

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN ĐIỆN TỬ
BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG



BẢN TÓM TẮT

THIẾT KẾ VÀ ĐÁNH GIÁ BỘ CHUYỂN ĐỔI GIAO THỨC APB SANG UART

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG

Sinh viên: **ĐINH ĐỨC ANH**
MSSV: 21161391

Môn học: Thiết kế hệ thống và vi mạch tích hợp

TP. HỒ CHÍ MINH – 05/2023

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH
KHOA ĐIỆN ĐIỆN TỬ
BỘ MÔN KỸ THUẬT MÁY TÍNH - VIỄN THÔNG

BẢN TÓM TẮT

THIẾT KẾ VÀ ĐÁNH GIÁ BỘ CHUYỂN ĐỔI GIAO THÚC APB SANG UART

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ - VIỄN THÔNG

Sinh viên: **ĐINH ĐỨC ANH**
MSSV: 21161391

Hướng dẫn: TS. **ĐỖ DUY TÂN**

Môn học: Thiết kế hệ thống và vi mạch tích hợp

TP. HỒ CHÍ MINH – 05/2023

LỜI CẢM ƠN

Sau khi hoàn thành xong đề tài, lời cảm ơn đầu tiên em xin gửi tới thầy Đỗ Duy Tân. Trong quá trình gặp mặt, báo cáo cũng như thảo luận để hoàn thành đề tài, thầy đã giúp em rất nhiều trong nhiều việc khác nhau, từ xây dựng nêu ý tưởng đề tài, tìm kiếm các nguồn tài liệu đến giúp đỡ trong cả quá trình thiết kế, đánh giá để hoàn thiện đề tài. Nhờ có thầy mà em đã có thể hoàn thành đề tài một cách hoàn chỉnh.

Kế tiếp, em xin cảm ơn đến các thầy cô trong khoa đã giảng dạy, giúp em có đầy đủ kiến thức cho việc làm đề tài lần này. Nhờ những kiến thức từ các môn cơ sở ngành cho đến cả môn chuyên ngành mà em mới có thể tận dụng để tiến hành thực hiện đề tài của mình. Đây là một điều quý giá mà em đã nhận được trong suốt thời gian học tập tại ngôi trường này.

Em cũng xin cảm ơn phía bộ môn Kỹ thuật máy tính – viễn thông đã tạo điều kiện cho việc thực hiện đề tài, thông báo lịch và hướng dẫn rõ hình thức nộp đề tài. Nhờ đề tài lần này, em đã có thể tận dụng những kiến thức đã học, tìm hiểu thêm các kiến thức mới, nâng cao khả năng sáng tạo của mình.

Xin trân trọng cảm ơn.

TÓM TẮT

Chuyển đổi qua lại giữa hai giao thức luôn là vấn đề được quan tâm trong quá trình thiết kế các hệ thống SOC (System on chip). Có những giao thức hoạt động với tốc độ xử lý rất nhanh và cũng có những giao thức có tốc độ xử lý chậm hơn nhiều so với xung của hệ thống bên trong.

Đề tài này hướng tới việc nghiên cứu, thiết kế và xây dựng nên bộ chuyển đổi hoàn chỉnh giữa bus APB và giao thức UART, cả hai giao thức này đều phổ biến trong quá trình thiết kế chip của ARM hoặc nơi khác. APB là bus thích hợp làm trung gian giữa bus hệ thống với các ngoại vi có tốc độ xử lý thấp với cấu trúc đơn giản. UART là một giao thức phổ biến trên mọi thiết bị hiện nay, hoạt động trên nhiều tần số khác nhau.

Bộ chuyển đổi hướng tới việc chuyển đổi hai giao thức APB 4 sang UART để dữ liệu có thể truyền qua lại giữa hai phía. Bộ chuyển đổi này cũng bao gồm cả các tín hiệu ngắt thông báo cho phía CPU xử lý, các thanh ghi có thể cấu hình được.

Việc đánh giá bộ chuyển đổi thông qua kết quả của từng trường hợp mô phỏng nhằm đảm bảo các tính năng của bộ chuyển đổi hoạt động chính xác.

CÁC TỪ VIẾT TẮT

| | |
|------|---|
| AMBA | Advanced High-performance Bus |
| APB | Advanced Peripheral Bus |
| ARM | Advanced RISC Machines |
| FIFO | First in first out |
| IC | Integrated circuit |
| IP | Intellectual Property |
| SOC | System on chip |
| UART | Universal Asynchronous Receiver-Transmitter |

CHƯƠNG 1 TỔNG QUAN ĐỀ TÀI

1.1 GIỚI THIỆU

System on chip hay SoC là một dạng mạch tích hợp xuất hiện phổ biến trên nhiều thiết bị hiện nay. Đây là một hệ thống bao gồm các thành phần bao gồm bộ xử lý trung tâm (CPU), bộ nhớ, các bộ giao tiếp ngoại vi. Các thành phần này được tích hợp trên một con chip và kết nối với nhau thông qua các giao thức và một hệ thống bus. Với các đặc trưng như trên, các nhà nghiên cứu đã xây dựng nên các kiến trúc tiêu chuẩn để phục vụ cho việc sản xuất cũng như phát triển các kiến trúc đó.

AMBA hay Advanced Microcontroller Bus Architecture là một trong những kiến trúc bus phổ biến hiện nay của công ty ARM. AMBA có các giao thức phục vụ cho các kết nối khác nhau như: AHB, ASB trong các bus hệ thống, APB trong các bus ngoại vi. Các giao thức này được kết nối thông qua các lõi IP phục vụ việc chuyển đổi qua loại giữa hai giao tiếp, điều này cũng tương tự với việc kết nối một ngoại vi với các thành phần bên trong.

APB là một giao thức được chuyên dùng trong việc giao tiếp với ngoại vi và các giao thức bên trong hệ thống. Đây là một giao thức không cần tốc độ xử lý nhanh, băng thông thấp, thích hợp để giao tiếp với các giao thức như I2C, SPI, UART. Qua nhiều năm, AMBA đã được phát triển lên nhiều phiên bản theo thời điểm hiện tại, từ đó APB cũng được phát triển. APB4 là một phiên bản đang được chú ý đến hiện nay, bus này được thêm 2 đường tín hiệu so với APB3, từ đó thêm các tính năng trong quá trình giao tiếp. Tuy nhiên, chính điều trên cũng gây ra vấn đề về việc xây dựng lõi IP cho việc phục vụ giao tiếp giữa APB với các ngoại vi có tốc độ xử lý thấp như UART.

1.2 MỤC TIÊU ĐỀ TÀI

Mục tiêu của đề tài hướng tới thực hiện hoàn chỉnh việc “Thiết kế và đánh giá bộ chuyển đổi giao thức APB sang UART”, bao gồm 2 nhiệm vụ chính sau:

- Thiết kế hoàn chỉnh một bộ chuyển đổi giao thức APB sang UART.

- Kiểm tra và đánh giá kết quả kiểm thử đạt được từ bộ chuyển đổi giao thức thông qua các trường hợp mô phỏng.

1.3 PHẠM VI GIỚI HẠN CỦA ĐỀ TÀI

Đề tài được thực hiện bằng việc xây dựng bộ chuyển đổi bằng ngôn ngữ mô tả phần cứng Verilog. Các file được tổng hợp trên phần mềm Xilinx ISE 17.7.

Bộ chuyển đổi được đánh giá thông qua các kết quả đạt được theo từng trường hợp kiểm tra (testcase) được quy định sẵn và đưa vào môi trường mô phỏng, công cụ mô phỏng ở đây là ISIM.

1.4 PHƯƠNG PHÁP NGHIÊN CỨU

Đề tài sử dụng các phương pháp nghiên cứu như sau:

- Phương pháp phân tích và tổng hợp.
- Phương pháp phân loại và hệ thống.
- Phương pháp mô phỏng.
- Phương pháp khảo sát hệ thống thực tế.

1.5 ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Đối tượng nghiên cứu của đề tài này là bộ chuyển đổi giao thức APB sang UART.

Phạm vi nghiên cứu đánh giá các thành phần của bộ chuyển đổi như khối cổng APB, khối truyền nhận UART, bộ tạo tốc độ truyền, các tín hiệu ngắn, hiệu năng của bộ chuyển đổi.

1.6 BỘ CỤC ĐỀ TÀI

Bộ cục của quyền khóa luận gồm các chương:

- Chương 1. TỔNG QUAN ĐỀ TÀI.

Giới thiệu tổng quan về nội dung, mục tiêu, giới hạn, đối tượng và phạm vi nghiên cứu của đề tài.

- Chương 2. CƠ SỞ LÝ THUYẾT.

Gồm các lý thuyết liên quan đến giao thức, môi trường mô phỏng của bộ chuyển đổi

- Chương 3. THIẾT KẾ HỆ THỐNG.

Bao gồm yêu cầu, sơ đồ thiết kế của hệ thống, lưu đồ hoạt động.

- Chương 4 KẾT QUẢ.

Kết quả mô phỏng của bộ chuyển đổi, đánh giá kết quả đạt được

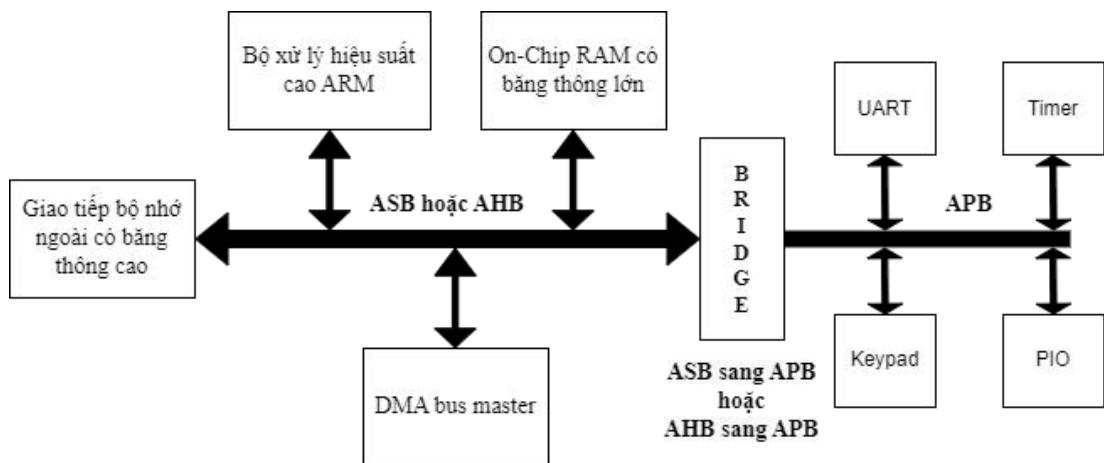
- Chương 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.

Đưa ra kết luận về việc thực hiện đề tài, nhận xét các kết quả đạt được, đưa ra những phương hướng phát triển khác cho bộ chuyển đổi.

CHƯƠNG 2 CƠ SỞ LÝ THUYẾT

2.1 TỔNG QUAN VỀ AMBA VÀ CHUẨN BUS APB

AMBA hay Advanced High-performance Bus là hệ thống đường truyền được phát triển bởi ARM Limited. AMBA là một tiêu chuẩn mở dùng để kết nối và quản lý các khối chức năng trong thiết kế của một System on Chip. AMBA bao gồm nhiều chuẩn con khác nhau theo từng tên gọi cũng như mục đích sử dụng [1]. Hình 2.1 thể hiện một sơ đồ khái niệm xây dựng theo AMBA:



Các chuẩn như ASB hoặc AHB được sử dụng trong các giao tiếp yêu cầu hiệu suất, băng thông cao, trong khi đó APB được sử dụng trong các giao tiếp với ngoại có tốc độ xử lý thấp như UART [1]. Đây là một đặc trưng của kiến trúc AMBA, Tại giữa hai chuẩn có tốc độ xử lý khác nhau có một cầu chuyển tiếp (Bridge), cầu này đóng vai trò chuyển đổi tín hiệu giữa hai chuẩn, từ đó đồng bộ với nhau trong quá trình hoạt động.

AMBA tạo điều kiện thuận lợi cho việc phát triển các thiết kế có đa bộ xử lý với nhiều bộ điều khiển và các thành phần với kiến trúc bus. Nguyên tắc thiết kế bus AMBA gồm các tiêu chí như sau [1, 2, 3]:

- Tạo điều kiện thuận lợi cho việc phát triển ngay lần đầu tiên các vi điều khiển nhúng sử dụng một hoặc nhiều CPU, GPU hay bộ xử lý tín hiệu.
- Độc lập về công nghệ, cho phép tái sử dụng lõi IP và hệ thống trên các quy trình IC đa dạng,

- Khuyến khích thiết kế hệ thống mô-đun để cải thiện tính độc lập của bộ xử lý và phát triển các thư viện IP hệ thống và ngoại vi có thể tái sử dụng
- Giảm thiểu cơ sở hạ tầng silicon đồng thời hỗ trợ giao tiếp trên chip hiệu suất cao và tiêu thụ điện năng thấp.

Qua từng các năm, AMBA đã được phát triển thêm nhiều các phiên bản khác để phục vụ cho yêu cầu ngày càng cao về việc thiết kế chip. AMBA có tổng cộng 5 phiên bản tính tới hiện tại bao gồm: AMBA, AMBA 1, AMBA 2, AMBA 3, AMBA 4 và AMBA5

2.2 TỔNG QUAN VỀ BUS APB

2.2.1 Giới thiệu về bus APB

APB (Advanced Peripheral Bus) dịch theo tiếng Việt là bus ngoại vi nâng cao là một phần của họ giao thức kiến trúc bus vi điều khiển nâng cao (AMBA). APB xác định một giao diện được tối ưu hóa để tiêu thụ điện năng ở mức tối thiểu và giảm độ phức tạp của giao diện [4].

APB hoạt động theo mô hình Master - Slave, trong đó slave thường là cầu nối giữa các tín hiệu bên trong với thiết bị ngoại vi. APB không có ống dẫn, đơn giản, thích hợp để giao tiếp với các ngoại vi có tốc độ xử lý và băng thông tương đối thấp so với tốc độ xử lý bên trong.

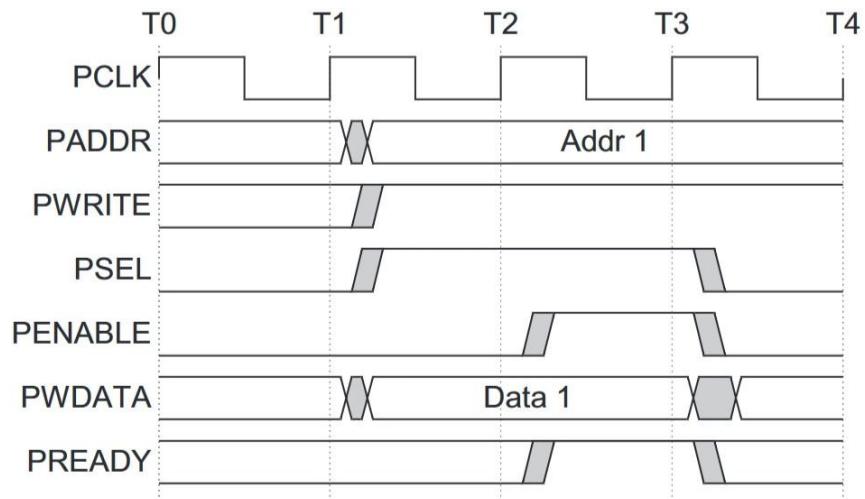
Qua các năm, APB đã được phát triển thêm nhiều phiên bản mới hơn lần lượt là: AMBA APB, AMBA 2 APB, AMBA 3 APB, AMBA 4 APB và AMBA 5 APB. Trong đó, các tên gọi thường được gọi tắt lại, ví dụ như AMBA 4 APB thành APB4.

2.2.2 Quá trình ghi dữ liệu của bus APB4

Việc ghi dữ liệu của bus APB4 được chia làm 2 kiểu là: Quá trình ghi không có trạng thái chờ và quá trình ghi có trạng thái chờ.

- Quá trình ghi không có trạng thái chờ:

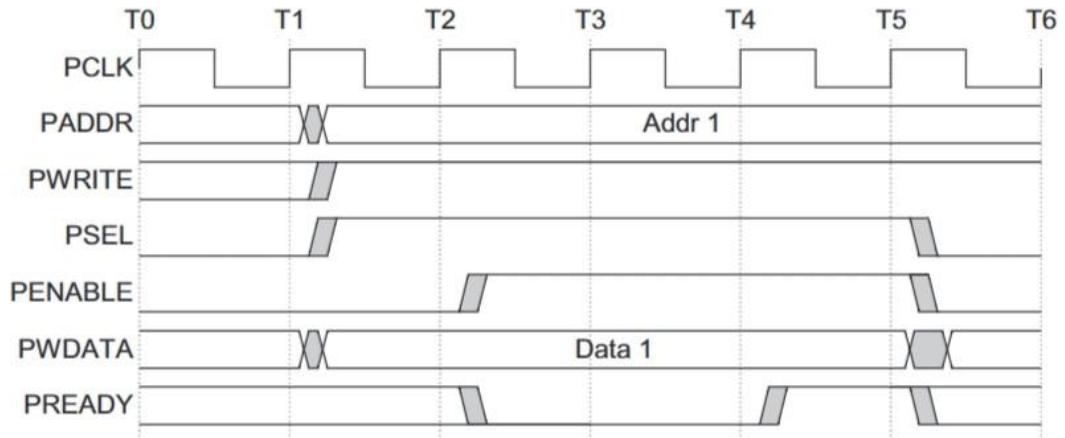
Quá trình ghi không có trạng thái chờ được mô tả theo hình 2.2 [4]:



Hình 2.2. Quá trình ghi không có trạng thái chờ

Quá trình ghi có trạng thái chờ:

Quá trình ghi có trạng thái chờ được mô tả như hình 2.3 [4]:



Hình 2.3. Quá trình ghi có trạng thái chờ

Tại quá trình ghi có trạng thái chờ, PREADY được kéo lên mức cao ở trạng thái không có quá trình truyền.

Sau khi xác nhận được mức cao của PSEL tại cạnh lên xung clock T1, PREADY được kéo về mức thấp tại cạnh lên của chu kỳ T2 và duy trì 2 chu kỳ xung. APB master xác nhận mức thấp của PREADY trong quá trình này nên tiếp tục quá trình giao tiếp.

Tại cạnh lên của chu kỳ T4, PREADY được kéo lên mức cao, chuẩn bị kết thúc quá trình truyền ở chu kỳ kế tiếp.

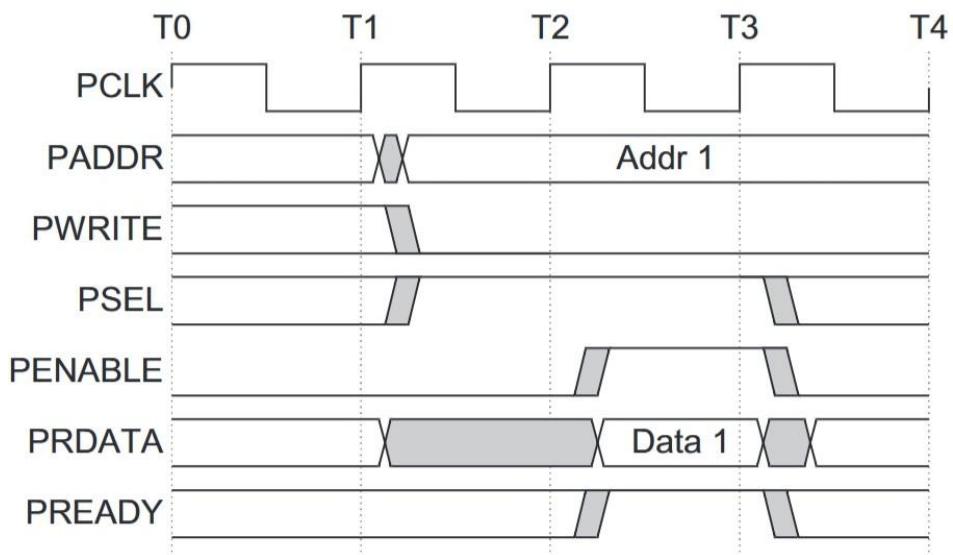
Tại cạnh lên của chu kỳ T5, khi PREADY được xác nhận, APB master tiến hành kéo PENABLE về mức thấp, PSEL có thể giữ nguyên để chuẩn bị cho chu kỳ kế tiếp.

2.2.3 Quá trình đọc dữ liệu của bus APB4

Giống như quá trình ghi, quá trình đọc dữ liệu của bus APB cũng được chia làm hai kiểu là: đọc không có trạng thái chờ và đọc có trạng thái chờ.

- Quá trình đọc không có trạng thái chờ:

Quá trình đọc không có trạng thái chờ được mô tả như hình 2.4 [4]:

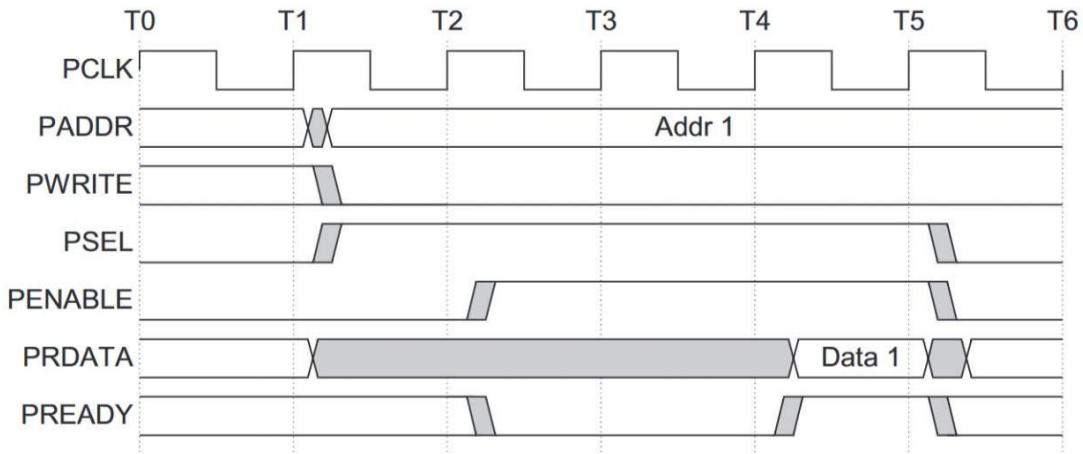


Hình 2.4. Quá trình đọc không có trạng thái chờ

Khác với quá trình ghi, tín hiệu PWRITE được kéo xuống mức thấp trong suốt quá trình truyền. Các tín hiệu còn lại đều hoạt động tương tự như quá trình ghi.

- Quá trình đọc có trạng thái chờ:

Quá trình đọc có trạng thái chờ được mô tả như hình 2.5 [4]:



Hình 2.5. Quá trình đọc có trạng thái chờ

Ở quá trình này, các tín hiệu cũng hoạt động tương tự như quá trình đọc không có trạng thái chờ. Riêng tín hiệu PREADY được phía APB slave duy trì khi không có quá trình truyền và kéo xuống mức thấp trong 2 chu kỳ sau khi xác nhận được mức cao ở tín hiệu PSEL nhằm kéo dài quá trình truyền.

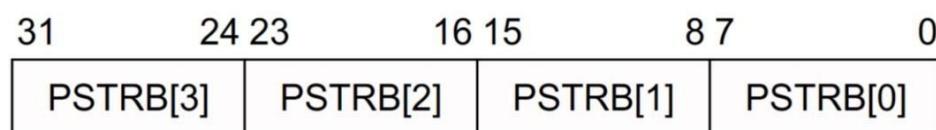
2.2.4 Chi tiết một vài tín hiệu của bus APB

Một vài tín hiệu mới có trong APB3 và APB4 được giới thiệu trong phần này.

- Tín hiệu PSTRB:

Tín hiệu PSTRB hay write strobe có thể quyết định byte hợp lệ của đường dữ liệu trong một quá trình ghi dữ liệu. Bằng cách này, APB master có thể quyết định byte nào của phẳng ghi bên APB slave có được cập nhật giá trị trong quá trình ghi hay không [4].

Một bit của PSTRB đại diện cho 8 bit hay 1 byte đường dữ liệu. Như vậy với PWDATA[(8n + 7):(8n)] thì có PSTRB[n] tương ứng.



Hình 2.6. Kết nối của PSTRB

Theo hình 2.6, với một đường dữ liệu 32 bit, có 4 bit PSTRB tương ứng. Trong đó, PSTRB[0] ứng với PWATA[7:0], PSTRB[1] ứng với PWATA[15:8],... cứ như vậy với hai bit còn lại.

- Tín hiệu PSLVERR:

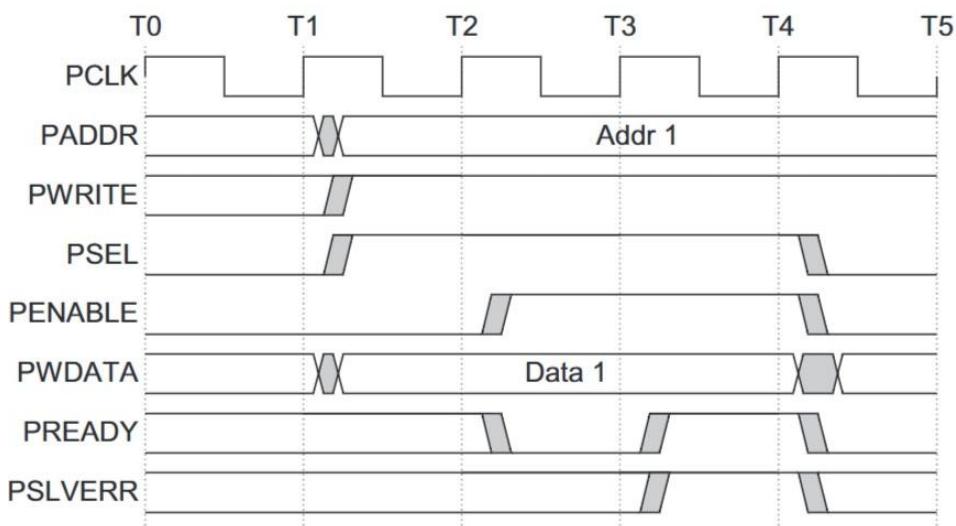
Tín hiệu PSLVERR được dùng để báo một quá trình truyền bị lỗi. Điều kiện lỗi có thể xảy ra ở cả quá trình đọc và ghi [4].

PSLVERR được phía APB slave kéo nêu nếu các điều kiện lỗi được đáp ứng. Giá trị PSLVERR chỉ hợp lệ ở chu kỳ cuối cùng của một quá trình giao tiếp, khi mà cả PENABLE và PREADY đều ở mức cao. Thông thường PSLVERR ở mức thấp và được đẩy lên mức cao nếu có lỗi xảy ra.

Các điều kiện lỗi thông thường liên quan đến việc các thanh ghi ở các thiết bị ngoại vi không được cập nhật dữ liệu trong quá trình ghi hay các dữ liệu trả về không hợp lệ trong quá trình đọc. Điều này xảy ra chủ yếu do địa chỉ đưa tới không đúng hoặc tín hiệu PSTRB không hợp lệ.

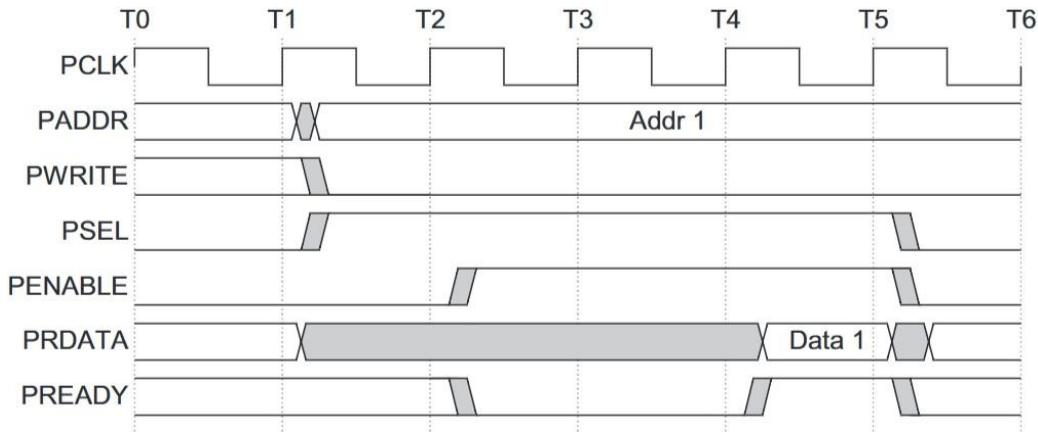
Tuy nhiên, PSLVERR không bắt buộc cho các thiết bị ngoại vi. Trong trường hợp các thiết bị ngoại vi không có tín hiệu báo lỗi, một đầu vào mức thấp được gắn vào ở phía APB master.

Hoạt động của PSLVERR được mô tả theo hai hình 2.7:



Hình 2.7. Quá trình ghi bị lỗi

Theo hình 2.7, PSLVERR kéo lên mức cao ở cạnh lên chu kỳ T3.



Hình 2.8. Quá trình đọc bị lỗi

Theo hình 2.8, PSLVERR được kéo lên mức cao ở chu kỳ T4.

- Tín hiệu PPROT:

Với các hệ thống phức tạp ngày nay, các thiết bị kết nối trong hệ thống thường cần có sự bảo vệ trước các giao dịch không hợp lệ. Trong bus APB, tín hiệu PPROT được dùng để thiết lập các cấp độ bảo vệ quyền truy cập. Tín hiệu này gồm 3 bit và được quy định như sau [4]:

PPROT[0]: Mức thấp thiết lập truy cập bình thường và mức cao thiết lập truy cập đặc quyền.

PPROT[1]: Mức thấp thiết lập truy cập bảo mật và mức cao thiết lập truy cập không bảo mật.

PPROT[2]: Mức thấp thiết lập truy cập dữ liệu và mức cao thiết lập truy cập lệnh.

2.3 TỔNG QUAN VỀ GIAO THỨC UART

2.3.1 Giới thiệu về giao thức UART

UART (Universal Asynchronous Receiver-Transmitter – Bộ truyền nhận dữ liệu không đồng bộ) là một giao thức dùng để giao tiếp không đồng bộ, trong đó định dạng dữ liệu và tốc độ truyền có thể cấu hình được [5].

UART thường là một phần của mạch tích hợp (IC) được sử dụng trong các truyền thông nối tiếp qua các cổng nối tiếp của máy tính hoặc các thiết bị ngoại vi. Một chip vi điều khiển thường tích hợp một hoặc nhiều thiết bị ngoại vi UART.

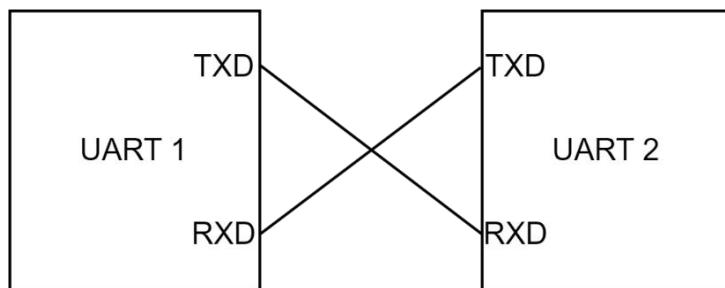
Một bộ UART thường chứa các thành phần sau:

- Một bộ tạo xung nhịp, được tính toán để có thể lấy mẫu dữ liệu ở giữa chu kỳ bit
- Hai thanh ghi dịch ở phía truyền và phía nhận, đi kèm với đó là hai bộ đệm hoặc FIDO truyền/nhận.
- Bộ điều khiển truyền nhận.
- Bộ logic điều khiển quá trình ghi/đọc.

UART có thể hoạt động ở 3 chế độ là: simplex (Truyền một chiều), song công (Cả hai thiết bị có thể truyền nhận cùng lúc) và bán song công (hai thiết bị thay phiên nhau truyền nhận).

