ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

ĐỒ ÁN TỐT NGHIỆP
CAPSTONE PROJECT

NGÀNH: KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

ĐỀ TÀI:

NGHIÊN CỨU VÀ THIẾT KẾ SIDEBAND ADAPTER CHO HỆ THỐNG DIE-TO-DIE TRÊN CHIP

Người hướng dẫn: **TS. NGUYỄN KHÁNH QUANG**
KS. TRỊNH KHẮC DUY
Sinh viên thực hiện: **HOÀNG LÊ NHẬT TRƯỜNG**
MSSV: 105200477
LỚP: 20TDHCLC3

Đà Nẵng, 6/2025

TÓM TẮT

Tên đề tài: Nghiên cứu và thiết kế Sideband Adapter cho hệ thống die-to-die trên chip
Sinh viên thực hiện: Hoàng Lê Nhật Trường
Số thẻ sinh viên: 105200477 Lớp: 20TDHCLC3
Trong bối cảnh công nghệ bán dẫn ngày càng phát triển như hiện nay, yêu cầu về các hệ thống tích hợp với hiệu suất cao và năng lượng tiêu thụ thấp đang rất được chú trọng. Một trong những xu hướng thiết kế tiên tiến để đáp ứng những nhu cầu này là kỹ thuật kết nối Die-to-Die, cho phép ghép nối nhiều dies (khối silicon) trên cùng một package. Kiến trúc này giúp cải thiện hiệu suất, tối ưu chi phí và linh hoạt hơn trong sản xuất so với thiết kế monolithic (chip đơn lẻ) truyền thống. Trong một hệ thống sử dụng nhiều dies, để đảm bảo sự phối hợp hiệu quả giữa các dies, không chỉ cần một giao thức truyền dữ liệu chính mà còn phải có một kênh phụ để trao đổi tín hiệu điều khiển, trạng thái và cấu hình. Kênh này thường được gọi là sideband, đóng vai trò hỗ trợ và đảm bảo sự đồng bộ giữa các dies trong hệ thống.
Việc thiết kế một Sideband adapter phù hợp đóng vai trò quan trọng trong việc đảm bảo khả năng mở rộng của hệ thống, giảm độ trễ giao tiếp và tối ưu hóa tiêu thụ năng lượng. Điều này đặc biệt quan trọng đối với các ứng dụng đòi hỏi băng thông cao, độ trễ thấp như AI, HPC (High-Performance Computing), mạng truyền thông và các hệ thống nhúng phức tạp.

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

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

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Hoàng Lê Nhật Trường Số thẻ sinh viên: 105200477

Lớp: 20TDHCLC3 Khoa: Điện Ngành: Kỹ thuật điều khiển và tự động hóa

1. Tên đề tài đồ án: Nghiên cứu và thiết kế Sideband Adapter cho hệ thống die-to-die trên chip.

2. Đề tài thuộc diện: ☐ Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện.

3. Các số liệu và dữ liệu ban đầu:

4. Nội đung các phần thuyết minh và tính toán:

Chương 1: Tổng quan về Sideband Adapter

Chương 2: Thiết kế Sideband Adapter

Chương 3: Kết quả thực nghiệm và đánh giá

5. Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):

6. Họ tên người hướng dẫn:	*Phần/ Nội dung:*
TS. Nguyễn Khánh Quang	- Giám sát tiến độ thực hiện. - Kiểm tra và cho ý kiến dựa trên phần công việc đã hoàn thành.
KS. Trịnh Khắc Duy	- Hướng dẫn kiến thức thực tế. - Cung cấp các tài liệu tham khảo.

7. Ngày giao nhiệm vụ đồ án: 17/2/2025

8. Ngày hoàn thành đồ án: 26/5/2025

Đà Nẵng, ngày tháng 06 năm 2025

Trưởng Bộ môn Tự động hoá **Người hướng dẫn**

TS. Giáp Quang Huy TS. Nguyễn Khánh Quang

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

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

PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP

(Phiếu dành cho người hướng dẫn/sinh viên)

Họ tên sinh viên: Hoàng Lê Nhật Trường Số thẻ SV : 105200477

Tên đề tài ĐATN: Nghiên cứu và thiết kế Sideband Adapter cho hệ thống die-to-die trên chip.

Họ tên người HD: Nguyễn Khánh Quang Đơn vị: Khoa Điện

Tuần	**Ngày**	**Khối lượng**		**GVHD ký tên**
		đã thực hiện (%)	**tiếp tục thực hiện (%)**	
1	17/2/2025	5%	95%	
2	24/2/2025	7%	93%	
3	3/3/2025	10%	90%	
4	10/3/2025	Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5	17/3/2025	20%	80%	
6	24/3/2025	30%	70%	
7	31/3/2025	50%	50%	
8	7/4/2025	Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
9	14/4/2025	70%	30%	
10	21/4/2025	75%	25%	
11	28/4/2025	85%	15%	
12	5/5/2025	Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13	12/5/2025	90%	10%	

14	19/5/2025	95%	5%	
15	26/5/2025	100%	0%	

LỜI NÓI ĐẦU

Với sự đồng ý của Khoa Điện, nhóm chúng tôi đã được nghiên cứu về đề tài: "Nghiên cứu và thiết kế Sideband Adapter cho hệ thống die-to-die trên chip".

Để hoàn thành đồ án này, chúng tôi đã nhận được sự chỉ bảo, hướng dẫn ân cần tỉ mỉ của thầy giáo hướng dẫn: TS. Nguyễn Khánh Quang cùng với sự hướng dẫn của anh Trịnh Khắc Duy thuộc công ty Synopsys.

Qua thời gian làm việc với thầy và anh hướng dẫn, tôi thấy mình trưởng thành nhiều và học hỏi thêm được nhiều kiến thức mới để có thể áp dụng ngay trong công việc. Thầy và anh không những đã hướng dẫn cho chúng tôi trong chuyên môn mà cũng còn cả phong cách, tác phong làm việc của một người kỹ sư vi mạch.

Chúng tôi xin chân thành bày tỏ lòng cảm ơn sâu sắc của mình đối với sự giúp đỡ quý báu đó của thầy giáo hướng dẫn. Tôi cũng xin cảm ơn các thầy, cô giáo trong Khoa Điện cùng anh hướng dẫn thuộc công ty Synopsys đã cho tôi những kiến thức như ngày hôm nay.

Tôi hiểu rằng hoàn thành một nghiên cứu, một đồ án tốt nghiệp kỹ sư không chỉ đòi hỏi kiến thức đã học được trong nhà trường, sự nhiệt tình, chăm chỉ trong công việc. Mà còn là cả một sự chuyên nghiệp, kinh nghiệm trong nghề. Tôi rất mong được sự chỉ bảo thêm nữa của các thầy, cô cũng như các anh thuộc công ty Synopsys.

Thời gian 5 năm học tại trường Đại học đã kết thúc và sau khi hoàn thành đồ án tốt nghiệp này, sinh viên chúng tôi sẽ là những kỹ sư trẻ phát triển ngành vi mạch vững mạnh. Tất cả nhưng kiến thức đã học trong 5 năm, đặc biệt là quá trình tham gia đồ án tốt nghiệp đã tạo cho chúng tôi sự tự tin để có thể bắt đầu công việc của một kỹ sư thiết kế vi mạch trong tương lai. Những kiến thức đó có được là sự hướng dẫn và chỉ bảo tận tình của các thầy giáo, cô giáo cũng như các anh thuộc công ty Synopsys.

Tôi xin chân thành cảm ơn!

LỜI CAM ĐOAN LIÊM CHÍNH HỌC THUẬT

Tôi xin cam đoan rằng đồ án tốt nghiệp với đề tài "Nghiên cứu và thiết kế Sideband Adapter cho hệ thống die-to-die trên chip" là nghiên cứu độc lập của chúng tôi với sự hỗ trợ từ giảng viên hướng dẫn TS. Nguyễn Khánh Quang và công ty Synopsys. Chúng tôi xin cam đoan toàn bộ số liệu được cung cấp từ báo cáo đều là của công ty và đây là kết quả nghiên cứu hoàn toàn trung thực, không sao chép từ bất kì một công trình nghiên cứu khác nào. Những tài liệu trích dẫn đều đã được ghi rõ nguồn gốc. Chúng tôi xin chịu hoàn toàn trách nhiệm nếu có bất kỳ sự sao chép, gian dối kết quả nào trong sản phẩm đồ án này.

Sinh viên thực hiện

Hoàng Lê Nhật Trường

MỤC LỤC

DANH SÁCH CÁC BẢNG, HÌNH VẼ

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

KÝ HIỆU	CHỮ VIẾT TẮT
Soc	System on chip
RTL	Register Transfer Level
HDL	Hardware Description Language
HPC	High-Performance Computing
FIFO	First in First out
CDC	Clock Domain Crossing
CA	Completion Arborted
FSM	Finite state machine
UR	Unsupport Request
CS	Completion Success
CP	Control parity
DP	Data parity
SPI	Serial Peripheral Interface
I2C	Inter-Integrated Circuit
UCIe	Universal Chiplet Interconnect Express
MSB	Most Significant Bit
LSB	Least Significant Bit
PHY	Physical
DUT	Design Under Test
DUA	Design Under Analyze
STA	Static Timing Analysis
DFT	Design For Test
DRC	Design Rule Check

MỞ ĐẦU

1. Mục đích thực hiện đề tài:

Hiện nay, có nhiều phương pháp để thiết kế kênh sideband trong hệ thống die-to-die, mỗi phương pháp đều có đặc điểm riêng về hiệu suất, độ tin cậy và khả năng mở rộng. Một số giải pháp phổ biến bao gồm việc sử dụng giao thức giao tiếp nối tiếp như I2C, SPI để truyền dữ liệu điều khiển giữa các die. Trong đó, I2C là một lựa chọn đơn giản, dễ triển khai nhưng có băng thông thấp, phù hợp cho các tín hiệu cấu hình. SPI cung cấp tốc độ cao hơn và khả năng truyền dữ liệu song song nhưng yêu cầu nhiều dây tín hiệu hơn. Một số giao thức liên kết chiplet cũng tích hợp kênh sideband để hỗ trợ truyền tín hiệu điều khiển và giám sát. Chẳng hạn, trong PCIe, kênh SMBus/I2C được sử dụng để truyền thông tin quản lý, trong khi các hệ thống sử dụng giao thức liên kết chiplet khác có thể có kênh sideband riêng biệt để điều khiển trạng thái liên kết hoặc trao đổi thông tin điều khiển giữa các die. Ngoài ra, một số phương pháp sử dụng tín hiệu GPIO đơn giản để trao đổi tín hiệu trạng thái nhưng bị hạn chế về số lượng tín hiệu có thể truyền.

Dựa trên các giải pháp hiện có, dự án này đề xuất thiết kế một sideband adapter giúp hai dies có thể giao tiếp hiệu quả mà không tiêu tốn quá nhiều tài nguyên. Giải pháp này sẽ sử dụng mô hình truyền dữ liệu tối ưu với cơ chế quản lí nguồn thông minh, đảm bảo tính mở rộng cho hệ thống.

2. Mục tiêu của đề tài:

Mục tiêu của dự án này là thiết kế một khối sideband adapter, giúp các adapter ở dies có thể nói chuyện với nhau. Gồm có các chức năng sau:

- Trao đổi quyền truy cập thanh ghi từ Adapter đến Adapter hoặc là Adapter đến PHY cục bộ (bao gồm cả lệnh đọc và ghi thanh ghi).
- Trao đổi các gói message từ Adapter này sang Adapter kia.
- Báo lỗi nếu một quyền truy cập thanh ghi phản hồi quá chậm (time out), kiểm tra lỗi và các chế độ giám sát khác.

3. Phạm vi và đối tượng nghiên cứu:

Đối tượng nghiên cứu:

- Sideband adapter và các đặc điểm kỹ thuật của nó.
- CDC, 4 ways hand shaking, arbiter, FIFO.

Phạm vi nghiên cứu:

- Thiết kế Sideband adapter đáp ứng tiêu chuẩn công nghiệp, đảm bảo các tính năng của nó.

- Áp dụng các công cụ và ngôn ngữ mô tả phần cứng (HDL) như System Verilog.
- Sử dụng các phần mềm mô phỏng của Synopsys như Verdi, Spyglass, Design Compile Ultra và DFTMAX.

4. Phương pháp nghiên cứu:

Để đánh giá hiệu quả của sideband adapter, các phương pháp nghiên cứu sẽ tập trung vào chức năng, hiệu suất và độ tin cậy.

Về chức năng, adapter cần đảm bảo thực hiện đầy đủ các nhiệm vụ bao gồm trao đổi quyền truy cập thanh ghi, truyền message giữa các adapter và cơ chế báo lỗi khi truy cập thanh ghi phản hồi quá chậm. Việc kiểm tra được thực hiện thông qua testbench mô phỏng để xác nhận rằng tất cả các chức năng hoạt động đúng như thiết kế.

Về hiệu suất, các chỉ số đánh giá bao gồm độ trễ khi truyền lệnh truy cập thanh ghi và message, băng thông tối đa của kênh sideband cũng như mức sử dụng tài nguyên phần cứng khi triển khai. Việc đo lường hiệu suất được thực hiện thông qua mô phỏng timing, kiểm tra tải giao tiếp trong điều kiện thực tế và phân tích kết quả tổng hợp phần cứng.

Về độ tin cậy, cần đánh giá khả năng phát hiện lỗi, tỷ lệ lỗi bit khi truyền dữ liệu và khả năng phục hồi khi xảy ra sự cố. Phương pháp kiểm tra bao gồm mô phỏng lỗi (fault injection) để xác minh cơ chế phát hiện lỗi và kiểm tra hệ thống với dữ liệu bị nhiễu để đánh giá tính ổn định.

Với việc tập trung nghiên cứu các tiêu chí này, việc phát triển sideband adapter có thể đảm bảo tính chính xác, hiệu suất cao và hoạt động ổn định trong hệ thống die-to-die.

5. Cấu trúc của đồ án tốt nghiệp:

- NHẬN XÉT CỦA NGƯỜI HƯỚNG DẪN
- NHẬN XÉT CỦA NGƯỜI PHẢN BIỆN
- TÓM TẮT
- NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP
- PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP
- LỜI NÓI ĐẦU
- CAM ĐOAN
- MỤC LỤC
- DANH SÁCH CÁC BẢNG, HÌNH VẼ
- DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT
- MỞ ĐẦU
- CHƯƠNG 1: TỔNG QUAN VỀ SIDEBAND ADAPTER

- CHƯƠNG 2: THIẾT KẾ SIDEBAND ADAPTER
- CHƯƠNG 3: KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ
- KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN
- TÀI LIỆU THAM KHẢO
- PHỤ LỤC

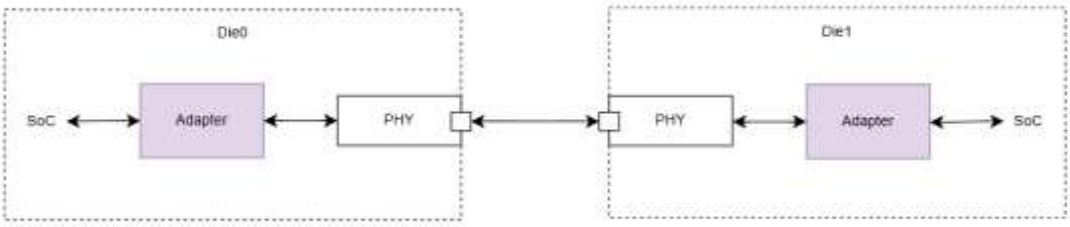
Chương 1: TỔNG QUAN VỀ SIDEBAND ADAPTER

1.1. Giới thiệu

Chương này cung cấp một cái nhìn tổng quan về hệ thống die-to-die trong quy trình sản xuất chip hiện đại, bao gồm các bước chính trong quy trình dự án từ khởi đầu đến hoàn thiện, nhấn mạnh vai trò của việc tối ưu hóa kết nối và tương tác giữa các die để nâng cao hiệu suất và độ tin cậy của sản phẩm. Đồng thời, chương còn trình bày các yêu cầu cụ thể đối với Sideband adapter trong thực tế, như khả năng truyền dữ liệu ổn định, phù hợp với tiêu chuẩn truyền thông cao cấp, đồng thời đảm bảo tích hợp linh hoạt và dễ bảo trì để đáp ứng các tiêu chí về hiệu suất và độ bền trong môi trường sản xuất tiên tiến.

1.2. Hệ thống die-to-die

Các kết nối giữa các chip trong cùng một gói (die-to-die interconnects) đánh dấu bước tiến quan trọng trong công nghệ bán dẫn, cho phép giao tiếp hiệu quả giữa nhiều chip silicon trong một gói duy nhất. Sáng kiến này giải quyết những thách thức của thiết kế chip đơn khối truyền thống bằng cách cung cấp hiệu suất cải thiện, chi phí hợp lý và khả năng mở rộng cao. Tiêu chuẩn UCIe đóng vai trò then chốt trong lĩnh vực này, giúp kết nối băng thông cao giữa các chiplet đa dạng về chức năng [1][2].



Hình 1.1: Vị trí adapter trong hệ thống die-to-die

Nhu cầu về các kết nối giữa các chip trong cùng một gói (die-to-die interconnects) được thúc đẩy bởi một số yếu tố chính.

Thứ nhất, nhu cầu ngày càng tăng về băng thông cao hơn, cần thiết cho các hoạt động như xem phim 4K trực tuyến và chơi game trực tuyến, đòi hỏi tốc độ truyền dữ liệu cao mà các kết nối die-to-die cung cấp, đảm bảo trải nghiệm sống động cho người dùng.

Thứ hai, khi định luật Moore gần đạt giới hạn vật lý của nó, những thách thức về quy mô truyền thống buộc phải có những giải pháp sáng tạo, các kết nối die-to-die giúp nâng cao hiệu suất mà không bị giới hạn bởi quy mô truyền thống.

Hơn nữa, khi các công nghệ sản xuất tiên tiến trở nên đắt đỏ hơn, việc phân chia các chip ASIC lớn thành các thành phần nhỏ hơn giúp tối ưu hóa chi phí, nâng cao hiệu

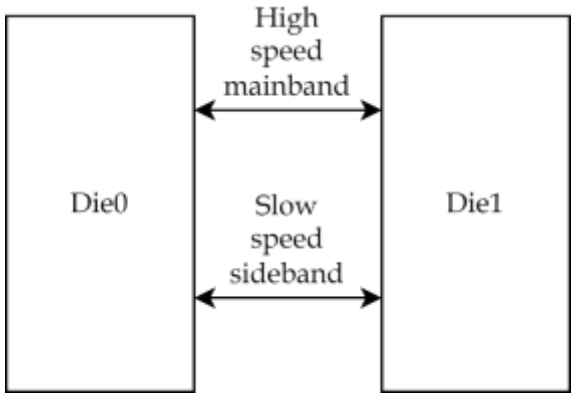
suất và sử dụng silicon hiệu quả hơn. Phương pháp thiết kế theo mô-đun này còn giúp tăng tính linh hoạt để thích ứng với nhu cầu thị trường và thúc đẩy phát triển công nghệ nhanh hơn. Ngoài ra, các kết nối die-to-die cung cấp khả năng truyền dữ liệu cao và độ trễ thấp cần thiết cho các ứng dụng đòi hỏi cao như các multi-terabit Ethernet switches và tích hợp quang học.

Cuối cùng, nỗ lực của các tổ chức trong việc thiết lập các tiêu chuẩn ngành cho các giao diện die-to-die là rất quan trọng để đảm bảo khả năng tương tác và tối đa hóa hiệu suất trên các hệ thống khác nhau [3].

Khi ngành công nghiệp bán dẫn hướng tới nâng cao hiệu suất và tiết kiệm năng lượng, các kết nối giữa các chip trong cùng một gói (die-to-die) đóng vai trò quan trọng để đạt được những mục tiêu này. Các công ty lớn như ARM, Meta, Intel và Samsung đã áp dụng phương pháp này để cải thiện các giải pháp bán dẫn của mình. Bằng cách cung cấp một giao diện tiêu chuẩn, UCIe giúp dễ dàng tương tác liền mạch giữa các chiplet từ các nhà cung cấp khác nhau, thúc đẩy chiến lược thiết kế đa chip linh hoạt và mở rộng theo mô-đun [2].

Hình 1.2: Các công ty sử dụng die-to-die

Khi kết nối 2 chip với nhau, dữ liệu tốc độ cao được truyền thông qua các nhóm đường truyền gọi là mainband. Lợi ích của nhóm này là nó có băng thông cao để phục vụ truyền dữ liệu lớn. Tuy nhiên, nhóm này yêu cầu được đào tạo để làm việc bình thường, do đó nó đòi hỏi sự giúp đỡ từ các nhóm đường truyền khác gọi là sideband. Nhóm Sideband cũng phục vụ trao đổi tham số và các mục đích cấu hình khác, sẽ được giải thích trong phần sau. Các đường sideband với tốc độ thấp và thông thường chỉ có một số cổng. Băng thông của sideband là thấp, nhưng có thể chấp nhận được vì việc sử dụng các làn này đang trao đổi dữ liệu không yêu cầu tốc độ [1].



Hình 1.3: Kết nối Die-to-die

Chú ý: Phy là một cây cầu kết nối 2 die trực tiếp. Mặc dù Adapter thực hiện nhiệm vụ truyền dữ liệu, nhưng nó không kết nối trực tiếp với giao diện của chip. Adapter có khả năng điều khiển PHY hoạt động ở chế độ yêu cầu, truy cập vào thanh ghi của PHY là một phần chức năng của nó.

1.3. Thông số của giao diện

Sideband adapter tương tác với nhiều giao diện khác nhau, bảng dưới đây sẽ mô tả chi tiết các giao diện đó:

Bảng 1.1: Thông số giao diện

Pin Name	Direction	Clock	Frequency	Description
CTL control port				
clk	in	-	2GHz	Clock.
sclk	in	-	20MHz	Clock.
rst_n	in	Async	-	Asynchronous reset, active low.
init	in	clk	-	Sync init, active high.
Sideband Interface				
pl_data[15:0]	in	clk	-	Data from PHY to the Adapter.
pl_valid	in	clk	-	Inform the data Phy2Adapter is valid.
pl_crd	in	clk	-	Credit from PHY to the Adapter.
lp_data[15:0]	out	clk	-	Same above but direction from Adapter to PHY.
lp_valid	out	clk	-	
lp_crd	out	clk	-	
CSR control				
csr_addr[11:0]	out	clk	-	Address to access.
csr_wdata[31:0]	out	clk	-	Write data.
csr_rdata[31:0]	in	clk	-	Read data.
csr_ce	out	clk	-	Enable to access CSR.

csr_we	out	clk	-	Write enable, valid when csr_ce=1 • 0 Read operation • 1 Write operation
csr_done	in	clk	-	Read operation
Mailbox info				
mb_req_valid_qst	in	Async	-	Mailbox register access operation request to remote Adapter. This is 4 ways handshaking signal with mb_resp_valid.
mb_req_addr_qst	in	Async	-	Address of the register.
mb_req_we_qst	in	Async	-	Write operation. • 0 Read operation • 1 Write operation
mb_req_data_qst[31:0]	in	Async	-	Write data.
mb_resp_valid	out	clk	-	Response valid.
mb_resp_sts	out	clk	-	Status of the operation. • 0 Error (see response in mb_resp_data) • 1 Done
mb_resp_data[63:0]	out	clk	-	Full sideband packet that is responded from the Remote Adapter or from the local PHY.
Parallel message interface				
msg_p2s_req	in	out	-	Parallel data to send to sideband.
msg_p2s_data[65:0]	in	out	-	
msg_p2s_ack	out	out	-	
msg_s2p_req	out	out	-	Sideband data that is converted to parallel interface.
msg_s2p_data[65:0]	out	out	-	
msg_s2p_ack	in	out	-	
Error signals				
op_e	out	clk	-	Operation error, check status bits. Only clear with sync init.
op_e_sts[N]	out	clk	-	Error status, each bit represents 1 error.

				Defined by implement solution.
Scan				
scan_rst[1:0]	in	Asyn	-	Scan Reset.

1.4. Giao thức Sideband

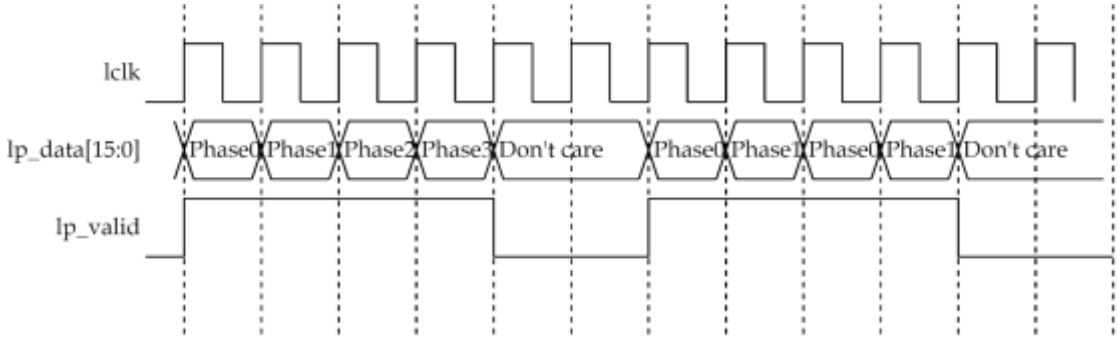
1.4.1. Giao thức của giao diện

• lp_*: Tín hiệu từ lớp logic sang lớp vật lý, được gửi từ Adapter đến PHY.

• pl_*: Tín hiệu từ lớp vật lý sang lớp logic, được gửi từ PHY đến Adapter.

Khi tín hiệu valid hợp lệ, dữ liệu trên giao diện là dữ liệu hợp lệ và 1 đoạn được gửi trên 1 chu kỳ đồng hồ. Tín hiệu valid hợp lệ nên được giữ cao cho đến khi tất cả các giai đoạn của gói đã được gửi. Các gói back-to-back được cho phép. Sau đó, có 2 gói back-to-back với 2 đoạn được gửi.

Số lượng pha được mã hóa trong trường opcode ở bảng opcode.



Hình 1.4: Giao thức sideband

pl_crd/lp_crd được gửi khi khối nhận đã gửi thành công 1 gói tin.

- Sau khi đặt lại, mặc định credit là 2. Nghĩa là Adapter/Receiver có thể gửi 2 gói tin liên tục mà không lo lắng PHY sẽ bị tràn và ngược lại.
- PHY trả lại pl_crd (Adapter trả lại lp_crd) khi gửi thành công gói tin (4 đoạn hoặc 2 đoạn) và sẵn sàng gửi gói tin mới.
- Adapter nên có một bộ đếm để đếm số lượng credit trong PHY. Nó chỉ gửi gói tin khi bộ đếm là không phải 0.

