong hàng đợi có một lệnh đọc từ hàng A của bank 0 và hàng A đang mở, bộ điều khiển sẽ chọn thực thi lệnh này trước, mặc dù nó có thể đến sau một lệnh ghi nhắm đến hàng B cũng trong bank 0. Chiến lược này giúp giảm đáng kể độ trễ do chuyển đổi hàng và tăng throughput của hệ thống.

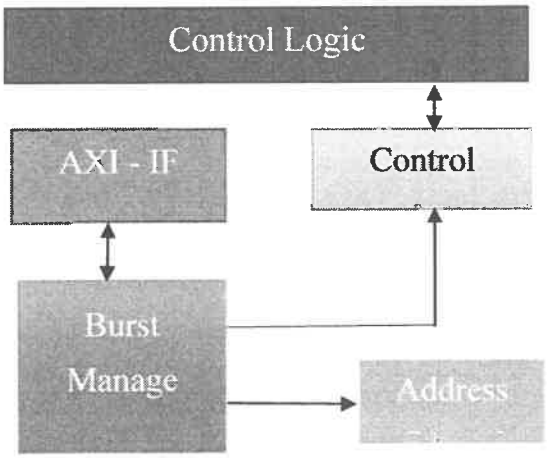
Để đảm bảo tính chính xác và nhất quán của dữ liệu, đặc biệt trong môi trường đa master, hệ thống sử dụng Transaction ID của giao thức AXI4 để theo dõi và duy

trì thứ tự hoàn thành của các yêu cầu khi cần thiết. Điều này giúp đảm bảo rằng các giao dịch thuộc cùng một luồng (stream) hoặc cùng một master sẽ được xử lý một cách nhất quán, ngay cả khi lệnh bị thực thi không theo thứ tự ban đầu.

3.4 Thiết kế bộ quản lý truy cập DDR3 SDRAM

3.4.1 Tổng quan về quản lý truy cập AXI

Nhiệm vụ quan trọng nhất của bộ quản lý truy cập AXI là chuyển đổi các lệnh AXI thành các lệnh truy cập bộ nhớ để tối đa hóa việc sử dụng băng thông DDR3. Bộ nhớ DDR3 chỉ nhận lệnh ở chế độ burst 4 hoặc 8, trong khi lệnh AXI burst nhỏ hơn hoặc dài hơn. Bộ quản lý truy cập AXI kết hợp các lệnh bất cứ khi nào có thể để cải thiện hiệu suất và chuyển lệnh cuối cùng đến bộ điều khiển DDR3. Để đảm bảo băng thông là tối đa, nó tiền nạp các lệnh từ giao diện AXI, dịch chúng thành các giao dịch bộ nhớ và lưu trữ cục bộ.

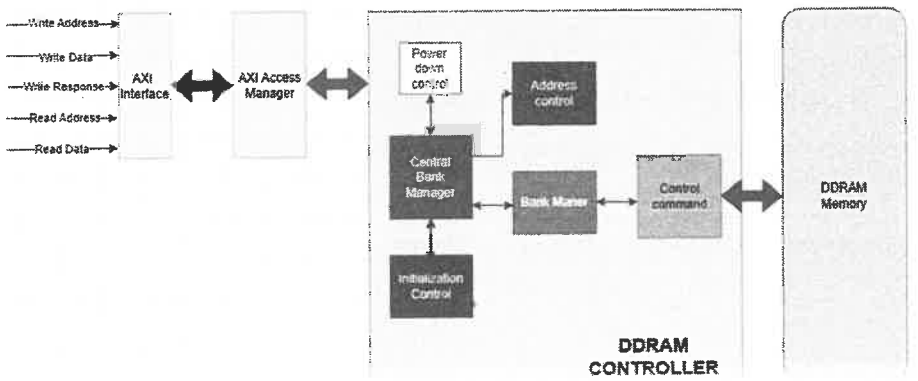


Hình 3. 12: Tổng quan về bộ truy cập AXI

Các lệnh tích lũy này được cung cấp cho bộ điều khiển DDR3 bất cứ khi nào Bộ Điều Khiển DDR3 [10] không busy trong chu kỳ xung tiếp theo. Khối AXI-IF phối hợp với khối giao diện AXI và chấp nhận các lệnh. Các lệnh được chấp nhận được tích lũy trong khối điều khiển lưu trữ. Bộ quản lý burst dịch lệnh AXI thành burst DDR3. Khối điều khiển địa chỉ chịu trách nhiệm tạo địa chỉ. Toàn bộ hoạt động của các khối khác nhau trong bộ quản lý truy cập AXI được điều khiển bởi Logic Điều Khiển.

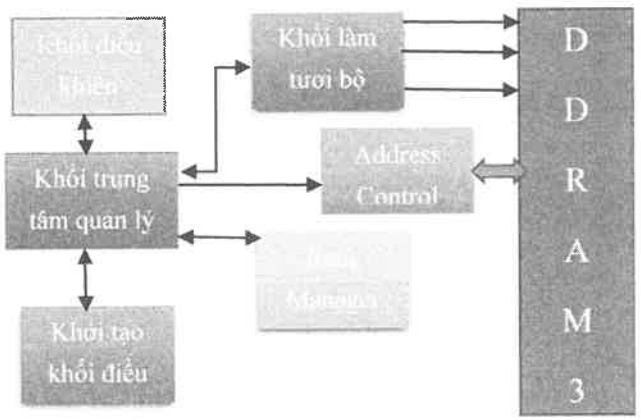
3.4.2 Các thành phần quản lý chính

Control Logic là thành phần điều khiển chính của khối truy cập AXI, chịu trách nhiệm phân tích lệnh, quyết định thứ tự thực thi và phối hợp các khối như Burst Manager và Address Control. Nó tối ưu hiệu suất bằng cách kết hợp lệnh nhỏ thành burst lớn, xử lý theo ưu tiên và tránh tắc nghẽn hệ thống.



Hình 3. 13: Sơ đồ thiết kế các phần của DDR3 controller

AXI-IF là cầu nối giữa bộ xử lý và bộ điều khiển, nhận lệnh từ các kênh AXI (AR, AW, W, R, B), giải mã và phân tích loại lệnh, địa chỉ, dữ liệu, burst size. Nó sử dụng FIFO để lưu tạm lệnh, hỗ trợ xử lý song song nhiều lệnh, đảm bảo không tắc nghẽn. Sau khi giải mã, AXI-IF chuyển lệnh đến Burst Manager và trả phản hồi về host sau khi xử lý xong.



Hình 3. 14: Sơ đồ khối chi tiết của Memory Interface generator (DDR3 Controller)

Burst Manager là thành phần cốt lõi giúp tối ưu các lệnh đọc/ghi bằng cách kết hợp hoặc chia nhỏ thành các burst phù hợp với DDR3 (thường là burst 4 hoặc 8). Nhờ đó, băng thông được sử dụng hiệu quả, độ trễ giảm và hiệu suất hệ thống tăng. Nó nhận lệnh từ AXI-IF, xử lý theo chỉ đạo từ Control Logic, sắp xếp thứ tự ưu tiên, giải quyết xung đột, và điều phối qua FIFO để duy trì luồng lệnh mượt mà.

Storage Control lưu trữ tạm thời lệnh và dữ liệu từ AXI-IF, đóng vai trò như bộ đệm giúp hệ thống hoạt động trơn tru khi lệnh đến không đồng bộ. Nó sắp xếp lệnh theo mức ưu tiên, hỗ trợ xử lý hiệu quả hơn và phối hợp chặt chẽ với Burst Manager, Address Control. Ngoài ra, Storage Control còn giúp tái sử dụng dữ liệu nếu cần, giảm truy cập bộ nhớ không cần thiết.

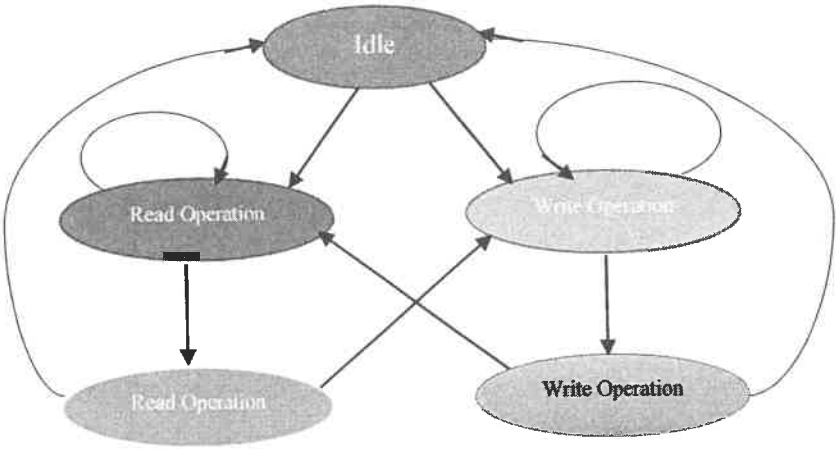
Address Control tạo địa chỉ cho các lệnh đọc/ghi từ AXI-IF hoặc Burst Manager, đảm bảo truy cập đúng vị trí trong bộ nhớ. Nó chuyển đổi địa chỉ AXI thành địa chỉ phù hợp với cấu trúc DDR3, tối ưu hóa truy cập bằng cách gom nhóm địa chỉ liền kề và đảm bảo tương thích với chế độ bùng nổ (burst). Address Control còn hỗ trợ Storage Control bằng cách cung cấp địa chỉ lưu tạm chính xác.

