Logic Design Lab

PROJECT DOCUMENTATION

UNIVERSAL ASYNCHRONOUS RECEIVER TRANSMITTER

SUBMITTED BY: GROUP 12
Astha Agarwal (110050018)
Ashish Sonone (110050022)
Anand Soni (110050037)
Mridul Jain (110040083)
Vikash Challa (110050077)

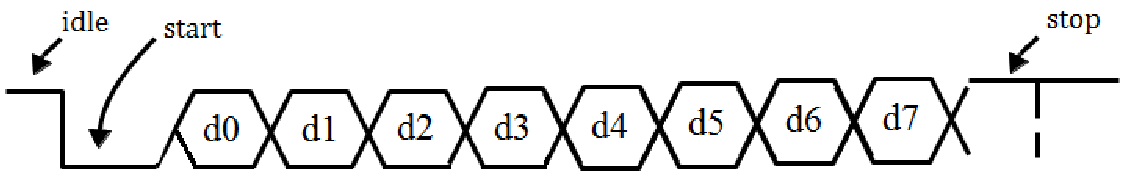
Objective

To design and simulate universal asynchronous receiver / transmitter circuit (UART) in Xilinx ISE. Implement the simulated design on ATLYS board.

* Baud Rate- 19.2 KHz
* Loop back the received data to transmitter.

Functional Specification

Data is transferred or received one byte at a time using the format shown in the figure below.



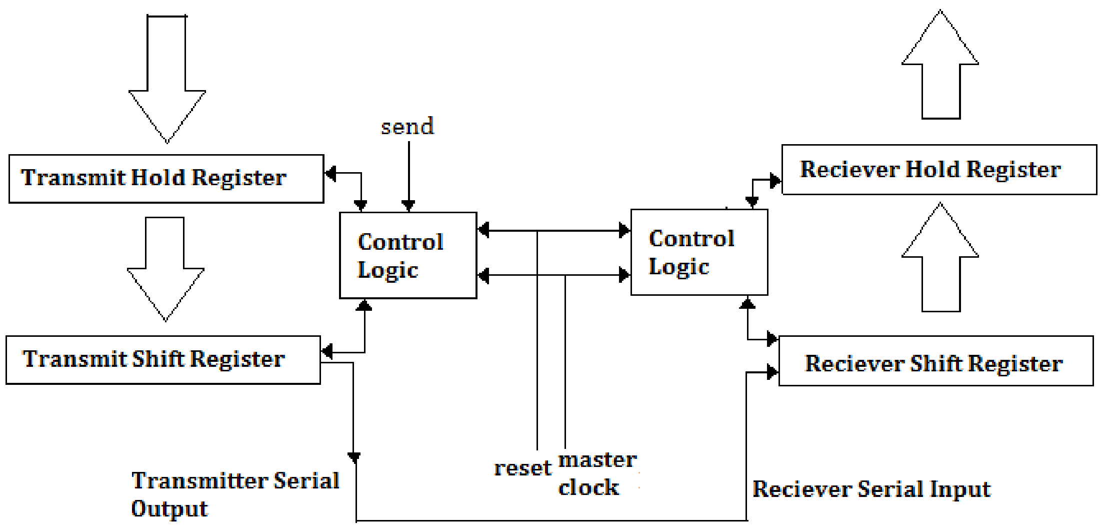
The transmission character is composed of an 8-bit data byte from d0 to d7, sent LSB first, preceded by a start bit (LOW), and followed by a stop bit (HIGH). When no character is being transmitted, the line is idle (HIGH).

Preview

A **Universal Asynchronous Receiver/Transmitter**, abbreviated **UART** is a hardware device that provides asynchronous serial communication between different devices. The UART can be used to control the process of breaking parallel data from the computer down into serial data that can be transmitted over the communication channel. At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Since UART allows the devices to communicate without the need to be synchronized; it is an asynchronous device. Since the data stream has no clock, data recovery depends on the transmitting device and the receiving device operating at close to the same bit rate. The UART receiver is responsible for the synchronization of the serial data stream and the recovery of data characters. The rate at which bits (pulses per second) are sent is called the baud rate. As specified in the functional specification, we are using a baud rate of 19200 Hz.

Procedure

The UART has one receiver module and one transmitter module. The input to receiver is provided by the user in serial form as shown above. The output of receiver is a parallel 8-bit output, acts as an input to the transmitter module. The transmitter module converts back this input into the same format as input i. e. after appending a low start bit before the input and high stop bit after it.