Để 2 UART có thể giao tiếp truyền/nhận với nhau, cả hai bên đều cần phải thiết lập các thông số sau đây giống nhau: Mức điện áp, tốc độ Baud rate, chế độ Parity, độ lớn bit dữ liệu, độ lớn stop bit, kiểm soát lưu lượng.

Thông thường, một bộ UART có 2 chân là TXD và RXD, trong đó chân TXD đóng vai trò là ngõ ra của bộ truyền (Transmitter) và RXD đóng vai trò là ngõ vào của bộ nhận (Receiver). Hai bộ UART kết nối với nhau được mô tả như hình 2.9:



Hình 2.9. Kết nối của 2 bộ UART

2.3.2 Khung dữ liệu của UART

Một khung dữ liệu của UART bao gồm 5 thành phần sau:

- Start bit: là bit 0 báo hiệu cho phía bộ nhận biết rằng một dữ liệu đang đến.
- Data bits: bits dữ liệu, thường được thiết lập khoảng 5 tới 9 bit.
- Parity bit: bit chẵn lẻ dùng để xác định dữ liệu có bị lỗi trong quá trình truyền hay không.
- Stop bit: thường khoảng là 1 tới 2 bit 1, báo hiệu kết thúc khung dữ liệu.

| | | | | | | | | | | |
|-----------|----|----|----|----|----|----|----|----|------------|----------|
| Start bit | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 | Parity bit | Stop bit |
|-----------|----|----|----|----|----|----|----|----|------------|----------|

Hình 2.10. Khung dữ liệu UART có 8 bit dữ liệu [5]

Ở trạng thái bình thường, khi không có quá trình giao tiếp, bộ truyền giữ ngõ ra ở mức cao. Nếu như có một dữ liệu cần truyền đi, bộ truyền tiến hành truyền nối tiếp các bit dữ liệu theo thứ tự là start bit, data bits, parity bit (nếu có) và stop bits, trong đó data bits được quy chuẩn là truyền từ LSB đến MSB.

Thông thường, các bộ UART thường truyền dữ liệu có độ lớn là 8 bit, và thường không có bit parity để hiệu suất giao tiếp.

2.3.3 Bộ truyền và nhận của UART

Bộ truyền và bộ nhận là hai thành phần quan trọng trong quá trình truyền nhận của UART

- Bộ truyền UART:

Bộ truyền UART sử dụng xung riêng chạy ở bội số của tốc độ dữ liệu [5]. Bộ truyền hoạt động khá đơn giản khi không cần phải xác nhận trạng thái đường truyền. Ngay khi hệ thống gửi một dữ liệu vào thanh ghi đệm, nếu như đang trong trạng thái không truyền dữ liệu nào, bộ truyền tiến hành truyền lần lượt các ký tự theo khung dữ liệu đã định sẵn, với start bit là mức thấp để báo hiệu cho phía bên kia chuẩn bị vào trạng thái nhận dữ liệu và stop bit là mức thấp báo hiệu kết thúc khung dữ liệu.

Do sự khác nhau lớn giữa tốc độ xử lý của CPU phía bên trong so với tốc độ truyền của UART, các bộ FIFO thường được thay thế để phía CPU có thể ghi nhiều dữ liệu trước khi được truyền đi thay vì phải đợi lần lượt từng dữ liệu được truyền hết mới bắt đầu tiến hành ghi.

Bộ truyền thông thường còn được thiết kế thêm một tín hiệu báo trạng thái bận, tức đang truyền một dữ liệu nào đó để phía CPU nhận biết. Ngoài ra, tín hiệu báo trạng thái rảnh cũng có thể được thiết kế thêm như một tín hiệu ngắn báo hiệu cho phía CPU tiến hành ghi dữ liệu vào đây.

- Bộ nhận UART:

Bộ nhận UART cũng sử dụng một xung riêng giống như bộ truyền [5]. Tuy nhiên, do yêu cầu hoạt động nên bộ nhận có thiết kế phức tạp hơn. Bộ nhận liên tục kiểm tra trạng thái của đường truyền có sự thay đổi nào hay không. Trong trường hợp phát hiện đường truyền được kéo xuống mức thấp, bộ nhận tiến hành lấy mẫu đường truyền sau nửa chu kỳ bit. Nếu mẫu ở mức cao, như vậy tín hiệu mức thấp kia là giả, bộ nhận về trạng thái ban đầu, nếu mẫu ở mức thấp, như vậy đây là start bit, bộ nhận vào trạng thái ghi dữ liệu. Các bit dữ liệu được lấy mẫu ở giữa chu kỳ bit và được đưa vào thanh ghi đệm hoặc bộ FIFO. Phía UART cũng đặt báo hiệu cho dữ liệu mới và thường có một tín hiệu ngắt báo cho phía xử lý trung tâm tiến hành chuyển dữ liệu.

Do không có bộ định thời cho UART, bộ nhận thường phải đồng bộ lại với bộ tạo xung bên trong UART sau mỗi lần thay đổi đường truyền nếu không được coi là xung giả.

Tương tự như bộ truyền, bộ nhận có thể sử dụng FIFO để tăng số lượng dữ liệu được ghi trước khi chuyển đi. Bộ nhận cũng có thể thiết kế thêm tín hiệu báo trạng thái bận hoặc tín hiệu báo có dữ liệu mới.

2.3.4 Các lỗi có trong UART

Dưới đây là một số lỗi có thể xuất hiện trong một bộ UART:

Lỗi tràn: Lỗi tràn xảy ra khi phía nhận không thể xử lý dữ liệu mới nhận trước khi dữ liệu kế tiếp được gửi đến. Nguyên nhân của lỗi này đến từ việc bộ nhận của UART thường được xây dựng thêm một bộ đệm có thể chứa nhiều dữ liệu, tuy nhiên, khi các bộ đệm này đầy, phía CPU không xử lý kịp việc chuyển dữ liệu khỏi bộ đệm đi nơi khác, từ đó dẫn đến dữ liệu bị mất đi.

Lỗi chạy ngầm: Lỗi chạy ngầm xảy ra ở bộ truyền sau khi dữ liệu truyền đi hoàn tất và bộ đệm ở phía truyền trống.

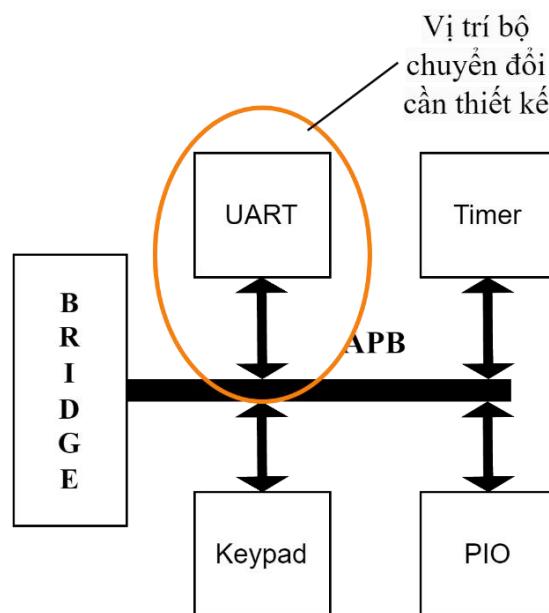
Lỗi đóng khung: Lỗi đóng khung xảy ra khi phía nhận của UART không phát hiện được stop bit. Do UART dựa trên start bit để bắt đầu quá trình nhận nên đến thời điểm lấy mẫu stop bit mong đợi ghi nhận ở trạng thái mức cao, tuy nhiên, nếu như phát hiện trạng thái mức thấp, khung dữ liệu được gửi đến đã bị lỗi.

Lỗi parity: lỗi parity xảy ra khi phía UART tiến hành tính toán tính chẵn lẻ của các bit dữ liệu và so sánh với parity bit được gửi đến, nếu như có sự khác nhau thì dữ liệu trên đường truyền đã bị lỗi.

CHƯƠNG 3 THIẾT KẾ BỘ CHUYỂN ĐỔI GIAO THỨC APB SANG UART

3.1 YÊU CẦU CỦA HỆ THỐNG

Bộ chuyển đổi giao thức APB sang UART là một phần của bus AMBA có nhiệm vụ làm trung gian giữa các thiết bị ngoại vi UART với hệ thống bus bên trong.



Hình 3.1. Vị trí bộ chuyển đổi APB sang UART.

Với nhiệm vụ như trên, bộ chuyển đổi giao thức APB sang UART cần thực hiện được các yêu cầu sau đây:

- Sử dụng được các tín hiệu từ hệ thống như xung hệ thống, tín hiệu reset.
- Tiếp nhận, xử lý các dữ liệu từ quá trình ghi của phía APB, chuyển đổi qua dạng tín hiệu UART để truyền đi.
- Tiếp nhận, xử lý các dữ liệu nhận được từ thiết bị UART bên ngoài, chuyển đổi qua bus APB.
- Có thể cài đặt, thiết lập được các thông số như tốc độ truyền, tín hiệu cho phép ngắt, tín hiệu cho phép lõi IP hoạt động.
- Đảm bảo quá trình hoạt động được chính xác, có thể thông báo các lỗi nếu xảy ra trong quá trình hoạt động.

3.2 THIẾT KẾ HỆ THỐNG

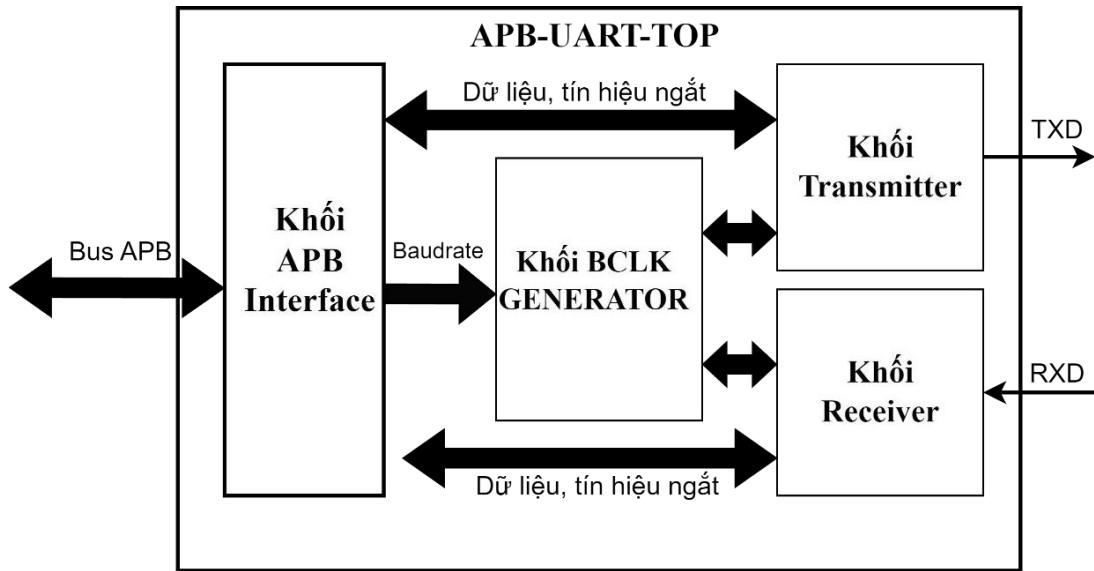
3.2.1 *Chức năng của hệ thống*

Dựa vào các yêu cầu đã đề ra, bộ chuyển đổi giao thức APB sang UART bao gồm các tính năng như sau:

- Bộ chuyển đổi sử dụng xung hệ thống và tín hiệu reset mức thấp từ hệ thống bên trong trong suốt quá trình hoạt động.
- Có cổng giao tiếp APB được thiết kế theo phiên bản APB4 có thể giao tiếp với phía APB master ở hệ thống bên trong. Ở quá trình ghi, cổng này có thể tiếp nhận dữ liệu để truyền đi tới thiết bị ngoại vi có giao tiếp UART, ngoài ra, bộ chuyển đổi còn có thể xử lý, thay đổi các thông số gồm: tốc độ truyền, cho phép bộ chuyển đổi hoạt động, cho phép chế độ parity, cho phép ngắt. Ở quá trình đọc, cổng có thể nhận dữ liệu từ các thiết bị ngoại vi để đưa vào đường dữ liệu đọc hoặc cho phép hệ thống đọc các thông số đã được thiết lập.
- Nhận và truyền dữ liệu theo giao thức UART dựa trên xung của hệ thống và xung riêng, tích hợp bộ FIFO,
- Có một bộ tạo xung riêng cho quá trình hoạt động của bộ thu và bộ nhận bên phía UART. Độ rộng của xung được dựa vào thông số tốc độ truyền đã được thiết lập.
- Bộ chuyển đổi có thể xử lý và thông báo các tín hiệu ngắt sau: Ngắt nguồn, ngắt tràn, ngắt lỗi khung, ngắt lỗi parity.

3.2.2 *Sơ đồ khối tổng quát*

Bộ chuyển đổi APB sẽ bao gồm nhiều khối chính được kết nối với nhau. Sơ đồ khối của bộ chuyển đổi được thể hiện theo như hình 3.2:



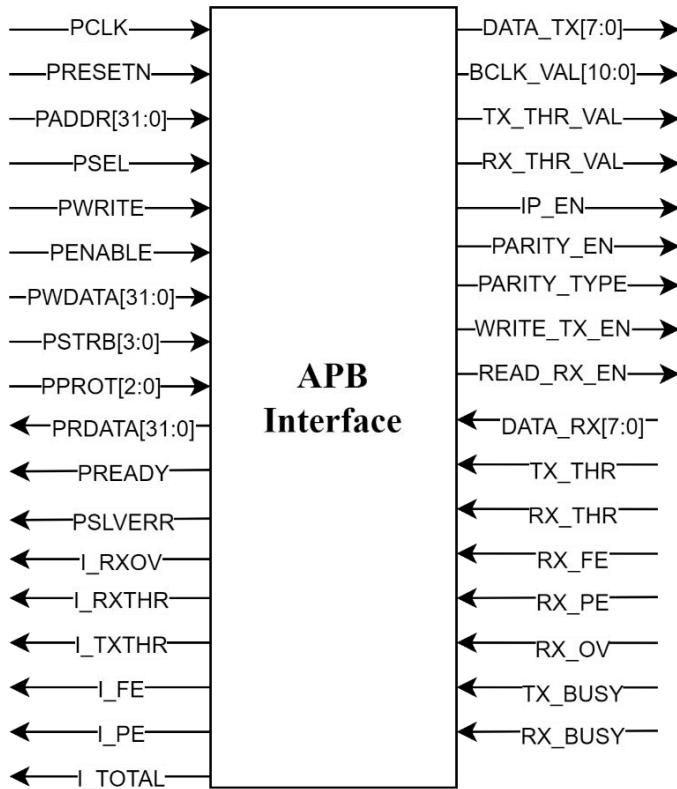
Hình 3.2 Sơ đồ khói của bộ chuyển đổi

Dựa vào các tính năng đã đề ra như trên, các khói chính của bộ chuyển đổi bao gồm:

- Khối APB Interface: Chịu trách nhiệm kết nối với hệ thống bus APB, xử lý các dữ liệu nhận được từ bus APB truyền sang UART và ngược lại, xử lý các tín hiệu ngắn và thông báo cho phía bộ xử lý trung tâm.
- Khối BCLK Generator: Nhận giá trị được thiết lập và tiến hành tạo xung riêng dựa trên xung từ hệ thống. Xung riêng này được sử dụng cho quá trình hoạt động của hai khói UART Receiver và UART Transmitter.
- Khối UART Receiver: Nhận dữ liệu từ đường truyền theo giao thức UART, đưa dữ liệu cho phía APB Interface. Ngoài ra, khói này còn có nhiệm vụ thông báo một số tín hiệu trạng thái, tín hiệu lỗi, tín hiệu ngắn.
- Khối UART Transmitter: Tiếp nhận dữ liệu từ phía APB và truyền qua chân TXD theo giao thức UART. Ngoài ra, khói này cũng thông báo các tín hiệu trạng thái, tín hiệu ngắn cho phía UART.

3.2.3 Thiết kế khói APB Interface

Khối APB Interface có nhiệm vụ luân chuyển dữ liệu giữa bus APB và các khói truyền nhận UART, thiết lập các thông số, xử lý các tín hiệu ngắn. Sơ đồ khói tổng quát của khói APB được thể hiện như hình 3.3:



Hình 3.3. Sơ đồ khái niệm APB Interface

Chi tiết các tín hiệu được liệt kê theo bảng 3.1:

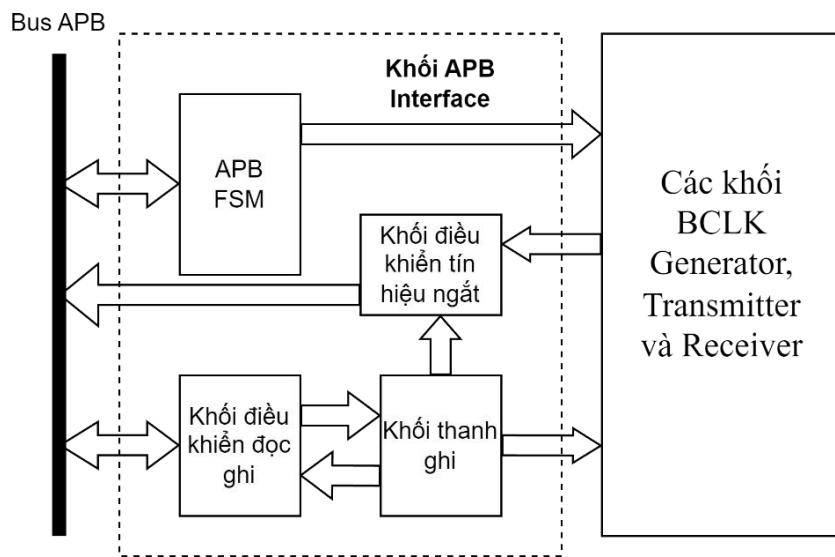
Bảng 3.1. Mô tả tín hiệu APB Interface

| Tên tín hiệu | Loại tín hiệu | Mô tả |
|--------------|---------------|--|
| PCLK | Tín hiệu vào | Xung hệ thống |
| PRESETN | Tín hiệu vào | Tín hiệu khởi động lại từ hệ thống |
| PADDR[31:0] | Tín hiệu vào | Địa chỉ thanh ghi |
| PSEL | Tín hiệu vào | Tín hiệu lựa chọn |
| PWRITE | Tín hiệu vào | Đặt trạng thái ghi hoặc đọc cho quá trình truyền. |
| PENABLE | Tín hiệu vào | Cho biết các chu kỳ kế tiếp trong quá trình truyền. |
| PWDATA[31:0] | Tín hiệu vào | Đường dữ liệu ghi từ bus APB có độ rộng 32 bit. |
| PSTRB[3:0] | Tín hiệu vào | Cho biết byte nào của thanh ghi được cập nhật trong quá trình ghi. |

| | | |
|---------------|--------------|--|
| PPROT[2:0] | Tín hiệu vào | Cho biết cấp độ bảo mật. |
| DATA_RX[7:0] | Tín hiệu vào | Dữ liệu có độ rộng 8 bit nhận được từ khói Receiver. |
| TX_THR | Tín hiệu vào | Thông báo bộ FIFO tại Transmitter có số ô trống bằng hoặc lớn hơn mức đã thiết lập. |
| RX_THR | Tín hiệu vào | Thông báo bộ FIFO tại Receiver có số ô đã được ghi dữ liệu bằng hoặc lớn hơn mức đã thiết lập. |
| RX_FE | Tín hiệu vào | Thông báo lỗi khung dữ liệu UART |
| RX_PE | Tín hiệu vào | Thông báo lỗi Parity |
| RX_OV | Tín hiệu vào | Thông báo tràn bộ đệm FIFO RX. |
| TX_BUSY | Tín hiệu vào | Thông báo bộ Transmitter đang hoạt động |
| RX_BUSY | Tín hiệu vào | Thông báo bộ Receiver đang hoạt động. |
| PRDATA[31:0] | Tín hiệu ra | Đường dữ liệu đọc cho phía bus APB có độ rộng 32 bit |
| PREADY | Tín hiệu ra | Thông báo trạng thái sẵn sàng. |
| PSLVERR | Tín hiệu ra | Thông báo quá trình truyền bị lỗi nếu có |
| DATA_TX[7:0] | Tín hiệu ra | Đường dữ liệu 8 bit nối tới phía Transmitter để truyền đi theo giao thức UART. |
| BCL_VAL[10:0] | Tín hiệu ra | Giá trị để tạo xung BCLK |
| TX_THR_VAL | Tín hiệu ra | Mức ngưỡng cho bộ FIFO của Transmitter |
| RX_THR_VAL | Tín hiệu ra | Mức ngưỡng cho bộ FIFO của Receiver |
| IP_EN | Tín hiệu ra | Cho phép hai bộ Transmitter và Receiver hoạt động. |
| PARITY_EN | Tín hiệu ra | Cho phép dùng phương pháp parity trong quá trình truyền nhận theo UART. |

| | | |
|-------------|-------------|--|
| WRITE_TX_EN | Tín hiệu ra | Cho phép ghi dữ liệu vào bộ FIFO của Transmitter |
| READ_RX_EN | Tín hiệu ra | Cho phép đọc dữ liệu từ bộ FIFO của Receiver |
| I_RXOV | Tín hiệu ra | Tín hiệu ngắt tràn |
| I_TXTTHR | Tín hiệu ra | Tín hiệu ngắt ngưỡng Transmitter |
| I_RXTHR | Tín hiệu ra | Tín hiệu ngắt ngưỡng Receiver |
| I_FE | Tín hiệu ra | Tín hiệu ngắt lỗi khung dữ liệu |
| I_PE | Tín hiệu ra | Tín hiệu ngắt lỗi Parity |
| I_TOTAL | Tín hiệu ra | Tín hiệu ngắt tổng |

Sơ đồ kết nối các khối chính có trong APB Interface:



Hình 3.4. Sơ đồ các khía cắm chính trong APB Interface

Khối APB Interface bao gồm các khía cắm chính như sau:

APB FSM: Một máy trạng thái hữu hạn hạ PREADY xuống mức thấp ở 2 chu kỳ nhảm kéo dài quá trình truyền, ngoài ra còn tạo tín hiệu đọc/ghi mỗi lần dữ liệu cần được ghi hoặc đọc.

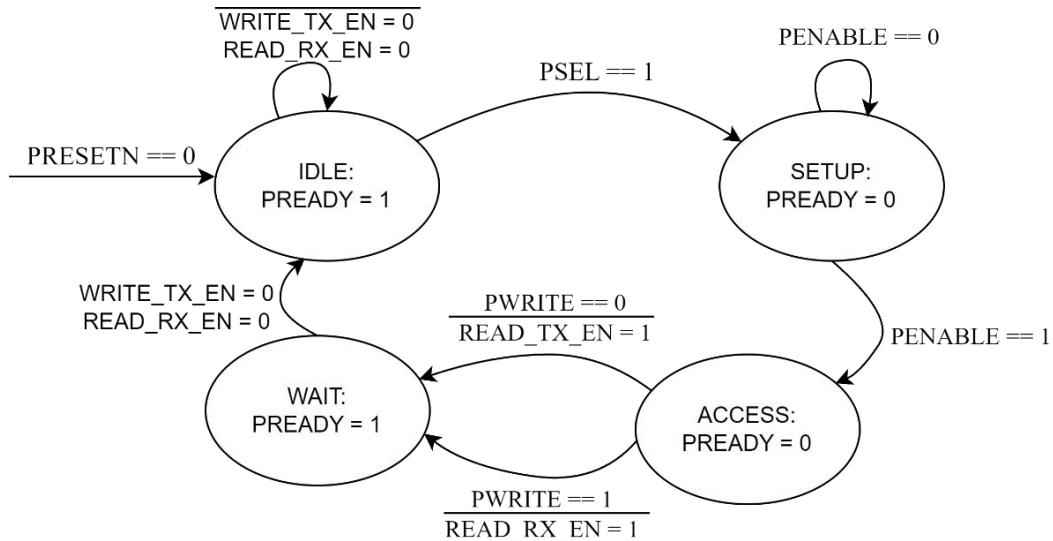
Khối điều khiển đọc ghi: Có nhiệm giải mã địa chỉ và thực hiện đọc ghi giá trị vào thanh ghi thích hợp.

Khối thanh ghi: gồm 4 thanh ghi là: thanh ghi dữ liệu, thanh ghi giá trị BCLK, thanh ghi cho phép và thanh ghi thiết lập mức ngưỡng.

- **Khối APB FSM:**

Khối APB FSM là một máy trạng thái hữu hạn có nhiệm vụ kéo dài quá trình truyền thông qua tín hiệu PREADY và nâng hai tín hiệu cho phép đọc ghi FIFO lên trong một chu kỳ xung PCLK.

Sơ đồ hoạt động của máy trạng thái hữu hạn được mô tả theo như hình 3.5:



Hình 3.5. Hoạt động của APB FSM

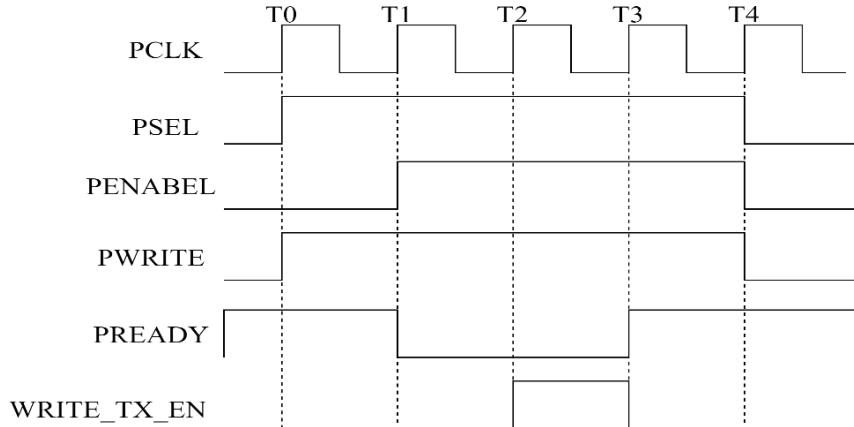
Máy trạng thái APB FSM gồm có 3 trạng thái là IDLE, SETUP và ACCESS.

Trạng thái IDLE: PREADY được giữ ở mức cao, hai tín hiệu READ_RX_EN và WRITE_TX_EN được giữ ở mức cao. Khối chuyển sang trạng thái kế tiếp là SETUP nếu như nhận thấy PSEL ở mức cao.

Trạng thái SETUP: PREADY được hạ xuống mức thấp. Khối chuyển sang trạng thái kế tiếp là ACCESS nếu như nhận thấy PENABLE ở mức cao.

Trạng thái ACCESS: PREADY vẫn giữ ở mức thấp, nếu như tín hiệu PWRITE ở mức cao, WRITE_TX_EN được nâng lên mức cao, nếu PWRITE ở mức thấp, READ_TX_EN được nâng lên mức cao. Máy chuyển sang trạng thái IDLE ở cạnh lên chu kỳ xung kế tiếp mà không cần điều kiện gì.

Ví dụ về một hoạt động ghi được thể hiện như hình 3.6:



Hình 3.6. Ví dụ về hoạt động ghi ở khối APB FSM

Tại T0, khi thấy PSEL, APB FSM chuyển qua trạng thái kế tiếp.

Tại T1, PREADY được hạ xuống mức thấp.

Tại T2, nhận thấy PWRITE ở mức cao nên WRITE_TX_EN được nâng lên mức cao.

Tại T3, WRITE_TX_EN được hạ xuống mức thấp, PREADY được nâng lên mức cao thông báo kết thúc quá trình truyền.

Tại T4, PENBALE được hạ xuống mức thấp, PSEL có thể được giữ ở mức cao nếu như tiếp tục một quá trình truyền khác.

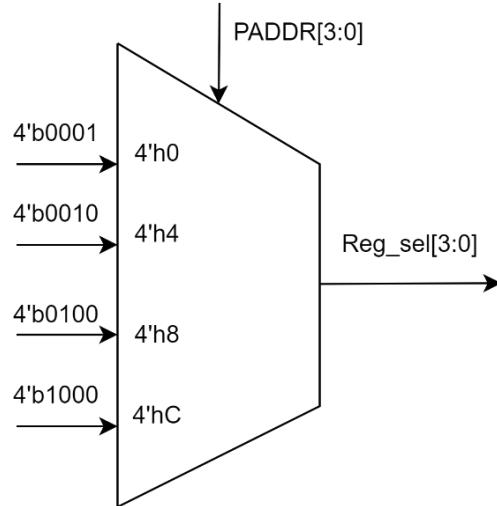
Lưu ý: WRITE_TX_EN và READ_TX_EN chỉ được thay đổi giá trị nếu như địa chỉ cho quá trình truyền là của thanh ghi dữ liệu.

• Khối điều khiển đọc ghi

Khối điều khiển đọc ghi có nhiệm vụ giải mã địa chỉ và thực hiện đọc ghi vào trị trí thanh ghi đúng như địa chỉ đã được đưa tới.

Khối bao gồm hai bộ giải mã địa chỉ, bộ giải mã ghi được gọi là Write Decoder và bộ giải mã đọc được gọi là Read Decoder.

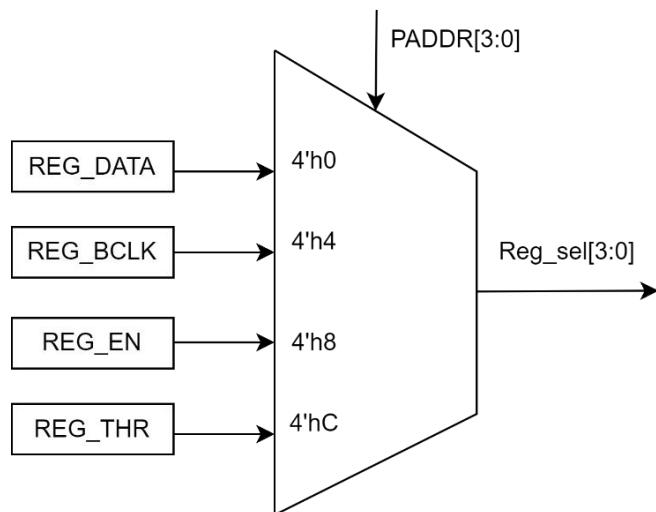
Tại Write Decoder, một thanh ghi Reg_sel được sử dụng để lựa chọn các thanh ghi có trong bộ chuyển đổi.



Hình 3.7. Bộ giải mã ghi

Mỗi thanh ghi được kết nối với các tín hiệu PSEL, PSTRB, PENABLE và một bit của Reg_sel, thanh ghi lưu dữ liệu từ PWDATA nếu như tất cả các tín hiệu trên đều bằng 1.

Ở bộ giải mã đọc, dữ liệu từ các thanh ghi được đưa trực tiếp tới ngõ ra PRDATA dữ theo địa chỉ được đưa tới.



Hình 3.8. Bộ giải mã đọc

Trong trường hợp địa chỉ thanh ghi không phù hợp hoặc tín hiệu PSTRB không cho phép ghi dữ liệu ở bất kỳ thanh ghi nào, tín hiệu PSLVERR sẽ được kéo lên mức cao.

