|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**  **VIỆN ĐIỆN TỬ - VIỄN THÔNG**  Ảnh có chứa văn bản, ký hiệu, thực phẩm  Mô tả được tạo tự động  **BÁO CÁO VLSI**  **Đề tài: Implementation of Universal Asynchronous Receiver Transmitter using System Verilog code**  **Nhóm sinh viên thực hiện:**   |  |  |  | | --- | --- | --- | | **Tên sinh viên** | **MSSV** | **Mã lớp** | | Phạm Ngọc Lâm | 20182628 | 129281 | | Phạm Minh Đức | 20172476 | 129281 |   **Giảng viên hướng dẫn:** TS. Phan Xuân Vũ  Hà Nội - 2022 |

MỤC LỤC

[MỤC LỤC 1](#_Toc92926327)

[DANH MỤC HÌNH ẢNH i](#_Toc92926328)

[DANH MỤC BẢNG ii](#_Toc92926329)

[LỜI NÓI ĐẦU iii](#_Toc92926330)

[CHƯƠNG 1. GIỚI THIỆU (INTRODUCTION) 1](#_Toc92926331)

[1.1 Giới thiệu chung 1](#_Toc92926332)

[1.2 Truyền thông UART 1](#_Toc92926333)

[1.2.1 Cấu trúc của Data framing 1](#_Toc92926334)

[1.2.2 UART Communication 2](#_Toc92926335)

[CHƯƠNG 2. ĐẶC TẢ THÔNG SỐ KĨ THUẬT (SPECIFICATION) 3](#_Toc92926336)

[2.1 UART Architecture 3](#_Toc92926337)

[2.1.1 Block diagram 3](#_Toc92926338)

[2.1.2 Interface signals 3](#_Toc92926339)

[2.1.3 Function description 4](#_Toc92926340)

[2.1.4 Architecture 4](#_Toc92926341)

[2.2 Module Clock Generator (uart\_generator\_clock) 6](#_Toc92926342)

[2.2.1 Interface signals 6](#_Toc92926343)

[2.3 Module FIFO (uart\_fifo) 6](#_Toc92926344)

[2.3.1 Interface signals 7](#_Toc92926345)

[2.4 Module Transmitter (uart\_transmitter) 7](#_Toc92926346)

[2.4.1 Interface signals 7](#_Toc92926347)

[2.4.2 Function description 8](#_Toc92926348)

[2.4.3 Transmitter architecture 9](#_Toc92926349)

[2.5 Module Receiver (uart\_receiver) 9](#_Toc92926350)

[2.5.1 Interface signals 10](#_Toc92926351)

[2.5.2 Receiver architecture 11](#_Toc92926352)

[CHƯƠNG 3. KIỂM THỬ (VERIFICATION) 12](#_Toc92926353)

[3.1 Kế hoạch kiểm thử 12](#_Toc92926354)

[3.1.1 Kịch bản – Test case 12](#_Toc92926355)

[3.2 Kết quả và đánh giá 12](#_Toc92926356)

[3.2.1 Module instruction\_fetch 12](#_Toc92926357)

[3.2.2 Module instruction\_decode 12](#_Toc92926358)

[3.2.3 Module execute 12](#_Toc92926359)

[3.2.4 Module memory\_access 13](#_Toc92926360)

[3.2.5 Module register\_write 13](#_Toc92926361)

[3.2.6 Register 13](#_Toc92926362)

[CHƯƠNG 4. KẾT LUẬN 14](#_Toc92926363)

[TÀI LIỆU THAM KHẢO 15](#_Toc92926364)

DANH MỤC HÌNH ẢNH

[Hình 1.1. Data Framing UART 1](#_Toc92926299)

[Hình 1.2. UART Communication 2](#_Toc92926300)

[Hình 2.1 UART Block diagram 3](#_Toc92926301)

[Hình 2.2. TX\_status\_register 4](#_Toc92926302)

[Hình 2.3. RX\_status\_register 4](#_Toc92926303)

[Hình 2.4. Sơ đồ kiến trúc tổng quát 4](#_Toc92926304)

[Hình 2.5. Sơ đồ kiến trúc chi tiết 5](#_Toc92926305)

[Hình 2.6. uart\_generator\_clock block diagram 6](#_Toc92926306)

[Hình 2.7. uart\_fifo block diagram 6](#_Toc92926307)

[Hình 2.8. uart\_transmitter block diagram 7](#_Toc92926308)

[Hình 2.9. Tín hiệu điều khiển và dataflow trong UART transmitter 8](#_Toc92926309)

[Hình 2.10. ASMD uart\_transmitter 9](#_Toc92926310)

[Hình 2.11. uart\_receiver block diagram 9](#_Toc92926311)

[Hình 2.12. ASMD uart\_receiver 11](#_Toc92926312)

[Hình 3.1 Mô phỏng timing diagram khối instruction\_fetch 12](#_Toc92926313)

[Hình 3.2 Mô phỏng timing diagram khối instruction\_decode 12](#_Toc92926314)

[Hình 3.3 Mô phỏng timing diagram khối execute 12](#_Toc92926315)

[Hình 3.4 Mô phỏng timing diagram khối memory\_access 13](#_Toc92926316)

[Hình 3.5 Mô phỏng timing diagram khối register\_write 13](#_Toc92926317)

[Hình 3.6 Mô phỏng 32 Registers 13](#_Toc92926318)

[Hình 3.10 Mô phỏng Data\_Memory 13](#_Toc92926319)

DANH MỤC BẢNG

[Bảng 2.1 uart\_protocol port description 3](#_Toc92926320)

[Bảng 2.2. uart\_generator\_clock port description 6](#_Toc92926321)

[Bảng 2.3. uart\_generator\_clock parameter 6](#_Toc92926322)

[Bảng 2.4. uart\_fifo port description 7](#_Toc92926323)

[Bảng 2.5. uart\_fifo parameter 7](#_Toc92926324)

[Bảng 2.6. uart\_transmitter port description 7](#_Toc92926325)

[Bảng 2.7. uart\_receiver port description 10](#_Toc92926326)

LỜI NÓI ĐẦU

UART, hay máy thu-phát không đồng bộ phổ quát, là một trong những giao thức truyền thông giữa thiết bị và thiết bị được sử dụng nhiều nhất. Các hệ thống nhúng, bộ 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 các đầu truyền và nhận của nó. Do vậy, nhóm chúng em triển khai UART sử dụng code System Verilog với các phần như sau:

**Chương 1: Giới thiệu**

**Chương 2: Đặc tả thông số kĩ thuật**

**Chương 3: Kiểm thử**

**Chương 4: Kết luận**

Nhóm chúng em xin chân thành cảm ơn TS Phan Xuân Vũ đã tận tâm hướng dẫn chúng em trong quá trình thực hiện bài tập lớn cũng như hoàn hiện báo cáo này !

# GIỚI THIỆU (INTRODUCTION)

*Chương này giới thiệu khái quát và các kiến thức cơ bản về UART.*

## Giới thiệu chung

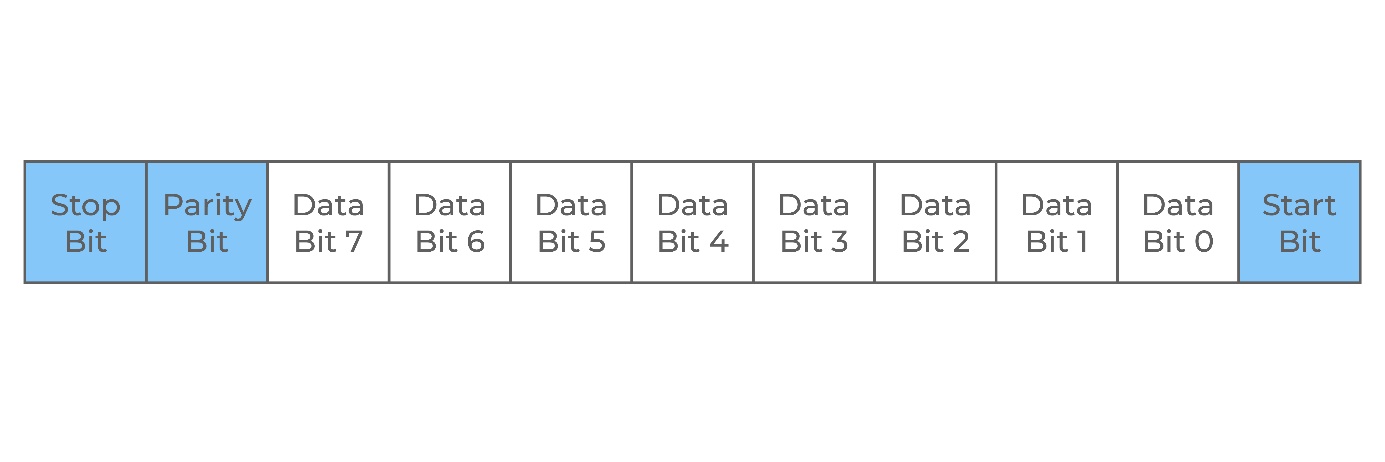
Universal asynchronous receiver – transmitter (UART) là một thiết bị cho truyền thông nối tiếp không đồng bộ, trong đó định dạng dữ liệu và tốc độ truyền tải được cấu hình. Nó gửi từng bit dữ liệu một, từ LSB đến MSB, được gói trong một khung truyền có các bit start và stop.