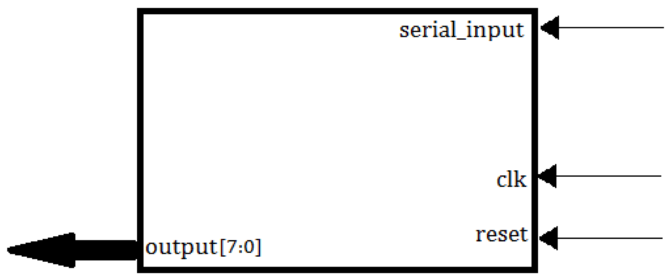


UART (Block)

Master-clock is same for both modules. We explain the functionalities of receiver and transmitter in the following sections one by one.

Receiver

The receiver shifts in data bit-by-bit and reassembles the data byte. The receiver tests the state of the incoming signal on each clock pulse, looking for the beginning of the start bit. If the apparent start bit lasts at least one-half of the bit time, it is valid and signals the start of data. If not, it is considered a false pulse and is ignored. After waiting a further bit time, the state of the line is again sampled and the resulting level clocked into a shift register. After 8 bit periods have elapsed, the contents of the shift register are made available (in parallel fashion) is sent as output.



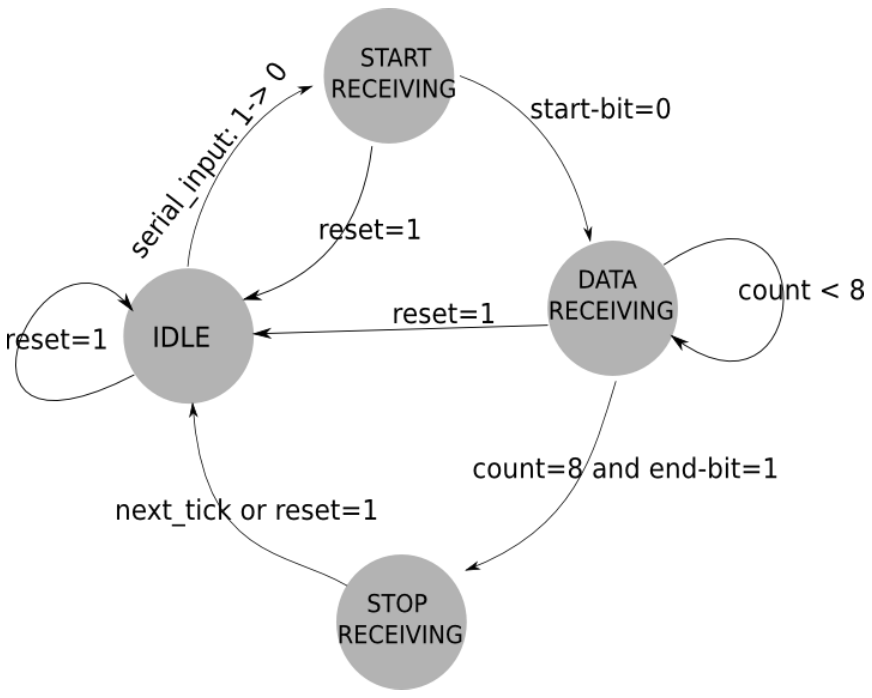
Block Diagram (Receiver)

output[7:0] : 8 bit parallel output

serial_input : held high when no transmission occurs

reset : gets us back to the idle state.

The following state machine implements the receiver:



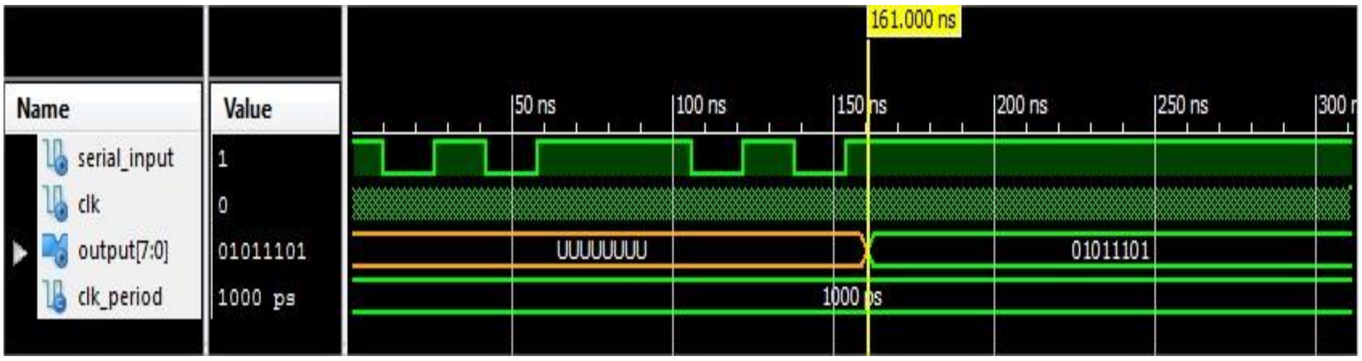
idle: when no output/input action is being performed, output is high

