

# Design and Implementation of UART using Verilog

Kavyashree S

1<sup>st</sup> semester, M Tech VLSI and Embedded Systems  
R.V College of Engineering  
Bengaluru, India  
[kavyasree131091@gmail.com](mailto:kavyasree131091@gmail.com)

**Abstract:** *Universal Asynchronous Receiver Transmitter (UART) is a serial communication interface. This paper presents the design of UART for FPGA based systems using Verilog. The UART design has programmable features for Transmission, Reception and Baud Rate generation. It has FIFO storage, programmable serial interface characteristics, complete status reporting capabilities and error detection. The design is implemented using Hardware Description Language Verilog. The design is simulated and verified on Xilinx ISE.*

**Keywords:** *UART, First IN First Out (FIFO), Baud Rate Generator (BRG), interrupts, registers*

## 1. Introduction

UART converts parallel data into serial and enables serial communication. Many processors need UART as an external interface. UART designed for FPGA can serve as an interface for FPGA based embedded systems which uses soft core processors. This reduces external routing problems and cost as FPGA has huge number of logic gates unused. FPGA can be a good solution for reconfiguration of the system hardware modification.

## 2. Problem Statement

For FPGA based applications, UART IC is used as Serial communication interface external to FPGA board. This is posing certain problems like:

- Extra space requirement to place IC on the board
- Routing problems
- High recurring engineering costs
- Flexibility issues while interfacing with different end devices.

The above problem can be approached by

- Implementing the UART IC characteristics on FPGA board to address the extra space requirement and routing problems.
- Designing for FPGA that has the ability to update the functionality by partial re-configuration of a portion of the design to address the cost and the flexibility issues.

## 4. Design Details

The design of UART mainly consists of 6 modules. Each module is designed individually and implemented using Verilog and simulated on Xilinx ISE and verified.

1. Interface
2. Configuration module
3. Baud Rate Generator
4. Transmitter
5. Receiver
6. Interrupt Controller

## 3. Proposed Solution

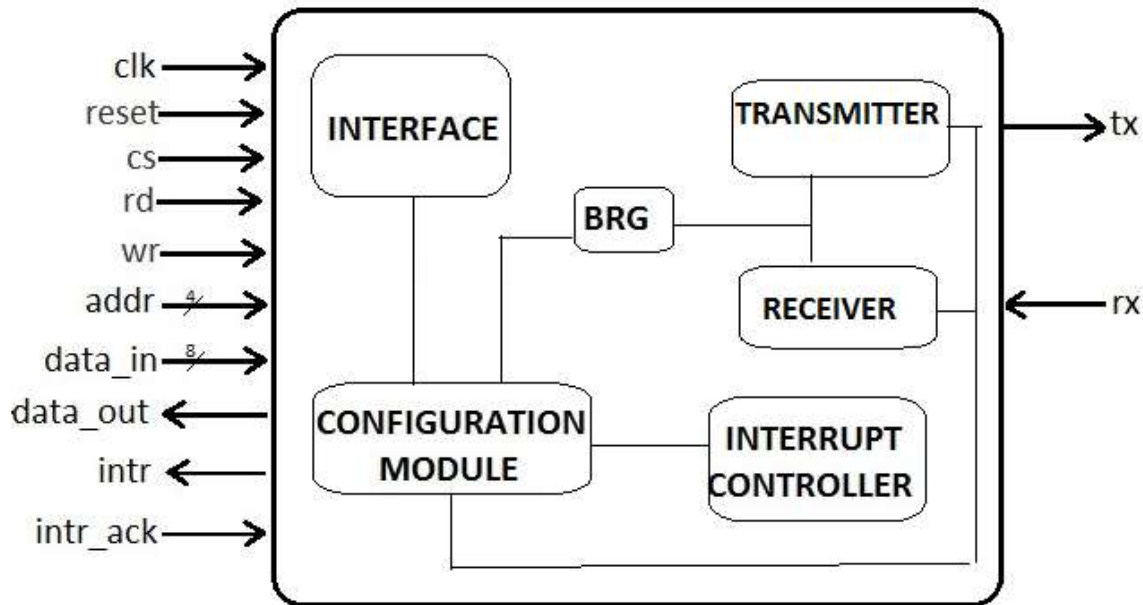


Fig 1.UART Module

### 1. Interface

The two subsystems involved in communication interfaced to UART module may be operating at different clock frequencies than UART. Therefore Interface module is used to synchronize the control signals (cs, rd, wr ) which indirectly synchronizes the address and data .