- **Khôi thanh ghi.**

Khối thanh ghi gồm 4 thanh ghi là: REG_DATA, REG_BCLK, REG_EN và REG_THR. Các thanh ghi được mô tả theo bảng 3.2:

Bảng 3.2. Các thanh ghi

| Tên thanh ghi | Loại | Độ rộng | Địa chỉ | Mô tả |
|---------------|---------|---------|---------|--|
| REG_DATA | Đọc/ghi | 8 bit | 0x00 | Lưu trữ tạm thời dữ liệu từ PWDATA hoặc rx_data. |
| REG_BCLK | Đọc/ghi | 11 bit | 0x04 | Lưu trữ giá trị tạo xung BCLK. |
| REG_EN | Đọc/ghi | 7 bit | 0x08 | Lưu trữ các thông số của các tín hiệu cho phép. |
| REG_THR | Đọc/ghi | 4 bit | 0x0C | Lưu trữ các thông số thiết lập mức ngưỡng. |

Chi tiết các thanh ghi:

Thanh ghi REG_DATA: Thanh ghi có độ rộng 8 bit. Thanh ghi này nhận dữ liệu từ PWDATA tại quá trình ghi và từ rx_data tại quá trình đọc. Thanh ghi được phép nhận dữ liệu tại cạnh lén của xung pclk khi cả 4 tín hiệu là reg_sel[0], psel, penable và pstrb[0] ở mức cao. Ngõ ra của thanh ghi được kết nối với đường tín hiệu tx_data[7:0] và paddr[31:0].

Thanh ghi REG_BCLK: Thanh ghi có độ rộng 11 bit, lưu trữ giá trị để tạo xung BCLK ở khối BCLK_generator. Thanh ghi được phép nhận dữ liệu tại cạnh lén của xung pclk khi cả 4 tín hiệu là reg_sel[1], psel, penable và pstrb[1:0] ở mức cao. Ngõ ra của thanh ghi kết nối với đường tín hiệu BCK_VAL[11:0].

Thanh ghi REG_EN: Thanh ghi có độ rộng 8 bit, lưu trữ các tín hiệu cho phép. Thanh ghi được phép nhận dữ liệu tại cạnh lén của xung pclk khi cả 4 tín hiệu là reg_sel[2], psel, penable và pstrb[1:0] ở mức cao. 8 bit của thanh ghi lần lượt tương ứng với các tín hiệu sau:

- REG_EN[0]: TXTHR_EN, cho phép tín hiệu ngắt ngưỡng Transmitter hoạt động.

- REG_EN[1]: RXTHR_EN, cho phép tín hiệu ngắt ngưỡng Receiver hoạt động.
- REG_EN[2]: RXOV_EN, cho phép tín hiệu ngắt tràn hoạt động.
- REG_EN[3]: PE_EN, cho phép tín hiệu ngắt lỗi parity hoạt động.
- REG_EN[4]: FE_EN, cho phép tín hiệu ngắt lỗi khung hoạt động.
- REG_EN[5]: IP_EN, cho phép lỗi IP hoạt động.
- REG_EN[6]: PARITY_EN, cho phép truyền nhận có chế độ parity.
- REG_EN[7]: PARITY_TYPE, cho phép truyền nhận parity chẵn hoặc lẻ.

Thanh ghi REG_THR: Thanh ghi có độ rộng 4 bit, dùng để thiết lập mức ngưỡng thông báo ở bộ FIFO thuộc hai khối Transmitter và Receiver. Thanh ghi được phép nhận dữ liệu tại cạnh lên của xung pclk khi cả 4 tín hiệu là reg_sel[3], psel, penable và pstrb[0] ở mức cao. Giá trị thiết lập mức ngưỡng theo giá trị của thanh ghi như sau:

Bảng 3.3. Các mức ngưỡng thiết lập

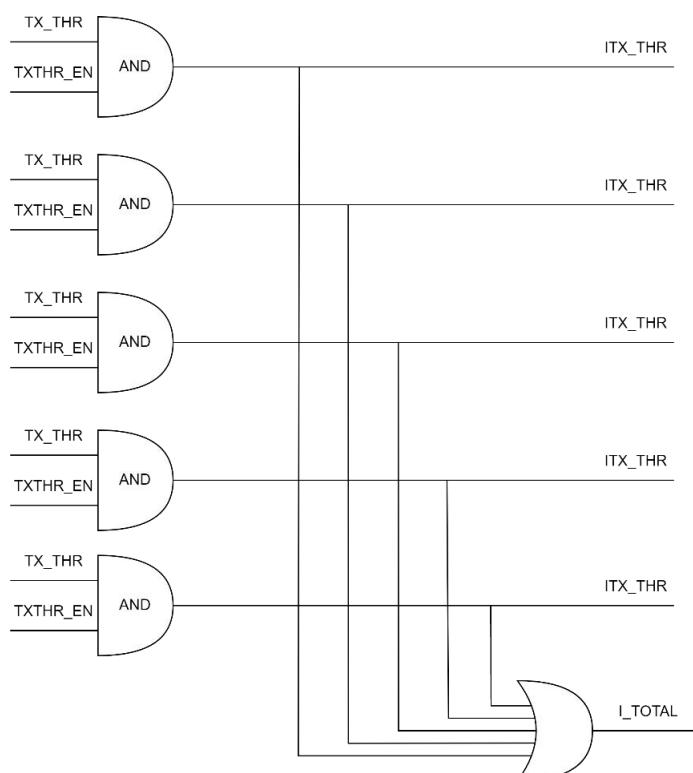
| Bit thanh ghi | Mô tả | Giá trị |
|------------------|--|---|
| REG_THR [1:0] | Thiết lập mức ngưỡng cho bộ Transmitter, thông báo ngắt nếu bộ FIFO có số ô trống nhiều hơn hoặc bằng với mức được thiết lập. Ngõ ra của khối này là tín hiệu tx_thr_val. | 00: Lớn hơn hoặc bằng 16 ô. 01: Lớn hơn hoặc bằng 14 ô. 10: Lớn hơn hoặc bằng 12 ô. 11: Lớn hơn hoặc bằng 8 ô. |
| REG_THR [3:2] | Thiết lập mức ngưỡng cho bộ Receiver, thông báo ngắt nếu bộ FIFO có số ô đã chứa dữ liệu nhiều hơn hoặc bằng với mức được thiết lập. Ngõ ra của khối này là tín hiệu tx_thr_val. | 00: Lớn hơn hoặc bằng 16 ô. 01: Lớn hơn hoặc bằng 8 ô. 10: Lớn hơn hoặc bằng 4 ô. 11: Lớn hơn hoặc bằng 2 ô. |

- Khối điều khiển tín hiệu ngắt.

Khối này có nhiệm vụ nhận tín hiệu ngắn từ các khối khác và thông báo ngắn dựa trên tín hiệu cho phép được thiết lập ở thanh ghi REG_EN. Các tín hiệu ngắn ra của khối này gồm các tín hiệu ngắn sau: ITX_THR, IRX_THR, I_PE, I_FRE, I_OV, I_TOTAL. Trong đó:

- ITX_THR: lên mức cao nếu bộ FIFO của Transmitter đạt hoặc vượt mức ngưỡng đã thiết lập và TXTHR_EN ở mức cao.
- IRX_THR: lên mức cao nếu bộ FIFO của Receiver đạt hoặc vượt mức ngưỡng đã thiết lập và RXTHR_EN ở mức cao.
- I_PE: lên mức cao nếu có lỗi parity xảy ra tại khối Receiver và PE_EN ở mức cao.
- I_FRE: lên mức cao nếu có lỗi khung dữ liệu xảy ra tại khối Receiver và FRE_EN ở mức cao.
- I_OV: lên mức cao nếu có lỗi tràn xảy ra tại khối Receiver và RXOV_EN ở mức cao.
- I_TOTAL: lên mức cao nếu bất kỳ tín hiệu ngắn nào lên mức cao.

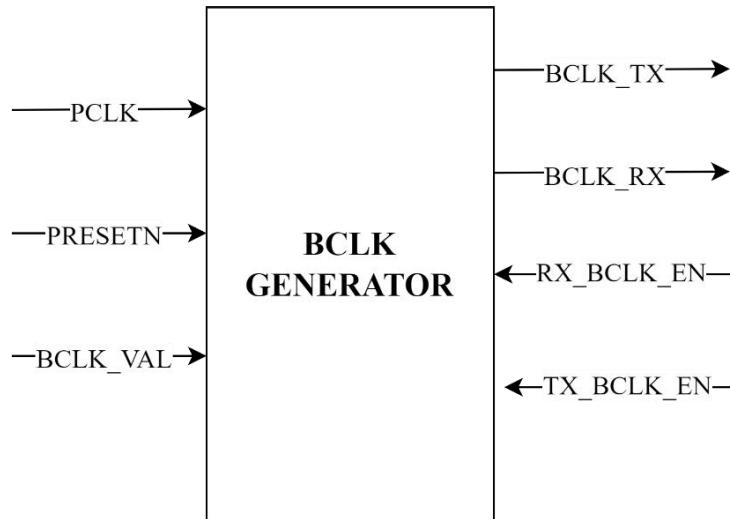
Sơ đồ kết nối của khối này như hình 3.9:



Hình 3.9. Sơ đồ kết nối khối điều khiển tín hiệu ngắn

3.2.4 Thiết kế khối BCLK Generator

Khối BCLK Generator có nhiệm vụ tạo xung clock BCLK từ xung PCLK của hệ thống bên trong. Xung BCLK được sử dụng cho quá trình hoạt động của hai khối Transmitter và Receiver. Sơ đồ khối tổng quát như hình 3.10:



Hình 3.10 Sơ đồ tổng quát khối BCLK Generator

Tín hiệu của khối BCLK Generator được mô tả theo bảng 3.4:

Bảng 3.4 Mô tả tín hiệu BCLK Generator

| Tên tín hiệu | Loại tín hiệu | Mô tả |
|--------------|---------------|--|
| PCLK | Tín hiệu vào | Xung từ hệ thống |
| RESETN | Tín hiệu vào | Tín hiệu khởi động lại từ hệ thống |
| BCLK_VAL | Tín hiệu vào | Giá trị để tạo BCLK. |
| TX_BCLK_EN | Tín hiệu vào | Cho phép tạo xung TX_BCLK để khôi Transmitter hoạt động. |
| RX_BCLK_EN | Tín hiệu vào | Cho phép tạo xung RX_BCLK để khôi Receiver hoạt động. |
| TX_BCLK | Tín hiệu ra | Xung BCLK cho khôi Transmitter |
| RX_BCLK | Tín hiệu ra | Xung BCLK cho khôi Receiver |

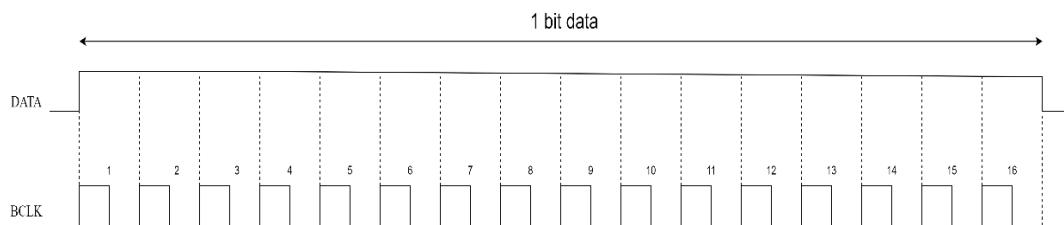
Giá trị BCLK_VAL mà khối BCLK Generator nhận được tính theo công thức như sau:

$$BCLK_VAL = \frac{ClockFrequency}{Baudrate * 16} \quad (3.1)$$

Trong đó:

- BCLK_VAL số lượng chu kỳ xung PCLK có trong một xung BCLK.
- ClockFrequency là tần số hoạt động của hệ thống hay số lượng xung có trong một giây, đơn vị là Hz.
- Tốc độ truyền là tốc độ truyền hay số lượng bit được xử lý trong một giây.

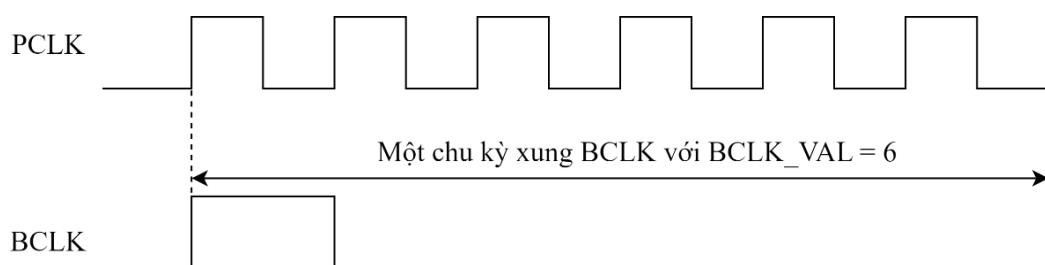
Về giá trị 16, đây là bội số giữa xung BCLK với tốc độ truyền. BCLK Generator có thể tạo ra 16 xung BCLK trong một chu kỳ bit của giao thức UART, tương đương với việc xung BCLK nhanh gấp 16 lần so với tốc độ truyền [5].



Hình 3.11. Xung BCLK trong 1 bit dữ liệu

Lý do việc tạo số lượng xung như vậy là nhằm mục đích giúp khói Transmitter thực hiện việc truyền dữ liệu và khói Receiver thực hiện việc lấy mẫu dữ liệu một cách dễ dàng hơn, cũng như đồng bộ hóa giữa một bên là xung có tần số cao và một bên là giao thức UART có tốc độ xử lý dữ liệu thấp.

Ngoài ra, trong một chu kỳ xung nhịp, BCLK chỉ giữ mức cao trong một chu kỳ của xung PCLK và mức thấp trong toàn bộ thời gian còn lại.



Hình 3.12. Ví dụ về một chu kỳ xung BCLK

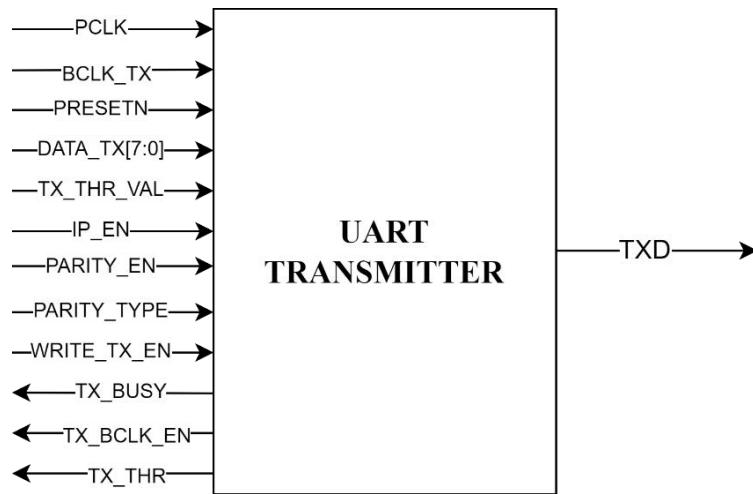
Theo như ví dụ ở hình 3.12, với BCLK_VAL bằng 6, một chu kỳ xung BCLK tương ứng với 6 chu kỳ xung PCLK, trong đó, mức cao duy trì trong chu kỳ PCLK đầu tiên và mức thấp được duy trì ở các chu kỳ còn lại.

Để cho hai khối Transmitter và Receiver hoạt động theo chế độ song công toàn phần, hai xung TX_BCLK và RX_BCLK được cấp riêng cho hai khối. Khối BCLK

Generator chỉ tạo xung BCLK khi có tín hiệu cho phép từ hai khối Transmitter và Receiver. Điều này nhằm giúp các khối bên phía UART có thể hoạt động đồng bộ với hệ thống, đặc biệt là khối Receiver.

3.2.5 Thiết kế khối Transmitter

Khối UART Transmitter có nhiệm vụ nhận dữ liệu có độ rộng 8 bit từ phía APB Interface và truyền đi theo giao thức UART nếu như tín hiệu cho phép IP_EN ở mức cao. Sơ đồ khái quát của UART



Hình 3.13. Sơ đồ khái quát của UART Transmitter

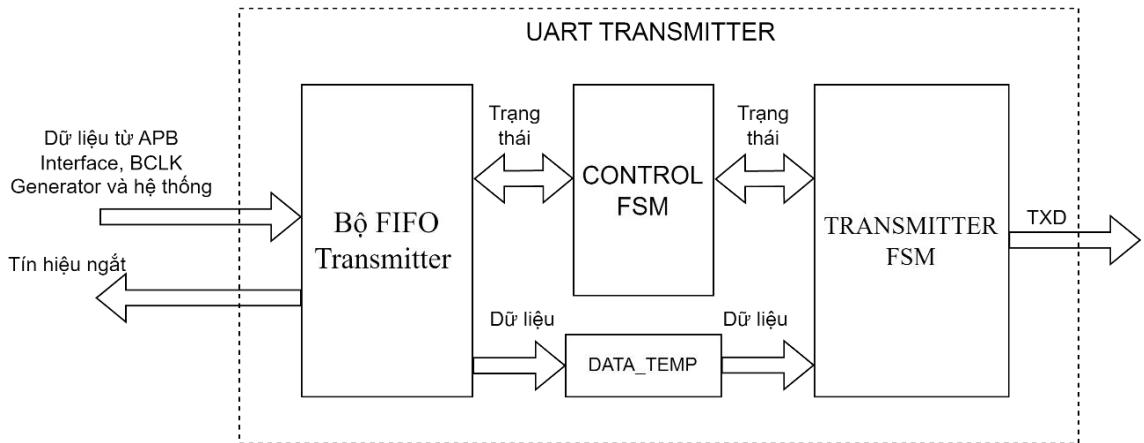
Các tín hiệu của khái quát của UART Transmitter được mô tả theo bảng 3.5:

Bảng 3.5 Mô tả tín hiệu UART Transmitter

| Tên tín hiệu | Loại tín hiệu | Mô tả |
|--------------|---------------|--|
| PCLK- | Tín hiệu vào | Xung của hệ thống. |
| BCLK_TX | Tín hiệu vào | Xung được tạo ra từ khái quát BCLK Generator, |
| RESETN | Tín hiệu vào | Tín hiệu khởi động lại từ hệ thống |
| DATA_TX[7:0] | Tín hiệu vào | Dữ liệu từ phía APB Interface |
| TX_THR_VAL | Tín hiệu vào | Giá trị mức ngưỡng thông báo tín hiệu ngắt |
| IP_EN | Tín hiệu vào | Tín hiệu cho phép khái quát UART Transmitter hoạt động |
| PARITY_EN | Tín hiệu vào | Tín hiệu cho phép truyền có parity |
| PARITY_TYPE | Tín hiệu vào | Cho phép truyền parity chẵn hoặc lẻ |

| | | |
|-------------|--------------|--|
| WRITE_TX_EN | Tín hiệu vào | Tín hiệu cho phép ghi vào bộ FIFO |
| TXD | Tín hiệu ra | Đường truyền theo giao thức UART |
| TX_BUSY | Tín hiệu ra | Thông báo khối UART Transmitter đang thực hiện truyền dữ liệu. |
| TX_BCLK_EN | Tín hiệu ra | Tín hiệu cho phép khối BCLK Generator tạo xung. |
| TX_THR | Tín hiệu ra | Tín hiệu ngắt thông báo bộ FIFO đã đầy hoặc vượt mức ngưỡng |

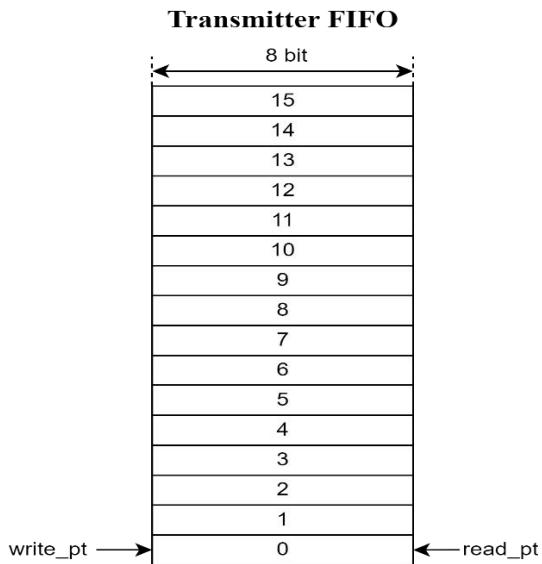
Khối UART Transmitter bao gồm các khối chính là: Bộ FIFO Transmitter, Control FSM, DATA_TEMP và TRANSMITTER FSM. Sơ đồ kết nối của các khối được thể hiện như hình 3.14:



Hình 3.14. Sơ đồ kết nối các khối chính trong UART Transmitter

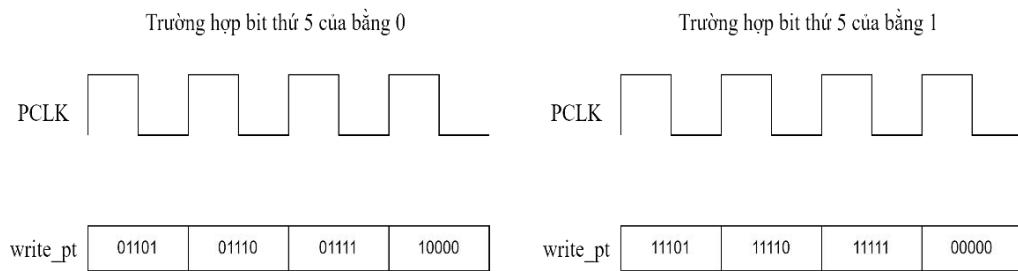
- **Bộ FIFO Transmitter:**

Bộ FIFO của khối Transmitter gồm có 16 ô dữ liệu, mỗi ô có độ rộng là 8 bit. Bộ FIFO có hai thanh ghi thực hiện nhiệm vụ đọc, ghi với tên gọi lần lượt là write_pt và read_pt.



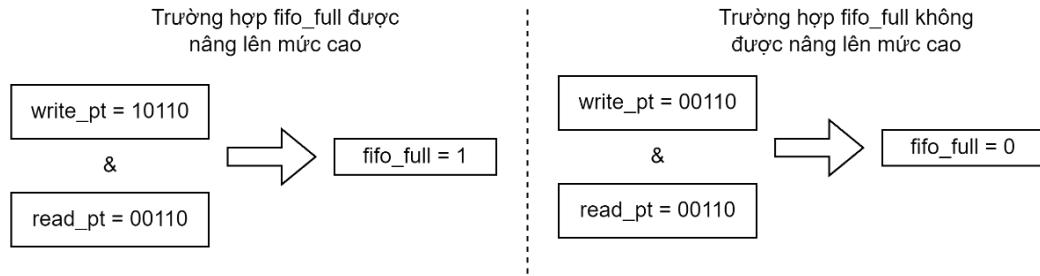
Hình 3.15. Cấu trúc bộ Transmitter FIFO

Thanh ghi `write_pt` có độ rộng là 5 bit. Nhiệm vụ của thanh ghi là cho biết vị trí của bộ FIFO được đưa dữ liệu vào và tăng thêm một giá trị với mỗi quá trình như vậy. Do bộ FIFO có tổng cộng 16 ô dữ liệu nên chỉ có 4 bit đầu tiên của `write_pt` được sử dụng trong quá trình ghi dữ liệu, đồng thời 4 bit này trở lại giá trị 0000 sau khi `write_pt` ghi hết 16 ô dữ liệu, đồng nghĩa với việc thanh ghi trở lại vị trí số 0 của bộ FIFO.



Hình 3.16. Các trường hợp thanh ghi quay về bị trí ban đầu

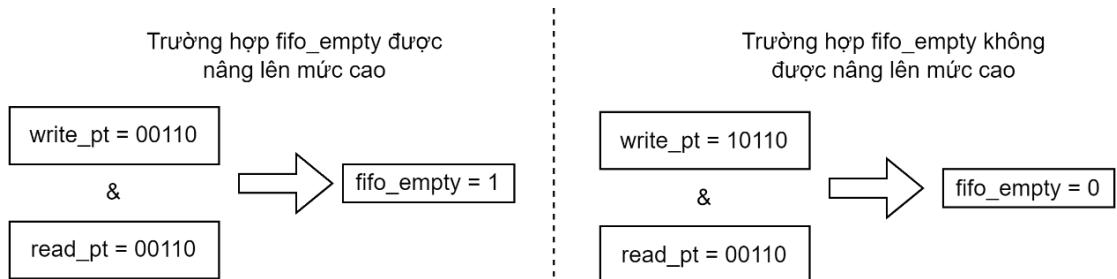
Thanh ghi `write_pt` chỉ hoạt động khi tín hiệu `ip_en`, `write_tx_en` ở mức cao và tín hiệu `fifo_full` mức thấp, trong đó, `fifo_full` là tín hiệu thông báo bộ FIFO đã đầy 16 ô dữ liệu. Tín hiệu `fifo_full` được nâng lên mức cao trong trường hợp 4 bit đầu của `write_pt` bằng với 4 bit đầu của `read_pt` và bit MSB của hai thanh ghi khác nhau.



Hình 3.17. Ví dụ về tín hiệu fifo_full

Để tránh việc bị ghi trùng lặp dữ liệu vào bộ FIFO, tín hiệu write_tx_en chỉ được phía APB Interface nâng lên mức cao trong khoảng thời gian là một chu kỳ xung PCLK mỗi khi có dữ liệu mới được ghi tới.

Thanh ghi read_pt cũng hoạt động tương tự như write_pt. Thanh ghi này có độ rộng 5 bit và cộng thêm một giá trị mỗi lần đọc dữ liệu. Thanh ghi read_pt hoạt động khi tín hiệu read_tx_en, ip_en ở mức cao và fifo_empty ở mức thấp, trong đó, tín hiệu fifo_empty thông báo không còn dữ liệu nào ở trong bộ FIFO. Tín hiệu fifo_empty được nâng lên mức cao khi cả 5 bit của write_pt và read_pt bằng nhau.



Hình 3.18. Ví dụ về tín hiệu fifo_empty

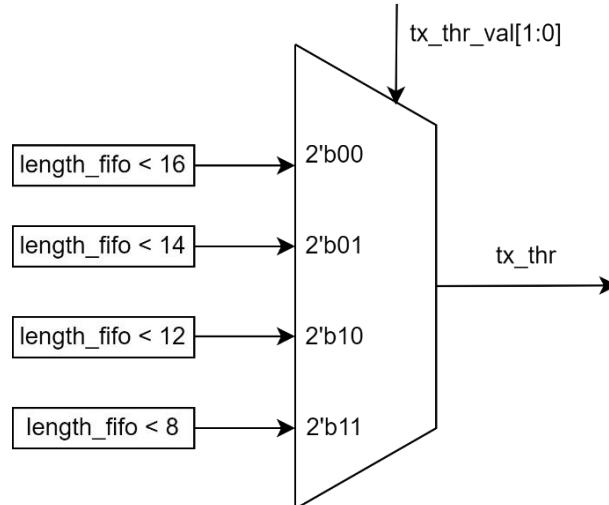
Do dữ liệu được đọc từ FIFO được chuyển sang thanh ghi khác để tiến hành truyền đi theo giao thức UART nên khoảng thời gian mỗi lần đọc dữ liệu là rất lớn cho với chu kỳ của xung PCLK. Để tránh việc dữ liệu chưa kịp truyền đi đã bị ghi đè lên, tín hiệu read_tx_en được nâng lên mức cao trong khoảng thời gian là một chu kỳ xung PCLK tại thời điểm trước khi truyền dữ liệu theo giao thức UART và sẽ giữ nguyên mức thấp trong quá trình truyền một dữ liệu 8 bit.

Ngoài ra, trong quá trình hoạt động của bộ FIFO, một thanh ghi length_fifo được dùng để tính số lượng ô FIFO đã chứa dữ liệu. Thanh ghi này cộng thêm một giá trị với mỗi quá trình ghi và trừ đi một giá trị với mỗi quá trình đọc. Giá trị của

thanh ghi length_fifo được dùng để xác định trạng thái cho tín hiệu ngắt tx_thr dựa trên mức ngưỡng tx_thr_val được thiết lập. Các trường hợp của tx_thr_val được mô tả như bảng 3.6:

Bảng 3.6. Mô tả các giá trị ngưỡng của Transmitter

| Giá trị tx_thr_val | Mô tả |
|--------------------|--|
| 00 | Nếu length_fifo < 16, tx_thr nâng lên mức cao. |
| 01 | Nếu length_fifo < 14, tx_thr nâng lên mức cao. |
| 10 | Nếu length_fifo < 12, tx_thr nâng lên mức cao. |
| 11 | Nếu length_fifo < 8, tx_thr nâng lên mức cao. |



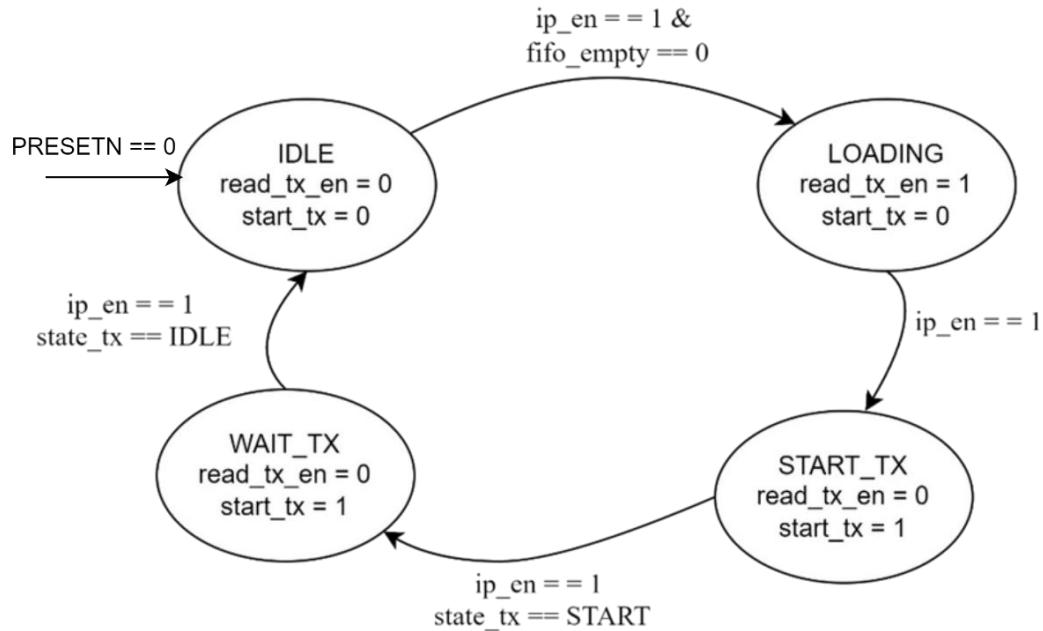
Hình 3.19. Sơ đồ kết nối của tín hiệu ngắt ngưỡng khói Transmitter

• Khối Control FSM

Khối Control FSM có hai nhiệm vụ, đầu tiên lái tín hiệu read_tx_en để thực hiện đọc dữ liệu từ bộ FIFO và đưa vào thanh ghi DATA_TEMP, do tín hiệu read_tx_en cần được lái thích hợp để tránh việc dữ liệu ở DATA_TEMP chưa kịp truyền đi theo giao thức UART đã bị dữ liệu khác ghi đè vào. Nhiệm vụ thứ hai là

lái tín hiệu start_tx để bắt đầu truyền dữ liệu theo giao thức UART ở khối Transmitter FSM.

Khối Control FSM gồm 3 trạng thái là: IDLE, LOADING và WAITING được mô tả theo hình 3.20:



Hình 3.20. Hoạt động của khối Control FSM

Trạng thái hoạt động của khối Control FSM được mô tả như sau:

IDLE: ở trạng thái này, cả hai tín hiệu read_tx_en và start_tx đều được lái về mức thấp. Khối chuyển qua trạng thái kế tiếp là LOADING nếu tín hiệu ip_en ở mức cao và tín hiệu fifo_empty ở mức thấp.