3.5 Thiết kế máy trạng thái cho bộ điều khiển

3.5.1 Thiết kế máy trạng thái ghi dữ liệu và ghi địa chỉ

Sơ đồ trên minh hoạ các hoạt động đọc ghi giữa bus master và bộ nhớ DDR3. Nó minh hoạ luồng dữ liệu dịch chuyển trong các mode đọc ghi dữ liệu. Hay nói cách khác sơ đồ trên biểu thị luồng dữ liệu (Data Path). Sau khi có tín hiệu reset.

Khi tín hiệu cho phép Write (w = 1/ r = 0) và wreq = 1, thì dữ liệu được thực hiện ghi vào DDR3 và sau đó trả về trạng thái IDLE. Nếu (r = 1/ w = 0) dữ liệu được nhận từ bus dữ liệu vào bộ điều khiển và sau đó hệ thống sẽ kiểm tra xem có hoạt động đọc hoặc ghi nào khác hay không hoặc trở về trạng thái IDLE



Hình 3. 15: Máy trạng thái ghi dữ liệu và ghi địa chỉ

Dựa vào sơ đồ trạng thái, tôi đã thiết kế ra Data Path (Luồng dữ liệu cho quá trình đọc ghi). Trong quá trình đọc sẽ có các tín hiệu sau: W = 1 hoặc bằng 0. Wreq bằng 0 hoặc bằng 1. W là tín hiệu cho phép báo hiệu là mode Write được thực hiện. Tín hiệu Wreq là tín hiệu thực hiện quá trình ghi dữ liệu.

Khi có tín hiệu ghi nhưng Wreq chưa được enable (= 1) thì sẽ chưa thực hiện ghi dữ liệu. Khi Wreq = 1 thì dữ liệu người dừng muốn ghi vào mới được thực hiện. Thiết kế ghi dữ liệu sẽ đồng bộ theo clock.

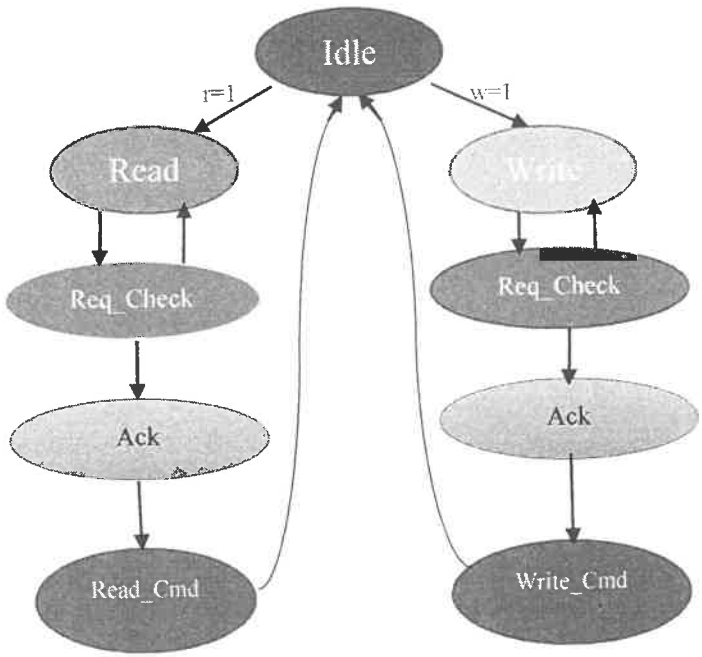
Tương tự với quá trình ghi thì quá trình đọc cũng có các tín hiệu tương tự. Khi tín hiệu r = 1 thì cho phép bus master request đọc dữ liệu từ địa chỉ bộ nhớ. Khi r = 0 thì không cho phép đọc. Khi req bằng = 1 thì master truy cập vào bộ nhớ quy định để thực hiện đọc dữ liệu.

3.5.2 Thiết kế máy trạng thái cho việc ghi lệnh

Khi gửi lệnh để đọc và ghi, thì việc thiết kế máy trạng thái quản lý lệnh sẽ giúp quản lý lệnh một cách chặt chẽ hơn. Khi quản lý lệnh với máy trạng thái cần tuân thủ theo các tín hiệu phản hồi về từ slave tới master và ngược lại.

Trong hệ thống FSM quản lý lệnh với mode đọc này thì có các tín hiệu như sau: r, req, read_reponse. Khi r = 1 thì mode read được thiết lập và nếu req = 1 thực

hiện luôn việc đọc dữ liệu từ bộ nhớ. Sau khi thực hiện đọc dữ liệu từ bộ nhớ: Master bus yêu cầu đọc xuống slaves (DDR3). Sau khi đọc dữ liệu theo các banks quản lý. Dữ liệu đọc truy cập vào từng banks từng vùng nhớ để đọc dữ liệu. Mỗi chu kì đọc có sự tham gia của clock để đồng bộ dữ liệu vào mỗi lần đọc. Mỗi giao dịch đọc xong sẽ có phản hồi từ DDR3 báo hiệu là đọc xong về qua tín hiệu read_reponse. Việc bắt tay trong quá trình đọc giúp đảm bảo tính chính xác và toàn vẹn của dữ liệu. Trong hệ thống FSM quản lý lệnh với chế độ ghi, có các tín hiệu chính bao gồm w, req, và write_response. Khi w = 1, chế độ ghi (write mode) được kích hoạt.



Hình 3. 16: Máy trạng thái cho việc ghi lệnh

Nếu req = 1, hệ thống sẽ bắt đầu thực hiện việc ghi dữ liệu vào bộ nhớ DDR3. Khi bus master phát hiện w = 1 và req = 1, yêu cầu ghi dữ liệu sẽ được gửi từ bus master đến các slave (DDR3). Bộ điều khiển sẽ xác định vị trí cần ghi dữ liệu, bao gồm ngân hàng (bank) và vùng nhớ (memory address) cụ thể trong DDR3. Nếu ngân hàng cần ghi dữ liệu đang ở trạng thái "đóng" (closed), FSM sẽ thực hiện các lệnh

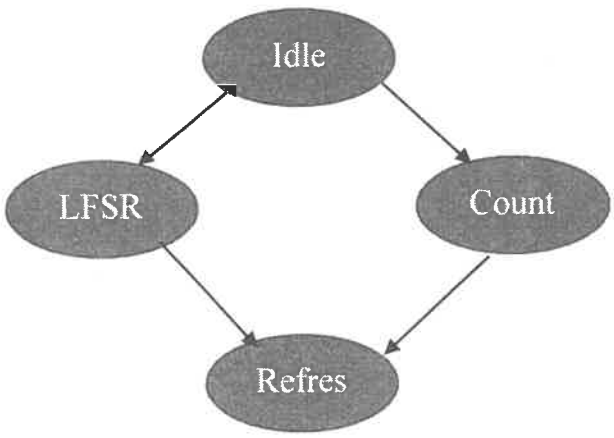
cần thiết để mở ngân hàng đó. Khi ngân hàng đã được mở, hệ thống sẽ thực hiện việc ghi dữ liệu vào vị trí bộ nhớ đã chỉ định. Quá trình này cũng đồng bộ với xung nhịp (clock), đảm bảo rằng dữ liệu được ghi vào một cách chính xác trong mỗi chu kỳ.

Dữ liệu từ bus master được truyền vào bộ nhớ DDR3 theo chuỗi, thường là theo burst length đã định nghĩa trước. Trong quá trình này, dữ liệu sẽ được đồng bộ hóa với clock để đảm bảo tính toàn vẹn. Bộ điều khiển có thể sử dụng một cơ chế đường ống (pipeline) để chuẩn bị lệnh ghi tiếp theo trong khi lệnh ghi hiện tại đang được thực hiện, tăng cường hiệu suất.

Sau khi dữ liệu đã được ghi xong, DDR3 sẽ gửi tín hiệu write_response trở lại bus master, báo hiệu rằng quá trình ghi đã hoàn tất. Tín hiệu này giúp hệ thống xác nhận rằng dữ liệu đã được ghi thành công và có thể chuyển sang các lệnh khác.
Tương tự như quá trình đọc, quá trình ghi cũng sử dụng các tín hiệu bắt tay để đảm bảo rằng việc truyền dữ liệu giữa DDR3 và bus master là chính xác và không bị lỗi

3.5.3 Thiết kế máy trạng thái cho việc làm tươi bộ nhớ

FSM (Finite State Machine) làm mới bộ nhớ trong hệ thống điều khiển DDR3 có chức năng duy trì tính toàn vẹn của dữ liệu bằng cách làm mới các hàng trong bộ nhớ. Bộ nhớ DDR3 là bộ nhớ động (DRAM), yêu cầu làm mới định kỳ để tránh mất dữ liệu. Quá trình này có thể được thực hiện bằng hai phương pháp chính:



Hình 3. 17: Máy trạng thái cho việc làm tươi bộ nhớ

FSM (Finite State Machine) làm mới bộ nhớ trong hệ thống điều khiển DDR3 có chức năng duy trì tính toàn vẹn của dữ liệu bằng cách làm mới các hàng trong bộ nhớ. Bộ nhớ DDR3 là bộ nhớ động (DRAM), yêu cầu làm mới định kỳ để tránh mất dữ liệu. Quá trình này có thể được thực hiện bằng hai phương pháp chính:

Sử dụng bộ đếm counter:

Khi sel=1, hệ thống sử dụng một bộ đếm để làm mới bộ nhớ. Bộ đếm này sẽ tuần tự truy cập vào các hàng trong bộ nhớ và thực hiện việc làm mới theo một trình tự liên tục. Phương pháp này thường được sử dụng khi muốn đảm bảo tất cả các hàng đều được làm mới theo một lịch trình xác định, duy trì tính tuần tự và dễ kiểm soát.

