BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

BÁO CÁO CUỐI KỲ

MÔN: THIẾT KẾ HỆ THỐNG VÀ VI MẠCH TÍCH HỢP

ĐỀ TÀI: HỆ THỐNG TRUYỀN THÔNG NỐI TIẾP BẤT ĐỒNG BỘ UART

GVHD: TS. Đỗ Duy Tân

SVTH:	MSSV
Lâm Đức Quan	21161351
Nguyễn Vĩnh Hưng	21161086
Phan Minh Quân	21161353
Phan Thị Lan Hương	21161322

Mã lớp: ICSD336764_22_2_07CLC

Lớp thứ 4 tiết 123

Tp. Hồ Chí Minh, tháng 5 năm 2023

THỨ TỰ	HỌ TÊN	MSSV	TỈ LỆ HOÀN THÀNH	KÝ TÊN
1	Lâm Đức Quan	21161351	100%	
2	Nguyễn Vĩnh Hưng	21161086	100%	
3	Phan Minh Quân	21161353	100%	
4	Phan Thị Lan Hương	21161322	100%	

Ghi chú:

- Tỷ lệ %=100%: Mức độ phần trăm của từng thành viên tham gia.
- Trưởng nhóm: Lâm Đức Quan

MỤC LỤC

CHƯƠNG 1: TỔNG QUAN

1.1 Đặt vấn đề

Ngày nay, khoa học và công nghệ đang phát triển mạnh mẽ, đạt được nhiều thành tựu trong mọi lĩnh vực, đặc biệt là lĩnh vực điện tử. Ứng dụng của FPGA (Field Programmable Gate Arrays) là thiết bị lập trình thông dụng, lập trình cho các vi mạch bán dẫn nhỏ, công suất thấp để tạo ra các hệ thống điều khiển tự động và giải quyết nhiều bài toán phức tạp.

Các giao thức truyền thông đóng một vai trò quan trọng trong việc tổ chức giao giữa các thiết bị. Được thiết kế khác nhau tùy thuộc vào yêu cầu hệ thống, các giao thức này có các quy tắc cụ thể được thống nhất giữa các thiết bị để truyền dữ liệu thành công, điển hình là UART. Các hệ thống nhúng, vi điều khiển và máy tính thường sử dụng UART như một dạng giao thức giao tiếp phần cứng giữa thiết bị với thiết bị. Giao tiếp UART hiện được sử dụng trong nhiều ứng dụng để giao tiếp với các module như Wifi, Bluetooth, Arduino và các vi điều khiển khác. Nó cũng là một tiêu chuẩn giao tiếp được sử dụng rộng rãi trong ngành công nghiệp. Để tìm hiểu thêm về giao tiếp giữa các thiết bị số, chúng em chọn và nghiên cứu về UART và thực hiện thiết kế kiểm thử 1 IC UART bằng ngôn ngữ mô tả phần cứng Verilog.

1.2. Mục tiêu

Hiểu thế nào là UART.

Nắm vững vai trò, chức năng của UART.

Hiểu rõ cấu trúc, hoạt động và chức năng từng khối của UART.

Thực hiện thiết kế các khối của UART sử dụng ngôn ngữ Verilog.

1.3. Nội dung nghiên cứu

Tìm hiểu vai trò, chức năng, và nguyên tắc hoạt động của từng khối UART.

Thiết kế kiểm thử 1 IC UART sử dụng ngôn ngữ mô tả phần cứng Verilog.

1.4. Bố cục

Chương 1: Tổng quan

Chương 2: Cơ sở lý thuyết

Chương 3: Thiết kế giao thức UART

Chương 4: Đánh giá qua testbench

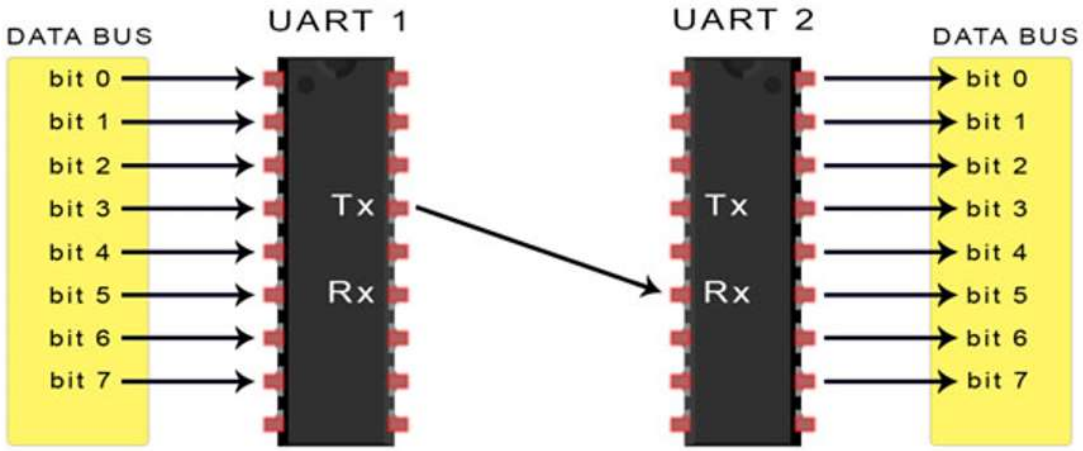
1.5.Giới hạn

Do đề tài này chúng em chỉ thực hiện nghiên cứu lý thuyết và mô phỏng trên phần mềm Xilinx nên vẫn còn bị giới hạn ở phần thực hành trên kit test.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. Giới thiệu về chuẩn giao tiếp UART

2.1.1. Khái niệm về UART



Theo định nghĩa, UART (Universal Asynchronous Receiver Transmitter) là một giao thức truyền thông phần cứng sử dụng giao tiếp nối tiếp không đồng bộ với tốc độ có thể định cấu hình. UART truyền dữ liệu không đồng bộ, có nghĩa là không có tín hiệu xung clock để đồng bộ hóa đầu ra của các bit từ UART truyền đến việc lấy mẫu các bit bởi UART nhận. Thay vì tín hiệu xung clock, UART truyền thêm các bit start và stop vào gói dữ liệu được chuyển. Các bit này xác định điểm bắt đầu và điểm kết thúc của gói dữ liệu để UART nhận biết khi nào bắt đầu đọc các bit.

Trong giao tiếp UART, hai UART giao tiếp trực tiếp với nhau. UART truyền chuyển đổi dữ liệu song song từ một thiết bị điều khiển như CPU thành dạng nối tiếp, truyền nó nối tiếp đến UART nhận, sau đó chuyển đổi dữ liệu nối tiếp trở lại thành dữ liệu song song cho thiết bị nhận.

Khi UART nhận phát hiện một bit start, nó bắt đầu đọc các bit đến ở một tần số cụ thể được gọi là tốc độ truyền (baud rate). Tốc độ truyền là thước đo tốc độ

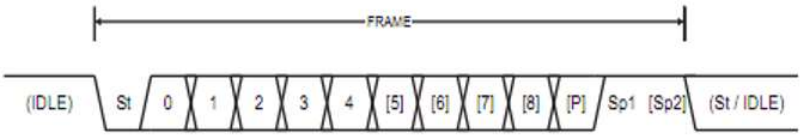
truyền dữ liệu, được biểu thị bằng bit trên giây (bps – bit per second). Cả hai UART đều phải hoạt động ở cùng một tốc độ truyền. Tốc độ truyền giữa UART truyền và nhận chỉ có thể chênh lệch khoảng 10% trước khi thời gian của các bit bị lệch quá xa.