1.4.2. Gói tin truy cập thanh ghi

Bảng dưới đây dưới đây cho thấy định dạng của gói tin yêu cầu truy cập thanh ghi:

Bảng 1.2: Thông tin gói tin truy cập thanh ghi

Byte	**3**				**2**				**1**				**0**			
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Header/Data	**Header**															
Phase 0	srcid[1:0]		tag[1:0]		cr	Reserved (all 0)							opcode[3:0]			
Phase 1	cp	dp	dstid[1:0]		addr[11:0]											

Header/Data	**Data**
Phase 2	data[15:0]
Phase 3	data[31:16]

Opcode khớp với thao tác ghi hoặc đọc bộ nhớ trong phần 1.4.5. Tại một thời điểm, chỉ có 2 gói yêu cầu truy cập đăng ký được phép. Điều đó có nghĩa là adapter cần chờ gói tin hoàn thành từ cái die khác trước khi gửi cái mới.

Adapter có credit là 4 khi đặt lại, đây là credit tối đa của Adapter. Lưu ý rằng credit này không phải là credit của giao diện sideband. Mỗi lần yêu cầu được gửi đi, một credit sẽ được sử dụng.

Trường "cr" trong gói trả về 1 credit cho remote Adapter.

Khi Adapter nhận được yêu cầu, nó sẽ xử lý lệnh và trả về thông báo hoàn tất truy cập Register.

Bảng 1.3: Thông tin gói tin trả về

Byte	**3**				**2**				**1**				**0**			
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Header/Data	**Header**															
Phase 0	srcid[1:0]		tag[1:0]		cr	Reserved (all 0)							opcode[3:0]			
Phase 1	cp	dp	dstid[1:0]		Reserved (all 0)										status[2:0]	
Header/Data	**Data**															
Phase 2	data[15:0]															
Phase 3	data[31:16]															

Bảng dưới đây cho thấy các trường của gói tin:

Bảng 1.4: Các trường của gói tin

Field	**Description**
srcid	See other tables
dstid	See other tables
opcode	See other tables
cp	Even parity bit for header
dp	Even parity bit for data
addr	Write/Read register packet: Address to write or read
tag	Tag to match the completion with request.
status	Completion packet: Status of the request • 00b: Completion Success (CS) the request is completed sucessfully. This status returns with or without data, depends on the request. • 01b: Unsupported Request (UR). This status always returns with data. The data contains the header of the request. It raises in cases: o Address not in range o Write to read only register

status	• 10b: Completion Aborted (CA). This status always returns with data. The data contains the header of the request. It raises in cases: o CP, DP error o Timeout o Any other errors • 11b: Stall (ST). When a completer needs more time to complete the request, it can send back Stall status for the requester resets its' timer.
data	Payload. Always 32bits in this design. This segment are available in Write request packet, Read Completion or Completion with UR or CA status.
cr	Credit for Adapter to Adapter request.

Địa chỉ phải nằm trong phạm vi được xác định như bên dưới hoặc gói tin trả về phải phản hồi với trạng thái UR như trong bảng dưới:

Bảng 1.5: Các vùng địa chỉ

Address Range	**Location**
000h-00Fh	Adapter
010h-01Fh	PHY
020h-02Fh	Adapter & PHY (LSB of data is in Adapter, MSB is PHY)
Other	Reserved. Send back UR if Adapter sees request within this range.

Nếu Adapter nhận được gói Register Access Request có địa chỉ trong vùng Mixed (Adapter & PHY), nó phải thực hiện các bước sau:

• Chuyển tiếp hoàn thành đến bộ điều khiển thanh ghi nội bộ.

• Chuyển tiếp yêu cầu đến PHY. Lưu ý rằng gói chuyển tiếp cần được cập nhật srcid và dstid.

• Gộp kết quả từ bộ điều khiển thanh ghi nội bộ và hoàn thành từ PHY cục bộ sau đó gửi hoàn thành đến remote adapter.

• Lưu ý khi nhận được hoàn thành, Adapter phải kiểm tra trường "tag" để khớp với yêu cầu của nó thay vì sử dụng "srcid" và "dstid".

1.4.3. Gói tin message

Bảng dưới đây thể hiện định dạng của gói tin message:

Bảng 1.6: Thông tin gói tin message

Byte	**3**				**2**				**1**				**0**			
Bits	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Header/Data	**Header**															
Phase 0	srcid[1:0]		Reserved		MsgCode[3:0]				Reserved				opcode[3:0]			
Phase 1	cp	dp	dstid[1:0]		MsgInfo[7:0]								MsgSubCode			

Header/Data	Data
Phase 2	data[15:0]
Phase 3	data[31:16]

Bảng dưới đây thể hiện các trường trong gói tin:

Bảng 1.7: Các trường trong gói tin message

Field	Description
srcid	See other tables
dstid	See other tables
opcode	See other tables
cp	Even parity bit for header
dp	Even parity bit for data
MsgCode[3:0]	Message Code. See other tables
MsgSubCode[3:0]	Message Sub Code. See other tables
MsgInfo[7:0]	Message Info. See other tables
data	Payload. Always 32bits in this design.

Tại một thời điểm, có 4 gói yêu cầu được phép. Có 2 loại gói tin nhắn, có dữ liệu và không có dữ liệu.

Bảng bên dưới hiển thị thông tin cho tin nhắn không có dữ liệu:

Bảng 1.8: Thông tin của tin nhắn không có dữ liệu

Message	MsgCode	MsgSubcode	MsgInfo	Description
{NOP.Crd}	0h	0h	0h: No credit return 1h: 1 credit return 2h: 2 credit return 3h: 3 credit return 4h: 4 credit return	No Operation message with credit for register access command.
{LinkMgmt.Active.Req}	1h	0h	Reserved	Request Adapter to move to following states: - Active - Low power - Retrain
{LinkMgmt.LP.Req}		1h		
{LinkMgmt.Retrain.Req}		2h		
{LinkMgmt.Active.Resp}	2h	0h	0h: Response Fh: Stall	Response to the request if MsgInfo = 0h. If MsgInfo = Fh, is
{LinkMgmt.LP.Resp}		1h		

{LinkMgmt.Retrain.Resp}		2h		received, the timeout timer reset to zero.

Bảng bên dưới hiển thị thông tin cho tin nhắn có dữ liệu:

Bảng 1.9: Thông tin của tin nhắn có dữ liệu

Message	Msg Code	MsgSubcode	Msg Info	Description
{AdvCap.Adapter}	0h	0h	0h	Advertise Adapter Capability. The data field is copied from CSR register. The request and response are the same for this packet.

Trong thiết kế này, Adapter chỉ chuyển tiếp tin nhắn, nó chỉ tập trung vào việc lưu trữ tin nhắn và chuyển tiếp đến đúng người nhận.

1.4.4. Timeout

Tất cả các gói yêu cầu sideband có thời gian chờ là 8ms, nếu bên hoàn thành không thể hoàn thành yêu cầu đúng hạn, bên này phải gửi lại phản hồi "Tạm dừng" để bên yêu cầu thiết lập lại bộ đếm thời gian của mình.

1.4.5. OPCODE, SRCID, DSTID

Bảng dưới đây thể hiện các trường opcode, srcid và dstid:

Bảng 1.10: Mô tả opcode, srcid và dstid

Field	Description
srcid	Source ID - 00b: From local adapter - 01b: From local PHY - 10b: Reserved - 11b: From remote PHY
dstid	Destination ID: - 00b: Reserved - 01b: To local PHY - 10b: To remote adapter - 11b: To remote PHY

Bảng 1.11: Thông tin giải mã opcode

Opcode encoding	Packet type
4'b0000	No data packet
4'b0001	Memory Write
4'b0010	Memory Read
4'b0011	Completion without data
4'b0100	Completion with data

4'b0101	Message without data
4'b0110	Message with data
Other	Reserved

1.5. Giao thức giao diện Adapter

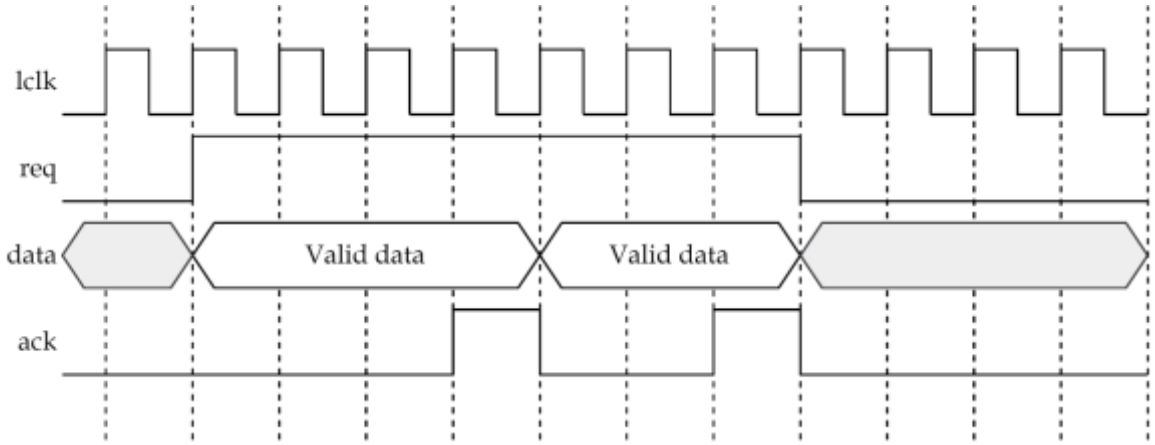
1.5.1. Giao diện Parallel message

Dữ liệu giao diện song song được sử dụng để trao đổi dữ liệu nội bộ nhằm đơn giản hóa việc truyền dữ liệu. Adapter được yêu cầu để chuyển đổi giao diện này sang giao thức sideband nối tiếp. Dữ liệu của giao diện này có 66 bit:

• Dữ liệu 64 bit cho 4 đoạn

• 2 bit cho số đoạn

Theo yêu cầu, dữ liệu được giữ cho đến khi tín hiệu xác nhận từ dữ liệu lên 1 (trong 1 chu kỳ).



Hình 1.5: giao thức request và acknowledge

1.5.2. Giao thức điều khiển CSR

Giao diện điều khiển enable (csr_ce) và done (csr_done) hoạt động giống như req/ack của giao diện parallel message.

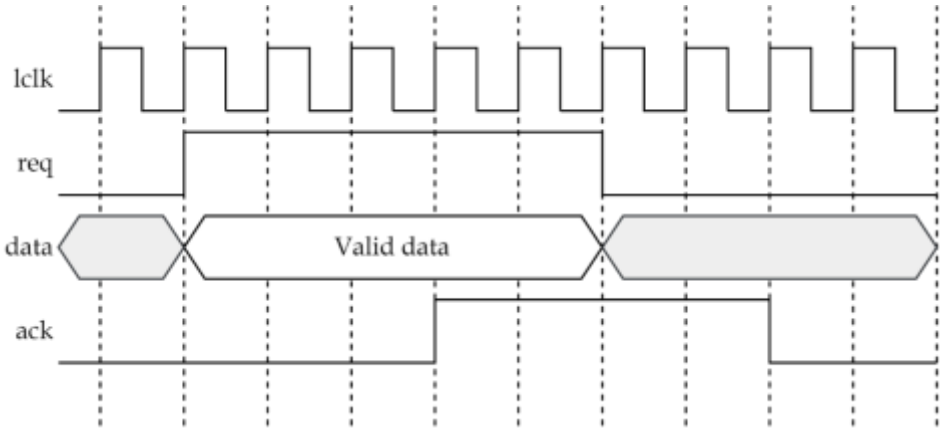
1.5.3. Yêu cầu từ mailbox

Trong dự án này, adapter chỉ gửi và nhận gói Register Access Request đến remote adapter bằng cách sử dụng yêu cầu mailbox từ thanh ghi CSR trực tiếp.

Theo yêu cầu của mb_req_valid_qst, adapter gửi yêu cầu và hoàn tất quy trình để cập nhật mb_resp_valid. 2 tín hiệu sử dụng giao thức 4 ways handshaking để trao đổi dữ liệu.

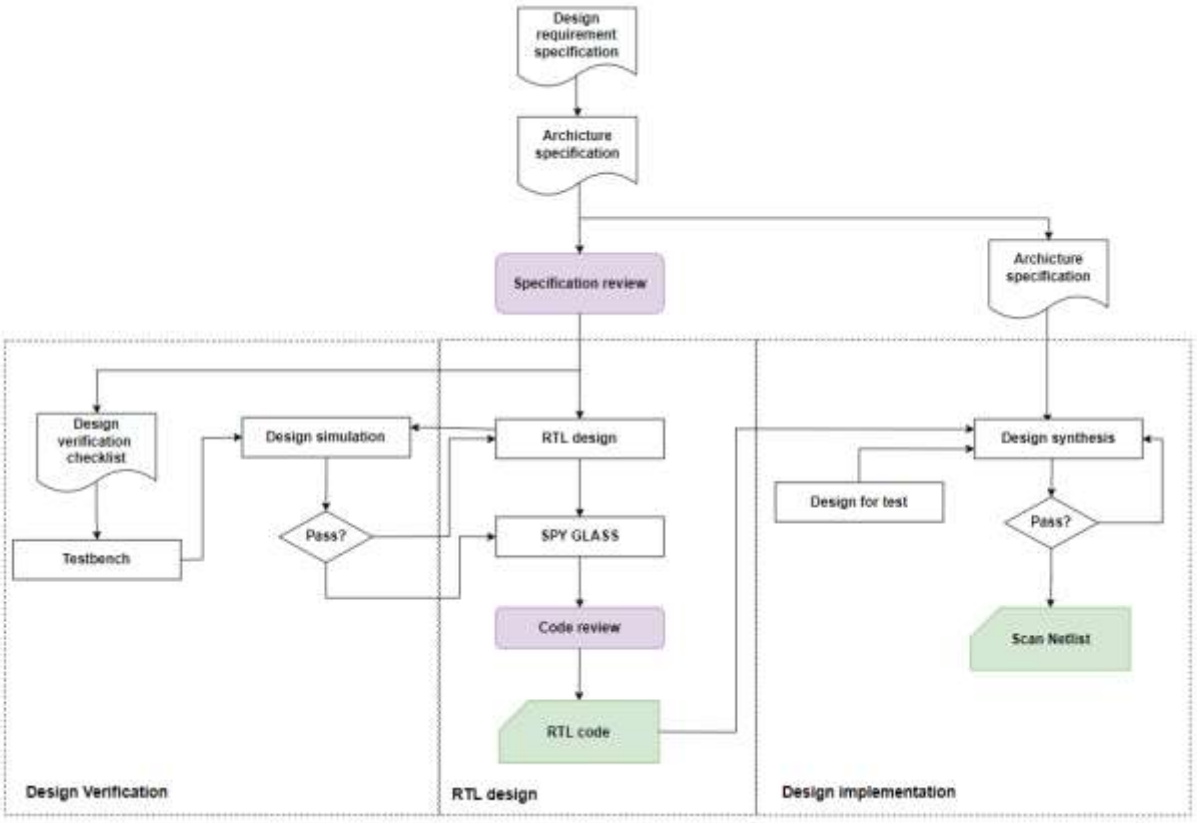
1.5.4. Giao thức 4 ways handshaking

Đối với dữ liệu không đồng bộ, cơ chế 4 ways handshaking req/ack được sử dụng để khắc phục sự cố xung nhịp giữa các miền.



Hình 1.6: Giao thức 4 ways handshaking

1.6. Quy trình thiết kế



Hình 1.7: Quy trình thiết kế

Đặc tả yêu cầu thiết kế: Đây là yêu cầu từ khách hàng hoặc các nhóm khác, mô tả chức năng và đặc điểm (tốc độ, diện tích, ...) của sản phẩm.

Đặc tả kiến trúc: Thiết kế kiến trúc của sản phẩm, chia thành các khối nhỏ hơn và chức năng cho từng khối.

Thiết kế RTL: Chuyển đổi kiến trúc từ dạng văn bản sang ngôn ngữ mô tả phần cứng (HDL) như Verilog hoặc SystemVerilog. Bước này có thể yêu cầu thiết kế kiến trúc ở quy mô nhỏ hơn, được gọi là thiết kế khối hoặc thiết kế kiến trúc vi mô.

Xác minh thiết kế: Xác minh xem mã RTL có khớp với đặc tả hay không. Bước này cũng sử dụng Verilog/SystemVerilog với các công cụ mô phỏng để xác minh thiết kế.

- Checklist: Bao gồm các mục để xác minh thiết kế.
- Testbench: Môi trường để xác minh thiết kế.
- Mô phỏng thiết kế: Đưa mã RTL vào môi trường và chạy mô phỏng để xác minh thiết kế.

Triển khai thiết kế: Chuyển đổi mã RTL thành các ô logic và tạo cơ sở dữ liệu vật lý để sản xuất.

- Synthesis: Chuyển đổi mã RTL thành các ô logic. Kết nối và chức năng của các ô vẫn được viết bằng cú pháp Verilog và được gọi là netlist cấp cổng (GLN). Sau giai đoạn này, có các bài kiểm tra để đảm bảo netlist và mã RTL khớp với nhau hoặc thiết kế có thể đáp ứng yêu cầu về hiệu suất, ...
- Thiết kế cho kiểm thử (DFT): Để xác minh thiết kế trong thế giới vật lý (khi được sản xuất), phải bao gồm các chức năng thử nghiệm bên cạnh các chức năng thông thường. Bước này thêm các chức năng vào thiết kế.

1.7. Kết luận

Chương này giúp chúng ta hiểu rõ khái niệm die-to-die và vai trò quan trọng của nó trong công nghệ sản xuất chip hiện đại, đồng thời cung cấp cái nhìn toàn diện về quy trình thiết kế chip từ những bước khởi đầu cho đến khi hoàn thiện sản phẩm, nhấn mạnh sự phối hợp chặt chẽ giữa các giai đoạn để đảm bảo chất lượng, hiệu suất và tối ưu hóa kết nối giữa các die trong hệ thống tích hợp cao.

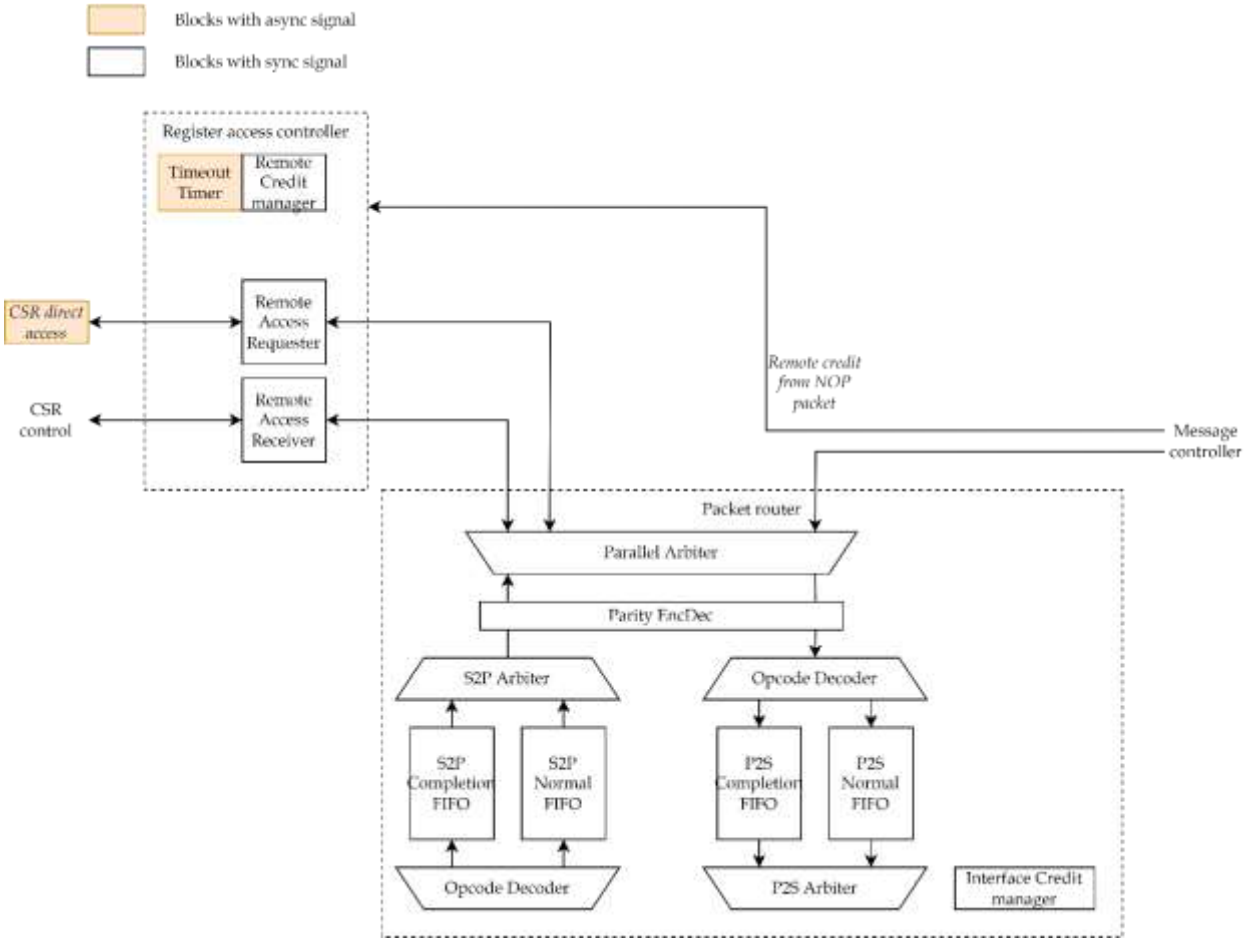
Chương 2: THIẾT KẾ SIDEBAND ADAPTER

2.1. Giới thiệu

Sideband adapter là một liên kết truyền thông đặc biệt cho phép hai dies (hoặc chip) trao đổi thông tin và phối hợp với nhau thông qua một khối điều khiển riêng biệt. Khối này xử lý ba chức năng chính liên quan đến truyền dữ liệu, giúp đảm bảo hai chip hoạt động cùng nhau một cách trơn tru và hiệu quả:

- Adapter chỉ gửi và nhận gói truy cập thanh ghi tới adapter ở die khác bằng mailbox request từ CSR direct access.
- Trao đổi gói tin message từ adapter này sang adapter ở die khác.
- Nhận và xử lí yêu cầu truy cập thanh ghi từ một die khác gửi đến.

Dưới đây là sơ đồ đề xuất ban đầu:



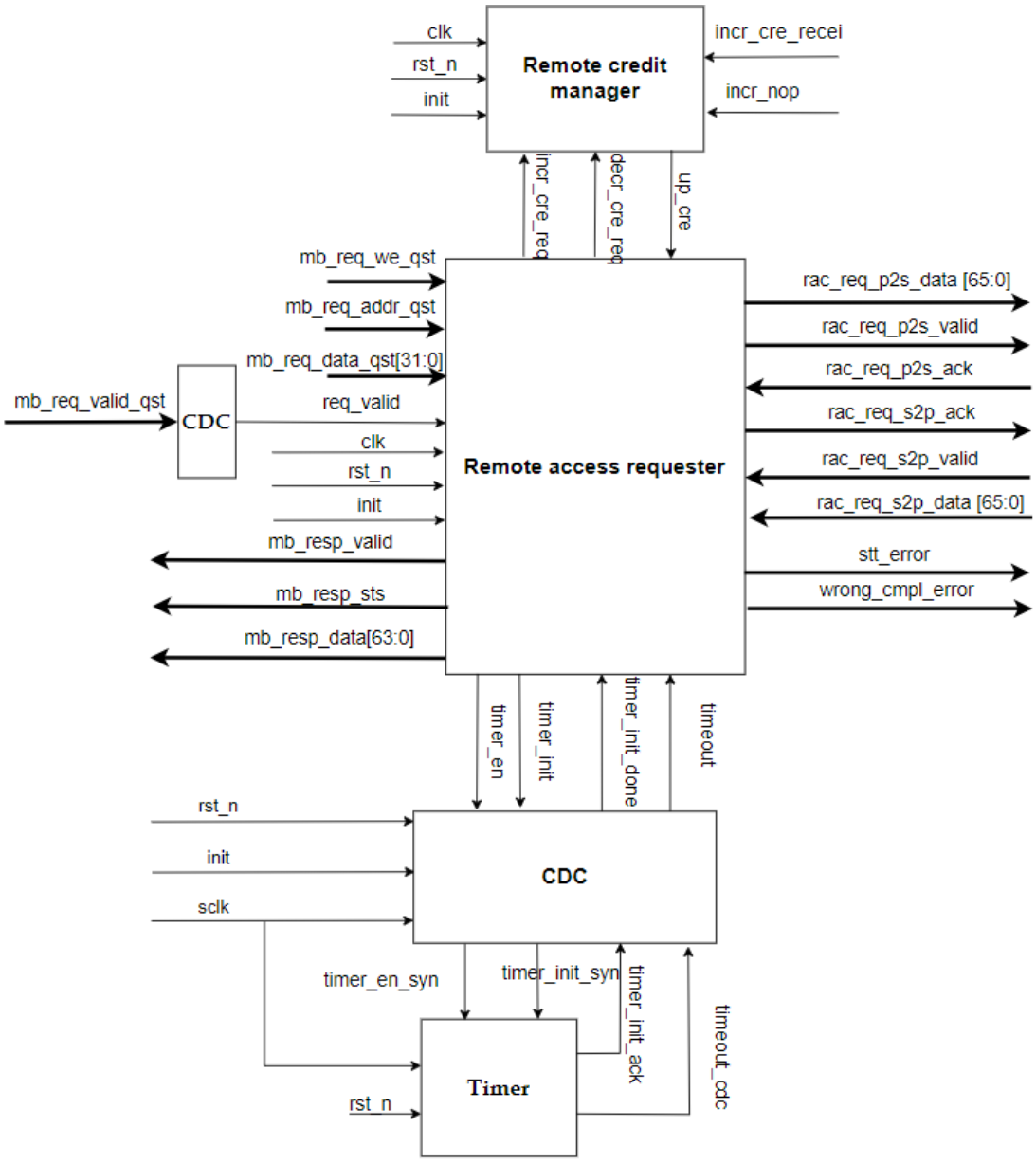
Hình 2.1: Đề xuất thiết kế

Chương này sẽ trình bày khái quát về các khối có trong hệ thống. Riêng khối Remote access requester, Remote credit manager và Timeout Timer sẽ được trình bày chi tiết về nguyên lý hoạt động, dạng sóng của từng khối nhỏ vì đó là phần mà tôi đảm nhiệm thiết kế.

2.2. Khối Remote access requester

2.2.1. Tổng quan

- Khi có yêu cầu được gửi tới, tín hiệu và dữ liệu sẽ được gửi tới thông qua một mailbox.
- Yêu cầu này sẽ được xử lý trong khối Remote access requester, nó sẽ kiểm tra credit và gửi tín hiệu để cập nhật số lượng credit trong khối Remote credit manager, cùng lúc đó nó sẽ gửi tín hiệu để kích hoạt bộ đếm thời gian và gửi gói tin yêu cầu tới khối tiếp theo.
- Khi có gói tin trả về từ giao diện parallel, khối Remote access requester này sẽ tiến hành kiểm tra credit để gửi về khối Remote credit manager và trạng trái của gói tin phản hồi để phản hồi về lại mailbox.