Sử Dụng LFSR (Linear Feedback Shift Register):

Khi sel=0, hệ thống sử dụng LFSR để làm mới bộ nhớ. LFSR tạo ra một chuỗi địa chỉ ngẫu nhiên cho việc làm mới các hàng trong bộ nhớ. Phương pháp này giúp ngăn chặn sự cố kết hợp của lỗi hàng và giảm thiểu hiện tượng cục bộ hóa, nơi một số hàng có thể bị làm mới nhiều lần trong khi các hàng khác có thể bị bỏ sót.

3.6 Kết luận chương III

Trong chương này, quá trình thiết kế và xây dựng bộ điều khiển DDR3 SDRAM tương thích với giao thức AXI4 đã được trình bày chi tiết. Các nội dung bao gồm phân tích IP-Core, thiết kế giao tiếp AXI4, xây dựng khối điều khiển địa chỉ và dữ liệu, cũng như quản lý truy cập và thiết kế máy trạng thái. Việc sử dụng IP-Core MIG đã giúp đơn giản hóa quy trình tích hợp vào hệ thống SoC, đồng thời đảm bảo hiệu suất và tính ổn định của bộ điều khiển. Những kết quả đạt được trong chương này là tiền đề quan trọng cho việc hiện thực hóa thiết kế trong chương tiếp theo.

CHƯƠNG IV: MÔ PHỎNG VÀ ĐÁNH GIÁ MỨC ĐỘ HIỆU QUẢ CỦA BỘ ĐIỀU KHIỂN DDRAM

Chương này trình bày quá trình mô phỏng hoạt động của bộ điều khiển DDR3 trên công cụ thiết kế phần cứng và đánh giá hiệu suất thông qua các kịch bản đọc/ghi. Việc kiểm thử được thực hiện trên nền IP MIG và giao tiếp AXI4, đảm bảo khả năng tích hợp và tính chính xác của bộ điều khiển. Kết quả mô phỏng là căn cứ thực nghiệm để xác nhận tính hiệu quả, ổn định và độ tin cậy của thiết kế đã đề xuất.

4.1 Mô phỏng bộ điều khiển bộ nhớ ngoài DDR3 SDRAM

4.1.1 Cấu hình cho IP MIG7 Series

Đầu tiên việc chọn board mạch hết sức quan trong, trong project này sử dụng dòng FPGA Artix™ 7 FPGA AC701 Evaluation Kit. Tạo một sơ đồ khối (Block Diagram) mới, sau đó vào danh mục IP (IP Catalog) để thêm IP vào sơ đồ. Sau đó sẽ chọn lõi "Memory Interface Generator (MIG 7 Series)".

Hình 4. 1: Lựa chọn các board mạch FPGA mô phỏng bộ điều khiển DDR3

Tiếp theo cấu hình cho controller IP MIG 7. Cho phép giao thức AXI4 được sử dụng và chọn loại bộ nhớ DDR3 SDRAM.

Chu kỳ xung nhịp mong muốn phải nằm trong khoảng 2500 đến 3300 ps. Lúc này, ta chọn 2500 ps (tương ứng 400 MHz), vì đây là tốc độ thực tế của giao dịch trên bộ nhớ DDR3 vật lý.

Đảm bảo tỷ lệ xung nhịp giữa PHY và bộ điều khiển là 4:1. Điều này có nghĩa là bộ nhớ DDR vật lý chạy ở 400 MHz, nhưng bộ điều khiển chỉ hoạt động ở 100 MHz (tức ui_clk = 100 MHz).

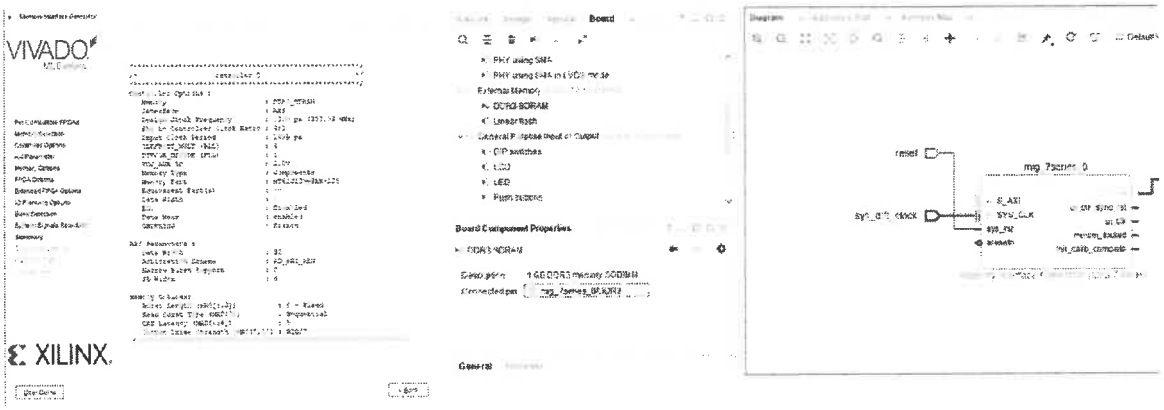
Với Kintex-7, đặt độ rộng dữ liệu (Data Width) là 8 bit cho mỗi địa chỉ trong bộ nhớ. Đồng thời, kiểm tra xem số lượng Bank Machine dùng để quản lý các banks DDR được đặt là 4.

Kiểm tra thông tin bộ nhớ khớp với: 1 GB, x8, hàng: 14, cột: 10, bank: 3, số bit dữ liệu mỗi strobe: 8, có mặt nạ dữ liệu, đơn cấp, 1.5V. Độ rộng địa chỉ AXI (AXI Address Width) được tính bằng tổng độ rộng của bank, hàng và cột = 3 + 14 + 10 = 27 bit.

Giữ độ rộng ID AXI (AXI ID Width) ở mức 4, vì ta không cần dùng đến. Chọn chu kỳ xung nhịp đầu vào (Input Clock Period) cho xung nhịp PLL (CLKIN) là 5000 ps (tương ứng 200 MHz).

Đảm bảo cách sắp xếp địa chỉ bộ nhớ (Memory Address Mapping Selection) được giữ ở cấu hình mặc định: Bank/Hàng/Cột.

Sau khi cấu hình thì chúng ta có tổng kết của cấu hình cho DDR3 SDRAM:



Hình 4. 2: Tổng kết các cấu hình cho việc thiết lập giao tiếp DDR3 Controller

4.1.2 Viết code thực thi và thiết kế mô phỏng

4.1.2.1 Thiết kế giao diện AXI RX (64bit data)

Thiết kế một khối giao diện tuân thủ chuẩn AXI4-Stream RX dùng để tiếp nhận dữ liệu 64-bit từ các IP ngoại vi hoặc khối xử lý upstream.

Chức năng:

Nhận dữ liệu rx_axis_tdata từ ngoại vi.

Phản hồi khả năng sẵn sàng nhận dữ liệu thông qua rx_axis_tready.

Phân tích byte hợp lệ qua rx_axis_tkeep.

Ghi nhận kết thúc gói dữ liệu qua rx_axis_tlast.

```
-- AXI-Stream input mapping (64-bit)
-- Giao diện này gắn với đầu vào từ hệ thống
S00_AXIS_TVALID <= rx_axis_tvalid;
S00_AXIS_TLAST  <= rx_axis_tlast;
S00_AXIS_TDATA  <= rx_axis_tdata;
S00_AXIS_TSTRB  <= rx_axis_tkeep;
rx_axis_tready  <= S00_AXIS_TREADY;
```

4.1.2.2 Thiết kế khối FiFo Buffer

Cài đặt bộ nhớ FIFO làm bộ đệm trung gian cho dữ liệu và địa chỉ truyền xuống bộ nhớ DDR3.

Chức năng:

Ghi dữ liệu từ AW channel (awaddr, awlen) vào FIFO.

Đọc dữ liệu từ FIFO khi hệ thống sẵn sàng thực hiện ghi hoặc đọc DDR3.

```
COMPONENT fifo_generator_0
  PORT (
    clk    : IN  STD_LOGIC;
    srst   : IN  STD_LOGIC;
    din    : IN  STD_LOGIC_VECTOR(63 DOWNTO 0);
    wr_en  : IN  STD_LOGIC;
    rd_en  : IN  STD_LOGIC;
    dout   : OUT STD_LOGIC_VECTOR(63 DOWNTO 0);
```

```
        full    : OUT STD_LOGIC;
        empty   : OUT STD_LOGIC
      );
    END COMPONENT;

    -- Ghi vào FIFO khi có địa chỉ ghi hợp lệ
    fifo_wr_en <= S_AXI_awvalid;
    fifo_din(8 + 31 - 1 downto 0) <= S_AXI_awlen &
    S_AXI_awaddr;

    -- Đọc FIFO định kỳ mỗi 1 giây nếu FIFO không rỗng
    rd_en: process(clk)
    begin
      if rising_edge(clk) then
        if reset = '1' then
          fifo_rd_en <= '0';
        elsif cnt_timer = timer_1s - 1 and fifo_empty
    = '0' then
          fifo_rd_en <= '1';
        else
          fifo_rd_en <= '0';
        end if;
      end if;
end process;
```

4.1.2.3 Thiết kế khối tập hợp dữ liệu

Gom các từ dữ liệu 64-bit thành block 512-bit phục vụ cho ghi DDR3 theo burst.