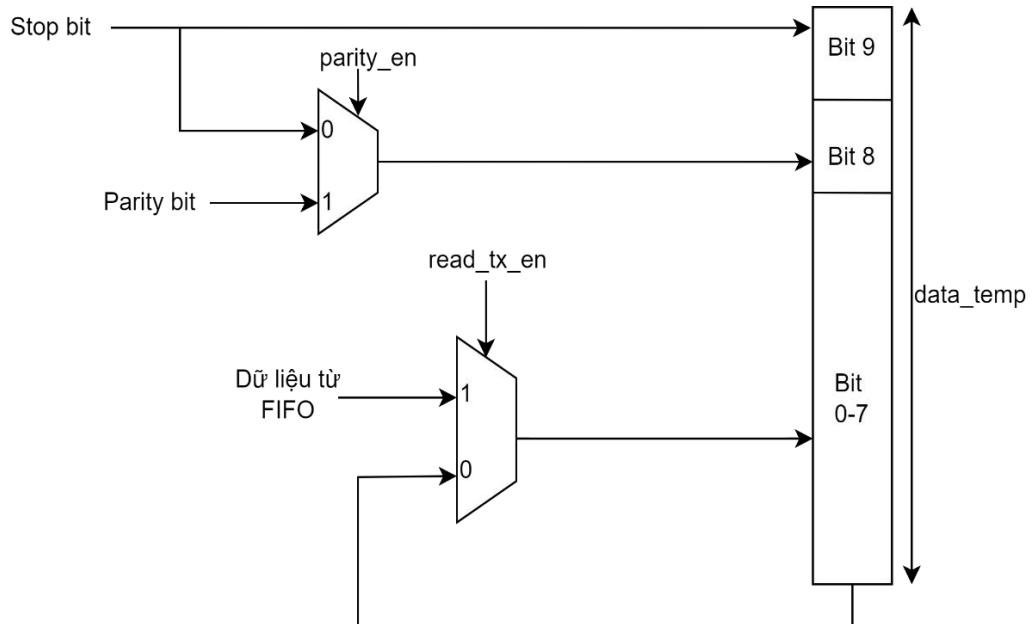
LOADING: tín hiệu read_tx_en được nâng lên mức cao nếu như tín hiệu ip_en ở mức cao, đồng thời, khối cũng chuyển qua trạng thái kế tiếp là START_TX.

START_TX: ở trạng thái này, tín hiệu read_tx_en được kéo về mức thấp, trong khi tín hiệu start_tx được nâng lên mức cao để bắt đầu quá trình truyền dữ liệu theo giao thức UART. Khối sẽ chuyển sang trạng thái kế tiếp là WAIT_TX nếu như trạng thái của khối Transmitter FSM chuyển sang START.

WAIT_TX: đây là trạng thái chờ của khối Control FSM, hai tín hiệu read_tx_en và start_tx vẫn giữ nguyên giá trị. Khối chuyển sang trạng thái kế tiếp là IDLE nếu như trạng thái của khối Transmitter FSM đã quay lại IDLE.

- **Thanh ghi DATA_TEMP**

Thanh ghi DATA_TEMP có nhiệm vụ lưu trữ dữ liệu tạm thời phục vụ cho quá trình truyền dữ liệu bằng giao thức UART.



Hình 3.21. Sơ đồ kết nối của thanh ghi DATA_TEMP

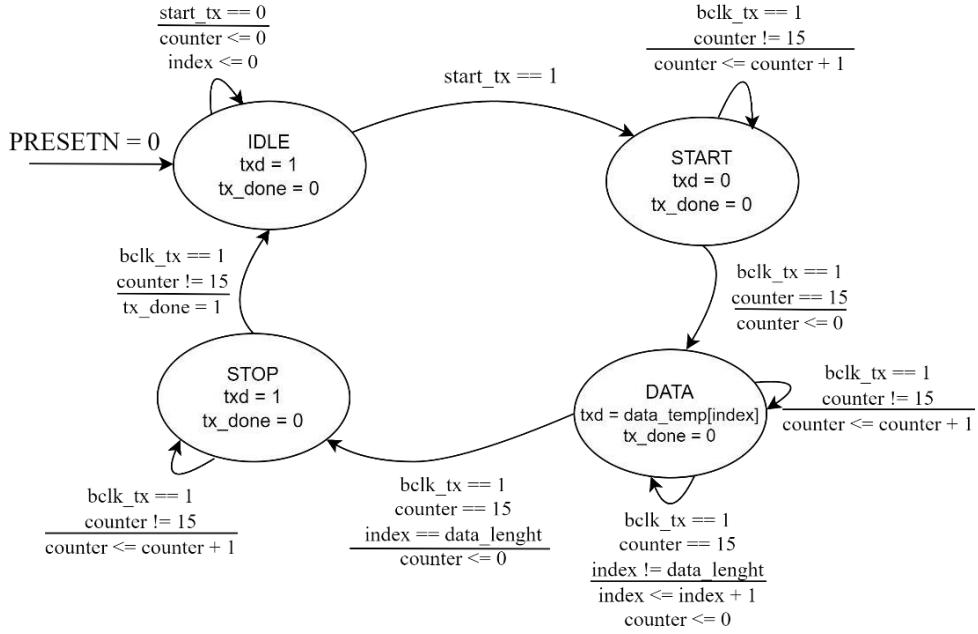
Các bit của DATA_TEMP sẽ chứa dữ liệu như sau:

- Bit 0 đến bit 7: Chứa 8 bit dữ liệu từ bộ FIFO.
- Bit 8: Chứa stop bit nếu tín hiệu PARITY_EN ở mức thấp và parity bit nếu tín hiệu PARITY_EN ở mức cao.
- Bit 9: Luôn luôn chứa stop bit.

Lưu ý: stop bit luôn là bit 1, parity bit là kết quả EXOR của toàn bộ 8 bit dữ liệu nếu ở chế độ parity lẻ và là đảo ngược của kết quả nếu ở chế độ parity chẵn.

- **Khối Transmitter FSM**

Khối Transmitter FSM là một máy trạng thái hữu hạn có nhiệm vụ truyền dữ liệu ở thanh ghi DATA_TEMP theo giao thức UART. Transmitter FSM hoạt động khi tín hiệu ip_en và start_tx ở mức cao. Các trạng thái của Transmitter FSM gồm: IDLE, START, DATA và STOP. Hoạt động của Transmitter FSM được mô tả như hình 3.22:



Hình 3.22. Hoạt động của khối Transmitter FSM

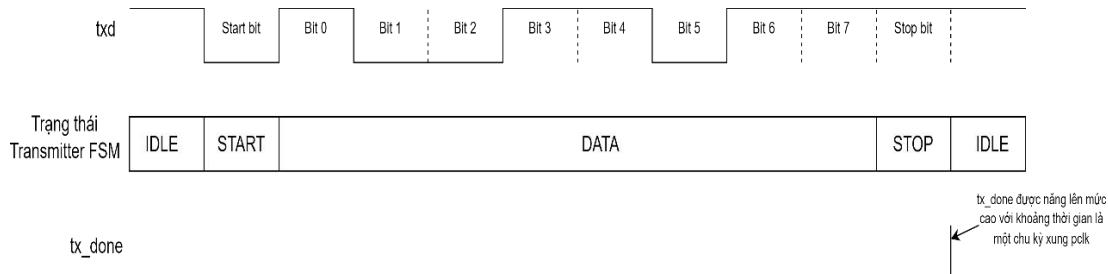
Hoạt động của khối Transmitter FSM được mô tả như sau:

Trạng thái IDLE: Ở trạng thái này, ngõ ra TXD luôn được giữ ở mức cao, tx_done luôn ở mức thấp. Khối Transmitter FSM sẽ chuyển sang trạng thái kế tiếp là START nếu nhận thấy tín hiệu start_tx lên mức cao.

Trạng thái START: Trạng thái này có mục đích là truyền start bit. Với mỗi cạnh lên của xung bclk_tx, thanh ghi counter cộng thêm một giá trị, khi giá trị này bằng với 15, tương ứng với 16 xung bclk cho một bit, khối Transmitter FSM chuyển sang trạng thái kế tiếp là DATA.

Trạng thái DATA: Các bit dữ liệu trong DATA_TEMP được truyền đi lần lượt dựa theo chỉ số index, trong đó, index thể hiện vị trí của bit trong DATA_TEMP được truyền. Ngõ ra TXD lái theo giá trị của bit trong DATA_TEMP, khởi đầu với index bằng 0 tương ứng với bit 0. Thanh ghi counter lần lượt tăng giá trị từ 0 đến 15 với mỗi cạnh lên của xung BCLK và quay về giá trị 0 sau khi đạt giá trị 15. Với mỗi lần thanh ghi counter đạt giá trị 15, biến index được cộng thêm một giá trị. Khối Transmitter FSM sẽ chuyển sang trạng thái kế tiếp là STOP nếu như counter đạt giá trị 15 và index đạt giá trị bằng với data_length, trong đó, data_length cho biết độ dài dữ liệu cần truyền, nếu như tín hiệu PARITY_EN ở mức thấp, data_length có giá trị là 8, ngược lại, data_length có giá trị là 9.

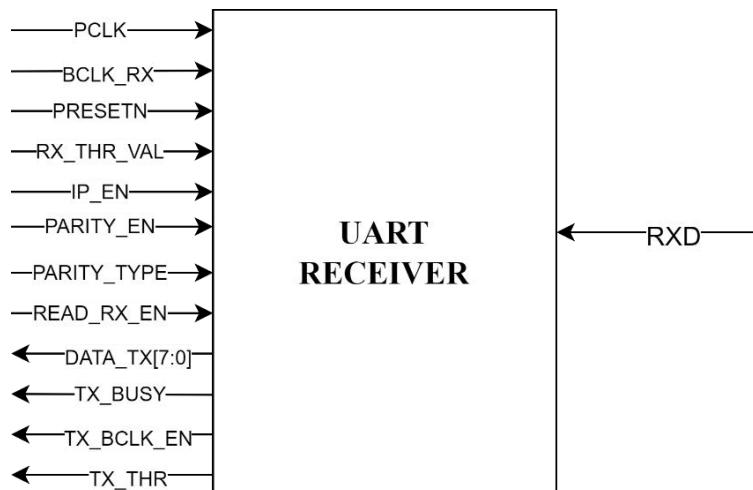
Trạng thái STOP: Trạng thái này có mục đích là truyền stop bit Khởi đầu với giá trị bằng 0, thanh ghi counter tăng một giá trị với mỗi cạnh lên xung bclk_tx. Khi counter có giá trị 15, tín hiệu tx_done được nâng lên mức cao và khôi Transmitter FSM chuyển sang trạng thái IDLE.



Hình 3.23. Ví dụ về trạng thái của Transmitter FSM trong quá trình truyền

3.2.6 Thiết kế khối Receiver

Khối UART Receiver có nhiệm vụ thực hiện nhận dữ liệu từ ngõ vào RXD và đưa dữ liệu vào bộ FIFO. Dữ liệu tại bộ FIFO được đưa sang thanh ghi REG_DATA của khối APB Interface nếu như có yêu cầu đọc từ bus APB. Sơ đồ khái niệm tổng quát của UART Receiver như sau:



Hình 3.24. Sơ đồ khái niệm tổng quát của Receiver

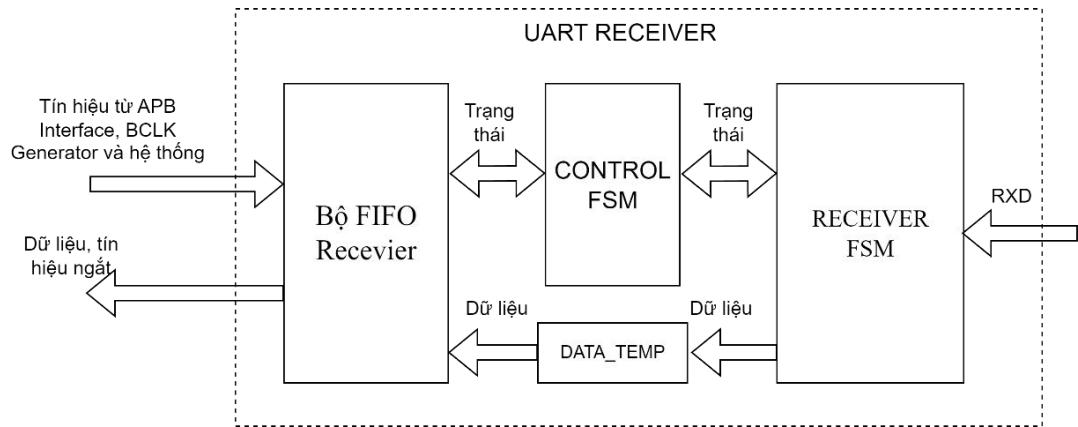
Các tín hiệu của khái niệm tổng quát của Receiver được mô tả theo như bảng 3.7:

Bảng 3.7 Mô tả tín hiệu khái niệm tổng quát của Receiver

| Tên tín hiệu | Loại tín hiệu | Mô tả |
|--------------|---------------|-------|
|--------------|---------------|-------|

| | | |
|-----------------|--------------|---|
| RXD | Tín hiệu vào | Đường nhận dữ liệu được truyền theo giao thức UART |
| PCLK | Tín hiệu vào | Xung của hệ thống |
| BCLK_RX | Tín hiệu vào | Xung được tạo từ khối BCLK Generator |
| PRESETN | Tín hiệu vào | Tín hiệu khởi động lại từ hệ thống |
| RX_THR_VAL[1:0] | Tín hiệu vào | Thiết lập giá trị cho mức ngưỡng |
| IP_EN | Tín hiệu vào | Tín hiệu cho phép khối Receiver hoạt động. |
| PARITY_EN | Tín hiệu vào | Tín hiệu cho phép thực hiện nhận có parity bit. |
| PARITY_TYPE | Tín hiệu vào | Tín hiệu thiết lập cho phép kiểm tra parity theo phương pháp chẵn hoặc lẻ |
| READ_RX_EN | Tín hiệu vào | Tín hiệu cho phép thực hiện đọc dữ liệu từ bộ FIFO |
| DATA_RX[7:0] | Tín hiệu ra | Ngõ ra của dữ liệu 8 bit lấy từ bộ FIFO |
| RX_BUSY | Tín hiệu ra | Thông báo Receiver đang thực hiện nhận dữ liệu theo giao thức UART |
| RX_BCLK_EN | Tín hiệu ra | Tín hiệu cho phép khối BCLK Generator tạo xung BCLK cho Receiver |
| RX_THR | Tín hiệu ra | Tín hiệu thông báo số ô nhớ chứa dữ liệu đã bằng hoặc vượt mức ngưỡng quy định. |
| RX_FE | Tín hiệu ra | Tín hiệu báo lỗi khung dữ liệu |
| RX_PE | Tín hiệu ra | Tín hiệu báo lỗi parity |
| RX_OV | Tín hiệu ra | Tín hiệu báo tràn bộ FIFO |

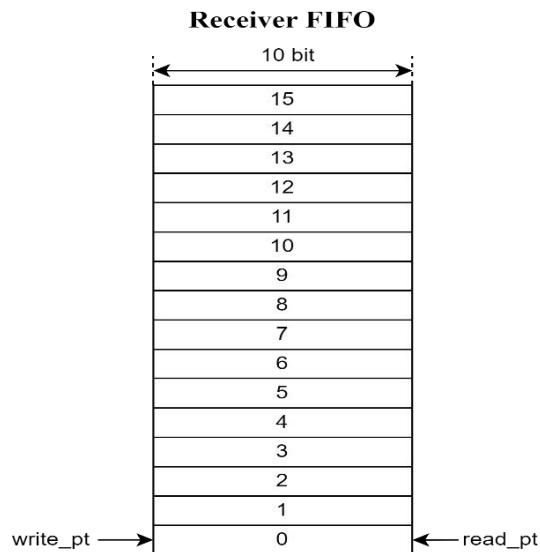
Khối UART RECEIVER bao gồm 4 khối chính: Khối FIFO Receiver, khối Control FSM, thanh ghi DATA_TEMP và khối Receiver FSM. Các khối này được nối với nhau theo sơ đồ ở hình 3.25:



Hình 3.25. Sơ đồ kết nối các khối chính của UART Receiver
Các khối chính được mô tả dưới đây:

- **Bộ FIFO Receiver**

Bộ FIFO Receiver bao gồm 16 ô dữ liệu, mỗi ô có độ rộng 10 bit. Quá trình đọc dữ liệu được thực hiện thông qua thanh ghi read_pt và ghi thông qua thanh ghi write_pt.



Hình 3.26. Cấu trúc bộ FIFO Receiver

Khác với bộ Transmitter, mỗi ô trong bộ FIFO Receiver có độ rộng 10 bit, trong đó gồm 8 bit dữ liệu và 2 bit báo lỗi parity và lỗi khung.

| | | |
|------------|-----------|---------------|
| Lõi parity | Lõi khung | 8 bit dữ liệu |
|------------|-----------|---------------|

Hình 3.27. Cấu tạo của một ô dữ liệu trong bộ FIFO Receiver

Cả hai con thanh ghi `read_pt` và `write_pt` đều có độ rộng là 5 bit, hai thanh ghi sẽ được cộng thêm một giá trị với mỗi cạnh bên của xung PCLK. Để tránh trường hợp các dữ liệu bị ghi lại trùng lặp hoặc đọc liên tục, tín hiệu `write_rx_en` được tạo thêm để phục vụ cho quá trình ghi và `read_rx_en` phục vụ cho quá trình đọc. Tín hiệu `write_en` được tạo ra từ khối Control FSM và được nâng lên mức cao trong khoảng thời gian là một chu kỳ xung PCLK mỗi khi Receiver nhận toàn bộ một dữ liệu mới. Tín hiệu `read_en` được tạo ra từ phía APB Interface mỗi khi có yêu cầu đọc dữ liệu từ hệ thống. Thanh ghi `read_pt` hoặc `write_pt` chỉ tăng giá trị khi `read_rx_en` hoặc `write_rx_en` ở mức cao.

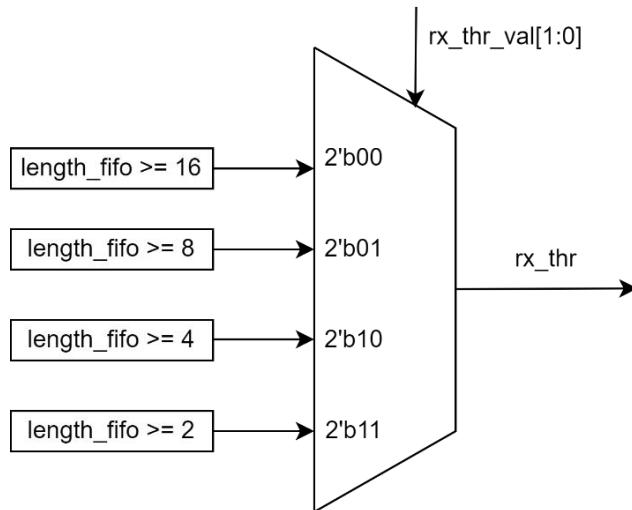
Giống như khối Transmitter, tín hiệu `fifo_full` dùng để thông báo bộ `fifo` đã đầy và `fifo_empty` thông báo bộ FIFO không còn dữ liệu bên trong. Tín hiệu `fifo_full` được nâng lên mức cao khi 4 bit đầu của `write_pt` và `read_pt` giống nhau, trong khi bit thứ 5 khác nhau. Tín hiệu `fifo_empty` được nâng lên mức cao khi toàn bộ 5 bit của `write_pt` và `read_pt` giống nhau.

Trong quá trình hoạt động của bộ FIFO, một thanh ghi `length_fifo` được tạo ra để tính số ô FIFO đã chứa dữ liệu, Thanh ghi này cộng thêm một giá trị với mỗi lần ghi và trừ đi một giá trị với mỗi lần đọc. Giá trị của thanh ghi được dùng để xét trạng thái của tín hiệu `RX_THR` dựa theo mức ngưỡng đã thiết lập ở tín hiệu `RX_THR_VAL`, trong đó, ngưỡng ở đây là số ô trong bộ FIFO đã chứa dữ liệu. Bảng 3.8 được dùng để mô tả các trường hợp phụ thuộc vào giá trị của `RX_THR_VAL`:

Bảng 3.8. Mô tả các giá trị ngưỡng của Receiver

| Giá trị <code>rx_thr_val</code> | Mô tả |
|---------------------------------|--|
| 00 | Nếu <code>length_fifo</code> ≥ 16 , <code>rx_thr</code> nâng lên mức cao. |
| 01 | Nếu <code>length_fifo</code> ≥ 8 , <code>rx_thr</code> nâng lên mức cao. |

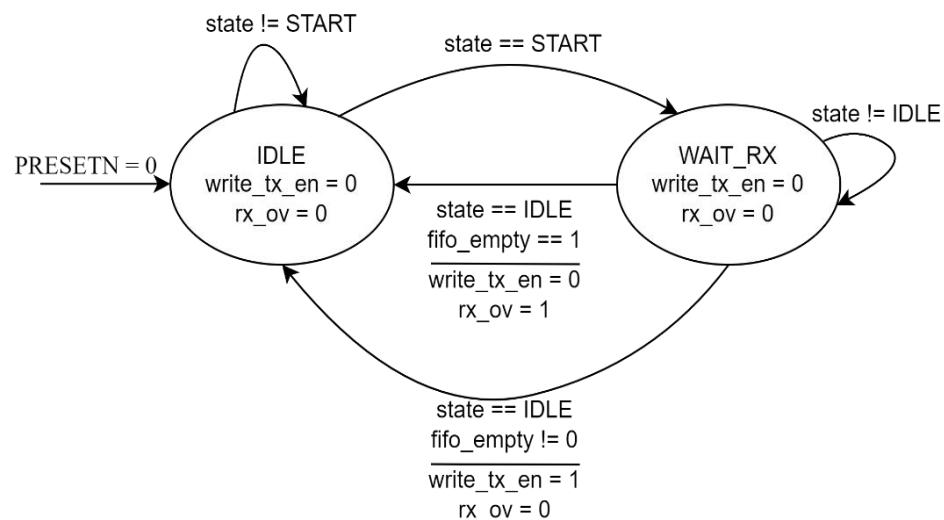
| | |
|----|---|
| 10 | Nếu $\text{length_fifo} \geq 4$, rx_thr nâng lên mức cao. |
| 11 | Nếu $\text{length_fifo} \geq 2$, rx_thr nâng lên mức cao. |



Hình 3.28. Sơ đồ kết nối của tín hiệu TX_THR

- **Khối Control FSM**

Khối Control FSM có nhiệm vụ nâng tín hiệu write_tx_en lên mức cao trong một khoảng thời gian thích hợp. Khối Control FSM gồm 2 trạng thái là IDLE và WAIT_RX. Hoạt động của Control FSM được mô tả như sau:



Hình 3.29 Hoạt động của khối Control FSM của Receiver

Các trạng thái của Control FSM của khối Receiver được mô tả như sau:

Trạng thái IDLE: khói Control FSM chờ nhận dữ liệu trong trạng thái này, tín hiệu write_tx_en và rx_ov đều được giữ ở mức thấp. Khối chuyển sang trạng thái kế tiếp là WAIT_RX nếu như state chuyển sang giá trị START, trong đó, state là thanh ghi trạng thái của khói Receiver FSM.

Trạng thái WAIT_RX: ở trạng thái này, tín hiệu write_tx_en và rx_ov vẫn được giữ ở mức thấp. Nếu như thanh ghi state chuyển về giá trị IDLE, khói Control FSM chuyển về trạng thái IDLE, đồng thời khi đó, tùy thuộc vào giá trị của tín hiệu fifo_full khi đó mà có các trường hợp sau:

- Trường hợp tín hiệu fifo_full ở mức thấp: Lúc này, tín hiệu write_tx_en được nâng lên mức cao trong một chu kỳ xung PCLK, tín hiệu rx_ov được giữ ở mức thấp.
- Trường hợp tín hiệu fifo_full ở mức cao: Lúc này, tín hiệu write_tx_en được giữ nguyên mức thấp, tín hiệu rx_ov được nâng lên mức cao trong một chu kỳ xung PCLK.

- **Thanh ghi DATA_TEMP**

Thanh ghi DATA_TEMP có nhiệm vụ lưu trữ tạm thời các bit dữ liệu nhận được trước khi đưa dữ liệu vào bộ FIFO.



Hình 3.30 Thanh ghi DATA_TEMP của Receiver

Thanh ghi DATA_TEMP có độ rộng là 10 bit, trong đó:

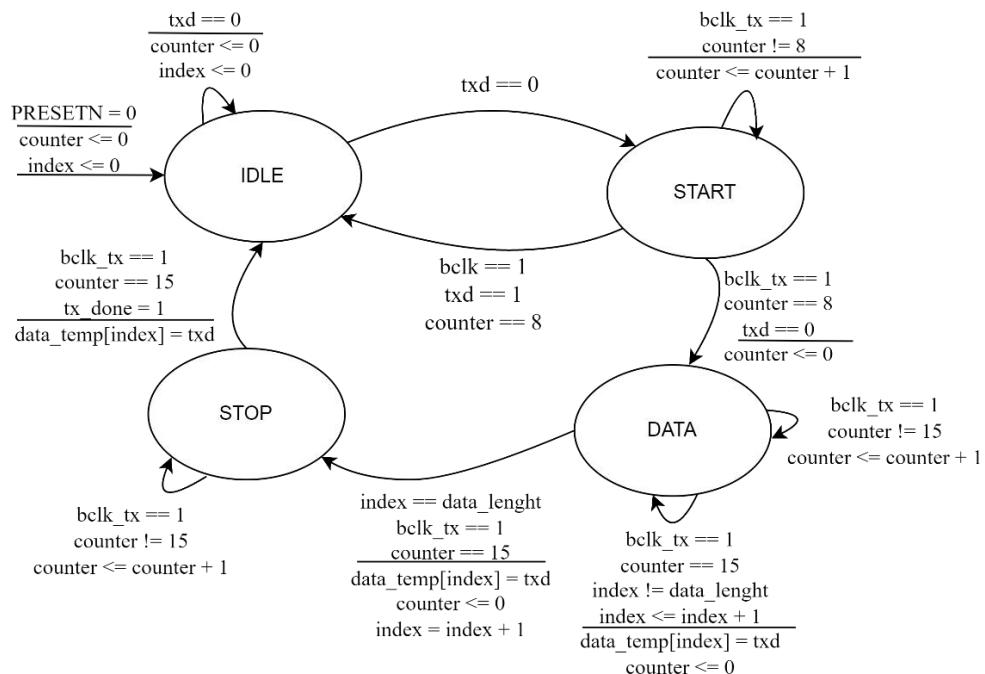
- 8 bit đầu chứa 8 bit dữ liệu nhận được.
- Bit thứ 9 chứa parity bit nếu tín hiệu PARITY_EN ở mức cao và chứa stop bit nếu PARITY_EN ở mức thấp.

- Bit thứ 10 chứa stop bit nếu PARITY_EN ở mức cao và không cần quan tâm nếu PARITY_EN ở mức thấp.

Dữ liệu từ thanh ghi DATA_TEMP được truyền vào bộ FIFO nếu tín hiệu write_rx_en ở mức cao và fifo_full ở mức thấp.

• Khối Receiver FSM

Khối Receiver FSM có nhiệm vụ nhận dữ liệu được truyền theo giao thức UART ở ngõ vào RXD. Do mức điện áp ở RXD có thể thay đổi bất cứ lúc nào nên khối này phải luôn trong trạng thái sẵn sàng để có thể đồng bộ hóa quá trình nhận dữ liệu, hạn chế độ trễ xảy ra. Đây là một máy trạng thái gồm có 4 trạng thái là: IDLE, START, DATA và STOP. Hoạt động của khối này được mô tả như hình 3.31:



Hình 3.31 Hoạt động của khối Receiver FSM

Các trạng thái của khối Receiver FSM được giải thích như sau:

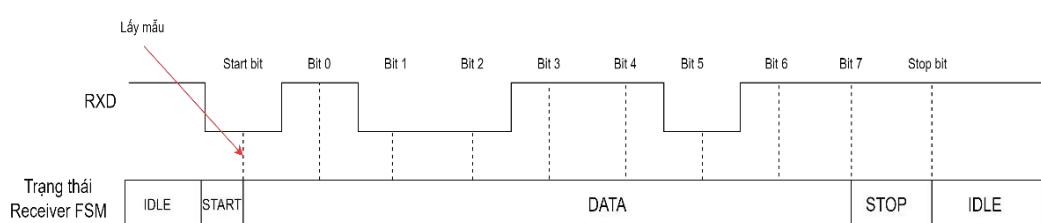
Trạng thái IDLE: Ở trạng thái này, khối Receiver FSM trong trạng thái chờ dữ liệu được gửi đến. Các thanh ghi counter và index đều được giữ ở giá trị là 0. Khối ngay lập tức chuyển sang trạng thái kế tiếp là START nếu như đường RXD được kéo xuống mức thấp.

Trạng thái START: Mục đích của trạng thái này là nhằm xác định start bit. Thanh ghi counter được cộng thêm một giá trị với mỗi cạnh lên của xung BCLK_RX. Nếu thanh ghi counter đạt giá trị bằng 8, hay thời điểm đang ở giữa chu kỳ của start bit, ngõ vào RXD được kiểm tra. Nếu ngõ vào RXD vẫn ở mức thấp, start bit là thật, khởi chuyển sang trạng thái kế tiếp là DATA, thanh ghi counter được đưa về giá trị 0. Nếu ngõ vào RXD ở mức cao, điều này chứng tỏ rằng tín hiệu mức thấp kia là giả và khởi quay lại trạng thái IDLE.

Trạng thái DATA: Đây là trạng thái tiến hành đưa dữ liệu từ TXD vào trong thanh ghi DATA_TEMP. Thanh ghi counter cộng thêm một giá trị với mỗi cạnh lên của xung BCLK. Khi giá trị counter đạt 15, tương đương với một chu kỳ bit của UART, giá trị từ TXD được đưa vào bit của DATA_TEMP có chỉ số tương ứng với giá trị của thanh ghi index, đồng thời, thanh ghi index cũng được cộng thêm một giá trị. Thanh ghi length_data được thêm vào để thể hiện độ rộng của bit dữ liệu, nếu tín hiệu PARITY_EN ở mức cao, length_data có giá trị là 9, ngược lại, length_data có giá trị là 9. Nếu như giá trị index bằng với length_data, khởi Receiver FSM chuyển sang trạng thái kế tiếp là STOP, đồng thời thanh ghi counter được đưa về giá trị 0.

Trạng thái STOP: Mục đích của trạng thái này là nhằm xác định stop bit. Thanh ghi counter được cộng thêm một với mỗi cạnh lên xung BCLK. Nếu counter đạt giá trị là 15, giá trị TXD được đưa vào bit của DATA_TEMP có chỉ số tương ứng index. Khởi cũng quay về trạng thái IDLE.

Hình 3.22 là một ví dụ về quá trình hoạt động của khối Receiver FSM:

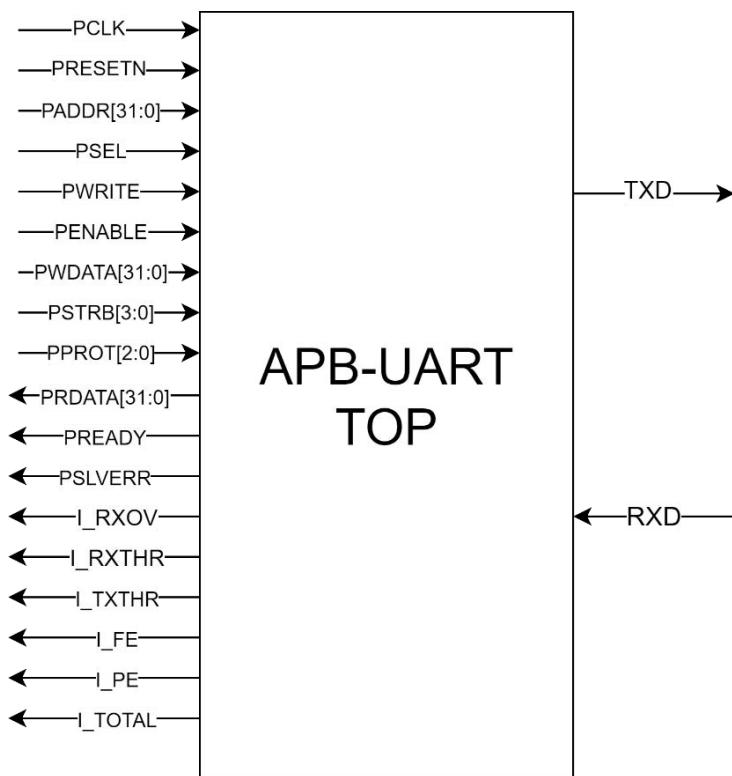


Hình 3.32 Ví dụ về quá trình hoạt động của Receiver FSM

Có thể thấy theo hình trên, vì thời điểm bắt đầu nâng giá trị counter là tại giữa chu kỳ của start bit, các bit dữ liệu, parity bit và stop bit đều được lấy mẫu tại giữa chu kỳ.

