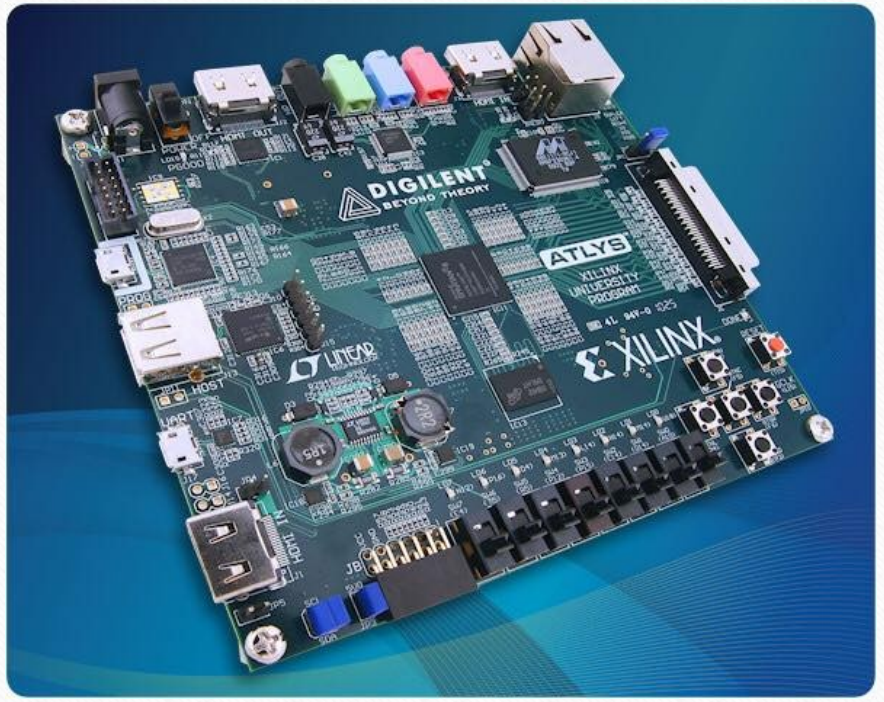
CS 226 Project

UART Design

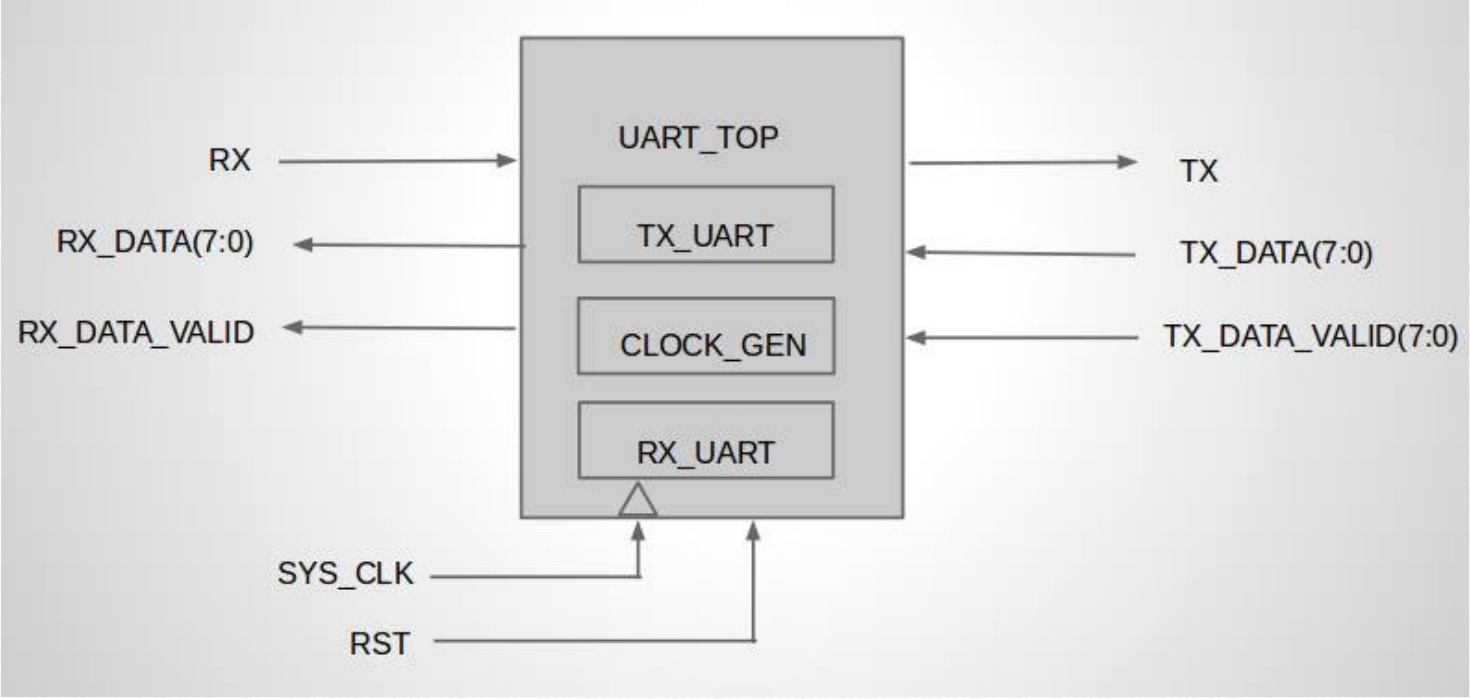
(Universal Asynchronous Receiver Transmitter Circuit)