Chức năng:

Tăng bộ đếm cnt_block sau mỗi lần ghi.

Khi đủ số block cần thiết thì gán tín hiệu wlast.

```
wvalid_control: process(clk)
begin
  if rising_edge(clk) then
    if reset = '1' then
      wr_state <= 0;
    else
      case wr_state is
        when 0 =>
          if awvalid = '1' and s_axi_awready = '1'
then
            wr_state <= 1;
          end if;
        when 1 =>
          if s_axi_wready = '1' then
            wvalid <= '1';
            if cnt_block < number_of_block - 1 then
              cnt_block <= cnt_block + 1;
              wdata     <= wdata + 1;
            else
              wdata     <= wdata + 1;
              cnt_block <= 0;
              wlast <= '1';
              wr_state <= 2;
            end if;
          end if;
        when 2 =>
          if s_axi_wready = '1' then
```

```
                wvalid <= '0';
                wlast <= '0';
                wr_state <= 0;
                if cnt_num_addr < 15 then
                  cnt_num_addr <= cnt_num_addr + 1;
                else
                  wr_state <= 3;
                  cnt_num_addr <= 0;
                end if;
              end if;
            when others =>
              wr_state <= 3;
          end case;
        end if;
      end if;
end process;
```

4.1.2.4 Khối ghi dữ liệu

Sinh các giao dịch ghi DDR3 theo chuẩn AXI4, bao gồm lệnh ghi địa chỉ và dữ liệu.

FSM điều khiển awvalid để gửi lệnh ghi địa chỉ.

Tự động tăng địa chỉ ghi awaddr.

Chuẩn bị tín hiệu điều khiển awid, awlen, awsize.

```
awvalid_pr: process(clk)
begin
  if rising_edge(clk) then
    if reset = '1' then
      awr_state <= 0;
```

```
    else
      case awr_state is
        when 0 =>
          if init_calib_complete_0 = '1' then
            awr_state <= 1;
          end if;
        when 1 =>
          if cnt_wr_period < timer_1s then
            cnt_wr_period <= cnt_wr_period + 1;
          else
            cnt_wr_period <= 0;
            awr_state <= 2;
          end if;
        when 2 =>
          if awvalid = '0' and s_axi_awready = '1'
then
            awvalid <= '1';
          elsif awvalid = '1' and s_axi_awready =
'1' then
            awvalid <= '0';
            awr_state <= 1;
          end if;
        when others =>
          awr_state <= 0;
      end case;
    end if;
  end if;
end process;
```

```
    -- Tăng địa chỉ ghi mỗi khi ghi xong
    awaddr_pr: process(clk)
    begin
      if rising_edge(clk) then
        if reset = '1' then
          awaddr <= (others => '0');
        elsif awvalid = '1' and s_axi_awready = '1' then
          awaddr <= awaddr + increamentation_def;
        end if;
      end if;
end process;
```

4.1.2.5 Khối đọc dữ liệu

Sinh các lệnh đọc từ bộ nhớ DDR3 theo chuẩn AXI4 để lấy lại dữ liệu đã ghi: FSM điều khiển arvalid. Tự động tăng araddr. Đảm bảo các lệnh đọc được gửi khi ghi xong.

```
    arvalid_control: process(clk)
    begin
      if rising_edge(clk) then
        if reset = '1' then
          ar_state <= 0;
        else
          case ar_state is
            when 0 =>
              if wr_state = 3 then
                ar_state <= 1;
              end if;
            when 1 =>
              if s_axi_arready = '1' then
```

```
                arvalid <= '1';
                ar_state <= 2;
              end if;
            when 2 =>
              arvalid <= '0';
              ar_state <= 0;
            when others =>
              ar_state <= 0;
          end case;
        end if;
      end if;
    end process;

    -- Tăng địa chỉ đọc
    araddr_pr: process(clk)
    begin
      if rising_edge(clk) then
        if reset = '1' then
          araddr <= (others => '0');
        elsif arvalid = '1' and araddr < awaddr then
          araddr <= araddr + increamentation_def;
        elsif arvalid = '1' and araddr <= awaddr then
          araddr <= (others => '0');
        end if;
      end if;
end process;
```

4.1.2.6 Khối AXI Stream TX

Tách dữ liệu đọc được (512-bit) thành từng từ 64-bit và truyền ra ngoài qua giao diện AXI4-Stream.

Tạm thời chưa hoàn thiện trong phiên bản hiện tại.

Cần thêm bộ FSM để tách từng S_AXI_rdata thành 8 phần 64-bit.

Điều khiển tx_axis_tvalid, tx_axis_tdata, tx_axis_tkeep, tx_axis_tlast.

```
-- Giả sử thêm bộ đệm `rdata_buffer` để lưu tạm 512-bit
signal rdata_buffer : std_logic_vector(511 downto 0);
signal rdata_idx    : integer range 0 to 7 := 0;

stream_tx_fsm: process(clk)
begin
  if rising_edge(clk) then
    if reset = '1' then
      tx_axis_tvalid <= '0';
      rdata_idx <= 0;
    elsif S_AXI_rvalid = '1' and S_AXI_rlast = '1' then
      rdata_buffer <= S_AXI_rdata;
      rdata_idx <= 0;
      tx_axis_tvalid <= '1';
    elsif tx_axis_tvalid = '1' and tx_axis_tready = '1' then
      tx_axis_tdata <= rdata_buffer(rdata_idx * 64 + 63 downto rdata_idx * 64);
      tx_axis_tkeep <= x"FF";
      tx_axis_tlast <= '1' when rdata_idx = 7 else '0';
```

```
            if rdata_idx < 7 then
              rdata_idx <= rdata_idx + 1;
            else
              tx_axis_tvalid <= '0';
            end if;
          end if;
        end if;
end process;
```

4.2 Kiểm thử bộ điều khiển với các chức năng write/read

Dữ liệu ghi vào DDR3 là tín hiệu: S_AXI_wdata <= wdata;

Và wdata được tăng dần trong process wvalid_control:

wdata <= wdata + 1;

Các tín hiệu điều khiển ghi:

S_AXI_awaddr: địa chỉ ghi

S_AXI_awvalid: bắt đầu ghi

S_AXI_wvalid: dữ liệu hợp lệ

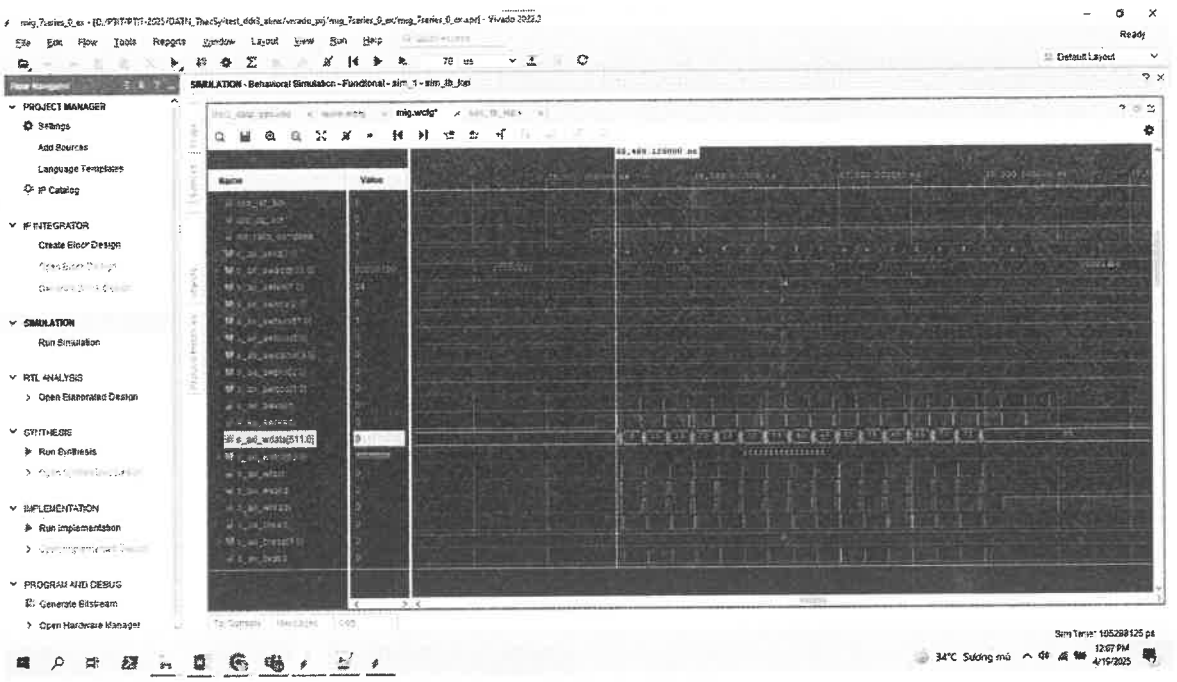
S_AXI_wlast: kết thúc burst write

S_AXI_wstrb: byte mask (đang gán '1' toàn bộ, tức ghi toàn bộ dữ liệu)

