**Universal Asynchronous Receiver/Transmitter**

**(UART)**

Microarchitecture Specification

**Table of Contents**

[**1. Features:**](#_gjdgxs) **4**

[**2.Block Diagrams**](#_30j0zll) **5**

[2.1 Level-0 Block Diagram](#_1fob9te) 5

[2.2 Level-1 Block Diagram](#_papyku9ujonu) 6

[**3.Signal Description**](#_2et92p0) **7**

[**4. Functional Description**](#_nahpha9lboss) **8**

[**4.1 Functional Overview:**](#_ge8mr1xrozjh) **8**

[4.1 UART Initialization using APB Interface](#_l6go403hsd96) 8

[4.2 Operations](#_hlqa57h6wtsb) 9

[4.2.1 Transmission](#_2s9i0cpsoiv) 9

[4.2.2 Reception](#_jee3k9bezca6) 9

[4.3 Interrupt Generation and Servicing of Interrupt:](#_nlc17u3ndsxf) 10

[**5. Block Description**](#_82m8dgj88ks3) **11**

[5.1 APB UART Interface](#_n0h1lf9wl63n) 11

[5.2 Clock generator](#_2gz0ls2mbo1m) 12

[5.3 Transmitter block](#_hyeznmu6pei) 13

[5.3.1 Transmit logic :](#_3lucv28ut9n1) 13

[5.3.2 State Diagram :](#_xa80j4rxb3jn) 14

[5.3.3 Parity Generator:](#_u8njmo8pn6fx) 15

[5.3.4 Transmit Shift Register:](#_p70k0g9txzo4) 15

[5.3.5 Waveform:](#_cvugktgj06v9) 16

[5.4 Receiver block](#_sapmuzoh7bjh) 17

[5.4.1 Block Diagram](#_exao5nh82d87) 17

[5.4.2 Description](#_kjkogz7zz1nz) 17

[5.4.3 Timing Diagram](#_7d3450d5cvdq) 17

[5.4.3 UART Receive FSM](#_tev1iyaj6tnm) 18

[5.4.3.1 Block Diagram](#_k5udqm7r1xyy) 18

[5.4.3.2 State Diagram](#_7q20qqsxcxnm) 19

[5.4.3.3 Description](#_b1m5fk25cw34) 19

[5.4.4 Receiver Shift Block](#_o44pqknc2obu) 20

[5.4.4.1 Block Diagram](#_3m2i2by2ur3f) 20

[5.4.4.2 Description](#_rdxo6414ithw) 20

[5.4.5 Parity Checker](#_9rd1tcaqxkhl) 21

[5.4.5.1 Block Diagram](#_johxoob556h0) 21

[5.4.5.2 Description](#_lgq50v34zhg9) 21

[5.5 Buffers](#_a7ei0rqpbf0y) 22

[5.5.1 Block Diagram of Buffers](#_vfkorcx4e6qs) 22

[5.5.2 Block Description of Buffers](#_k9pioki0b0ly) 22

[5.6 Interrupt generation block](#_du7jtbgg9ek0) 23

[**6.Registers**](#_s7fmqpne64y) **24**

[6.1 Register List](#_39mki8z3uuta) 24

[6.2.1 Receive Buffer Register (RBR):](#_dkzl4xi504aw) 25

[6.2.2 Transmit Holding Register (THR):](#_ne3dw72pgqpt) 25

[6.2.3 Interrupt Enable Register (IER):](#_gtn1jb5m9fud) 25

[6.2.4 Interrupt Identification Register (IIR):](#_tq7r2ymm8hq6) 26

[6.2.5 FIFO Control Register (FCR):](#_biwae9viu5c4) 27

[6.2.6 Line Control Register (LCR):](#_uktu9rqxmhjo) 28

[6.2.7 Line Status Register (LSR):](#_5egqzlxggm8c) 29

[6.2.8 Divisor LSB Latch (DLL):](#_jz52h9ec4mcq) 32

[6.2.9 Divisor MSB Latch (DLH):](#_azme95tcqux1) 32

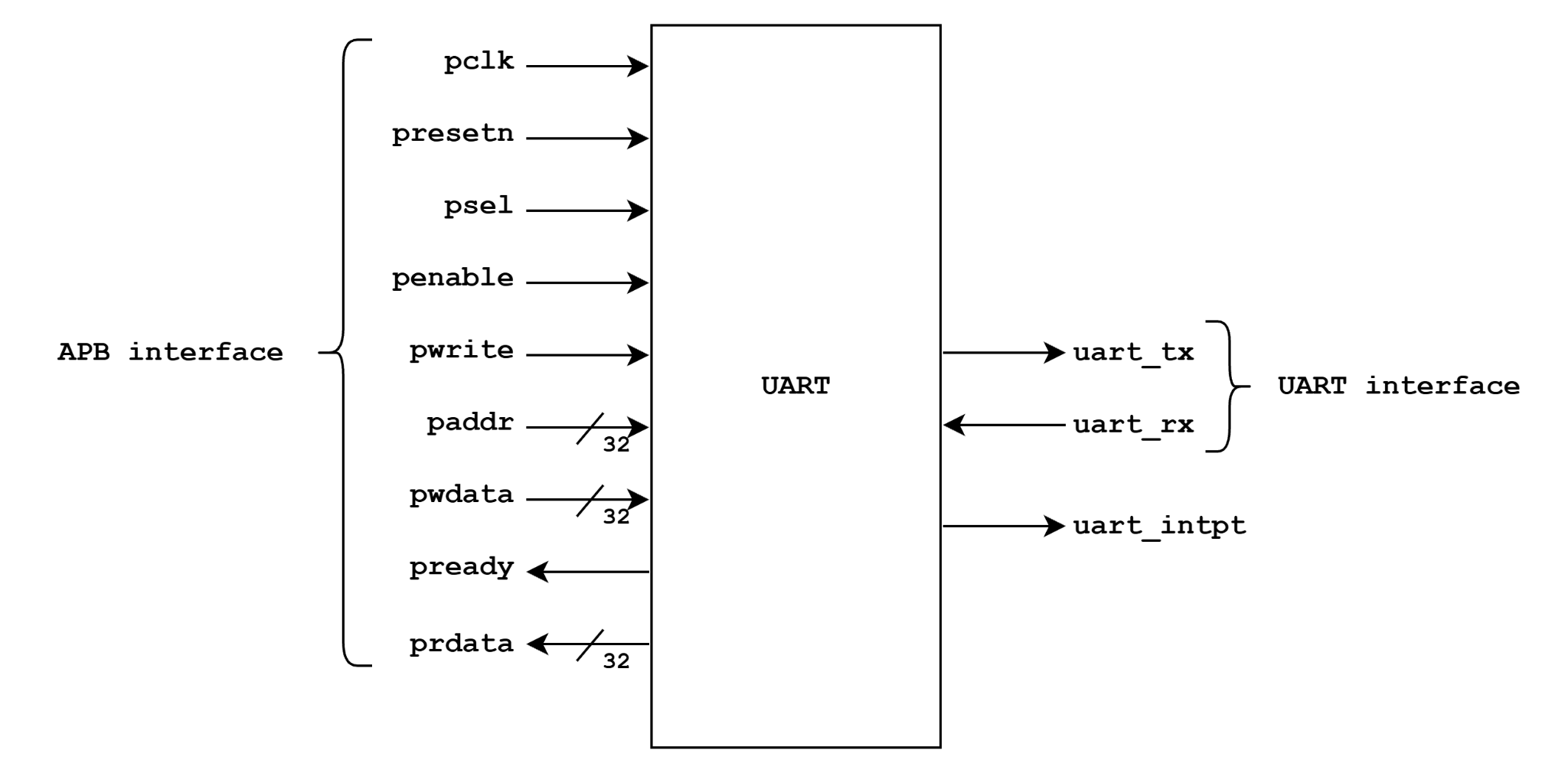
[6.2.10 Power and emulation Management Register (PWREMU\_MGMT):](#_mcjb7g1mv2jj) 33

# **Features:**

* Full duplex asynchronous serial data transfer.
* Variable length of transfer word from 5 to 8 bits
* LSB first data transfer starting with one start bit and ending with one or two stop bits.
* Even, odd, stick and no parity generation and detection.
* Sample Occurs at a baud rate 16 times faster than the transmit baud rate.
* UART supports both fifo and non-fifo mode operations where in fifo mode the receiver and transmitter FIFO stores up to 16 bytes of data.
* Detection of parity, break, frame or overrun error and generation of corresponding interrupts if enabled.
* Programmable Interrupt Service routine.
* Supports Loopback feature for diagnostic purpose in which the UART transmitter is internally connected to the UART receiver for verifying the transmit and receive data paths without connecting to another UART.