3.2.7 Sơ đồ kết nối bộ chuyển đổi giao thức APB – UART

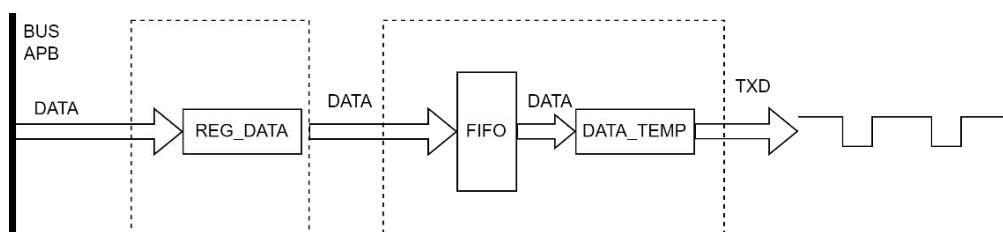
Sau khi thiết các khối chính, bộ chuyển đổi sẽ được hoàn thiện bằng cách kết nối các khối này với nhau. Sơ đồ khái quát của bộ chuyển đổi APB-UART như hình dưới đây:



Hình 3.33. Sơ đồ khái quát bộ chuyển đổi APB-UART

Quá trình hoạt động của bộ chuyển đổi APB-UART được tóm tắt như sau:

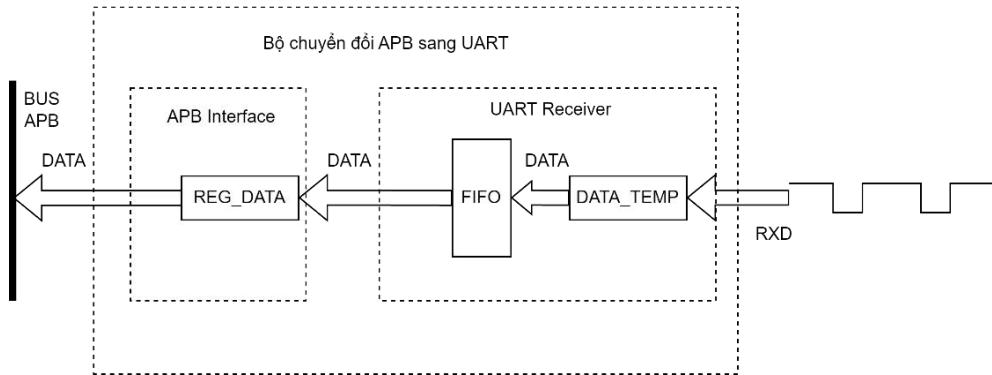
Quá trình truyền dữ liệu được thể hiện như hình 3.34:



Hình 3.34. Tóm tắt quá trình truyền dữ liệu từ APB sang UART

Ở quá trình truyền dữ liệu từ phía bus APB sang UART, dữ liệu được đưa vào thanh ghi REG_DATA. Sau đó, dữ liệu từ thanh ghi được đưa qua bộ FIFO, đưa tới thanh ghi DATA_TEMP và cuối cùng là chuyển sang dạng dữ liệu nối tiếp ở ngõ ra TXD.

Quá trình nhận dữ liệu được thể hiện như hình 3.35:



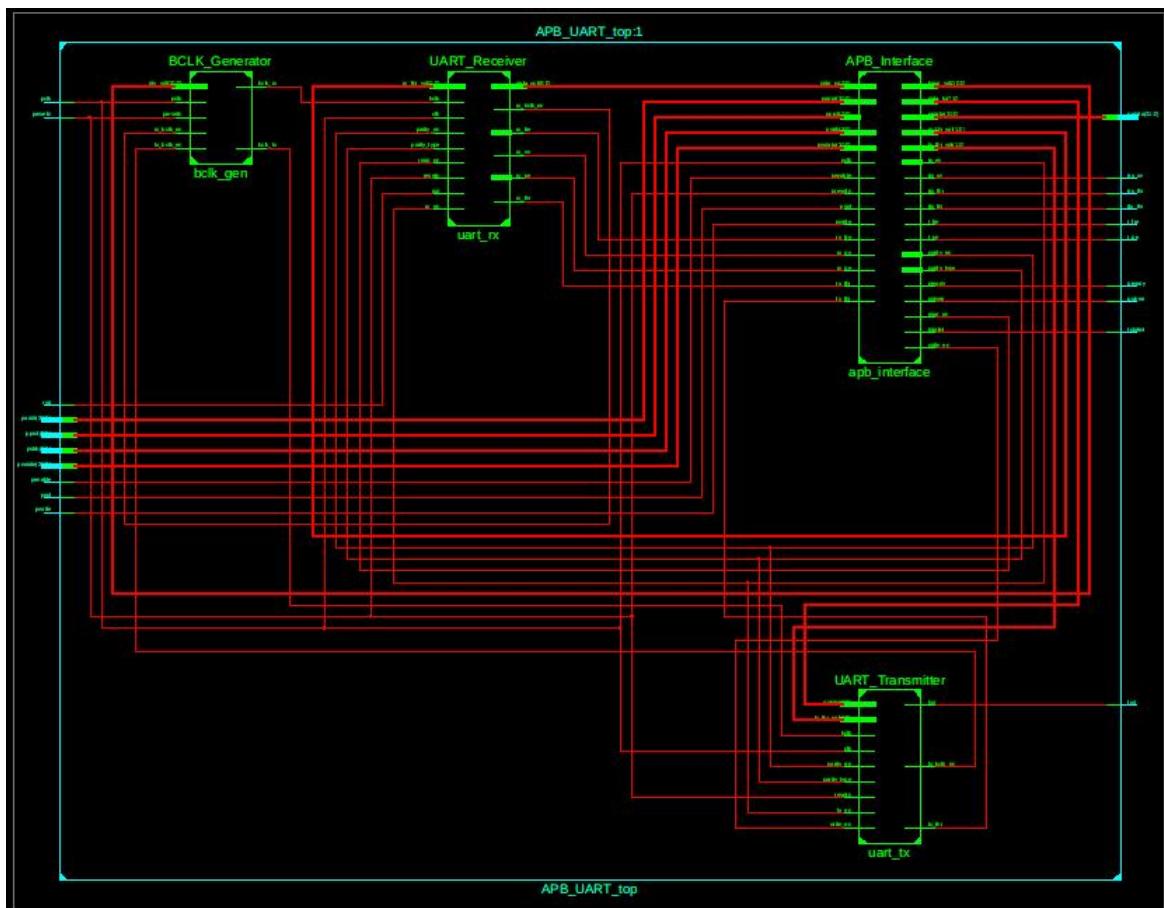
Hình 3.35. Tóm tắt quá trình nhận dữ liệu từ UART sang APB

Ở quá trình nhận dữ liệu từ phía UART sang APB, dữ liệu được đưa vào thanh ghi DATA_TEMP. Sau đó, dữ liệu từ thanh ghi được đưa qua bộ FIFO, kế tiếp là đưa tới thanh ghi REG_DATA và cuối cùng được đọc bởi phía bus APB.

CHƯƠNG 4 KẾT QUẢ VÀ ĐÁNH GIÁ

4.1 SƠ ĐỒ KẾT NỐI BỘ CHUYỂN ĐỔI GIAO THỨC APB SANG UART

Sau khi thiết kế, lập trình và tổng hợp các khối chức năng bằng phần mềm Xilinx ISE, bộ chuyển đổi giao thức APB sang UART gồm các khối APB Interface, BCLK Generator, UART Receiver và UART Transmitter được kết nối hoàn chỉnh với nhau như hình 4.1.



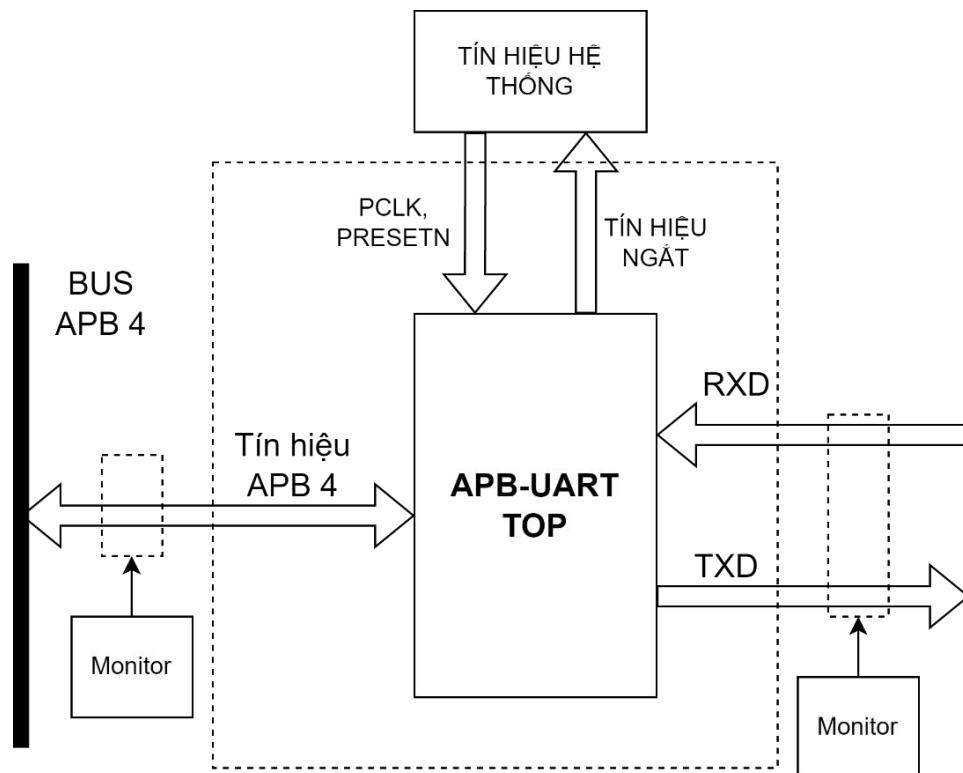
Hình 4.1. Sơ đồ kết nối các khối trong APB-UART TOP

Trong đó, các đường tín hiệu phía bus APB sẽ được nối với khối APB Interface. Các đường RXD và TXD sẽ lần lượt nối với các khối UART Receiver và UART Transmitter.

4.2 KẾT QUẢ XÁC MINH VÀ KIỂM THỬ CHỨC NĂNG CỦA BỘ CHUYỂN ĐỔI GIAO THỨC APB SANG UART

4.2.1 Xây dựng môi trường mô phỏng để xác minh và kiểm thử chức năng bộ chuyển đổi giao thức APB sang UART

Để việc xác minh và kiểm thử các chức năng của bộ chuyển đổi được thực hiện chính xác, một môi trường mô phỏng đã được xây dựng nhằm đáp ứng các yêu cầu trên. Cấu trúc của môi trường mô phỏng được thể hiện theo hình 4.2:



Hình 4.2. Môi trường mô phỏng bộ chuyển đổi giao thức APB sang UART

Trong đó, các khối APB MASTER, UART DEVICE, TÍN HIỆU HỆ THỐNG được mô tả bằng các test case.

Môi trường mô phỏng đã được xây dựng với tần số hoạt động là 100MHz tương ứng với một chu kỳ xung PCLK bằng 10 ns. Tốc độ truyền được chọn mặc định ở quá trình này là 460800 bit/s tương ứng với giá trị được gửi vào thanh ghi REG_BCLK là 14.

4.2.2 Kết quả kiểm thử quá trình truyền dữ liệu

- Các trường hợp kiểm thử chức năng của quá trình truyền:

Các trường hợp (testcase) kiểm thử được liệt kê và mô tả cụ thể trong bảng 4.1.

Bảng 4.1. Các testcase cho truyền dữ liệu

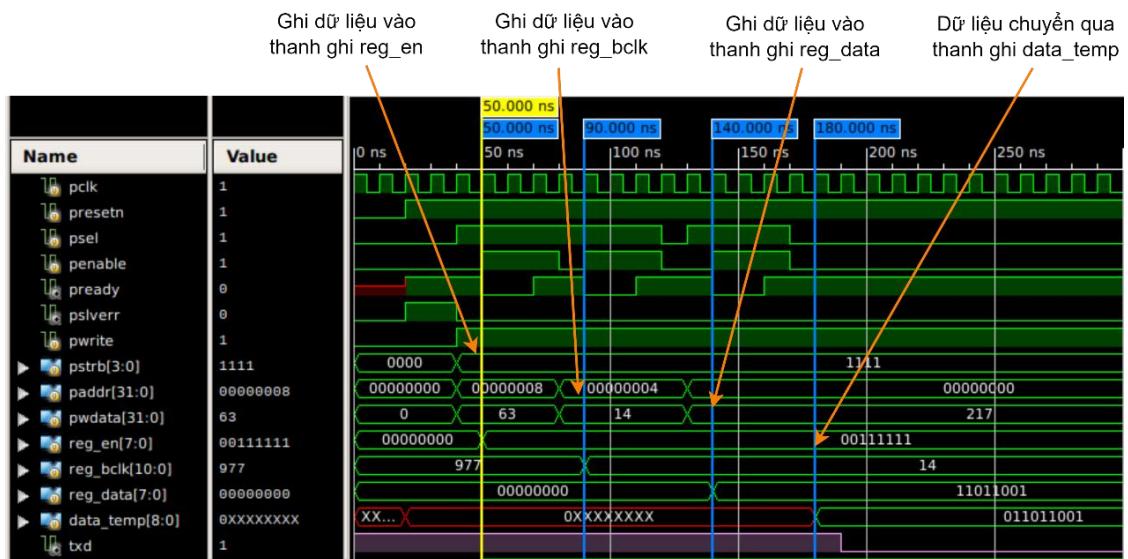
| Tên testcase | Mô tả quá trình |
|---------------------------------------|---|
| Truyền dữ liệu không có chế độ Parity | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 00111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> <p>Sau đó, thiết lập giá trị 14 (tốc độ truyền = 480600) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004.</p> <p>Cuối cùng thiết lập giá trị 11011001 cho thanh ghi REG_DATA ở địa chỉ 0x00.</p> |
| Truyền dữ liệu có chế độ Parity chẵn | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 11111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> <p>Sau đó, thiết lập giá trị 14 (tốc độ truyền = 480600) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004.</p> <p>Cuối cùng thiết lập giá trị 11011001 cho thanh ghi REG_DATA ở địa chỉ 0x00000000.</p> |
| Truyền dữ liệu có chế độ Parity lẻ | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> |

| | |
|---|--|
| | Nâng PRESETN lên mức cao. Thiết lập giá trị 0111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008. Sau đó, thiết lập giá trị 14 (tốc độ truyền = 480600) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004. Cuối cùng thiết lập giá trị 11011001 cho thanh ghi REG_DATA ở địa chỉ 0x00000000. |
| Truyền 2 dữ liệu liên tiếp | Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp. Nâng PRESETN lên mức cao. Thiết lập giá trị 0111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008. Sau đó, thiết lập giá trị 14 (tốc độ truyền = 480600) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004. Cuối cùng thiết lập hai giá trị 11011001 và giá trị 01010110 liên tiếp cho thanh ghi REG_DATA ở địa chỉ 0x00000000. |
| Truyền 2 dữ liệu liên tiếp với tốc độ truyền 115200 | Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp. Nâng PRESETN lên mức cao. Thiết lập giá trị 0011111 cho thanh ghi REG_EN tại địa chỉ 0x00000008. Sau đó, thiết lập giá trị 54 (tốc độ truyền = 115200) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004, điều này tương đương với tốc độ |

| | |
|--|---|
| | <p>truyền là 115200 ở tần số hoạt động 100MHz.</p> <p>Cuối cùng thiết lập giá trị 11011001 cho thanh ghi REG_DATA ở địa chỉ 0x00000000.</p> |
|--|---|

• Kết quả truyền dữ liệu không có chế độ Parity

Kết quả truyền dữ liệu từ phía APB sang phía UART bao gồm 2 giai đoạn, đó là giai đoạn thiết lập và kết quả quá trình truyền. Giai đoạn thiết lập được thể hiện ở hình 4.3:



Hình 4.3. Kết quả thiết lập thông số truyền không có chế độ Parity

Quá trình thiết lập các thông số được thể hiện ở hình 4.3. Từ 0ns đến 20ns, tín hiệu PRESETN ở mức thấp để tiến hành khởi động lại hệ thống, TxD được giữ ở mức cao.

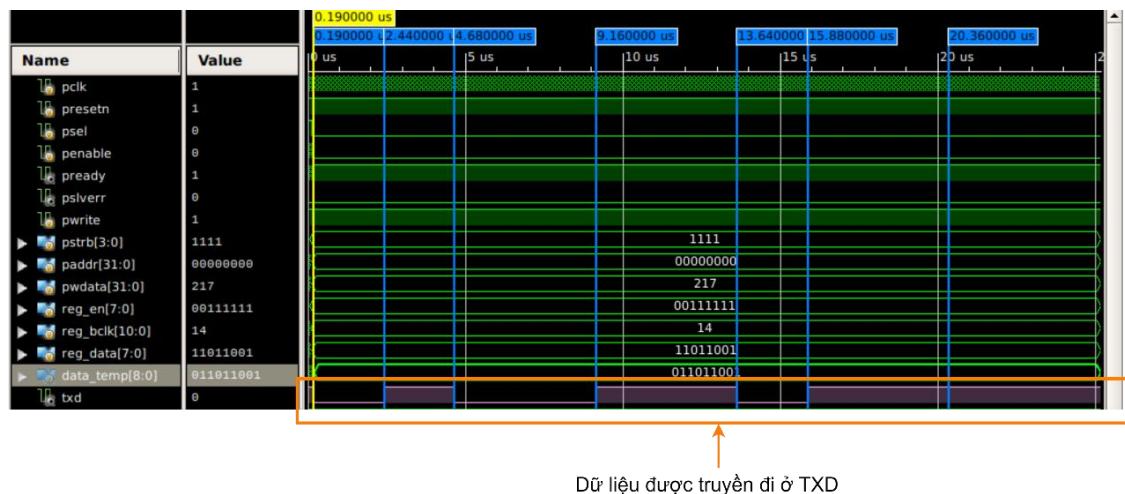
Từ 40ns đến 80ns, quá trình ghi có trạng thái để thiết lập thông số cho thanh ghi cho phép được bắt đầu. Tại 40 ns, PSEL = 1, báo hiệu một quá trình truyền bắt đầu, khi này PADDR được thiết lập giá trị là 0x00000008, PWDATA được thiết lập giá trị là 63 (00111111). Tại chy kỳ kế tiếp, PENABLE được nâng lên mức cao, dữ liệu từ PWDATA được đưa vào thanh ghi REG_EN. Như vậy, tín hiệu cho phép truyền nhận Parity ở mức thấp.

Từ 80ns đến 120, quá trình ghi có trạng thái chờ với mục đích là thiết lập thông số cho thanh ghi REG_BCLK được bắt đầu. Tại 80 ns, PENABLE được kéo xuống mức thấp. Lúc này PADDR được thiết lập giá trị 0x00000004, PWDATA được thiếp lập giá trị 14. Tại 90 ns, dữ liệu từ PWDATA được đưa vào thanh ghi REG_BCLK. Sau đó, PENABLE và PSEL được kéo xuống mức thấp để kết thúc quá trình ghi.

Từ Tại 130 ns, PSEL được kéo lên cao trở lại, bắt đầu quá trình ghi dữ liệu để truyền đi. Tại 140 ns, PENABLE được nâng lên mức cao, thanh ghi REG_DATA tiếp nhận dữ liệu ở 8 bit đầu tiên của tín hiệu PWDATA với giá trị nhị phân là 11011001.

Sau 4 chu kỳ, tại thời điểm 180 ns, thanh ghi DATA_TEMP ở khối UART Transmitter đã tiếp nhận dữ liệu và gắn vào 8 bit đầu tiên của mình.

Kết tiếp là kết quả truyền dữ liệu ở chân TXD được thể hiện trong hình 4.4:



Hình 4.4. Kết quả truyền dữ liệu không có chế độ Parity

Dữ liệu cần truyền đi trong quá trình này là 11011001, dựa vào điều kiện này có thể phân tích sóng ngõ ra tại TXD dựa theo hình 4.3 như sau:

Tại thời điểm 0.19 us, TXD được kéo xuống mức thấp, báo hiệu quá trình truyền bắt đầu. Tới thời điểm 2.44 us, TXD được kéo lên mức cao, bit 0 bắt đầu được truyền. Start bit có độ dài là 225 us.

Từ 2.44 us đến 4.68 us, tương ứng với khoảng thời gian là 224 us, TXD được kéo lên mức cao hay bit 0 được truyền.

Từ 4.68 us đến 9.16 us, tương ứng với khoảng thời gian là 448 us, TxD được kéo xuống mức thấp hay bit 1 và bit 2 được truyền.

Từ 9.16 us đến 13.64 us, tương ứng với khoảng thời gian là 448 us, TxD được kéo lên mức cao hay bit 3 và bit 4 được truyền.

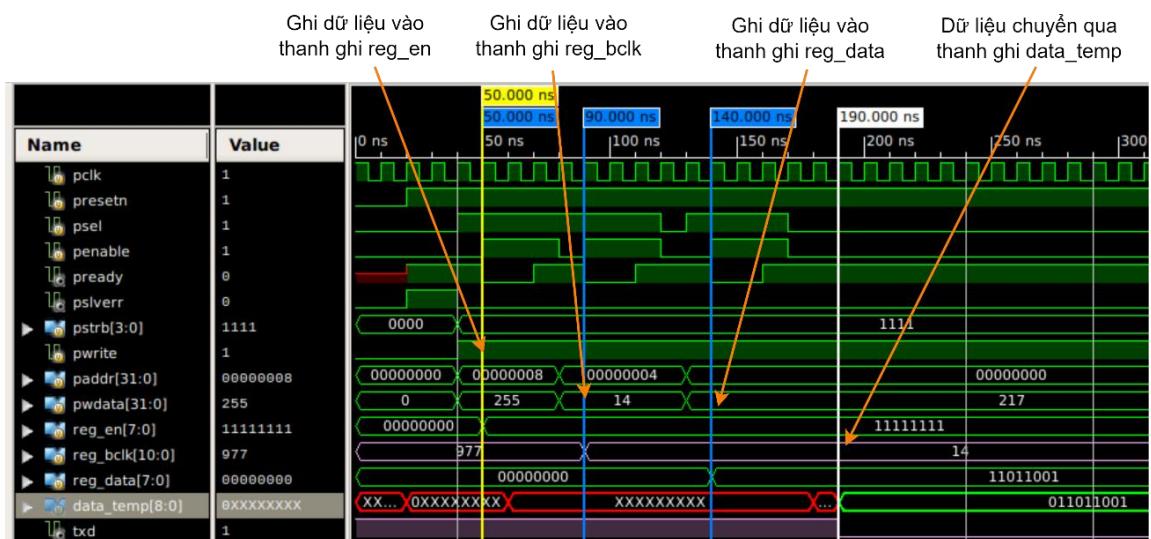
Từ 13.64 us đến 15.88 us, tương ứng với khoảng thời gian là 224 us, TxD được kéo xuống mức thấp hay bit 5 được truyền.

Từ 15.88 us đến 22.6 us, tương ứng với khoảng thời gian là 672 us, TxD được kéo lên mức cao hay bit 6, bit 7 và Stop bit được truyền.

Như vậy, từ thời điểm 0.19 us đến 22.6 us, các bit lần lượt là start bit, 8 bit dữ liệu là 11011001 và stop bit đã được truyền đi.

- Kết quả truyền dữ liệu có chế độ Parity chẵn**

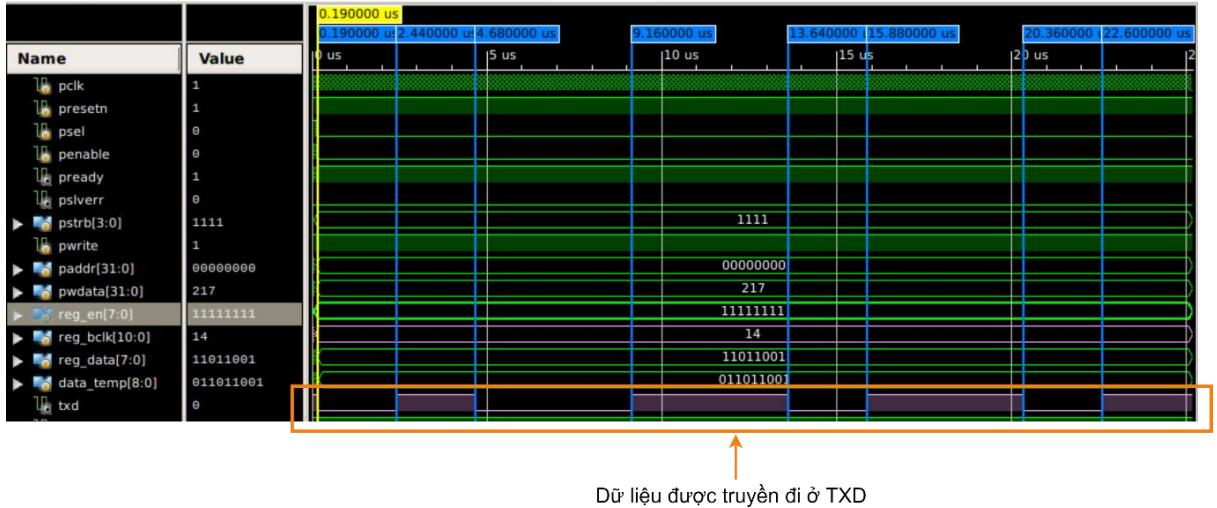
Quá trình thiết lập các thông số được thể hiện ở hình 4.5:



Hình 4.5. Kết quả thiết lập thông số truyền có chế độ Parity chẵn

Quá trình thiết lập các thông số giống ở đây tương tự như quá trình truyền không có parity, tuy nhiên, giá trị thiết lập cho thanh ghi REG_EN ở trường hợp này là 255 hay 11111111, tương ứng với tín hiệu cho phép truyền nhận parity ở mức cao và chế độ truyền là parity chẵn. Khi này, dữ liệu ở thanh ghi DATA_TEMP của khôi Transmitter giá trị là 01011001 tại thời điểm 190 ns, với 8 bit đầu tương ứng với 8 bit dữ liệu và bit cuối tương ứng với parity bit.

Kế tiếp là kết quả truyền dữ liệu ở chân TxD được thể hiện trong hình 4.6:



Hình 4.6. Kết quả truyền dữ liệu có chế độ Parity chẵn

Dữ liệu cần truyền đi trong quá trình này là 11011001 kèm với parity bit ở đây là 0, dựa vào điều kiện này có thể phân tích sóng ngõ ra tại TXD dựa theo hình 4.6 như sau:

Tại thời điểm 0.19 us, TXD được kéo xuống mức thấp, báo hiệu quá trình truyền bắt đầu. Tới thời điểm 2.44 us, TXD được kéo lên mức cao, đồng nghĩa với bit 0 bắt đầu được truyền. Start bit có độ dài là 225 us.

Từ 2.44 us đến 4.68 us, tương ứng với khoảng thời gian là 2.24 us, TXD được kéo lên mức cao hay bit 0 được truyền.

Từ 4.68 us đến 9.16 us, tương ứng với khoảng thời gian là 4.48 us, TXD được kéo xuống mức thấp hay bit 1 và bit 2 được truyền.

Từ 9.16 us đến 13.64 us, tương ứng với khoảng thời gian là 4.48 us, TXD được kéo lên mức cao hay bit 3 và bit 4 được truyền.

Từ 13.64 us đến 15.88 us, tương ứng với khoảng thời gian là 2.24 us, TXD được kéo xuống mức thấp hay bit 5 được truyền.

Từ 15.88 us đến 20.36 us, tương ứng với khoảng thời gian là 4.48 us, TXD được kéo lên mức cao hay bit 6 và bit 7 được truyền.

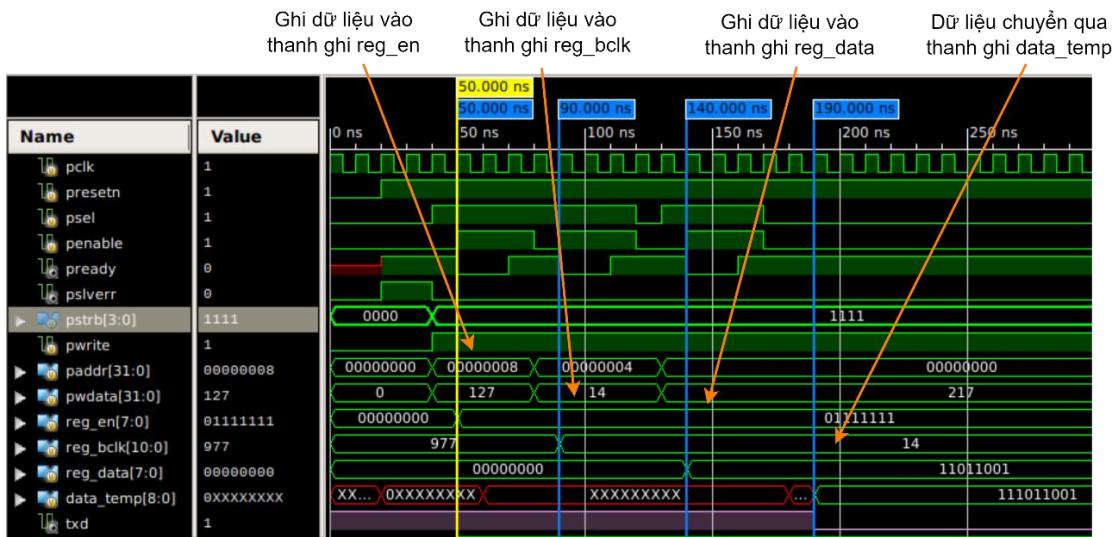
Từ 20.36 us đến 22,6 us, tương ứng với khoảng thời gian là 2.24 us, TXD được kéo xuống mức thấp đồng nghĩa với parity bit được truyền. Vì đang ở chế độ parity chẵn, dữ liệu 11011001 có 5 bit 1 nên parity bit là bit 0.

Từ 22.6 us trở đi, TXD được nâng lên mức cao, tương ứng với stop bit được truyền đi.

Như vậy, từ thời điểm 0.19 us đến 22.6 us, các bit lần lượt là start bit, 8 bit dữ liệu là 110111001 và parity bit đã được truyền đi.