Hình 2.2: Mô tả tổng quan khối Remote access requester

2.2.2. Thông số và sơ đồ chi tiết của khối

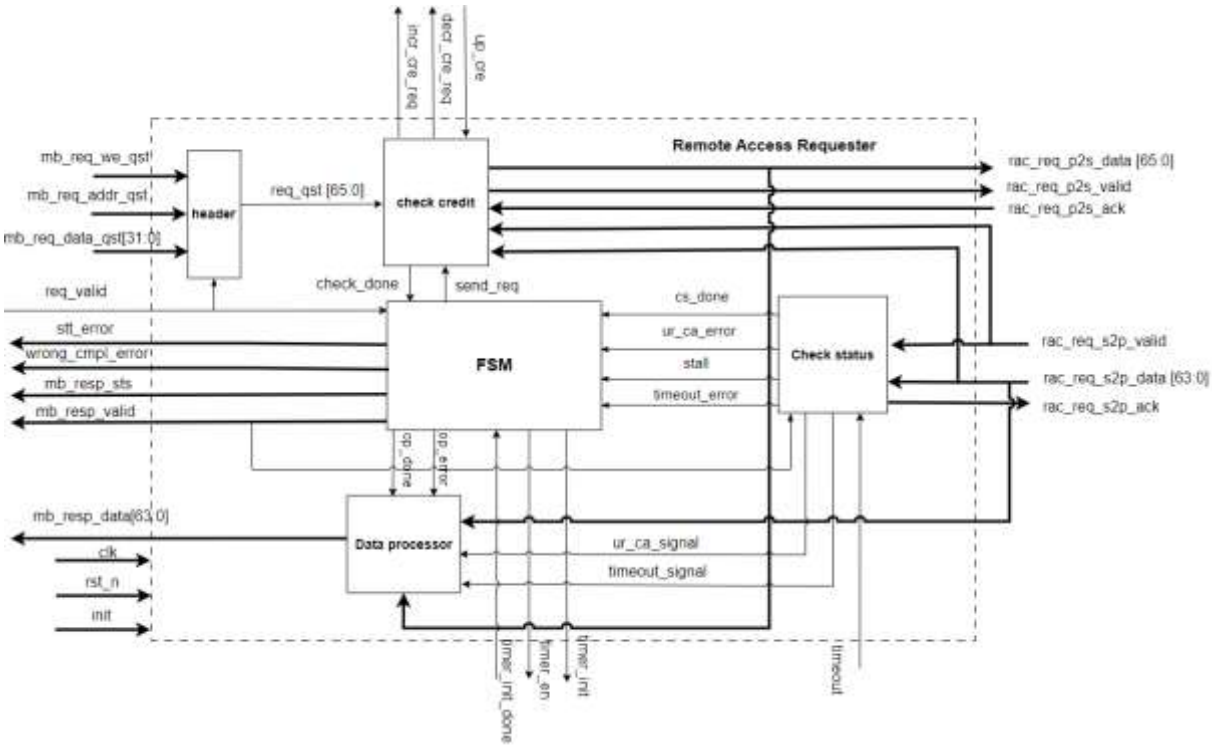
Bảng dưới đây mô tả đầy đủ các thông số của khối:

Bảng 2.1: Thông số của các giao diện Remote access requester

Pin name	Direction	Clock	Size	Description
Remote access requester interface ports				
CTL control port				
clk	Input	_	1	2GHz clock.
rst_n	Input	async	1	Asynchronous reset, active low.
init	Input	clk	1	Sync init, active high.
Mailbox info				
mb_req_we_qst	Input	async	1	Write operation.
mb_req_addr_qst	Input	async	12	Address signal of register.
mb_req_data_qst	Input	async	32	Write data.
mb_resp_valid	Output	clk	1	Response signal to mailbox.
mb_resp_sts	Output	clk	1	Status signal to mailbox .
mb_resp_data	Output	clk	66	Data bus response from the Remote access requester block.
Parallel interface				
rac_req_p2s_ack	Input	clk	1	Response signal from parallel interface after send request.
rac_req_s2p_data	Input	clk	66	Data bus signals of completion from parallel interface. The size is 66
rac_req_s2p_valid	Input	clk	1	Valid signal of completion from parallel interface.
rac_req_p2s_data	Output	clk	66	Data bus from the Remote access requester block to parallel interface. The size is 66
rac_req_p2s_valid	Output	clk	1	Valid request signal to parallel interface.
rac_req_s2p_ack	Output	clk	1	Ack valid respond to parallel interface.
Remote credit manager Interface Ports				
up_cre	Input	clk	4	Number of credit in Remote credit manager.
incr_cre	Output	clk	1	Increase credit signal to Remote credit manager block.
decr_cre	Output	clk	1	Decrease credit signal to Remote credit manager block.
CDC Interface Ports				
timeout	Input	clk	1	Time out signal from timer.
timer_init_done	Input	clk	1	Timer init acknoledge signal.
timer_init	Output	clk	1	Reset timer signal.
timer_en	Output	clk	1	Enable timer signal.

req_valid	Input	clk	1	Valid request signal after synchrnous from CDC block.
Error Interface				
stt_error	Output	clk	1	Error signal in operation.
wrong_cmpl_error	Output	clk	1	Error signal when do not send request yet but receive completion.

Dưới đây là sơ đồ chi tiết của khối:



Hình 2.3: Mô tả chi tiết khối Remote access requester

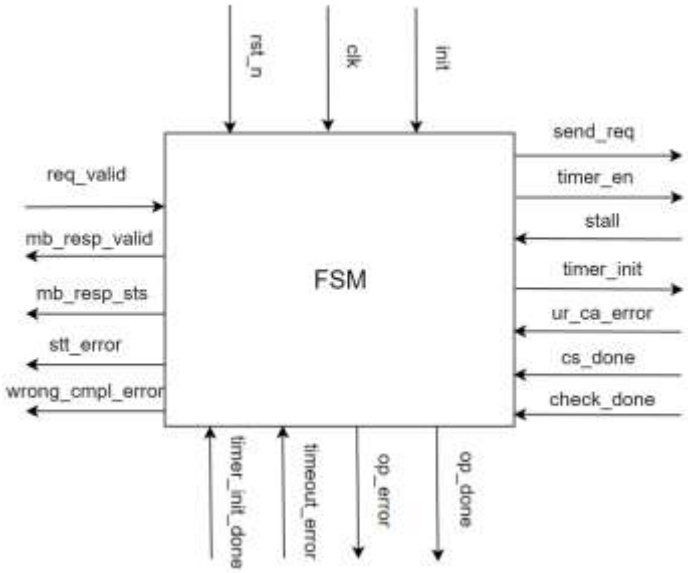
Khối Remote access requester bao gồm các khối con sau:

- Khối header dùng để tạo gói tin yêu cầu từ tín hiệu và dữ liệu từ mailbox.
- Khối check credit dùng để kiểm tra số lượng credit trong Remote credit manager và gửi tín hiệu để cập nhật số lượng credit có trong Remote credit manager.
- Khối check status dùng để kiểm tra trạng trái của gói tin trả về và gửi các tín hiệu trạng thái tới các khối FSM và data processor.
- Khối FSM dùng để điều khiển và giám sát trạng thái của khối Remote access requester.
- Khối data processor dùng để tạo gói tin phản hồi về lại mailbox.

Các khối này tương tác với nhau để vận hành và xử lý gói tin một cách tối ưu và chặt chẽ. FSM đóng vai trò như trung tâm của khối, tạo điều kiện thuận lợi cho việc hiểu và triển khai các hệ thống phức tạp bằng cách chỉ định các trạng thái, chuyển tiếp và hành động đầu ra [4].

2.2.3. Khối FSM

Khối này điều khiển và giám sát khi nào gửi gói tin yêu cầu và nhận gói tin trả về, cho biết các trạng thái của hệ thống.



Hình 2.4: Khối FSM của Remote access receiver

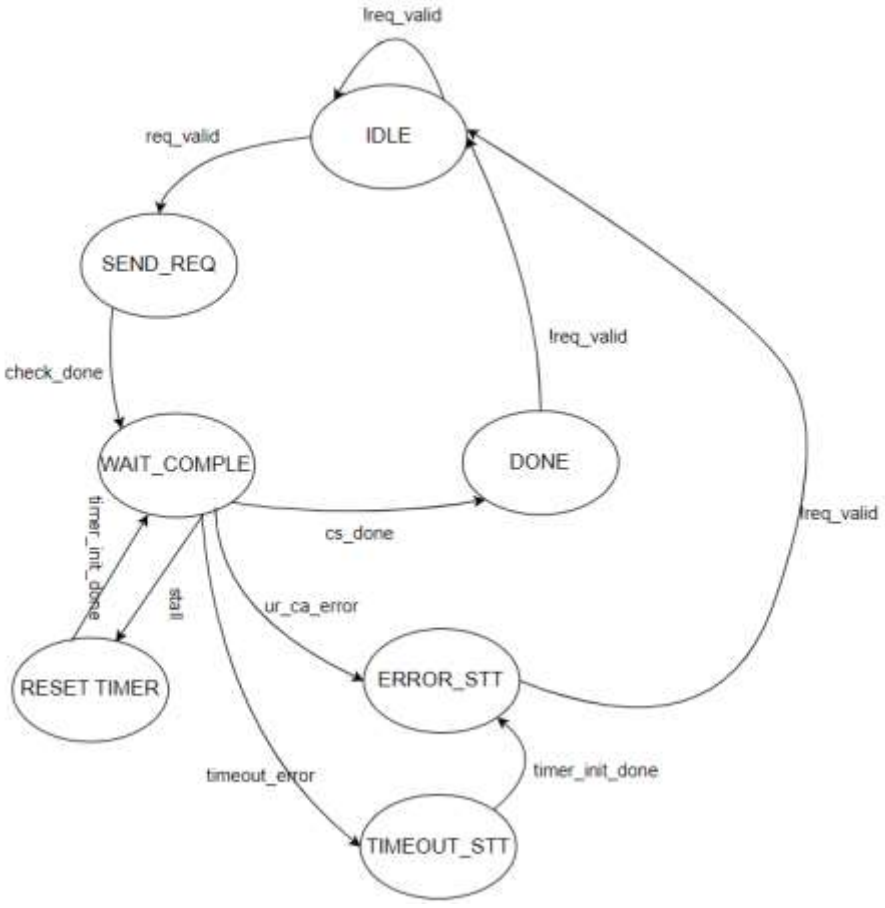
Bảng dưới đây mô tả chi tiết các thông số của khối:

Bảng 2.2: Mô tả chi tiết thông số của khối FSM

Pin name	Direction	Clock	Size	Description
FSM Ports				
clk	Input	_	1	2GHz clock.
rst_n	Input	Async	1	Asynchronous reset, active low.
init	Input	clk	1	Sync init, active high.
stall	Input	clk	1	Reset signal from check status block.
ur_ca_error	Input	clk	1	Error signal when status of completion is error.
cs_done	Input	clk	1	Done signal when status of completion is done.
check_done	Input	clk	1	Acknowledge signal from check_credit block when send_req is operate done.
req_valid	Input	clk	1	Valid signal from mailbox after synchronous.
timeout_error	Input	clk	1	Time out error signal from Check status block.
timer_init_done	Input	clk	1	Reset timer acknowledge signal.
mb_resp_sts	Output	clk	1	Status of operation respond to mailbox.
mb_resp_valid	Output	clk	1	Valid signal respond to mailbox.

timer_init	Output	clk	1	Reset signal send to timer.
timer_en	Output	clk	1	Enable signal send to timer.
send_req	Output	clk	1	Send request signal to check credit block.
op_done	Output	clk	1	Done signal send to data processor block to process data.
op_error	Output	clk	1	Error signal send to data processor block to process data.
stt_error	Output	clk	1	Operation error when status bit is error.
wrong_cmpl_error	Output	clk	1	Error signal when do not send request yet but receive completion.

Sơ đồ dưới đây mô tả các trạng thái của FSM:



Hình 2.5: Mô tả các trạng thái của FSM

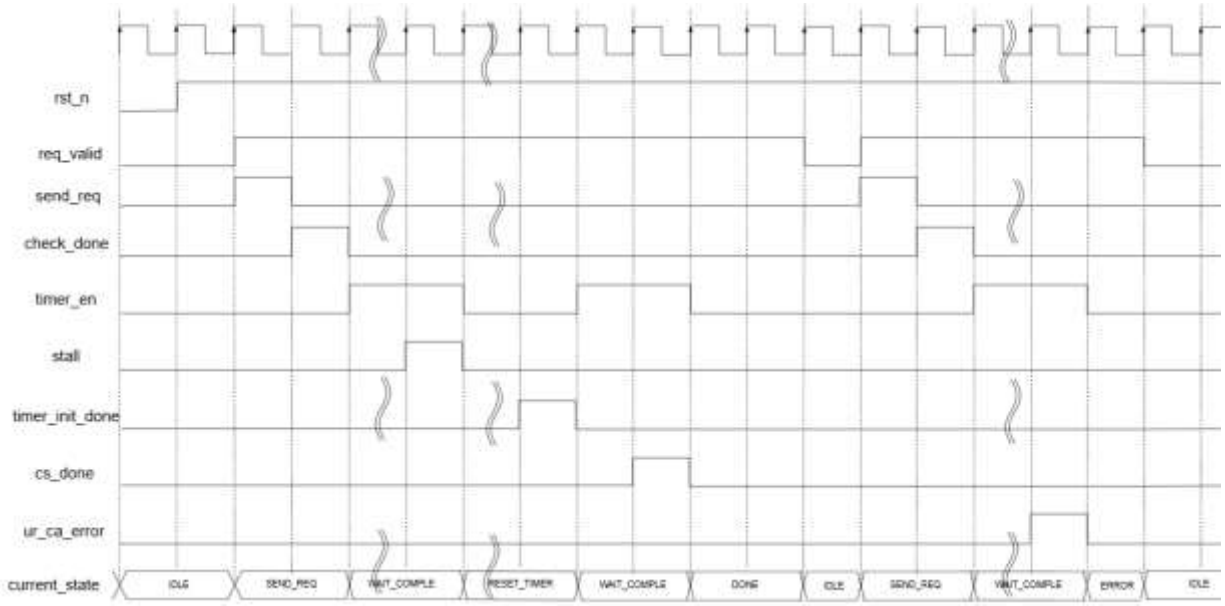
FSM hoạt động theo những trạng thái sau:

- IDLE: Đây là trạng thái ban đầu của FSM.
- SEND_REQ: Khi có tín hiệu req_valid từ mailbox, FSM sẽ gửi tín hiệu send_req tới khối check_credit để kiểm tra credit, sau khi kiểm tra xong FSM sẽ nhận được tín hiệu check_done và chuyển sang trạng thái tiếp theo là WAIT_COMPLE.
- WAIT_COMPLE: Ở trạng thái này, FSM sẽ chờ gói tin trả về.

  - Nếu FSM nhận được gói tin với tín hiệu stall, nghĩa là phía bên nhận yêu cầu cần thêm thời gian để xử lý và FSM sẽ gửi tín hiệu đến bộ đếm thời gian để reset thời gian.
  - Nếu FSM nhận được gói tin với tín hiệu cs_done, nghĩa là gói tin trả về đã hoàn thành và chuyển sang trạng thái DONE.
  - Nếu FSM nhận được gói tin với tín hiệu ur_ca_signal, nghĩa là gói tin trả về bị lỗi và chuyển sang trạng thái ERROR.
  - Nếu FSM nhận được gói tin với tín hiệu timeout_error_signal, nghĩa là đã quá thời gian mà không nhận được gói tin trả về và chuyển sang trạng thái TIMEOUT.
- RESET_TIMER: Nếu bên phía nhận yêu cầu không xử lý kịp, FSM sẽ nhận được tín hiệu stall để reset bộ đếm thời gian.
- TIMEOUT: Khi có tín hiệu timeout_error, FSM sẽ gửi tín hiệu để reset bộ đếm thời gian. Khi reset xong, nó sẽ chuyển sang trạng thái ERROR.
- ERROR: Khi có tín hiệu ur_ca_error hoặc timer_init_done, FSM sẽ gửi tín hiệu phản hồi về lại cho mailbox và tín hiệu op_error cho khối data_processor để xử lý gói tin trả về.
- DONE: Khi có tín hiệu cs_done, FSM sẽ gửi tín hiệu phản hồi về lại cho mailbox và tín hiệu op_done cho khối data_processor để xử lý gói tin trả về.

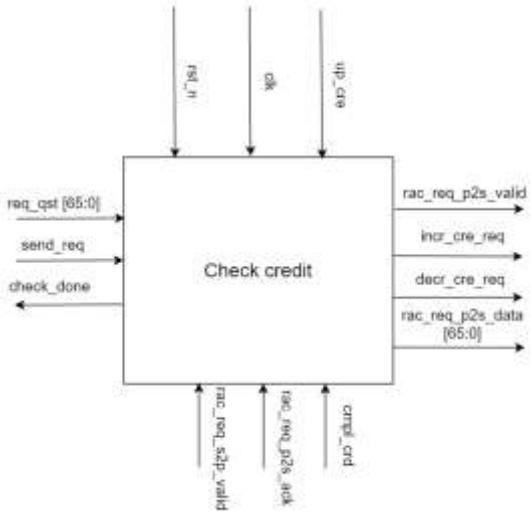
Chú ý: Khi ở trạng thái IDLE, tức là chúng ta chưa gửi gói tin yêu cầu. Nếu FSM nhận được gói tin trả về lúc này, tức là đây là gói tin lỗi. Lúc này FSM sẽ gửi tín hiệu wrong_cmpl_error đến giao diện Error.

Dưới đây là dạng sóng của khối:



Hình 2.6: Dạng sóng của khối FSM

2.2.4. Khối kiểm tra credit



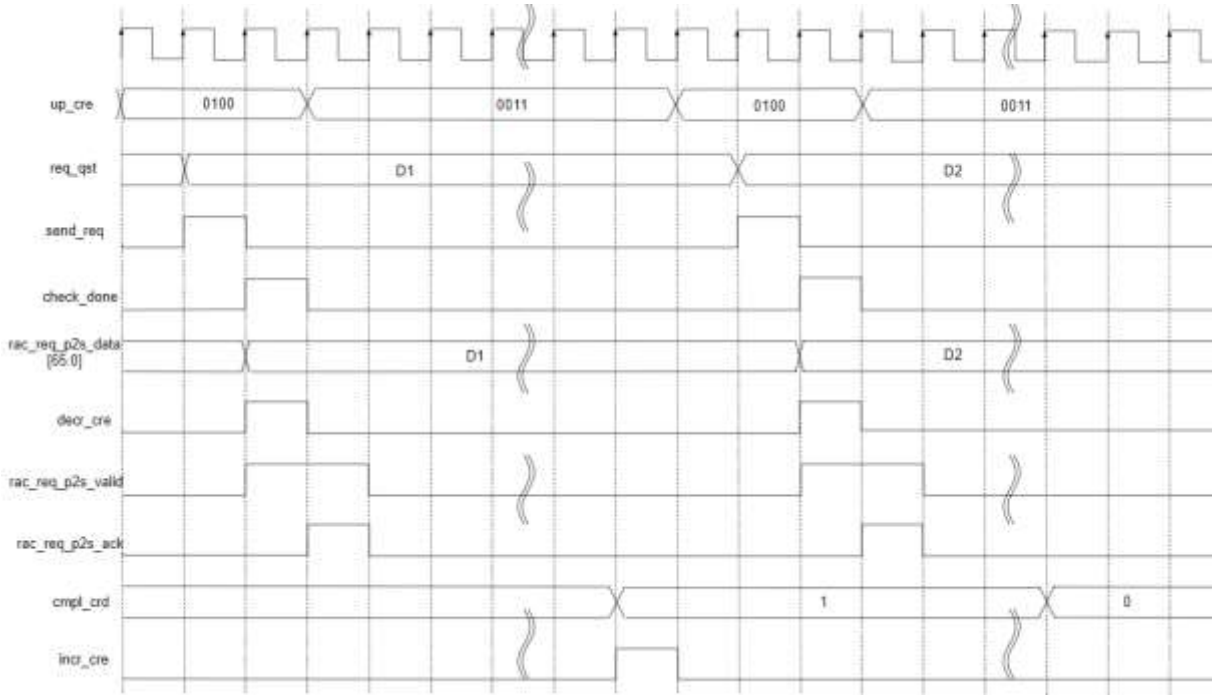
Hình 2.7: Khối kiểm tra credit

Khối này sẽ kiểm soát lúc nào sẽ gửi gói tin yêu cầu. Đồng thời nó sẽ gửi tín hiệu đến khối Remote credit manager để cập nhật credit. Khi nó nhận được tín hiệu send_req, nó sẽ kiểm tra credit trong khối Remote credit manager, nếu còn credit thì mới được gửi gói tin yêu cầu. Cùng lúc đó nó cũng sẽ gửi tín hiệu check_done về lại cho khối FSM.

Bảng 2.3: Mô tả chi tiết thông số của khối kiểm tra credit

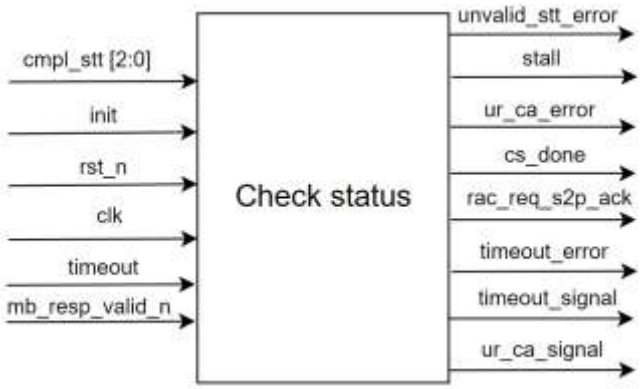
Pin name	Direction	Clock	Size	Description
Check credit Ports				
clk	Input	_	1	2GHz clock.
rst_n	Input	Async	1	Asynchronous reset, active low.
req_qst	Input	clk	66	Data bus signals from the header.
rac_req_p2s_ack	Input	clk	1	Response signal from parallel interface after send request.
up_cre	Input	clk	3	Number of credit in Remote credit manager.
cmpl_crd	Input	clk	1	Credit bit of completion.
rac_req_s2p_valid	Input	clk	1	Valid signal of completion from parallel interface.
send_req	Input	clk	1	Send request signal to operate check_credit block.
rac_req_p2s_data	Output	clk	66	Remote access request send to parallel interface .
rac_req_p2s_valid	Output	clk	1	Valid signal of remote access request
check_done	Output	clk	1	Acknowledge signal send to FSM.
decr_cre_req	Output	clk	1	Decrease credit signal to Remote credit manager block.
incr_cre_req	Output	clk	1	Increase credit signal to Remote credit manager block.

Dưới đây là dạng sóng của khối:



Hình 2.8: Dạng sóng của khối kiểm tra credit

2.2.5. Khối kiểm tra trạng thái



Hình 2.9: Khối kiểm tra trạng thái

Khối này được sử dụng để kiểm tra trạng thái của gói tin trả về. Có 4 trạng thái: gói tin trả về thành công, gói tin trả về bị loại bỏ, yêu cầu không được hỗ trợ và dừng lại.

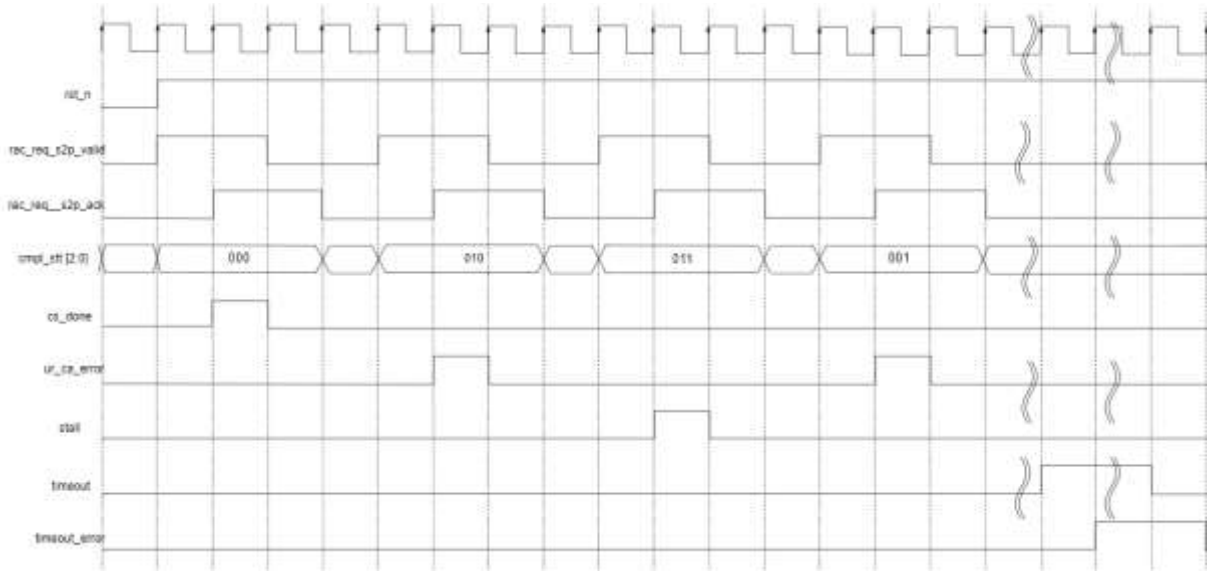
- Nếu trạng thái là 000, khối Check status sẽ gửi tín hiệu cs_done.
- Nếu trạng thái là 010 hoặc 001, khối Check status sẽ gửi tín hiệu ur_ca_error và ur_ca_signal.
- Nếu trạng thái là 011, khối Check status sẽ gửi tín hiệu stall.
- Nếu gói tín trả về không phản hồi kịp thời (sau 8ms), khối Data processor sẽ nhận tín hiệu hết thời gian từ bộ đếm thời gian trong miền xung nhịp khác. Khi nhận được tín hiệu timeout_syn, khối này sẽ gửi tín hiệu timeout_error đến FSM và tín hiệu timeout_signal đến khối Data processor.

Bảng dưới đây mô tả chi tiết các thông số của khối:

Bảng 2.4: Mô tả chi tiết thông số của khối kiểm tra trạng thái

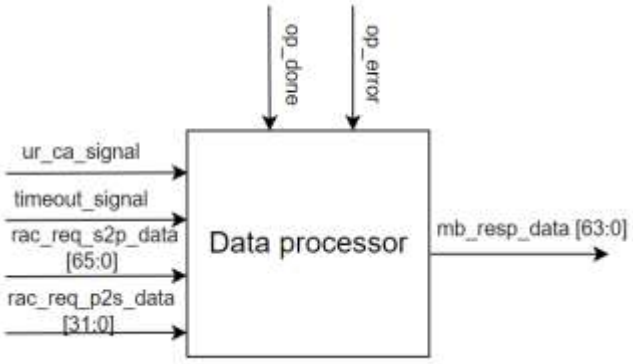
Pin name	Direction	Clock	Size	Description
Check status Ports				
clk	Input	_	1	2GHz clock.
rst_n	Input	Async	1	Asynchronous reset, active low.
init	Input	clk	1	Sync init, active high.
timeout	Input	clk	1	Timeout signal from timer.
cmpl_stt	Input	clk	3	Status bit of completion.
rac_req_s2p_valid	Input	clk	1	Valid signal of completion from parallel interface.
send_req	Input	clk	1	Send request signal send to operate check_credit block.
mb_resp_valid_n	Input	clk	1	Valid signal respond to mailbox.
stall	Output	clk	1	Reset timer signal send to FSM.
ur_ca_error	Output	clk	1	Error signal send to FSM when status of completion is error.
cs_done	Output	clk	1	Done signal send to FSM when status of completion is done.
timeout_signal	Output	clk	1	Time out error signal send to data_processor block to process data.
ur_ca_signal	Output	clk	1	Ur_ca error signal send to data_processor block to process data.
rac_req_s2p_ack	Output	clk	1	Acknowledge signal of completion request send back to parallel interface.