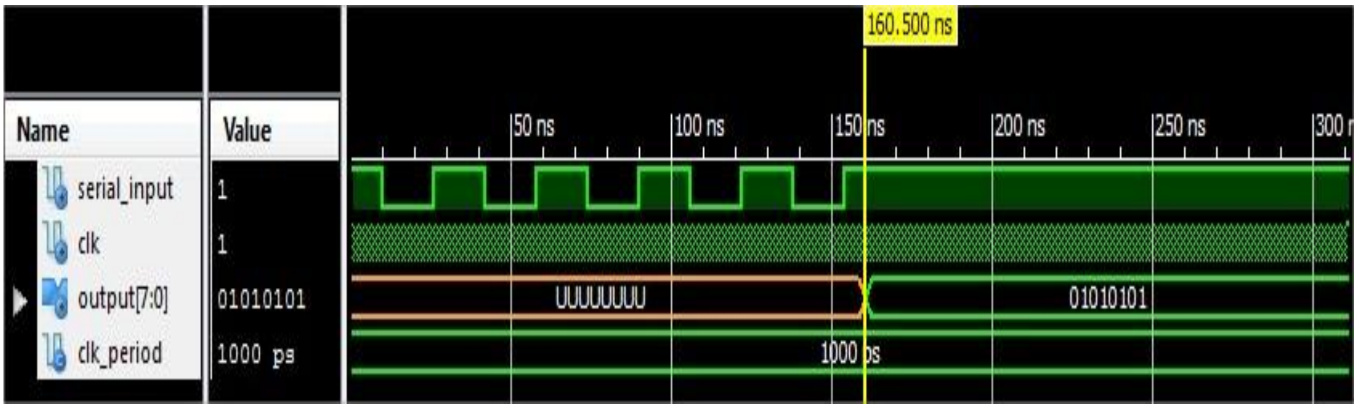
start_receiveing: Enters when detects start_bit

data_ receiveing: Recieving the 8 data bits (using counter and SIPO) through the serial input

stop_ receiveing: Enters when detects a high bit after receiving 8 data bits.

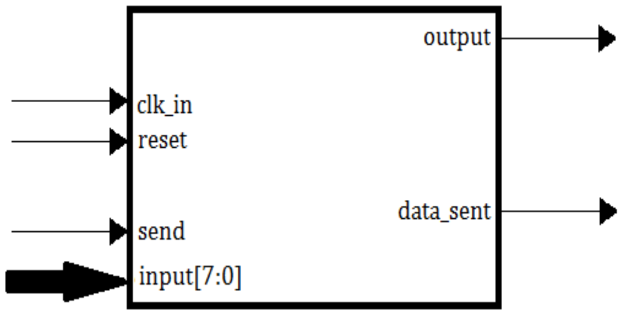
Timing Diagrams:





Transmitter

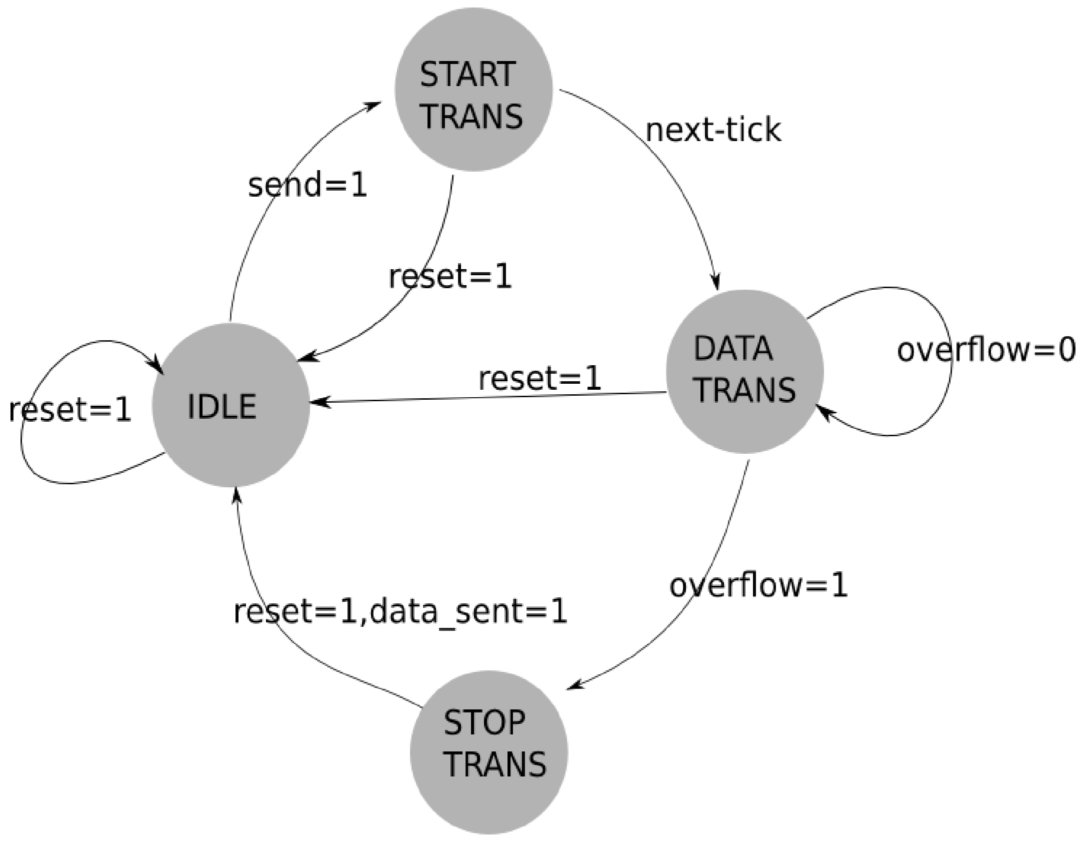
The transmitter is a special **shift** register that loads data in parallel and then shifts it out bit-by-bit. The process of transmitting data through the UART begins when **send** input is triggered (goes from high to low). The UART hardware generates a start bit, shifts the required number of data bits out to the line, generates and appends the stop bit.



input[7:0] : 8 bit parallel input
data_sent: High when data is read out.
output : held high when no transmission occurs
reset : gets us back to the idle state.
send: send bit to load and start transfer of parallel input

Block Diagram (Transmitter)

We use simple state machine to implement the transmitter:



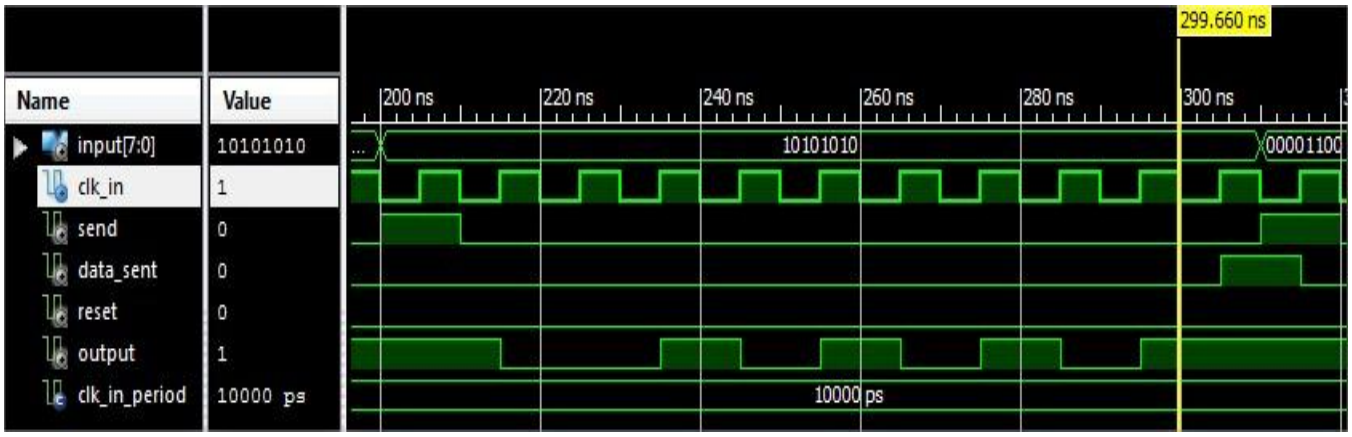
idle: when no output/input action is being performed, output is high