- **Kết quả truyền dữ liệu có chế độ Parity lẻ**

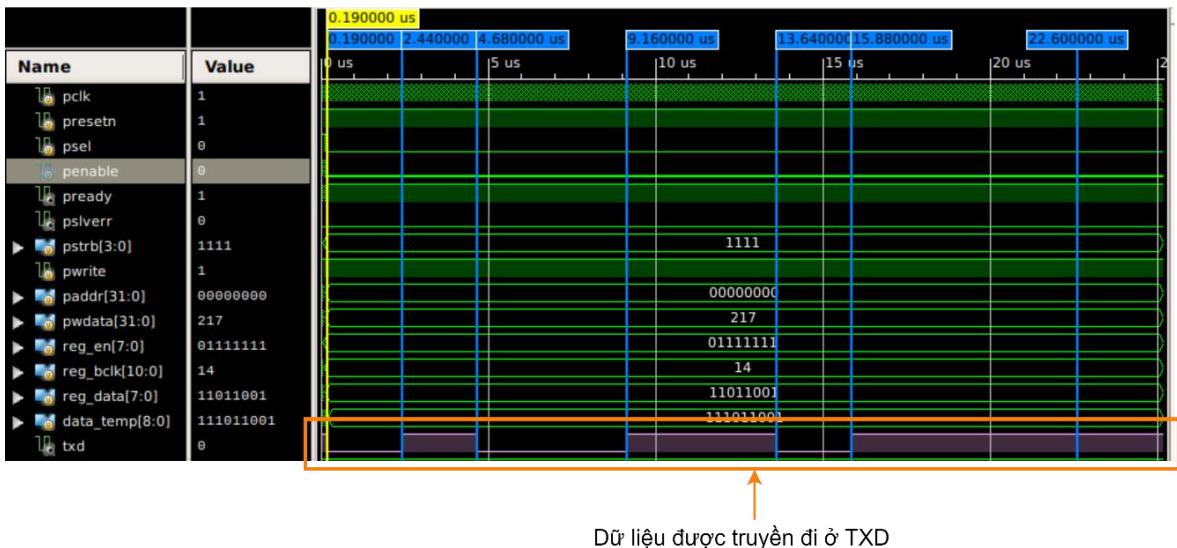
Quá trình thiết lập các thông số được thể hiện ở hình 4.7:



Hình 4.7. Kết quả thiết lập thông số truyền có chế độ Parity lẻ

Quá trình thiết lập các thông số giống như tại quá trình truyền không có parity, tuy nhiên, tại 50ns, giá trị thiết lập cho thanh ghi REG_EN ở trường hợp này là 127 (8'b01111111), tương ứng với tín hiệu cho phép truyền nhận Parity ở mức cao và thực hiện ở chế độ parity lẻ. Tại 190 ns, giá trị thanh ghi DATA_TEMP tại transmitter là 111011001 tương ứng với parity bit khi này là 1.

Ké tiếp là kết quả truyền dữ liệu ở chân TxD được thể hiện trong hình 4.8:



Hình 4.8. Kết quả truyền dữ liệu có chế độ Parity lẻ

Dữ liệu cần truyền đi trong quá trình này là 11011001 kèm với parity bit ở đây là 1, dựa vào điều kiện này có thể phân tích sóng ngõ ra tại TXD dựa theo hình 4.8 như sau:

Tại thời điểm 0.19 us, TXD được kéo xuống mức thấp, báo hiệu quá trình truyền bắt đầu. Tới thời điểm 2.44 us, TXD được kéo lên mức cao, đồng nghĩa với bit 0 bắt đầu được truyền. Start bit có độ dài là 225 us.

Từ 2.44 us đến 4.68 us, tương ứng với khoảng thời gian là 224 us, TXD được kéo lên mức cao hay bit 0 được truyền.

Từ 4.68 us đến 9.16 us, tương ứng với khoảng thời gian là 448 us, TXD được kéo xuống mức thấp hay bit 1 và bit 2 được truyền.

Từ 9.16 us đến 13.64 us, tương ứng với khoảng thời gian là 448 us, TXD được kéo lên mức cao hay bit 3 và bit 4 được truyền.

Từ 13.64 us đến 15.88 us, tương ứng với khoảng thời gian là 224 us, TXD được kéo xuống mức thấp hay bit 5 được truyền.

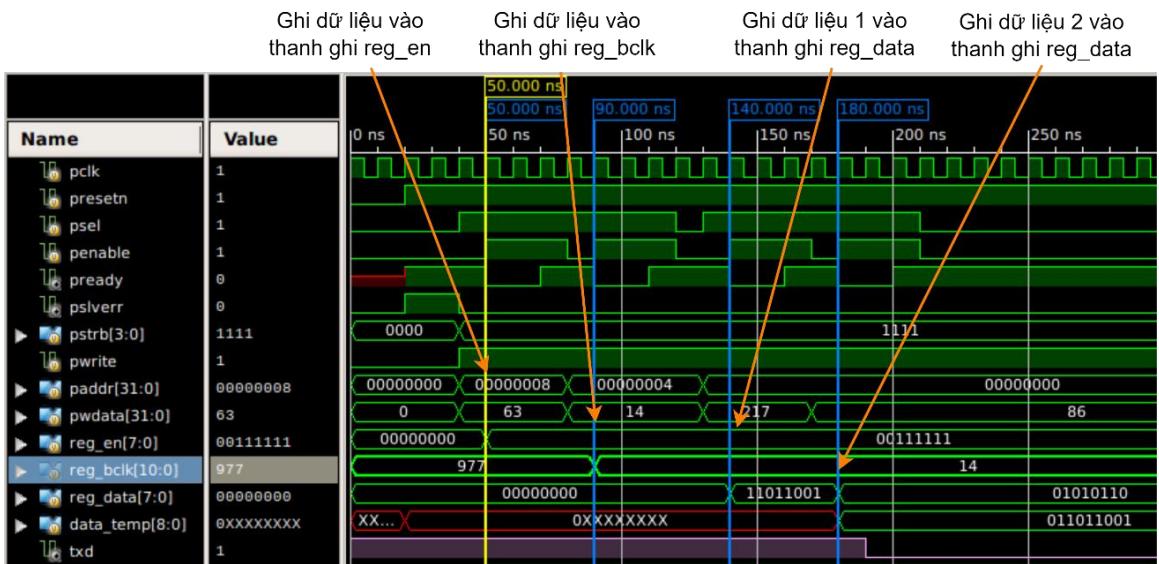
Từ 15.88 us đến 22.6 us, tương ứng với khoảng thời gian là 672 us, TXD được kéo lên mức cao, bit 6, bit 7 và parity bit được truyền đi.

Từ 22.6 us trở đi, TXD được kéo lên mức cao hay stop bit được truyền.

Như vậy, từ thời điểm 0.19 us đến 22.6 us, các bit lần lượt là start bit, 8 bit dữ liệu là 11011001 và parity bit đã được truyền đi.

- **Kết quả truyền hai dữ liệu liên tiếp**

Quá trình thiết lập các thông số được thể hiện như hình 4.9:

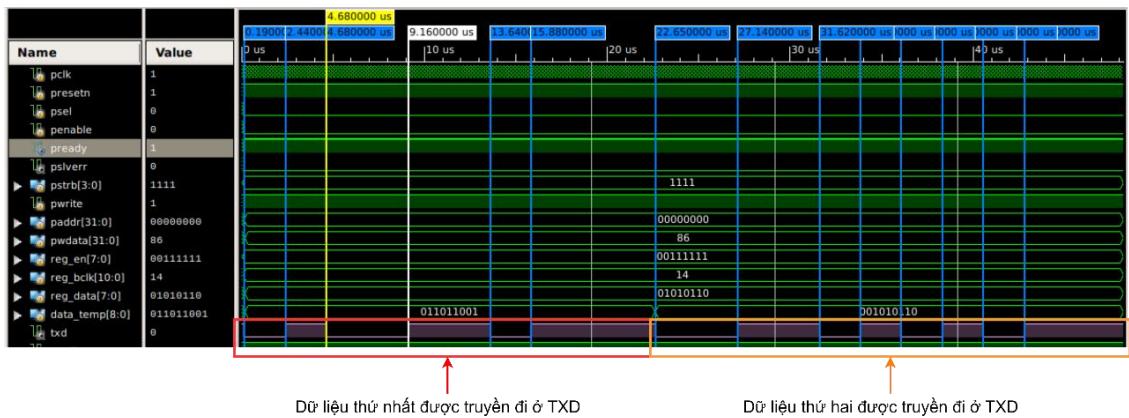


Hình 4.9. Kết quả thiết lập thông số truyền hai dữ liệu

Quá trình thiết lập các thông số tại thanh ghi REG_EN và REG_BCLK tương tự như quá trình truyền dữ liệu không có chế độ parity.

Tại thời điểm 140 ns, giá trị 11011001 được thiết lập cho thanh ghi REG_DATA. Sau đó 4 chu kỳ xung PCLK, tại thời điểm 180 ns, giá trị 01010110 được thiết lập cho REG_DATA. Khi này thanh ghi DATA_TEMP cũng đã tiếp nhận dữ liệu từ trước đó là 11011001.

Kết quả quá trình truyền hai dữ liệu như sau:



Hình 4.10. Kết quả truyền hai dữ liệu liên tiếp

Tại thời điểm 0.19 us tới 2.44us, TXD được kéo xuống mức thấp, start bit được truyền báo hiệu cho việc chuẩn bị truyền dữ liệu thứ nhất là 11011001.

Từ 2.44 us đến 4.68 us, TXD được kéo lên mức cao, bit 0 được truyền. Từ 4.68 us đến 9.16 us, TXD được kéo xuống mức thấp, bit 1 và 2 được truyền. Từ 9.16 us đến 13.64 us, TXD được kéo lên mức cao, bit 3 và 4 được truyền. Từ 13.64 us đến 15.88 us, TXD được kéo xuống mức thấp, bit 5 được truyền. Từ 15.88 us đến 22.65 us, bit 6, bit 7 và stop bit được truyền đi. Như vậy, từ thời điểm 0.19 us đến 22.65 us, dữ liệu thứ nhất đã được truyền đi.

Tại thời điểm 22.65, TXD được kéo xuống mức thấp, start bit được truyền báo hiệu cho việc chuẩn bị truyền dữ liệu thứ hai là 01010110.

Từ 22.65 đến 27.14, TXD được kéo xuống mức thấp, start và bit 0 được truyền đi. Từ 27.14 us đến 31.62 us, TXD được kéo lên mức cao, bit 1 và 2 được truyền. Từ 31.62 us đến 33.86 us, TXD được kéo xuống mức thấp, bit 3 được truyền. Từ 33.86 us đến 36.1 us, TXD được kéo lên mức cao, bit 4 được truyền. Từ 36.1 us đến 38.34 us, TXD được kéo xuống mức thấp, bit 5 được truyền đi. Từ 38.34 us đến 40.58 us, TXD được kéo lên mức cao, bit 6 được truyền đi. Từ 40.58 us đến 42.82 us, TXD được kéo xuống mức thấp, bit 7 được truyền đi. Như vậy, từ thời điểm 22.65 us đến 42.82 us, dữ liệu thứ hai đã được truyền đi.

Có thể thấy trong quá trình truyền hai dữ liệu liên tiếp, có độ trễ giữa hai quá trình truyền. Với số xung PCLK cho mỗi BCLK là 14, như vậy với mỗi bit truyền có độ dài tương ứng với 224 xung PCLK hay 2.24 us.

Nếu chúng ta xét thời điểm bắt đầu truyền là 0.19 us đến khi TXD được kéo xuống thấp cho lần truyền thứ hai là 22.65 us, khoảng thời gian có được của lần truyền thứ nhất là 22.46 us. Trong khi đó, thời gian truyền cho dữ liệu thứ nhất được dự kiến là 22.4 us tương đương với thời gian truyền 10 bit, bao gồm start bit, 8 bit dữ liệu và stop bit. Như vậy thời gian truyền thực tế trễ hơn so với dự kiến là 0.6 us. Số liệu trên có thể tóm tắt ở bảng 4.2:

Bảng 4.2. So sánh thời gian truyền thực tế và dự kiến

| | Dự kiến | Thực tế |
|-----------|---------|---------|
| Thời gian | 22.4 us | 22.46 |

Như vậy, tốc độ truyền trên thực tế chậm hơn 0.268 % so với dự kiến.

- **Kết quả truyền hai dữ liệu liên tiếp với tốc độ truyền khác**

Quá trình thiết lập thông số được thể hiện ở hình 4.11:



Hình 4.11. Kết quả thiết lập thông số truyền hai dữ liệu với tốc độ truyền khác

Ngoại trừ số 54 được ghi vào thanh ghi REG_BCLK thay vì 14, tương đương với tốc độ truyền là 115200 460800 thay cho 460800 bits/giây, các quá thông số còn lại đều tương tự như ở quá truyền 2 dữ liệu liên tiếp ở tốc độ cũ.

Kết quả của việc truyền dữ liệu như sau:



Hình 4.12. Kết quả truyền hai dữ liệu với tốc độ truyền khác

Dựa theo hình 4.12 có thể thấy rằng việc truyền dữ liệu thứ nhất là 11011001 và dữ liệu thứ 2 là 01010110 đúng như theo chuẩn giao thức UART.

Trong đó, khoảng thời gian từ lúc TXD được kéo xuống mức thấp để truyền start bit của dữ liệu thứ nhất đến khi kéo xuống lần nữa để truyền start bit cho dữ liệu thứ hai là từ 0.19 us đến 86.65 us, tương đương với 86.46 us. Trong khi đó, với tốc độ truyền là 115200, mỗi bit mất khoảng thời gian là 8.64 us, toàn bộ dữ

liệu thứ nhất truyền đi mất khoảng 86.4 us. Như vậy, tốc độ truyền thực tế chậm hơn 0.069% so với tốc độ dự kiến.

4.2.3 Kết quả kiểm thử quá trình nhận dữ liệu

- Các trường hợp kiểm thử chức năng của quá trình nhận**

Các trường hợp (testcase) kiểm thử được liệt kê và mô tả cụ thể trong bảng 4.3:

Bảng 4.3. Các testcase cho nhận dữ liệu

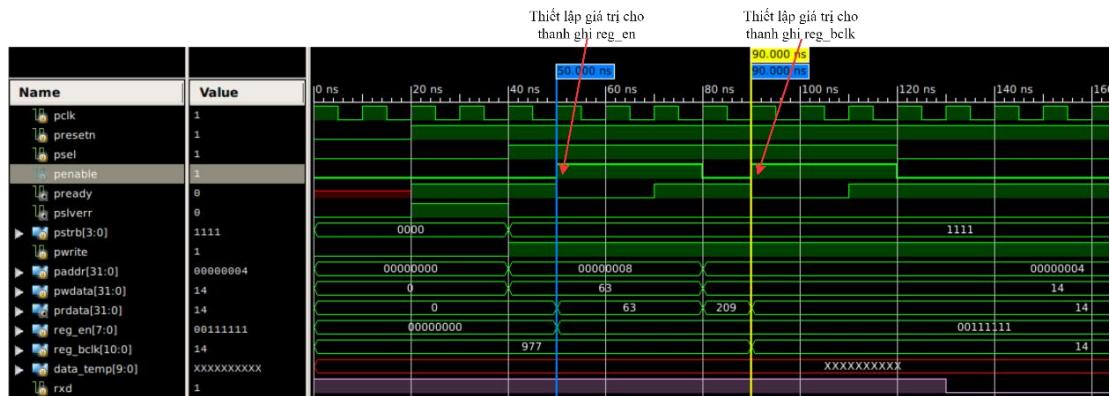
| Tên testcase | Mô tả quá trình |
|-------------------------------------|--|
| Nhận dữ liệu không có chế độ Parity | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 00111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> <p>Sau đó, thiết lập giá trị 14 (Tốc độ truyền = 480600) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004.</p> <p>Phía RXD nhận được dữ liệu là 11011001. Sau khoảng thời gian đó, thực hiện đọc thanh ghi REG_DATA.</p> |
| Nhận dữ liệu có chế độ Parity chẵn | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 11111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> <p>Sau đó, thiết lập giá trị 14 (Tốc độ truyền = 480600) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004.</p> <p>Phía RXD nhận được dữ liệu là 11011011 với parity bit là 1. Sau</p> |

| | |
|--|--|
| | khoảng thời gian đó, thực hiện đọc thanh ghi REG_DATA. |
| Nhận dữ liệu có chế độ Parity lẻ | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 01111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> <p>Sau đó, thiết lập giá trị 14 (Tốc độ truyền = 480600) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004.</p> <p>Phía RXD nhận được dữ liệu là 01100110 với parity bit là 0. Sau khoảng thời gian đó, thực hiện đọc thanh ghi REG_DATA.</p> |
| Nhận 2 dữ liệu liên tiếp | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 00111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> <p>Sau đó, thiết lập giá trị 14 (Tốc độ truyền = 480600) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004.</p> <p>Phía RXD nhận được dữ liệu là 11011011. Sau đó, RXD nhận tiếp dữ liệu là 01100110.</p> |
| Nhận dữ liệu với tốc độ Tốc độ truyền 115200 | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 00111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> |

| | |
|--------------------|---|
| | Sau đó, thiết lập giá trị 54 (Tốc độ truyền = 115200) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004. Phía RXD nhận được dữ liệu là 11011001. Sau khoảng thời gian đó, thực hiện đọc thanh ghi REG_DATA. |
| Nhận start bit giả | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp. Nâng PRESETN lên mức cao. Thiết lập giá trị 00111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008. Sau đó, thiết lập giá trị 54 (Tốc độ truyền = 115200) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004. Phía RXD kéo xuống mức thấp trong khoảng thời gian là 1 us (bé hơn $\frac{1}{2}$ chu kỳ start bit), sau đó kéo lên lại mức cao.</p> |

- Kết quả nhận dữ liệu không có chế độ Parity**

Quá trình thiết lập các thông số được thể hiện ở hình 4.13:



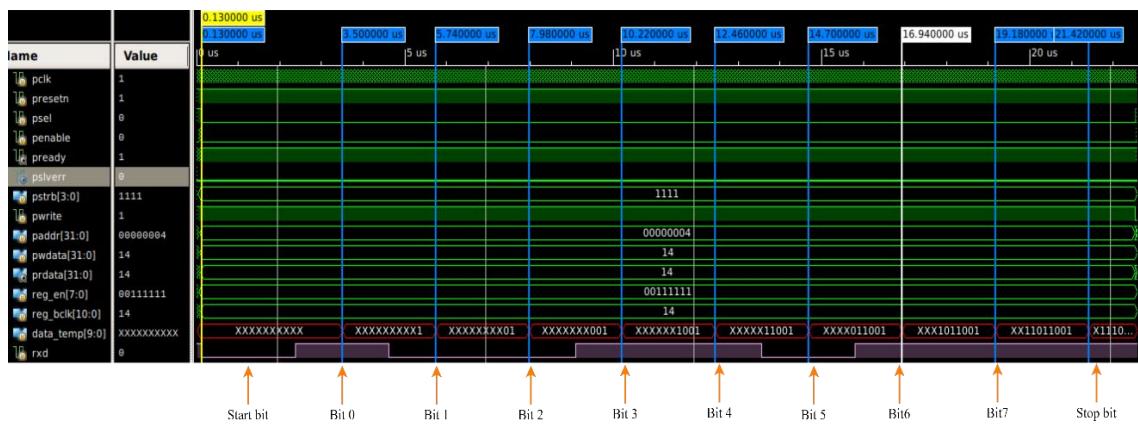
Hình 4.13. Kết quả thiết lập thông số nhận không có chế độ Parity

Từ 0ns đến 20ns, tín hiệu PRESETN ở mức thấp để tiến hành khởi động lại hệ thống, READY và PSLVERR đều ở mức thấp

Từ 40ns đến 80ns là một quá trình ghi có trạng thái chờ Trong đó, PSEL được kéo lên mức cao ở thời điểm 40ns để lựa chọn thiết bị và bắt đầu quá trình ghi, địa chỉ PADDR được thiết lập giá trị là 00000008, pwdata được thiết lập giá trị là 63 (8'b00111111). Ở chu kỳ PCLK kế tiếp, PENABLE được đưa lên mức cao, dữ liệu ở PWpdata được đưa vào thanh ghi REG_EN tại thời điểm 50 ns.

Từ 80ns đến 120ns là một quá trình ghi có trạng thái chờ, địa chỉ PADDR được thiết lập giá trị 00000004 và PWpdata được thiết lập giá trị 14. Giá trị 14 được đưa vào thanh ghi REG_BCLK tại thời điểm 50 ns để thiết lập tốc độ truyền là 480600.

Sau khi thiết lập, RXD được nhận dữ liệu như hình 4.14:



Hình 4.14. Quá trình nhận dữ liệu không có chế độ Parity

Tại thời điểm 0.13 us, RXD được kéo xuống mức thấp, start bit bắt đầu được nhận. Quá trình nhận dữ liệu được tóm gọn theo bảng 4.4:

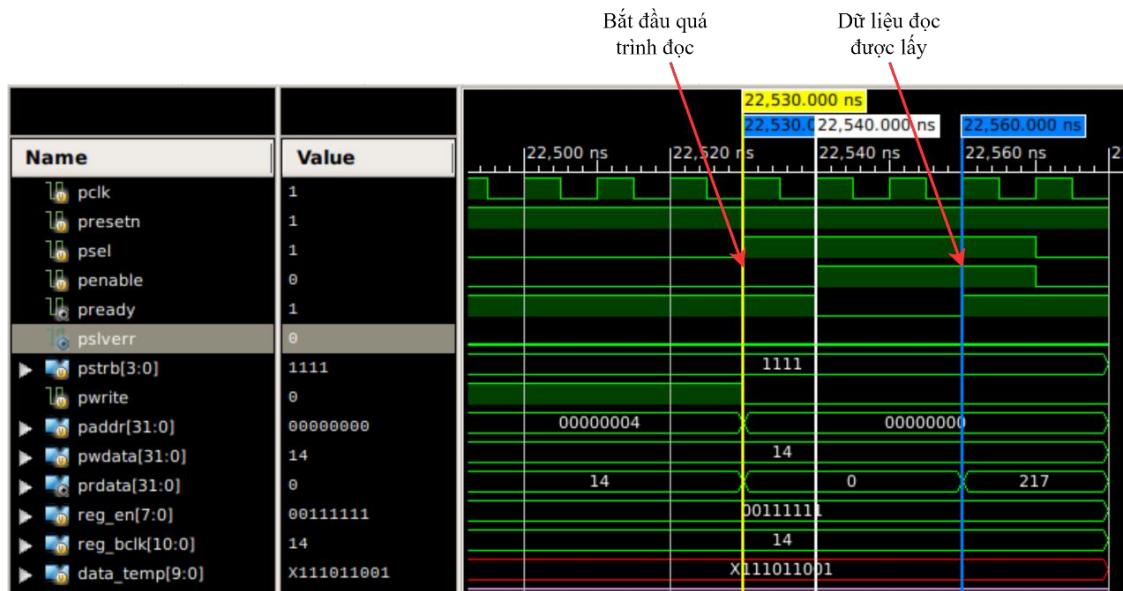
Bảng 4.4. Dữ liệu nhận vào thanh ghi không có Parity

| Thời điểm | Giá trị | Vị trí bit trong thanh ghi DATA_TEMP |
|-----------|---------|--------------------------------------|
| 3.38 us | 1 | Bit 0 |
| 5.74 us | 0 | Bit 1 |
| 7.98 us | 0 | Bit 2 |
| 10.22 us | 1 | Bit 3 |
| 12.46 us | 1 | Bit 4 |
| 14.7 us | 0 | Bit 5 |

| | | |
|----------|---|-------|
| 16.94 us | 1 | Bit 6 |
| 19.18 us | 1 | Bit 7 |
| 21.42 us | 1 | Bit 8 |

Từ đây, 8 bit dữ liệu nhận được là 11011001. Vì quá trình nhận không có chế độ parity nên bit thứ 8 tương đương với stop bit là 1.

Kết quả đọc dữ liệu như hình 4.15:



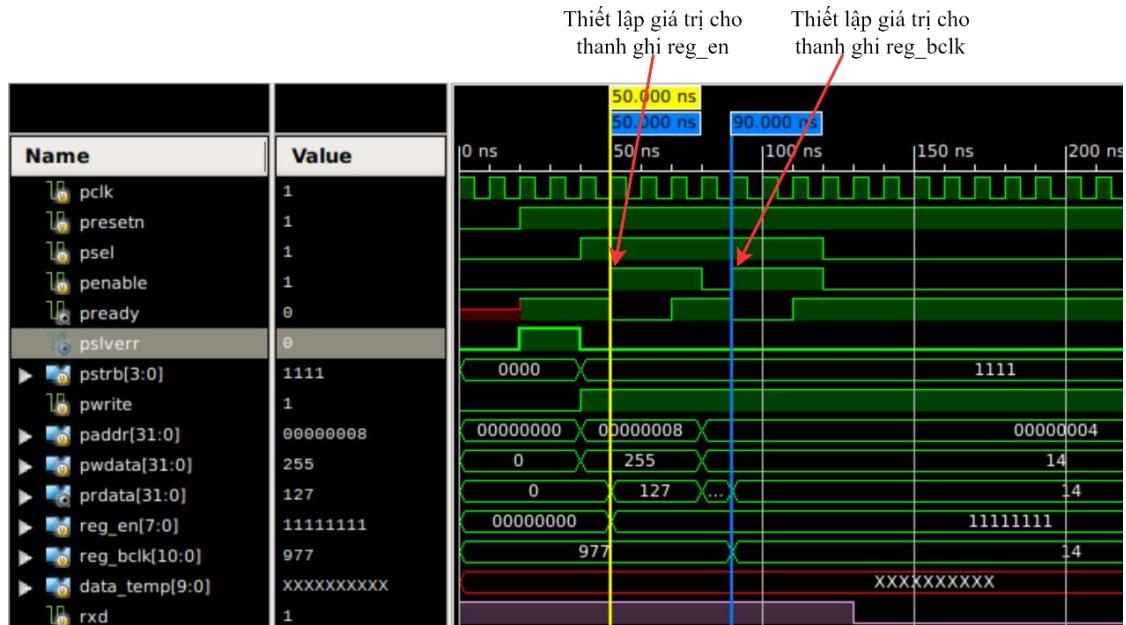
Hình 4.15. Kết quả đọc dữ liệu ở quá trình nhận không có chế độ Parity

Tại thời điểm 22530 ns, PSEL được nâng lên mức cao, PWRITE được hạ xuống mức thấp, bắt đầu quá trình đọc. Tại 22540 ns, PENABLE = 1.

Tại thời điểm 22560, khi này PREADY đã lên mức cao, dữ liệu đọc được đưa ra, khi này, PRDATA có giá trị là 217(8'b11011001). Như vậy, dữ liệu đọc được đúng như dữ liệu nhận được.

- **Kết quả nhận dữ liệu có chế độ Parity chẵn**

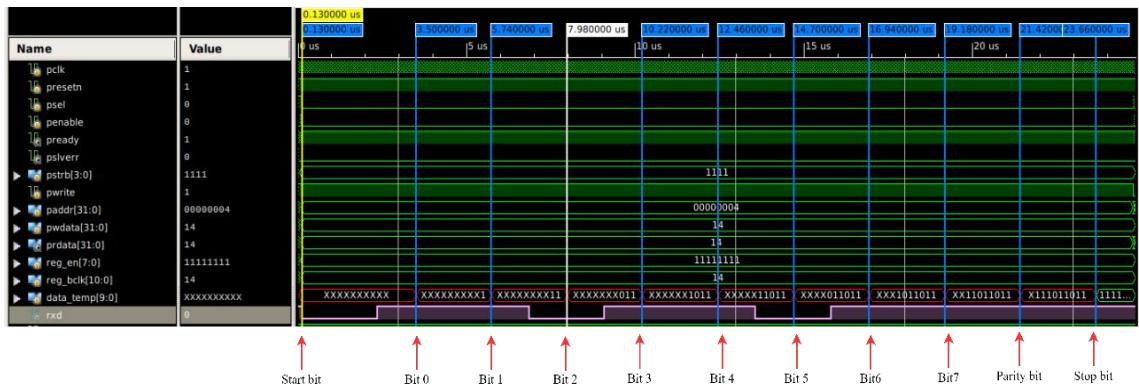
Quá trình thiết lập thông số như hình 4.16:



Hình 4.16. Kết quả thiết lập thông số nhận có chế độ Parity chẵn

Việc thiết lập giống như quá trình nhận không có chế độ Parity, ngoại trừ thông số được ghi vào thanh ghi REG_EN là 11111111 thay vì 00111111.

Sau khi thiết lập, RXD được nhận dữ liệu như hình dưới đây:



Hình 4.17. Quá trình nhận dữ liệu có chế độ Parity chẵn

Tại thời điểm 0.13 us, RXD được kéo xuống mức thấp, start bit bắt đầu được nhận. Quá trình nhận dữ liệu được tóm gọn theo bảng 4.5:

Bảng 4.5. Dữ liệu nhận vào thanh ghi có Parity chẵn

| Thời điểm | Giá trị | Vị trí bit trong thanh ghi DATA_TEMP |
|-----------|---------|--------------------------------------|
| 3.5 us | 1 | Bit 0 |
| 5.74 us | 1 | Bit 1 |

| | | |
|----------|---|-------|
| 7.98 us | 0 | Bit 2 |
| 10.22 us | 1 | Bit 3 |
| 12.46 us | 1 | Bit 4 |
| 14.7 us | 0 | Bit 5 |
| 16.94 us | 1 | Bit 6 |
| 19.18 us | 1 | Bit 7 |
| 21.42 us | 1 | Bit 8 |
| 23.66 us | 1 | Bit 9 |

Có thể thấy rằng, 8 bit dữ liệu nhận được là 11011011. Vì quá trình nhận có chế độ parity chẵn nên bit thứ 8 là parity bit có giá trị là 1. Bit thứ 9 tương đương với stop bit là 1.

Kết quả đọc dữ liệu như hình 4.18:

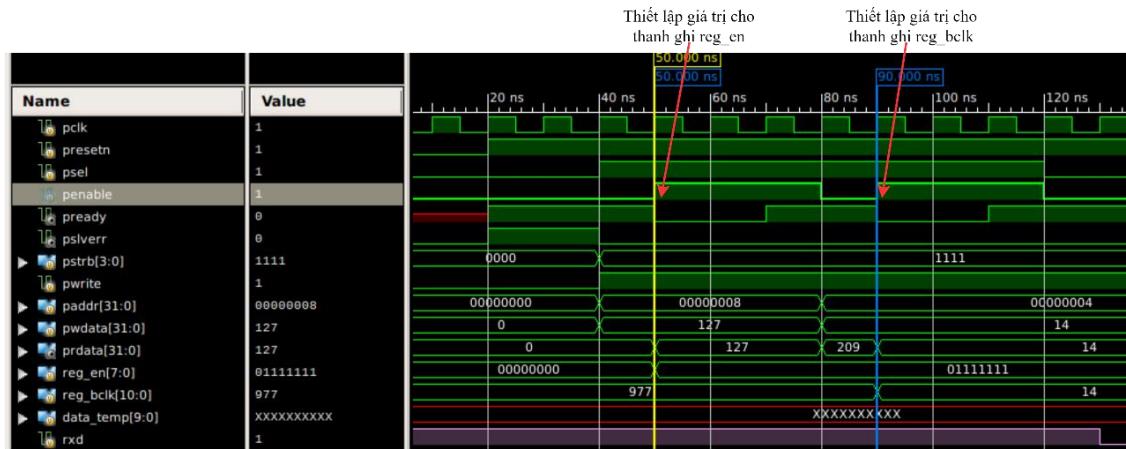


Hình 4.18. Kết quả đọc dữ liệu ở quá trình nhận có chế độ Parity chẵn
Tại thời điểm 24770 ns, PSEL được nâng lên mức cao, PWRITE được hạ xuống mức thấp, bắt đầu quá trình đọc. Tại 24780 ns, PENABLE = 1.

Tại thời điểm 24800, khi này PREADY đã lên mức cao, dữ liệu đọc được đưa ra, khi này, PRDATA có giá trị là 219(8'b11011011). Như vậy, dữ liệu đọc được đúng như dữ liệu nhận được.