Cả hai UART cũng phải được cấu hình để truyền và nhận cùng một cấu trúc gói dữ liệu.

Số lượng dây sử dụng	2
Tốc độ truyền (Tốc độ baud)	9600, 19200, 38400, 57600, 115200, 230400, 460800, 921600, 1000000, 1500000
Phương pháp truyền	Bất đồng bộ
Truyền nối tiếp hay song song?	Nối tiếp
Số lượng thiết bị chủ tối đa	1
Số lượng thiết bị tớ tối đa	1

2.1.2. Thông số cơ bản và khung truyền dữ liệu

Figure 19-4. Frame Formats



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

Sơ đồ khung truyền UART

Các thông số cơ bản cần nắm trong giao tiếp UART bao gồm:

Baud rate (tốc độ Baud): Khoảng thời gian để 1 bit được truyền đi. Phải được cài đặt giống nhau ở cả phần gửi và nhận. Các thông số tốc độ Baudrate thường hay sử dụng dể giao tiếp với máy tính là 1200, 2400, 4800, 9600, ..., 115200.

IDLE(chế độ trống): lúc này tín hiệu luôn ở mức 1 khi dữ liệu trống, không có frame nào được truyền đi.

Bit START: Để bắt đầu truyền dữ liệu, UART chuyển đường truyền từ mức "1" xuống mức "0" trong một chu kỳ clock. Khi đó, nó bắt đầu đọc các bit trong khung dữ liệu theo tần số của tốc độ truyền. Đây là một bit bắt buộc cần có trong giao thức UART.

Frame (khung truyền): Khung truyền quy định về mỗi lần truyền bao nhiêu

Data: Dữ liệu để truyền đi có độ dài từ 5 bit đến 8 bit nếu dùng bit chẵn lẻ. Nếu không dùng bit chẵn lẻ, data có thể dài đến 9 bit. Thông thường, Bit có trọng số nhỏ nhất LSB được truyền trước sau đó đến bit MSB.

Bit parity:

Bit parity sẽ là phương án giúp UART nhận cho biết liệu có bất kỳ dữ liệu nào đã thay đổi trong quá trình truyền hay không. Bit có thể bị thay đổi bởi bức xạ điện từ, tốc độ truyền không khớp hoặc truyền dữ liệu khoảng cách xa. Sau khi UART nhận đọc khung dữ liệu, nó sẽ đếm số bit có giá trị là 1 và kiểm tra xem tổng số là số chẵn hay lẻ.

Có 2 loại Parity đó là Parity chẵn (even parity) và parity lẻ (odd parity). Parity chẵn nghĩa là số bit 1 trong trong data truyền cùng với bit Parity luôn là số chẵn, ngược lại nếu Parity lẽ nghĩa là số bit 1 trong data truyền cùng với bit Parity luôn là số lẽ. Bit Parity không phải là bit bắt buộc và vì thế chúng ta có thể loại bỏ bit này ra khỏi khung truyền.

Bit stop: Ngược lại với bit start,bit Stop sẽ truyền mức "1"dùng để thông báo kết thúc quá trình truyền dữ liệu. Bit stop có thể là 1; 1,5 hoặc 2. (là bit bắt buộc như Start bit).

Như vậy, quá trình truyền dữ liệu của UART diễn ra dưới dạng các gói dữ liệu, bắt đầu bằng một bit bắt đầu, đường mức cao được kéo xuống thấp. Sau bit bắt đầu là 5 đến 9 bit dữ liệu truyền trong khung dữ liệu của gói, theo sau là bit chẵn lẻ tùy chọn để xác minh việc truyền dữ liệu thích hợp. Sau cùng, một hoặc nhiều bit dừng được truyền ở nơi đường đặt ở mức cao. Thế là kết thúc một gói dữ liệu được truyền đi.

2.1.3. Chức năng và ứng dụng của UART

2.1.3.1. Chức năng

Chức năng chính của UART là truyền dữ liệu nối tiếp. Trong UART, giao tiếp giữa hai thiết bị có thể được thực hiện theo hai cách là giao tiếp dữ liệu nối tiếp và giao tiếp dữ liệu song song.

Có nghĩa rằng trong giao tiếp UART, hai UART giao tiếp trực tiếp với nhau. UART truyền chuyển đổi dữ liệu song song từ một thiết bị điều khiển như CPU thành dạng nối tiếp, truyền nó nối tiếp đến UART nhận, sau đó chuyển đổi dữ liệu nối tiếp trở lại thành dữ liệu song song cho thiết bị nhận.

Từ chức năng trên nó đã góp phần làm nền vai trò vô cùng quan trọng như các hệ thống nhúng, vi điều khiển và máy tính hầu hết sử dụng UART như một dạng giao thức giao tiếp phần cứng giữa thiết bị và thiết bị. Trong số các giao thức truyền thông hiện có, UART chỉ sử dụng hai dây cho bên truyền và bên nhận.

2.1.3.2. Ứng dụng của UART trong truyền dữ liệu

Thông thường, UART dùng để:

Giao tiếp máy tính với các thiết bị ngoại vi: UART là giao diện truyền thông chuẩn cho việc giao tiếp giữa máy tính và các thiết bị ngoại vi như chuột, bàn phím, máy in, cổng serial, v.v.

Truyền dữ liệu giữa vi xử lý và các cảm biến: UART được sử dụng để truyền dữ liệu giữa vi xử lý và các cảm biến như cảm biến nhiệt độ, cảm biến ánh sáng, cảm biến khoảng cách, v.v. Các cảm biến này thường được kết nối với vi xử lý thông qua giao diện UART để truyền dữ liệu về vi xử lý để xử lý và hiển thị.

Giao tiếp giữa các vi xử lý: UART được sử dụng để truyền dữ liệu giữa các vi xử lý trong các ứng dụng như mạng điều khiển, hệ thống nhúng và hệ thống điều khiển tự động.

Truyền dữ liệu giữa các thiết bị điện tử: UART được sử dụng để truyền dữ liệu giữa các thiết bị điện tử như vi xử lý, module RF, cổng Ethernet, v.v.

Giao tiếp truyền thông trong các thiết bị đo lường: UART được sử dụng trong các thiết bị đo lường như đồng hồ đo tốc độ, đồng hồ đo nhiệt độ, v.v.

Giao tiếp truyền thông trong các ứng dụng điện tử tiêu thụ ít điện năng: UART được sử dụng trong các ứng dụng điện tử tiêu thụ ít điện năng như các thiết bị IoT, cảm biến không dây, thiết bị điều khiển từ xa, v.v.

Trong thực tế thì không có phương thức truyền dẫn nào là tối ưu cả, tuy nhiên đối với UART thì đã gần như đáp ứng đủ hết nhu cầu hiện nay. Sau đây là một số ưu và nhược điểm của loại truyền dẫn không dây này:

*** Ưu điểm:**

Chỉ sử dụng hai dây để truyền dữ liệu.