Nó là một trong những thiết bị giao tiếp máy tính sớm nhất, được sử dụng để gắn máy viết chữ từ xa cho một bảng điều khiển. Nó cũng là một hệ thống phần cứng ban đầu cho Internet.

UART thường là một mạch tích hợp (IC) được sử dụng cho giao tiếp nối tiếp qua máy tính hoặc cổng nối tiếp thiết bị ngoại vi.

## Truyền thông UART

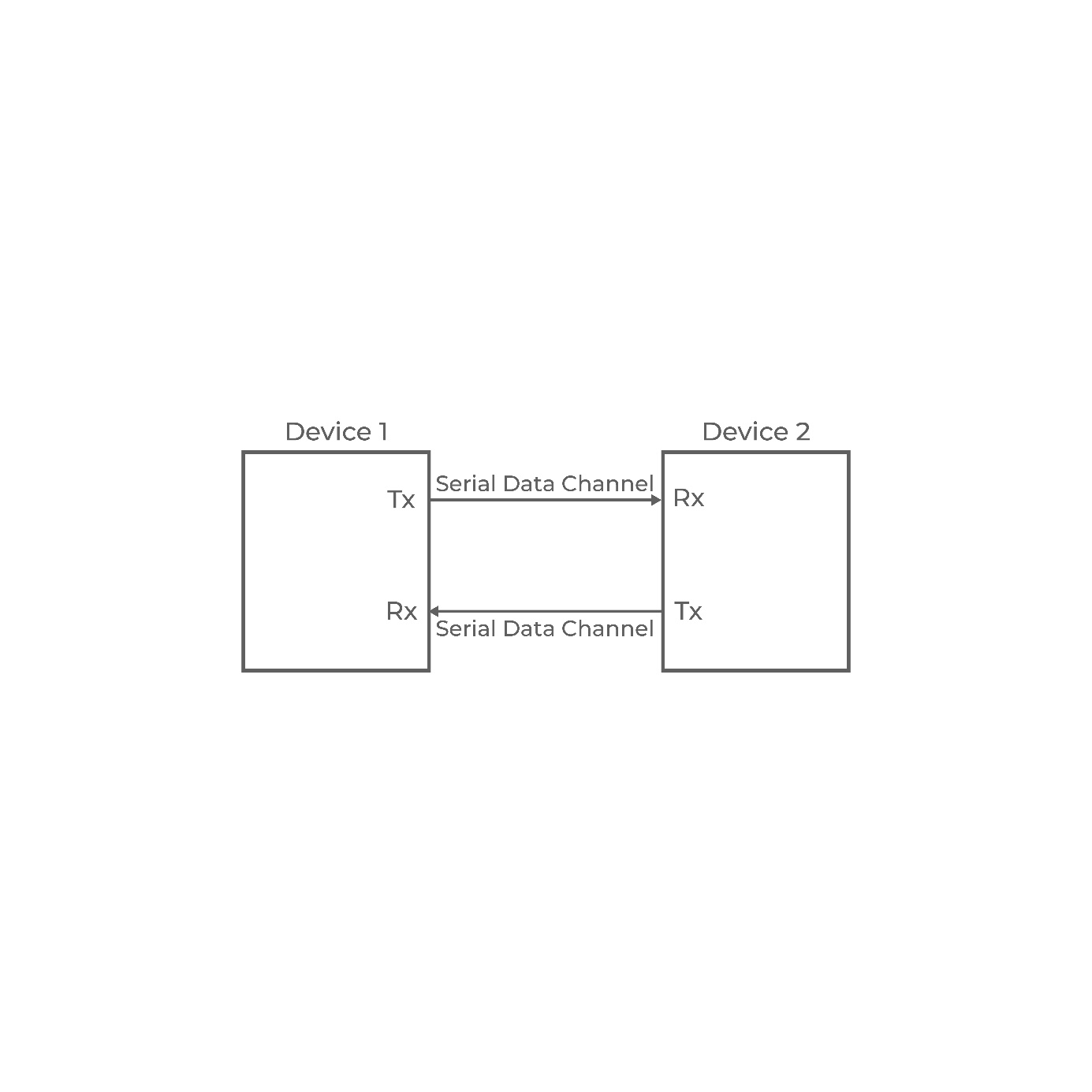
### Cấu trúc của Data framing



Hình 1.1. Data Framing UART

* Start Bit: Bit đồng bộ hóa được đặt trước dữ liệu. Để bắt đầu truyền dữ liệu, UART kéo đường dữ liệu từ mức điện áp cao (Mức 1) xuống mức điện áp thấp (Mức 0). Chỉ có một Start Bit.
* Stop Bit: Bit dừng được đặt ở cuối của gói dữ liệu. Thường chỉ sử dụng 1 bit. Đề dừng truyền dữ liệu, UART giữ đường dữ liệu ở mức điện áp cao.
* Parity Bit: Bit chẵn lẻ cho phép người nhận đảm bảo dữ liệu được thu thập có đúng hay không. Bit này không được sử dụng rộng rãi nên không bắt buộc.

### UART Communication



Hình 1.2. UART Communication

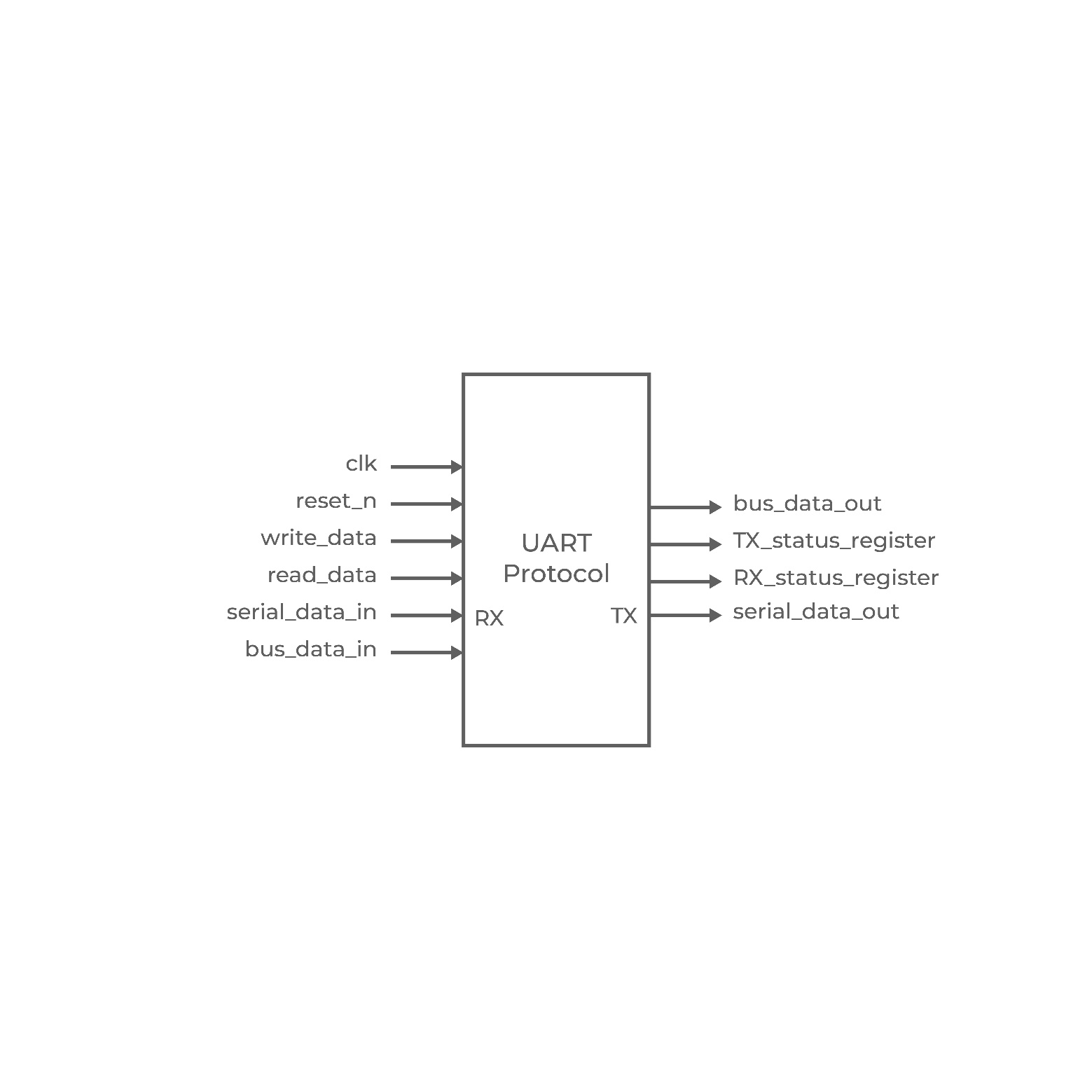
Trong giao tiếp dữ liệu nối tiếp, dữ liệu có thể được truyền qua một cáp hoặc một đường dây ở dạng bit-bit và nó chỉ cần hai cáp.