Dưới đây là dạng sóng của khối:



Hình 2.10: Dạng sóng của khối kiểm tra trạng thái

2.2.6. Khối xử lý dữ liệu



Hình 2.11: Khối xử lý dữ liệu của Remote access requester

Khối này được sử dụng để xử lý dữ liệu phản hồi đến mailbox.

- Nếu nó nhận được tín hiệu op_done, điều đó có nghĩa là gói tin trả về thành công sau đó chuyển gói tin trả về đến mailbox.
- Nếu nó nhận được op_error, điều đó có nghĩa là gói tin trả về là lỗi, sau đó nó sẽ tạo gói tin trả về với dữ liệu được yêu cầu và chuyển đến mailbox.

Gói tin trả về lỗi có 2 loại:

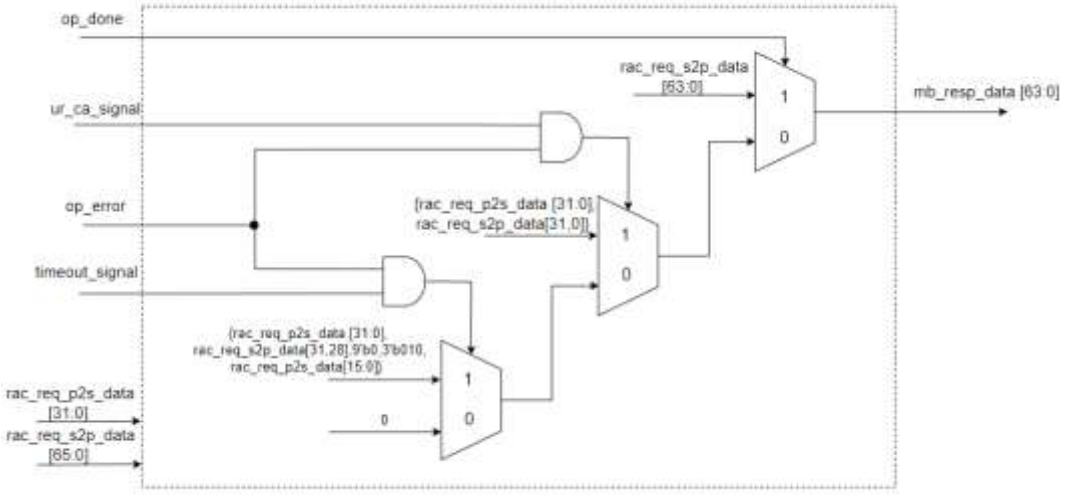
- Loại thứ nhất là do gói tin trả về có trạng thái bị lỗi là UR hoặc CA.
- Loại thứ 2 là do không nhận được gói tin trả về, nhận được tín hiệu timeout và hệ thống tự tạo gói tin lỗi để gửi về mailbox.

Bảng dưới đây mô tả chi tiết các thông số của khối:

Bảng 2.5: Mô tả chi tiết thông số khối xử lý dữ liệu của Remote access requester

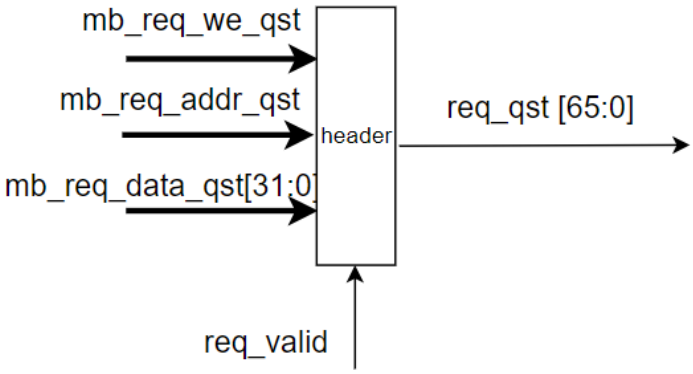
Pin name	Direction	Clock	Size	Description
Data processor Ports				
timeout_signal	Input	clk	1	Timeout error signal from check status block.
ur_ca_signal	Input	clk	1	Ur_ca error signal from check status block.
rac_req_s2p_data	Input	clk	66	Data bus signals of completion from parallel interface.
rac_req_p2s_data	Input	clk	32	Remote access request send to parallel interface.
op_done	Input	clk	1	Done signal send from FSM to process done data.
op_error	Input	clk	1	Error signal send from FSM to process error data.
mb_resp_data	Output	clk	64	Data bus response from the Remote access requester.

Dưới đây là mô tả về nguyên lý hoạt động của khối



Bảng 2.6: Nguyên lý hoạt động của khối xử lý dữ liệu

2.2.7. Khối header



Hình 2.12: Khối header

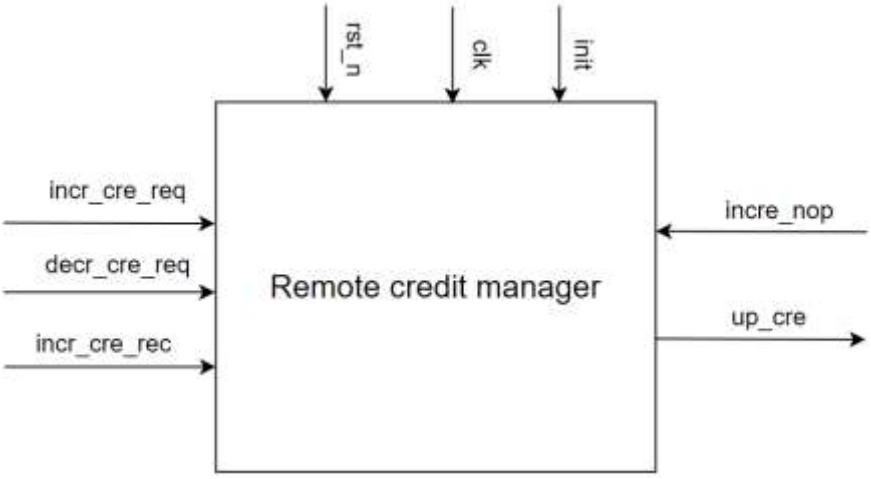
Khối header dùng để tạo gói tin yêu cầu từ tín hiệu và dữ liệu được gửi từ mailbox. Bảng dưới đây mô tả chi tiết các thông số của khối:

Bảng 2.7: Mô tả chi tiết thông số của khối header

Pin name	Direction	Clock	Size	Description
Header Ports				
mb_req_we_qst	Input	async	1	Write operation.
mb_req_addr_qst	Input	async	1	Address signal of register.
mb_req_data_qst	Input	async	32	Write data.
req_qst	Output	clk	66	Packet request from mailbox.
req_valid	Input	clk	1	Valid request signal to generate request.

2.3. Khối Remote credit manager

Adapter có số lượng credit là 4 khi khởi tạo, nó cũng là số credit tối đa của Adapter. Khi gửi một yêu cầu, sẽ có một credit bị tiêu thụ. Trường "cr" trong gói tin phản hồi sẽ cộng lại một credit cho khối này.



Hình 2.13: Khối Remote credit manager

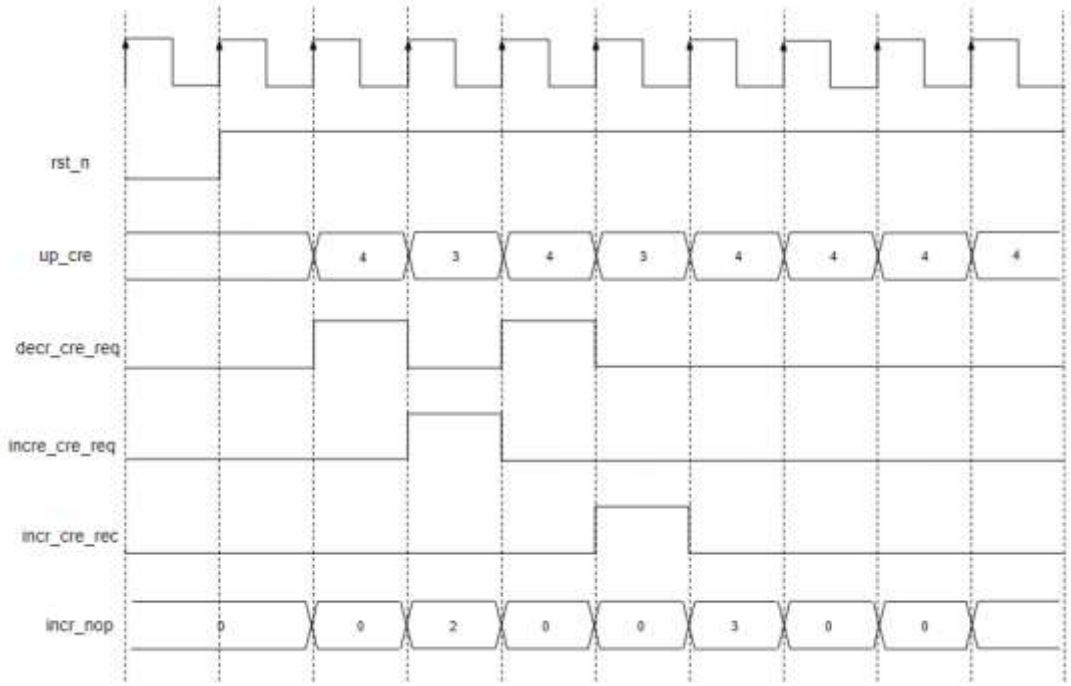
Dưới đây là bảng mô tả chi tiết các chân của khối:

Bảng 2.8: Mô tả chi tiết thông số của khối Remote credit manager

Pin name	Direction	Clock	Size	Description
Remote credit manager Ports				
incr_cre_req	Input	clk	1	Increase credit signal from Remote access requester block.
decr_cre_req	Input	clk	1	Decrease credit signal from Remote access requester block.
incr_cre_rec	Input	clk	1	Increase credit signal from Remote access receiver block.
clk	Input	_	1	2GHz clock.
rst_n	Input	Async	1	Asynchronous reset, active low.
incr_nop	Input	clk	3	Number of credit send from PHY to Remote credit manager block.
up_cre	Output	clk	3	Number of credit in Remote credit manager.

Khối remote credit manager quản lý credit trong gói tin, khối này thực thi khi nhận được tín hiệu incr_cre_req, decr_cre_req từ khối remote access requester hoặc incr_cre_rec từ khối remote access receiver. Khi nhận được các tín hiệu này, nó sẽ cập nhật giá trị trong biến up_cre.

Ngoài ra nó còn nhận tín hiệu tăng credit incr_nop từ phía message, incr_nop này có giá trị tối đa là 4. Khi nhận đồng thời nhiều tín hiệu một lúc, giá trị của biến up_cre có thể vượt quá giá trị tối đa của nó là 4. Đối với trường hợp này, ta sẽ đặt biến up_cre bằng 4. Việc giá trị tối đa của biến up_cre là 4 là vì trong trường hợp truyền gói tin từ adapter này đến adapter kia, có lúc bên phía adapter kia không xử lý kịp và không trả credit về. Nếu ta truyền tiếp sẽ mất luôn gói tin. Vì vậy sẽ có một lượng credit giới hạn cho mỗi khối remote credit manager.

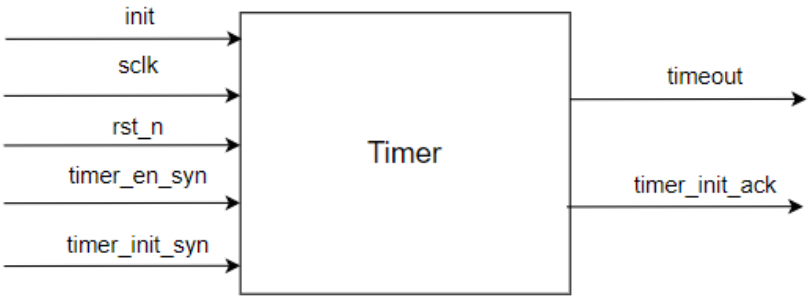


Hình 2.14: Dạng sóng của khối Remote credit manager

2.4. Khối Timer và khối CDC

2.4.1. Khối Timer

Chúng ta cần một bộ đếm để đếm 8ms cho thiết kế này. Tuy nhiên, nếu đếm trong miền đồng hồ clk (2GHz) sẽ cần rất nhiều chu kỳ đồng hồ để đếm đủ 8ms. Vì vậy, chúng ta cần sử dụng bộ đếm này trong miền đồng hồ sclk (20MHz) để đếm 8ms. Khối này sẽ thực thi khi nhận tín hiệu en_timer từ FSM trong Remote access register. Khi đếm đủ 8ms, điều này có nghĩa là đã không phản hồi gói tin đúng thời gian quy định.



Hình 2.15: Khối Timer

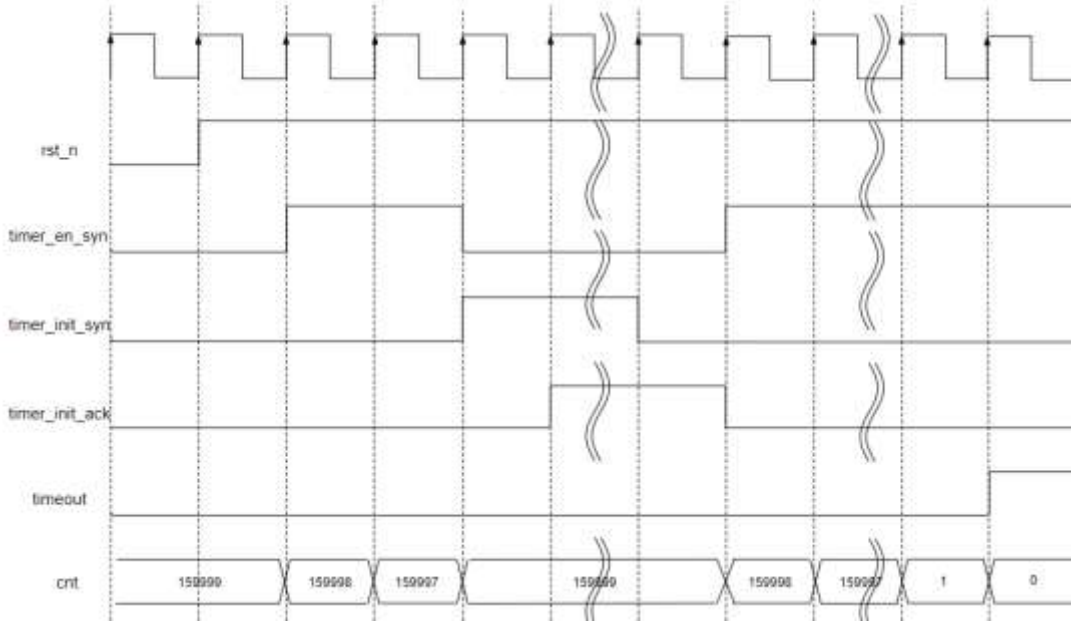
Dưới đây là bảng mô tả chi tiết các chân của khối:

Bảng 2.9: Mô tả chi tiết thông số của khối Timer

Pin name	Direction	Clock	Size	Description
Timer Ports				
timer_en_syn	Input	sclk	1	Enable timer signal from CDC block.
timer_nit_syn	Input	sclk	1	Init timer signal from CDC block.
sclk	Input	_	1	20MHz clock.

rst_n	Input	async	1	Asynchronous reset, active low.
init	Input	clk	1	Sync init, active high.
timeout	Output	sclk	1	Done signal when timer counter is max.
timer_init_ack	Output	sclk	1	Init ack signal respond to FSM.

Dưới đây là dạng sóng của khối:

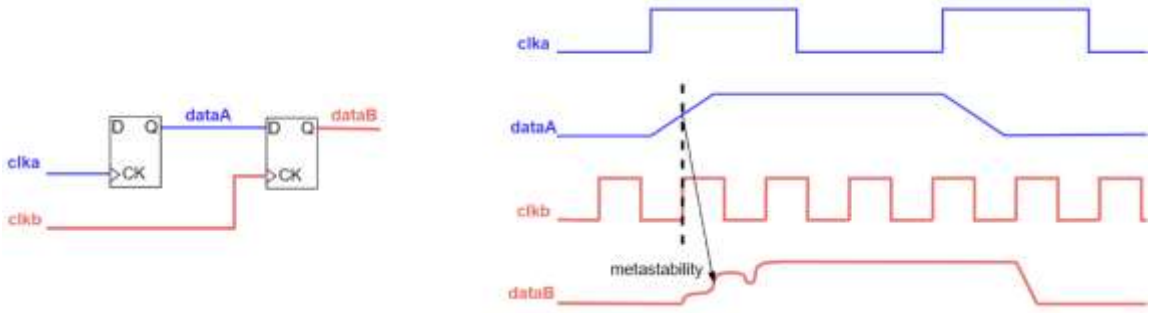


Hình 2.16: Dạng sóng của timer

2.4.2. Khối CDC

Trong thiết kế, tín hiệu từ hộp thư (mailbox) đến Remote access register hoặc từ Remote access register đến bộ đếm timer có các miền clock khác nhau. Điều này dẫn đến tồn tại trạng thái bất ổn định vì xuất hiện sự vi phạm định thời (timing) giữa các tín hiệu đến và tín hiệu xung clock gây ra hiện tượng dao động.

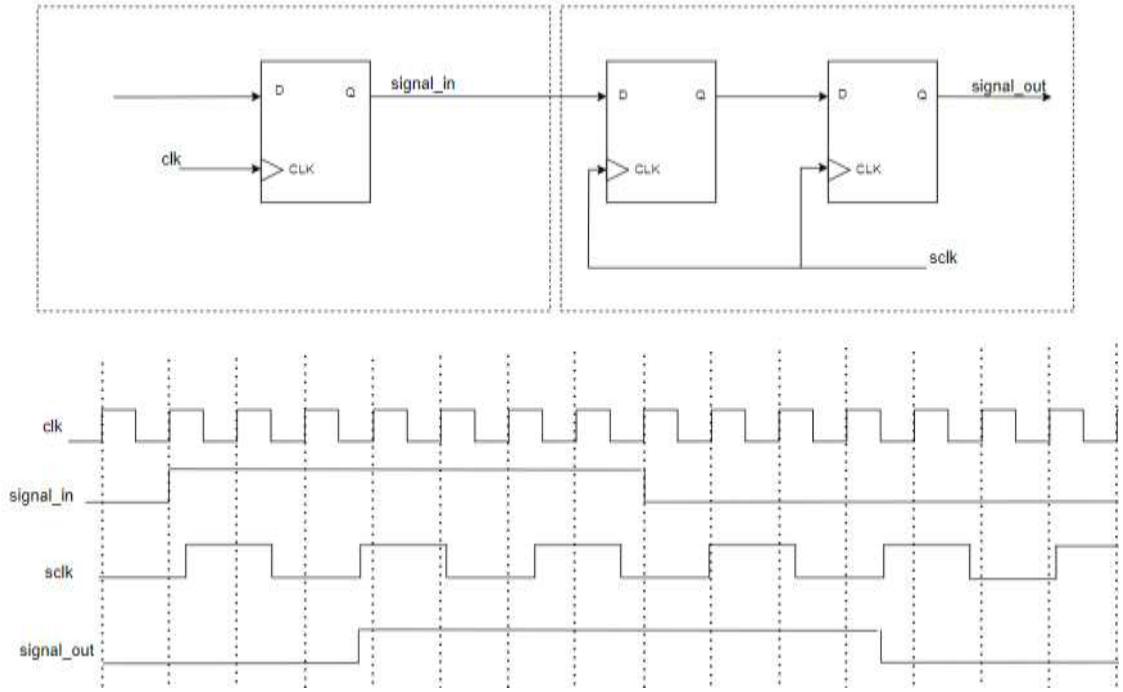
Trạng thái bất ổn định đề cập đến các tín hiệu không xác định các trạng thái 0 hoặc 1 ổn định trong một thời gian tại một số điểm trong quá trình hoạt động bình thường của một thiết kế. Trong thiết kế có nhiều clock, trạng thái bất ổn định không thể tránh nhưng những tác động bất lợi của trạng thái bất ổn định có thể được trung hòa [5].



Hình 2.17: Minh họa hiện tượng metastability

Giải pháp cho vấn đề này là sử dụng phương pháp mở – lấy mẫu các tín hiệu bằng bộ đồng bộ (synchronizer) để đồng bộ hóa các tín hiệu điều khiển giữa sclk và clk. Tức là sử dụng 2 flip flop.

Hình vẽ dưới đây mô tả quá trình đồng bộ 2 flip flop:



Hình 2.18: Dạng sóng của phương pháp đồng bộ 2 flip flop

Một flip-flop đầu tiên lấy mẫu tín hiệu bất đồng bộ vào miền clock mới và chờ một chu kỳ clock đầy đủ để cho phép bất ổn định của tín hiệu đầu ra của giai đoạn 1 giảm đi, sau đó tín hiệu của giai đoạn 1 được lấy mẫu bởi cùng một đồng hồ vào flip-flop thứ hai, với mục tiêu là tín hiệu của giai đoạn 2 hiện đã ổn định và hợp lệ, đã được đồng bộ hóa và sẵn sàng phân phối trong miền clock mới [5].

Dưới đây là bảng mô tả chi tiết các chân của khối:

Bảng 2.10: Mô tả chi tiết thông số khối CDC

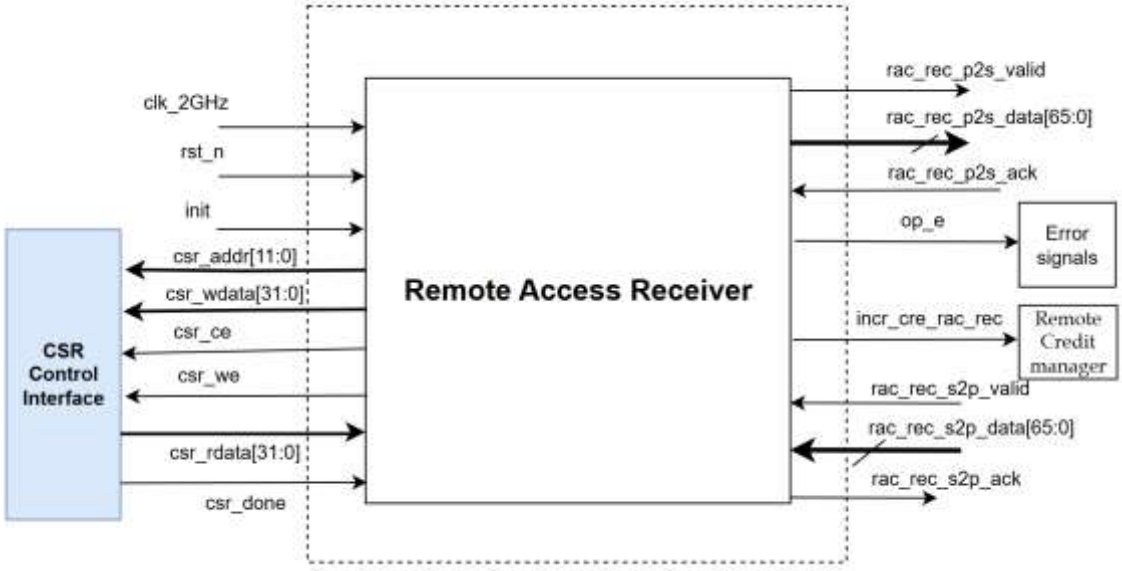
Pin name	Direction	Clock	Size	Description
CDC Ports				
timer_en	Input	clk	1	Enable timer signal from Remote access requester block.
timer_init	Input	clk	1	Init timer signal from Remote access requester block.
timeout	Input	sclk	1	Time out signal from timer.
mb_req_ valid_qst	Input	asyn	1	Valid signal from mailbox.
init	Input	clk	1	Sync init, active high.
sclk	Input	_	1	20MHz clock.
clk	Input	_	1	2GHz clock.

timer_en_syn	Output	sclk	1	Enable timer signal from Timer block.
timer_init_syn	Output	sclk	1	Init timer signal from Timer block.
timeout_syn	Output	clk	1	Timeout signal after synchronous.
req_valid	Output	clk	1	Valid signal from mailbox after synchronous.

2.5. Khối Remote access receiver

2.5.1. Tổng quan

Khối Remote Access Receiver sẽ nhận gói tin từ Bộ định tuyến gói tin (Adapter to Adapter), nó nhận gói tin yêu cầu gồm 66 bit chứa 12 bit phạm vi địa chỉ với 3 trường hợp vị trí: Adapter, Adapter & PHY, PHY hoặc các trường hợp khác (lỗi). Sau đó, khi Adapter nhận yêu cầu, nó xử lý lệnh và trả về gói tin trả về Register. Khối này cũng kiểm tra tính hợp lệ của gói tin, nếu không hợp lệ, gói tin sẽ bị bỏ qua và chờ gói tiếp theo.



