Digital Design Verification

Lab Manual # 11 – Communication Protocols

(UART using SV)

Release: 1.0

Date: 28-Feb-2024

NUST Chip Design Centre (NCDC), Islamabad, Pakistan

Revision History

Revision Number	Revision Date	Revision By	Nature of Revision	Approved By
1.0	28/02/2024	Hira Sohail, Muhammad Bilal	Complete manual	Sir Musaddiq

Contents

Objective

- Clearly articulate the definition, function, and module division of UART to provide a comprehensive understanding of its role in communication systems.
- Instruct students on building UART using System Verilog emphasizing a detailed simulation process.

Tools

- Xilinx Vivado

Introduction

In a Serial UART (Universal Asynchronous Receiver Transmitter) communications system, a transmitter is used to send a digital signal to a receiver. Compared to other communication protocols, UART is very useful because it is an asynchronous system; it can transmit signals without needing to synchronize with a clock signal. This method of transmission is extremely useful for reducing wires and I/O pins compared to parallel systems which transmit data faster but are far more complex. The serial transmission signal is sent in binary format, one data point after another, and flagged by start and stop bits, not decided by a clock signal. This is where the term Asynchronous Serial Transmission comes from.

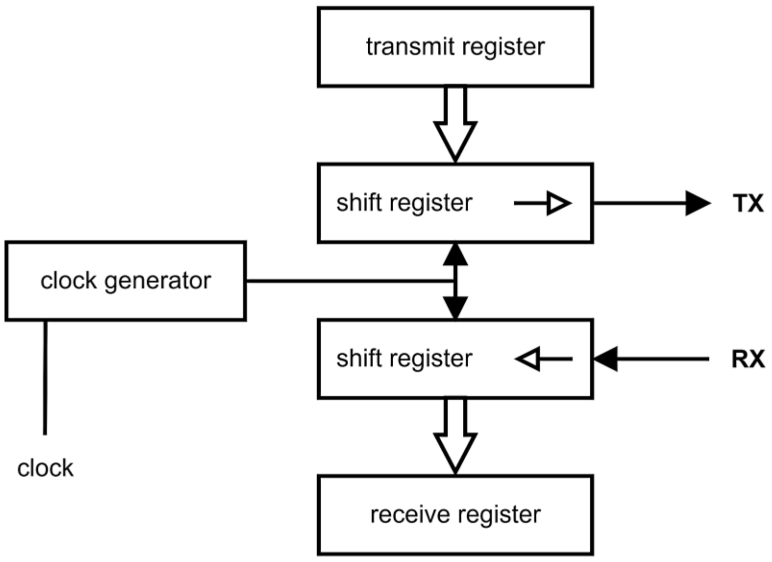


Fig 1: Block Diagram of UART

To successfully transmit an asynchronous signal, the following items must be considered:

- baud rate
- data bits
- synchronization bits
- parity bits

Baud Rate

The baud rate, specified in bits-per-second (bps), indicates how fast a signal is sent over a serial line. One of the most common baud rates is 9600bps which will be used as the default for this lab. In a communications system, both the receiver and the transmitter will have to match baud rates to successfully transfer data.

Data Bits and Endian-ness

Data bits are derived from the actual signal being sent over the line. The standard data packet is 8-bits, or a single byte, but can be other sizes. Since these bits are sent in a serial manner, the receiver needs to know whether the first bit received is the most significant bit or least significant bit. This is known as the endian notation of the data. In a big-endian system, the most significant bit is stored first with the following bits in decreasing significant order. The little-endian format is the exact opposite and stores the least significant bit in the first location. As with the baud rate, the endian notation of the communications system can be decided beforehand and hard-coded into the system.

Synchronization Bits

Besides data bits, the serial protocol also includes a start and stop bit that bookend the serial data packet. The start and stop bits do not transmit data, but rather mark the beginning and end of a data packet. The start bit is always set as a data line going from high to low with the stop bit going back from the last data signal to high. In this case, a constant high line indicates an idle state.