Team Members:
Jagannath Vishal.R(130050043)
Mohammed Owais Khan(130050050)
Venkata Aditya Kusupati(130050054)
Sanampudi Venkata Sailesh(130050058)
Venkateshwara Rao Chippada(130050074)

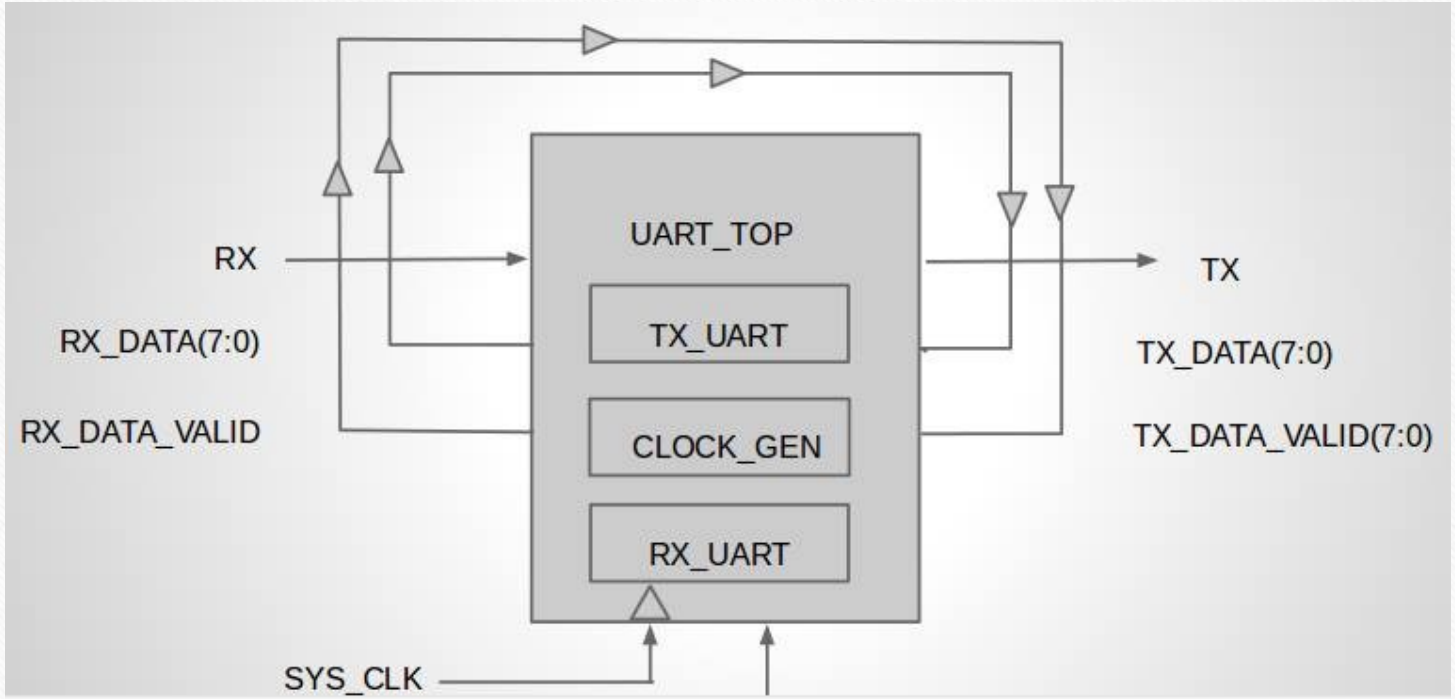
Using Xilinx ISE



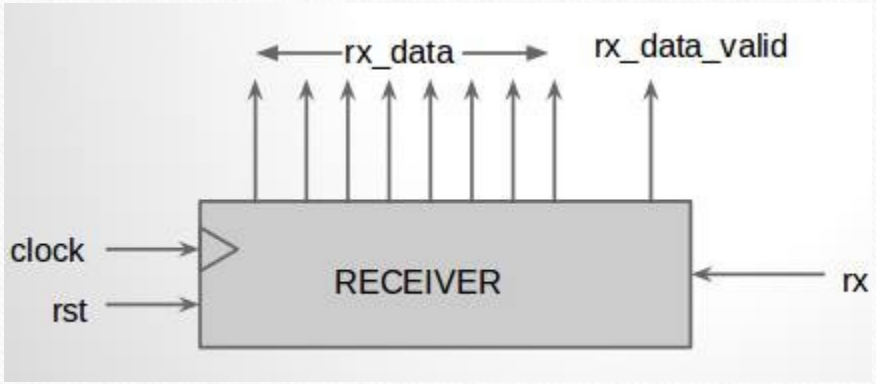
RTL Specification (Block Diagram)