Hình 2.19: Sơ đồ tổng quan Remote Access Receiver

2.5.2. Thông số và sơ đồ chi tiết của khối

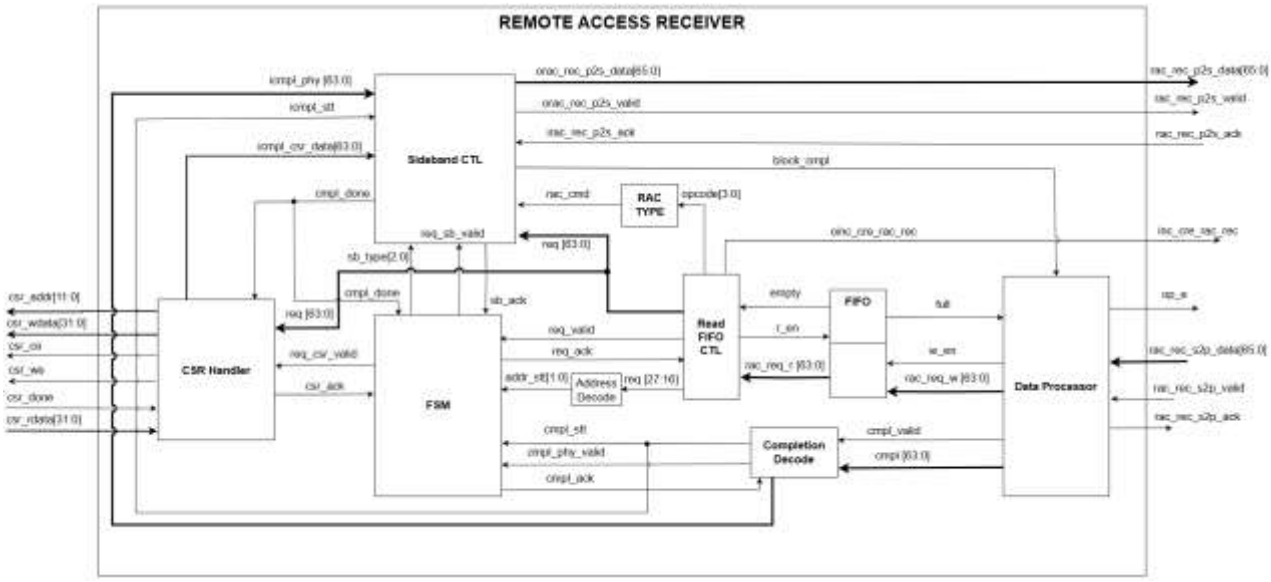
Bảng dưới đây mô tả đầy đủ các thông số của khối:

Bảng 2.11: Thông số của các giao diện remote access receiver

Pin name	Direction	Clock	Size	Description
CTL control ports				
clk	Input	-	1	Clock.
rst_n	Input	Async	1	Asynchronous reset, active low.
init	Input	clk	1	Sync init, active high.
Remote Access Receiver Ports				
rac_rec_s2p_valid	Input	clk	1	Inform the data from Packet Router to Remote Access Receiver is valid.

rac_rec_p2s_valid	Output	clk	1	Inform the data from Remote Access Receiver to Packet Router is valid.
rac_rec_s2p_data [65:0]	Input	clk	66	Data to the Remote Access Receiver.
rac_rec_p2s_data [65:0]	Output	clk	66	Data from the Remote Access Receiver.
rac_rec_s2p_ack	Input	clk	1	The acknowledgment signal confirms that data has been successfully received, and the system is ready to receive the next data.
rac_rec_p2s_ack	Input	clk	1	The acknowledgment signal confirms that data has been successfully received, and the system is ready to receive the next data.
oincr_cre_rac_rec	Output	clk	1	Credit from PHY to the Adapter
op_e	Output	clk	1	Operation error, check opcode bits. Only clear with sync init.
csr_addr[11:0]	Output	clk	12	Address to access CSR.
csr_wdata[31:0]	Output	clk	32	Write data.
csr_rdata[31:0]	Input	clk	32	Read data.
csr_ce	Output	clk	1	Enable to access CSR.
csr_we	Output	clk	1	Write enable, valid when csr_ce=1 • 0 Read operation • 1 Write operation
csr_done	Input	clk	1	Operation done.

Dưới đây là sơ đồ chi tiết của khối:



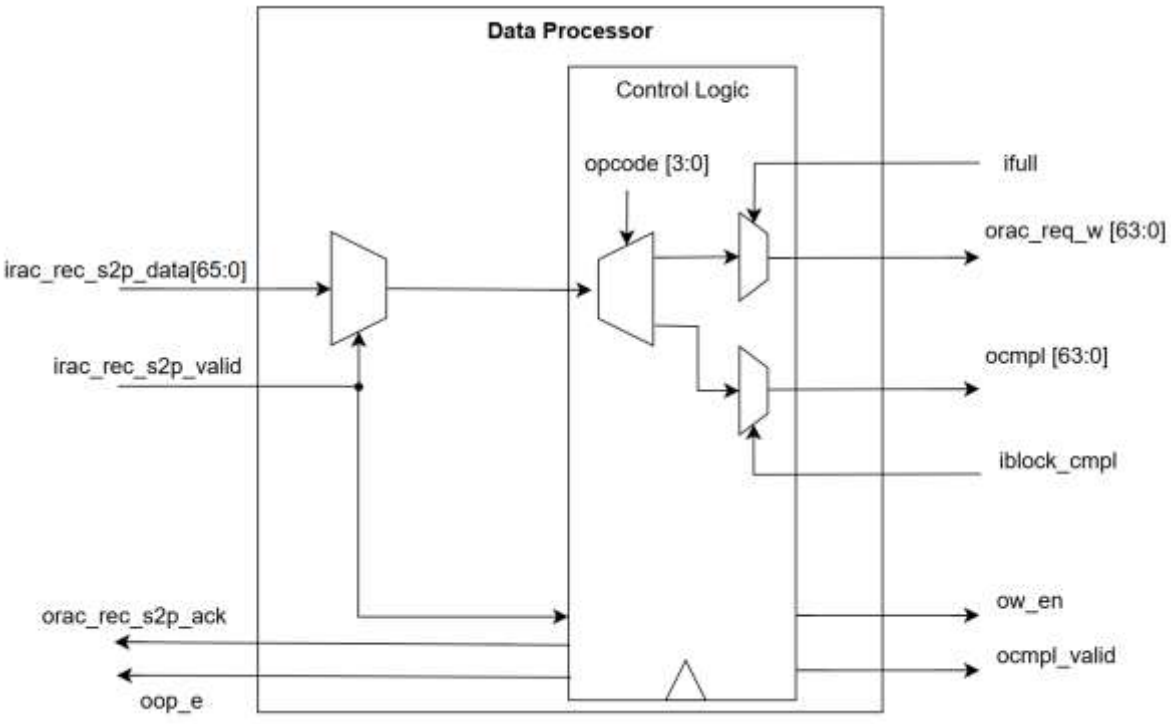
Hình 2.20: Mô tả chi tiết khối Remote access receiver

Từ vấn đề, chúng ta sẽ cần các khối sau:

- Khối Xử lý Dữ liệu (Data Processor) sẽ giải mã opcode từ giao diện song song và trả về giá trị opcode, xác định xem đó là yêu cầu hay hoàn tất. Nếu opcode không hợp lệ, gói tin sẽ bị bỏ qua và tín hiệu lỗi sẽ được xuất ra.
- Khối FIFO sẽ nhận dữ liệu yêu cầu từ khối Xử lý Dữ liệu để xử lý từng yêu cầu một.
- Khối Điều khiển FIFO đọc (Read FIFO CTL) sẽ kiểm soát việc đọc dữ liệu từ FIFO và kiểm tra tín dụng trong gói tin.
- Khối Giải mã Hoàn tất (Completion Decode) sẽ kiểm tra trạng thái trong gói hoàn tất.
- Khối Giải mã Địa chỉ (Address Decode) sẽ giải mã và cung cấp trạng thái địa chỉ.
- Khối RAC TYPE sẽ xử lý opcode và tạo ra lệnh tương ứng.
- FSM sẽ kiểm soát trạng thái của thiết kế với ba trạng thái: Adapter, Adapter & PHY, và Lỗi, cùng với trạng thái IDLE khi hệ thống không hoạt động.
- Khối Xử lý CSR (CSR Handler) sẽ trao đổi thông tin với giao diện Điều khiển CSR.
- Khối Điều khiển Sideband (SB_CTL) sẽ xử lý các gói tin trả về hoặc gói tin yêu cầu (đến PHY cục bộ) trước khi xuất ra dựa trên sb_type.

2.5.3. Khối xử lý dữ liệu

Khối xử lý dữ liệu chịu trách nhiệm phân loại các gói tin thành hai luồng: yêu cầu và hoàn tất.



Hình 2.21: Khối xử lý dữ liệu của remote access receiver

- Nếu gói tin có opcode thuộc loại yêu cầu và FIFO chưa đầy, gói tin sẽ được ghi vào FIFO.
- Nếu FIFO đầy, hệ thống sẽ báo lỗi nếu gói tin nhận được vẫn là gói yêu cầu. Đối với các gói hoàn tất, sẽ kiểm tra với tín hiệu block_cmpl (ngăn chặn việc nhận hai gói hoàn tất liên tiếp).
- Nếu block_cmpl bằng 0, gói hoàn tất được coi là hợp lệ. Ngược lại, hệ thống sẽ báo lỗi.
- Nếu opcode không thuộc hai loại này, hệ thống cũng sẽ báo lỗi về opcode.

Bảng dưới đây mô tả đầy đủ các thông số của khối:

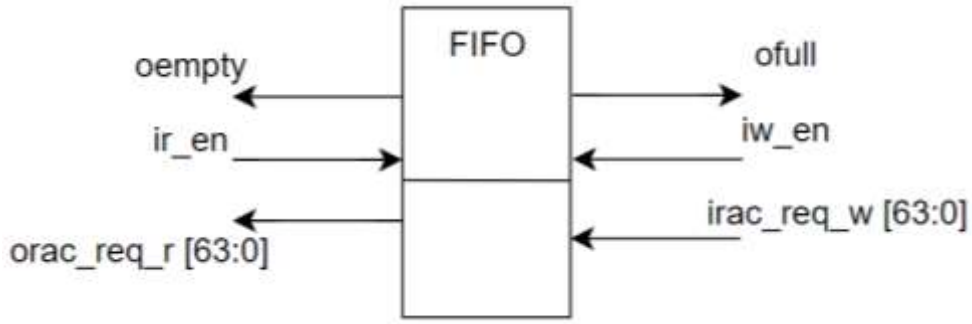
Bảng 2.12: Mô tả chi tiết thông số khối xử lý dữ liệu của Remote access requester

Pin Name	Direction	Clock	Size	Description
CTL control Ports				
clk	Input	-	1	Clock
rst_n	Input	Async	1	Asynchronous reset, active low.
init	Input	clk	1	Sync init, active high.
Data Processor Ports				
irac_rec_s2p_valid	Input	clk	1	Inform the data from Packet Router to Remote Access Receiver is valid.
irac_rec_s2p_data[65:0]	Input	clk	66	Data to the Remote Access Receiver.
orac_rec_s2p_ack	Output	clk	1	The acknowledgment signal confirms that data has been successfully received and the system is ready to receive the next data.
oop_e	Output	clk	1	Operation error, check opcode bits. Only clear with sync init.
ifull	Input	clk	1	Signal full of FIFO.
ow_en	Output	clk	1	Write enable.
orac_req_w [63:0]	Output	clk	64	Write request data to FIFO.
ocmpl [63:0]	Output	clk	64	Completion data.
ocmpl_valid	Output	clk	1	Inform the completion data is valid.
iblock_cmpl	Input	clk	1	Block case completion packet be error.

2.5.4. Khối FIFO của Remote access receiver

Trong thiết kế, chỉ cho phép tối đa 2 gói 2 Register access request cùng lúc, nghĩa là Adapter phải chờ nhận gói hoàn tất từ die khác trước khi gửi gói mới. Để quản lý điều

này, một FIFO có độ sâu 2 là cần thiết để lưu trữ các gói yêu cầu, nhằm đảm bảo hệ thống hoạt động hiệu quả và tránh tràn bộ đệm khi xử lý các yêu cầu đồng thời.



Hình 2.22: Khối FIFO của remote access receiver

FIFO được ghi bằng các gói tin đầu vào 64-bit từ mô-đun Xử lý Dữ liệu và đọc ra các gói tin 64-bit đầu ra tới khối điều khiển FIFO đọc. FIFO giúp giữ lại các gói tin đã gửi để xử lý sau này trong trường hợp bộ Remote access receiver chưa hoàn thành xử lý gói tin trước đó. Dữ liệu được ghi vào FIFO ở mỗi chu kỳ đồng hồ khi tín hiệu kích hoạt ghi (write enable) hoạt động (Arbiter có gói để gửi) và FIFO không đầy, con trỏ ghi sẽ tăng sau mỗi lần ghi thành công. Dữ liệu có thể đọc ra từ FIFO ở mỗi chu kỳ đồng hồ khi tín hiệu kích hoạt đọc (read enable) hoạt động và FIFO không trống, con trỏ đọc sẽ tăng sau mỗi lần đọc thành công.

Bảng 2.13: Mô tả chi tiết thông số khối FIFO của Remote access receiver

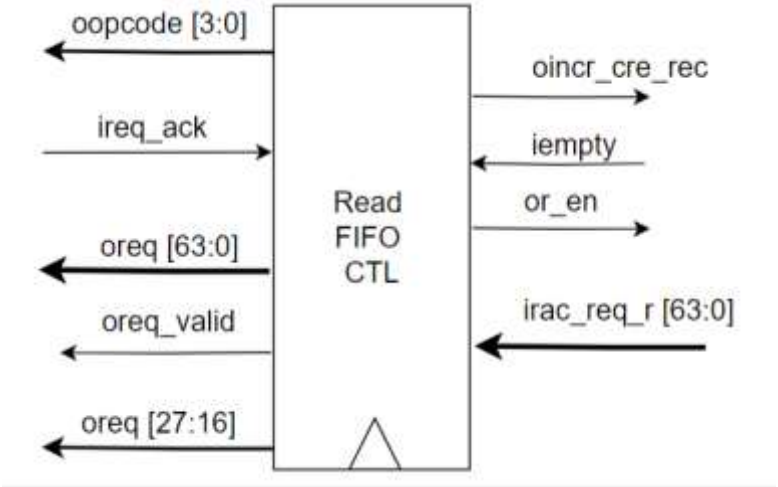
Pin Name	Direction	Clock	Size	Description
CTL control Ports				
clk	Input	-	1	Clock.
rst_n	Input	Async	1	Asynchronous reset, active low.
init	Input	clk	1	Sync init, active high.
FIFO rec Ports				
ifull	Output	clk	1	Signal full of FIFO.
iw_en	Input	clk	1	Write enable.
irac_req_w [63:0]	Input	clk	64	Write request data to FIFO.
ir_en	Input	clk	1	Read enable.
oempty	Output	clk	1	Signal empty of FIFO.
orac_req_r [63:0]	Output	clk	64	Read request data from FIFO.

2.5.5. Khối Read FIFO CTL

Khối điều khiển FIFO đọc (Read FIFO CTL) quản lý việc lấy các gói tin 64-bit từ FIFO, gửi chúng đến bộ xử lý CSR và các block Sideband CTL để xử lý, đồng thời truyền các tín hiệu yêu cầu hợp lệ (request valid) tới FSM. Nó kiểm soát việc đọc trong hai trường hợp:

Đầu tiên, khi các block tiếp theo xác nhận đã hoàn thành xử lý và FIFO không trống.

Thứ hai, khi các block đó đang trong trạng thái IDLE và có gói tin mới sẵn sàng trong FIFO. Cùng với mỗi gói tin được truyền đi, tín hiệu hợp lệ (valid) cũng được kích hoạt để thông báo dữ liệu truyền đi là hợp lệ, đảm bảo quá trình truyền dữ liệu diễn ra chính xác và hiệu quả trong hệ thống.



Hình 2.23: Khối điều khiển FIFO đọc

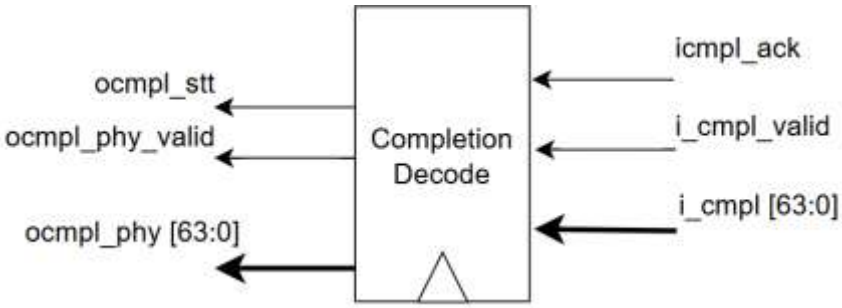
Bảng dưới đây mô tả đầy đủ các thông số của khối:

Bảng 2.14: Mô tả chi tiết thông số khối xử lý dữ liệu của Remote access receiver

Pin Name	Direction	Clock	Size	Description
CTL control Ports				
clk	Input	-	1	Clock.
rst_n	Input	Async	1	Asynchronous reset, active low.
init	Input	clk	1	Sync init, active high.
Read FIFO CTL Ports				
or_en	Output	clk	1	Read enable.
iempty	Input	clk	1	Signal empty of FIFO.
irac_req_r	Input	clk	64	Read request data from FIFO.
oincr_cre_rec	Output	clk	1	Increase credit.
oreq_valid	Output	clk	1	Inform the request data is valid.
oreq[63:0]	Output	clk	64	Request Data.
oreq[27:16]	Output	clk	12	
oopcode[3:0]	Output	clk	4	
ireq_ack	Input	clk	1	The acknowledgment signal confirms that data has been successfully received, and the system is ready to send the next data.

2.5.6. Khối giải mã hoàn tất

Khối giải mã hoàn tất (Completion Decode) sẽ kiểm tra trạng thái trong gói tin trả về, các trường hợp lỗi đã được giải thích trong trường trạng thái.



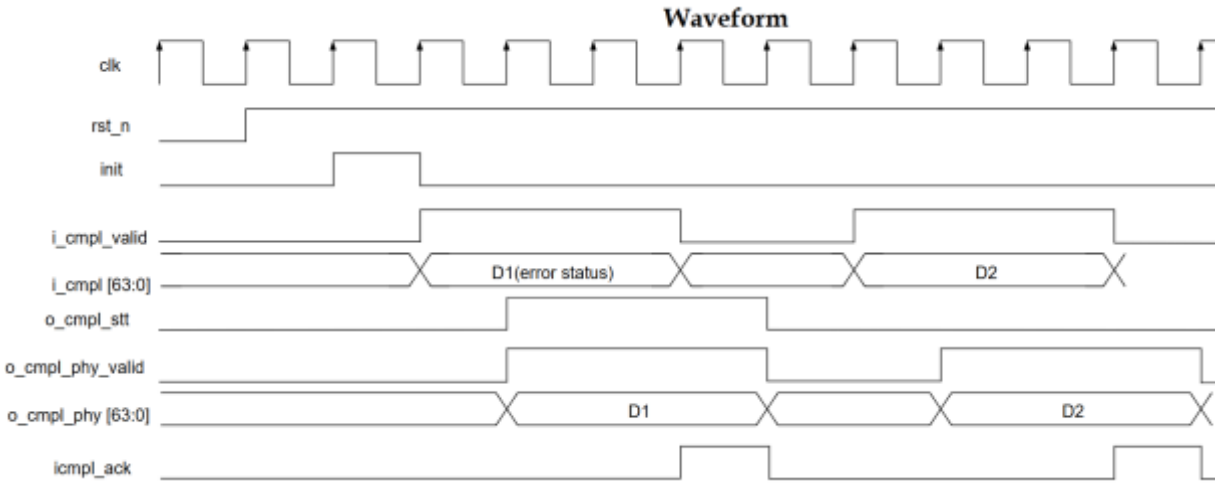
Hình 2.24: Khối giải mã hoàn tất

Bảng dưới đây mô tả đầy đủ các thông số của khối:

Bảng 2.15: Mô tả chi tiết thông số khối giải mã hoàn tất của Remote access receiver

Pin Name	Direction	Clock	Size	Description
Completion Decode Ports				
ocmpl_phy_valid	Output	clk	1	Inform the data from PHY is valid.
ocmpl_phy [63:0]	Output	clk	64	Completion data from PHY.
icmpl_ack	Input	clk	1	The acknowledgment signal confirms that data has been successfully received, and the system is ready to receive the next data.
icmpl [63:0]	Output	clk	64	Completion data from PHY.
ocmpl_stt	Output	clk	1	Status of completion data with 1 is true and 0 is error.
icmpl_valid	Input	clk	1	Inform the completion data is valid.

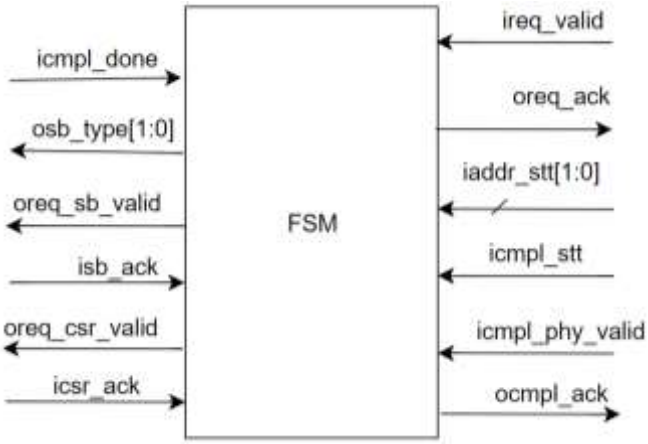
Dưới đây là dạng sóng của khối:



Hình 2.25: Dạng sóng của khối giải mã hoàn tất

2.5.7. Khối FSM

FSM sẽ điều khiển các trạng thái của khối với các trạng thái: Adapter, Adapter & PHY, Error.



Hình 2.26: Khối FSM của Remote access receiver

Bảng dưới đây mô tả đầy đủ các thông số của khối:

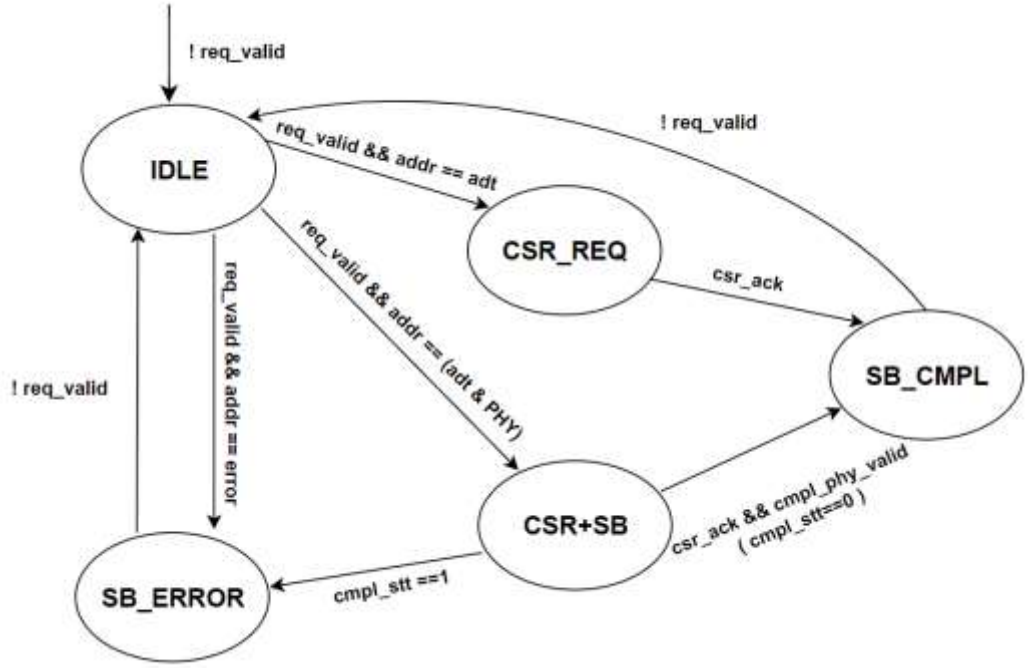
Bảng 2.16: Mô tả chi tiết thông số khối FSM của Remote access receiver

Pin Name	Direction	Clock	Size	Description
FSM Ports				
ireq_valid	Input	clk	1	Inform the request data is valid.
iaddr_stt[1:0]	Input	clk	2	Status of address.
oreq_ack	Output	clk	1	The acknowledgment signal confirms that data has been successfully received, and the system is ready to receive the next data.
icmpl_phy_valid	Input	clk	1	Inform the data from PHY is valid.
icmpl_stt	Input	clk	1	Status of completion data with 1 is true and 0 is error.
ocmpl_ack	Output	clk	1	The acknowledgment signal confirms that data has been successfully received, and the system is ready to receive the next data.
oreq_csr_valid	Output	clk	1	Inform the request data to CSR is valid
icsr_ack	Input	clk	1	The signal confirms that request CSR has been completed.
oreq_sb_valid	Output	clk	1	Inform the request data to Sideband CTL is valid.
osb_type[1:0]	Output	clk	2	See below.
isb_ack	Input	clk	1	The signal confirms that request PHY packet has been transmitted.
icmpl_done	Input	clk	1	The signal confirms that completion packet has been transmitted.

osb_type[1:0] có 4 loại

- Req_phy (2'b00)
- cmpl_csr (2'b01)
- cmpl_mix (2'b11)
- cmpl_error (2'b10**)**

Sơ đồ dưới đây mô tả các trạng thái của FSM:



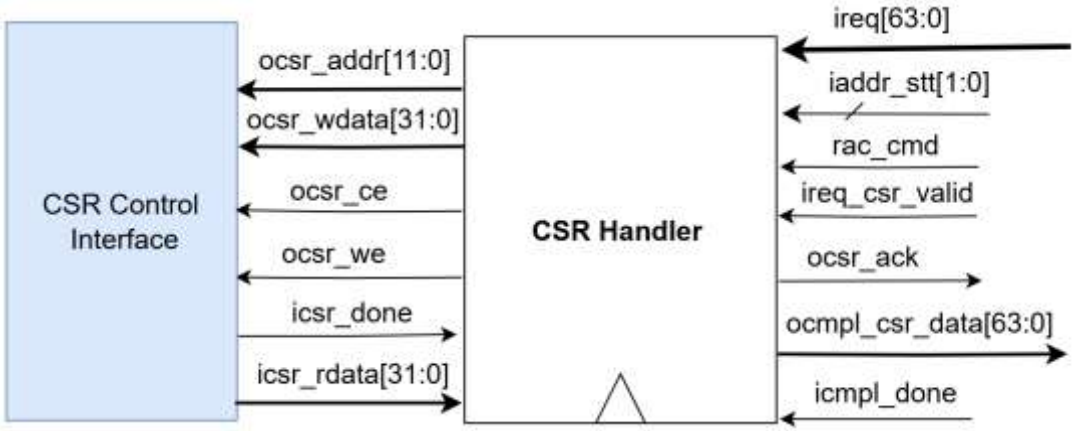
Hình 2.27: Sơ đồ trạng thái FSM của Remote access receiver

FSM hoạt động qua các trạng thái sau:

- IDLE: Trạng thái mặc định của mô-đun này.
- CSR_REQ: Trong trạng thái này, khối FSM sẽ gửi req_csr_valid đến Bộ xử lý CSR và chờ csr_ack để truyền trạng thái.
- CSR_SB: Trong trạng thái này, khối FSM sẽ gửi req_csr_valid đến Bộ xử lý CSR và gửi req_sb_valid cùng với sb_type[1:0] = 2'b00 (req_PHY) đến khối Điều khiển Sideband.
- SB_ERROR: Trong trạng thái này, khối FSM sẽ gửi req_sb_valid và sb_type[1:0] = 2'b11 (lỗi) đến khối Điều khiển Sideband.
- SB_CMPL: Trong trạng thái này, khối FSM sẽ gửi req_sb_valid và sb_type[1:0] = 2'b01 (cmpl_csr) đến khối Điều khiển Sideband để tạo gói hoàn tất với dữ liệu từ CSR hoặc gửi req_sb_valid và sb_type[1:0] = 2'b11 (cmpl_mix) đến khối Điều khiển Sideband để tạo gói tin trả về với dữ liệu, trong đó byte thấp (LSB) nằm trong Adapter, byte cao (MSB) nằm trong PHY.

