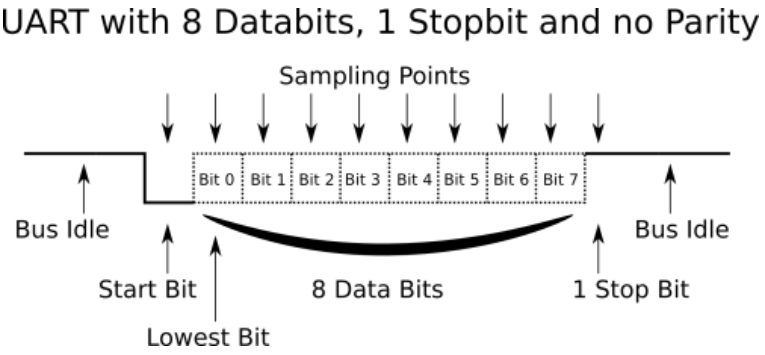
User Guide for FPGA UART

The provided code has two major parts: MATLAB Script and Verilog code.

Important notice: Before you start to modify the code, please read all the comments in the code and try to understand how it works.

❖ UART



The Universal Asynchronous Receiver/Transmitter (UART) controller is the key component of the serial communications subsystem of a computer. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes.

In this lab's configuration, the BAUD rate is 9600, which means each data bit will hold for 1/9600s on the data bus. Each sending operation has two starting bits (1, 0), 8 data bits (LSB sends first), and one ending bit (1). The typical UART BAUD rate includes 9600, 19200, 38400, etc. The higher the BAUD rate, the faster the transmission, but the error rate is also increasing. If you want to use a higher BAUD rate, you can change the related variables both in MATLAB script and Verilog code. **Make sure that the BAUD rate on both sides are the same!**

❖ MATLAB Script

In this Lab, we will use MATLAB to control of the UART on PC side.

For most of the PCs in PHO115, the UART port is on 'COM5', while several others has UART port on 'COM4'. You can go the device manager to find out which port you should use and change it accordingly in the MATLAB script. The port name and BAUD rate is set during the definition of the serial port in function **serial()**.

The provided MATLAB script can perform 3 different communication modes:

1. Send one input image data at a time and read it back immediately. It is strongly suggested that you perform this test first to determine the onboard UART's functionality. When performing this operation**, you need to set *top_direct_send.v* as the top module in Xilinx ISE and generate the bit file**.
2. Send a series of data generated by PC first and read back all data at once. When performing this operation**, you need to set *top.v* as the top module in Xilinx ISE and generate the bit file**. This part will help you test the on-chip memory as all the received data will be written on to the Block RAM.
3. Send all the image data first and read back the processed image data. **You need to set the *top.v* as the top module in Xilinx and generate the .bit file**.

   As our PC is working on a much higher frequency comparing with UART, **between two consecutive sending operations a delay is required**. The delay can be inserted by the following function:

   pause(num_of_secs);

❖ Verilog code

1. The Verilog code has the following major parts:

   receiver.v : UART receiver module, get one 8-bit data at a time from UART RX port.

   transmitter.v : UART transmitter module, send one 8-bit data at a time from UART TX port.

   receiver_sampler.v : Sample the rx_data line 10 times faster than UART frequency, and send the result to receiver. Implemented as an simple error correction mechanism to deal with the clock skew.

   clk_div.v : generate 9600Hz clock signal for UART and 96000Hz clock signal for sampling.

   PLL_9_6M.v : Phase lock loop that generate clock signal for UART.

   Data_collect.v : collect the pre-defined number of data from PC and store them in memory.

   Data_send.v : get the processed data from memory and send them back to PC.

   Data_process.v : currently implemented as send the received data directly back to PC. **For this Lab, you need to include your data processing design under this module.**

2. There are two top modules in this set of code: top_direct_send.v and top.v.

In top_direct_send.v, the transmitter and receiver are directly connected to each other. So each time a data is received from PC, FPGA will send it back immediately. This simple implementation will help you determine the functionality of UART.

In top.v, it will first collect a certain number of data from PC and write to memory one by one. After that, it will read data from memory and send those back to PC.

3. Parameters in Verilog.
   In the top module, there are four parameters:
   NUM_DATA: Total number of data you want to process. Related with the input image size.
   CLK_RATE: The output clock rate from PLL, do no change this.
   BAUD_RATE: Baud rate, pre-set as 9600. This should be the same as your baud rate in MATLAB.
   SAMPLE_RATE: UART_rx singal sampling rate. Suggested to keep this value.

4. FPGA GPIO define:

There are two pre-defined ucf file for the two different top_modules. **Before you add them to your design, you need to set the related top file as the top module.**

The received data is displayed on the 8 on board LEDs(for top.ucf, LD7 is connected to a different signal). The rst signal is connect to T10 (SW0). **You need to give a reset signal every time before you initializing the MATLAB script on PC.**

There are two LEDs that near the UART connector, LD10 and LD11. This two LEDs are hardwired to UART's rx and tx port accordingly. By observing this two LEDs, you can determine if data has been received by or sent from FPGA.

❖ Image Processing

In image processing, a kernel, convolution matrix, or mask is a small matrix useful for blurring, sharpening, embossing, edge-detection, and more. This is accomplished by means of convolution between a kernel and an image.

Use your designed 3x3 systolic array to perform the convolution of two different sized images. Each convolution will generate one data point in the new image. For the edge points, there are no operation needed. But you need to design a sliding window to scan through all the inner data points in image.

The convolution operation should be Gaussian blur and Edge detection. Please refer to the link below to get detailed information.

https://en.wikipedia.org/wiki/Kernel_(image_processing)