# ĐẶC TẢ THÔNG SỐ KĨ THUẬT (SPECIFICATION)

*Chương này mô tả thông số kĩ thuật và kiến trúc chi tiết của từng phần có trong kiến trúc UART được nhóm triển khai.*

## UART Architecture

### Block diagram



Hình 2.1 UART Block diagram

Hình 2.1 mô tả các Input và Output của module UART protocol.

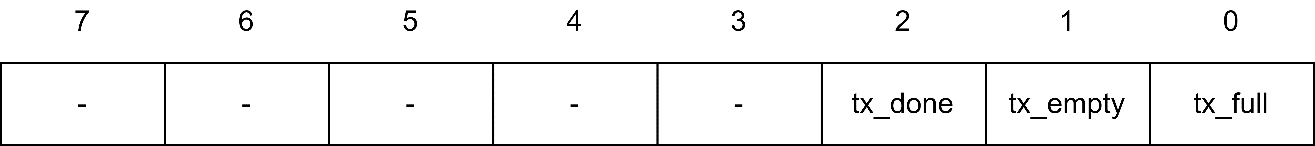
### Interface signals

Bảng 2.1 uart\_protocol port description

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal name** | **Width** | **Input/Output** | **Description** |
| **clk** | 1 | Input | Tín hiệu xung đồng hồ |
| **reset\_n** | 1 | Input | Tín hiệu reset tích cực mức thấp |
| **write\_data** | 1 | Input | Tín hiệu ghi vào FIFO từ CPU |
| **read\_data** | 1 | Input | Tín hiệu lấy dữ liệu trong FIFO từ CPU |
| **serial\_data\_in** | 1 | Input | Dữ liệu vào nối tiếp |
| **bus\_data\_in** | DATA\_SIZE | Input | Bus dữ liệu phía RX |
| **bus\_data\_out** | DATA\_SIZE | Output | Bus dữ liệu phía TX |
| **TX\_status\_register** | 8 | Output | Thanh ghi trạng thái TX |
| **RX\_status\_register** | 8 | Output | Thanh ghi trạng thái RX |
| **serial\_data\_out** | 1 | Ouput | Dữ liệu ra nối tiếp |

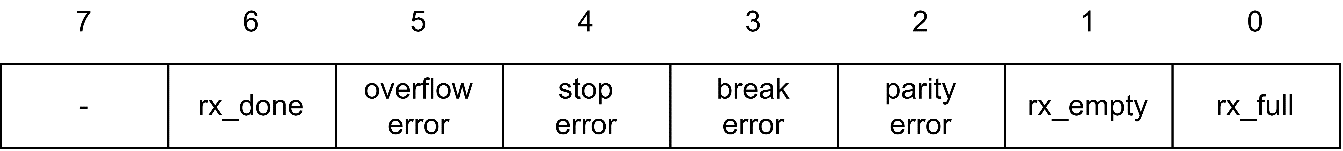
Thanh ghi trạng thái được biểu diễn như sau:

* TX\_status\_register:



Hình 2.2. TX\_status\_register

* RX\_status\_register



Hình 2.3. RX\_status\_register

### Function description

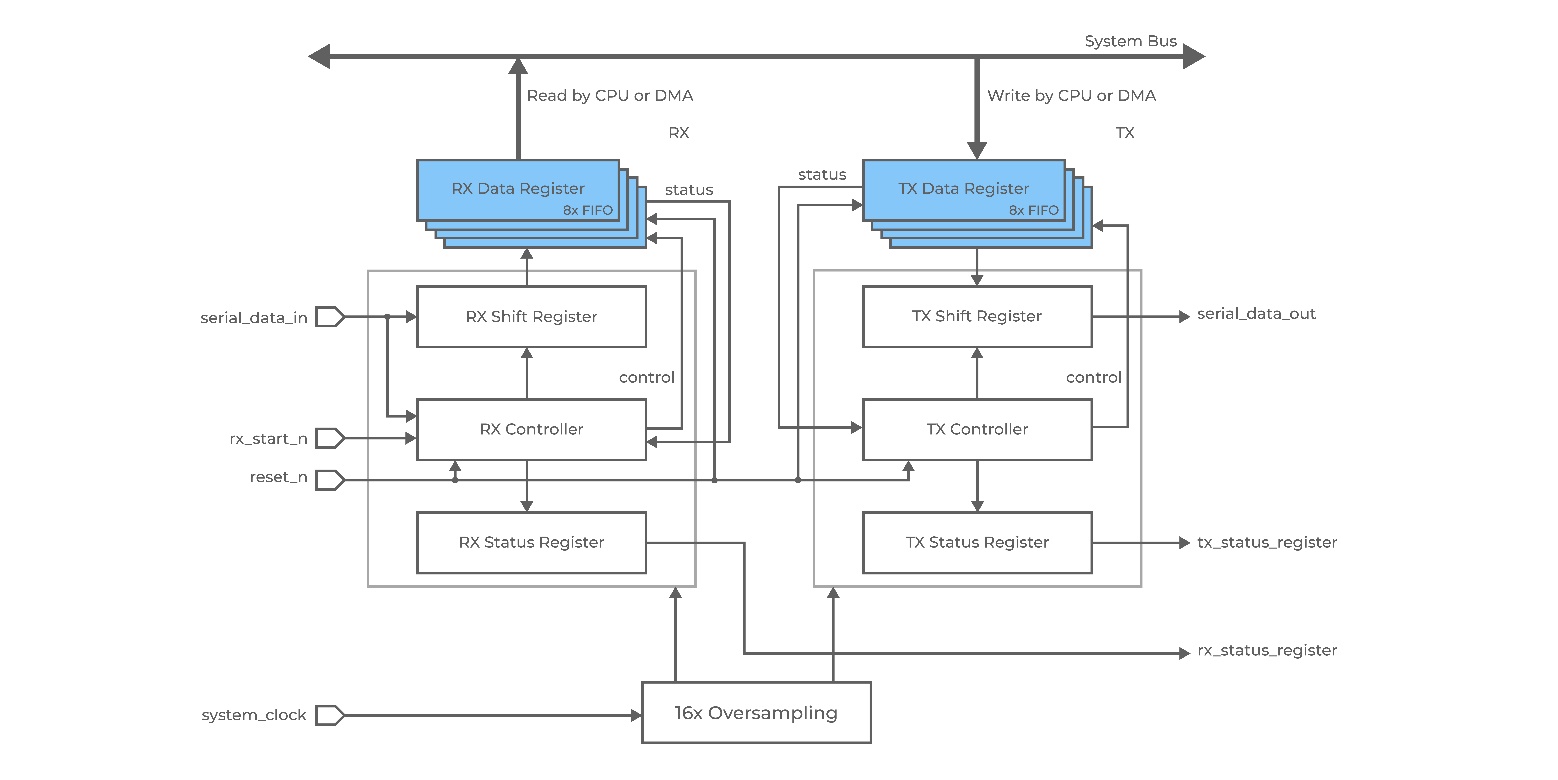
**Features:**

* Auto tuning baud generation (default 9600 bps)
* Detection of Framing, Parity, Stop, Break, Overflow errors
* Full Duplex
* 16x oversampling
* RX and TX data buffers (default = 8)
* 8 bit data
* start and stop bit