2.5.8. Khối CSR Handler

Khối CSR Handler nhận yêu cầu hợp lệ (req_csr_valid) từ FSM và sau đó tách gói yêu cầu (64 bit) để phù hợp với giao diện điều khiển CSR. Khi nhận được tín hiệu csr_done và dữ liệu (đối với yêu cầu dữ liệu đọc), CSR Handler sẽ lấy dữ liệu và gửi đến khối Điều khiển Sideband, đồng thời gửi tín hiệu csr_ack trở lại FSM. Sau khi khối Điều khiển Sideband hoàn tất xử lý, nó sẽ gửi tín hiệu cmpl_done để báo hiệu quá trình đã hoàn tất và sẽ tắt tín hiệu csr_ack.



Hình 2.28: Khối CSR Handler

Bảng dưới đây mô tả đầy đủ các thông số của khối:

Bảng 2.17: Mô tả chi tiết thông số khối CSR Handler

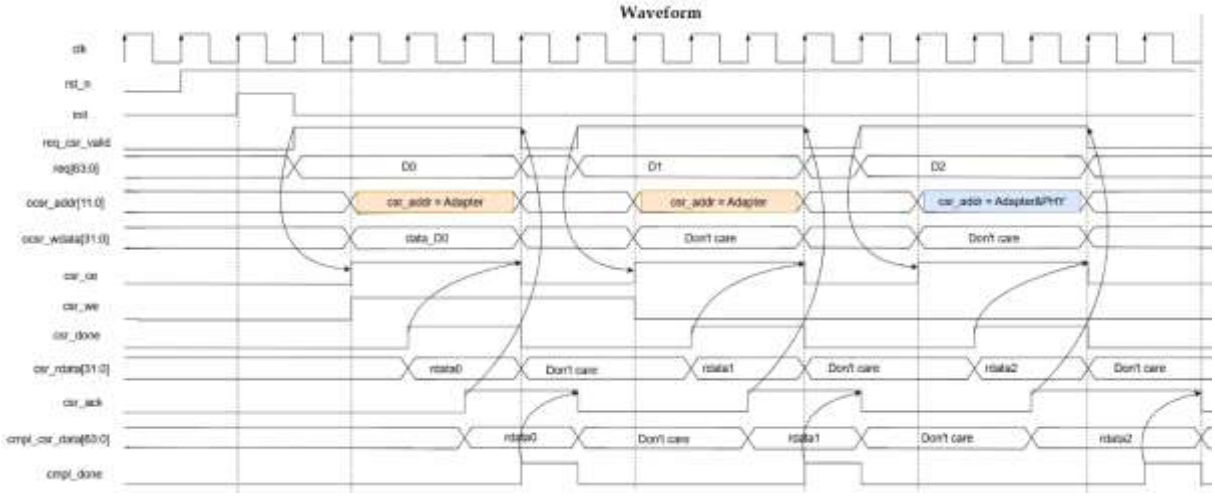
Pin Name	Direction	Clock	Size	Description
CTL control Ports				
clk	Input	-	1	Clock.
rst_n	Input	Async	1	Asynchronous reset, active low.
init	Input	clk	1	Sync init, active high.
CSR Handler Ports				
ireq_csr_valid	Input	clk	1	Inform the request data to CSR is valid.
ireq[63:0]	Input	clk	64	Request data.
icsr_ack	Input	clk	1	The signal confirms that request CSR has been completed.
ocsr_addr[11:0]	Output	clk	12	Address to access CSR.
ocsr_wdata[31:0]	Output	clk	32	Write data.
icsr_rdata[31:0]	Input	clk	32	Read data.
ocsr_ce	Output	clk	1	Enable to access CSR.
ocsr_we	Output	clk	1	Write enable, valid when csr_ce=1 • 0 Read operation • 1 Write operation

ocmpl_csr_data[63:0]	Output	clk	64	Completion data with data from CSR Control.
icmpl_done	Input	clk	1	The signal confirms that completion packet has been transmitted.

csr_we: Write enable, bật lên khi csr_ce = 1

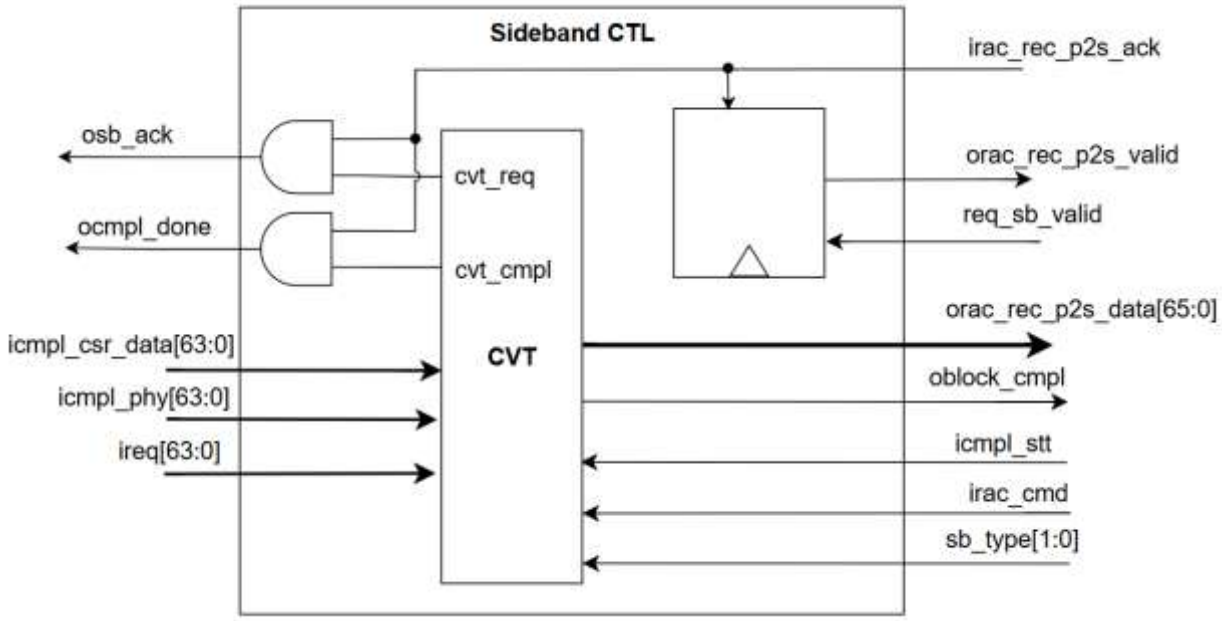
- 0 : Hoạt động đọc khi opcode = Memory read
- 1 :Hoạt động ghi khi opcode = Memory write

Nếu addr = Adapter&Phy, nó sẽ che đi bit dữ liệu MSB và gửi yêu cầu với bit dữ liệu LSB.



Hình 2.29: Dạng sóng của CSR Handler

2.5.9. Khối điều khiển Sideband



Hình 2.30: Khối điều khiển Sideband

Khối Điều khiển Sideband nhận yêu cầu Sideband hợp lệ từ FSM cùng với sb_type để xác định loại dữ liệu cần xử lý. Dựa trên sb_type, nó sẽ chuyển đổi dữ liệu thành đầu ra (giao diện song song với 66 bit). Khi nhận được tín hiệu xác nhận (acknowledge) từ bộ phân phối, khối sẽ tắt tín hiệu hợp lệ và gửi sb_ack hoặc cmpl_done trở lại các khối tương ứng để báo hiệu rằng dữ liệu đã được xử lý và chuyển thành công.
Bảng dưới đây mô tả đầy đủ các thông số của khối:

Bảng 2.18: Mô tả chi tiết thông số khối điều khiển Sideband

Pin Name	Direction	Clock	Size	Description
CTL control Ports				
clk	Input	-	1	Clock
rst_n	Input	Async	1	Asynchronous reset, active low.
init	Input	clk	1	Sync init, active high.
Sideband CTL Ports				
ireq[63:0]	Input	clk	64	Request data.
oreq_sb_valid	Output	clk	1	Inform the request data to Sideband CTL is valid.
osb_type[1:0]	Output	clk	2	See below.
icmpl_phy [63:0]	Output	clk	64	Completion data from PHY.
icmpl_stt	Input	clk	1	Status of completion data with 1 is true and 0 is error.
icmpl_csr_data[63:0]	Input	clk	64	Completion data with data from CSR Control.
orac_rec_p2s_valid	Output	clk	1	Inform the output data packet is valid.
orac_rec_p2s_data[65:0]	Output	clk	66	Output data packet.
irac_rec_p2s_ack	Input	clk	1	The acknowledgment signal confirms that data has been successfully received and the system is ready to receive the next data.
osb_ack	Output	clk	1	The signal confirms that request PHY packet has been transmitted.
ocmpl_done	Output	clk	1	The signal confirms that completion packet has been transmitted.

The "sb_type[1:0]" có 4 loại:

- *req (2'b00)* → Điều khiển SideBand sẽ che mặt dưới của dữ liệu yêu cầu, đặt tag[1:0] = 2'b01 và gửi yêu cầu đến PHY cục bộ.

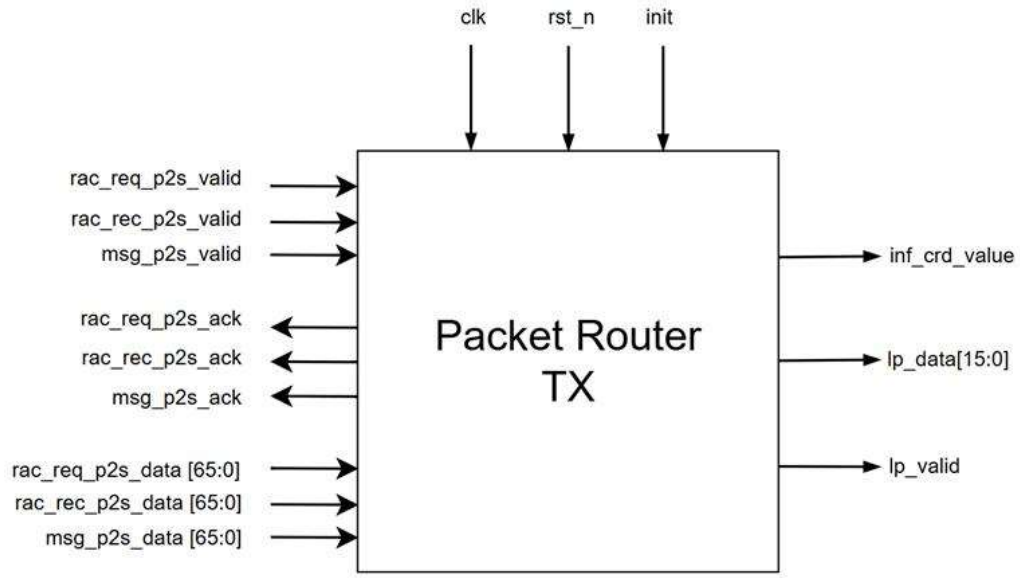
- *cmpl_csr (2'b01)* → Điều khiển SideBand nhận dữ liệu CSR và sau đó tạo gói hoàn tất để gửi đi.
- *cmpl_mix (2'b11)* → Điều khiển SideBand nhận dữ liệu CSR và chờ các yêu cầu dữ liệu hoàn tất, sau đó hợp nhất chúng và gửi đi.
- *cmpl_error (2'b10)* → Điều khiển SideBand sẽ tạo một gói hoàn tất lỗi với dữ liệu là Header của gói tin yêu cầu và gửi đi.

2.6. Khối Packet TX Router (Khối định tuyến gói tin truyền đi)

2.6.1. Tổng quan

Khối này sẽ nhận các gói tin theo dạng song song từ khối Remote access requester, khối Remote access receiver và bộ điều khiển message, sử dụng một bộ phân xử (arbiter) để cho phép các gói tin lần lượt vào.

- Khối này cũng sẽ tính toán lại giá trị parity của gói tin đó.
- Sau đó, gói tin được lưu trữ trong FIFO dựa trên opcode, trước khi được truyền đến PHY thông qua một bộ phân xử khác.



Hình 2.31: Sơ đồ khối – khối định tuyến gói tin truyền đi

2.6.2. Thông số và sơ đồ chi tiết của khối

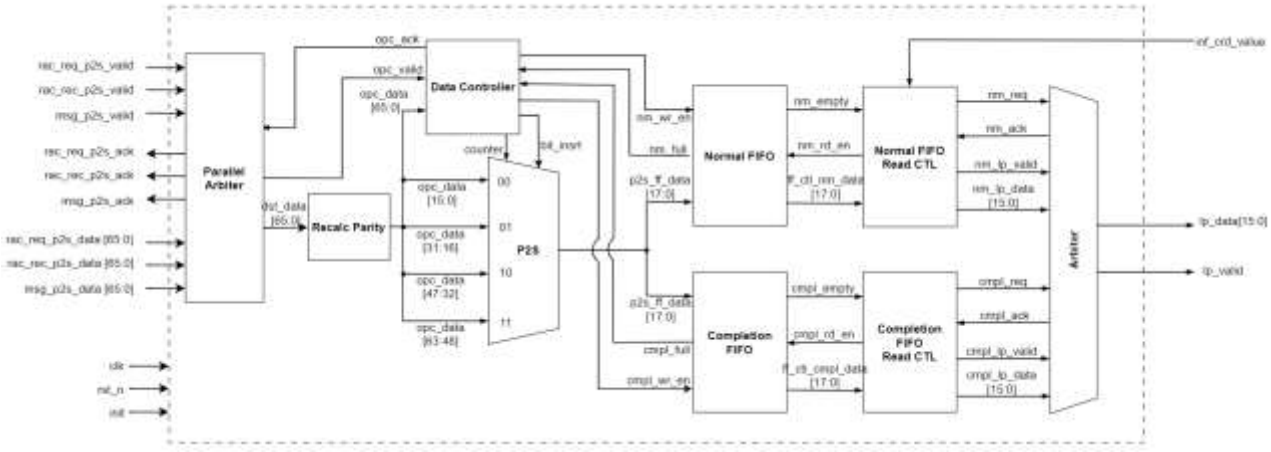
Bảng dưới đây mô tả đầy đủ các thông số của khối:

Bảng 2.19: Thông số của các giao diện khối Packet TX Router

Pin name	Direction	Clock	Size	Description
CTL Control Ports				
clk	Input	_	1	2GHz clock.
rst_n	Input	Async	1	Asynchronous reset, active low.

init	Input	clk	1	Sync init, active high.
Parallel Interface				
rac_req_p2s_valid	Input	clk	1	Request signal from the Remote Access Requester block.
rac_req_p2s_data	Input	clk	66	Data bus signals from the Remote Access Requester block.
rac_rec_p2s_valid	Input	clk	1	Request signal from the Remote Access Receiver block.
rac_rec_p2s_data	Input	clk	66	Data bus signals from the Remote Access Receiver block.
msg_p2s_valid	Input	clk	1	Request signal from the Message Controller.
msg_p2s_data	Input	clk	66	Data bus signals from the Message Controller.
rac_req_p2s_ack	Output	clk	1	Acknowledge signal to the Remote Access Requester block.
rac_rec_p2s_ack	Output	clk	1	Acknowledge signal to the Remote Access Receiver block.
msg_p2s_ack	Output	clk	1	Acknowledge signal to the Message Controller.
Sideband Interface				
lp_data [15:0]	Output	clk	16	Data from the Adapter to PHY.
lp_valid	Output	clk	1	Inform the data Adapter2Phy is valid.

Dưới đây là sơ đồ chi tiết của khối:



Hình 2.32: Mô tả chi tiết khối định tuyến gói tin truyền đi

Khối định tuyến gói tin truyền đi bao gồm các khối con sau:

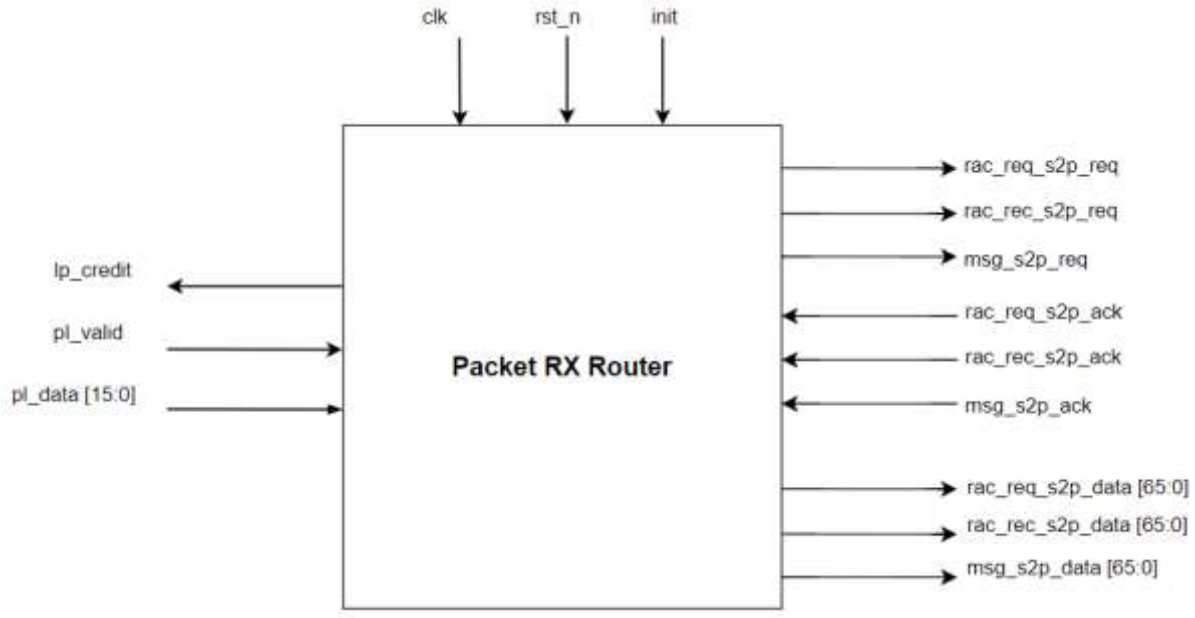
- Khối parallel arbiter giúp cho các gói tin đi qua một cách lần lượt tránh xung đột lẫn nhau khi có nhiều gói tin muốn truyền đi.
- Khối recalc parity dùng để tính lại giá trị parity của gói tin sau khi khởi tạo hoặc được cập nhật.
- Khối TX data controller dùng để phân loại gói tin và lưu trữ chúng vào FIFO tương ứng.
- Khối parallel to serial dùng để chuyển gói tin từ dạng parallel sang dạng serial.
- Khối count packet FIFO là nơi lưu trữ các gói tin trước khi chúng được truyền đi.
- Khối FIFO read CTL dùng để đọc các gói tin từ FIFO.
- Khối P2S arbiter dùng để phân xử các gói tin đi ra từ 2 FIFO, tránh tình trạng xung đột khi truyền gói tin.

2.7. Khối Packet RX Router (Khối định tuyến gói tin nhận)

2.7.1. Tổng quan

Khối này sẽ nhận các gói tin theo dạng nối tiếp từ PHY, các gói tin được lưu vào FIFO tương ứng dựa trên opcode. Các FIFO này sẽ gửi yêu cầu đến arbiter để thông báo rằng có một gói tin cần được truyền đi.

Sau đó, gói tin sẽ được chuyển đổi sang dạng song song và kiểm tra tính chẵn lẻ. Cuối cùng, các gói tin này sẽ được định tuyến đến đích tương ứng thông qua khối destination decoder.



Hình 2.33: Khối định tuyến gói tin nhận

2.7.2. Thông số và sơ đồ chi tiết của khối

Bảng dưới đây mô tả đầy đủ các thông số của khối:

Bảng 2.20: Thông số của các giao diện khối Packet RX Router

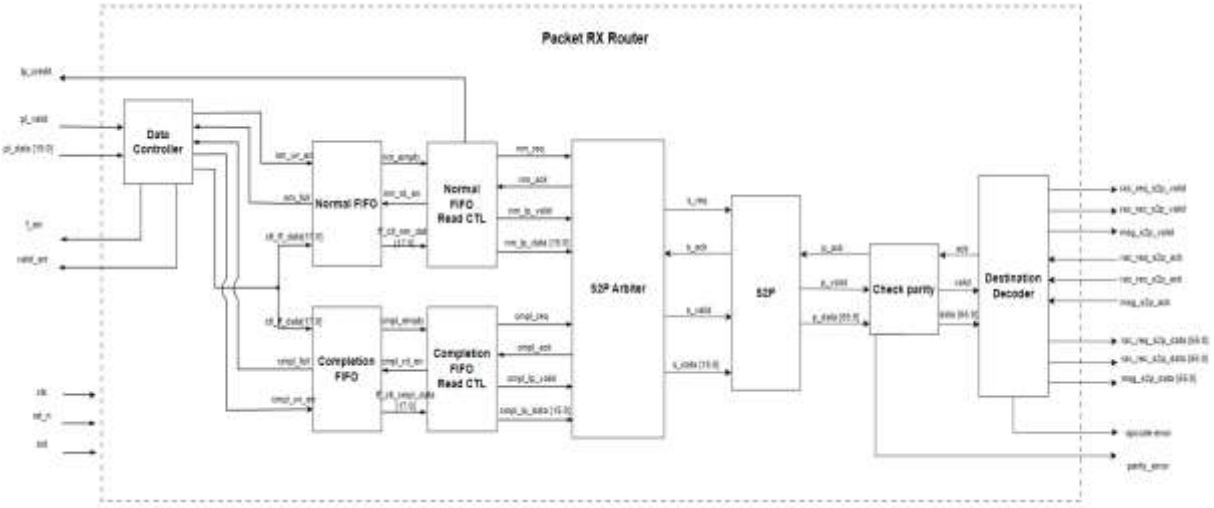
Pin name	Direction	Clock	Size	Description
Interface Ports				
clk	Input	_	1	2GHz clock.
rst_n	Input	Async	1	Asynchronous reset, active low.
pl_valid	Input	clk	1	Inform the data Phy2Adapter is valid.
pl_data	Input	clk	16	Data from PHY to the Adapter.
lp_credit	Output	clk	1	Credit from PHY to the Adapter
rac_req_s2p_valid	Output	clk	1	Request signal to the Remote access requester block.
rac_req_s2p_data	Output	clk	66	Data bus signals to the Remote access requester block.
rac_rec_s2p_valid	Output	clk	1	Request signal to the Remote access receiver block.
rac_rec_s2p_data	Output	clk	66	Data bus signals to the Remote access receiver block.
msg_s2p_valid	Output	clk	1	Request signal to the Message Controller.
msg_s2p_data	Output	clk	66	Data bus signals to the Message Controller.
rac_req_s2p_ack	Input	clk	1	Acknowledge signal from the Remote access requester block.
rac_rec_s2p_ack	Input	clk	1	Acknowledge signal from the Remote access receiver block.
msg_s2p_ack	Input	clk	1	Acknowledge signal from the Message Controller.

Khối định tuyến gói tin nhận bao gồm các khối con sau:

- Khối RX data controller dùng để phân loại gói tin nhận được và lưu trữ chúng vào FIFO tương ứng.
- Khối count packet FIFO là nơi lưu trữ các gói tin nhận được
- Khối FIFO read CTL dùng để đọc các gói tin từ FIFO.
- Khối S2P arbiter dùng để phân xử các gói tin đi ra từ 2 FIFO, tránh tình trạng xung đột đảm bảo các gói tin sẽ được xử lí một cách lần lượt.

- Khối serial to parallel dùng để chuyển gói tin từ dạng serial thành sang parallel.
- Khối check parity dùng để kiểm tra tính toàn vẹn của gói tin, nếu sai thì gói tin sẽ được bỏ qua.
- Khối destination decoder dùng để định tuyến các gói tin nhận đến các khối xử lí tiếp theo tương ứng.

Dưới đây là sơ đồ chi tiết của khối:

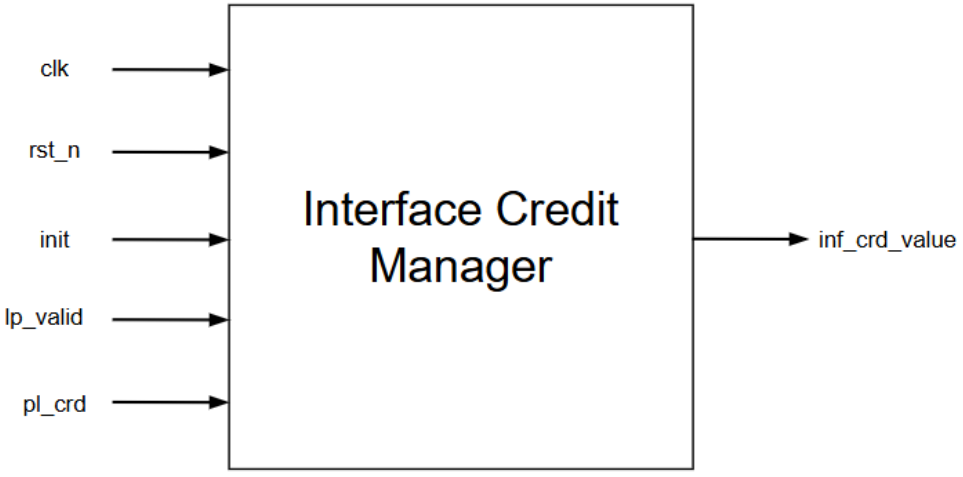


Hình 2.34: Mô tả chi tiết khối định tuyến gói tin nhận

2.8. Khối Interface credit manager

Adapter nên có bộ đếm để đếm số lượng credit trong PHY – gọi là Quản lý Tín dụng Giao diện (Interface Credit Manager). Sau khi đặt lại, credit mặc định là 2. Adapter chỉ gửi gói tin mới khi bộ đếm không bằng không. Điều này có nghĩa là bộ chuyển đổi có thể gửi liên tiếp 2 gói tin mà không lo PHY bị tràn.

PHY sẽ trả về pl_crd khi đã xử lý một gói tin (4 pha hoặc 2 pha) và sẵn sàng cho gói tiếp theo.



