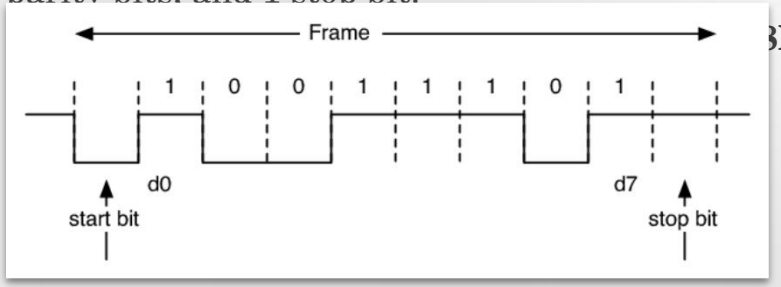
ECCS 1721 Digital Logic
Serial Communication

SERIAL COMMUNICATION

- So far our hardware descriptions have had inputs of 16 bits or less.
  - Modern computers and devices have word sizes of 32 or 64 bits.
- Almost any practical application requires the transmission of an arbitrary amount of data.
  - If you want to transfer a gigabyte of data to your friend quickly, how do you do it?
    - Having one wire for each bit means you would have around 1 billion wires, which is impractical from both a physical and financial point of view.
- Many different *serial* communication protocols have been invented to address the shortcomings of *parallel* communication.
  - Here, *serial* refers to bits being transmitted *in series*, i.e., one after another, or *sequentially*.
- Serial communication is used everywhere.
  - Almost all modern long-distance networking is serial.
  - USB stands for Universal Serial Bus – any USB device uses serial communication.
  - Hard drives and some solid state drives use SATA, or Serial Advanced

THE UART PROTOCOL

- UART, or Universal Asynchronous Receiver/Transmitter, is a common serial communication protocol.
- A UART *frame*, the smallest unit of data transmitted over a UART connection, has the following parameters:
  - Baud rate – a previously agreed upon rate of data transmission
  - Number of data bits
  - Number of parity bits
  - Number of stop bits
- For this lab, we will use a baud rate of 9600 (the most common), 8 data bits (i.e., one byte), no parity bits, and 1 stop bit.