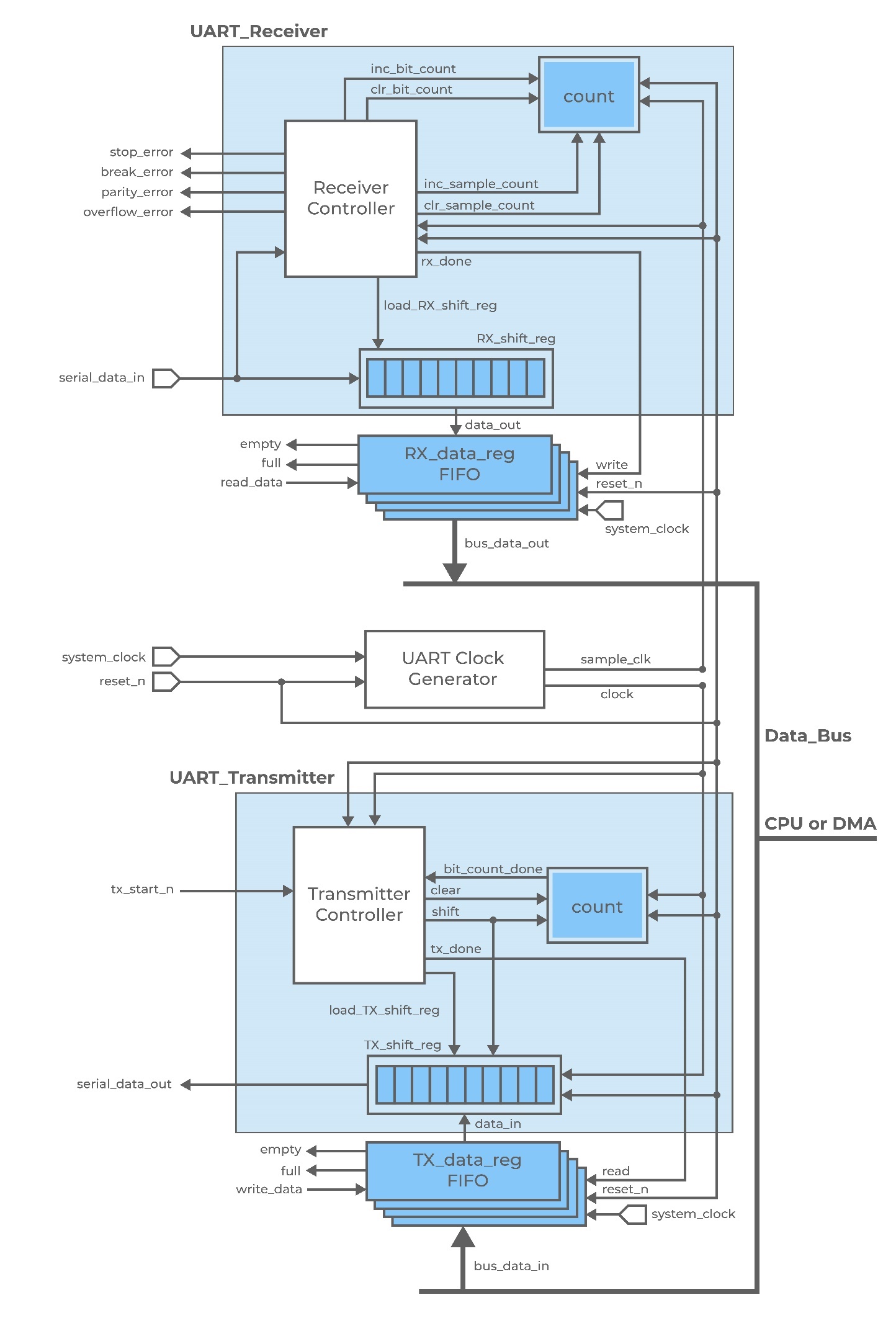
**Module:**

* Clock Generator
* FIFO
* Transmitter
* Receiver
* UART Protocol

### Architecture

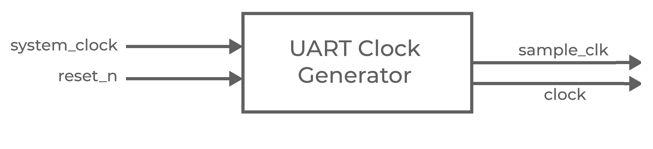


Hình 2.4. Sơ đồ kiến trúc tổng quát



Hình 2.5. Sơ đồ kiến trúc chi tiết

## Module Clock Generator (uart\_generator\_clock)



Hình 2.6. uart\_generator\_clock block diagram

### Interface signals

Bảng 2.2. uart\_generator\_clock port description

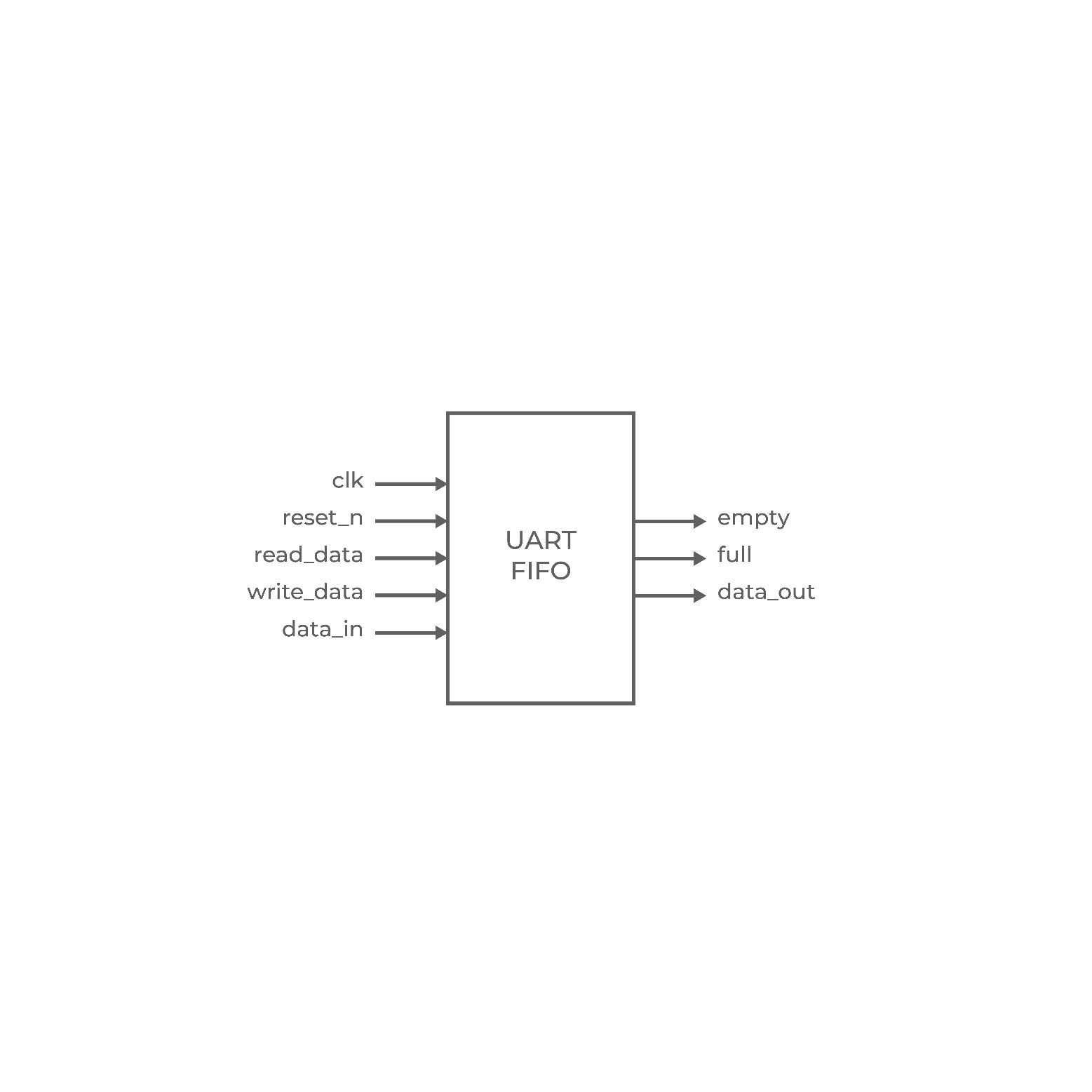
|  |  |  |  |
| --- | --- | --- | --- |
| **Signal name** | **Width** | **Input/Output** | **Description** |
| **clk** | 1 | Input | Tín hiệu xung đồng hồ |
| **reset\_n** | 1 | Input | Tín hiệu reset tích cực mức thấp |
| **clock** | 1 | Output | Clock f = BAUD\_RATE cho Transmitter |
| **sample\_clk** | 1 | Output | Clock f = 16\*BAUD\_RATE cho Receiver |