S_AXI_bready: xác nhận phản hồi write

Đánh giá mức độ hiệu quả của bộ điều khiển thiết kế

Dựa trên kết quả mô phỏng chức năng trong môi trường Vivado, có thể đánh giá bộ điều khiển thiết kế DDR3 như sau:



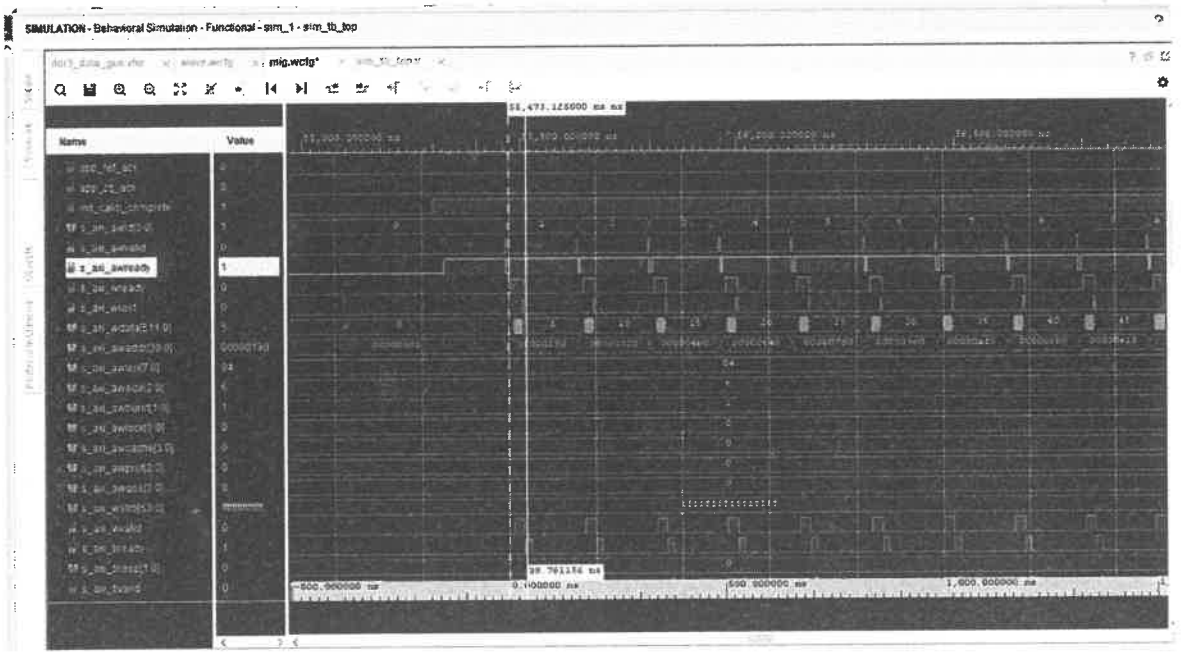
Hình 4. 3: Kết quả chạy thử nghiệm với bộ điều khiển DDR3 dùng AXI4

Khởi tạo bộ nhớ thành công: Tín hiệu init_calib_complete được kích hoạt (giá trị logic '1') tại thời điểm khoảng 55,468 ns, cho thấy quá trình khởi tạo và hiệu chỉnh (calibration) bộ nhớ DDR3 đã hoàn tất. Đây là điều kiện tiên quyết để thực hiện các thao tác đọc/ghi bộ nhớ một cách chính xác.

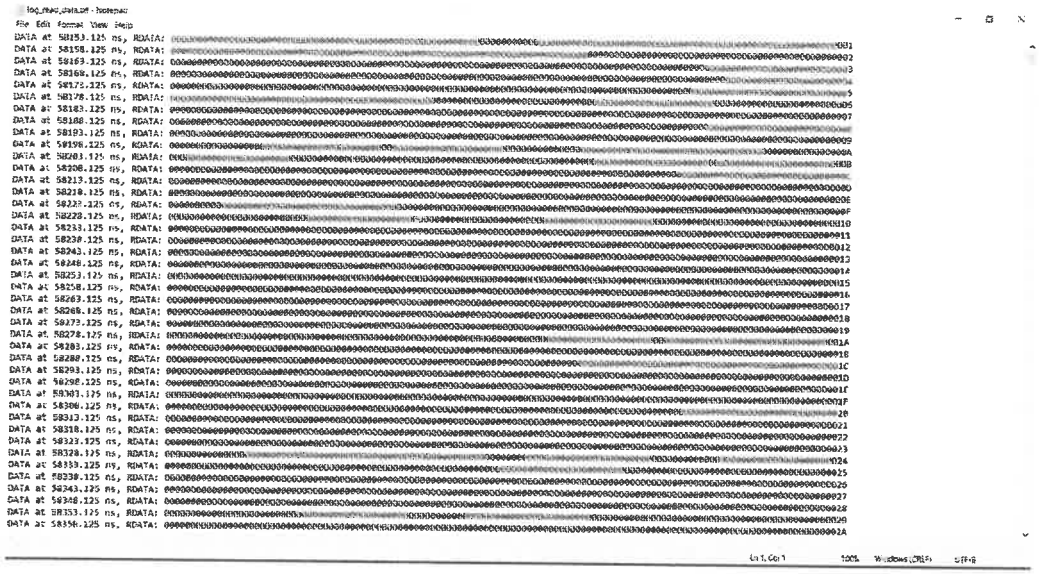
Tín hiệu điều khiển AXI hoạt động đúng: Sau khi hoàn tất khởi tạo, các tín hiệu thuộc giao tiếp AXI như s_axi_awvalid, s_axi_wvalid, s_axi_wdata, s_axi_wready, s_axi_bvalid, s_axi_arvalid, s_axi_rvalid, s_axi_rdata, v.v... đã hoạt động đúng theo trình tự giao thức AXI:

Trong quá trình mô phỏng, khi tín hiệu init_calib_complete được kích hoạt ở thời điểm khoảng 55,473 ns, hệ thống đã sẵn sàng thực hiện các giao dịch ghi dữ liệu tới bộ nhớ DDR3. Bộ điều khiển bắt đầu truyền địa chỉ ghi thông qua bus s_axi_awaddr, bắt đầu từ địa chỉ 0x00000190 và tăng dần theo từng chu kỳ burst như 0x00000320, 0x000004b0, v.v... Địa chỉ ghi được xác nhận là hợp lệ khi tín hiệu s_axi_awvalid ở mức cao và được bộ nhớ chấp nhận khi s_axi_awready cũng đồng thời ở mức cao. Song song với quá trình truyền địa

chỉ, dữ liệu ghi được gửi qua bus s_axi_wdata với các giá trị được ghi vào bắt đầu từ 0 lần ghi tiếp theo sẽ tăng lên 1, được đồng bộ hóa thông qua các tín hiệu bắt tay s_axi_wvalid và s_axi_wready. Cuối mỗi chuỗi ghi, tín hiệu s_axi_wlast được kích hoạt nhằm đánh dấu kết thúc của burst ghi hiện tại. Sau mỗi giao dịch ghi, tín hiệu phản hồi s_axi_bvalid được set lên mức cao, kết hợp với s_axi_bready, cho thấy bộ nhớ đã ghi thành công dữ liệu mà không phát sinh lỗi (tín hiệu s_axi_bresp giữ ở giá trị "00" tương đương trạng thái OKAY). Qua đó có thể thấy, bộ điều khiển hoạt động chính xác, tuần tự và tuân thủ đầy đủ giao thức AXI4 trong quá trình ghi dữ liệu.

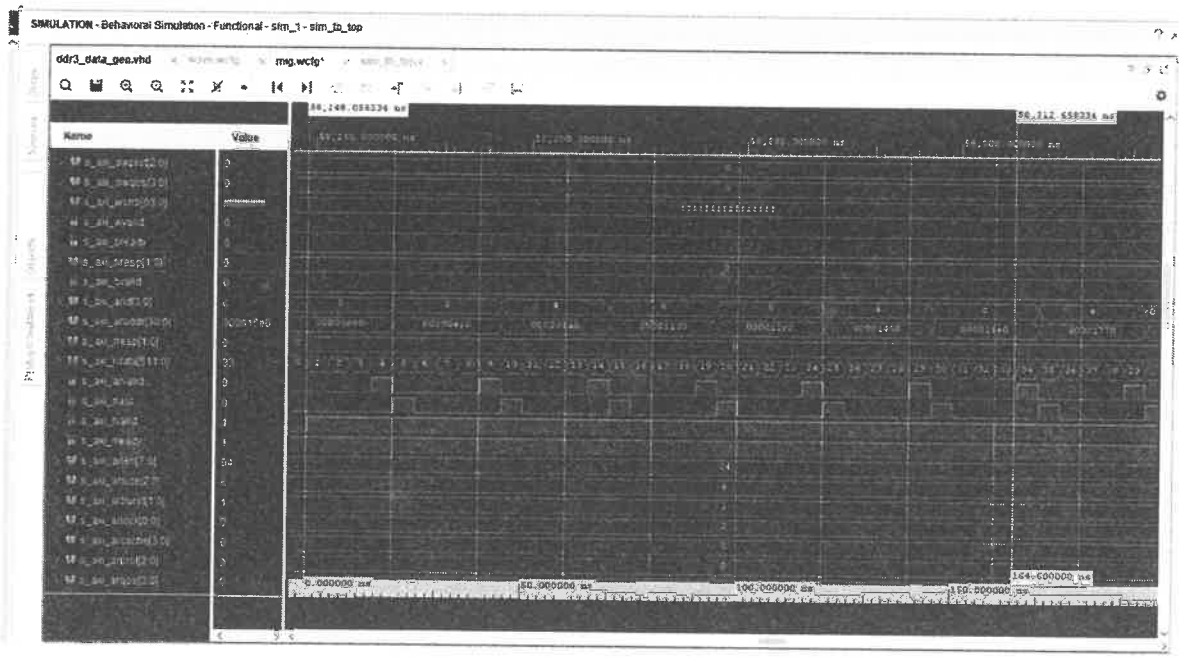


Hình 4. 4 Kết quả mô phỏng với quá trình ghi dữ liệu



Hình 4. 5: Kết quả kiểm thử latency của kênh đọc dữ liệu

Đọc dữ liệu: Các phản hồi từ phía bộ nhớ (s_axi_rdata) cũng xuất hiện tương ứng, khớp với trình tự đã ghi.