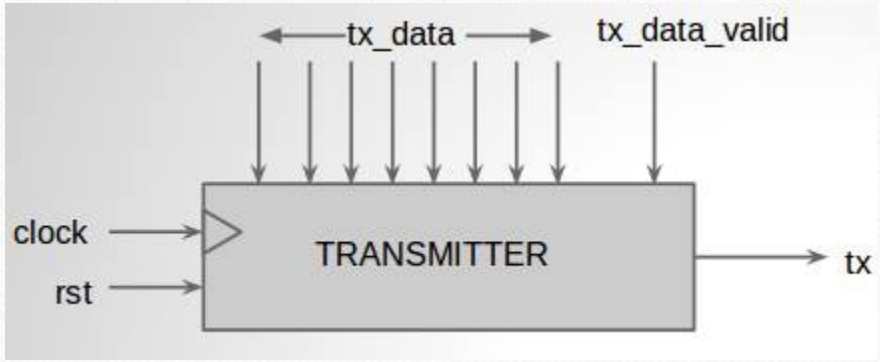


RTL Specification with Loopback



Receiver and Transmitter Chip Diagrams





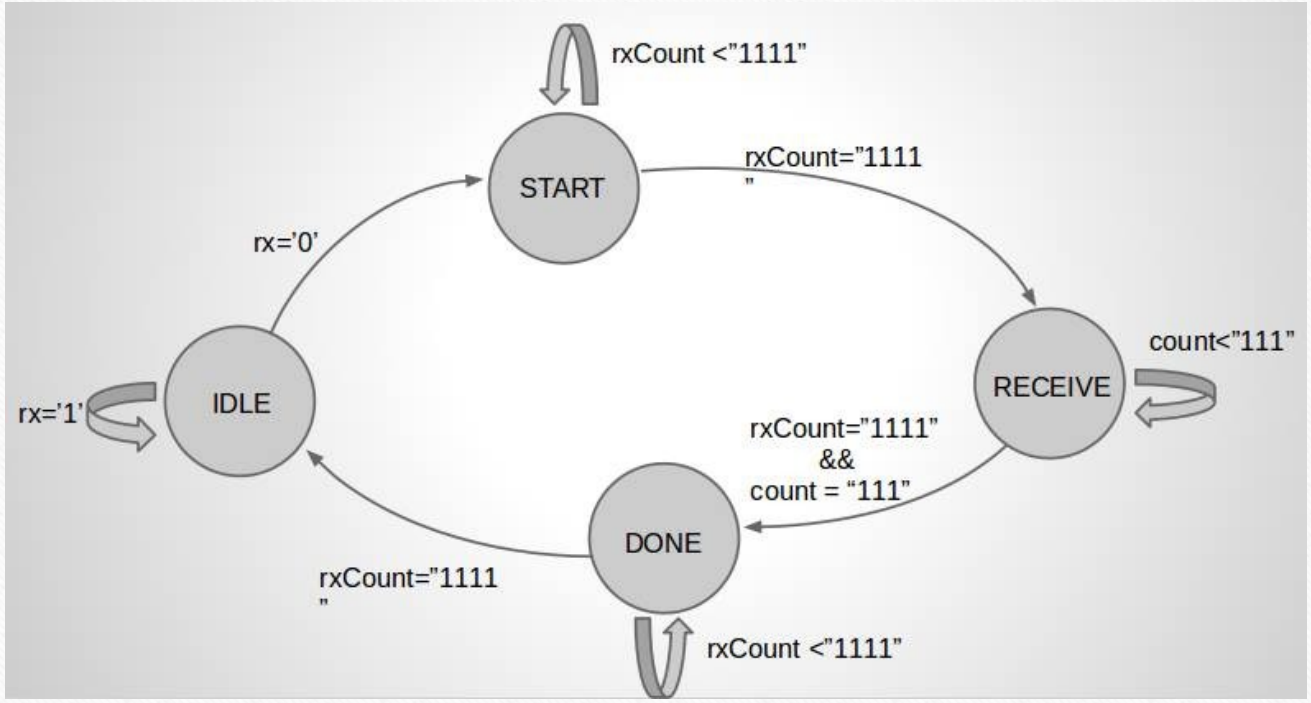
clock_gen.vhd

- The System Clock is 100MHz and the required Baud Rate is 19.2kHz. Each bit is sampled 16 times.
- It turns out that the system clock is too fast for the requirements. So, clock_gen generates an alternate clock, which provides the required frequency.
- For this, the new clock is incremented every (100x1000)/(19.2x16x2)=163 system clock cycles.

rx_uart.vhd

- This program receives the sampled data(each bit repeated 16 times), in serial order, groups the data into bytes and outputs them.
- The FSM has four states, namely IDLE, START, RECEIVE, DONE.
- The system remains in IDLE until a zero is encountered and then moves to START.
- Once all the other 15 sampling bits of zero are received, it stays in START and then moves into RECEIVE.
- The state remains RECEIVE until next seven bytes are received and then it moves to DONE, from where it moves to IDLE after resetting values.