Module Clock Generator chia tần để giảm tần số 100MHz về Baud Rate chuẩn của UART

Bảng 2.3. uart\_generator\_clock parameter

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Default** | **Description** |
| **SYS\_FREQ** | 100000000 | Tần số hoạt động của System Clock |
| **BAUD\_RATE** | 9600 | Tốc độ Baud trên kênh truyền |
| **CLOCK** | SYS\_FREQ/BAUD\_RATE | Độ chia clock |
| **SAMPLE** | 16 | Lấy mẫu x16 |
| **BAUD\_DVSR** | SYS\_FREQ/(SAMPLE\*BAUD\_RATE) | Độ chia tần số lấy mẫu |

## Module FIFO (uart\_fifo)



Hình 2.7. uart\_fifo block diagram

### Interface signals

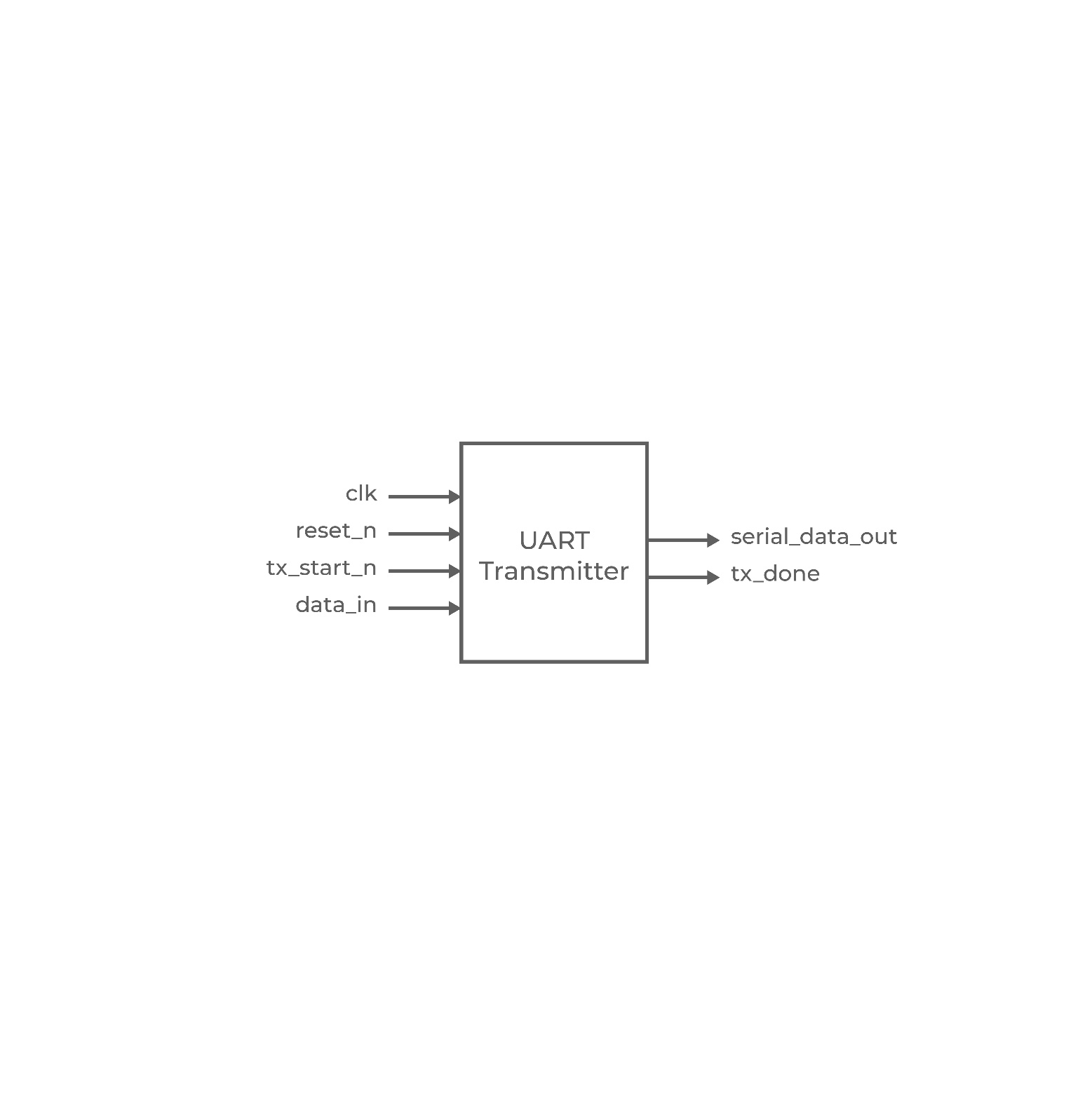
Bảng 2.4. uart\_fifo port description

|  |  |  |  |
| --- | --- | --- | --- |
| **Signal name** | **Width** | **Input/Output** | **Description** |
| **clk** | 1 | Input | Tín hiệu xung đồng hồ |
| **reset\_n** | 1 | Input | Tín hiệu reset tích cực mức thấp |
| **data\_in** | DATA\_SIZE | Input | Dữ liệu ghi vào FIFO |
| **write** | 1 | Input | Cho phép ghi vào FIFO |
| **read** | 1 | Input | Cho phép đọc từ FIFO |
| **data\_out** | DATA\_SIZE | Output | Dữ liệu đọc ra từ FIFO |
| **full** | 1 | Output | Tín hiệu báo FIFO đầy |
| **empty** | 1 | Output | Tín hiệu báo FIFO trống |

Bảng 2.5. uart\_fifo parameter

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Default** | **Description** |
| **DATA\_SIZE** | 8 | Kích thước dữ liệu |
| **SIZE\_FIFO** | 8 | Số lượng bộ đệm |
| **ADDR\_WIDTH** | $clog2(SIZE\_FIFO) | Số bit đánh địa chỉ |

## Module Transmitter (uart\_transmitter)



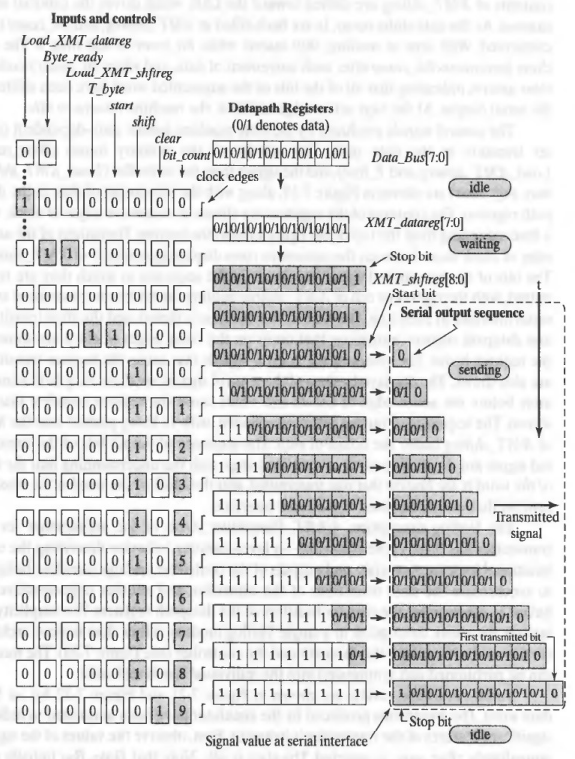
Hình 2.8. uart\_transmitter block diagram