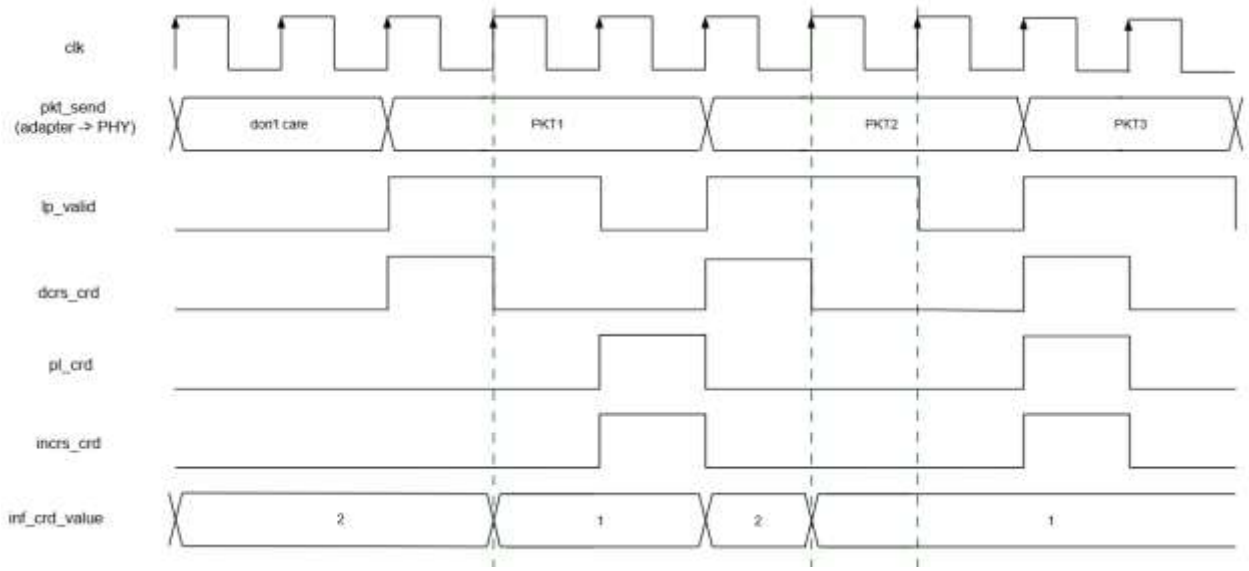
Hình 2.35: Khối Interface Credit Manger

Bảng dưới đây mô tả đầy đủ các thông số của khối:

Bảng 2.21: Mô tả chi tiết thông số của khối Interface Credit Manager

Pin name	Direction	Clock	Size	Description
CTL Control Ports				
clk	Input	_	1	2GHz clock.
rst_n	Input	Async	1	Asynchronous reset, active low.
init	Input	clk	1	Sync init, active high.
FIFO Interface				
lp_valid	Input	clk	1	Inform the lp_data sent from Adapter to PHY is valid.
pl_crd	Input	clk	1	Credit from PHY to the Adapter.
inf_crd_value	Output	clk	2	Number of credits in the PHY.

Dưới đây là dạng sóng của khối:



Hình 2.36: Dạng sóng của Interface credit manager

Khi tín hiệu hợp lệ (valid) được kích hoạt (cho biết một gói tin mới đã được chuyển từ Adapter sang PHY), giá trị của inf_crd_value sẽ giảm đi 1. Khi PHY đã xử lý xong một gói tin và pl_crd được kích hoạt, giá trị của inf_crd_value sẽ tăng thêm 1. Để quản lý quá trình tăng và giảm này, hai biến tạm thời, dcrs_crd và incrs_crd đã được tạo ra. Các biến này phát hiện khi nào cần cập nhật giá trị của inf_crd_value và kích hoạt tín hiệu cập nhật phù hợp.

2.9. Kết luận

Qua chương này, ta đã thiết kế xong tất cả các khối để thực hiện các chức năng theo yêu cầu của thiết kế. Để kiểm chứng thiết kế có hoạt động đúng với mong muốn không, ta sẽ tiến hành thực nghiệm ở chương 3.

Chương 3: KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ

3.1. Giới thiệu

Để đảm bảo chất lượng và độ tin cậy của thiết kế phần cứng, quy trình đánh giá được thực hiện qua ba giai đoạn chính:

- Xác minh thiết kế (Design Verification).
- Tổng hợp logic và phân tích thời gian tĩnh (Synthesis & STA).
- Thiết kế cho kiểm thử (Design for Testability – DFT).

Chương này trình bày quy trình thực nghiệm và đánh giá của thiết kế.

3.2. Xác minh thiết kế

Design verification (xác minh thiết kế) là quá trình kiểm tra và đánh giá để đảm bảo rằng một thiết kế đáp ứng các yêu cầu kỹ thuật, tiêu chuẩn và mục tiêu ban đầu đề ra, qua đó xác nhận rằng sản phẩm hoặc hệ thống đã được thiết kế đúng cách trước khi tiến hành sản xuất hoặc triển khai thực tế.

3.2.1. Môi trường kiểm thử

Để xác minh thiết kế, môi trường xác minh thiết kế là cần thiết. Trong phần này, một môi trường đơn giản sẽ được xây dựng bao gồm:

+ Thiết kế phần tử số (digital top design).

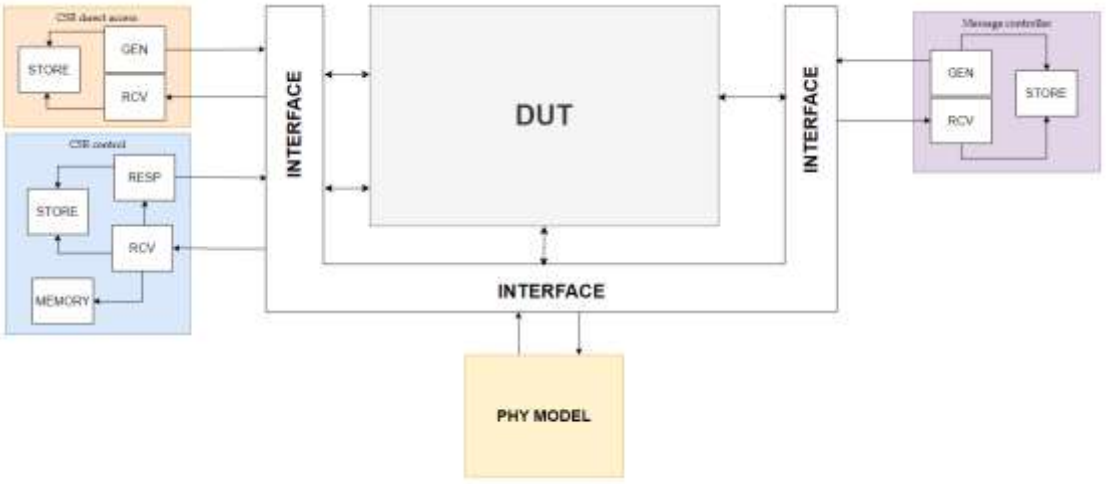
+ Các mô hình hành vi của các khối CSR truy cập trực tiếp, điều khiển CSR, bộ điều khiển tin nhắn và mô hình PHY.

+ Giao diện để kết nối DUT với các mô hình.

+ Các mục kiểm thử theo danh sách kiểm tra xác minh thiết kế.

+ Môi trường tổng thể bao gồm tất cả các thành phần trên.

Dưới đây là hình ảnh mô tả môi trường của xác minh thiết kế:



Hình 3.1: Cấu trúc môi trường kiểm thử

3.2.2. Các trường hợp xác minh

Bảng dưới đây là tổng hợp tất cả các trường hợp xác minh:

Bảng 3.1: Danh sách các trường hợp kiểm tra

Tính năng	Tính năng phụ	Tính năng phụ 2	Trạng thái
Xử lý gói tin từ Adapter đến PHY	Message CTL	Message controller gửi message có dữ liệu tới PHY	Hoàn thành
		Message controller gửi message không có dữ liệu tới PHY	Hoàn thành
		Message controller nhận message có dữ liệu từ PHY	Hoàn thành
		Message controller nhận message không có dữ liệu từ PHY	Hoàn thành
		Message controller gửi ngẫu nhiên có và không có dữ liệu đến PHY	Hoàn thành
		Message controller nhận ngẫu nhiên có và không có dữ liệu đến PHY	Hoàn thành
	Reset/Init	Reset/Init khi gửi gói tin	Hoàn thành
	CSR Direct Access (Mailbox) gửi gói tin thông thường đến PHY	CSR Direct Access gửi yêu cầu đọc cho adapter bên kia	Hoàn thành

		CSR Direct Access gửi yêu cầu viết cho adapter bên kia	Hoàn thành
		CSR Direct Access nhận gói tin completion từ PHY với trạng thái là stall để yêu cầu reset timer	Hoàn thành
		CSR Direct Access nhận gói tin completion từ PHY với trạng thái là stall để yêu cầu reset timer. Sau đó nhận gói tin với trạng thái là done	Hoàn thành
		CSR Direct Access nhận gói tin completion từ PHY (trước đó đã gửi request đọc/ghi) từ PHY với status là Completion Success	Hoàn thành
	CSR Direct Access (Mailbox) nhận gói tin lỗi từ PHY	CSR Direct Access nhận gói tin trả về với trạng thái lỗi	Hoàn thành
		CSR Direct Access gửi tới PHY với chức năng đọc dữ liệu nhưng không nhận được gói tin trả về để kiểm tra timeout > 8ms	Hoàn thành
		4 ways handshaking	Hoàn thành
Xử lý gói tin từ Phy đến Adapter	Remote adapter gửi gói tin yêu cầu với địa chỉ thuộc vùng Adapter	Gói tin ghi dữ liệu	Hoàn thành
		Gói tin đọc dữ liệu	Hoàn thành
	Remote adapter gửi gói tin yêu cầu với địa chỉ thuộc vùng Adapter+Phy	Gói tin ghi dữ liệu	Hoàn thành

		Gói tin đọc dữ liệu	Hoàn thành
	Sideband interface gửi gói tin lỗi	Opcode không hợp lệ	Hoàn thành
		Địa chỉ không hợp lệ	Hoàn thành
		Nhận gói tin trả về trong khi không gửi gói tin yêu cầu	Hoàn thành
		Gói tin trả về với trạng thái lỗi gửi đến khối Remote access receiver	Hoàn thành
	Adapter nhận nhiều loại gói tin		Hoàn thành
	Adapter nhận và gửi gói tin từ nhiều nguồn		Hoàn thành
	Adapter nhận gói tin trả về từ PHY với tag lỗi		Hoàn thành

3.2.3. Phát hiện lỗi

Khi tiến hành thực nghiệm các trường hợp xác minh ở mục 3.2.2. Khi phát hiện lỗi ta sẽ quay lại bước đầu tiên để sửa là sửa code RTL của những khối lỗi. Sau đó kiểm tra Spyglass rồi tiến hành xác minh lại lần nữa. Ta sẽ cập nhật code cho tới khi nào không còn lỗi nào thì sẽ chuyển sang bước tiếp theo là Synthesis và STA. Kết quả cuối cùng của giai đoạn xác minh này là 1 tệp chứa tất cả đường dẫn của các tệp RTL của thiết kế để phục vụ cho quá trình tổng hợp.

Từ kết quả của các trường hợp kiểm thử, ta rút ra được bảng sau:

Bảng 3.2: Danh sách lỗi được phát hiện

STT	Lỗi phát hiện
1	Khi timer đếm hơn 8ms nhưng mb_req_valid và mb_resp_valid không giống waveform mong đợi.
2	csr_we bị vô hiệu hóa trong quá trình ghi bộ nhớ, do đó Remote access receiver không thể ghi dữ liệu vào bộ nhớ
3	Count Packet FIFO trong Packet Router chỉ có thể lưu trữ 6 gói tin đầu tiên vì giới hạn trên của DEPTH FIFO chưa được đặt khi cập nhật biến đếm
4	FIFO trong Remote access receiver không thể xử lý khi wr_en và rd_en lên 1 cùng một thời điểm
5	csr_we kích hoạt trong quá trình đọc bộ nhớ, vì vậy Remote access receiver không đọc được dữ liệu từ bộ nhớ
6	Remote access receiver không thể xử lý 2 gói tin liên tiếp dạng hỗn hợp
7	Trong khối Packet router, dữ liệu P2S Completion FIFO được đọc cùng với lp_crd.
8	Khi nhận nhiều gói tin lỗi liên tục, cờ lỗi chỉ báo một lần duy nhất cho gói tin đầu tiên

3.3. Tổng hợp logic và phân tích thời gian tĩnh

Trong quy trình thiết kế hệ thống số, sau công đoạn xác minh thiết kế, thiết kế RTL sẽ được chuyển sang bước tổng hợp. Ở bước này, netlist RTL được chuyển đổi thành các cổng logic. Các ràng buộc thiết kế (dựa trên đặc tả kỹ thuật) và thư viện chuyên dụng sẽ được sử dụng để xác định cách công cụ ánh xạ và tối ưu hóa thiết kế. Kết quả sau quá trình tổng hợp cần được phân tích và đánh giá.

Phân tích thời gian tĩnh (STA) là một trong những kỹ thuật phổ biến được sử dụng để kiểm tra thời gian trong thiết kế số. Nếu xảy ra vi phạm thời gian (như vi phạm thời gian thiết lập hoặc thời gian giữ), chip sau khi chế tạo có thể không hoạt động đúng

chức năng như mong muốn. Trong quy trình thiết kế số, STA được sử dụng để xác minh rằng thiết kế có thể hoạt động chính xác sau khi được sản xuất.

Khác với phân tích thời gian động (dynamic timing analysis), STA không phụ thuộc vào giá trị dữ liệu cụ thể tại các chân đầu vào. Thay vào đó, STA thực hiện phân tích một cách tĩnh bằng cách lan truyền các độ trễ qua các cổng logic. Nguyên lý chính của STA là cộng thời gian đến của tín hiệu với độ trễ của từng cổng logic, sau đó tìm giá trị lớn nhất trong số các thời điểm đến tại mỗi nút (net). Toàn bộ thiết kế sẽ được phân tích một lần, và tất cả các đường truyền và kịch bản có thể xảy ra sẽ được kiểm tra để đảm bảo đáp ứng các yêu cầu về thời gian [6].

3.3.1. Các ràng buộc

Việc thiết lập các thông số như tần số, chu kỳ và độ bất định (uncertainty) cho các clock giúp công cụ tổng hợp (synthesis) và phân tích timing (STA) hiểu rõ yêu cầu về thời gian của hệ thống. Nhờ đó, công cụ có thể tối ưu hóa, kiểm tra và đảm bảo thiết kế đáp ứng đúng các tiêu chí về timing, giúp mạch hoạt động ổn định ở các tần số mong muốn. Đây là bước quan trọng đầu tiên trong quá trình thiết kế vi mạch số.
Dưới đây là các bảng ràng buộc cho quá trình tổng hợp:

Bảng 3.3: Synthesis – các thông số để ràng buộc

SDC clock name	Clock domain	Frequency	Period (ns)
CLK	clk	2 GHz	0.5
SCLK	sclk	20 MHz	50

Bảng 3.4: Synthesis – clock uncertainty

SDC clock name	Clock uncertainty	
	Setup	Hold
CLK	0.025	0.025
SCLK	2.5	2.5

Bảng 3.5: Synthesis – độ trễ đầu vào và đầu ra

Type	Post direction	Value (ns)
		clk
Minimum delay	Input	0.100
	Output	0.100
Maximum delay	Inout	0.330
	Output	0.330

3.3.2. Kết quả tổng hợp logic và phân tích thời gian tĩnh

Trong thiết kế này, chúng tôi chỉ tập trung vào đường dẫn reg2reg. Các đường dẫn còn lại còn có một số hạn chế và sẽ được cải thiện trong tương lai. Đối với đường dẫn reg2reg bên trong thanh ghi Remote access requester, các đầu vào được kích hoạt bởi clk và các thanh ghi sẽ lấy mẫu bởi sclk. Việc sử dụng các miền clocks khác nhau đã dẫn đến không thể đồng bộ hóa chính xác các đường dẫn này về mặt thời gian.

Dưới đây là bảng kết quả của giai đoạn Synthesis và STA

Bảng 3.6: Kết quả Synthesis và STA

Start point	**Start point clock domain**	**Endpoint**	**End point clock domain**	**Path class**	**Status**
Mailbox inputs	asyn	RAC_REQ register	clk	in2 reg	Exception
Message inputs	clk	PACKET_TX_ROUTER registers	clk	in2 reg	Met
Phy inputs	clk	PACKET_RX_ROUTER registers	clk	in2 reg	Met
CSR inputs	clk	PACKET_TX_ROUTER registers	clk	in2 reg	Met
RAC_REQ register	clk	RAC_REQ registers	sclk	reg2 reg	Exception
RAC_REQ registers	clk	PACKET_TX_ROUTER registers	clk	reg2 reg	Met
PACKET_RX_ ROUTER register	clk	RAC_REQ registers	clk	reg2 reg	Met
PACKET_RX_ ROUTER register	clk	RAC_REC registers	clk	reg2 reg	Met
PACKET_TX_ ROUTER registers	clk	Phy outputs	clk	reg2 out	Met
RAC_REQ registers	clk	Mailbox outputs	asyn	reg2 out	Met
RAC_REC registers	clk	CSR outputs	clk	reg2 out	Met
PACKET_RX_ ROUTER registers	clk	Message outputs	clk	reg2 out	Met

- Báo cáo area:

  Tổng cell area của thiết kế là 5688.92 μm^2

```
Number of ports:                         5651
Number of nets:                         15439
Number of cells:                        10313
Number of combinational cells:           7425
Number of sequential cells:              2835
Number of macros/black boxes:               0
Number of buf/inv:                       1160
Number of references:                      13

Combinational area:               1858.985864
Buf/Inv area:                      147.813118
Noncombinational area:            3829.932151
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (Wire load has zero net area)

Total cell area:                  5688.918016
Total area:                 undefined
```

Hình 3.2: Báo cáo area

- Báo cáo power:

```
  Cell Internal Power  =  12.5498 mW   (95%)
  Net Switching Power  = 645.9316 uW    (5%)
                         ---------
Total Dynamic Power    =  13.1958 mW  (100%)

Cell Leakage Power     = 378.9969 nW


                 Internal         Switching         Leakage           Total
Power Group      Power            Power             Power             Power   (   %    ) Attrs
----------------------------------------------------------------------------------------------
io_pad             0.0000           0.0000            0.0000            0.0000  (   0.00%)
memory             0.0000           0.0000            0.0000            0.0000  (   0.00%)
black_box          0.0000           0.0000            0.0000            0.0000  (   0.00%)
clock_network     12.2615           0.0000            0.0000           12.2615  (  92.92%)  i
register      6.5666e-02        1.4908e-02          289.9094        8.0858e-02  (   0.61%)
sequential         0.0000           0.0000            0.0000            0.0000  (   0.00%)
combinational      0.2226           0.6310           86.7816            0.8538  (   6.47%)
----------------------------------------------------------------------------------------------
Total            12.5498 mW         0.6459 mW      376.6910 nW       13.1961 mW
```

Hình 3.3: Báo cáo power

Cell leakage power rất thấp (khoảng 378,9969 nW), góp phần không đáng kể vào tổng tiêu thụ năng lượng, cho thấy thiết kế RTL đã được tối ưu hóa tốt để giảm rò rỉ, phù hợp với mục tiêu thiết kế tiết kiệm năng lượng và hiệu quả.

3.4. Thiết kế cho kiểm thử

DFT là một kỹ thuật được áp dụng trong quá trình thiết kế mạch số, nhằm thêm vào các mạch kiểm thử bên trong mạch chính. Các mạch bổ sung này cho phép thực hiện việc kiểm thử toàn diện và dễ dàng sau khi sản xuất, mà không phụ thuộc vào chức năng cụ thể của thiết kế. Quá trình sản xuất không thể đảm bảo rằng tất cả các chip được tạo ra đều đạt chất lượng và không có lỗi. Các lỗi này thường là lỗi vật lý phát sinh trong

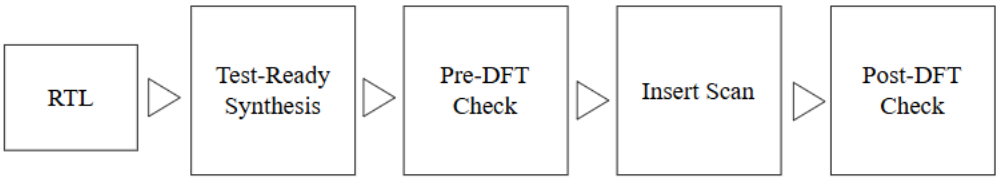
quá trình chế tạo, chứ không phải lỗi chức năng hay lỗi logic do thiết kế sai. Trong dự án của chúng tôi, kỹ thuật *scan insertion* thuộc DFT được áp dụng để chèn các mạch bổ sung. Kỹ thuật này giúp tăng khả năng điều khiển và quan sát bên trong mạch, từ đó nâng cao độ bao phủ kiểm thử và khả năng phát hiện lỗi.

Yêu cầu đối với DFT:

- Tất cả các thanh ghi trong thiết kế đều được đưa vào chuỗi quét.
- Tất cả các chuỗi quét sử dụng phương pháp quét tiêu chuẩn.
- Tất cả các chuỗi quét đều có lockup latch tại đầu ra của chúng (trước cổng ra scan_out).
- Tất cả các chuỗi quét có độ dài chuỗi quét tối đa dưới 200.

3.4.1. DFT Design Rule Check (DRC)

Đây là tóm tắt cho quy trình chèn scan:



Hình 3.4: Quy trình chèn scan

Pre-DFT DRC: là giai đoạn kiểm tra thiết kế trước khi chèn DFT nhằm đảm bảo rằng thiết kế hiện tại:

- Đáp ứng đầy đủ các yêu cầu cho việc chèn DFT.
- Không tồn tại các lỗi logic, cấu trúc hoặc kiến trúc cản trở quá trình DFT về sau.

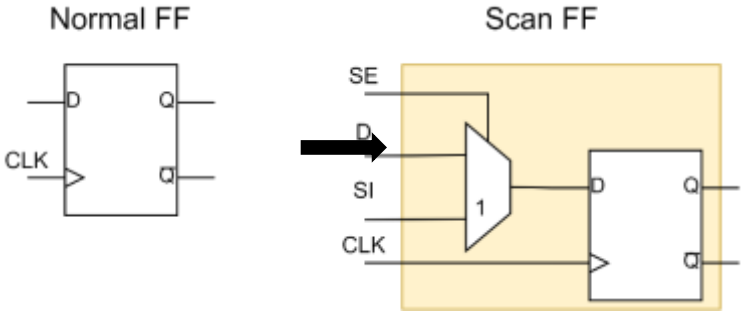
Dưới đây là kết quả của pre-DFT DRC sau khi chạy tool:

```
In mode: all_dft...
  Pre-DFT DRC enabled

Information: Starting test design rule checking. (TEST-222)
  Loading test protocol
  ...basic checks...
  ...basic sequential cell checks...
    ...checking for scan equivalents...
  ...checking vector rules...
  ...checking pre-dft rules...

---------------------------------------------------------------
  DRC Report

  Total violations: 0

---------------------------------------------------------------


Test Design rule checking did not find violations

---------------------------------------------------------------
  Sequential Cell Report

  0 out of 2835 sequential cells have violations
```

Hình 3.5: Kết quả pre_DFT

Chèn DFT: Thêm các phần tử kiểm thử vào thiết kế.

Trong kỹ thuật này, tất cả các flip-flop thông thường trong thiết kế sẽ được chuyển đổi thành *scan flip-flop*, ngoại trừ những flip-flop bị loại trừ bởi kỹ sư thiết kế hoặc những flip-flop vi phạm các quy tắc kiểm tra DFT (DFT DRC violations). Việc chuyển đổi này giúp tăng khả năng kiểm soát và quan sát các tín hiệu bên trong mạch, từ đó cho phép phát hiện hiệu quả các lỗi như stuck-at-0 hoặc stuck-at-1 trong quá trình kiểm thử sau sản xuất.



Hình 3.6: Chuyển flip-flop thông thường thành scan flip-flop

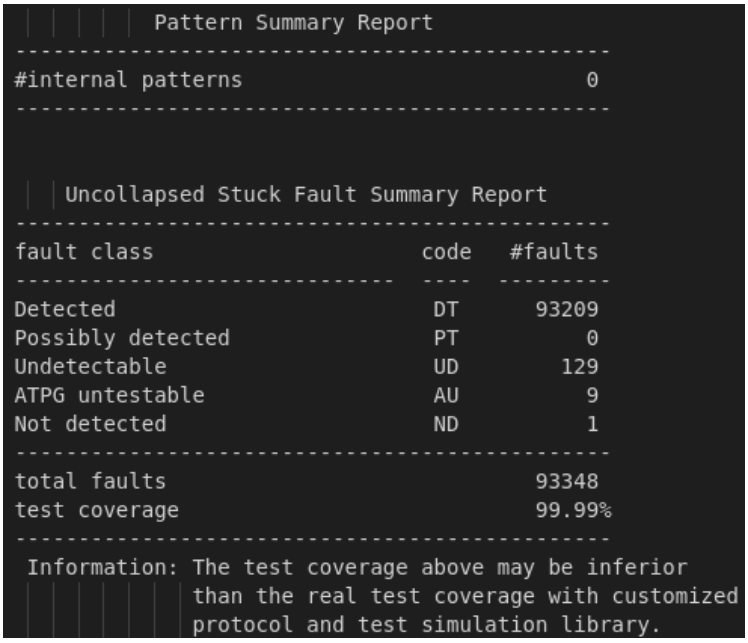
Các *scan flip-flop* sau khi được chèn vào thiết kế sẽ được kết nối với nhau để tạo thành các *chuỗi scan (scan chains).* Mỗi chuỗi scan là một dãy các flip-flop được nối tiếp nhau, cho phép dữ liệu kiểm thử được dịch chuyển tuần tự qua toàn bộ chuỗi. Số lượng chuỗi scan trong một thiết kế phụ thuộc vào nhiều yếu tố kỹ thuật trong đặc tả DFT, chẳng hạn như độ phức tạp của mạch, yêu cầu về thời gian kiểm thử, và khả năng điều khiển/quan sát tín hiệu bên trong mạch.

```
Scan_path    Len   ScanDataIn  ScanDataOut ScanEnable  MasterClock SlaveClock
-----------  ----- ----------- ----------- ----------- ----------- -----------
I clk_scan_chain_0 176 scan_in[0] scan_out[0] scan_shift[0] clk     -
I clk_scan_chain_1 176 scan_in[1] scan_out[1] scan_shift[0] clk     -
I clk_scan_chain_2 176 scan_in[2] scan_out[2] scan_shift[0] clk     -
I clk_scan_chain_3 176 scan_in[3] scan_out[3] scan_shift[0] clk     -
I clk_scan_chain_4 176 scan_in[4] scan_out[4] scan_shift[0] clk     -
I clk_scan_chain_5 176 scan_in[5] scan_out[5] scan_shift[0] clk     -
I clk_scan_chain_6 176 scan_in[6] scan_out[6] scan_shift[0] clk     -
I clk_scan_chain_7 176 scan_in[7] scan_out[7] scan_shift[0] clk     -
I clk_scan_chain_8 176 scan_in[8] scan_out[8] scan_shift[0] clk     -
I clk_scan_chain_9 175 scan_in[9] scan_out[9] scan_shift[0] clk     -
I clk_scan_chain_10 175 scan_in[10] scan_out[10] scan_shift[0] clk -
I clk_scan_chain_11 175 scan_in[11] scan_out[11] scan_shift[0] clk -
I clk_scan_chain_12 175 scan_in[12] scan_out[12] scan_shift[0] clk -
I clk_scan_chain_13 175 scan_in[13] scan_out[13] scan_shift[0] clk -
I clk_scan_chain_14 175 scan_in[14] scan_out[14] scan_shift[0] clk -
I clk_scan_chain_15 175 scan_in[15] scan_out[15] scan_shift[0] clk -
I sclk_scan_chain_0 26 scan_in[16] scan_out[16] scan_shift[1] sclk -
```

Hình 3.7: Báo cáo chuỗi scan

Tổng số logic tuần tự sau tổng hợp là 2835 cells. So sánh với tổng số cells trong 17 chuỗi quét cũng là 2835 cells, điều này có nghĩa là tất cả các flip-flop trong thiết kế đã được chuyển đổi thành flip-flop quét và được thêm vào các chuỗi quét.