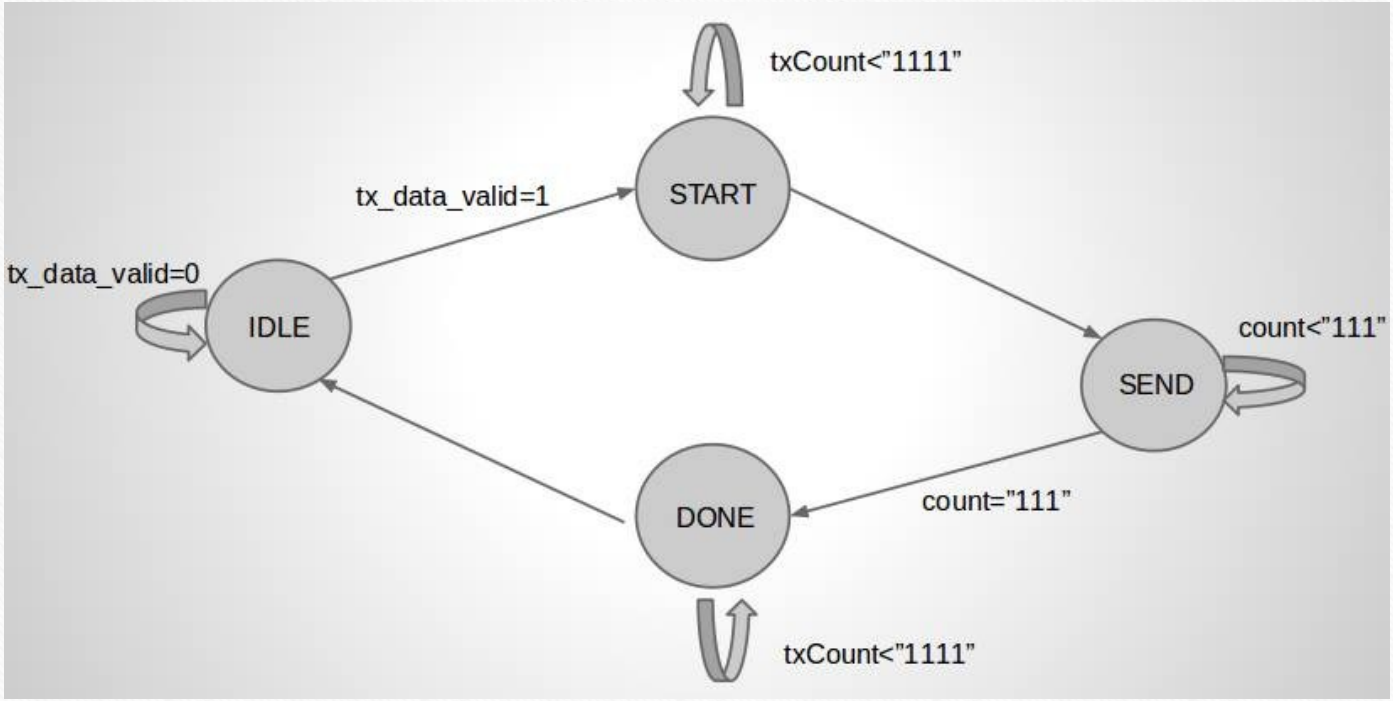
Receiver Side FSM



tx_uart.vhd

- The program receives a byte of data and converts it into the sampled and encoded form.
- This FSM has four states, namely IDLE, START, SEND, DONE.
- When in IDLE state, the output is 1. When the program receives the input data, it moves into the START state.
- In START state, the program outputs the start sequence, (sampled ‘0’) after which it enters the SEND state.
- In the SEND state, the program send the sampled data obtained from the input. It then proceed into the DONE state.
- In DONE state, all values are reset, and the program returns to the IDLE state.

Transmitter Side FSM



uart_top.vhd

- This program unifies the rx_uart.vhd file and the tx_uart.vhd file for the purpose of the demonstration.
- The data from rx_uart.vhd is looped back as parameters into tx_uart.vhd, and this is done as a part of this file.

Modular Interconnections

- Both rx_uart.vhd and tx_uart.vhd use the services of clock_gen.vhd for the modified clock which they require.