### 2. Configuration module

Configuration module is a set of controlling, monitoring and data transfer registers. Configuration module enables the UART to be suited for several applications. It allows the UART to become programmable to different

configurations. The registers may be read only, write only or read write. This module consists of 9 registers and each is accessed using 3 bit address lines to this module. All the registers are of 8 bit. The memory mapping is shown in fig.2

### 3. Baud Rate Generator(BRG)

BRG specifies the rate at which transmission or reception should happen. During transmission the data is shifted out to Tx line at Baud Rate and in reception data on Rx line is sampled at baud rate. It is configurable with 8 bit latch registers (DLL and DLH) to different baud rates.

Divisor (decimal) = (clock frequency) / (baud rate x clock sampling rate)

## Memory Mapping

<u>Register</u>	<u>Address</u>	<u>R/W</u>
Transmit Holding Register (THR)	0x00	W
Receive Holding Register (RHR)	0x01	R
FIFO Control Register (FCR)	0x02	W
Line Control Register (LCR)	0x03	R/W
Line Status Register (LSR)	0x04	R
Interrupt Enable Register (IER)	0x05	R/W
Interrupt Status Register (ISR)	0x06	R
Divide Latch Low Byte (DLL)	0x07	W
Divide Latch High Byte (DLH)	0x08	W

Fig 2: Memory map of Registers

### 4. Transmitter

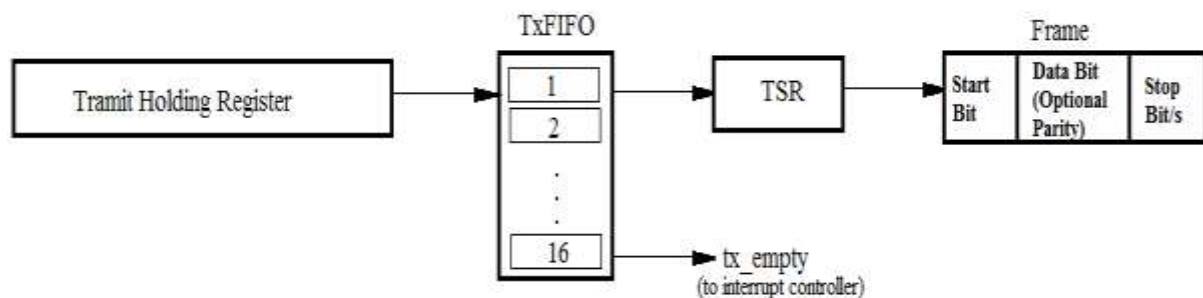


Fig 3: Transmitter Subsystem

The UART Transmitter consists of TxFIFO, Transmit Shift Register (TSR) which functions along with Transmit Holding Register (THR). The Transmit holding register is an 8-bit register

providing a data interface to the host processor. TxFIFO is used to buffer the data to be transmitted when two subsystems are operating at different rates. Tx FIFO stores the data up to 16 bytes. Transmit

shift register (TSR) shifts out the data to Tx line at baud rate after framing.

Framing is done to synchronize the communication. It consists of start, stop, parity and data bits and optional parity are specified by Line Control Register (LCR). The TSR is a state machine having five states- idle, start, data, parity (optional) and stop.

idle: The Tx line has a value '1' when there is no transmission.

start: Appends a bit '0' at the start of data transmission.

data: Shifts all data bits (specified by FCR) at baud rate (specified by DLL and DLH).

parity: An optional parity bit is appended as specified by LCR. The parity type may be odd or even.

stop: Specify the end of one frame transmission (specified by LCR).