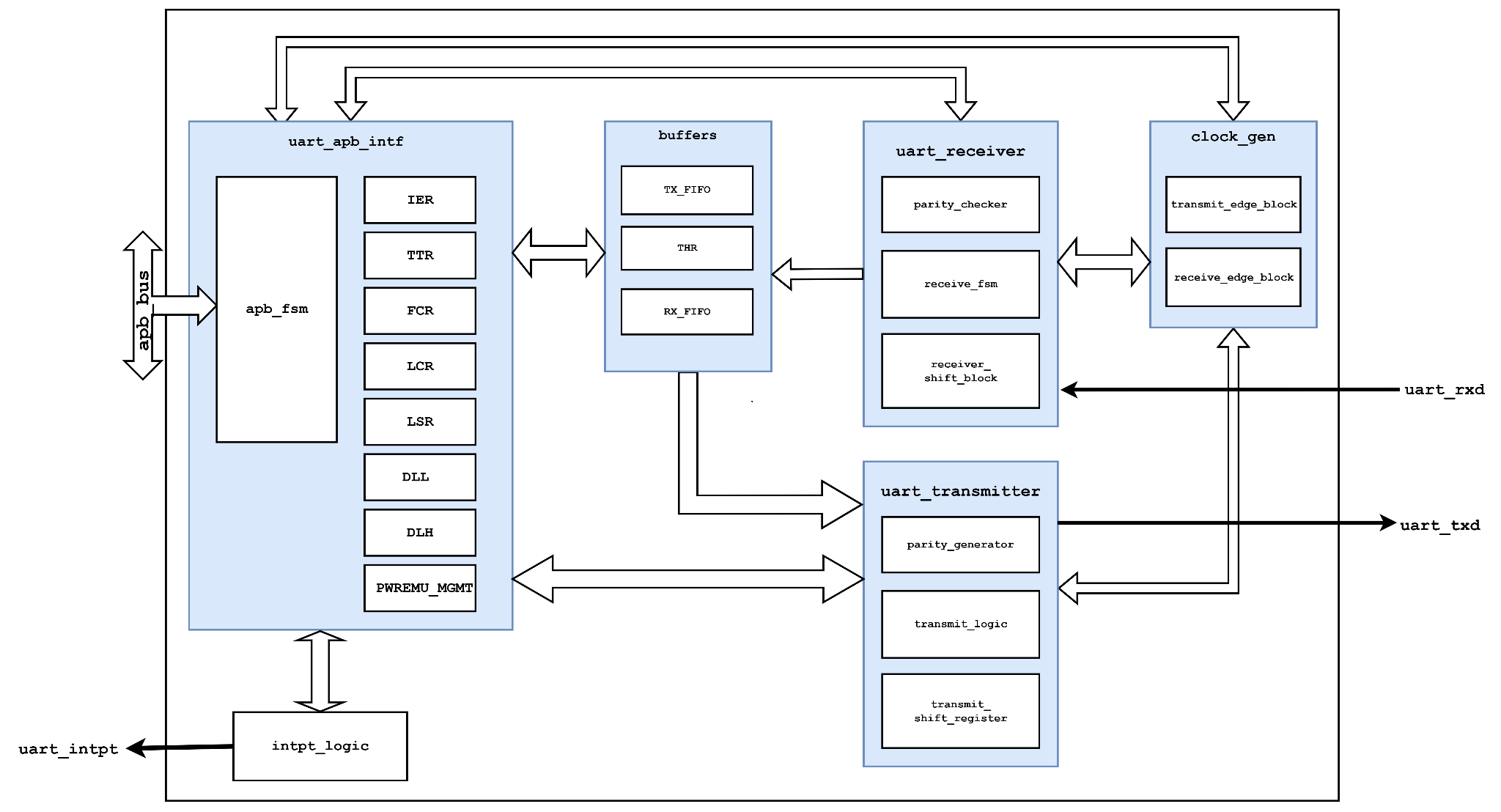
# **2.Block Diagrams**

## **2.1 Level-0 Block Diagram**



**Fig-1: Level-0 block diagram of UART**

## **2.2 Level-1 Block Diagram**



**Fig-2: Level-1 block diagram of UART**

## 

# **3.Signal Description**

| **Name** | **Width** | **Direction** | **Description** |
| --- | --- | --- | --- |
| pclk | 1 | Input | APB system clock. 100 MHz |
| presetn | 1 | Input | APB system reset. ACTIVE\_LOW asynchronous |
| psel | 1 | Input | APB chip select |
| penable | 1 | Input | APB enable |
| paddr | 32 | Input | APB address bus |
| pwdata | 32 | Input | APB write data bus |
| pwrite | 1 | Input | APB write/read. 0 : read, 1 : write |
| prdata | 32 | Output | APB read data bus |
| pready | 1 | Output | Peripheral ready to complete transfer |
| uart\_txd | 1 | Input | Serial data output |
| uart\_rxd | 1 | Input | Serial data input |
| uart\_intpt | 1 | Output | UART interrupt |

Table-1: Signal Description

# **4. Functional Description**

# **4.1 Functional Overview:**

The UART performs asynchronous full duplex serial data transmission with another UART sharing the same baud rate configuration. The baud rate is set by configuring the clock divisor registers. The UART possesses independent clock generation for transmission and reception of data, in which the receive clock is 16 times faster than the transmit clock for accurate sampling of data.The UART samples data at 7th,8th and 9th cycle of sample clock to capture the most valid and stable data. The frame of data transmission and reception is set by configuring the Line Control Register which controls the size of data bits, number of stop bits and enabling the generation and detection of parity bits. UART has both FIFO and Non-FIFO mode of operation where in Non-FIFO mode the UART can perform a transmit or a receive of a single frame of data. In FIFO mode the UART can perform a maximum transmission of 16 data frames and a maximum reception of 16 data frames. In fifo mode the received data is stored along with parity, frame and break error bit. Interrupt is generated when the interrupt enable bits of the corresponding events are set. Interrupts are serviced by CPU reading the Line Status Register (LSR). The transmitter and receiver is enabled by setting the UTRST and URRST bit of Power and Emulation management register. Clearing these bits will disable the transmitter and receiver respectively.The transmitter and receiver will also go to their reset state. The UART also comes with a Loopback feature through which the UART can be placed in the Diagnostic mode, which internally connects UART output back to the UART input. The transmitter- receiver data paths and their interrupts can be verified in this mode.

## **4.1 UART Initialization using APB Interface**

After presentn is de-asserted, the UART is configured to perform the desired serial data transmission and reception. It is important to mention that for a successful data communication between two it is necessary to be operated at the same configuration.For initializing the UART necessary configurations are provided by writing to the necessary registers through an APB\_Register Interface.

* At First the Desired baud rate is set by writing the appropriate clock divisor values to the divisor latch registers (DLL and DHH).
* If the FIFOs will be used, select the desired trigger level and enable the FIFOs by writing the appropriate values to the FIFO control register (FCR). The FIFOEN bit in FCR must be set first, before the other bits in FCR are configured.
* Choose the desired data communication setting by writing the appropriate values to the Line Control Register(LCR).
* Lastly enable the UART transmitter and receiver by setting the UTRST and URRST bits in the Power and Emulation management register (PWREMU\_MGMT).

## **4.2 Operations**

### **4.2.1 Transmission**

The UART transmitter section consists of Transmitter Hold Register (THR), Transmitter Shift Register (TSR) and Transmit logic block.When UART is in FIFO mode, THR is a 16-byte depth FIFO. The Transmit Logic block is a function of the UART Line Control Register(LCR). Based on the settings chosen in LCR, the UART transmitter sends the following to the receiving device:

1. 1 START bit
2. 5, 6, 7, or 8 data bits
3. 1 PARITY bit (optional)
4. 1 or 2 STOP bits.

The THR receives data from the APB-Register interface, and when TSR is ready the USART moves the data from THR to TSR.The UART shifts out the data bits (LSB first) in the TSR and transmits the data through the UART\_TXD pin. In the non-FIFO mode, if THR is empty and the THR empty interrupt is enabled in the interrupt enable register (IER), an interrupt is generated. This interrupt is cleared when a character is loaded into THR. In the FIFO mode, the interrupt is generated when the transmitter FIFO is empty, and it is cleared when at least one byte is loaded into the FIFO.

### **4.2.2 Reception**

The UART receiver section consists of a Receiver Shift Register (RSR), receiver buffer register and Receiver Logic block. When the UART is in the FIFO mode, RBR is a 16-byte FIFO. Timing is supplied by the receiver clock. Receiver logic block is a function of the UART Line Control Register (LCR). Based on the settings chosen in LCR, the UART receiver accepts the following from the transmitting device:

1. 1 STA RT bit
2. 5, 6, 7, or 8 data bits
3. 1 PARITY bit (optional)
4. 1 STOP bit (any other STOP bits transferred with the above data are not detected)

RSR receives the data bits through the UART\_RXD pin. The RSR moves the data bits from to the RBR register or the receiver FIFO if the FIFO mode is enabled. In FIFO mode the UART also stores bits of error status information next to each received character, to record a parity error, framing error, or break. In the non-FIFO mode, when a character is placed in RBR and the receiver data-ready interrupt is enabled in the interrupt enable register (IER), an interrupt is generated. This interrupt is cleared when the character is read from RBR. In the FIFO mode, the interrupt is generated when the FIFO is filled to the trigger level selected in the FIFO control register (FCR), and it is cleared when the FIFO contents drop below the trigger level.