- In view of demonstration, uart_top.vhd uses the services of rx_uart.vhd and tx_uart.vhd.
- This is done by assigning the values of tx_data and tx_data_valid variables of the tx_uart.vhd file to rx_data and rx_data_valid variables of the rx_uart.vhd file, in the uart_top.vhd file.

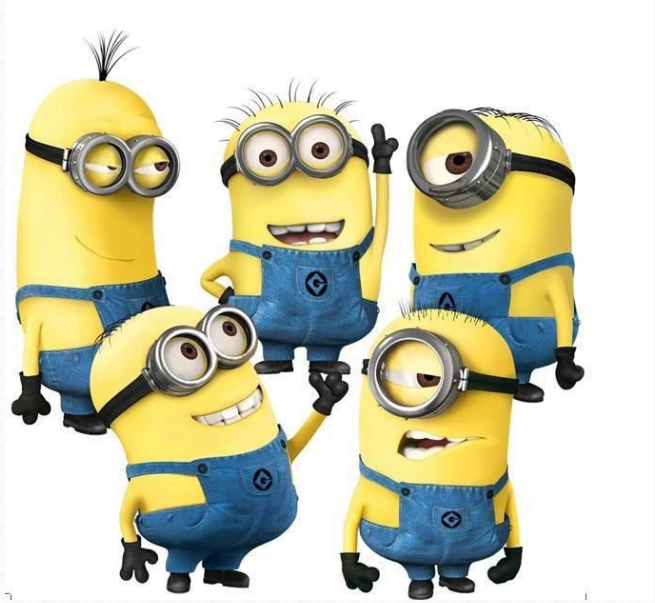
Interesting Inferences

We learnt a lot about the working of input and output modules of computational systems. We learnt about how errors are tackled in real systems by using sampling of input data. Also, as has been the case with many other projects done, we also learnt more about the power and potential of object-oriented programming and modularisation of the task at hand.

Individual Contributions.

Team Member	Contribution
Jagannath Vishal.R	clock_gen.vhd and Report.
Mohammed Owais Khan	uart_top.vhd and its test bench
Venkata Aditya Kusupati	tx_uart.vhd and its test bench
Sanampudi Venkata Sailesh	rx_uart.vhd and its test bench
Venkateshwara Rao Chippada	uart_top.vhd and Xilinx and TeraTerm Implementation

Thanks a Lot!!!



P.S. Everything was awesome