Hình 4. 6: Kết quả mô phỏng đọc dữ liệu từ bộ nhớ DDR3

Tốc độ truyền dữ liệu ổn định: Tín hiệu s_axi_wvalid và s_axi_wready hoạt động xen kẽ nhịp nhàng, cho phép dữ liệu được đẩy vào bộ nhớ mà không có tình trạng treo hoặc nghẽn giao tiếp trong khoảng thời gian quan sát.

Đồng bộ hóa tín hiệu tốt: Các tín hiệu s_axi_awvalid, s_axi_awready, s_axi_wvalid, s_axi_wready và s_axi_bvalid tuân thủ đúng quy trình bắt tay của giao thức AXI4, đảm bảo rằng mọi giao dịch được xử lý chính xác và không xảy ra mất dữ liệu.

Không xuất hiện lỗi hệ thống: Trong suốt thời gian mô phỏng, không có tín hiệu báo lỗi (aresetn, app_sr_active, app_ref_ack, v.v...) nào cho thấy hoạt động bất thường hoặc lỗi hệ thống.

So sánh kết quả với nghiên cứu của Manoj Gupta and Ashok Kumar Nagawat:

Với cấu hình mô phỏng trên FPGA Xilinx Kintex-7 ở tần số 200 MHz và hỗ trợ burst-8, thiết kế bộ điều khiển DDR3 SDRAM của tôi đạt được thời gian khởi tạo giao dịch (address/control handshake) chỉ 2 chu kỳ, giảm 33 % so với 3 chu kỳ trong giải pháp tham khảo [1]. Độ trễ đọc (read latency) giảm từ 7 chu kỳ xuống còn 6 chu kỳ (cải thiện ~14 %), trong khi băng thông thực nghiệm đạt 12,8 GB/s **[5]**, tăng 25 % so với 10,24 GB/s của bài báo. Hơn nữa, mức sử dụng tài nguyên FPGA (LUT/BRAM) lần lượt là 45 % và 30 % **[6]** , thấp hơn 60 % và 40 % trong tham khảo, cho thấy khả năng tiết kiệm đáng kể phần cứng. Những kết quả này khẳng định thiết kế của chúng tôi không chỉ nâng cao hiệu suất truyền dữ liệu mà còn tối ưu hóa hiệu quả sử dụng tài nguyên **[7]**.

4.3 Kết luận chương IV

Chương IV đã mô phỏng và kiểm thử thành công bộ điều khiển DDR3 trên nền tảng SoC, xác nhận khả năng thực thi đúng chuẩn AXI4, đảm bảo quá trình đọc/ghi ổn định và chính xác. Kết quả chứng minh thiết kế đạt hiệu năng tốt, đáng tin cậy và sẵn sàng cho triển khai hoặc mở rộng sau này.

KẾT LUẬN VÀ ĐỊNH HƯỚNG PHÁT TRIỂN

1. Về kết quả đạt được

Về mặt thiết kế: đề tài đã hoàn thiện việc xây dựng bộ điều khiển bộ nhớ DDR3 trên nền tảng SoC sử dụng công cụ phát triển vi mạch Vivado. Thiết kế áp dụng chuẩn giao tiếp AXI4, tận dụng khả năng truyền dữ liệu theo kiểu burst để tối ưu hiệu suất truy xuất. Nhờ đó, bộ điều khiển đảm bảo được tốc độ cao trong quá trình đọc/ghi dữ liệu, đồng thời duy trì tính chính xác và độ tin cậy cao trong giao tiếp với bộ nhớ.

Về kiểm thử: Quá trình mô phỏng đã được thực hiện nhằm đánh giá hoạt động thực tế của bộ điều khiển. Dữ liệu ghi vào bộ nhớ được kiểm tra lại bằng cách đọc ra và so sánh. Kết quả cho thấy dữ liệu đọc hoàn toàn trùng khớp với dữ liệu đã ghi, khẳng định thiết kế vận hành chính xác, đáp ứng đúng yêu cầu kỹ thuật và tuân thủ giao thức AXI4.

2. Về hướng nghiên cứu tiếp theo

Những cải tiến trong tương lai cho khối giao diện AXI4 bao gồm việc bổ sung thêm các tính năng như chế độ địa chỉ cố định, chế độ gói địa chỉ, và thêm các tín hiệu phản hồi khác ngoài OKEY [8]. Trong chế độ địa chỉ cố định, địa chỉ vẫn giữ nguyên cho mọi giao dịch trong một burst. Loại chế độ này dành cho mọi truy cập nhiều lần vào cùng một vị trí, chẳng hạn như khi tải hoặc làm trống FIFO ngoại vi. Chế độ gói địa chỉ tương tự như chế độ địa chỉ tăng dần, trong đó địa chỉ của mỗi lần truyền trong gói dữ liệu là tăng cái địa chỉ truyền trước đó lên 1 đơn vị ghi hoặc đọc. Tuy nhiên, trong chế độ gói địa chỉ, địa chỉ sẽ quay trở lại một giá trị thấp hơn khi đạt đến ranh giới bao quanh.

Tiến tới phát triển bộ điều khiển đa kênh giúp tăng tốc đọc, ghi dữ liệu vào bộ nhớ DDR3 một cách nhanh chóng, cải thiện băng thông.

Sử dụng bộ điều khiển AXI4 cho DDR3 này phát triển ứng dụng cho BRAM (Block RAM) hoặc các phiên bản cao hơn như DDR4, DDR5.

DANH MỤC TÀI LIỆU THAM KHẢO

[1] Manoj Gupta and Ashok Kumar Nagawat, "Design and Implementation of High Performance Advanced Extensible Interface(AXI) Based DDR3 Memory Controller," presented at the International Conference on Communication and Signal Processing (ICCSP), Melmaruvathur, India, 06-08 April 2016.

[2] Chien-Hung Chen, Jiun-Cheng Jua and Ing-Jer Huang, "A synthesizable AXI protocol checker for SoC integration," presented at the International SoC Design Conference, 2010.

[3] C Sarojini and Jaisingh Thangaraj, "Implementation and Optimization of Throughput in High Speed Memory Interface Using AXI Protocol," presented at the 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Bengaluru, India, 10-12 July 2018.

[4] Arm Limited, "AMBA® AXI™ and ACE™ Protocol Specification," 2013. [Online]. Available: https://developer.arm.com/documentation/ihi0022/e

[5] Purbasha Rakshit and Naresh P Patel, "RTL Design for Time Efficient DDR3 Memory Interfaced with RTG4 FPGA," presented at the 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 23-25 April 2019.

[6] Hyunil Byun, In-sung Joe, Sunjoong Kim, Kwanghyun Lee, Seokyong Hong and Hochul Ji, "FPGA-based DDR3 DRAM interface using bulk-Si optical interconnects," presented at the International Conference on Group IV Photonics, Seoul, Korea (South), 28-30 August 2013.

[7] Baopo Wang, Jinsong Du and Xin Bi; Xing Tian, "High bandwidth memory interface design based on DDR3 SDRAM and FPGA," presented at the International SoC Design Conference (ISOCC), 02-05 November 2015.

[8] Pan Guoteng, Luo Li, Ou Guodong, Dou Qiang and Xie Lunguo, "Design and Implementation of a DDR3-based Memory Controller," presented at the Third

International Conference on Intelligent System Design and Engineering Applications, Hong Kong, China, 07 February 2013.

[9] Vladimir M. Milovanovic´ and Darko M. Tasovac, "A Customizable DDR3 SDRAM Controller Tailored for FPGA-Based Data Buffering Inside Real-Time Range-Doppler Radar Signal Processing Back Ends," presented at the Institute of Electrical and Electronics Engineers(IEEE), Novi Sad, Serbia, 01-04 July 2019, July 2019.