## **4.3 Interrupt Generation and Servicing of Interrupt:**

When the receiver- transmitter FIFO is enabled in the FIFO control register (FCR) and the receiver-transmitter interrupts are enabled in the interrupt enable register (IER), several types of interrupts may be generated during both transmission and reception. The following are described below:

* The receiver data-ready interrupt is issued to the CPU when the FIFO has reached the trigger level that is programmed in FCR. It is cleared when the CPU reads enough characters from the FIFO such that the FIFO drops below its programmed trigger level.
* The receiver line status interrupt is generated in response to an overrun error, a parity error, a framing error, or a break.
* The data-ready (DR) bit in the line status register (LSR) indicates the presence or absence of characters in the receiver FIFO. The DR bit is set when a character is transferred from the receiver shift register (RSR) to the empty receiver FIFO. The DR bit remains set until the FIFO is empty again.
* The transmitter holding register empty interrupt occurs when the transmitter FIFO is empty. It is cleared when the transmitter hold register (THR) is loaded.

All the interrupts request generation signals are ORed together to generate a single interrupt signal which is received by the CPU. The CPU services a specific interrupt request based on priority by reading the bits of the Line Status Register (LSR).

# **5. Block Description**

## **5.1 APB UART Interface**



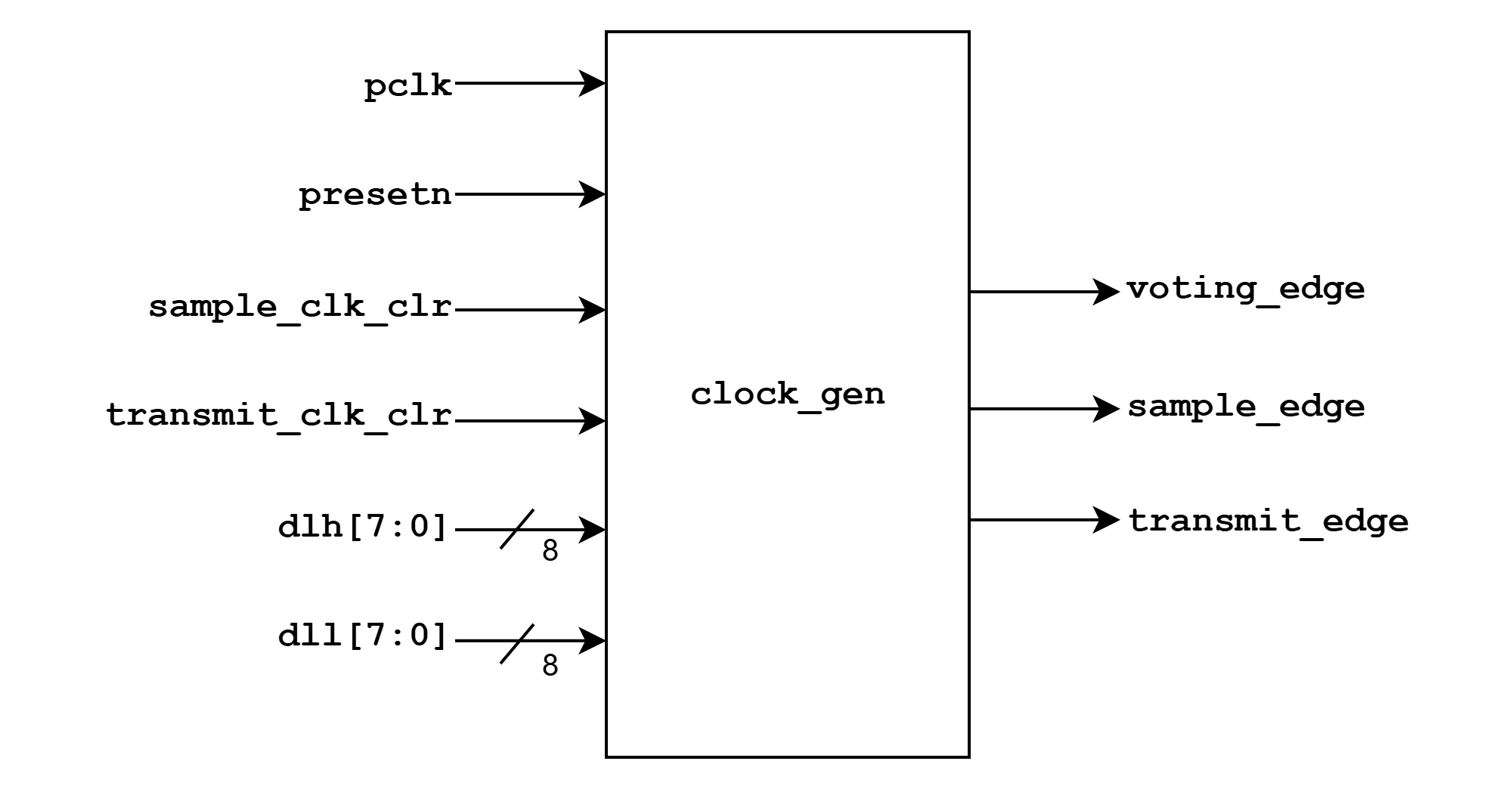
## 

## 

## 

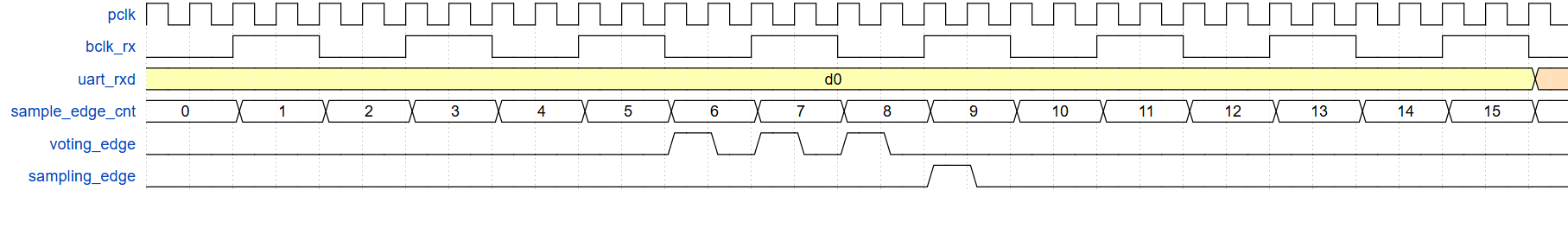
## **5.2 Clock generator**

Clock generator block generates transmit, voting and receive signal pulses depending on the value of DLL and DLH (baud rate divisor values) registers. . As UART is asynchronous, the transmit and receive operation can occur independently. As a result, they both need independent clock dividers.

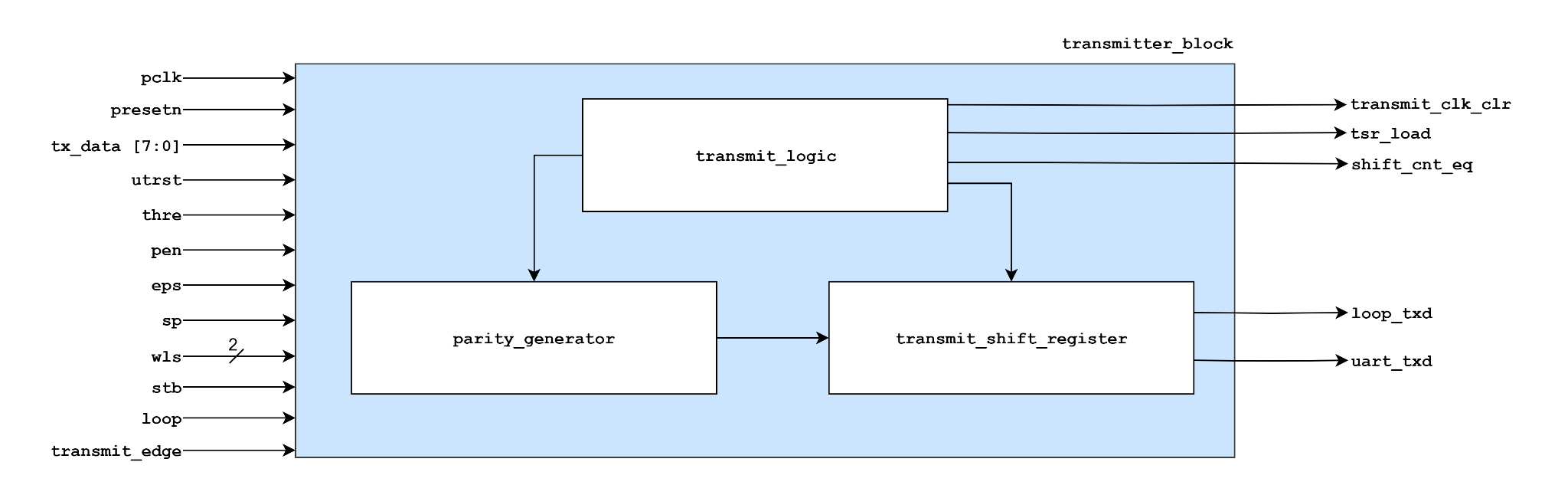


## 

The Sampling edge generation is as follows:



## **5.3 Transmitter block**



The UART transmitter block is responsible for transmission of parallel data loaded from THR register or Transmit\_fifio into serial data. This transmitter block is responsible for the generation of parity, and generation of necessary transmit logic as a function of control register bits coming from the APB register interface through which data bits of a particular frame get transmitted.