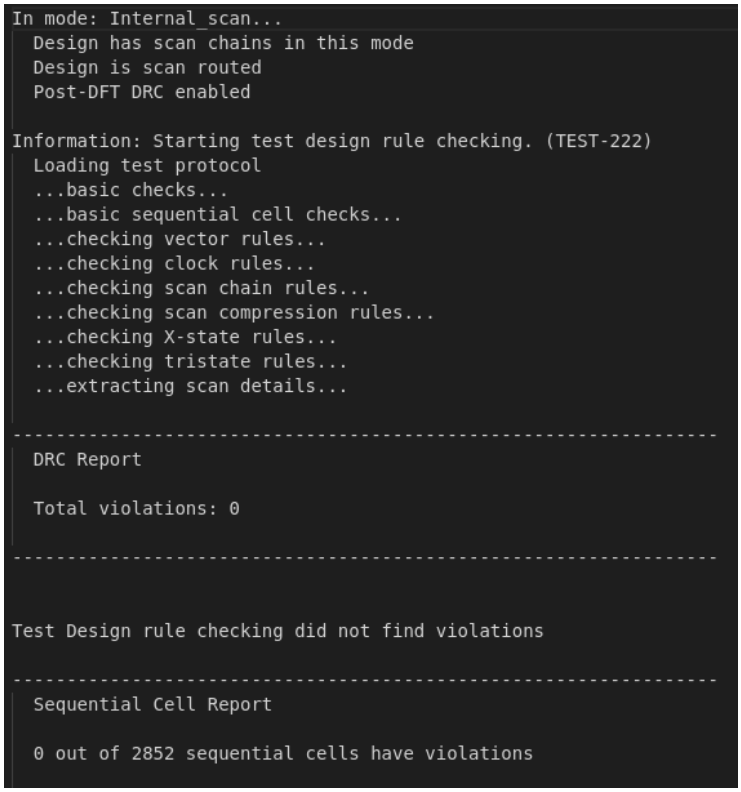
Coverage: Đo lường mức độ mà các lỗi tiềm năng trong thiết kế có thể được phát hiện bởi các vector kiểm thử.



```
        Pattern Summary Report
-----------------------------------------------
#internal patterns                          0
-----------------------------------------------


    Uncollapsed Stuck Fault Summary Report
-----------------------------------------------
fault class                     code   #faults
------------------------------  ----  ---------
Detected                         DT      93209
Possibly detected                PT          0
Undetectable                     UD        129
ATPG untestable                  AU          9
Not detected                     ND          1
-----------------------------------------------
total faults                               93348
test coverage                              99.99%
-----------------------------------------------
 Information: The test coverage above may be inferior
              than the real test coverage with customized
              protocol and test simulation library.
```

Hình 3.8: Báo cáo độ bao phủ

Post_DFT DRC: Kiểm tra lại thiết kế sau khi đã chèn các phần tử DFT.



```
In mode: Internal_scan...
  Design has scan chains in this mode
  Design is scan routed
  Post-DFT DRC enabled

Information: Starting test design rule checking. (TEST-222)
  Loading test protocol
  ...basic checks...
  ...basic sequential cell checks...
  ...checking vector rules...
  ...checking clock rules...
  ...checking scan chain rules...
  ...checking scan compression rules...
  ...checking X-state rules...
  ...checking tristate rules...
  ...extracting scan details...

--------------------------------------------------------------
  DRC Report

  Total violations: 0

--------------------------------------------------------------


Test Design rule checking did not find violations

--------------------------------------------------------------
  Sequential Cell Report

  0 out of 2852 sequential cells have violations
```

Hình 3.9: Kết quả post DFT

3.4.2. Kiểm tra formality cho Post DFT

Trước khi kiểm tra formality của thiết kế sau khi thêm chuỗi quét, các tín hiệu scan_shift cần được kéo về 0 để vô hiệu hóa logic quét đã được thêm vào sau khi chèn chuỗi quét.

Bảng 3.7: Cấu hình cho kiểm tra post DFT fomality

Netlist type	Design type	Constraint	Ports
RTL	Reference	set constant 0	scan_shift [0]
RTL	Reference	set constant 0	scan_shift [1]
SCAN	Implemented	set constant 0	scan_shift [0]
SCAN	Implemented	set constant 0	scan_shift [1]

Formality là một công cụ của Synopsys dùng để so sánh tính tương đương chức năng (formal equivalence checking) giữa hai phiên bản thiết kế số: ở đây là RTL (Register Transfer Level) và netlist sau khi DFT. Mục đích là đảm bảo rằng quá trình tổng hợp và DFT không làm thay đổi chức năng logic của thiết kế.

```
********************************** Verification Results **********************************
Verification SUCCEEDED
   ATTENTION: synopsys_auto_setup mode was enabled.
              See Synopsys Auto Setup Summary for details.
   ATTENTION: RTL interpretation messages were produced during link
              of reference design.
              Verification results may disagree with a logic simulator.
------------------------------------------------------------------------
 Reference design: ref:/WORK/top
 Implementation design: imp:/WORK/top
 2920 Passing compare points
------------------------------------------------------------------------------------------
Matched Compare Points     BBPin    Loop    BBNet     Cut     Port     DFF     LAT   TOTAL
------------------------------------------------------------------------------------------
Passing (equivalent)           0       0        0       0      199    2721       0    2920
Failing (not equivalent)       0       0        0       0        0       0       0       0
Not Compared
  Constant reg                                                        28       0      28
  Don't verify                 0       0        0       0       18       0       0      18
  Unread                       0       0        0       0        0      82       0      82
******************************************************************************************
```

Hình 3.10: Kết quả kiểm tra fomality

3.5. Kết luận

Sau quá trình tiến hành các thử nghiệm mô phỏng một cách kỹ lưỡng và có hệ thống, chúng ta đã thu thập được các dữ liệu và kết quả cần thiết để tổng hợp và phân tích thông qua các quy trình:

- RTL.
- Xác minh thiết kế.
- Tổng hợp logic và phân tích thời gian tĩnh.
- Thiết kế cho kiểm thử.

Từ đó rút ra bảng đánh giá sau:

Bảng 3.8: Kết quả các phần thiết kế

Tính năng	Tính năng phụ	Kết quả
Thiết kế RTL	Kiểm tra chức năng	Giống với yêu cầu thiết kế
	Kiểm tra Spyglass	Không có lỗi
Xác minh thiết kế	Checklist	Đã kiểm trả tất cả chức năng
	Độ bao phủ	100%
Synthesis/STA	Synthesis & STA	Không có lỗi
	Kiểm tra formality	Không có lỗi
DFT	Chèn SCAN	Tất cả scan cell đã được thêm vào chuỗi SCAN

Dựa trên kết quả của bảng đánh giá, thiết kế của chúng ta đã đáp ứng và vượt qua các tiêu chuẩn thực tế đề ra, không còn tồn tại lỗi về logic và chức năng sau khi thực hiện các phương pháp thử nghiệm thực nghiệm, chứng tỏ tính khả thi và độ tin cậy của giải pháp đề xuất.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết luận

Nhóm đã nắm rõ được kiến thức cơ bản về quy trình thiết kế vi mạch số, nắm rõ được từng quy trình có có vai trò gì trong quy trình thiết kế.

Thiết kế thành công Sideband Adapter từ những yêu cầu ban đầu, dựa trên các giao thức và kỹ thuật hiện có. Khối Sideband Adapter đã có thể thực hiện những chức năng theo mong muốn là truyền nhận và xử lý gói tin. Có thể tự xử lý và báo lỗi khi gặp các vấn đề lỗi. Khối này được mô tả chi tiết các thành phần, dạng sóng và nguyên lý hoạt động của từng phần. Giúp hiểu rõ hơn về Sideband Adapter.

Một kế hoạch xác minh đã được xây dựng để xác định các tính năng cần kiểm tra. Kế hoạch này tập trung vào việc mô phỏng và kiểm thử xem khối Sideband Adapter có thể nhận và xử lý gói tin được hay không. Để phục vụ cho quá trình xác minh, một môi trường kiểm thử hoàn chỉnh đã được phát triển, bao gồm các thành phần như Phy model, CSR direct access, CSR control, message controller và giao diện (Interface) đóng vai trò kết nối giữa môi trường mô phỏng và khối DUT.

Hạn chế

Ở phần Synthesis vẫn còn một số timing path bị lỗi, chủ yếu là ở đường in2reg và reg2reg. Nguyên nhân là do phần thiết kế RTL vẫn chưa tối ưu, gây lỗi timing ở một số đường như đã nói.

Hướng phát triển

Trong tương lai sẽ tiến hành tối ưu RTL để không còn các lỗi timing ở bước Synthesis. Ngoài ra sẽ phát triển thêm nhiều chức năng để có xử lý được nhiều loại gói tin hơn.

TÀI LIỆU THAM KHẢO

[1.] Synopsys, Universal Chiplet Interconnect Express (UCIe) Specification, Version 1.0, 2023. [Online]. Available: https://www.synopsys.com. [Accessed: Jan.–Jun. 2025].

[2.] Paul McLellan, "Die-To-Die Chiplet Communication", Semiconductor Engineering, 2022. [Online]. Available: https://semiengineering.com. [Accessed: Apr. 2025].

[3.] R. Zhao, B. Li, Z. Guo, and X. Li, "A Lightweight and Reliable Sideband Interface for Die-to-Die," in Proc. IEEE, 2022. [Online]. Available: https://ieeexplore.ieee.org. [Accessed: Feb. 2025].

[4.] C. E. Cummings and H. Chambers, Finite State Machine (FSM) Design & Synthesis using SystemVerilog, Sunburst Design, Inc., 2019. [Online]. Available: https://www.sunburst-design.com. [Accessed: Apr.–May. 2025].

[5.] C. E. Cummings, Clock Domain Crossing (CDC) Design & Verification Techniques Using SystemVerilog, Sunburst Design, Inc., 2008. [Online]. Available: https://www.sunburst-design.com. [Accessed: Jan.–Jun. 2025].

[6.] J. Bhasker and R. Chadha, Static Timing Analysis for Nanometer Designs, Springer, 2008. [Online]. Available: https://link.springer.com. [Accessed: Apr. 2025].

PHỤ LỤC

Phụ lục này trình bày danh sách các mạch logic cấp cổng (netlist) sau khi hoàn tất phần thiết kế cho kiểm thử (DFT), thể hiện kết nối chính xác của các tế bào tiêu chuẩn (standard cell) và mô-đun, sẵn sàng để bàn giao cho nhóm thiết kế vật lý thực hiện bố trí mạch và chế tạo chip. Netlist đã được tổng hợp dựa trên mô hình RTL đã được xác minh của khối Sideband Adapter, sử dụng công cụ Synopsys Design Compiler, đồng thời phản ánh tất cả các tối ưu hóa về chức năng và các ràng buộc đã được áp dụng trong quá trình tổng hợp để đảm bảo hiệu suất tối ưu cho thiết kế.

```
////////////////////////////////////////////////////////////
// Created by: Synopsys DC Ultra(TM) in wire load mode
// Version   : W-2024.09-SP5
// Date      : Tue May 20 17:50:03 2025
////////////////////////////////////////////////////////////
module
sideband_adapter_P_DWIDTH66_P_PKT_DWIDTH64_S_DWIDTH16_FF_DWIDTH18_OPCODE4_PHASE_OPC2_DEPTH24_PACKET6_CMPL_DEPTH8_CMPL_PACKET2_BIT_INSRT2_FF_CNT2_COUNT4_SRC3_CREDIT2_test_1 (
      clk, rst_n, init, rac_req_p2s_valid, rac_req_p2s_data, rac_req_p2s_ack,
      rac_rec_p2s_valid, rac_rec_p2s_data, rac_rec_p2s_ack, msg_p2s_valid,
      msg_p2s_data, msg_p2s_ack, lp_data, lp_valid, pl_crd, test_si16,
      test_si15, test_si14, test_si13, test_si12, test_si11, test_si10,
      test_si9, test_si8, test_si7, test_si6, test_si5, test_si4, test_si3,
      test_si2, test_si1, test_so16, test_so15, test_so14, test_so13,
      test_so12, test_so11, test_so10, test_so9, test_so8, test_so7,
      test_so6, test_so5, test_so4, test_so3, test_so2, test_so1, test_se );
  input [65:0] rac_req_p2s_data;
  input [65:0] rac_rec_p2s_data;
  input [65:0] msg_p2s_data;
  output [15:0] lp_data;
  input clk, rst_n, init, rac_req_p2s_valid, rac_rec_p2s_valid, msg_p2s_valid,
       pl_crd, test_si16, test_si15, test_si14, test_si13, test_si12,
       test_si11, test_si10, test_si9, test_si8, test_si7, test_si6,
       test_si5, test_si4, test_si3, test_si2, test_si1, test_se;
  output rac_req_p2s_ack, rac_rec_p2s_ack, msg_p2s_ack, lp_valid, test_so16,
```

```
    test_so15, test_so14, test_so13, test_so12, test_so11, test_so10,
    test_so9, test_so8, test_so7, test_so6, test_so5, test_so4, test_so3,
    test_so2, test_so1;
wire   nm_req, cmpl_req, \tx_top/counter[1] , \tx_top/counter[0] ,
    \inf_crd_manager/lp_valid_prev , \tx_top/tx_control/n18 ,
    \tx_top/nm_ff/count[2] , \tx_top/nm_ff/count[1] ,
    \tx_top/nm_ff/count[0] , \tx_top/nm_ff/rd_ptr[4] ,
    \tx_top/nm_ff/rd_ptr[3] , \tx_top/nm_ff/rd_ptr[2] ,
    \tx_top/nm_ff/rd_ptr[1] , \tx_top/nm_ff/rd_ptr[0] ,
    \tx_top/nm_ff/fifo[65] , \tx_top/nm_ff/fifo[64] ,
    \tx_top/nm_ff/fifo[63] , \tx_top/nm_ff/fifo[62] ,
    \tx_top/nm_ff/fifo[61] , \tx_top/nm_ff/fifo[60] ,
    \tx_top/nm_ff/fifo[59] , \tx_top/nm_ff/fifo[58] ,
    \tx_top/nm_ff/fifo[57] , \tx_top/nm_ff/fifo[56] ,
    \tx_top/nm_ff/fifo[55] , \tx_top/nm_ff/fifo[54] ,
    \tx_top/nm_ff/fifo[53] , \tx_top/nm_ff/fifo[52] ,
    \tx_top/nm_ff/fifo[51] , \tx_top/nm_ff/fifo[50] ,
    \tx_top/nm_ff/fifo[49] , \tx_top/nm_ff/fifo[48] ,
    \tx_top/nm_ff/fifo[47] , \tx_top/nm_ff/fifo[46] ,
    \tx_top/nm_ff/fifo[45] , \tx_top/nm_ff/fifo[44] ,
    \tx_top/nm_ff/fifo[43] , \tx_top/nm_ff/fifo[42] ,
    \tx_top/nm_ff/fifo[41] , \tx_top/nm_ff/fifo[40] ,
    \tx_top/nm_ff/fifo[39] , \tx_top/nm_ff/fifo[38] ,
    \tx_top/nm_ff/fifo[37] , \tx_top/nm_ff/fifo[36] ,
    \tx_top/nm_ff/fifo[35] , \tx_top/nm_ff/fifo[34] ,
    \tx_top/nm_ff/fifo[33] , \tx_top/nm_ff/fifo[32] ,
    \tx_top/nm_ff/fifo[31] , \tx_top/nm_ff/fifo[30] ,
    \tx_top/nm_ff/fifo[29] , \tx_top/nm_ff/fifo[28] ,
    \tx_top/nm_ff/fifo[27] , \tx_top/nm_ff/fifo[26] ,
    \tx_top/nm_ff/fifo[25] , \tx_top/nm_ff/fifo[24] ,
    \tx_top/nm_ff/fifo[23] , \tx_top/nm_ff/fifo[22] ,
    \tx_top/nm_ff/fifo[21] , \tx_top/nm_ff/fifo[20] ,
    \tx_top/nm_ff/fifo[19] , \tx_top/nm_ff/fifo[18] ,
    \tx_top/nm_ff/fifo[17] , \tx_top/nm_ff/fifo[16] ,
    \tx_top/nm_ff/fifo[15] , \tx_top/nm_ff/fifo[14] ,
    \tx_top/nm_ff/fifo[13] , \tx_top/nm_ff/fifo[12] ,
```

\tx_top/nm_ff/fifo[11] , \tx_top/nm_ff/fifo[10] ,
\tx_top/nm_ff/fifo[9] , \tx_top/nm_ff/fifo[8] ,
\tx_top/nm_ff/fifo[7] , \tx_top/nm_ff/fifo[6] ,
\tx_top/nm_ff/fifo[5] , \tx_top/nm_ff/fifo[4] ,
\tx_top/nm_ff/fifo[3] , \tx_top/nm_ff/fifo[2] ,
\tx_top/nm_ff/fifo[1] , \tx_top/nm_ff/fifo[0] ,
\tx_top/nm_ff/wr_ptr[4] , \tx_top/nm_ff/wr_ptr[3] ,
\tx_top/nm_ff/wr_ptr[2] , \tx_top/nm_ff/wr_ptr[1] ,
\tx_top/nm_ff/wr_ptr[0] , \tx_top/cmpl_ff/count[1] ,
\tx_top/cmpl_ff/count[0] , \tx_top/cmpl_ff/rd_ptr[2] ,
\tx_top/cmpl_ff/rd_ptr[1] , \tx_top/cmpl_ff/rd_ptr[0] ,
\tx_top/cmpl_ff/fifo[65] , \tx_top/cmpl_ff/fifo[64] ,
\tx_top/cmpl_ff/fifo[63] , \tx_top/cmpl_ff/fifo[62] ,
\tx_top/cmpl_ff/fifo[61] , \tx_top/cmpl_ff/fifo[60] ,
\tx_top/cmpl_ff/fifo[59] , \tx_top/cmpl_ff/fifo[58] ,
\tx_top/cmpl_ff/fifo[57] , \tx_top/cmpl_ff/fifo[56] ,
\tx_top/cmpl_ff/fifo[55] , \tx_top/cmpl_ff/fifo[54] ,
\tx_top/cmpl_ff/fifo[53] , \tx_top/cmpl_ff/fifo[52] ,
\tx_top/cmpl_ff/fifo[51] , \tx_top/cmpl_ff/fifo[50] ,
\tx_top/cmpl_ff/fifo[49] , \tx_top/cmpl_ff/fifo[48] ,
\tx_top/cmpl_ff/fifo[47] , \tx_top/cmpl_ff/fifo[46] ,
\tx_top/cmpl_ff/fifo[45] , \tx_top/cmpl_ff/fifo[44] ,
\tx_top/cmpl_ff/fifo[43] , \tx_top/cmpl_ff/fifo[42] ,
\tx_top/cmpl_ff/fifo[41] , \tx_top/cmpl_ff/fifo[40] ,
\tx_top/cmpl_ff/fifo[39] , \tx_top/cmpl_ff/fifo[38] ,
\tx_top/cmpl_ff/fifo[37] , \tx_top/cmpl_ff/fifo[36] ,
\tx_top/cmpl_ff/fifo[35] , \tx_top/cmpl_ff/fifo[34] ,
\tx_top/cmpl_ff/fifo[33] , \tx_top/cmpl_ff/fifo[32] ,
\tx_top/cmpl_ff/fifo[31] , \tx_top/cmpl_ff/fifo[30] ,
\tx_top/cmpl_ff/fifo[29] , \tx_top/cmpl_ff/fifo[28] ,
\tx_top/cmpl_ff/fifo[27] , \tx_top/cmpl_ff/fifo[26] ,
\tx_top/cmpl_ff/fifo[25] , \tx_top/cmpl_ff/fifo[24] ,
\tx_top/cmpl_ff/fifo[23] , \tx_top/cmpl_ff/fifo[22] ,
\tx_top/cmpl_ff/fifo[21] , \tx_top/cmpl_ff/fifo[20] ,
\tx_top/cmpl_ff/fifo[19] , \tx_top/cmpl_ff/fifo[18] ,
\tx_top/cmpl_ff/fifo[17] , \tx_top/cmpl_ff/fifo[16] ,

```
        \tx_top/cmpl_ff/fifo[15] , \tx_top/cmpl_ff/fifo[14] ,
        \tx_top/cmpl_ff/fifo[13] , \tx_top/cmpl_ff/fifo[12] ,
        \tx_top/cmpl_ff/fifo[11] , \tx_top/cmpl_ff/fifo[10] ,
        \tx_top/cmpl_ff/fifo[9] , \tx_top/cmpl_ff/fifo[8] ,
        \tx_top/cmpl_ff/fifo[7] , \tx_top/cmpl_ff/fifo[6] ,
        \tx_top/cmpl_ff/fifo[5] , \tx_top/cmpl_ff/fifo[4] ,
        \tx_top/cmpl_ff/fifo[3] , \tx_top/cmpl_ff/fifo[2] ,
        \tx_top/cmpl_ff/fifo[1] , \tx_top/cmpl_ff/fifo[0] ,
        \tx_top/cmpl_ff/wr_ptr[2] , \tx_top/cmpl_ff/wr_ptr[1] ,
        \tx_top/cmpl_ff/wr_ptr[0] , \tx_top/nm_ctl/n4 ,
        \tx_top/nm_ctl/phase_opc_q[1] , \tx_top/nm_ctl/phase_opc_q[0] ,
        \parralel_arbiter/u_parallel_arbiter/grant_idx[1] ,
        \parralel_arbiter/u_parallel_arbiter/grant_idx[0] ,
        \p2s_arbiter/u__tmp462/n6 , \tx_top/cmpl_ctl/n9 ,
        \tx_top/cmpl_ctl/n5 , \tx_top/cmpl_ctl/n4 ,
        \tx_top/cmpl_ctl/phase_opc_q[1] , \tx_top/cmpl_ctl/phase_opc_q[0] ,
HDBLVT14_AN2_1 U728 ( .A1(\rac_req_inst/top_remote_access_requester_1/n3 ),
    .A2(n457), .X(n513) );
HDBLVT14_AN2_1 U729 ( .A1(rac_req_p2s_data[28]), .A2(n457), .X(n514) );
HDBLVT14_AN2_1 U730 ( .A1(rac_req_p2s_data[15]), .A2(n457), .X(n515) );
HDBLVT14_AN2_1 U731 ( .A1(rac_req_p2s_data[14]), .A2(n457), .X(n516) );
HDBLVT14_AN2_1 U732 ( .A1(rac_req_p2s_data[13]), .A2(n457), .X(n517) );
HDBLVT14_AN2_1 U733 ( .A1(rac_req_p2s_data[12]), .A2(n457), .X(n518) );
HDBLVT14_AN2_1 U734 ( .A1(rac_req_p2s_data[10]), .A2(n457), .X(n519) );
HDBLVT14_ND2_1 U735 ( .A1(rac_req_p2s_data[0]), .A2(n457), .X(n448) );
HDBLVT14_ND2_1 U736 ( .A1(mb_req_we_qst), .A2(n449), .X(n458) );
HDBLVT14_ND2_1 U737 ( .A1(rac_req_p2s_data[1]), .A2(n457), .X(n451) );
HDBSVT14_INV_0P5 U738 ( .A(mb_req_we_qst), .X(n450) );
HDBLVT14_ND2_1 U739 ( .A1(n450), .A2(n449), .X(n455) );
HDBLVT14_ND2_1 U740 ( .A1(rac_req_p2s_data[11]), .A2(n457), .X(n452) );
HDBLVT14_ND2_1 U741 ( .A1(rac_req_p2s_data[29]), .A2(n457), .X(n454) );
HDBLVT14_ND2_1 U742 ( .A1(rac_req_p2s_data[64]), .A2(n457), .X(n456) );
HDBLVT14_ND2_1 U743 ( .A1(rac_req_p2s_data[65]), .A2(n457), .X(n459) );
HDBLVT14_LDNQ_2 LOCKUP ( .D(n534), .G(sclk), .Q(scan_out[16]) );
HDBLVT14_TIE1_4 U321 ( .X(n236) );
HDBLVT14_TIE0_4 U322 ( .X(n238) );
HDBSVT14_ND2_CDC_0P5 U326 ( .A1(n738), .A2(n292), .X(n728) );
```

```
  HDBLVT14_INV_16 U335 ( .A(init), .X(n729) );
  HDBLVT14_INV_16 U346 ( .A(init), .X(n730) );
  HDBLVT14_BUF_S_20 U383 ( .A(n736), .X(n731) );
  HDBLVT14_OR2_4 U500 ( .A1(n736), .A2(n747), .X(n732) );
  HDBSVT14_ND2_CDC_0P5 U540 ( .A1(n266), .A2(n306), .X(n733) );
  HDBSVT14_OAI21_1 U545 ( .A1(n746), .A2(n292), .B(n745), .X(n734) );
  HDBLVT14_AN2_4 U548 ( .A1(n306), .A2(n253), .X(n735) );
  HDBLVT14_AN2_4 U561 ( .A1(n306), .A2(n253), .X(n736) );
  HDBLVT14_AO22_1 U563 ( .A1(n739), .A2(n268), .B1(n737), .B2(n276), .X(n279)
       );
  HDBLVT14_INV_16 U564 ( .A(n738), .X(n737) );
  HDBLVT14_NR2_MM_16 U579 ( .A1(n735), .A2(n747), .X(n738) );
  HDBSVT14_AN2_0P5 U744 ( .A1(n745), .A2(n733), .X(n739) );
  HDBLVT14_OR2_4 U745 ( .A1(n731), .A2(n740), .X(n304) );
  HDBLVT14_OR2_4 U746 ( .A1(n288), .A2(n741), .X(n740) );
  HDBLVT14_INV_16 U747 ( .A(n300), .X(n741) );
  HDBSVT14_OAI21_1 U748 ( .A1(n746), .A2(n292), .B(n745), .X(n742) );
  HDBLVT14_AN2_4 U749 ( .A1(n730), .A2(n749), .X(n743) );
  HDBLVT14_AN2_4 U750 ( .A1(n729), .A2(n749), .X(n744) );
  HDBLVT14_ND2_MM_16 U751 ( .A1(n743), .A2(n253), .X(n745) );
  HDBLVT14_OR2_4 U752 ( .A1(n736), .A2(n747), .X(n746) );
  HDBLVT14_INV_16 U753 ( .A(n744), .X(n747) );
  HDBLVT14_AN2_4 U754 ( .A1(n748), .A2(n749), .X(n306) );
  HDBLVT14_INV_16 U755 ( .A(init), .X(n748) );
  HDBLVT14_INV_16 U756 ( .A(\rac_req_inst/timer_init_syn ), .X(n749) );
  HDBLVT14_OR2_4 U757 ( .A1(f_err), .A2(n750), .X(n751) );
  HDBLVT14_INV_16 U758 ( .A(n752), .X(n750) );
  HDBLVT14_NR2_MM_4 U759 ( .A1(n239), .A2(n751), .X(n241) );
  HDBLVT14_INV_16 U760 ( .A(n246), .X(n752) );
endmodule
```