Parity Bits

Parity bits are a basic error checking method. When data is being transferred, occasionally that data becomes lost along the way or a transition is missed by the receiver. Sometimes the last data bit is used as a parity check to see if data has been missed. If the total sum of the data packet is even, then the parity bit is set to 0. Alternatively, if the sum of the data byte is odd, then the bit is set to 1. That way, the receiver can count how much data was received, compare that to the parity bit, and determine if all of the data has been received. Parity bits are useful for validating data but is optional.

Top level diagram

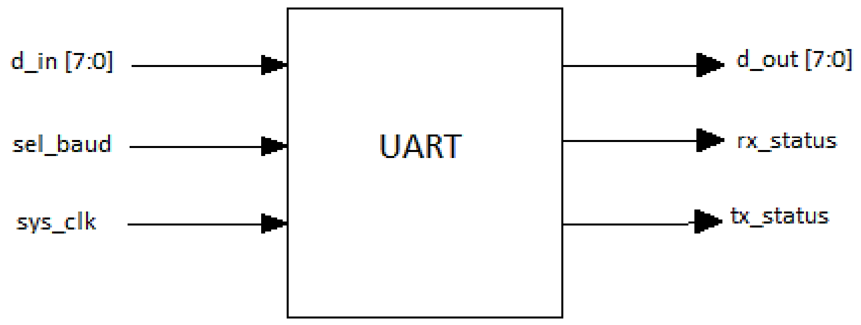


Fig 2: Top level diagram of UART.

Pins	Width	I/O	Description
d_in	8 bits	Input	Input data from data bus.
sel_baud	2 bits	Input	Used to select different baud set for different values of pin.
sys_clk	1 bit	Input	Input clock to UART from FPGA board onwhich it runs.
d_out	8 bits	Output	Output data sent to data bus.
rx_status	1 bit	Output	Shows status of receiver.
tx_status	1 bit	Output	Shows status of transmitter.

Table 1: Pin description for top level diagram of UART.

Architecture

UART architecture consists of three main blocks Transmitter, Receiver and Baud rate generator. The data is taken from the computer or the peripheral devices. Below are the block level micro-architecture and working of the modules.