The Transmitter block consists of three sub-blocks:

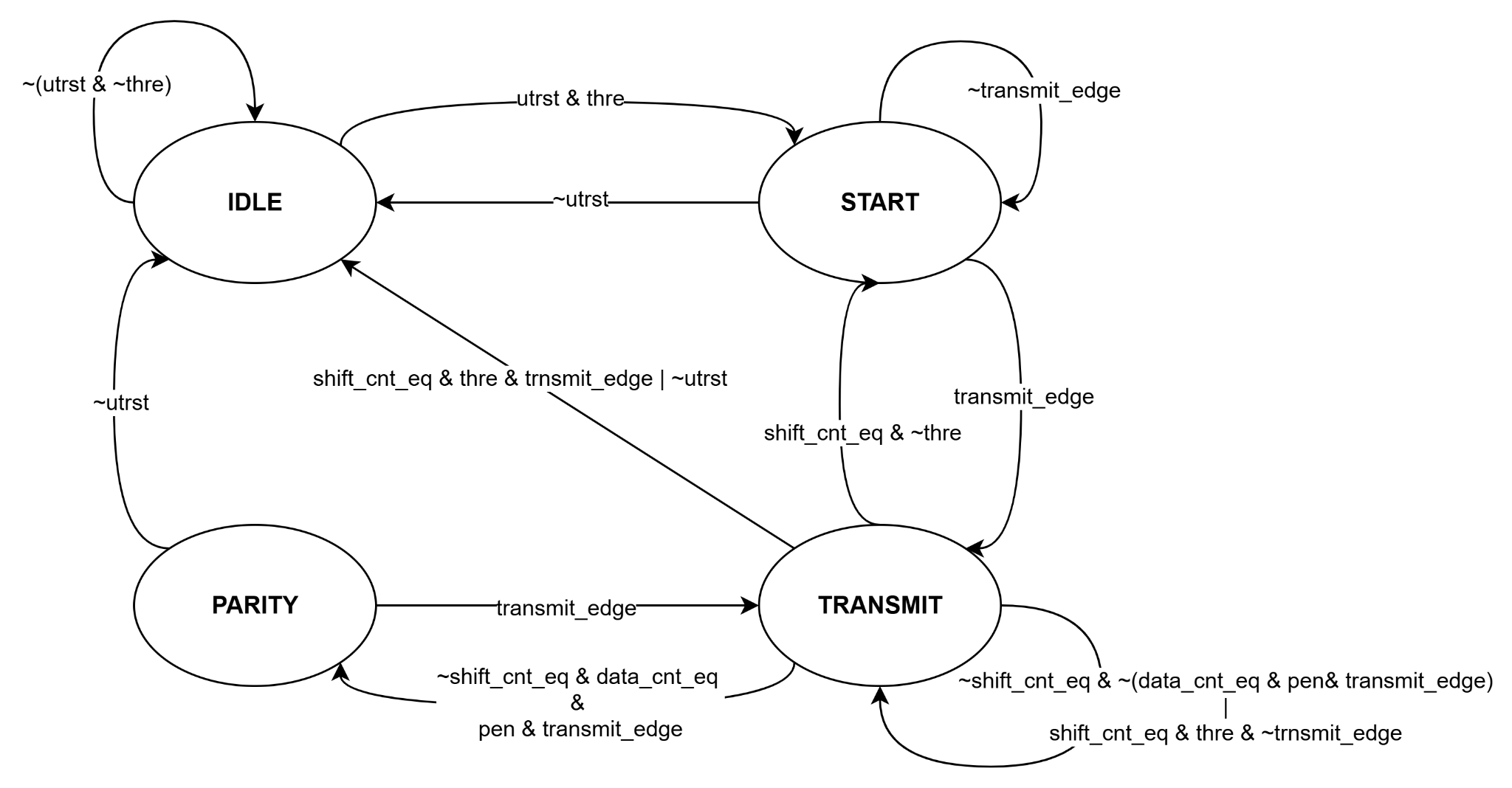
1. Transmit Logic
2. Parity Generator
3. Transmit Shift Register.

### **5.3.1 Transmit logic :**

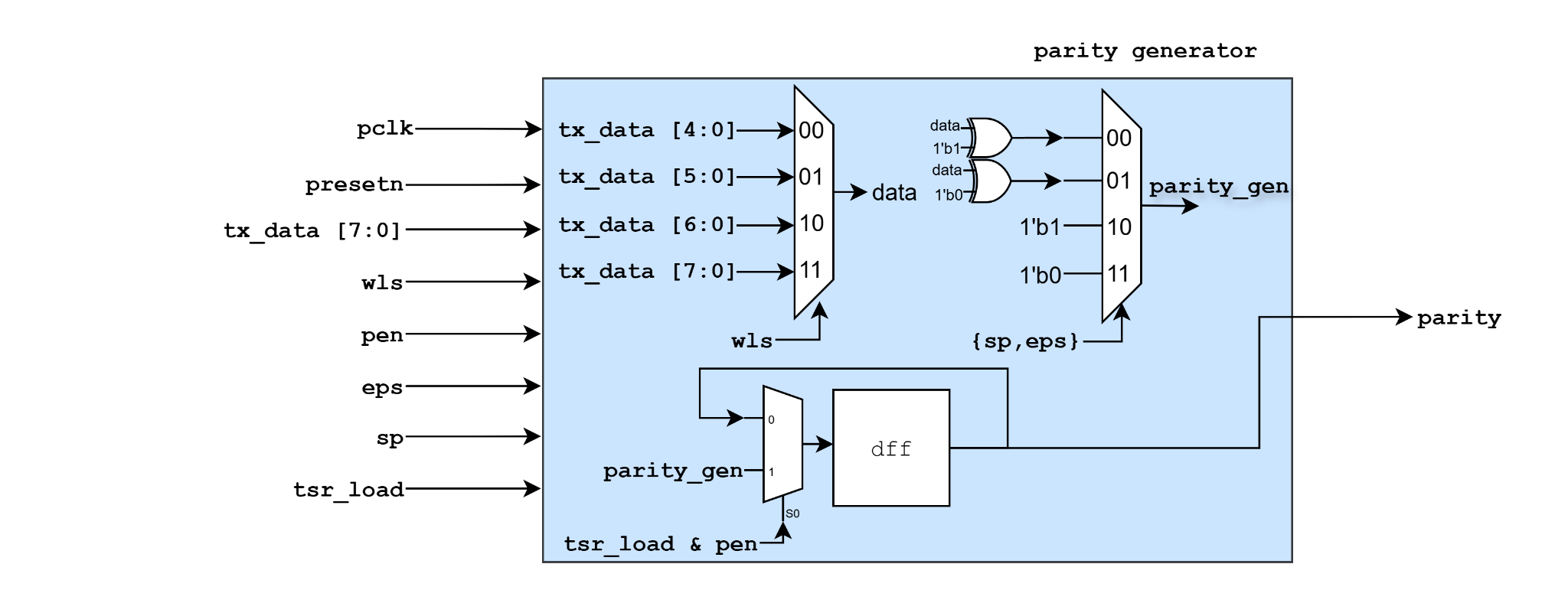
The transmit logic block consists of transmit FSM and transmit count logic. The Transmit FSM controls the transmit operation of the UART based on the signal obtained from the tx\_count\_logic block. The tx\_count\_logic block ensures the total number of bits transmitted from a start bit to stop bit of a data frame and that particular transmission gets completed after that.

The transmit\_fsm consists of 4-states: IDLE , START, TRANSMIT and PARITY state. IN START state the data from **THR** register or Transmit Fifo gets loaded into the Transmit Shift Register depending on whether transmit data is written or not. The parallely loaded data in the Transmit Shift Register shifts data serially in the TRANSMIT state. The parity bit is transmitted in the PARITY state if parity is enabled. After the Parity bit is transmitted , the transmit FSM goes back to Transmit FSM to complete the transmission of remaining stop bits.This completes the transmission of a particular data frame. After a data transmission is complete. It checks if **THRE** bit of line status register is set or clear. If **THRE** bit is set it means there is no new data to be transmitted and FSM goes from TRANSMIT state to IDLE state and if new data is present to be transmitted FSM goes from TRANSMIT state to START state for new data to get transmitted.