Không cần tín hiệu đồng hồ.

Có một bit chẵn lẻ để cho phép kiểm tra lỗi.

Cấu trúc của gói dữ liệu có thể được thay đổi miễn là cả hai bên được thiết lập cho nó.

Phương pháp truyền đơn giản, giá thành thấp.

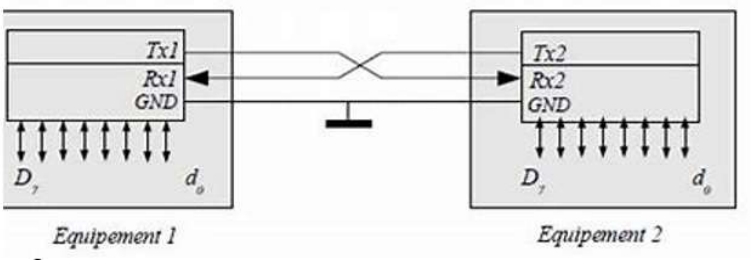
*** Nhược điểm:**

Kích thước của khung dữ liệu được giới hạn tối đa là 9 bit.

Không phù hợp với các hệ thống đòi hỏi nhiều thiết bị master và slave.

Tốc độ truyền của mỗi UART phải nằm trong khoảng 10%.

2.2. Đặc điểm và nguyên lý hoạt động



2.2.1. Đặc điểm

Trong sơ đồ UART có 3 dây cần lưu ý :

Transmitter (Tx): dây dùng để gửi dữ liệu.

Receiver (Rx): dây dùng để nhận dữ liệu.

Dây GND: dùng để tạo một mức tham chiếu dùng để so sánh với các tín hiệu trong giao tiếp. Nếu không có dây mass trong khối UART, các tín hiệu có thể không được đọc chính xác và gây ra các lỗi trong giao tiếp.

2.2.2. Nguyên lý hoạt động của UART

Chân Tx (truyền) của một chip kết nối trực tiếp với chân Rx (nhận) của chip kia và ngược lại. Quá trình truyền thường sẽ diễn ra ở 3.3V hoặc 5V. UART là một giao thức giữa một master và một slave. Trong đó một thiết bị được thiết lập để giao tiếp với chỉ một thiết bị khác.

Dữ liệu truyền đến và đi từ UART song song với thiết bị điều khiển. Khi tín hiệu gửi trên chân Tx, UART đầu tiên sẽ dịch thông tin song song này thành nối tiếp và truyền đến thiết bị nhận. Chân Rx của UART thứ 2 sẽ biến đổi nó trở lại thành song song để giao tiếp với thiết bị điều khiển.

Dữ liệu truyền qua UART đóng thành các gói (packet). Mỗi gói chứa 1 bit bắt đầu, 5 đến 9 bit dữ liệu (tùy thuộc vào UART), 1 bit chẵn lẻ tùy chọn và 1 hoặc 2 bit dừng.

Ngoài ra UART có thể truyền theo một trong ba chế độ:

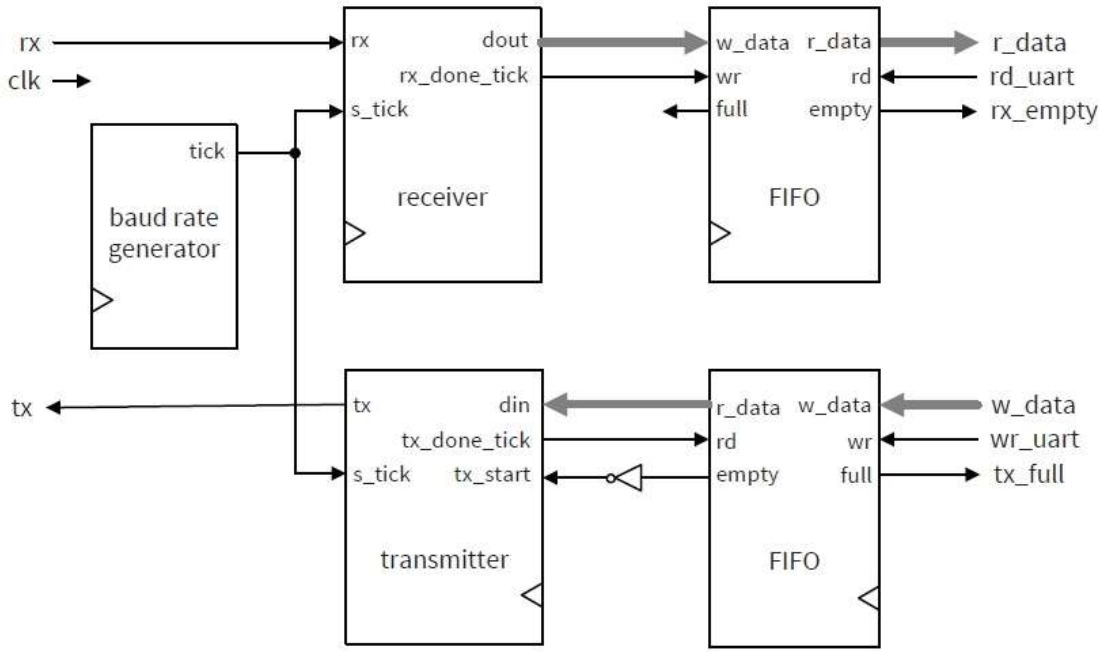
Simplex: Chỉ giao tiếp một chiều.

Half duplex: Dữ liệu đi theo một hướng tại một thời điểm.

Full duplex: Giao tiếp đồng thời đến và đi từ mỗi master và slave

CHƯƠNG 3: THIẾT KẾ GIAO THỨC UART

3.1. Sơ đồ khối thiết kế UART



Sơ đồ khối thiết kế bộ UART

3.2. Mô tả thiết kế bộ UART

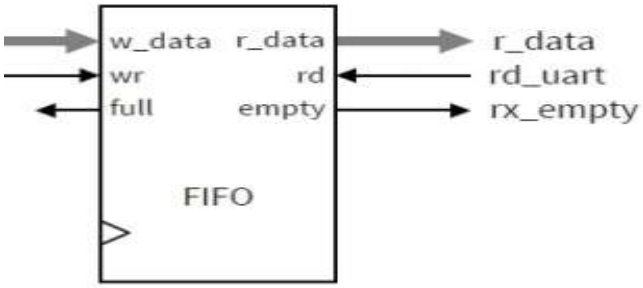
3.2.1. Khối tốc độ baud

Bộ tạo tốc độ baud (Baud Rate Generator) tạo ra tín hiệu lấy mẫu có tần số bằng đúng 16 lần tốc độ baud được chỉ định của UART. Để tránh làm giao các xung đồng hồ và vi phạm nguyên tắc thiết kế đồng bộ, tín hiệu lấy mẫu phải hoạt động như một chân enable thay vì xung đồng hồ đối với bộ thu UART. Các tốc độ baud tiêu chuẩn của UART bao gồm 50, 75, 110, 300, 600, 1.200, 2.400, 4.800, 9.600, 14.400, 19.200, 38.400, 57.600, 115.200, 128.000 và 230.400, 460.800.