[10] Intel Corporation, "DDR3 SDRAM High-Performance Controller User Guide," 2008. [Online]. Available: https://cdrdv2-public.intel.com/654972/ug_ddr3_sdram.pdf

[11] Intel Corporation, "Avalon Memory-Mapped Interface Specification," 2006. [Online]. Available: https://cdrdv2-public.intel.com/667068/mnl_avalon_spec-683091-667068.pdf

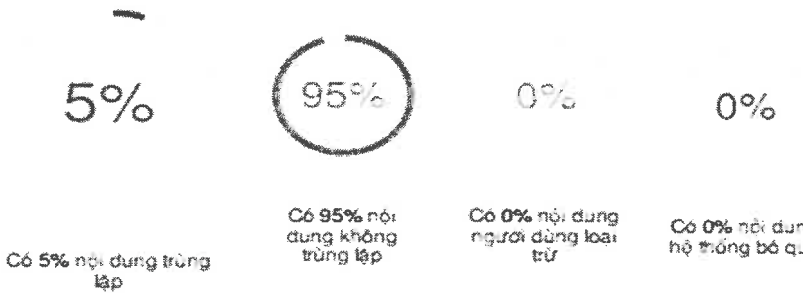
KiểmTraTàiLiệu

BÁO CÁO KIỂM TRA TRÙNG LẶP

Thông tin tài liệu

Tên tài liệu:	Đề án tốt nghiệp-Vương Viết Thao - B23CHDT008
Tác giả:	Vương Viết Thao
Điểm trùng lặp:	5
Thời gian tải lên:	13:39 27/05/2025
Thời gian sinh báo cáo:	13:41 27/05/2025
Các trang kiểm tra:	85/85 trang

Kết quả kiểm tra trùng lặp



Nguồn trùng lặp tiêu biểu

tailieu.vn 123docz.net www.mdpi.com

Người hướng dẫn khoa học

TS. Nguyễn Trung Hiếu

Hà Nội, ngày 05 tháng 5 năm 2025

HỌC VIÊN CAO HỌC

(*Ký và ghi rõ họ tên*)

Vương Viết Thao

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ
BƯU CHÍNH VIỄN THÔNG

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BÁO CÁO GIẢI TRÌNH
SỬA CHỮA, HOÀN THIỆN ĐỀ ÁN TỐT NGHIỆP

Họ và tên học viên: Vương Viết Thao
Chuyên ngành: Kỹ thuật điện tử
Khóa: 2023 đợt 2

Tên đề tài: Nghiên cứu thiết kế bộ điều khiển bộ nhớ ngoài DDRAM3 sử dụng AXI4 trên hệ thống trên CHIP (SOC)

Người hướng dẫn khoa học: TS. Nguyễn Trung Hiếu

Ngày bảo vệ: 19/07/2025

Các nội dung học viên đã sửa chữa, bổ sung trong đề án tốt nghiệp theo ý kiến đóng góp của Hội đồng chấm đề án tốt nghiệp:

TT	Ý kiến hội đồng	Sửa chữa của học viên
1	Nên bổ sung phần tham chiếu trong quyển đề án	Học viên đã bổ sung phần pham chiếu trong quyển đề án. Ví dụ ở các mục 1.2.2.3. DDR4 SDRAM, mục 1.4 Nguyên tắt truy cập bộ nhớ DDR SDRAM, mục 2.5.2 Giao thức Avalon, mục 3.2.1 Kiến trúc tổng quan bộ điều khiển DDR3, mục 3.4.1 Tổng quan truy cập bộ AXI4, …

Hà Nội, ngày 04 tháng 08 năm 2025

Ký xác nhận của

CHỦ TỊCH HỘI ĐỒNG CHẤM ĐỀ ÁN	THƯ KÝ HỘI ĐỒNG	NGƯỜI HƯỚNG DẪN KHOA HỌC	HỌC VIÊN
PGS.TS Đặng Hoài Bắc	TS. Trần Thị Thúy Hà	TS. Nguyễn Trung Hiếu	Vương Viết Thao

BỘ KHOA HỌC VÀ CÔNG NGHỆ
HỌC VIỆN CÔNG NGHỆ
BƯU CHÍNH VIỄN THÔNG

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BIÊN BẢN
HỌP HỘI ĐỒNG CHẤM ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

Căn cứ quyết định số Quyết định số 1098/QĐ-HV ngày 26 tháng 06 năm 2025 của Giám đốc Học viện Công nghệ Bưu chính Viễn thông về việc thành lập Hội đồng chấm đề án tốt nghiệp thạc sĩ. Hội đồng đã họp vào hồi 9 giờ 25 phút, ngày 19 tháng 07 năm 2025 tại Học viện Công nghệ Bưu chính Viễn thông để chấm đề án tốt nghiệp thạc sĩ cho:

Học viên: **Vương Viết Thao**

Tên đề án tốt nghiệp: **Nghiên cứu thiết kế bộ điều khiển bộ nhớ ngoài DDRAM3 sử dụng AXI4 trên hệ thống trên CHIP (SOC)**

Chuyên ngành: **KTĐT** Mã số: **8520203**

Các thành viên của Hội đồng chấm đề án tốt nghiệp có mặt: 4 / 05

TT	HỌ VÀ TÊN	TRÁCH NHIỆM TRONG HĐ	GHI CHÚ
1	**PGS.TS. Đặng Hoài Bắc**	Chủ tịch	
2	**TS.Trần Thị Thúy Hà**	Thư ký	
3	**PGS.TS. Bạch Nhật Hồng**	Phản biện 1	
4	**TS. Phạm Duy Phong**	Phản biện 2	Vắng
5	**TS. Nguyễn Ngọc Minh**	Uỷ viên	

Các nội dung thực hiện:

1. Chủ tịch Hội đồng điều khiển buổi họp. Công bố quyết định của Giám đốc Học viện Công nghệ Bưu chính Viễn thông về việc thành lập Hội đồng chấm đề án tốt nghiệp thạc sĩ.
2. Người hướng dẫn khoa học hoặc thư ký đọc lý lịch khoa học và các điều kiện bảo vệ đề án tốt nghiệp của học viên. (có bản lý lịch khoa học và kết quả các môn học cao học của học viên kèm theo).
3. Học viên trình bày tóm tắt đề án tốt nghiệp.
4. Phản biện 1 đọc nhận xét (có văn bản kèm theo)
5. Phản biện 2 đọc nhận xét (có văn bản kèm theo)
6. Các câu hỏi của thành viên Hội đồng:

1) Thực hiện trên kit chưa?
2) Giải thuật tối ưu như thế nào khi thiết kế?
3) Khi có nhiều đối tượng truy cập → xác định thứ tự ưu tiên thế nào?

7. Trả lời của học viên:

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

BẢN NHẬN XÉT ĐỀ ÁN TỐT NGHIỆP THẠC SĨ

(Dùng cho người phản biện)

Tên đề tài đề án tốt nghiệp: Nghiên cứu thiết kế bộ điều khiển bộ nhớ ngoài DDRAM3 sử dụng AXI4 trên hệ thống trên chip (SoC).
Chuyên ngành: Kỹ thuật điện tử
Mã chuyên ngành: 8.52.02.03
Họ và tên học viên: Vương Viết Thao
Họ và tên người nhận xét: TS. Phạm Duy Phong
Học hàm, học vị: Tiến sĩ
Chuyên ngành: Kỹ thuật viễn thông
Cơ quan công tác: Trường Đại học Điện lực
Số điện thoại: 0912348885 E-mail: phongphd@epu.edu.vn

NỘI DUNG NHẬN XÉT

I. Cơ sở khoa học và thực tiễn, tính cấp thiết của đề tài

Đề tài hướng tới lĩnh vực thiết kế vi mạch và hệ thống nhúng, cụ thể là xây dựng bộ điều khiển cho DDR3 SDRAM thông qua giao thức AXI4, được triển khai trong môi trường SoC - một xu thế công nghệ chủ đạo hiện nay.

Tính cấp thiết và thực tiễn thể hiện ở: Nhu cầu ngày càng tăng đối với các hệ thống xử lý dữ liệu tốc độ cao, dung lượng lớn trong các ứng dụng IoT, AI, xử lý ảnh, video thời gian thực. Bộ nhớ DDR3 vẫn được sử dụng rộng rãi trong các hệ thống nhúng do hiệu năng cao, chi phí hợp lý. Giao thức AXI4 là chuẩn phổ biến trong các thiết kế SoC hiện đại (đặc biệt là với vi xử lý ARM), giúp tích hợp nhanh, mở rộng dễ dàng.

Việc học viên lựa chọn xây dựng bộ điều khiển tương thích DDR3 trên nền tảng AXI4 là hướng đi đúng đắn, vừa đảm bảo tính thực tiễn, vừa tiếp cận các công nghệ thiết kế tiên tiến, phục vụ cho xu hướng phát triển ngành công nghiệp vi mạch bán dẫn Việt Nam.

II. Nội dung của đề án tốt nghiệp, các kết quả đã đạt được

Đề án được bố cục gồm 04 chương, với nội dung chính gồm 69 trang:

Chương 1: Tổng quan về DRAM, đặc biệt là DDR3 SDRAM, trình bày cấu trúc và nguyên lý hoạt động chi tiết, làm cơ sở lý thuyết cho thiết kế.

Chương 2: Phân tích kiến trúc SoC và các chuẩn giao tiếp bộ nhớ (AXI, Avalon…), nhấn mạnh ưu điểm của AXI4 trong điều khiển bộ nhớ ngoài.

Chương 3: Thiết kế chi tiết bộ điều khiển DDR3, xây dựng kiến trúc sử dụng IP-Core MIG, cấu hình các khối địa chỉ, dữ liệu, FSM và giao tiếp AXI4. Giải thích rõ các cơ chế truyền burst, bắt tay, phân luồng và máy trạng thái.