### **5.3.2 State Diagram :**



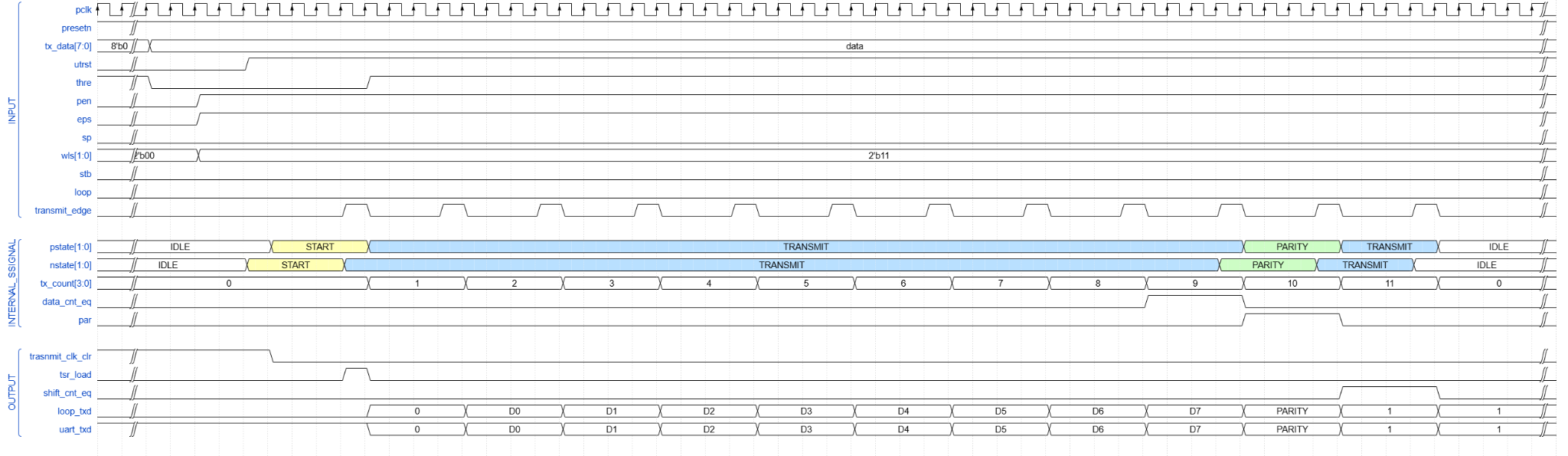
### **5.3.3 Parity Generator:**

This block is responsible for the generation of the parity bit that is being transmitted after transmission of the start bit and configured amount of data bit. The ***wls[1:0]***, ***sp***, ***eps*** signal coming from the apb-register interface is responsible for configuring the type of parity bit to be generated depending on the length of the transmitted data configured through the **wls[1:0]** bit. The calculated parity to be transmitted is stored in a single bit register when the tsr\_loadsignal sends a pulse and the **pen** bit is set.

### **5.3.4 Transmit Shift Register:**

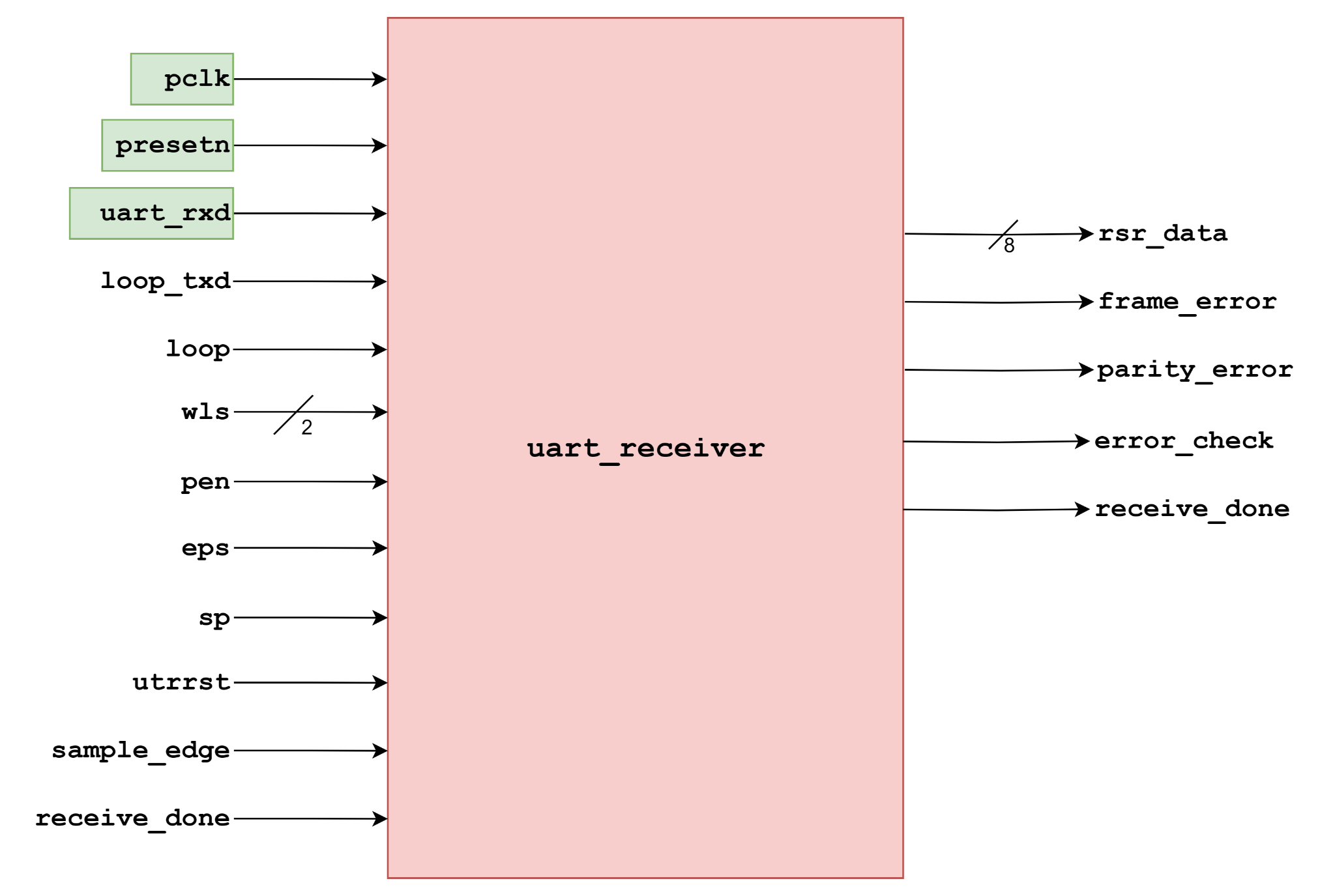
In the Transmit Shift Register block when **tsr\_load** signal generates a pulse**,** data is parallely loaded from the transmit fifo or THR register depending on whether fifo is enabled or disabled and serially shifts data bits, starting with a start bit(low and ending with parity bit and 1 or two stop bits. The rate at which the bit gets shifted out depends on the ***shift\_en*** signal.The ***shift\_en*** signal coming from the transmit FSM generates a pulse at a baud rate configured through the DIVISOR register in the apb-register interface. The **loop** signal coming from the **LCR** register bypasses the transmitted bit from UART\_txd to loop\_txd. The UART\_txd signal is always asserted to HIGH.

### **5.3.5 Waveform:**



## **5.4 Receiver block**

### **5.4.1 Block Diagram**

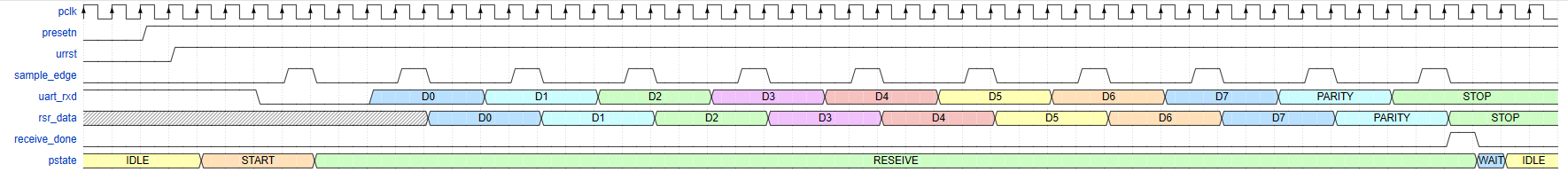


*Figure : Block diagram of UART receiver*

### **5.4.2 Description**

This block contains UART receiver FSM, receiver shift register block, and parity checker block.