Lấy ví dụ đối với tốc độ baud là 19.200, tốc độ lấy mẫu phải là 307.200 (tức là 19.200*16). Với tốc độ xung Clock hệ thống là 10 MHz, bộ tạo tốc độ baud cần một bộ đếm MOD-32 (10MHz/307.200). Thông thường, xung Clock hệ thống có tần số rất cao, thay vì dùng một loạt các bộ chia lớn để có được tốc độ truyền mong muốn, sẽ rẻ hơn nếu có một bộ chia lớn duy nhất theo sau bởi một bộ chia

nhỏ. Cách này sẽ hoạt động vì tần số UART tiêu chuẩn là ước số của 2. Ví dụ, cho tốc độ xung Clock hệ thống là 2.457.600, 2 ngõ ra mong muốn đạt được là 9.600 baud hoặc 19.200 baud. Theo lẽ thông thường, ta cần 1 bộ đếm MOD-128 cho ngõ ra 19.200 baud và 1 bộ đếm MOD-256 cho ngõ ra 9.600 baud. Thay vào đó, ta có thể dùng 1 bộ đếm MOD-128 (tạo ngõ ra 19.200 baud) theo sau đó là 1 đếm MOD-2 để lấy ngõ ra 9.600 baud.

3.2.2. Khối đệm FIFO



Khối lưu dữ liệu FIFO

Bộ đệm FIFO Trong tính toán và trong lý thuyết hệ thống, là một phương pháp để tổ chức thao tác cấu trúc dữ liệu. Trong đó mục nhập cũ nhất (đầu tiên), hoặc 'đầu' của hàng xếp, được xử lý đầu tiên. Quá trình xử lý như vậy tương tự như phục vụ mọi người trong khu vực hàng đợi trên cơ sở ai đến trước được phục vụ trước, theo cùng một trình tự mà họ đã đến cho tới đuôi của hàng đợi. Bộ đệm FIFO thường được sử dụng trong các mạch điện tử để đệm và điều khiển luồng giữa phần cứng và phần mềm. Ở dạng phần cứng, FIFO chủ yếu bao gồm một tập hợp các con trỏ đọc và ghi, logic lưu trữ và điều khiển. Bộ nhớ có thể là bộ nhớ truy cập ngẫu nhiên tĩnh (SRAM), flip-flops, chốt hoặc bất kỳ hình thức lưu trữ phù hợp nào khác. Đối với các FIFO có kích thước không nhỏ, SRAM hai cổng thường được sử dụng, trong đó một cổng dành riêng để ghi và cổng còn lại để đọc. Có nhiều cách để tạo một bộ đệm FIFO, tuy nhiên cách được sử dụng trong bài tập mô phỏng này là bộ FIFO vòng (array-base buffer). Như tên gọi của nó

(array-base), bộ đệm này được thực hiện dựa trên một mảng. Kèm theo đó là 2 con trỏ Write và Read. Mỗi khi nhận lệnh ghi, con trỏ pWrite sẽ ghi data vào bộ đệm, sau đó sẽ tăng lên 1 đơn vị. Mỗi khi nhận lệnh đọc, con trỏ pRead sẽ tăng lên một. Sau đó đọc giá trị từ bộ đệm ra. Khi 1 con trỏ tới được cuối mảng, nó sẽ cuộn lại vị trí đầu tiên.

Đó là lý do vì sao gọi đây là bộ đệm vòng.

Bộ đệm này gồm 2 cờ: empty (trống) và full (đầy).

- Cờ full: là trạng thái khi con trỏ ghi đã thực hiện ghi dữ liệu được một vòng tròn và gặp con trỏ đọc tại vòng tròn thứ 2. Nói cách khác, con trỏ đọc trùng với con trỏ ghi khi vòng quay con trỏ ghi lớn hơn con trỏ đọc 1 vòng. Dữ liệu chưa được đọc ra mà đã có tín hiệu ghi vào ô nhớ đó. Khi đó ta sẽ không được phép ghi dữ liệu vào nữa.

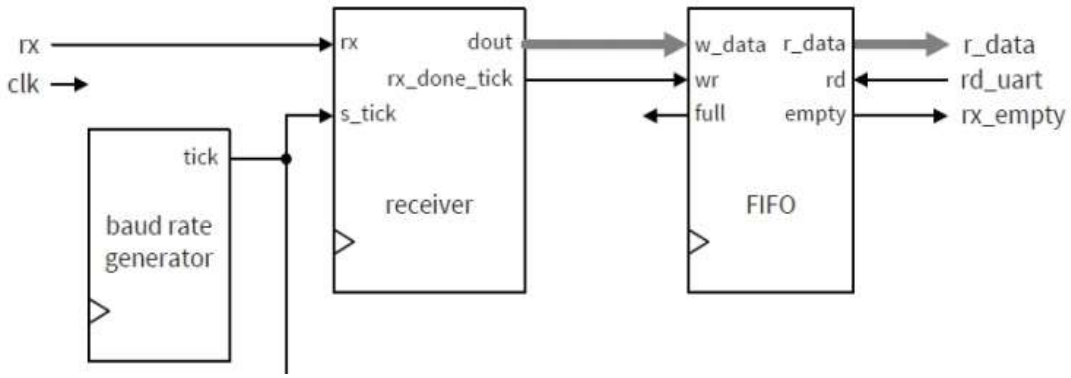
- Cờ empty: là trạng thái con trỏ đọc trùng với con trỏ ghi khi cả 2 con trỏ cùng một vòng. Dữ liệu chưa được ghi vào đã có tín hiệu đọc ra, xem như dữ liệu cũng bị mất.

- Độ sâu của FIFO: tương ứng số phần tử tối đa mà FIFO có thể lưu trữ được.

- Băng thông của FIFO: tương đương với kích thướng của một phần tử dữ liệu được đọc/viết trong một chu kỳ đọc/viết.

3.2.3. Khối nhận UART

3.2.3.1. Sơ lược



Khối nhận là một trong hai khối quan trọng nhất cấu thành UART (khối nhận và khối phát).

- Bộ nhận (UART reveiver): mạch lấy dữ liệu thông qua quá trình lấy mẫu.

- Bộ tạo tốc độ truyền (Baud rate generator): mạch tạo ra các tín hiệu enable lấy mẫu với tần số dựa vào tần số baud truyền.

- Mạch giao diện (Interface circuit): mạch cung cấp bộ nhớ đệm và kiểm soát trạng thái để cho điều khiển truy xuất và xử lí dữ liệu.

Bộ nhận (UART- Receiver) là bộ phận chính của khối thực hiện việc nhận dữ liệu nối tiếp từ đường truyền và kiểm tra, sau đó chuyển dữ liệu từ nối tiếp thành song song và đưa tới vi điều khiển hoặc máy tính.

Mạch giao diện (Interface circuit): trong một hệ thống UART thường được xem như là một mạch ngoại vi để truyền dữ liệu nối tiếp và các hệ thống truy cập và điều khiển cũng như như nhận dữ liệu một cách định kì. Vì thế mạch có 2 chức năng gồm tín hiệu thông báo có dữ liệu mới để tránh việc truy xuất dữ liệu bị trùng lập và cung cấp bộ nhớ đệm.