### Interface signals

Bảng 2.6. uart\_transmitter port description

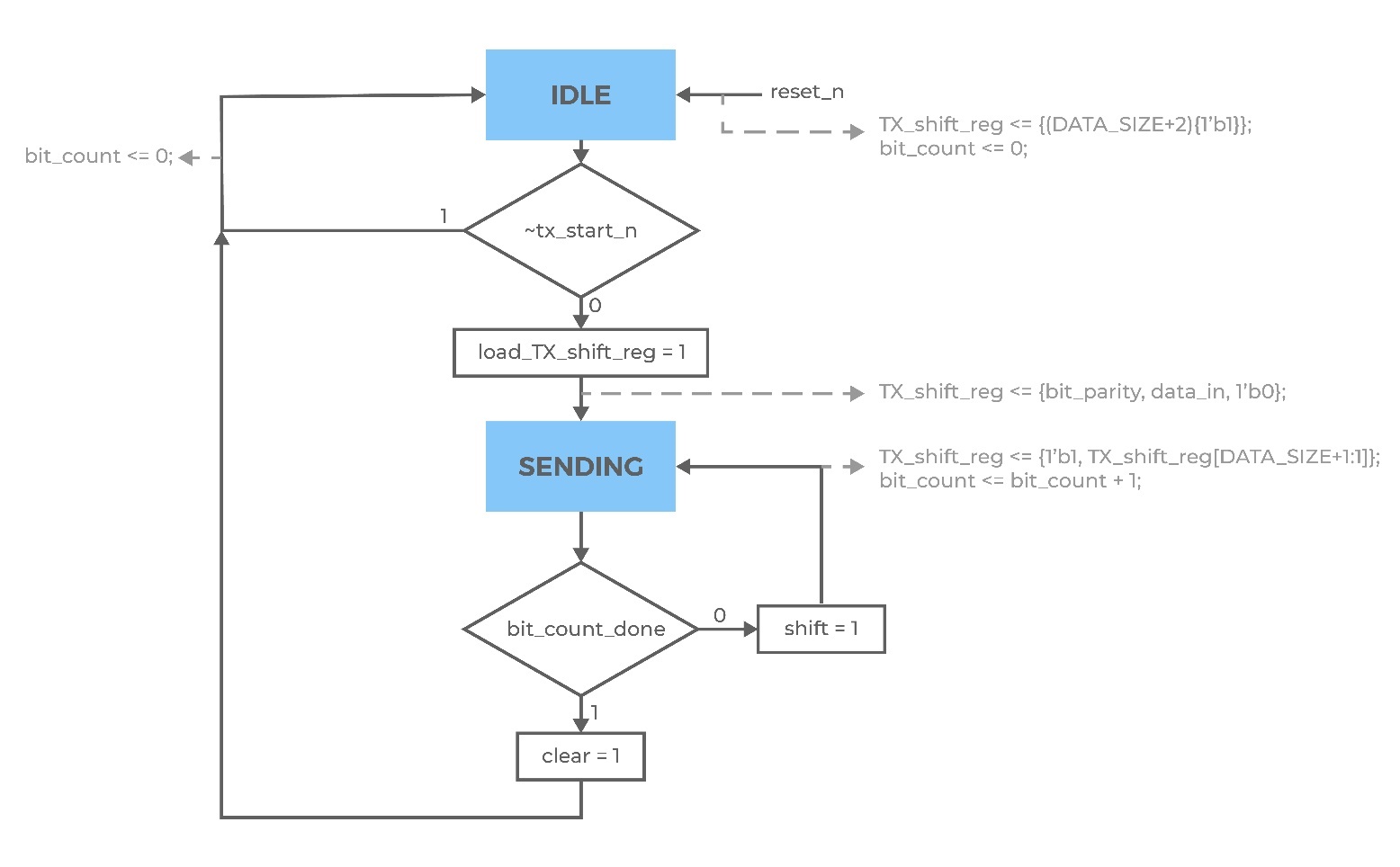
| **Signal name** | **Width** | **Input/Output** | **Description** |
| --- | --- | --- | --- |
| **clk** | 1 | Input | Tín hiệu xung đồng hồ |
| **reset\_n** | 1 | Input | Tín hiệu reset tích cực mức thấp |
| **tx\_start\_n** | 1 | Input | Tín hiệu kích hoạt transmitter |
| **data\_in** | DATA\_SIZE | Input | Dữ liệu được lấy từ TX\_FIFO |
| **serial\_data\_out** | 1 | Output | Dữ liệu ra nối tiếp |
| **tx\_done** | 1 | Output | Tín hiệu báo truyền dữ liệu xong |

### Function description



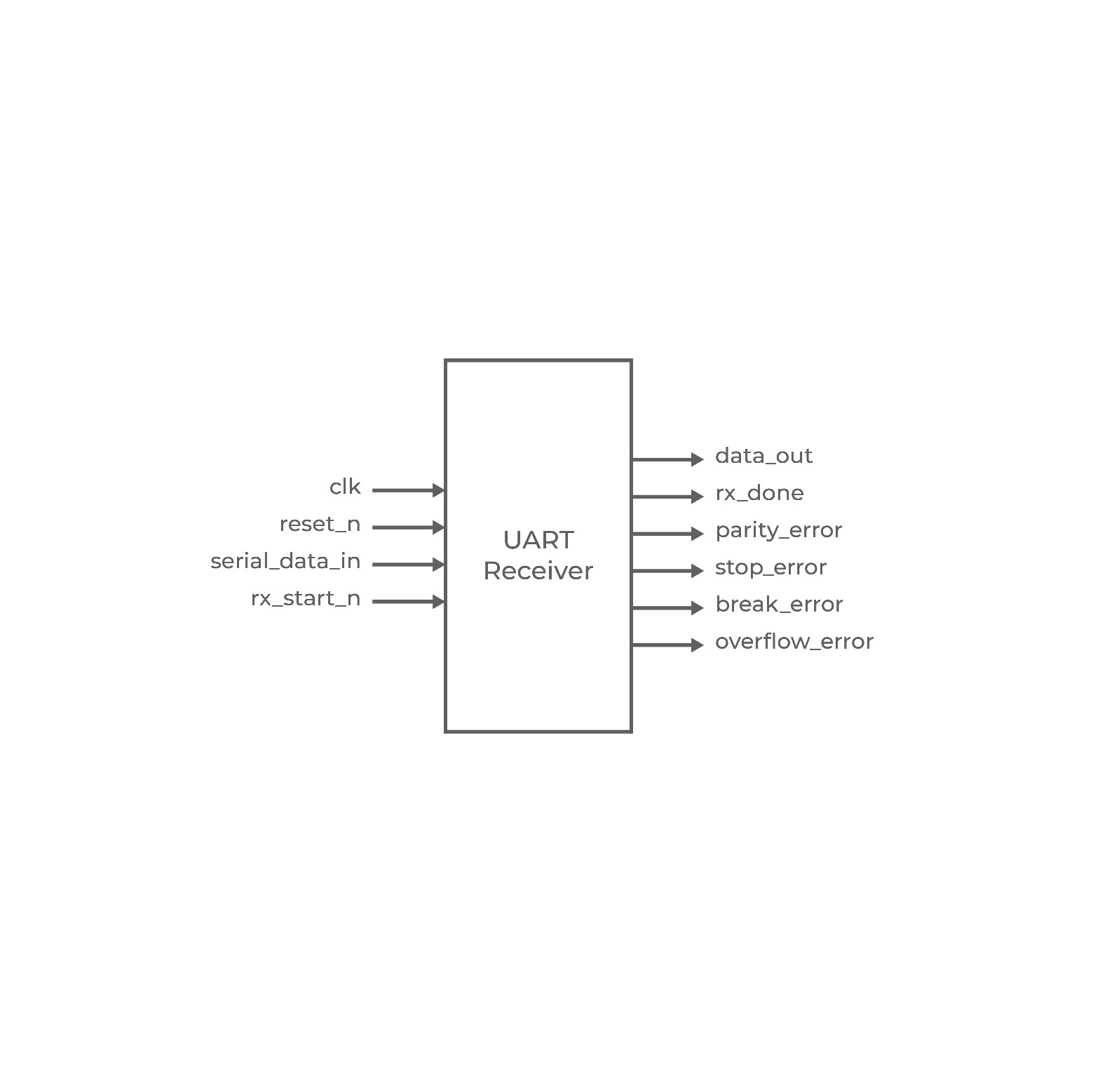
Hình 2.9. Tín hiệu điều khiển và dataflow trong UART transmitter