### **5.4.3 Timing Diagram**



*Figure : Timing diagram of UART receiver*

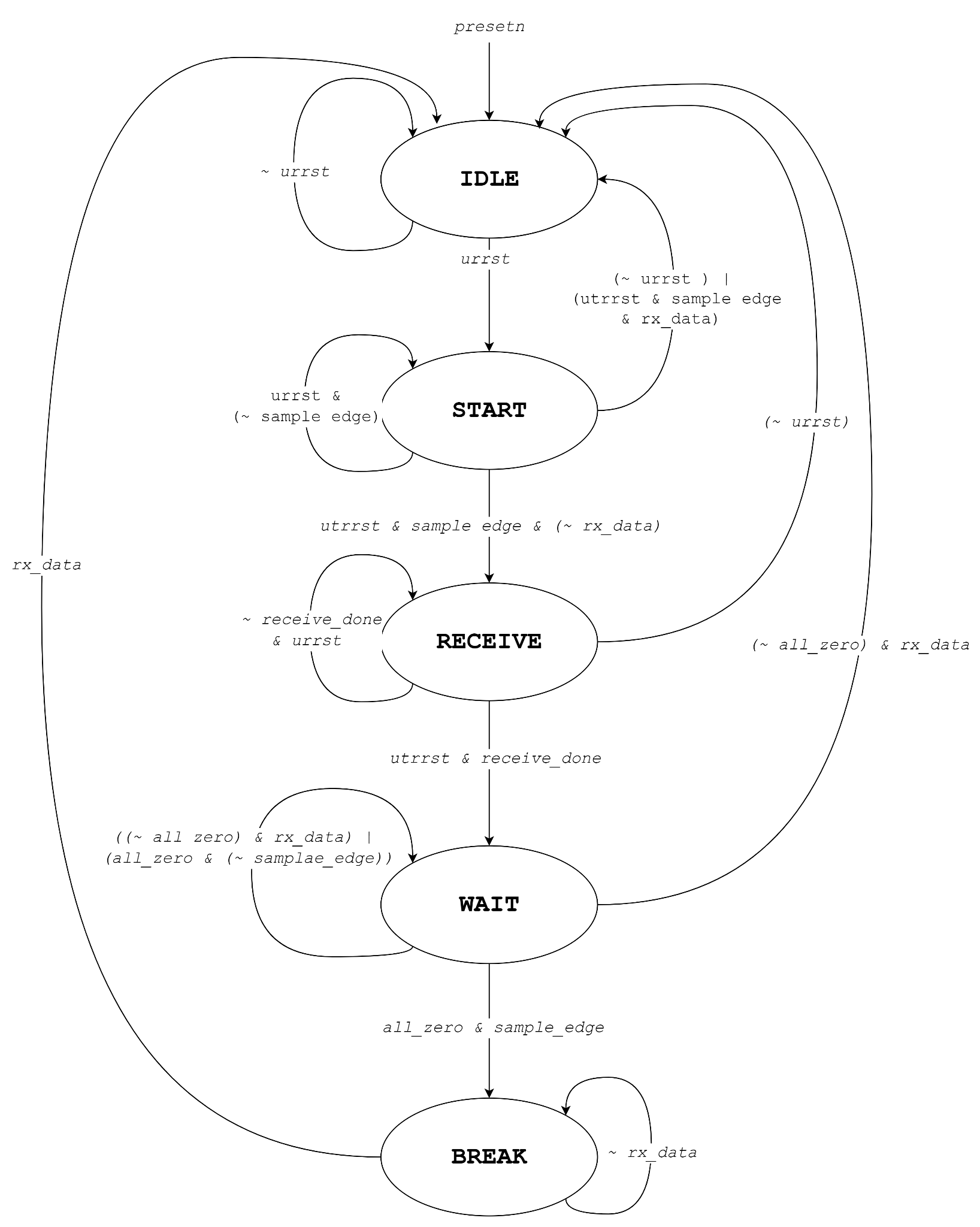
### **5.4.3 UART Receive FSM**

#### **5.4.3.1 Block Diagram**



*Figure : Block diagram of UART receive FSM*

#### **5.4.3.2 State Diagram**



*Figure : State diagram of UART receive FSM*

#### **5.4.3.3 Description**

This block functions as a controller using a Finite State Machine (FSM) with five distinct states: **IDLE, START, RECEIVE, WAIT**, and **BREAK**. The FSM begins in the **IDLE** state and transitions to START when utrrst is high and uart\_rxd is low. In the **START** state, it waits for a sample edge to validate the start bit. If valid, it transitions to the **RECEIVE** state, where receive\_shift\_en and receive\_frame\_counter\_en are enabled at every sample edge. Upon receiving the complete frame (receive\_done), the FSM moves to the **WAIT** state to check for a stop bit or break condition. If the line stays low, it enters the **BREAK** state; otherwise, it returns to **IDLE**. The error\_check signal is asserted for one cycle when reception completes, and receive\_load\_en is activated when valid data is ready or a break ends.

### **5.4.4 Receiver Shift Block**

#### **5.4.4.1 Block Diagram**



*Figure : Block diagram of Receive Shift Block*

#### **5.4.4.2 Description**

This block implements the UART receiver shift logic. It captures serial input data (uart\_rxd) and shifts it into a 10-bit register when the receive\_shift\_en signal is high. The source of the serial data can be selected between uart\_rxd and loop\_txd based on the loop control signal.

The shift operation is handled by a **universal shift register**, configured for right-shift when enabled. The wls signal determines the word length, and the pen signal selects whether to include a parity bit in the output.

The shifted data is processed into either data\_with\_parity or data\_without\_parity, and the result is sent out via rsr\_data. The received parity bit is extracted to received\_parity. A frame error is flagged on frame\_error when error\_check is high and the stop bit (LSB of shift\_reg\_out) is low.

### **5.4.5 Parity Checker**

#### **5.4.5.1 Block Diagram**



*Figure : Block diagram of Parity checker*

#### **5.4.5.2 Description**

This block performs parity checking on received UART data. It takes the 8-bit data from rsr\_data and computes its parity. The type of parity used is controlled by the pen (parity enable), eps (even parity select), and sp (stick parity) signals.

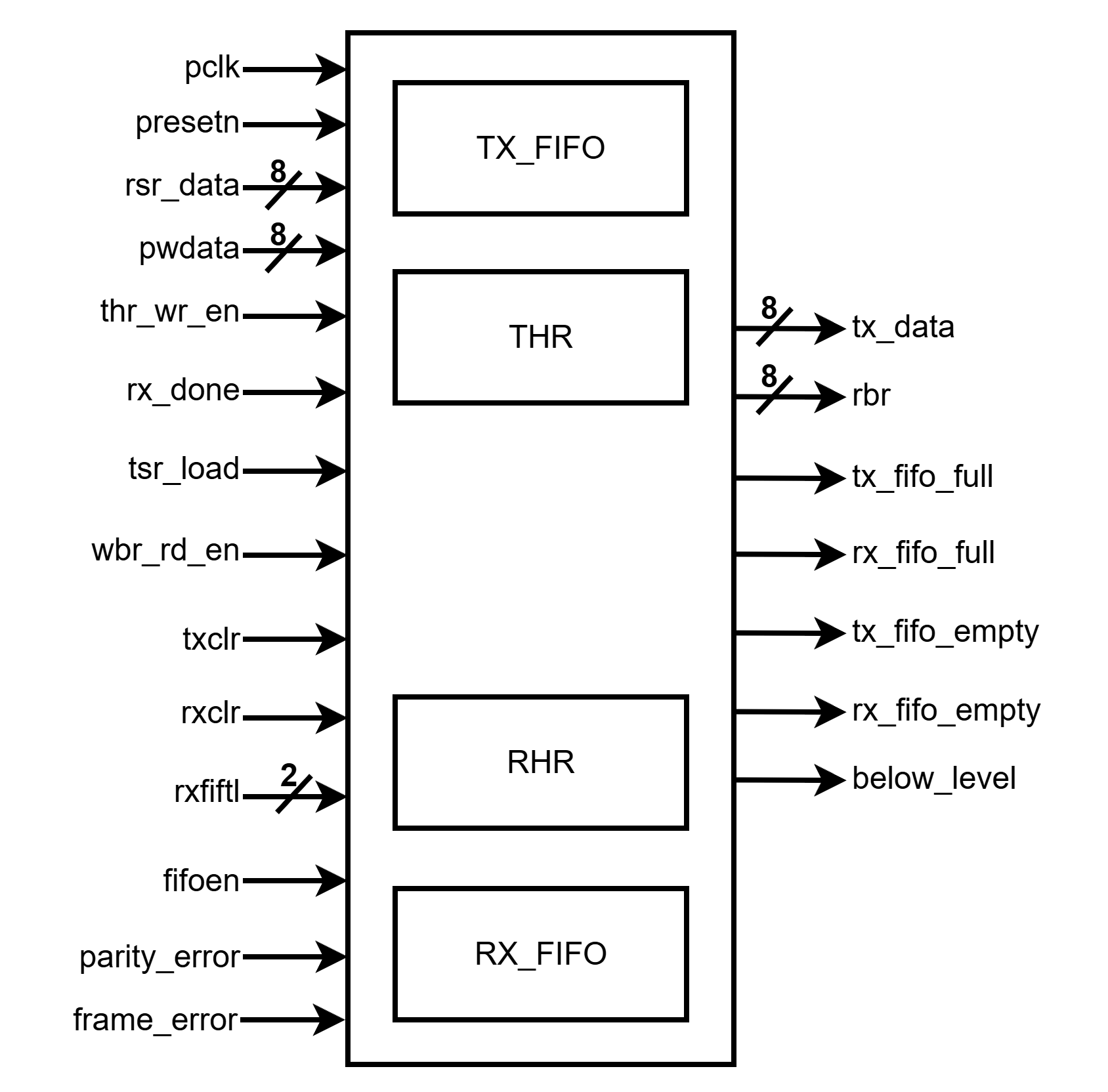
The computed\_parity is calculated as the XOR of all bits in rsr\_data. The expected\_parity is determined based on the eps and sp signals:

* If sp is high, a fixed parity bit is used, inverted or not depending on eps.
* Otherwise, the parity is computed dynamically as even or odd.