3.2.3.2. Hoạt động của bộ nhận UART

Nguyên lý hoạt động của bộ nhận được thể hiện ở lưu đồ thuật toán bên dưới.

Bộ nhận thực chất là một máy trạng thái Mealy với đầu ra của máy phụ thuộc vào trạng thái hiện tại và đầu vào. Các đầu vào của bộ nhận (UART Receiver) gồm tín hiệu xung clock, reset, tín hiệu s_tick, và đường tín hiệu truyền rx. Ngõ ra bộ nhận bao gồm dữ liệu ra song song và tín hiệu rx_done_tick.

Tín hiệu s_tick là tín hiệu enable lấy mẫu từ ngõ ra bộ tao tốc độ truyền (Baud rate generator) với tần số tín hiệu phát bằng 16 lần tốc độ baud truyền dùng để cung cấp cho bộ nhận tốc độ lấy mẫu và nó đã được giải thích rõ ở mục Bộ tạo tốc độ Baud .

Mạch hoạt động ở 4 trạng thái : (Idle), (Start), (Data) và (Stop):

- Trạng thái Idle: khi bộ nhận vừa nhận tín hiệu reset hoặc không có dữ liệu truyền tới, bộ nhận sẽ hoạt động ở trạng thái nhàn rỗi. Sau khi nhận được tín hiệu 0 tức là bit 0 start ở đầu mỗi frame dữ liệu, bộ nhận sẽ chuyển đến trạng thái Start.

- Trạng thái bắt đầu Start: khi hoạt động ở trạng thái start bộ nhận sẽ lấy mẫu 8 lần trong nửa chu kì xung của bit start để đảm bảo không phải tín hiệu nhiễu.

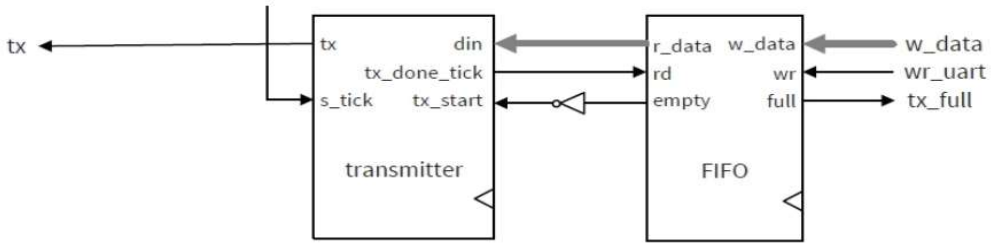
Nếu sau nửa chu kì xung tín hiệu vẫn là 0 thì bộ nhận sẽ chuyển đến trạng thái kế tiếp là dữ liệu (data) còn nếu không thì sẽ quay lại trạng thái nhàn rỗi (Idle).

- Trạng thái dữ liệu Data: Khi bộ nhận được chuyển đến hoạt động ở trạng thái Data, bộ nhận sẽ lấy mẫu 16 lần trong toàn thời gian nhận 1 bit dữ liệu để đảm bảo tính chính xác của tín hiệu nhận được. Sau khi lấy mẫu đủ 16 lần và đảm bảo tín hiệu là ổn định thì bit dữ liệu sẽ được dịch vào thanh ghi. Các bit dữ liệu được phát lần lượt từ bit LSB đến MSB của data và tín hiệu đi đến bộ nhận cũng đến lần lượt theo thứ tự từ LSB đến MSB. Sau khi nhận biết đã thu được đầy thanh ghi dữ liệu tức là thu đủ dữ liệu, bộ thu sẽ chuyển đến trạng thái dừng (stop).

- Trạng thái dừng (Stop): Khi hoạt động ở trạng thái dừng, bộ nhận sẽ lấy mẫu 1 bit stop cuối frame và sẽ quay lại trạng thái Idle khi kết thúc lấy mẫu vùng có độ rộng 1 xung clock. Đồng thời cờ rx_done_tick cũng lưu giá trị 1. Dữ liệu vừa nhận được sẽ được ghi vào bộ đệm FIFO.

3.3.4. Khối phát UART

3.3.4.1. Sơ lược



Khối phát UART

Khối phát là khối còn lại trong 2 khối quan trọng nhất cấu thành UART (khối nhận và khối phát). Theo sơ đồ khối hệ thống con khối phát UART (UART Transmitting subsystem) hình bên dưới, khối bao gồm 3 thành phần chính:

Bộ phát (UART- Transmitter): mạch đóng gói và phát dữ liệu.

Bộ tạo tốc độ truyền (Baud rate generator): mạch tạo ra các tín hiệu enable lấy mẫu với tần số dựa vào tần số baud truyền.

Mạch giao diện (Interface circuit): mạch cung cấp bộ nhớ đệm và kiểm soát trạng thái để cho điều khiển truy xuất và xử lí dữ liệu.

Bộ phát (UART- Transmitter) là bộ phận chính của khối thực hiện việc nhận dữ liệu song song từ hệ thống thông qua bộ đệm FIFO, sau đó chuyển dữ liệu từ song song thành nối tiếp và thêm các bit parity kiểm tra lỗi cũng như đóng gói truyền thành một frame hoàn chỉnh và truyền nối tiếp trên đường dây.

3.3.4.2. Hoạt động của bộ truyền UART

Nguyên lý hoạt động của bộ phát được thể hiện ở lưu đồ thuật toán bên dưới.

Bộ phát có thể được miêu tả dưới dạng máy trạng thái hoặc theo dạng hành vi với các chức năng tương tự. Các đầu vào của bộ phát (UART Transmisstor) gồm tín hiệu xung clock, tín hiệu s_tick, các tín hiệu enable và thông báo trạng thái trống bộ đệm FIFO và đường tín hiệu truyền đi. Ngõ ra bộ nhận bao gồm dữ liệu ra nối tiếp tx và tín hiệu tx_done_tick thông báo kết thúc truyền 1 frame.