- **Kết quả nhận dữ liệu có chế độ Parity lẻ**

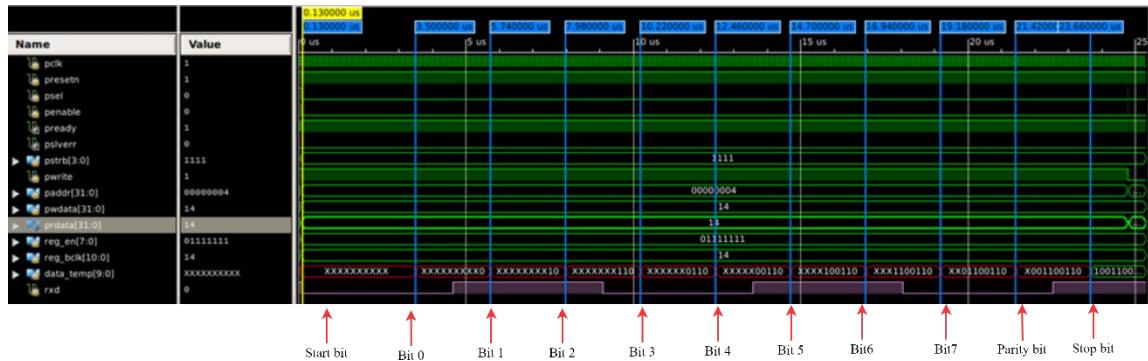
Quá trình thiết lập thông số như hình 4.19:



Hình 4.19. Kết quả thiết lập thông số nhận có chế độ Parity lẻ

Việc thiết lập giống như quá trình nhận không có chế độ Parity, ngoại trừ thông số được ghi vào thanh ghi REG_EN là 01111111 thay vì 00111111. Dữ liệu được gửi tới là 011001101.

Sau khi thiết lập, RXD được nhận dữ liệu như hình 4.20:



Hình 4.20. Quá trình nhận dữ liệu có chế độ Parity lẻ

Tại thời điểm 0.13 us, RXD được kéo xuống mức thấp, start bit bắt đầu được nhận. Quá trình nhận dữ liệu được tóm gọn theo bảng 4.6:

Bảng 4.6. Dữ liệu nhận vào thanh ghi có Parity lẻ

| Thời điểm | Giá trị | Vị trí bit trong thanh ghi DATA_TEMP |
|-----------|---------|--------------------------------------|
| 3.5 us | 0 | Bit 0 |
| 5.74 us | 1 | Bit 1 |
| 7.98 us | 1 | Bit 2 |
| 10.22 us | 0 | Bit 3 |

| | | |
|----------|---|-------|
| 12.46 us | 0 | Bit 4 |
| 14.7 us | 1 | Bit 5 |
| 16.94 us | 1 | Bit 6 |
| 19.18 us | 0 | Bit 7 |
| 21.42 us | 0 | Bit 8 |
| 23.66 us | 1 | Bit 9 |

Có thể thấy rằng, 8 bit dữ liệu nhận được là 01100110. Vì quá trình nhận có chế độ parity chẵn nên bit thứ 8 là parity bit có giá trị là 0. Bit thứ 9 tương đương với stop bit là 1.

Kết quả đọc dữ liệu như hình 4.21:



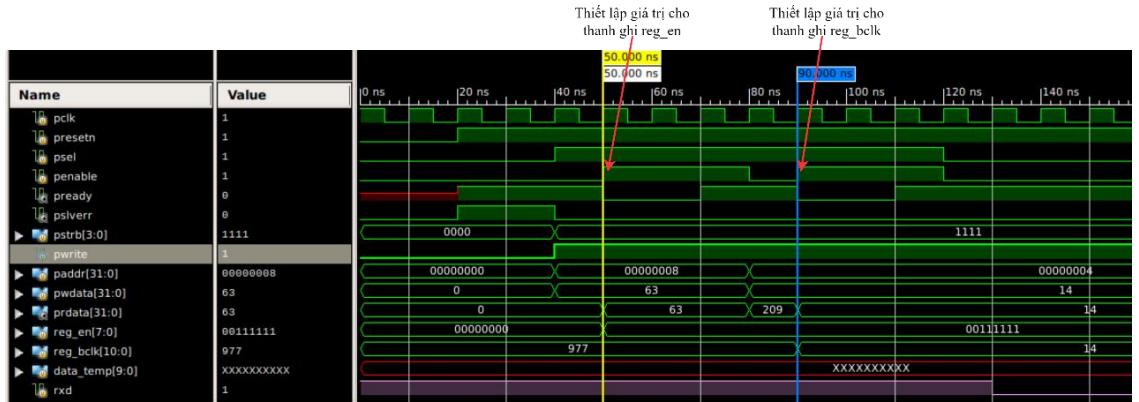
Hình 4.21 Kết quả đọc dữ liệu ở quá trình nhận có chế độ Parity lẻ

Tại thời điểm 24770 ns, PSEL được nâng lên mức cao, PWRITE được hạ xuống mức thấp, bắt đầu quá trình đọc. Tại 24780 ns, PENABLE = 1.

Tại thời điểm 24800, khi này PREADY đã lên mức cao, dữ liệu đọc được đưa ra, khi này, PRDATA có giá trị là 102(8'b01100110). Như vậy, dữ liệu đọc được đúng như dữ liệu nhận được.

- Kết quả nhận 2 dữ liệu liên tiếp**

Quá trình thiết lập thông số như hình 4.22:

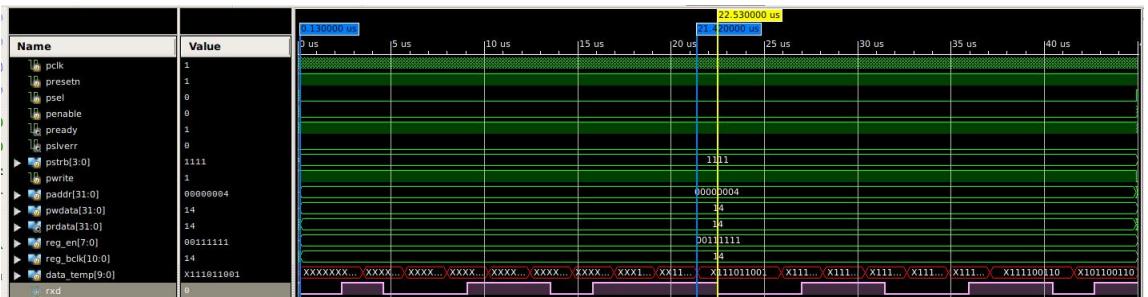


Hình 4.22. Kết quả thiết lập thông số nhận hai dữ liệu liên tiếp

Thanh ghi REG_EN được thiết lập giá trị 00111111 do quá trình nhận 2 dữ liệu này không có chế độ parity. Thanh ghi REG_BCLK được thiết lập giá trị là 14.

Tại thời điểm 130 ns, RXD bắt đầu nhận liệu liên tiếp 2 dữ liệu lần lượt là 11011001 và 01100110.

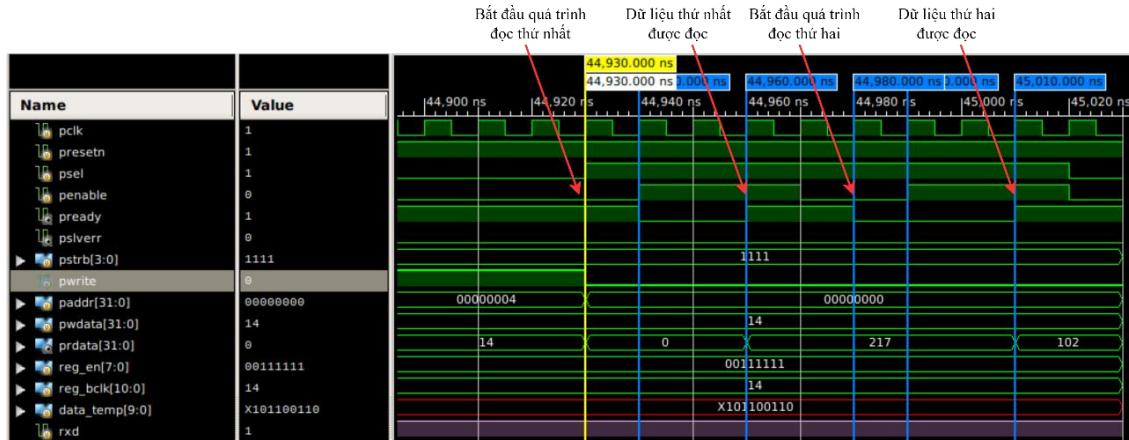
Kết quả nhận dữ liệu như sau:



Hình 4.23. Kết quả quá trình nhận 2 dữ liệu liên tiếp

Dựa vào hình 4.23 có thể thấy rằng, từ thời điểm 0.13 us, dữ liệu 11011001 được gửi tới. Tại thời điểm 21.42 us, các dữ liệu được đưa vào thanh ghi DATA_TEMP đầy đủ. Từ thời điểm 22.53, dữ liệu 01100110 được gửi tới, sau đó đã được đưa vào thanh ghi DATA_TEMP đầy đủ.

Sau khi đã nhận được 2 dữ liệu, phía APB Master tiến hành đọc liên tiếp hai dữ liệu. Kết quả đọc như hình 4.24:



Hình 4.24 Kết quả đọc hai dữ liệu liên tiếp

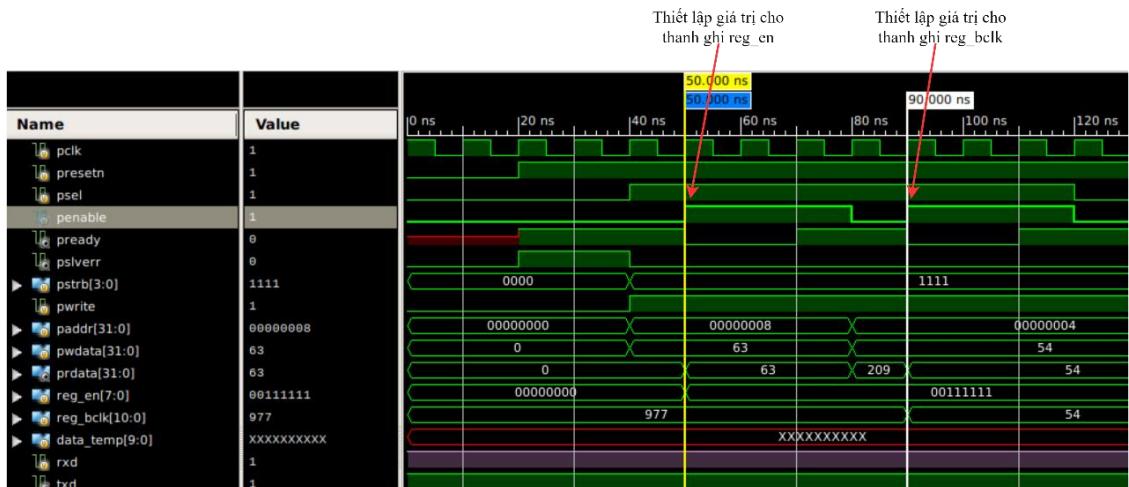
Tại thời điểm 44930 ns, PSEL được nâng lên mức cao để tiến hành đọc dữ liệu thứ nhất. Tại thời điểm 44960, khi này PENABLE và PREADY đều bằng 1, dữ liệu thứ nhất được đưa ra PRDATA có giá trị là 217(8'b110111001).

Tại thời điểm 44980 ns, PSEL được nâng lên mức cao để tiến hành đọc dữ liệu thứ nhất. Tại thời điểm 445010, khi này PENABLE và PREADY đều bằng 1, dữ liệu thứ nhất được đưa ra PRDATA có giá trị là 102(8'b01100110).

Như vậy, quá trình nhận dữ liệu và đưa ra PRDATA hoạt động chính xác.

- **Kết quả nhận dữ liệu với tốc độ truyền 115200**

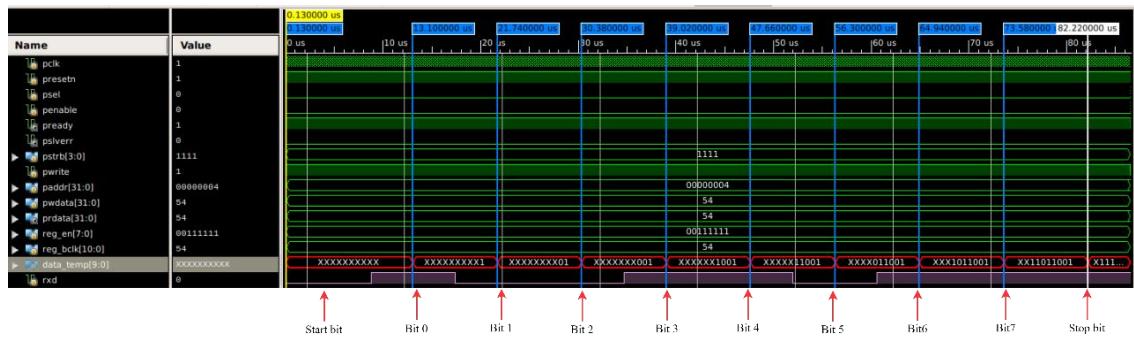
Quá trình thiết lập các thông số được thể hiện ở hình 4.25:



Hình 4.25. Kết quả thiết lập thông số nhận ở tốc độ truyền khác

Giá trị 00111111 được thiết lập cho thanh ghi REG_EN. Thanh ghi REG_BCLK được thiết lập giá trị là 54, tương đương với tốc độ truyền là 115200.

Sau khi thiết lập, RXD được nhận dữ liệu như hình 4.26:



Hình 4.26. Dữ liệu nhận từ RXD tại quá trình nhận với tốc độ truyền khác

Tại thời điểm 0.13 us, RXD được kéo xuống mức thấp, start bit bắt đầu được nhận. Quá trình nhận dữ liệu được tóm gọn theo bảng 4.7:

Bảng 4.7. Dữ liệu nhận vào thanh ghi với tốc độ truyền 115200

| Thời điểm | Giá trị | Vị trí bit trong thanh ghi DATA_TEMP |
|-----------|---------|--------------------------------------|
| 13.1 us | 1 | Bit 0 |
| 21.74 us | 0 | Bit 1 |
| 30.38 us | 0 | Bit 2 |
| 39.02 us | 1 | Bit 3 |
| 47.66 us | 1 | Bit 4 |
| 56.3 us | 0 | Bit 5 |
| 64.94 us | 1 | Bit 6 |
| 73.58 us | 1 | Bit 7 |
| 82.22 us | 1 | Bit 8 |

Từ đây, 8 bit dữ liệu nhận được là 11011001. Vì quá trình nhận không có chế độ parity nên bit thứ 8 tương đương với stop bit là 1.

Kết quả đọc dữ liệu như hình 4.27:

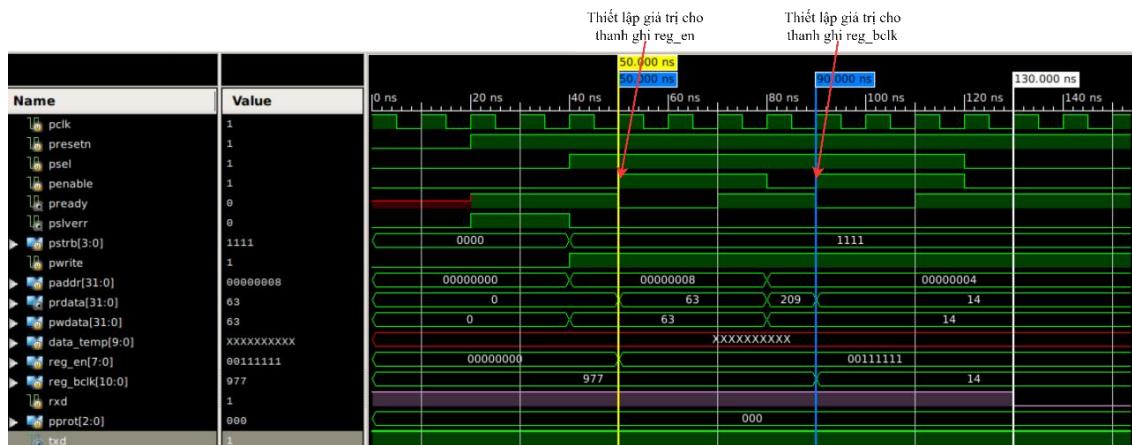


Hình 4.27. Kết quả đọc dữ liệu ở quá trình nhận với tốc độ truyền khác
Tại thời điểm 86530 ns, PSEL được nâng lên mức cao, PWRITE được hạ xuống mức thấp, bắt đầu quá trình đọc. Tại 86540 ns, PENABLE = 1.

Tại thời điểm 86560, khi này PREADY đã lên mức cao, dữ liệu đọc được đưa ra, khi này, PRDATA có giá trị là 217(8'b11011001). Như vậy, dữ liệu đọc được đúng như dữ liệu nhận được.

- **Kết quả truyền start bit giả**

Quá trình thiết lập thông số như sau:

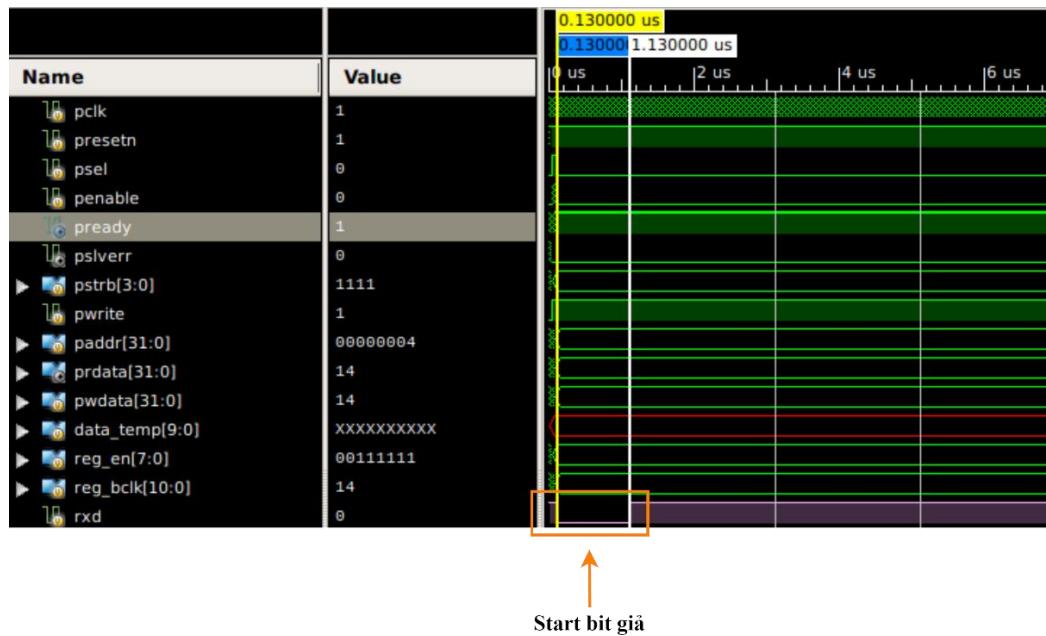


Hình 4.28. Kết quả thiết lập thông số truyền start bit giả

Tại trường hợp này, RXD sẽ được kéo xuống mức thấp trong một khoảng thời gian bé hơn một nửa chu kỳ bit của UART nhằm tạo một start bit giả để kiểm tra khả năng tránh start bit giả.

Thanh ghi REG_EN được ghi giá trị là 00111111, thanh ghi BCLK được ghi giá trị 14.

Kết quả nhận start bit giả như hình 4.29:



Hình 4.29. Kết quả nhận start bit giả

Tại thời điểm 0.13us, TXD được kéo xuống, start bit được gửi tới. Tuy nhiên, sau đó 1 us, TXD được kéo lên mức cao. Chu kỳ của start bit trong tốc độ truyền này là 2.24 us. Thời gian kéo dài của tín hiệu mức thấp kia là 1 us, không bằng một nửa chu kỳ của start bit. Chính vì thế, từ thời điểm 1.13 us trở đi, thanh ghi DATA_TEMP ở phía Receiver không hề được nhận dữ liệu nào.

Như vậy, bộ chuyển đổi có khả năng tránh được các start bit giả được truyền tới nếu như khoảng thời gian của start bit giả nhỏ hơn một nửa chu kỳ bit.

4.2.4 Kết quả kiểm thử truyền nhận dữ liệu đồng thời

- Trường hợp kiểm thử chức năng của quá trình nhận cùng lúc

Trường hợp này được tạo ra nhằm mục đích xác định khả năng truyền nhận cùng lúc của bộ chuyển đổi APB sang UART.

Trường hợp kiểm thử được liệt kê và mô tả cụ thể như sau: Đầu tiên, hệ thống được khởi động lại bằng tín hiệu PRESETN mức thấp. Sau đó, nâng PRESETN lên mức cao. Tiếp theo, giá trị 00111111 được thiết lập cho thanh ghi REG_EN tại địa chỉ 0x00000008 để thiết lập chế độ truyền nhận. Thiết lập giá trị 14 cho thanh ghi REG_BCLK ở địa chỉ 0x00000004. Thiết lập giá trị 11011001 cho thanh ghi REG_DATA ở địa chỉ 0x00000000. Đồng thời, phía RXD có dữ liệu được gửi tới.

Sau khoảng thời gian đó, quá trình đọc có trạng chờ được bắt đầu ở phía APB với mục đích lấy giá trị ở thanh ghi REG_DATA..

- **Kết quả truyền nhận dữ liệu cùng lúc**

Quá trình thiết lập các thông số được thể hiện ở hình 4.30:



Hình 4.30. Kết quả thiết lập thông số truyền nhận cùng lúc

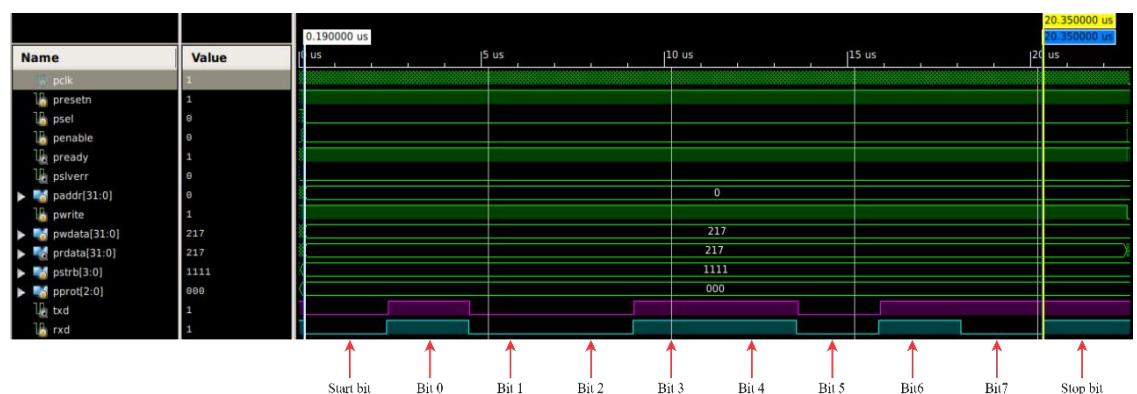
Từ 0ns đến 20ns, tín hiệu PRESETN ở mức thấp để tiến hành khởi động lại hệ thống.

Từ 40ns đến 80ns là một quá trình ghi có trạng thái chờ, PADDR được thiết lập giá trị 0x00000008, PWDATA được thiết lập giá trị 63(8'b00111111).

Từ 80ns đến 120ns là một quá trình ghi có trạng thái chờ, PADDR được thiết lập giá trị 0x00000004, PWDATA được thiết lập giá trị 14.

Từ 130ns đến 170ns là một quá trình ghi PADDR được thiết lập giá trị, PWDATA được thiết lập giá trị 217 (8'b1011001). Thanh ghi REG_DATA mang dữ liệu là 10111001

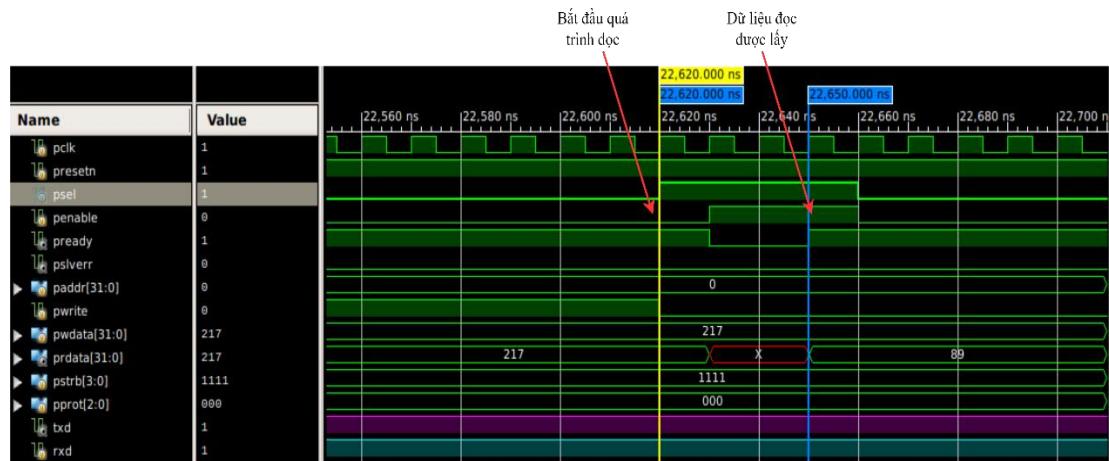
Kết quả quá trình truyền nhận cùng lúc được thể hiện như hình 4.31:



Hình 4.31. Kết quả quá trình truyền nhận cùng lúc

Tại 0.19 us, TxD được kéo xuống mức thấp, quá trình truyền được bắt đầu. Cùng lúc đấy, RXD bắt đầu nhận được tín hiệu mức thấp, bắt đầu quá trình nhận. Theo hình 4.31, dữ liệu nhận được tại chân RXD có giá trị là 01011001.

Kết quả quá trình đọc dữ liệu được thể hiện như hình 4.32:



Hình 4.32. Kết quả đọc trong quá trình truyền nhận cùng lúc

Hình 4.32 thể hiện kết quả của việc đọc dữ liệu ở thanh ghi REG_DATA, chi tiết của quá trình này như sau:

Tại thời điểm 22620 ns, PSEL được nâng lên mức cao, quá trình đọc được bắt đầu. PADDR được thiết lập giá trị 0x00000000.

Tại thời điểm 22650 ns, PREADY được nâng lên mức cao, dữ liệu ở thanh ghi REG_DATA được đọc. Khi này, PRDATA có giá trị 89 (8'b01011001), giá trị này giống như giá trị mà nhận được ở chân RXD.

Như vậy, quá trình truyền và nhận ở hai chân TxD và RxD của bộ chuyển đổi đã diễn ra đúng như mong đợi.

4.2.5 Kết quả kiểm thử các tín hiệu ngắn, tín hiệu PSVLERR

- Các trường hợp kiểm thử các tín hiệu ngắn, tín hiệu PSLVERR**

Các trường hợp (testcase) kiểm thử được liệt kê và mô tả cụ thể trong bảng:

Bảng 4.8. Các testcase cho kích hoạt tín hiệu ngắn, tín hiệu PSLVERR

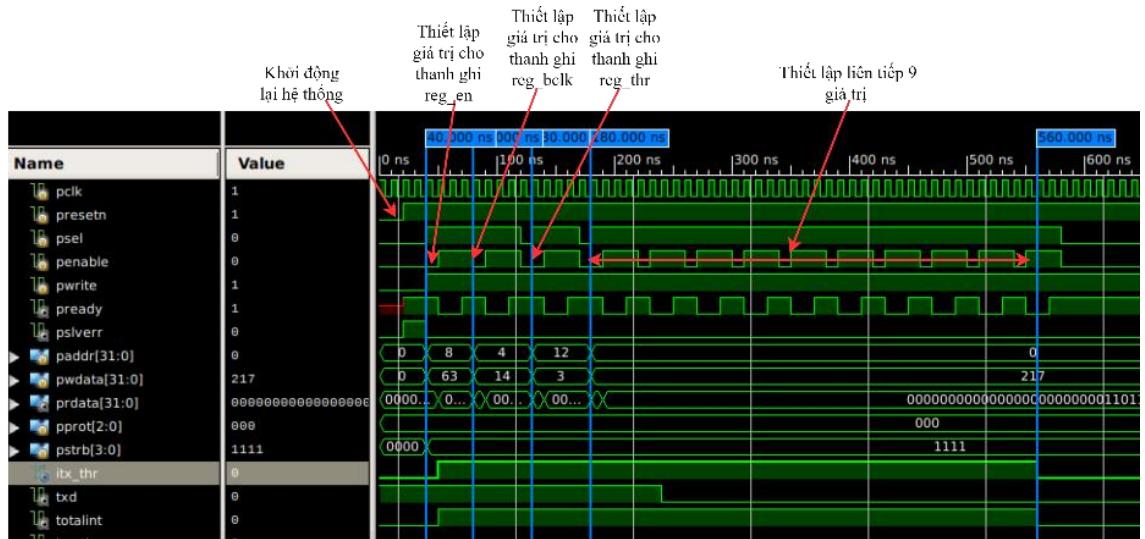
| Tên testcase | Mô tả quá trình |
|--|--|
| Kích hoạt tín hiệu ngắn ngưỡng Transmitter | Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp. |

| | |
|---|--|
| | <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 11111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> <p>Thiết lập giá trị 14 cho thanh ghi REG_BCLK ở địa chỉ 0x00000004.</p> <p>Kế tiếp, thiết lập giá trị 0011 cho thanh ghi REG_THR ở địa chỉ 0x0000000C.</p> <p>Cuối cùng thiết lập giá trị 11011001 cho thanh ghi reg_data ở địa chỉ 0x00, lập lại thêm quá trình này 10 lần.</p> |
| Kích hoạt tín hiệu ngắt nguồn Receiver | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 00111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> <p>Sau đó, thiết lập giá trị 14 cho thanh ghi REG_BCLK ở địa chỉ 0x00000004.</p> <p>Kết tiếp, thiết lập giá trị 1100 cho thanh ghi REG_THR ở địa chỉ 0x0000000C.</p> <p>RXD gửi liên tiếp 2 dữ liệu 8 bit.</p> |
| Kích hoạt tín hiệu ngắt tràn | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 00111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> <p>Sau đó, thiết lập giá trị 14 cho thanh ghi REG_BCLK ở địa chỉ 0x00000004.</p> <p>RXD gửi liên tiếp 17 dữ liệu 8 bit.</p> |
| Kích hoạt tín hiệu ngắt lỗi parity | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> |

| | |
|-----------------------------------|---|
| | <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 11111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> <p>Sau đó, thiết lập giá trị 14 (Tốc độ truyền = 480600) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004.</p> <p>Phía RXD nhận được dữ liệu là 11011101 với parity bit là 0.</p> |
| Kích hoạt tín hiệu ngắt lỗi khung | <p>Khởi động lại hệ thống bằng tín hiệu PRESETN mức thấp.</p> <p>Nâng PRESETN lên mức cao. Thiết lập giá trị 11111111 cho thanh ghi REG_EN tại địa chỉ 0x00000008.</p> <p>Sau đó, thiết lập giá trị 14 (Tốc độ truyền = 480600) cho thanh ghi REG_BCLK ở địa chỉ 0x00000004.</p> <p>Phía RXD nhận được dữ liệu là 11011101 với parity bit và stop bit là 0.</p> |
| Kích hoạt tín hiệu PSLVERR | <p>Gồm 2 tác vụ:</p> <p>Đầu tiên là thực hiện quá trình ghi với địa chỉ là 0x00000019. Tiếp theo thực hiện quá trình ghi với địa chỉ là 0x00000008.</p> <p>Thứ hai là thực hiện quá trình ghi với địa chỉ là 0x00000008 với tín hiệu PSTRB là 1110. Tiếp theo thực hiện ghi cùng với địa chỉ như vậy như tín hiệu PSTRB khi này là 1111.</p> |

- **Kết quả kích hoạt tín hiệu ngắt nguồn Transmitter**

Kết quả thiết lập các thông số ban đầu được thể hiện ở hình 4.33.



Hình 4.33. Kết quả kích hoạt tín hiệu ITX_THR

Từ 0ns đến 20ns, tín hiệu PRESETN ở mức thấp để tiến hành khởi động lại hệ thống.