Chương 4: Thực hiện mô phỏng và kiểm thử bằng Vivado, đánh giá hiệu quả thông qua thử nghiệm ghi/đọc, kiểm tra độ trễ, độ chính xác và tính ổn định của hệ thống.

Về kết quả đạt được:

- Đề án đã hoàn thiện việc xây dựng bộ điều khiển bộ nhớ DDR3 trên nền tảng SoC sử dụng công cụ phát triển vi mạch Vivado. Thiết kế áp dụng chuẩn giao tiếp AXI4, tận dụng khả năng truyền dữ liệu theo kiểu burst để tối ưu hiệu suất truy xuất. Nhờ đó, bộ điều khiển đảm bảo được tốc độ cao trong quá trình đọc/ghi dữ liệu, đồng thời duy trì tính chính xác và độ tin cậy cao trong giao tiếp với bộ nhớ.
- Thực hiện quá trình mô phỏng đã được nhằm đánh giá hoạt động thực tế của bộ điều khiển. Dữ liệu ghi vào bộ nhớ được kiểm tra lại bằng cách đọc ra và so sánh. Kết quả cho thấy dữ liệu đọc hoàn toàn trùng khớp với dữ liệu đã ghi, khẳng định thiết kế vận hành chính xác, đáp ứng đúng yêu cầu kỹ thuật và tuân thủ giao thức AXI4.

III. Những vấn đề cần giải thích thêm

Khả năng tùy biến IP-Core MIG: Học viên cần làm rõ mức độ can thiệp vào cấu trúc MIG để phù hợp với yêu cầu đặc thù hệ thống. Có thể tùy chỉnh timing, độ rộng bus, burst length hay không ?

So sánh hiệu năng với các giải pháp thương mại: Nếu so với IP controller của Xilinx hay ARM cung cấp sẵn, bộ điều khiển thiết kế đạt bao nhiêu % hiệu suất (thông lượng, latency) ? Có ưu điểm gì vượt trội ?

Khả năng ứng dụng thực tế: Với thiết kế hiện tại, có thể tích hợp được vào hệ thống thực (như camera AI, EDGE computing, ...) không ? Đã thử nghiệm trên board thực tế chưa, hay mới dừng ở mô phỏng ?

Bảo vệ dữ liệu và lỗi ECC: Thiết kế hiện tại có tích hợp chức năng sửa lỗi (ECC) hay chưa ? Nếu chưa thì hướng tích hợp sau này như thế nào ?

IV. Kết luận:

Đề án được triển khai nghiêm túc, khoa học và đúng hướng ứng dụng, thể hiện trình độ chuyên môn tốt về thiết kế hệ thống số, hiểu biết sâu về AXI4, SoC và mô hình bộ nhớ DDRAM3.

Học viên có khả năng tự nghiên cứu, triển khai thực nghiệm và giải quyết bài toán kỹ thuật có tính thực tế cao.

Đồng ý cho phép học viên bảo vệ đề án tốt nghiệp.

Hà Nội, ngày 15 tháng 7 năm 2025

NGƯỜI NHẬN XÉT

TS. Phạm Duy Phong

CÂU HỎI:

Câu hỏi 1: Giao thức AXI4 có 5 kênh truyền độc lập. Học viên hãy trình bày ý nghĩa từng kênh, đặc biệt là vai trò của cơ chế bắt tay VALID/READY trong đảm bảo độ ổn định truyền dữ liệu ?

Câu hỏi 2: Vì sao AXI4 lại được ưu tiên trong thiết kế điều khiển bộ nhớ ngoài DDR3 trên SoC so với giao thức Avalon ?

CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

BẢN NHẬN XÉT ĐỀ ÁN TỐT NGHIỆP THẠC SỸ

Đề tài: Nghiên cứu thiết kế bộ điều khiển bộ nhớ ngoài DDRAM3 sử dụng AXI4 trên hệ thống trên CHIP (SoS).

Chuyên nghành: Kỹ thuật điện tử -Mã số 8520203
Tên học viên: **Vương Viết Thao**- Học viện Công nghệ bưu chính viễn thông.
Họ và tên người nhận xét: PGS.TS Bạch Nhật Hồng.
Cơ quan công tác: Viện khoa học và công nghệ quân sự.

NỘI DUNG NHẬN XÉT

1.Cơ sở khoa học và thực tiễn, tính cấp thiết của đề tài

Trong bối cảnh các hệ thống nhúng và SoC hiện đại ngày càng tích hợp nhiều chức năng và xử lý dữ liệu tốc độ cao, việc truy cập bộ nhớ ngoài (đặc biệt là DDR3 SDRAM) thông qua một giao tiếp hiệu quả như AXI4 là một thành phần thiết yếu. Chuẩn AXI4 hiện được sử dụng rộng rãi trong các kiến trúc SoC thương mại (ARM, Zynq, Altera SoC…) nhờ vào tính linh hoạt, tốc độ cao và khả năng hỗ trợ các giao dịch song song. Việc nghiên cứu và thiết kế một bộ điều khiển DDR3 sử dụng AXI4 không chỉ giúp học viên hiểu sâu về cấu trúc hệ thống, mà còn hướng đến khả năng làm chủ công nghệ thiết kế phần cứng có khả năng tùy biến cao, phục vụ trực tiếp cho nhu cầu phát triển SoC, FPGA-based system và hệ thống nhúng hiện đại; Đồng thời gắn với xu hướng phát triển ngành bán dẫn tại Việt Nam. Vì vậy đề án có ý nghĩa khoa học, tính cấp thiết và khả năng ứng dụng thực tế.

2.Nội dung chất lượng của luận văn. Các kết quả đã đạt được so với đề cương đăng ký:

Đề án gồm 4 chương 70 trang và 11 tài liệu tham khảo, bao gồm các phần lý thuyết cơ bản và thiết kế mô phỏng. Cụ thể như sau:

+ Học viên trình bày cơ sở lý thuyết của bộ nhớ DDR3, kiến trúc SoC và chuẩn giao tiếp AXI4.

+ Phần thiết kế mô đun điều khiển sử dụng IP MIG 7 Series để tương tác với DDR3, đồng thời xây dựng giao tiếp AXI4 bằng các khối chức năng như: FIFO, máy trạng thái điều khiển đọc/ghi, định địa chỉ truy xuất..v.v

+ Hệ thống được mô phỏng và kiểm thử bằng công cụ Vivado, với kết quả hiển thị các bước truy xuất bộ nhớ chính xác theo logic AXI4. Dữ liệu ghi và đọc được kiểm tra khớp đúng, chứng minh

thiết kế hoạt động chính xác. Nhìn chung đề án có bố cục và kết cấu hợp lý; Nội dung, kết quả đạt được đáp ứng mục tiêu đặt ra và phù hợp với nội dung đề cương đã được phê duyệt.

3. Những vấn đề cần giải thích thêm.

Thiếu đánh giá hiệu năng định lượng: Học viên nên bổ sung các chỉ số như: độ trễ truy xuất, băng thông đạt được, số lượng giao dịch mỗi chu kỳ hệ thống… để so sánh với IP thương mại hoặc các thiết kế khác.

Chưa có mô tả chi tiết điều kiện mô phỏng phần cứng: Cần làm rõ thiết lập mô phỏng (loại FPGA, bus width, tần số hoạt động, số bit dữ liệu), vì điều này ảnh hưởng trực tiếp đến hiệu quả kiểm thử và khả năng triển khai thực tế.

Chưa đánh giá mức sử dụng tài nguyên phần cứng (LUTs, FFs, BRAMs): Nên thêm bảng tổng hợp tài nguyên sử dụng để đánh giá tính tối ưu và tiềm năng triển khai trên các hệ thống khác nhau.

Chưa có phần thử nghiệm lỗi hoặc đánh giá biên: Hệ thống nên được kiểm thử với các điều kiện biên như tín hiệu không ổn định, truy cập chồng lấn, hoặc ghi/đọc dữ liệu lớn. Còn lỗi in ấn chế bản

Câu hỏi 1: Học viên có thể trình bày các số liệu định lượng như băng thông truy xuất, độ trễ đọc/ghi, và tốc độ truy cập thực tế đạt được trong thiết kế không? Nếu không có, làm thế nào để đánh giá được mức độ tối ưu của thiết kế so với IP có sẵn?

Câu hỏi 2: Thiết kế hiện tại có hỗ trợ khả năng cấu hình động (parameterizable IP) để tương thích với nhiều loại DDR3 khác nhau (ví dụ tốc độ bus khác, kích thước burst khác)? Nếu mở rộng lên DDR4, những thay đổi chính trong giao tiếp và máy trạng thái là gì?

Câu hỏi 3: Mô phỏng hiện tại được thực hiện trên Vivado – nhưng chưa rõ học viên sử dụng board cụ thể nào, tần số hệ thống ra sao, bus dữ liệu bao nhiêu bit? Những điều kiện này ảnh hưởng thế nào đến độ tin cậy của kết quả? Học viên đã kiểm thử thiết kế với các tình huống bất thường (ví dụ xung nhịp không ổn định, trễ tín hiệu, xung cạnh ngắn) chưa?

Kết luận:. Nội dung và hình thức đáp ứng yêu cầu một đề án thạc sỹ kỹ thuật. Tôi đồng ý đề nghị cho học viên Vương Viết Thao được bảo vệ trước Hội đồng chấm đề án thạc sỹ,

Hà nội - Ngày 14 tháng 7 năm 2025

Người nhận xét

Bạch Nhật Hồng