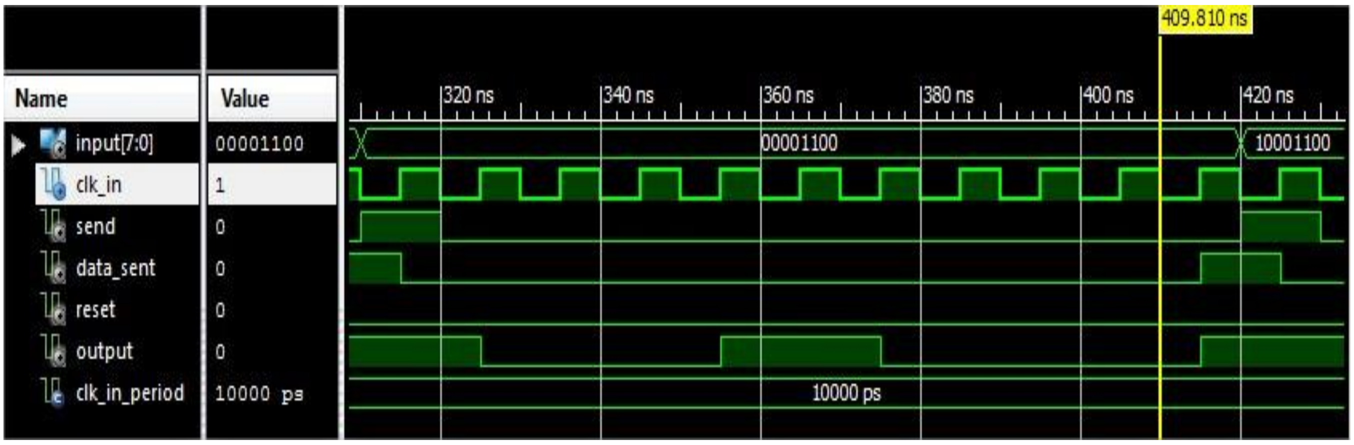
start_transmission: Enters this state when send is '1' (load the parallel input in register), outputs low start bit

data_transmission: Transmitting the 8 data bits (using counter and PISO), by load as 0

stop_transmission: Adding high bit to indicate completion of data transfer

Timing Diagrams:



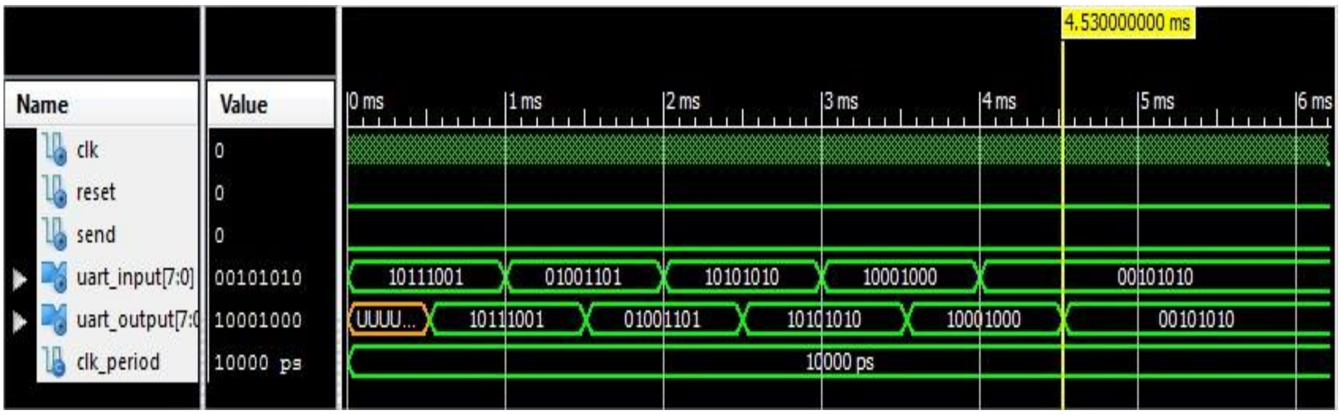


Final UART Simulation

So, now we have implemented the receiver and transmitter modules. All we need to do is to combine them together and make a sample UART.
As specified, we use a master clock of 100 MHz (as this is the frequency of clock in ATLYS board) to generate a clock of 19.2 KHz for Transmitter and 370.2 KHz for Receiver (8 times the frequency of Transmitter). For both the components, master clock is the same.

The output of the transmitter acts as the input for receiver. Input to UART is a parallel 8-bit input and output is also the same, after passing through the transmitter and receiver respectively.

Timing Diagram:



4.530000000 ms
Name
Value
0 ms
1 ms
2 ms
3 ms
4 ms
5 ms
6 ms
clk
0
reset
0
send
0
uart_input[7:0]
00101010
10111001
01001101
10101010
10001000
00101010
uart_output[7:0]
10001000
UUUU...
10111001
01001101
10101010
10001000
00101010
clk_period
10000 ps
10000 ps