Từ 40ns đến 80ns, một quá trình ghi có trạng thái chờ để thiết lập thông số cho thanh ghi REG_EN được bắt đầu, địa chỉ PADDR được thiết lập giá trị 0x00000008, giá trị được thiết lập cho PWDATA là 63 (8'b00111111).

Từ 80ns đến 120ns là một quá trình ghi có trạng thái chờ để thiết lập giá trị cho thanh ghi REG_BCLK, địa chỉ PADDR được thiết lập giá trị 0x00000004, PWDATA được thiết lập giá trị 14.

Từ 130ns đến 170ns là một quá trình ghi để thiết lập giá trị cho thanh ghi REG_THR, PADDR được thiết lập giá trị 0x0000000C, PWDATA được thiết lập giá trị 32'b11.

Từ 180ns trở đi, PSEL và PWRITE liên tục ở mức cao, các dữ liệu được đưa vào liên tục. Quá trình này kéo dài tới 560 ns, khi mà tín hiệu I_TXTHR được kéo xuống mức thấp. Việc này đồng nghĩa với số ô trống có trong FIFO của Transmitter thấp hơn mức ngưỡng là 8 ô. Nguyên do là từ 180 ns đến 560 ns tương đương với khoảng thời gian là 380 ns, mỗi quá trình đọc mất 40 ns, như vậy, có tới 10 dữ liệu đã được ghi vào bộ FIFO, một dữ liệu đang được truyền đi nên còn 9 dữ liệu đang lưu trữ.

- **Kích hoạt tín hiệu ngắt ngưỡng Receiver**

Kết quả thiết lập các thông số được thể hiện như hình 4.34.



Hình 4.34. Kết quả thiết lập thông số kích hoạt tín hiệu IRX_THR
Để tiến hành kích hoạt tín hiệu ngắt ngưỡng ở bộ Receiver, một giá trị
4'b1100 sẽ được thiết lập vào thanh ghi REG_THR để thiết lập mức ngưỡng ở bộ
Receiver. Chi tiết của quá trình thiết lập thông số như sau:

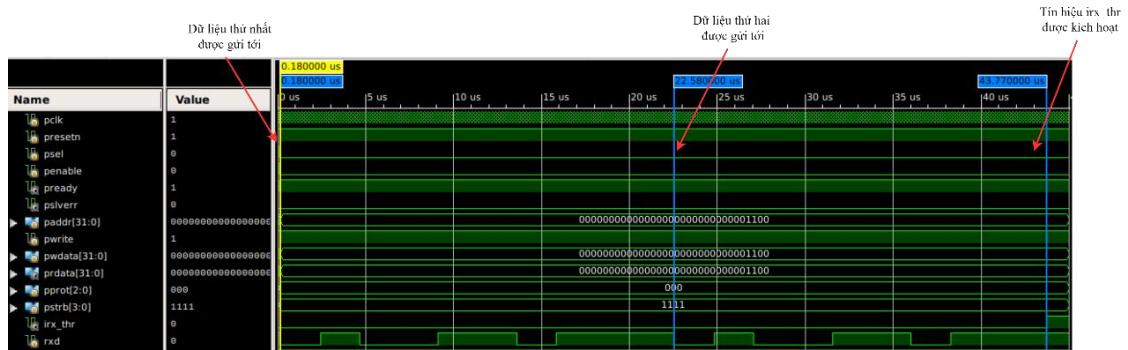
Từ 0ns đến 20ns, tín hiệu PRESETN ở mức thấp để tiến hành khởi động lại hệ
thống.

Từ 40ns đến 80ns là một quá trình ghi có trạng thái chờ, giá trị 63
(8'b00111111) được thiết lập cho thanh ghi REG_EN, cho phép tín hiệu ngắt
ngưỡng được kích hoạt.

Từ 80ns đến 120ns là một quá trình ghi, giá trị 14 được thiết lập cho thanh ghi
REG_BCLK.

Từ 130ns đến 170ns là một quá trình ghi, PADDR được thiết lập giá trị
0x0000000C, PWDATA được thiết lập giá trị 12 (4'b1100). Khi này, mức ngưỡng
được thiết lập cho bộ Receiver sẽ là 11, tương ứng với tín hiệu ngắt sẽ kích hoạt
khi có từ 2 ô trống lên trong bộ FIFO chứa dữ liệu.

Kết quả của tín hiệu ngắt sau khi nhận hai dữ liệu được thể hiện như hình sau



Hình 4.35. Kết quả kích hoạt tín hiệu IRX_THR

Lần lượt hai dữ liệu sẽ được gửi tới ở chân RXD.

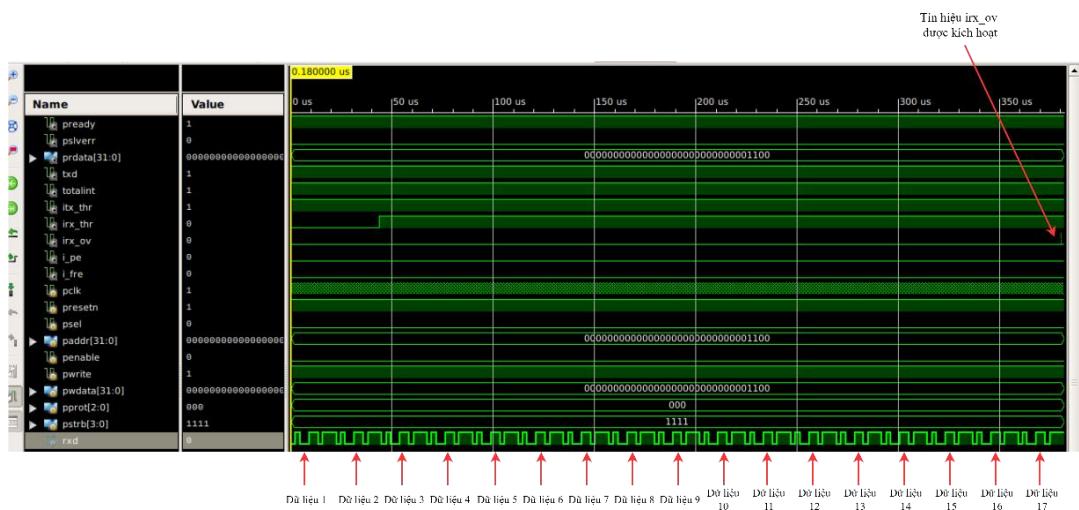
Tại thời điểm 0.18 us, dữ liệu thứ nhất được gửi tới thông qua RXD.

Tại thời điểm 22.58 us, dữ liệu thứ hai được gửi tới thông qua RXD.

Tại thời điểm 43.77 us, bộ FIFO tại Receiver đã có 2 ô chứa dữ liệu. Do đã đáp ứng mức ngưỡng, tín hiệu IRX_THR đã được kéo lên mức cao. Như vậy, tín hiệu ngắt ngưỡng ở bộ Receiver đã hoạt động đúng theo tính năng quy định.

• Kích hoạt tín hiệu ngắt tràn

Kết quả thiết lập thông số và kích hoạt tín hiệu ngắt tràn được thể hiện như hình 4.36.



Hình 4.36. Kết quả kích hoạt tín hiệu IRX_OV

Để kích hoạt tín hiệu ngắt tràn, bộ chuyển đổi sẽ được thiết lập các thông số cho phép và tốc độ truyền, sau đó, lần lượt 17 dữ liệu với mỗi dữ liệu có độ rộng 8 bit và được đóng khung theo giao thức UART sẽ được gửi tới ở chân RXD. Chi tiết của quá trình này như sau:

Từ 0ns đến 20ns, tín hiệu PRESETN ở mức thấp để tiến hành khởi động lại hệ thống.

Từ 40ns đến 80ns là một quá trình ghi có trạng thái chờ, giá trị 63 (8'b00111111) được thiết lập cho thanh ghi REG_EN, cho phép tín hiệu ngắt nguồn được kích hoạt.

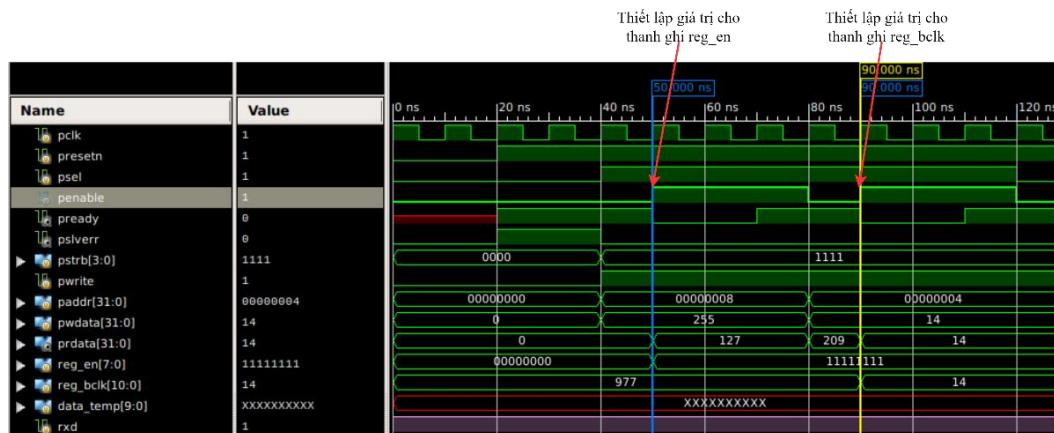
Từ 80ns đến 120ns là một quá trình ghi, giá trị 14 được thiết lập cho thanh ghi REG_BCLK.

Từ 180 ns trở đi, lần lượt các dữ liệu được gửi vào Receiver. Tại cuối thời điểm của dữ liệu thứ 17 được truyền tới, bộ FIFO của Receiver khi này đã chứa đầy 16 ô dữ liệu, phía bus APB không thực hiện việc đọc dữ liệu trong quá trình này, chính vì thế, tín hiệu IRX_OV được nâng lên mức cao trong một khoảng thời gian là một chu kỳ xung PCLK, báo hiệu bộ FIFO đã tràn và dữ liệu thứ 17 không được ghi vào.

Với kết quả như trên, bộ chuyển đổi đã xử lý tín hiệu ngắt tràn và thông báo đúng theo nhu cầu năng đã quy định.

- Kết quả kích hoạt tín hiệu ngắt lỗi parity**

Quá trình thiết lập thông số được thể hiện ở hình 4.37.

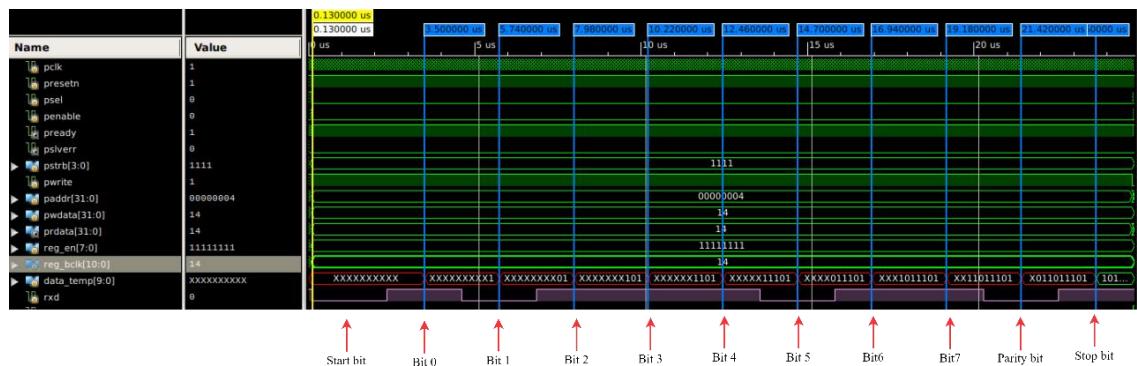


Hình 4.37. Kết quả thiết lập thông số cho kích hoạt tín hiệu I_PE

Để kích hoạt tín hiệu ngắt lỗi parity, bộ chuyển đổi sẽ được thiết lập ở chế độ truyền nhận parity chẵn. Sau đó, một dữ liệu có giá trị là 11011101 kèm với parity bit là 0 được đưa tới ở chân RXD. Chi tiết của quá trình thiết lập thông số như sau:

Từ 0 đến 20 ns, tín hiệu PRESETN ở mức thấp để tiến hành khởi động lại. Tại thời điểm 40 ns, PSEL được nâng lên mức cao, bắt đầu quá trình truyền. Tại 50 ns, giá trị 11111111 được đưa vào thanh ghi REG_EN. Tại 90 ns, giá trị 14 được đưa vào thanh ghi REG_BCLK.

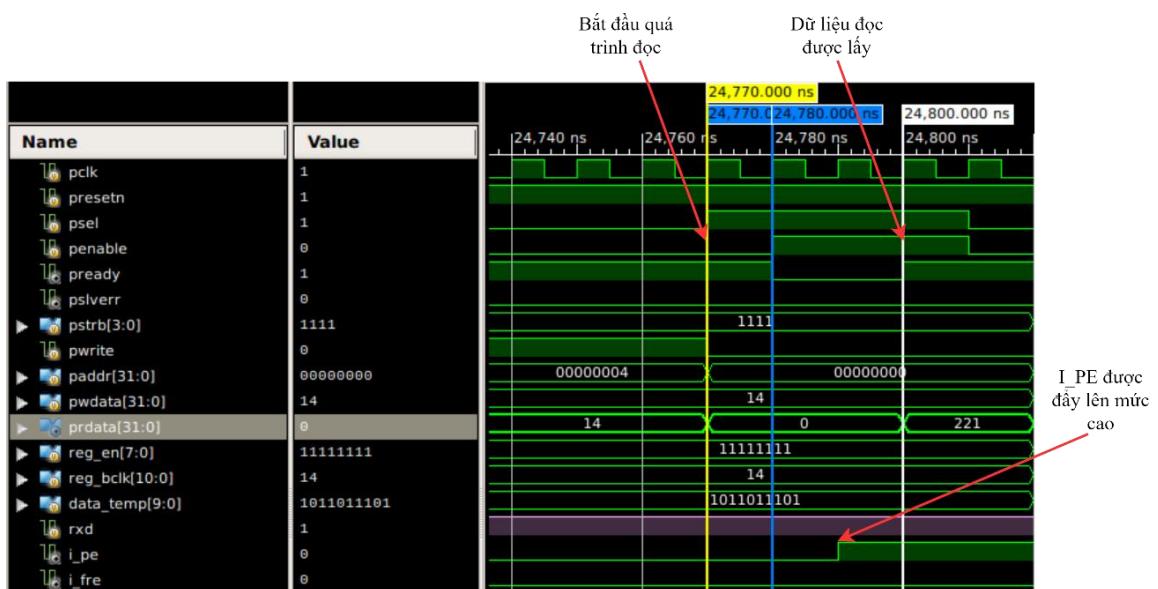
Kế tiếp đó, RXD được nhận dữ liệu là 11011101 với parity bit là 0. Kết quả của quá trình truyền như hình 4.38:



Hình 4.38. Kết quả nhận được ở quá trình kích hoạt tín hiệu I_PE

Dựa theo hình 4.38, dữ liệu nhận được ở đây là 11011101, giá trị parity ở đây là 0. Do số bit 1 trong dữ liệu nhận được là số chẵn, chế độ đang hoạt động là chế độ parity chẵn, chính vì vậy, dữ liệu nhận được đã bị lỗi.

Kết quả quá trình đọc diễn ra như hình 4.39:



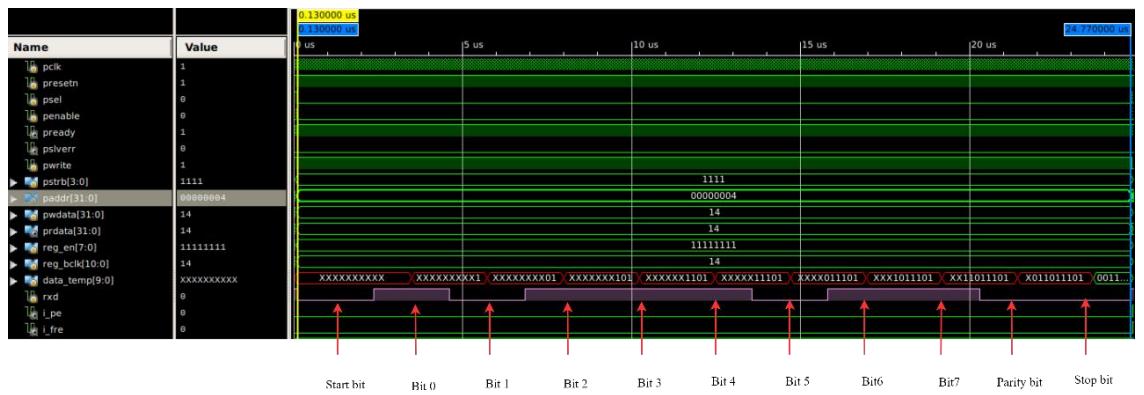
Hình 4.39. Kết quả đọc dữ liệu có lỗi parity

Tại thời điểm 24770 ns, PSEL được kéo lên mức cao, PWRITE được kéo xuống mức thấp, bắt đầu quá trình đọc. Tại thời điểm 24780 ns, PWRITE được kéo lên mức cao. Tại thời điểm 24800, PREADY được đưa lên mức cao, khi này dữ liệu 221(8'b11011101) được đưa ra phía PRDATA, đồng thời, tín hiệu I_PE cũng ở mức cao, báo hiệu tín hiệu này đang bị lỗi parity.

Như vậy, tín hiệu ngắt lỗi parity đã hoạt động đúng theo như chức năng quy định.

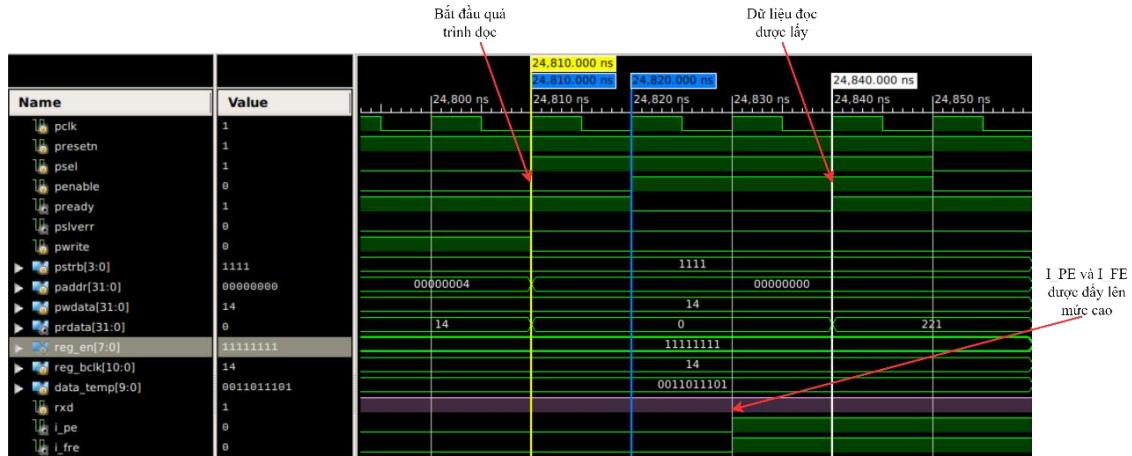
- Kết quả kích hoạt tín hiệu ngắt lỗi khung**

Quá trình thiết lập tương tự như ở kích hoạt ngắt lỗi parity. Tuy nhiên, để có thể kích hoạt tín hiệu ngắt lỗi khung, stop bit đã thay đổi thành giá trị 0. Kết quả truyền dữ liệu có lỗi khung được thể hiện như hình 4.40.



Hình 4.4.40. Kết quả truyền dữ liệu có lỗi khung và parity

Theo hình 4.40, sau khi lần lượt các start bit, 8 bit dữ liệu và parity bit được truyền tới, stop bit được đưa tới mang giá trị là 0, khung truyền đã bị lỗi dữ liệu. Khi này, quá trình đọc diễn ra như hình 4.41:



Hình 4.41. Kết quả khi đọc dữ liệu bị lỗi parity và khung truyền

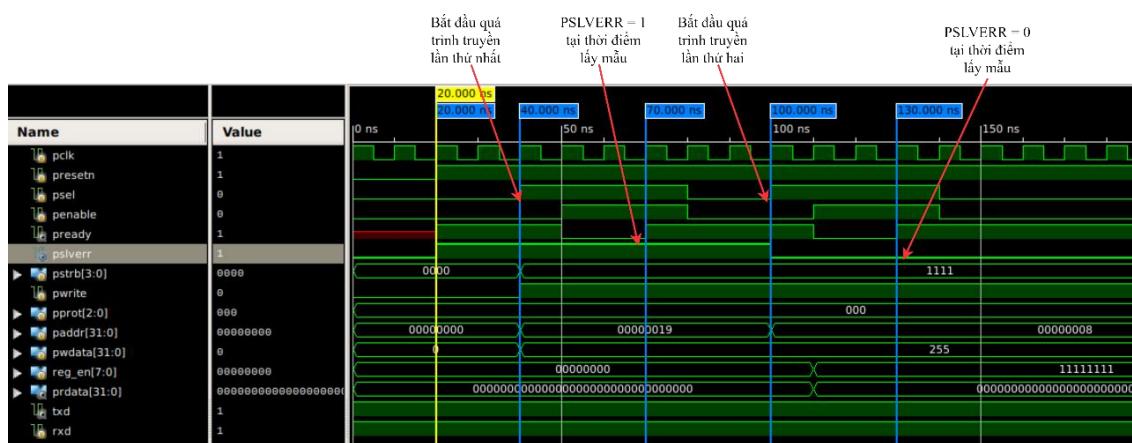
Tại thời điểm cả PSEL, PENABLE và PREADY đều ở mức cao, dữ liệu được đưa ra phía PRDATA, kèm với đó là tín hiệu I_PE và I_FRE đều được nâng lên mức cao, cho thấy dữ liệu nhận được đã bị lỗi parity và lỗi khung.

Như vậy, khả năng báo lỗi truyền và lỗi parity đều hoạt động chính xác.

- Kích hoạt tín hiệu PSLVERR

Tín hiệu PSLVERR được lấy mẫu tại thời điểm cả PSEL, PENABLE và PREADY đều ở mức cao. Điều kiện để PSLVERR trong bộ chuyển đổi này lên mức cao gồm 2 cái là: địa chỉ không tương thích và PSTRBB không phù hợp.

Ở điều kiện địa chỉ không tương thích, kết quả như hình 4.42:



Hình 4.42. Kết quả của PSLVERR khi địa chỉ không phù hợp

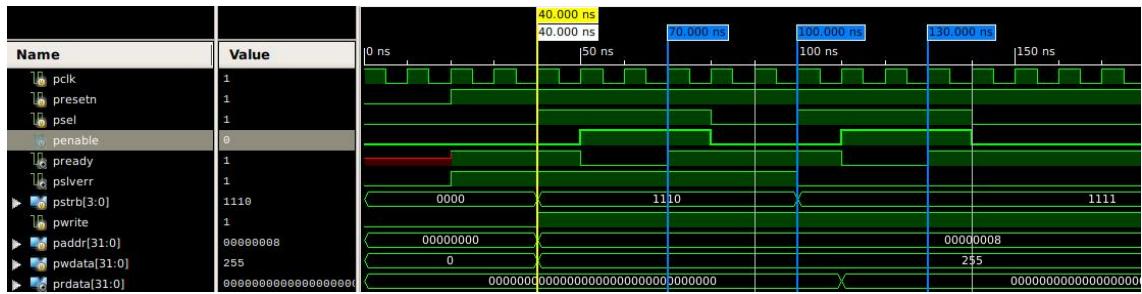
Dựa vào hình 4.42 có thể nhận xét như sau:

Tại thời điểm 40 ns, quá trình truyền thứ nhất được bắt đầu. Địa chỉ truyền vào khi này là 0x00000013, đây là địa chỉ không tồn tại trong bộ chuyển đổi. Chính vì

vậy, tại thời điểm 70 ns, khi mà cả PSEL, PENABLE và PREADY đều ở mức thấp, PSLVERR cũng ở mức cao.

Trong khi đó tại quá trình truyền thứ hai, địa chỉ được truyền vào là 0x00000008, đây là địa chỉ phù hợp nên PSLVERR ở mức thấp tại thời điểm PSEL, PENABLE và PREADY ở mức cao.

Còn ở điều kiện PSRTB không phù hợp, kết quả được thể hiện như hình 4.43:



Hình 4.43. Kết quả của PSLVERR khi PSTRB không phù hợp

Dựa vào hình 4.43 có thể nhận xét như sau:

Tại thời điểm 40 ns, quá trình truyền thứ nhất được bắt đầu. PSTRB được đưa vào có giá trị là 1110, vì vậy, ngay cả khi địa chỉ hợp lệ, giá trị cũng không được cập nhật cho thanh ghi. Chính vì thế, tại thời điểm 70 ns, PSLVERR ở mức cao khi PSEL, PENABLE và PREADY đều ở mức cao.

Ở quá trình truyền thứ hai, PSTRB bằng 1111, cho phép dữ liệu được cập nhật cho thanh ghi, chính vì thế mà PSLVERR ở mức thấp tại thời điểm 130 ns.

Như vậy, tín hiệu PSLVERR đã hoạt động đúng theo tính năng định sẵn.

4.3 ĐÁNH GIÁ KẾT QUẢ

Dựa theo các kết quả kiểm thử ở phần 4.2, các trường hợp mô phỏng được tổng hợp và đánh giá theo bảng 4.9.

Bảng 4.9 Tổng hợp trường hợp mô phỏng được tổng hợp và đánh giá

| Số thứ tự | Tên trường hợp | Đánh giá |
|-----------|--|---------------------|
| 1 | Truyền dữ liệu từ APB sang UART không có chế độ parity | Hoạt động chính xác |
| 2 | Truyền dữ liệu từ APB sang UART có chế độ parity chẵn | Hoạt động chính xác |

| | | |
|----|---|--|
| 3 | Truyền dữ liệu từ APB sang UART có chế độ parity lẻ | Hoạt động chính xác |
| 4 | Truyền liên tiếp 2 dữ liệu từ APB sang UART không có chế độ parity | Hoạt động chính xác. Tuy nhiên, vẫn có độ trễ giữa hai dữ liệu, dẫn đến tốc độ truyền thực tế chậm hơn so với lý thuyết. |
| 5 | Nhận dữ liệu từ UART sang APB không có chế độ parity | Hoạt động chính xác |
| 6 | Nhận dữ liệu từ UART sang APB có chế độ parity chẵn | Hoạt động chính xác |
| 7 | Nhận dữ liệu từ UART sang APB chế độ parity lẻ | Hoạt động chính xác |
| 8 | Nhận liên tiếp 2 dữ liệu từ UART sang APB không có chế độ parity | Hoạt động chính xác |
| 9 | Nhận dữ liệu từ UART sang APB không có chế độ parity với tốc độ truyền khác | Hoạt động chính xác |
| 10 | Truyền và nhận dữ liệu đồng thời ở phía UART | Hoạt động chính xác |
| 11 | Nhận start bit giả từ UART | Hoạt động chính xác |
| 12 | Kích hoạt tín hiệu ngắt nguồng Transmitter | Hoạt động chính xác |
| 13 | Kích hoạt tín hiệu ngắt nguồng Receiver | Hoạt động chính xác |
| 14 | Kích hoạt tín hiệu ngắt tràn | Hoạt động chính xác |
| 15 | Kích hoạt tín hiệu ngắt lỗi parity | Hoạt động chính xác |
| 16 | Kích hoạt tín hiệu ngắt lỗi khung | Hoạt động chính xác |
| 17 | Kích hoạt tín hiệu PSLVERR | Hoạt động chính xác |

Từ các kết quả đánh giá được tổng hợp ở bảng 4.9, bộ chuyển đổi giao thức APB sang UART đã thực hiện đúng các chức năng sau:

- Truyền nhận dữ liệu từ phía bus APB sang UART và ngược lại. Có thể truyền nhận dữ liệu liên tiếp, đồng thời và trên các tốc độ truyền khác nhau.
- Bộ chuyển đổi có thể cấu hình được các tín hiệu cho phép, các mức ngưỡng, tốc độ truyền.
- Thiết lập các tín hiệu ngắn đúng theo chức năng đã quy định.

Như vậy, bộ chuyển đổi đã hoạt động đúng như những chức năng đã đề ra.

CHƯƠNG 5 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 KẾT LUẬN

Sau khi thực hiện đề tài “Thiết kế bộ chuyển đổi giao thức APB sang UART”, tôi đã đưa ra những kết luận sau:

- Về ưu điểm:
 - Đề tài đã thực hiện được đầy đủ các tính năng đề ra.
 - Bộ chuyển đổi được thiết kế trên phiên bản APB 4 nên có thể dùng trong hệ thống bus AMBA 4.
 - Đề tài đã được kiểm chứng đầy đủ các tính năng dựa theo các kết quả có được.
 - Bộ chuyển đổi có thể hoạt động trên nhiều tần số khác nhau nhờ việc tính toán giá trị tốc độ truyền không phụ thuộc vào tần số hiện tại.
 - Người lập trình có thể thay đổi các cấu hình thông số như cho phép thông báo tín hiệu ngắn, các mức ngưỡng ở bộ nhận và truyền, tốc độ truyền, chế độ parity.
- Về nhược điểm:
 - Bộ chuyển đổi (IP UART) chưa được đánh giá trong một môi trường mô phỏng công nghiệp UVM.
 - Các tính năng gồm thay đổi độ rộng dữ liệu truyền nhận, thay đổi bội số ở khối tạo xung chưa được đưa vào.
 - Tín hiệu PSTRB và PPROT ở bus APB 4 chưa được sử dụng hiệu quả.
 - Thời gian cho quá trình đọc ghi và chuyển dữ liệu từ APB sang UART còn dài, mất nhiều chu kỳ xung clock cho một quá trình truyền.
 - Bộ chuyển đổi chưa được kiểm thử hoàn chỉnh trên một thiết bị FPGA, chưa kiểm tra, đánh giá timing, thời gian trễ.

5.2 HƯỚNG PHÁT TRIỂN

Bộ chuyển đổi giao thức APB sang UART được áp dụng rất phổ biến hiện nay. Tôi xin đề xuất các hướng phát triển sau để bộ chuyển đổi hoàn thiện hơn:

- Thiết kế và phát triển môi trường mô phỏng, kiểm tra thông qua môi trường mô phỏng UVM.
- Kiểm thử các tính năng của IP UART trên FPGA. Đánh giá thời gian trễ, công suất đầy đủ.
- Áp dụng được tín hiệu PPROT vào trong quá trình hoạt động của bộ chuyển đổi.
- Thay đổi được độ rộng dữ liệu có thể truyền, nhận.
- Thay đổi bộ số ở khói tạo xung.

TÀI LIỆU THAM KHẢO

- [1] ARM Limited, "AMBA Specification", ARM IHI 0024C, 1999.
- [2] S, Kavyashree, "Design and Implementation of UART using Verilog", 12 December, 2015.
- [3] Bhargav Tarpara, Heli ShahP and Chinmay ModiP, "Design & Implementation of Advance Peripheral Bus Protocol", June 2015.
- [4] ARM Limited, AMBA APB Protocol, 2003 - 2010.
- [5] Texas Instruments, KeyStone Architecture Universal Asynchronous Receiver/Transmitter (UART) User Guide, Nov. 2010.
- [6] Institute of Electrical and Electronics Engineers, Inc, "IEEE Standard for Verilog Hardware Description Language", 7 April 2006.
- [7] Mary Grace Legaspi and Eric Peña, "UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter", December 2020.
- [8] Neeta Choubey and H.R.Singh, "IMPLEMENTATION OF UART WITH STATUS REGISTER AND APB", July-August 2015.
- [9] Anushka Dwivedi and Dr. Anand Jatti, "DESIGN AND IMPLEMENTATION OF AMBA APB PROTOCOL", July-2022.