### Transmitter architecture



Hình 2.10. ASMD uart\_transmitter

## Module Receiver (uart\_receiver)



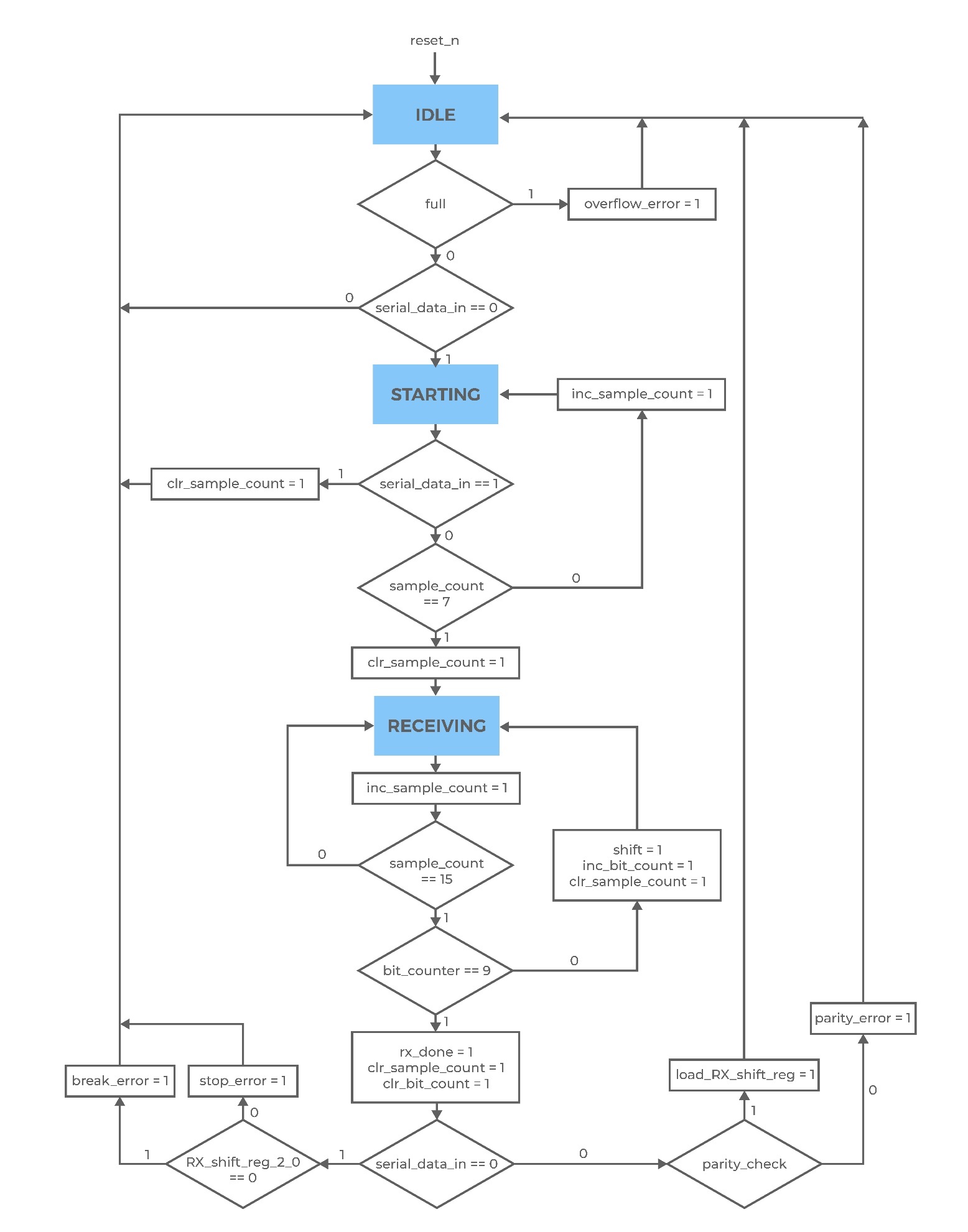
Hình 2.11. uart\_receiver block diagram

### Interface signals

Bảng 2.7. uart\_receiver port description

| **Signal name** | **Width** | **Input/Output** | **Description** |
| --- | --- | --- | --- |
| **clk** | 1 | Input | Tín hiệu xung đồng hồ |
| **reset\_n** | 1 | Input | Tín hiệu reset tích cực mức thấp |
| **serial\_data\_in** | 1 | Input | Dữ liệu vào nối tiếp |
| **rx\_start\_n** | 1 | Input | Tín hiệu cho phép nhận dữ liệu |
| **data\_out** | DATA\_SIZE | Output | Dữ liệu ghi vào RX\_FIFO |
| **rx\_done** | 1 | Output | Tín hiệu báo đã nhận đủ khung dữ liệu |
| **parity\_error** | 1 | Output | Lỗi bit chẵn/lẻ |
| **stop\_error** | 1 | Output | Lỗi bit dừng |
| **break\_error** | 1 | Output | Lỗi mất dữ liệu |
| **overflow\_error** | 1 | Output | Lỗi tràn bộ FIFO |

### Receiver architecture



Hình 2.12. ASMD uart\_receiver

# KIỂM THỬ (VERIFICATION)

*Chương này trình bày các kết quả mô phỏng cho từng khối và toàn bộ thiết kế được triển khai bằng ngôn ngữ System Verilog trên phần mềm Questa Sim.*

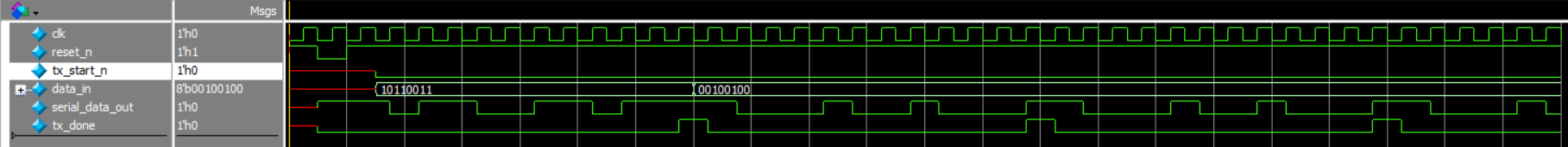
## Kế hoạch kiểm thử

### Kịch bản – Test case

* Kiểm tra quá trình reset, đảm bảo các đầu ra được thiết lập về đúng giá trị.
* Test module uart\_generator\_clock, đảm bảo clock và sample clock được gen đúng.
* Test module uart\_fifo, đảm bảo dữ liệu đọc ghi đúng, tín hiệu báo full và empty báo đúng.
* Test module uart\_transmitter, đảm bảo chạy đúng theo sơ đồ ASMD.
* Test module uart\_receiver, đảm bảo chạy đúng theo sơ đồ ASMD.
* Kiểm tra quá trình truyền dữ liệu từ TX và quá trình nhận dữ liệu của RX.
* Kiểm tra quá trình đọc ghi liên tục với FIFO.

## Kết quả và đánh giá

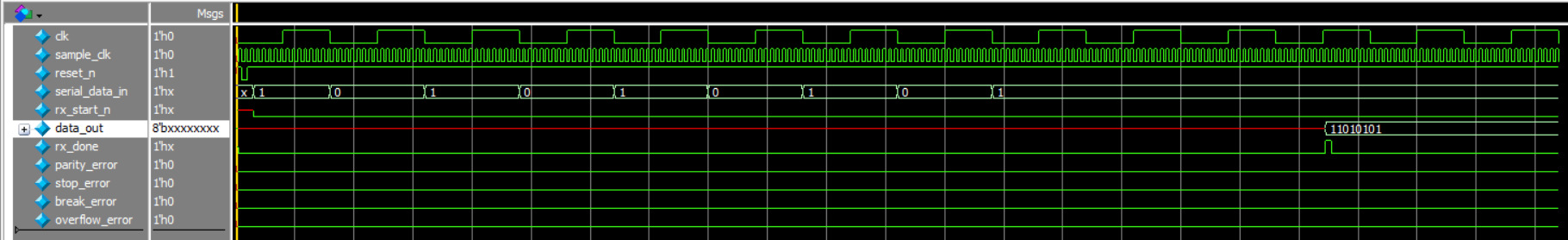
### Module uart \_transmitter



Hình 3.1 Mô phỏng timing diagram khối uart\_transmitter

Hình 3.1 cho thấy serial\_data\_out đã có đưa ra đầy đủ bit start, stop và data giống với data\_in đưa vào.