## 5. Receiver

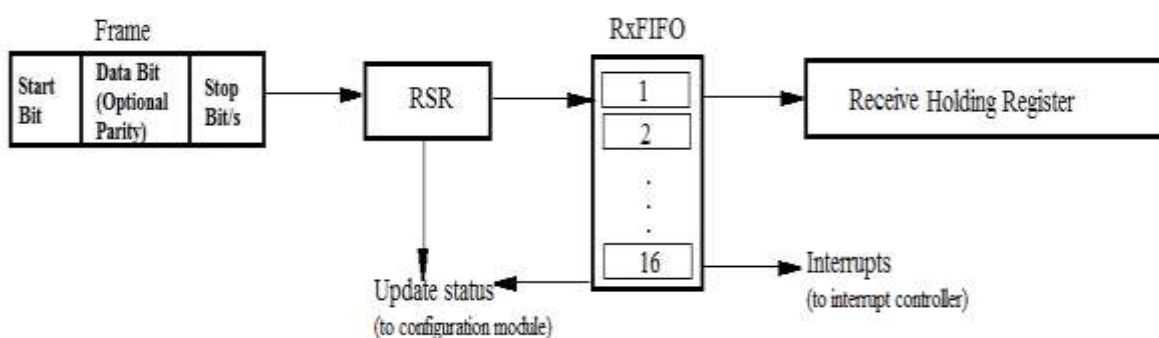


Fig 4: Receiving Subsystem

The receiver section contains an 8-bit Receive Shift Register (RSR) and 16 bytes of FIFO which works along with a byte-wide Receive Holding Register (RHR).

The RSR samples the Rx line at baud rate at the center of each bit period to avoid false detection like glitches. RSR is serial to parallel converter. It removes the framing content bits like start, stop and uses parity to detect errors. If there were any error(s), they are reported in the LSR register bits. The sampled data is put into Rx FIFO and it is read out through RHR address onto data\_out. RHR reads continuously the new data stored in Rx FIFO.

## 6. Interrupt Controller

The receiving event and transmitting events are interrupt driven that is an interrupt occurs when either Tx FIFO or Rx FIFO reaches trigger levels.

These are serviced by corresponding transmission and reception. Apart from this during transmission and reception there may be many undesired events. These may lead to improper functioning of the

UART. Interrupt controller handles such situations and takes a course of action to come of situation to proper functioning. The user can configure the interrupts via the Interrupt Enable Register (IER). When the processor is interrupted, it can find out more details regarding the interrupt by reading the Interrupt Identity Register (IIR). The design is capable of handling 4 interrupts.

Timeout interrupt: Interrupt arises when a particular data remains in the Rx FIFO for a very long time. The timeout counter is reset by following events:

- When new data is received from the UART
- When RHR is read

- When Rx FIFO is empty

As in many standard UART ICs the timeout is designed to be 44 bit-times which allows enough large time to fill more data.

**Trigger Level:** When FIFO(RxFIFO or TxFIFO) reaches a certain level configured by the user in FIFO control Register (FCR) this interrupt is raised. This is used to avoid the reading one byte at a time. When the FIFO reaches the level all the data are transmitted or received data is read at once saving the system's resources. The FIFO trigger level can be set to 1, 4, 8 and 14 bytes. If the received data does not reach the level then the timeout interrupt is used to read out the data.

**Tx empty:** It is raised when TxFIFO is empty and transmission is attempted.

**Receive data error:** This interrupt is raised when there are errors in the received data. The errors may be parity or framing errors.