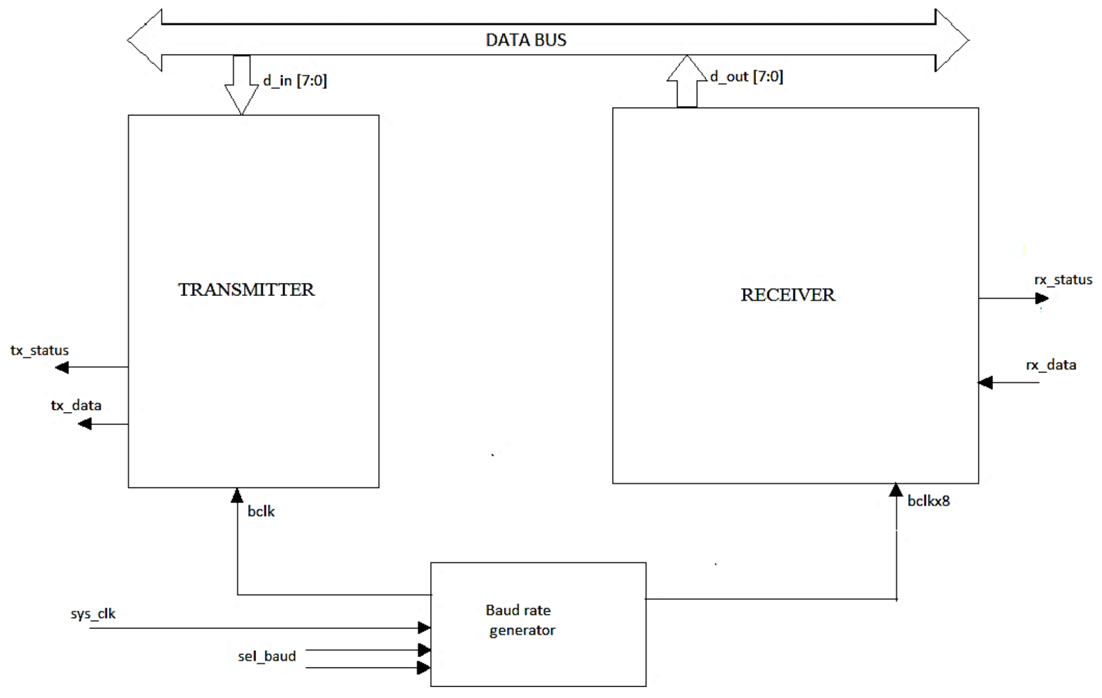


Fig 3: Architecture of UART

Transmitter Module

Block level micro-architecture of transmitter module is given below,

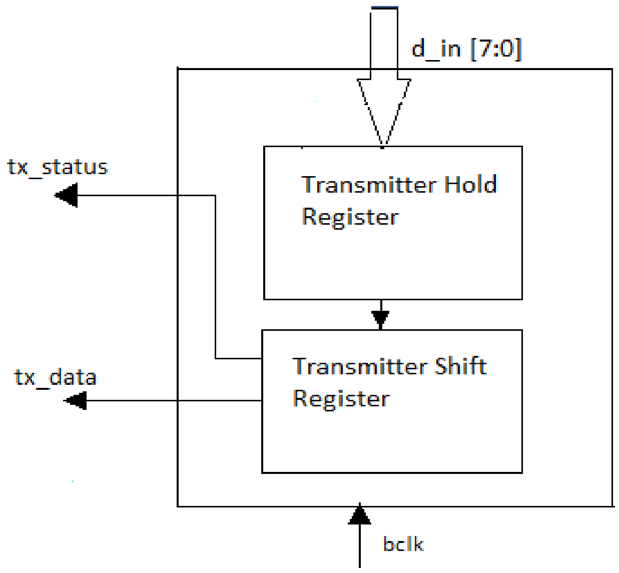


Fig 4: Micro-architecture of transmitter module.

All the operations in transmitter are with respect to clock which runs on multiple baud rates. The transmitter converts the 8-bit parallel data into serial data and adds start bit at start of the data frame and stop bit at the end of the data frame.

Pins	Width	I/O	Description
d_in	8 bits	Input	Input to the transmitter form data bus.
tx_status	1 bit	Output	Shows the status of the transmitter module.
tx_data	1 bit	Output	Sends data to receiver bit by bit.
bclk	1 bit	Input	Input clock to transmitter module from baud rategenerator.

Table 2: Pin description of transmitter module.

In transmitter, the 8-bit data from data bus is loaded to transmitter hold register (THR), it is an 8-bit register to store the data parallelly through d_in pin. Then after all the data is loaded in THR then data is loaded in Transmitter shift Register (TSR) parallelly. In TSR, start bitand stop bit is added to the data bits, transmitter shift register is of 10-bit size. TSR convert parallel data into serial form and then data is transmitted serially bit by bit through tx_out. While transmitting data LSB of the data bit is transmitted first and then remaining data is transmitted and last stop bit is transmitted to stop the transmission. The clock given to transmitter block isthe baud clock (bclk) on which the baud rate is set. Counter is used in TSR to count the data bits. Until and unless all the data is transmitted no other data can be stored in hold register.

Transmitter is designed based on the FSM given below,

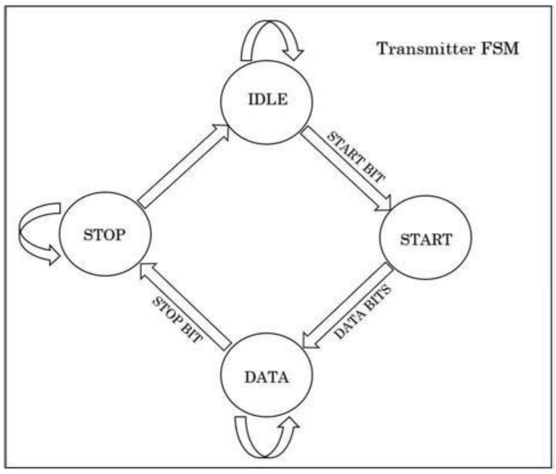


Fig 5: FSM for transmitter module.