The received parity (received\_parity) is then compared with the expected\_parity. If they do not match and parity is enabled (pen), the output parity\_error is asserted

## **5.5 Buffers**

### **5.5.1 Block Diagram of Buffers**

****

**Fig.:** Block diagram of buffers block.

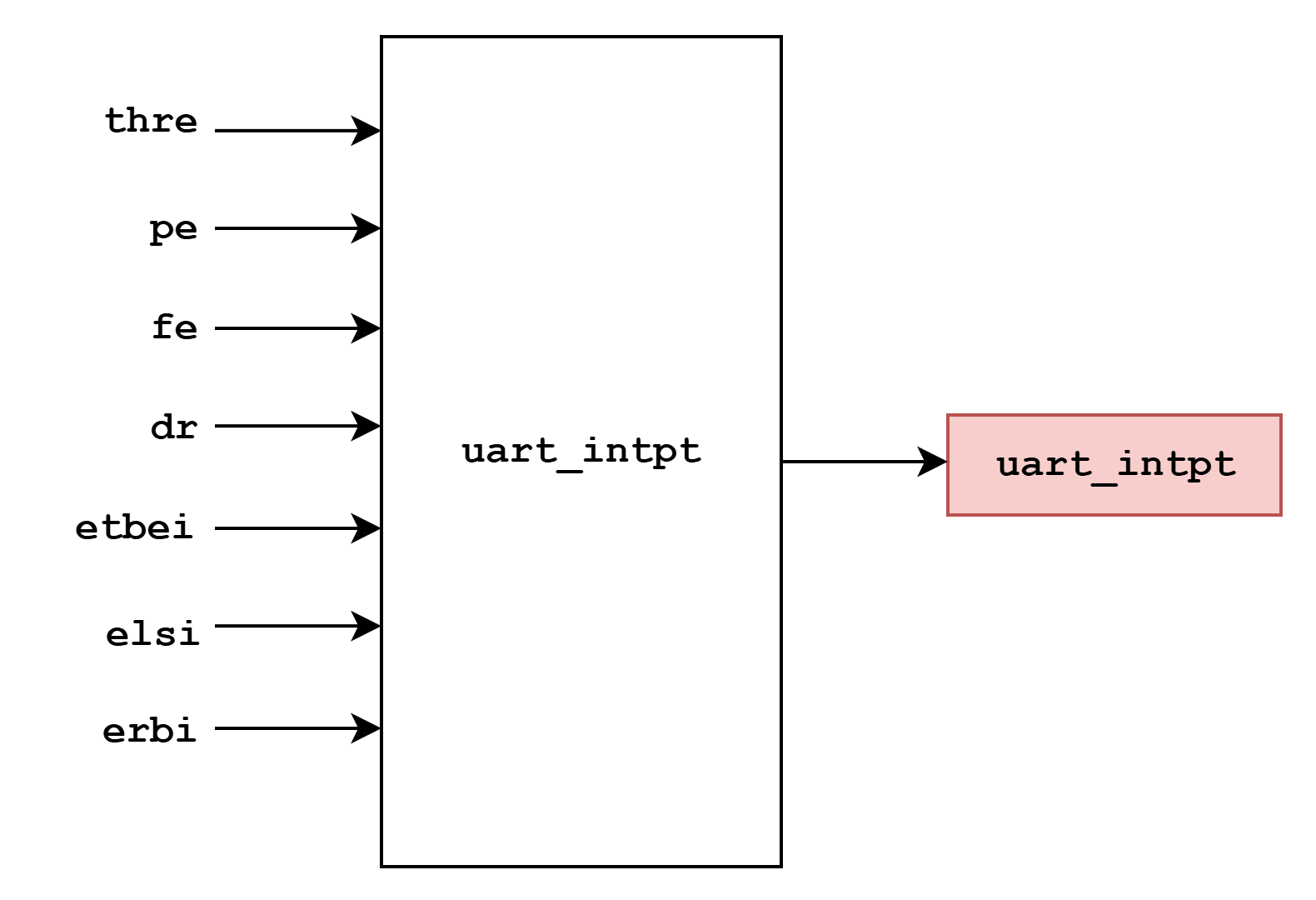
### **5.5.2 Block Description of Buffers**

This block consists of transmit and receive fifo and the(transmit holding register) the and rhr(receive holding register). Recieve and transmit fifo is of 16 depth. THR and RHR are designed using depth 1 fifo. Transmit fifo data width is 8 bit and receive fifo data width is 10 bit(8 bit data wih 2 bit errors(parity and frame error). Datapath is selected by the ***fifoen*** input coming from ***FCR***(FIFO Control Register).

This block generates the data which is needed in the next step for transmit or receive. Generated data are ***tx\_data*** which goes to the transmitter block and ***rbr*** which is the receive buffer register from which data is read by the apb interface. It also generates the empty or full condition signal of the fifos or the holding registers. It additionally generates a ***below\_level*** signal which indicates if the receiver fifo is filled with data which is below a certain level configured by ***rxfiftl***.

## **5.6 Interrupt generation block**

UART generates an interrupt when there is an error during transfer. It can be parity error, frame error or data-ready status for the receiver. Interrupt can be masked with the interrupt enable register (IER).



# **6.Registers**

## **6.1 Register List**

| **Name** | **Address** | **Width** | **Access** | **Description** |
| --- | --- | --- | --- | --- |
| RBR | 0h | 32 | R | [Receive Buffer register](#_dkzl4xi504aw) |
| THR | 0h | 32 | W | [Transmitter Holding Register](#_ne3dw72pgqpt) |
| IER | 4h | 32 | R/W | [Interrupt Enable Register](#_gtn1jb5m9fud) |
| IIR | 8h | 32 | R | [Interrupt Identification Register](#_tq7r2ymm8hq6) |
| FCR | 8h | 32 | W | [FIFO Control Register](#_biwae9viu5c4) |
| LCR | Ch | 32 | R/W | [Line Control Register](#_uktu9rqxmhjo) |
| LSR | 14h | 32 | R | [Line Status Register](#_5egqzlxggm8c) |
| DLL | 20h | 32 | R/W | [Divisor LSB Latch](#_jz52h9ec4mcq) |
| DLH | 24h | 32 | R/W | [Divisor MSB Latch](#_azme95tcqux1) |
| PWREMU\_MGMT | 30h | 32 | R/W | [Power and Emulation Management Register](#_mcjb7g1mv2jj) |

Table-: Register List

### 6.2.1 Receive Buffer Register (RBR):

| **Bit** | **31:8** | **7:0** |
| --- | --- | --- |
| **Name** | **Reserved** | **DATA** |
| **Access** | **R** | **R** |
| **Reset Value** | **0** | **0** |

**Bit[7:0]- DATA: Received data**

### 6.2.2 Transmit Holding Register (THR):

| **Bit** | **31:8** | **7:0** |
| --- | --- | --- |
| **Name** | **Reserved** | **DATA** |
| **Access** | **R** | **W** |
| **Reset Value** | **0** | **0** |

**Bit[7:0]- DATA: Data to transmit**

### 6.2.3 Interrupt Enable Register (IER):

| **Bit** | **31:3** | **2** | **1** | **0** |
| --- | --- | --- | --- | --- |
| **Name** | **Reserved** | **ELSI** | **ETBEI** | **ERBI** |
| **Access** | **R** | **R/W** | **R/W** | **R/W** |
| **Reset Value** | **0** | **0** | **0** | **0** |

**Bit[2]- ELSI: Receiver Line Status Enable**

When 0, Receiver line status interrupt is disabled.

When 1, Receiver line status interrupt is enabled.

**Bit[1]- ETBEI: Transmitter holding Register Empty Interrupt Enable**

When 0, Transmitter holding register empty interrupt is disabled.

When 1, Transmitter holding register empty interrupt is enabled.

**Bit[0]- ELSI: Receiver Data Available Interrupt Enable**

When 0, Receiver data available interrupt is disabled.

When 1, Receiver data available interrupt is enabled.

### 6.2.4 Interrupt Identification Register (IIR):

| **Bit** | **31:8** | **7:6** | **5:1** | **0** |
| --- | --- | --- | --- | --- |
| **Name** | **Reserved** | **FIFOEN** | **Reserved** | **IPEND** |
| **Access** | **R** | **R** | **R/W** | **R** |
| **Reset Value** | **0** | **0** | **0** | **1** |

**Bit[7:6]- FIFOEN: FIFOs enabled.**

When, 0 = Non-FIFO mode.

1 = Reserved

2 = Reserved

3 = FIFOs are enabled. FIFOEN bit in FIFO Control Register (FCR) is set to 1.

**Bit[0]- IPEND: Interrupt pending.**

When any UART interrupt is generated and is enabled in IER, IPEND is forced to 0. IPEND remains 0 until all pending interrupts are cleared or until a hardware reset occurs. If no interrupts are enabled, IPEND is never forced to 0.

When 0, Interrupts pending

When 1, No interrupts pending

### 6.2.5 FIFO Control Register (FCR):

| **Bit** | **31:8** | **7:6** | **5:3** | **2** | **1** | **0** |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Reserved** | **RXFIFTL** | **Reserved** | **TXCLR** | **RXCLR** | **FIFOEN** |
| **Access** | **R** | **R/W** | **R/W** | **W1C** | **W1C** | **R** |
| **Reset Value** | **0** | **0** | **0** | **0** | **0** | **1** |

**Bit[2]- RXFIFTL: Receiver FIFO trigger level**

When the trigger level is reached, a receiver data-ready interrupt is generated (if the interrupt request is enabled). When the FIFO drops below the trigger level, the interrupt is cleared.

When, 0 = 1 byte

1 = 4 bytes

2 = 8 bytes

3 = 14 bytes

**Bit[2]- TXCLR: Transmitter FIFO Clear**

Write a 1 to TXCLR to clear the bit.

When 0, No effect.

When 1, Clears transmitter FIFO and resets the transmitter FIFO counter. The shift register is not cleared.

**Bit[1]- RXCLR: Receiver FIFO Clear**

Write a 1 to RXCLR to clear the bit.