Tín hiệu s_tick được lấy từ Baud rate generator với tốc độ bằng tốc độ baud để

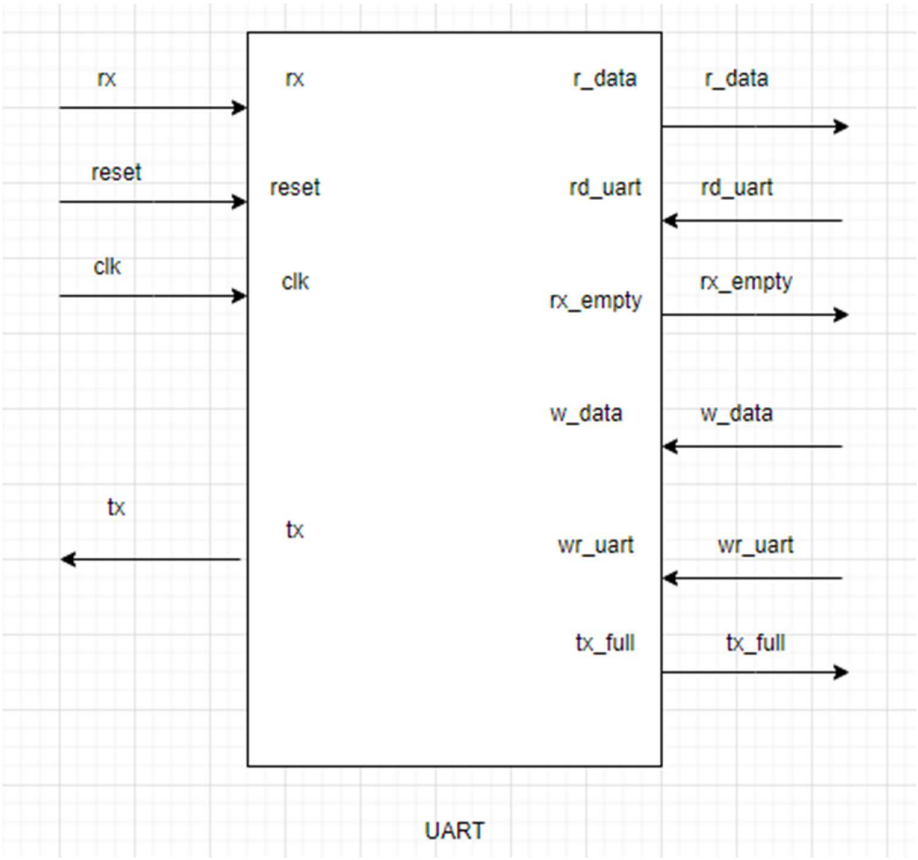
điều khiển tốc độ truyền dữ liệu và nguyên lí hoạt động đã được giải thích rõ ở mục Bộ tạo tốc độ Baud .

Ban đầu khi FIFO trống cờ Enable sẽ tắt và chờ đến khi bộ đệm FIFO nhận được dữ liệu được gửi từ máy tính. Sau khi bộ đệm không còn trống, cờ Enable bật lên 1, nếu tín hiệu Start cho phép mạch hoạt động cũng là 1 thì mạch sẽ vào trạng thái start và thanh ghi dữ liệu của bộ phát sẽ nạp dữ liệu từ bộ đệm FIFO vào, đồng thời tạo tín hiệu 0 trên đường truyền tương đương bit 0 start báo hiệu đầu frame truyền, cờ fsh tức tx_done_tick là thông báo kết thúc truyền xong 1 frame được gán 0. Sau đó dựa vào điều khiển của bộ Controller, tín hiệu start sau đó được đưa về 0 và bộ phát bắt đầu trạng thái phát. Trong trường hợp bỏ đi bit parity không bắt buộc, bộ phát sẽ lần lượt gửi đi từng bit dữ liệu nối tiếp trong thanh ghi dữ liệu. Đến khi hết dữ liệu thì bộ phátsẽ phát đi một bit 1 cuối tương đương với tín hiệu stop trong frame truyền. Sau đó 1 chu kì xung clock thì cờ fsh

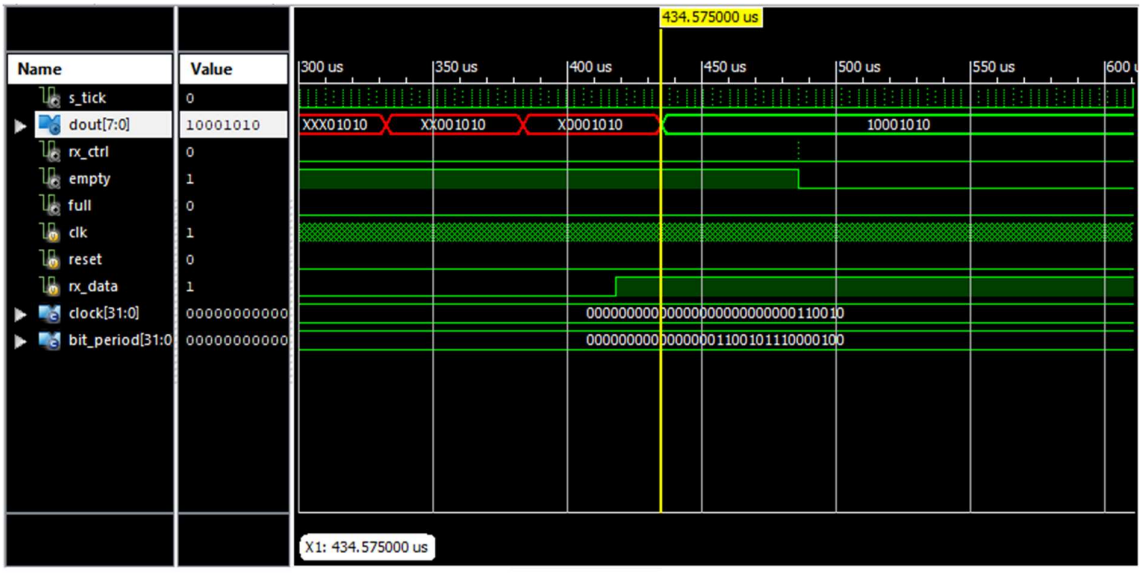
sẽ lên 1 thông báo hoàn tất việc truyền dữ liệu. Quá trình truyền 1 frame kết thúc và nếu còn dữ liệu trong bộ FIFO thì bộ điều khiển Controller.

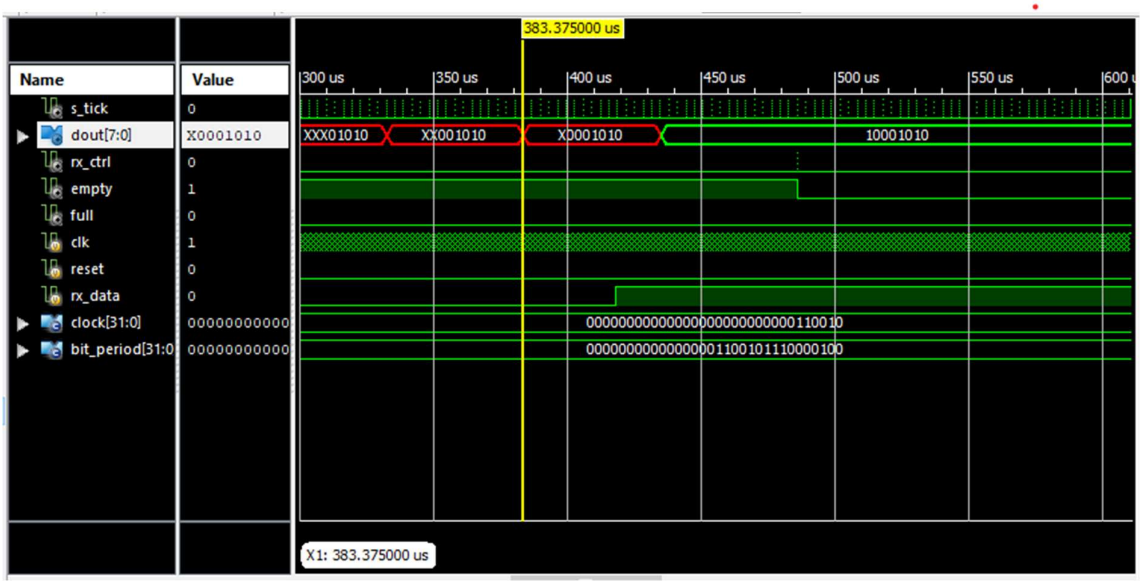
CHƯƠNG 4: ĐÁNH GIÁ QUA TEST BENCH

4.1. Mô hình test bench tổng quát



4.2.Kết quả


434.575000 us
Name
Value
s_tick
dout[7:0]
rx_ctrl
empty
full
clk
reset
rx_data
clock[31:0]
bit_period[31:0
10001010
300 us
350 us
400 us
450 us
500 us
550 us
XXX01010
XX001010
X0001010
10001010
X1: 434.575000 us