Transmitter FSM have four states they are named as follows:

Idle State.

Start State.

Data State.

Stop State.

In transmitter FSM the first state is idle state in which transmitter remains in idle condition. It means that nothing will happen in this state. But when the desired input is given to this state, there is change of state that takes place. Here when valid start bit (logic ‘0’) is given as input then state changes from idle to start state means in idle state there is a detection of valid start bit. When invalid input is given in idle state then transmitter remains in same state i.e. idle state. Output of this state is logic ‘0’ for valid input (logic ‘0’).

When transmitter is in start state input is logic ‘0’ which is valid so there is change of state from the start state to data state. If input is not logic ‘0’ then transmitter goes in idle state. Start state will not change to data state until valid start bit is detected.

In data state transmitter sends frame sequence given to the input of transmitter and when complete frame sequence is transmitted then state changes to stop state otherwise transmitter will remain in same state until the complete transmission of frame sequence.

Stop state of transmitter will detect stop bit (logic ‘1’). If there will be any error in stop bit, then transmitter remains in stop state only otherwise it will again go to idle state to detect anotherstart bit. The above process continues till required number of bits will be transmitted.

Receiver Module

Block level micro-architecture of receiver module is given below,

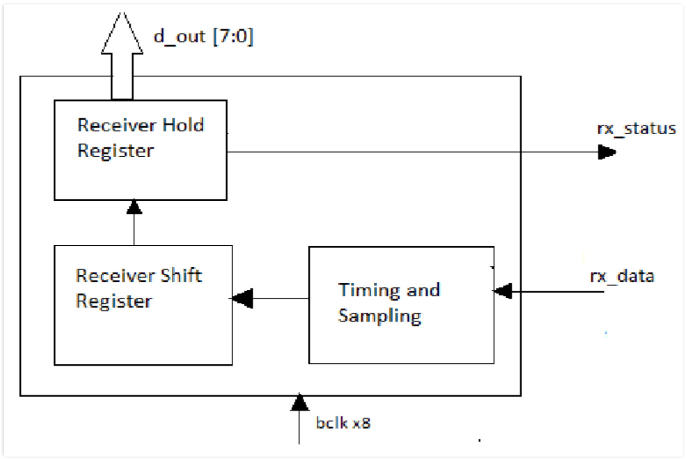


Fig. 6 Micro-architecture of receiver module.

The receiver module is more complex than the transmitter module. It runs 4 X baud rate faster than transmitter. The serial data is received from transmitter through rx_data pin 1-bit at a time serially. Then, rx_data pin jumps into logic 0 from logic 1 indicating beginning of the data frame. The start bit is identified by change in level from high to low level and stop bit by change in the level from low to high. After identifying start bit, all the data bits are sampled and counted using the bit counter generated in the receiver. The counter counts the positive edge of the clock at every riding edge of the clock 1 bit is counted. When count equals to 8 then it says all bits are received and stored in an 8 bit Receiver Shift Register (RSR) and the data bits are converted into parallel data. Then it sends the data bits to Receiver Hold Register (RHR) which is also an 8 bit register. RSR send only data bits to RHR, start bit and stop bit is not sent. Then the parallel data is sent to the data bus or peripheral device.

Pin	Width	I/O	Description
rx_data	1 bit	Input	Input data from transmitter.
bclkx8	1 bit	Input	Input clock to receiver from baud rate generator. Receiver clock is 8 times faster than transmitter clock.
rx_status	1 bit	Output	Shows status of receiver block.
d_out	8 bit	Output	Sends output data to PC or peripheral devices.

Table 3. Pin description for receiver module.