When 0, No effect.

When 1, Clears receiver FIFO and resets the receiver FIFO counter. The shift register is not cleared.

**Bit[0]- FIFIOEN: Transmitter and Receiver FIFO mode enable**

FIFOEN must be set before other FCR bits are written to or the FCR bits are not programmed. Clearing this bit clears the FIFO counters.

When 0, The transmitter and receiver FIFOs are disabled, and the FIFO pointers are cleared.

When 1, The transmitter and receiver FIFOs are enabled.

### 6.2.6 Line Control Register (LCR):

| **Bit** | **31:8** | **7** | **6** | **5** | **4** | **3** | **2** | **1:0** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Reserved** | **LOOP** | **BC** | **SP** | **EPS** | **PEN** | **STB** | **WLS** |
| **Access** | **R** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** | **R/W** |
| **Reset Value** | **0** | **0** | **0** | **0** | **0** | **0** | **0** | **0** |

**Bit[7]- LOOP: Loopback mode enable.**

When 0, Loopback mode is disabled.

When 1, Loopback mode is enabled. The UART\_TXD signal is set High and the UART\_RXD pin is disconnected. The output of the Transmitter Shift Register (TSR) is looped back in the Receiver Shift Register (RSR).

**Bit[6]- BC: Break control.**

When 0, Break condition is disabled.

When 1, Break condition is transmitted to the receiving UART. A break condition is a condition in which the UART\_TXD signal is forced to the spacing state.

**Bit[5]- SP: Stick parity.**

The SP bit works in conjunction with the EPS and PEN bits.

When 0, Stick parity is disabled.

When 1, Stick parity is enabled. When odd parity is selected (EPS = 0) the parity bit is transmitted as 1. When even parity is selected (EPS = 1) the parity bit is transmitted as 0.

**Bit[4]- EPS: Even parity select.**

Selects the parity when parity is enabled (PEN = 1). The EPS bit works in conjunction with the SP and PEN bits.

When 0, Odd parity is selected (an odd number of logic 1s is transmitted or checked in the data and PARITY bits).

When 1, Even parity is selected (an even number of logic 1s is transmitted or checked in the data and PARITY bits).

**Bit[3]- PEN: parity enabled.**

The PEN bit works in conjunction with the SP and EPS bits.

When 0, No PARITY bit is transmitted or checked.

When 1, Parity bit is generated in transmitted data and is checked in received data between the last data word bit and the first STOP bit.

**Bit[2]- STB: Number of Stop bits generated.**

The STB specifies the number of stop bits in each transmitted character.The receiver clocks only the first stop bit regardless of the number of stop bits selected.

When 0, One stop bit is generated.

When 1, Two stop bits are generated.

**Bit[1:0]- WLS: Word length select.**

Number of bits in each transmitted or received serial character

When, 0 = 5 bits

1 = 6 bits

2 = 7 bits

3 = 8 bits

### 6.2.7 Line Status Register (LSR):

| **Bit** | **31:8** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Reserved** | **RXFIFOE** | **TEMT** | **THRE** | **BI** | **FE** | **PE** | **OE** | **DR** |
| **Access** | **R** | **R** | **R** | **R** | **R** | **R** | **R** | **R** | **R** |
| **Reset Value** | **0** | **0** | **1** | **1** | **0** | **0** | **0** | **0** | **0** |

**Bit[7]- RXFIFOE: Receive FIFO Error**

**In Non-FIFO mode,**

When 0, There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver buffer register (RBR).

When 1, There is a parity error, framing error, or break indicator in the receiver buffer register (RBR).

**In FIFO mode,**

When 0, There has been no error, or RXFIFOE was cleared because the CPU read the erroneous character from the receiver FIFO and there are no more errors in the receiver FIFO.

When 1, At least one parity error, framing error, or break indicator in the receiver FIFO.

**Bit[6]- TEMT: Transmitter empty Indicator**

**In Non-FIFO mode,**

When 0, Either the transmitter holding register (THR) or the transmitter shift register (TSR) contains a data character.

When 1, Both the transmitter holding register (THR) and the transmitter shift register (TSR) are empty.

**In FIFO mode,**

When 0, either the transmitter FIFO or the transmitter shift register (TSR) contains a data character.

When 1, Both the transmitter FIFO and the transmitter shift register (TSR) are empty.

**Bit[5]- TEMT: Transmitter Holding Register empty Indicator**

**In non-FIFO mode,**

When 0,The Transmitter holding register (THR) is not empty. THR has been loaded by the CPU.

When 1, The Transmitter holding register (THR) is empty (ready to accept a new character). The content of THR has been transferred to the transmitter shift register (TSR).

**In FIFO mode,**

When 0, The Transmitter FIFO is not empty. At least one character has been written to the transmitter FIFO. The transmitter FIFO may be written to if it is not full.

When 1, The Transmitter FIFO is empty. The last character in the FIFO has been transferred to the transmitter shift register (TSR).

**Bit[4]- BI: Break Indicator**

**In non-FIFO mode,**

When 0, No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver buffer register (RBR).

When 1, A break has been detected with the character in the receiver buffer register (RBR).

**In FIFO mode,**

When 0, No break has been detected, or the BI bit was cleared because the CPU read the erroneous character from the receiver FIFO and the next character to be read from the FIFO has no break indicator.

When 1, A break has been detected with the character at the top of the receiver FIFO.

**Bit[3]- BI: Frame error Indicator**

A frame error occurs when the received character does not have a valid stop bit. In response to a framing error, the UART sets the FE bit and waits until the signal on the RX pin goes high. When the RX signal goes high, the receiver is ready to detect a new START bit and receive new data.

**In non-FIFO mode,**

When 0, No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR).

When 1, A framing error has been detected with the character in the receiver buffer register (RBR).

**In FIFO mode,**

When 0, No framing error has been detected, or the FE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no framing error.

When 1, A framing error has been detected with the character at the top of the receiver FIFO.

**Bit[2]- BI: Parity error Indicator**

A parity error occurs when the parity of the received character does not match the parity selected with the EPS bit in the line control register (LCR).

**In non-FIFO mode,**

When 0, No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver buffer register (RBR).

When 1, A parity error has been detected with the character in the receiver buffer register (RBR).

**In FIFO mode,**

When 0, No parity error has been detected, or the PE bit was cleared because the CPU read the erroneous data from the receiver FIFO and the next character to be read from the FIFO has no parity error.

When 1, A parity error has been detected with the character at the top of the receiver FIFO.

**Bit[1]- OE: Overrun error Indicator**

**In non-FIFO mode,**

When 0, No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR).

When 1, Overrun error has been detected. Before the character in the receiver buffer register (RBR) could be read, it was overwritten by the next character arriving in RBR.

**In FIFO mode,**

When 0, No overrun error has been detected, or the OE bit was cleared because the CPU read the content of the line status register (LSR).

When 1, Overrun error has been detected. An overrun error occurs only after the FIFO is full and the next character has been completely received in the shift register. An overrun error is indicated to the CPU as soon as it happens. The new character overwrites the character in the shift register, but it is not transferred to the FIFO.

**Bit[0]- DR: Data ready Indicator.**

**In non-FIFO mode,**

When 0, Data is not ready, or the DR bit was cleared because the character was read from the receiver buffer register (RBR).

When 1, Data is ready. A complete incoming character has been received and transferred into the receiver buffer register (RBR).

**In FIFO mode,**

When 0, Data is not ready, or the DR bit was cleared because all of the characters in the receiver FIFO have been read.

When 1, Data is ready. There is at least one unread character in the receiver FIFO. If the FIFO is empty, the DR bit is set as soon as a complete incoming character has been received and transferred into the FIFO. The DR bit remains set until the FIFO is empty again.

### 6.2.8 Divisor LSB Latch (DLL):

| **Bit** | **31:8** | **7:0** |
| --- | --- | --- |
| **Name** | **Reserved** | **DLL** |
| **Access** | **R** | **R/W** |
| **Reset Value** | **0** | **0** |

**Bit[7:0]- DLL:**

The 8 least-significant bits (MSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator.

### 6.2.9 Divisor MSB Latch (DLH):

| **Bit** | **31:8** | **7:0** |
| --- | --- | --- |
| **Name** | **Reserved** | **DLH** |
| **Access** | **R** | **R/W** |
| **Reset Value** | **0** | **0** |

**Bit[7:0]- DLH:**

The 8 most-significant bits (MSBs) of the 16-bit divisor for generation of the baud clock in the baud rate generator. Maximum baud rate is 128 kbps.

### 6.2.10 Power and emulation Management Register (PWREMU\_MGMT):

| **Bit** | **31:16** | **15** | **14** | **13:0** |
| --- | --- | --- | --- | --- |
| **Name** | **Reserved** | **UTRST** | **URRST** | **Reserved** |
| **Access** | **R** | **R/W** | **R/W** | **R/W** |
| **Reset Value** | **0** | **0** | **0** | **0** |

**Bit[15]- UTRST: UART Transmitter Reset**

When 0, the Transmitter is disabled and in reset state.

When 1, The Transmitter is enabled

**Bit[14]- URRST: Receiver FIFO Clear**

When 0, the Receiver is disabled and in reset state.

When 1, The Receiver is enabled.