383.375000 us
Name
Value
s_tick
dout[7:0]
rx_ctrl
empty
full
clk
reset
rx_data
clock[31:0]
bit_period[31:0
X0001010
300 us
350 us
400 us
450 us
500 us
550 us
XXX01010
XX001010
X0001010
10001010
X1: 383.375000 us

4.3. Nhận xét và đánh giá

Nhận xét về kết quả sau khi mô phỏng testbench và so sánh với lý thuyết:

- Kết quả mô phỏng trên testbench cho thấy khối nhận UART đã nhận dữ liệu nối tiếp và chuyển thành dữ liệu ra song song đúng như lý thuyết đã đề cập. Hệ thống đã truyền một dữ liệu 8 bit là số hex 8AH, tương ứng với số nhị phân là 10001010, hình ảnh testbench cho thấy khối nhận UART nhận đúng dữ liệu là 10001010.

- Về tốc độ nhận: xét giữa khoảng thời gian khi nhận 1 bit từ 383.375us đến 434.575us (như trên ảnh), thời gian nhận 1 bit là (434.575 - 383.375)us = 51.2us, tức trong 1s có thể nhận được 1s/(51.2us) = 19531 bit. So sánh với tốc độ baud đã chọn là 19200bps, tốc độ nhận ở khối nhận lại là 19531bps cho thấy có sai số giữa lý thuyết và thực nghiệm. Mức sai số là ~1.72%. Nguyên nhân của sai số này là do khi thực hiện chia để đếm mod lấy mẫu đã không lấy nguyên vẹn thương số mà chỉ lấy phần nguyên của thương. Cụ thể, xung clock hệ thống là 10MHz, baud rate 19200, lấy mẫu 16 lần: (10MHz/(19200*16)) = 32.5520833 nhưng đếm mod chỉ lấy phần nguyên là 32 do đó dẫn đến sai số. Thêm vào đó, sai số này càng lớn khi tốc độ baud được chọn càng lớn.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Chúng em đã hoàn thiện quá trình thiết kế các module của UART theo các bước lên specification, phân tích yêu cầu, lên sơ đồ khối, lập sơ đồ chuyển trạng thái, sơ đồ thuật toán, lập trình verilog và kiểm thử. Các kết quả mô phỏng thu được đã đáp ứng được yêu cầu đặt ra trước khi thiết kế.

5.2. Hướng phát triển

Tuy thiết kế đã đáp ứng được yêu cầu ban đầu đặt ra nhưng để tối ưu hơn ta có thể thiết kế lõi UART có thêm 1 số tính năng:

Cấu hình tốc độ Baud và Data truyền trên cùng 1 thanh ghi.

Thiết kế thanh ghi dịch để có thể truyền thêm bit Parity kiểm tra chẵn lẻ.

Thiết kế thêm 1 số tín hiệu báo việc truyền/nhận đang diễn ra.

Thiết kế thêm một số chuẩn giao tiếp giữa UART với MCU và các thiết bị ngoại vi khác.

PHỤ LỤC

Khối tốc độ baud

```
module Baud_rate
#(parameter N = 6, M = 32 //dem mod 10MHz/(19200*16) = 32)
(
input wire clk, reset,
output wire tick
);
reg [N-1:0] q;
always @(posedge clk, posedge reset)
if (reset)
q <= 0;
else
q <= q + 1;
assign tick = (q == (M - 1)) ? 1'b1 : 1'b0;
endmodule
```

Khối FIFO:

```
module FIFO
#(parameter w=8,s=4)
(empty,full,rd,wr,rs,wd,r);

input wire [w-1:0]wd;
input rd,wr,rs;
output reg empty,full;
output reg[w-1:0]r;

reg [w-1:0] mem [4**s-1:0];
```

```
reg[s-1:0] wpt;
reg[s-1:0] rpt;

always @ (posedge rd, posedge rs,posedge wr)
begin
      if(rs)
        begin
          empty<=1;
          full<=0;
          wpt<=0;
          rpt<=0;
        end
      else if (rd)
        begin
              if(rpt==wpt)
              empty<=1;
                if (~empty)
                begin
                      r<=mem[rpt];
                      rpt<=rpt+1'b1;
                      full<=0;
                end
          end
                  else if(wr)
                  begin
                        if (wpt+1==rpt)
                          full<=1;
                          if (~full)
                          begin
```

```
                    mem[wpt]<=wd;
                    wpt<= wpt+1'b1;
                    empty<=0;
                end
            end
end
endmodule
```

Transmitter:

```
module Transmitter
#(parameter dbit=8, stop=1) //so bit data & bit stop
(tx,data,str,fsh,clk,enable);
input [dbit-1:0] data;
input clk,enable,str;
output reg tx,fsh;
reg [8:0] c;
reg [dbit-1:0] d;

always @(posedge clk)
if(enable)
case (str) //enable: tin hieu cho phep cap tu FIFO
  1'b1: //khoi tao va truyen bit start
  begin
  d<=data;
  c<=9'b111111110; //bien dem databit & stop bit
  tx<=0;
  fsh<=0; //tin hieu bao hoan thanh phat 1 frame data
  end
  1'b0: //truyen data
```

```
  begin
                if (c[8]) //data tu fifo duoc truyen noi tiep qua ngo ra tx
                    begin
                    tx<= d[0];
                    d<=d>>1;
                    c<=c<<1;
                    end
                else //chen them bit stop
                begin
                            tx<=1;
                            if(tx)
                            fsh<=1;/*tin hieu fsh len 1 thong bao hoan thanh
viec truyen toi fifo, fifo tao tin hieu enable va xuat frame du lieu tiep theo*/
                end
    end
  endcase
  endmodule
```

Interface (first cycle setup):

```
module Controller(fsh,empty,str,rd,enb,clk);
input wire clk,fsh,empty;
output reg str,rd,enb;
reg c;

always @*
begin
  case(empty)
        1'b0:
                begin
```

```
                enb=1;
            if(c)
                begin
                    rd=1;
                    str=1;
                    c=0;
                end
            else
                begin
                    rd=0;
                    str=0;
                end
            if(fsh)
                begin
                    rd=1;
                    str=1;
                end
            end
        1'b1:
            begin
                c=1;
                enb=0;
            end
  endcase
end
endmodule
```

Complete UART_TRANSMITTER_SUBSYSTEM:

```
module Transmitter_subsystem(tx,clk,wd,wr,full,rs,EMPTY,fsh,str,rd);
input wire clk,wr,rs;
```

```
input wire [7:0] wd;
output wire full,tx,EMPTY,fsh,str,rd;
wire [7:0] net;
wire d;
FIFO B1(.wd(wd),.wr(wr),.full(full),.empty(EMPTY),.rd(rd),.r(net),.rs(rs));
Controller B2(.fsh(fsh),.enb(d),.str(str),.rd(rd),.empty(EMPTY),.clk(clk));
Transmitter B3(.tx(tx),.clk(clk),.str(str),.enable(d),.fsh(fsh),.data(net));

Endmodule
```