Receiver is designed based on FSM used as given below,

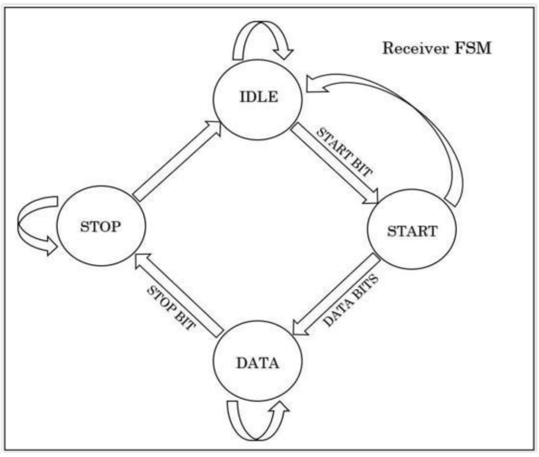


Fig 7: FSM for receiver module.

Receiver FSM also have same four states like transmitter they are named as follows:

Idle State.

Start State.

Data State.

Stop State.

In receiver FSM first state is idle state in which receiver remains in idle condition means nothing will going to happen in this state. But when desired input is given to this state then there is change of state takes place. Here when valid start bit (logic ‘0’) detected idle state changes from idle to start state. When wrong bit is received in idle state then receiver remains in idlestate. Output of this state is logic ‘0’ for valid input (logic ‘0’).

When receiver is in start state then it will not detect any bit because valid bit is already detected in idle state, so receiver simply changes it’s states from start to data state. In data state,if valid frame sequence received then there will be change in state from data to stop state otherwise receiver will remain in same state until the arrival of valid data.

Stop state of receiver will detect stop bit (logic ‘1’). If there will be any error in stop bit, then receiver remains in stop state only otherwise it will again go to idle state to detect another start bit. The above process continues till required number of bits will be received.

Both transmitter and receiver work on mealy machine means output depends on state as well asinput. There is change in output when any one of them changes (state or input).

Baud rate Generator Module and oversampling

Block level micro-architecture of baud rate generator is given below,

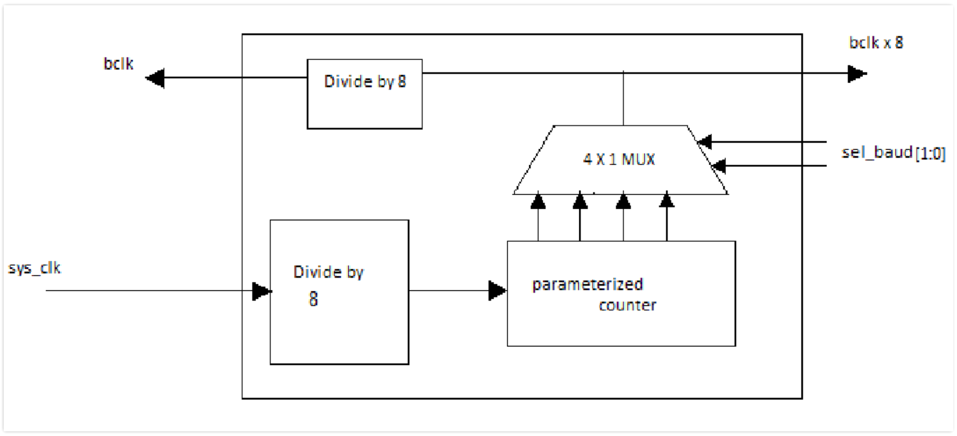


Fig 8: Micro-architecture of baud rate generator.