## 5. Results and Discussion

The UART designed is tested and verified. Each component is individually tested and the integrated module is tested for the overall function. The simulated results are shown below:

### Transmission

The data sent is transmitted on the Tx line at baud rate. The baud rate is configured to 57.6kbps. The word length is 8 bits. The data 10010111 is transmitted on Tx line with start bit '0', stop bit '1'

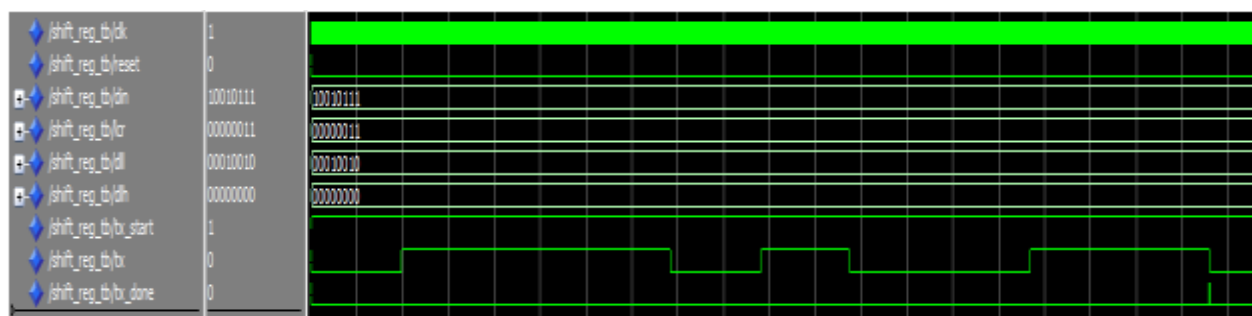


Fig. 5 Simulation Waveform of Transmission

### A. Receiver



Fig. 6. Simulation Waveform of Receiver

### B. Complete UART Module

The UART with all the controls is shown. It shows the transmission of data '11100100' on Tx with start and stop bits and also the receiving of '11101000' on Rx line illustrating the duplex behavior.

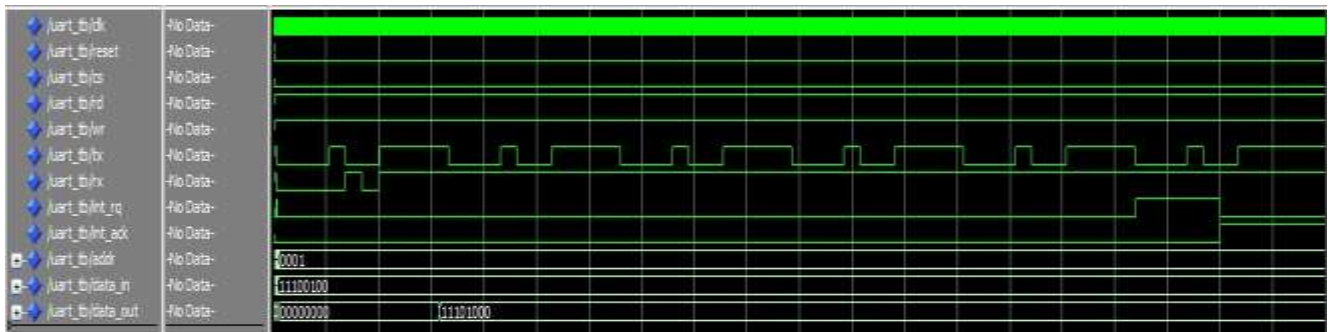


Fig. 7 Simulation Waveform of UART Module

## 6. Conclusion

In this paper the design of UART is described along with the individual modules contained in it. Each module is described using Verilog and simulated using Xilinx ISE. This design provides a programmable UART suitable for variety of FPGA based systems reducing the routing problems, cost issues and improving flexibility and integrity.

## 7. References

1. Fang Yi-yuan, Chen Xue-jun, "Design and Simulation of UART Serial Communication Module Based on VHDL", 3rd International Workshop on Intelligent Systems and Applications, 2011, Wuhan, China.
2. Mahat, N.F., "Design of a 9-bit UART module based on Verilog HDL", 10th IEEE conference on semiconductor electronics, 2012.
3. Datasheets
  - Texas Instruments
  - Exar
  - Incore.
4. Microcomputer Systems- the 8086/8088 Family: Yu-cheng Liu & Glenn A. Gibson
5. Pong P Chu, "FPGA prototyping by Verilog Examples", Wiley publications