Receiver:

```
module UART_receiver(clk, reset, rx_data, s_tick, rx_ctrl, dout);
input wire clk, reset, rx_data, s_tick;
output wire rx_ctrl;
output wire [7:0] dout;
localparam [1:0]
  S_IDLE = 2'b00,
  S_START = 2'b01,
  S_DATA = 2'b10,
  S_STOP = 2'b11;

reg [1:0] current_state, next_state;
reg [7:0] r_dout;
reg [3:0] r_count;
reg   r_rx_ctrl;
reg   r_rx_data;
reg [2:0] r_check_8bit;
//GHI DATA NHAN VAO THANH GHI
always @(posedge clk)
```

```
  begin
        r_rx_data <= rx_data;
  end
//THANH GHI TRANG THAI current_state
always @(posedge clk, posedge reset)
  begin
        if (reset)
              current_state <= S_IDLE;
        else
              current_state <= next_state;
  end
//TIEN TRINH TRONG MOI TRANG THAI current_state
always @(posedge clk)
  case(current_state [1:0])
        S_IDLE:
        begin
                    r_rx_ctrl <= 0;
                    r_count <= 0;
                    r_check_8bit <= 0;
                    if (r_rx_data == 1'b0)
                          next_state <= S_START;
                    else
                          next_state <= S_IDLE;
        end

        S_START:
        begin
        if (s_tick)
                          if (r_count == 7)
```

```
                    begin
                            if (r_rx_data == 1'b0)
                            begin

                             next_state <= S_DATA;

                            r_count <= 0;
                                    end
                            else next_state <= S_IDLE;
                    end
                else
                    begin
                            r_count <= r_count + 1;
                            next_state <= S_START;
                    end
end

S_DATA:
begin
if (s_tick)
        if (r_count == 15)
            begin
                r_count <= 0;
                r_dout[r_check_8bit]<= r_rx_data;
                    if (r_check_8bit == 3'b111)
                    begin
                            next_state <= S_STOP;
                            r_check_8bit <= 0;
                    end
```

```
                                else
                                begin
                                r_check_8bit <= r_check_8bit +1;
                                next_state <= S_DATA;
                                end
                        end
                else
                        begin
                                r_count <= r_count + 1;
                                next_state <= S_DATA;
                        end
end

S_STOP:
begin
if (s_tick)
                        if (r_count == 15)
                                begin
                                        r_rx_ctrl <= 1;
                                        r_count <= 0;
                                        next_state <= S_IDLE;
                                end
                        else
                                begin
                                r_count <= r_count + 1;
                                next_state <= S_STOP;
                                end
end
default: next_state <= S_IDLE;
```

```
    endcase
  assign rx_ctrl = r_rx_ctrl;
  assign dout = r_dout;
  endmodule
```

UART_Rx_Subsystem:

```
  module UART_comp
  #(parameter w = 8, s = 4,N = 6, M = 32)
  (
  input wire clk, reset,
  input wire rx_data,
  output wire s_tick,
  output wire [w-1:0] dout,
  output wire rx_ctrl, empty, full);
  Baud_rate #(.N(N),.M(M)) B(.clk(clk),.reset(reset),.tick(s_tick));
  UART_receiver
  R(.clk(clk),.reset(reset),.rx_data(rx_data),.s_tick(s_tick),.rx_ctrl(rx_ctrl),.do
ut(dout));
  FIFO                                                    #(.w(w),.s(s))
F(.wd(dout),.rs(reset),.wr(rx_ctrl),.rd(),.empty(empty),.full(full),.r());

  endmodule
```

Test bench:

```
module sub_tb;

    // Inputs
    reg clk;
    reg [7:0] wd;
    reg wr;
    reg rs;
```

```
// Outputs
wire tx;
wire full;
wire EMPTY;
wire fsh;
wire str;
wire rd;

// Instantiate the Unit Under Test (UUT)
Transmitter_subsystem uut (
        .tx(tx),
        .clk(clk),
        .wd(wd),
        .wr(wr),
        .full(full),
        .rs(rs),
        .EMPTY(EMPTY),
        .fsh(fsh),
        .str(str),
        .rd(rd)
);

initial begin
        // Initialize Inputs
        clk = 0;
        wd = 0;
        wr = 0;
        rs = 0;
```

```
        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here

    end
  initial
    begin
        forever #5 clk=~clk;
    end
initial
begin
#2 rs=1;
#5 rs=0;
#7 wr=1;
        wd=8'b01001110;
#5
        wr=0;
#10
        wr=1;
        wd=8'b01110110;
#5
        wr=0;
#10
        wr=1;
        wd=8'b10110110;
#5 wr=0;
end
```

```
endmodule

module UART_comp_tb;
parameter clock = 50;
parameter bit_period = 52100;
// Inputs
reg clk;
reg reset;
reg rx_data;
// Outputs
wire s_tick;
wire [7:0] dout;
                wire rx_ctrl;
                wire empty;
                wire full;

// Instantiate the Unit Under Test (UUT)
UART_comp                                                                uut
(.clk(clk),.reset(reset),.rx_data(rx_data),.s_tick(s_tick),.dout(dout),.rx_ctrl(rx_ctr
l),.empty(empty),.full(full));
//Tao task de thuc hien gui du lieu cho bo receiver theo toc do baud
task UART_SEND_DATA;
input [7:0] i_Data;
integer ii;
begin
// Send Start Bit
rx_data <= 1'b0;
#(bit_period);
// Send Data Byte
```

```
for (ii=0; ii<8; ii=ii+1)
begin
        rx_data <= i_Data[ii];
        #(bit_period);
end
// Send Stop Bit
rx_data <= 1'b1;
#(bit_period);
end
endtask
always
#(clock/2) clk = ~clk;
// Main Testing:
initial
begin
                clk = 0;
                reset = 1;
                rx_data = 1'b1;
                #1000;
                reset = 0;
                @(posedge clk);
                UART_SEND_DATA(8'h8A); //send ‘10001010’
end
endmodule
```

TÀI LIỆU THAM KHẢO

1. Báo cáo môn học : “ Thiết kế VLSI Thiết kế chuẩn giao tiếp UART bằng verilog” Đại học Bách Khoa Hà Nội.

2. Đồ án chuyên ngành “ Thiết kế bộ truyền nhận UART 8 nạp trên kit FPGA “ Đại học Bách Khoa Đà Nẵng, Đỗ Tiến Thành, Đà Nẵng, 2014.

3. Giao tiếp UART là gì? Cách thức hoạt động, ưu nhược điểm và các ứng dụng, ngày truy cập: 28/4/2023

Link: dientusangtaovn.com

4. Khái niệm cơ bản về truyền thông UART, sơ đồ khối, ứng dụng | News-Cáo công nghệ, ngày truy cập: 28/4/2023

Link: caocongnghe.com

5. Basics of UART Communication, ngày truy cập: 28/4/2023

Link: circuitbasics.com

6. Hệ thống truyền thông nối tiếp bất đồng bộ UART, Trần Thành Lũy, ĐHSPKT.TPHCM