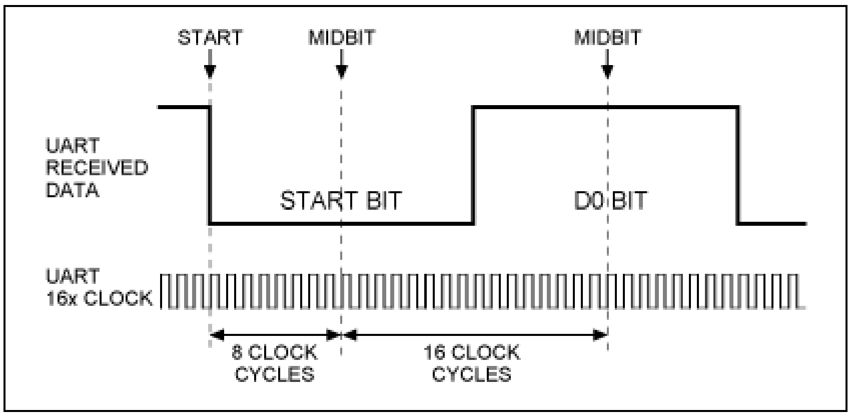
Baud rate generator is also used to divide frequency. By using given clock frequency and requested baud rate we can calculate the frequency factor. In baud rate generator, the input tothe generator is system clock which is available on FPGA board in our project we used 100 MHz. The clock is first divided by 8 by using the clock divider to get the data bits on the positive edge of the clock. Then a parameterized counter is used to count the data bits and the counter runs on the divided clock. The output of the parameterized counter is divided into different divisors of different baud rates to run the process on these baud rates. The divisor is found by the formula proposed as,

Divisor = sys. Clock / (8 x baud rate)

Pin	Width	I/O	Description
sys_clk	1 bit	Input	Input clock to baud rate generator from FPGA board.
sel_baud	2 bits	Input	Select lines to select different baud rates.
bclk	1 bit	Output	Output clock given to transmitter module.
bclk x 8	1 bit	Output	Output clock given to receiver module.

Table 4. Pin description for baud rate generator.

A specific baud rate is selected from a 4x1 multiplexer according to the select lines. Then the output of this multiplexer is given to transmitter and receiver on which they run. The receiver baud clock is 8 times faster than transmitter baud clock.To get this difference in clock speeds the output of the multiplexer is first given to divide by 8 block and then given to transmitter block and the output of the multiplexer is directly given to receiver.



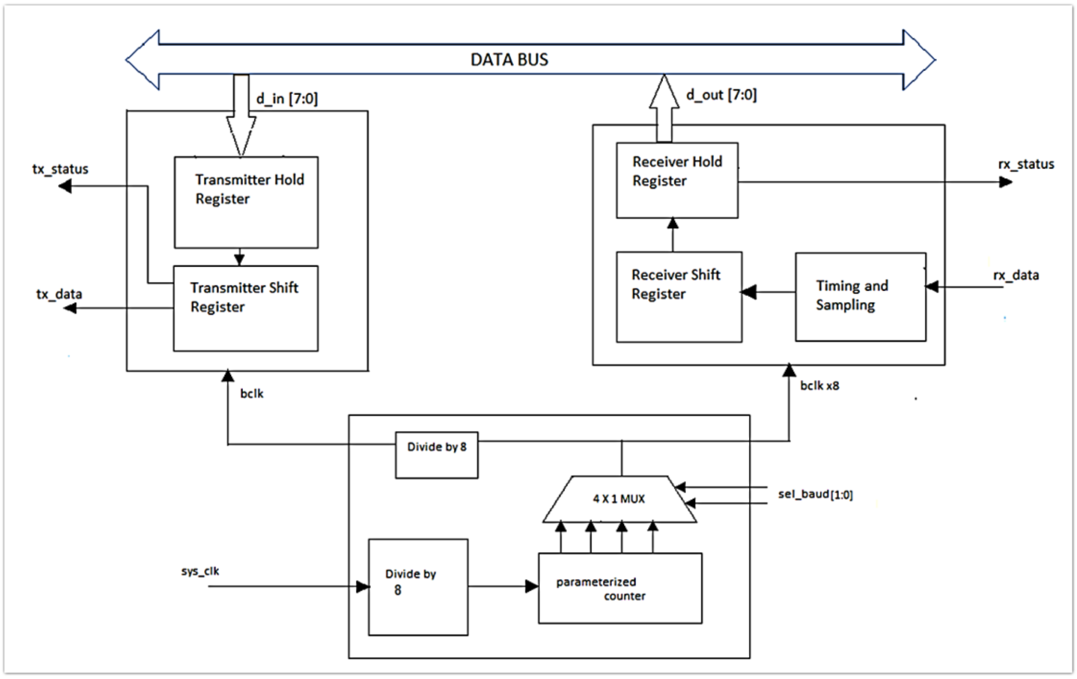


Fig 8: Micro-architecture UART.