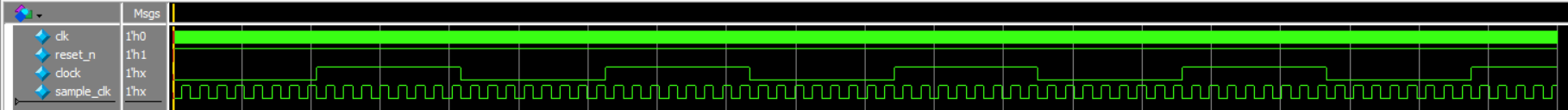
### Module uart\_receiver



Hình 3.2 Mô phỏng timing diagram khối uart\_receiver

Hình 3.2 cho thấy data\_out trả về kết quả đúng với dãy bit vào từ serial\_data\_in.

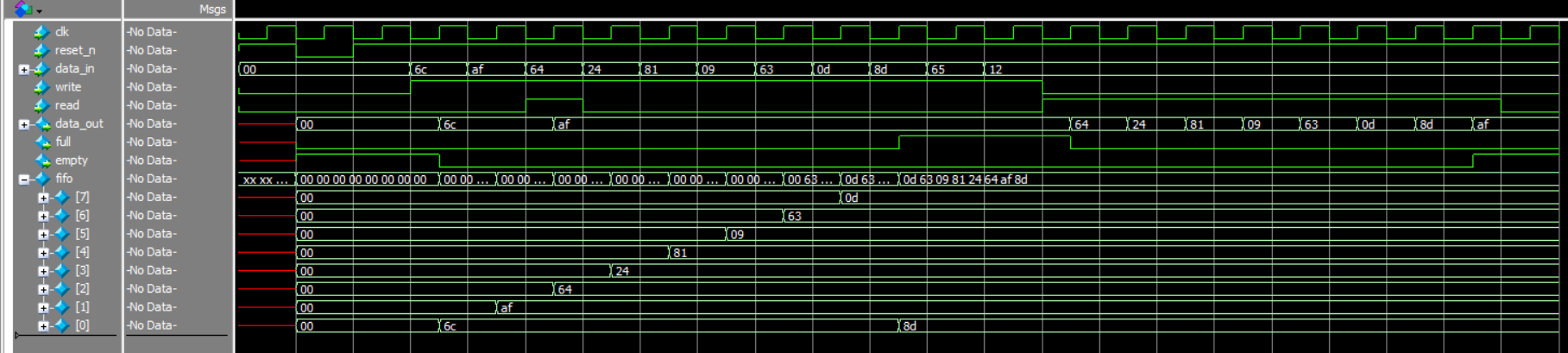
### Module uart\_generator\_clock



Hình 3.3 Mô phỏng timing diagram khối uart\_generator\_clock

Hình 3.3 cho thấy kết quả 2 clock được tạo ra từ clock 100MHz ban đầu.

### Module uart\_fifo



Hình 3.4 Mô phỏng timing diagram khối uart\_fifo

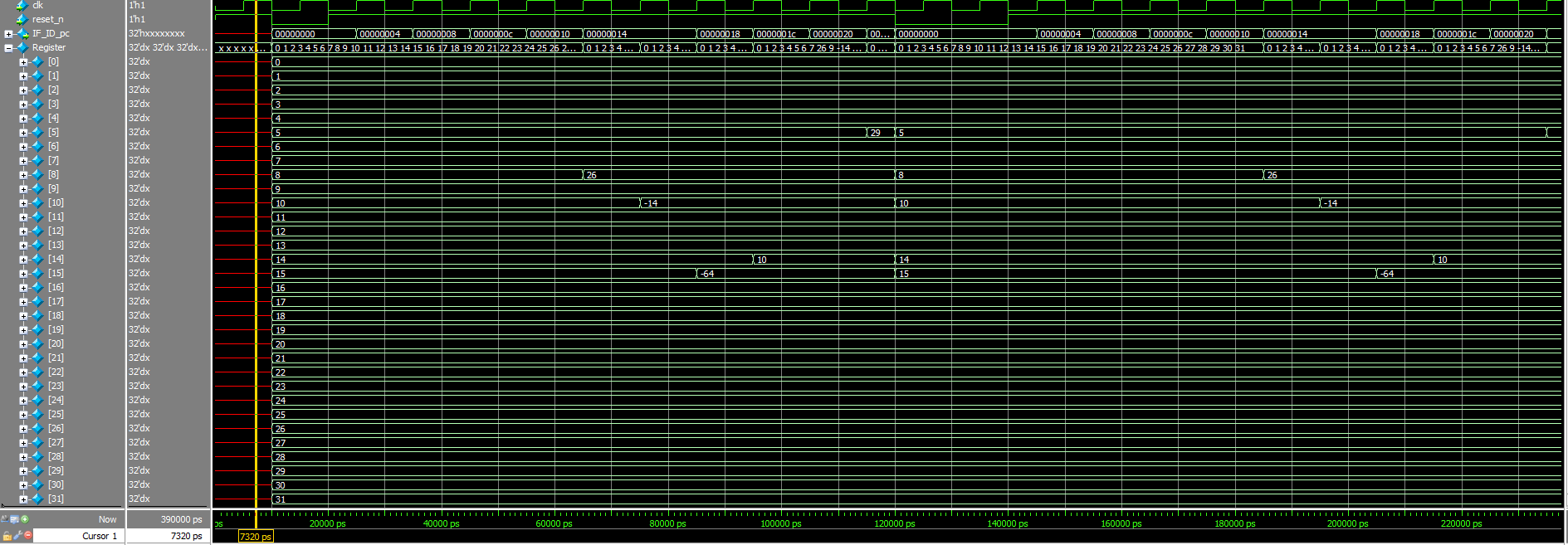
Hình 3.4 cho thấy việc đọc ghi của FIFO đã đúng với yêu cầu. Các tín hiệu báo full, empty đã lên đúng chu kỳ.

### Module register\_write

Hình 3.5 Mô phỏng timing diagram khối register\_write

Hình 3.5 cho thấy kết quả đầu ra của khối **Register Write** hoạt động đúng với logic của thiết kế.

### Register



Hình 3.6 Mô phỏng 32 Registers

Hình 3.6

Hình 3.10 Mô phỏng Data\_Memory

Hình 3.10 mô tả sự thay đổi của dữ liệu bên trong **Data Memory**, dữ liệu thay đổi đúng như kịch bản tính toán của các câu lệnh được nạp sẵn trong **Instruction Memory**.

# KẾT LUẬN

Báo cáo này đã trình bày và triển khai kiến trúc của một RISCV32I processor áp dụng kĩ thuật pipeline nhằm tăng tốc độ xử lí, bên cạnh đó, thiết kế đã xữ lí được toàn bộ các hazard có thể xảy ra trong quá trình xử lí thực hiện lệnh của CPU bao gồm data hazard, mem hazard và control hazard. Thiết kế được tiến hành triển khai bằng ngôn ngữ mô tả phần cứng SystemVerilog và mô phỏng kiểm thử trên phần mềm ModelSim. Cho ra kết quả hoạt động đúng với yêu cầu vào ra. Kiến trúc có thể thực hiện 4 kiểu lệnh assembly trong tập lệnh của RISCV gồm: R-type, I-type, S-type, B-type, tuy nhiên chưa thể thực hiện được tất cả các lệnh có trong tập lệnh của RISCV (xấp xỉ 40 lệnh). Trong tương lai nhóm sẽ tiến hành hoàn thiện và triển khai kiến trúc của một RISCV32I processor hoàn chỉnh với đầy đủ các chức năng, thực hiện được đầy đủ các lệnh trong kiến trúc tập lệnh của RISCV một cách tối ưu nhất.

TÀI LIỆU THAM KHẢO

1. <https://en.wikipedia.org/wiki/RISC-V> Ngày truy cập cuối cùng: 16/5/2021
2. Slide: RISC – V 2021. PGS TS. Nguyễn Đức Minh
3. Computer Organization and Design Risc – V The Hardware Software Interface by David A. Patterson and John L. Hennessy
4. Link project: https://github.com/chien172431/riscv32.git