Lab Task

Design a full-duplex UART protocol using System Verilog. Verify your test cases by writing testbench.

Specifications:

Following are the specification extracted and implemented.

1. Mode: Full Duplex.

2. Data Frame Sequence: 8 bits

3. Primary Frequency: 100MHz

5. 1 start bit, 1 stop bit. No parity bit.

6. No. of flags: 2.

FPGA Prototyping & Verification:

Use Tera Term 5 software to test your UART Transceiver.

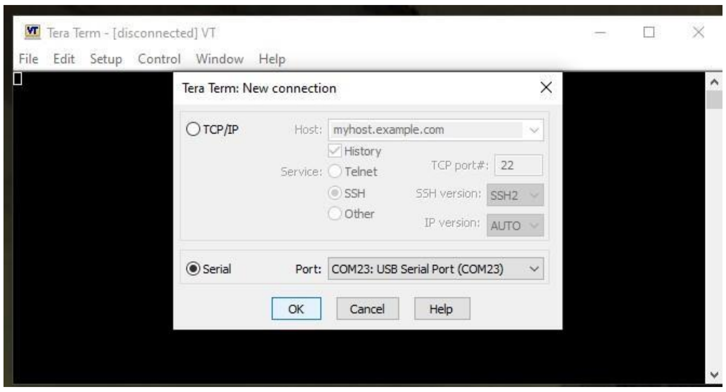
Link: https://github.com/TeraTermProject/teraterm/releases/tag/v5.2

Tx Port on FPGA: C4

Rx Port on FPGA: D4

Steps:

1. choose serial Port



Tera Term - [disconnected] VT

File Edit Setup Control Window Help

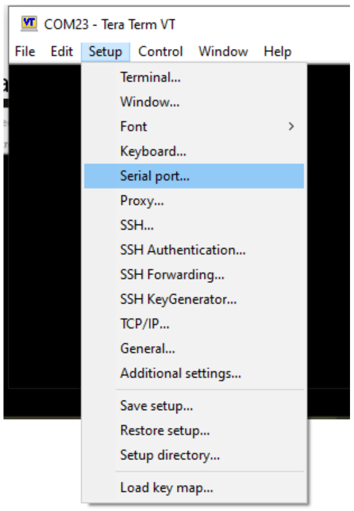
Tera Term: New connection

( ) TCP/IP Host: myhost.example.com
[✓] History
Service: ( ) Telnet TCP port#: 22
(•) SSH SSH version: SSH2
( ) Other IP version: AUTO

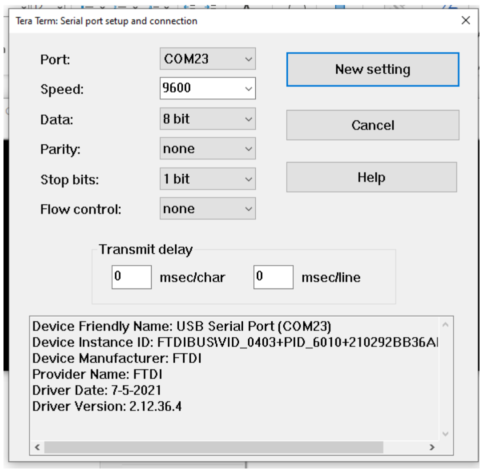
(•) Serial Port: COM23: USB Serial Port (COM23)

OK Cancel Help

2. Edit Serial Port



3. Choose Your Settings.



Port: COM23
Speed: 9600
Data: 8 bit
Parity: none
Stop bits: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

Device Friendly Name: USB Serial Port (COM23)
Device Instance ID: FTDIBUS\VID_0403+PID_6010+210292BB36AI
Device Manufacturer: FTDI
Provider Name: FTDI
Driver Date: 7-5-2021
Driver Version: 2.12.36.4