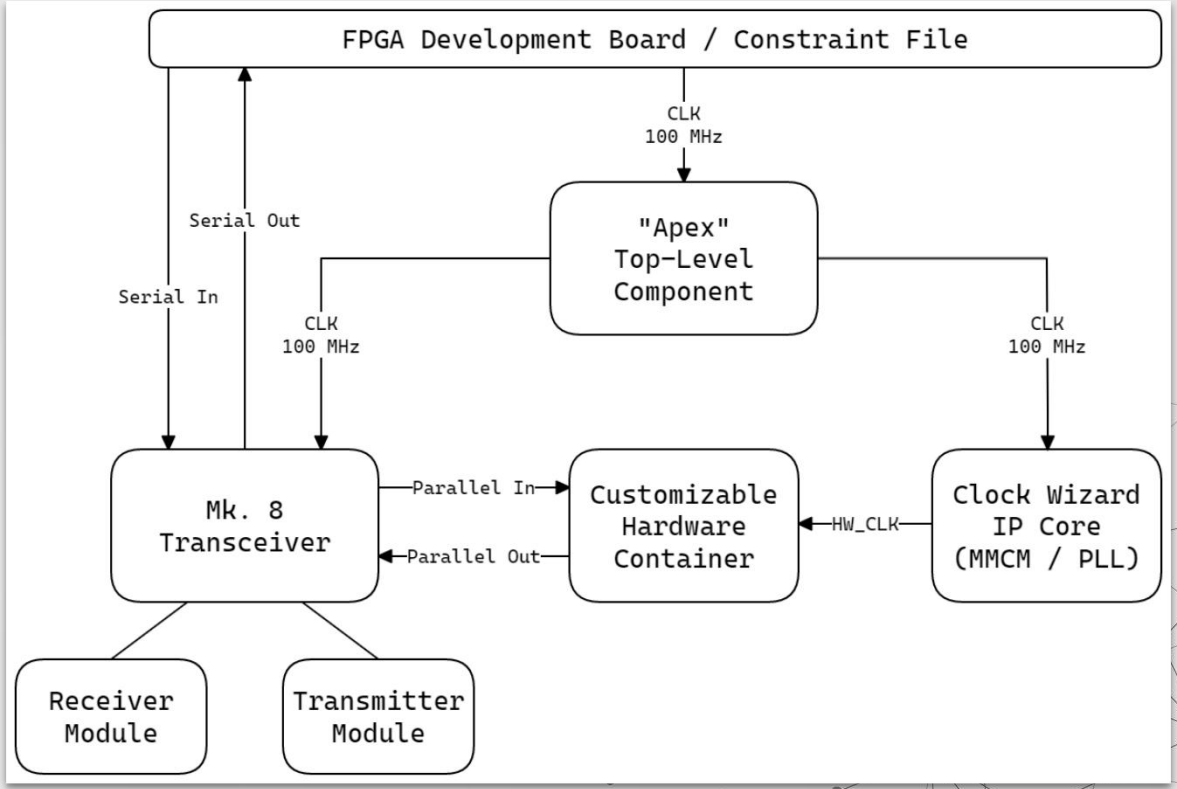


8N1, for 8 data bits, no parity bits, and 1 stop bit.

https://www.researchgate.net/figure/8N1-Data-Transmission_fig1_4253336

SERIAL TRANSCEIVER

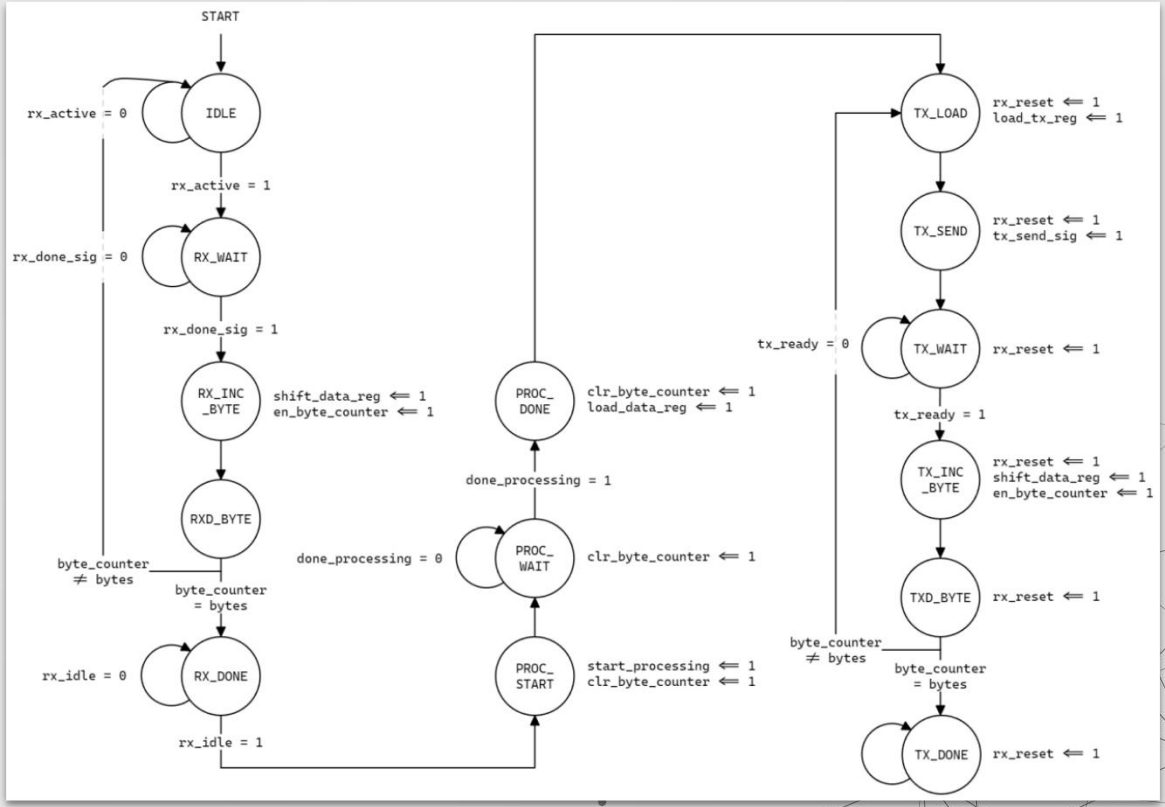
- The serial transceiver we will use for this lab is structured as shown on the right.
- The FPGA board has dedicated TX (transmitter) and RX (receiver) pins attached to the micro-USB port.
    - These are routed to the actual transceiver module.
- The customizable hardware container will contain the device under test, which in this lab will be a priority encoder, decoder, or barrel shifter.
- The clock wizard is an IP (intellectual property) block which provides a cleanly divided clock using the MMCM (mixed-mode clock manager) and PLLs (phase-locked loops) on the FPGA.



https://github.com/ALUminaries/Serial-Transceiver

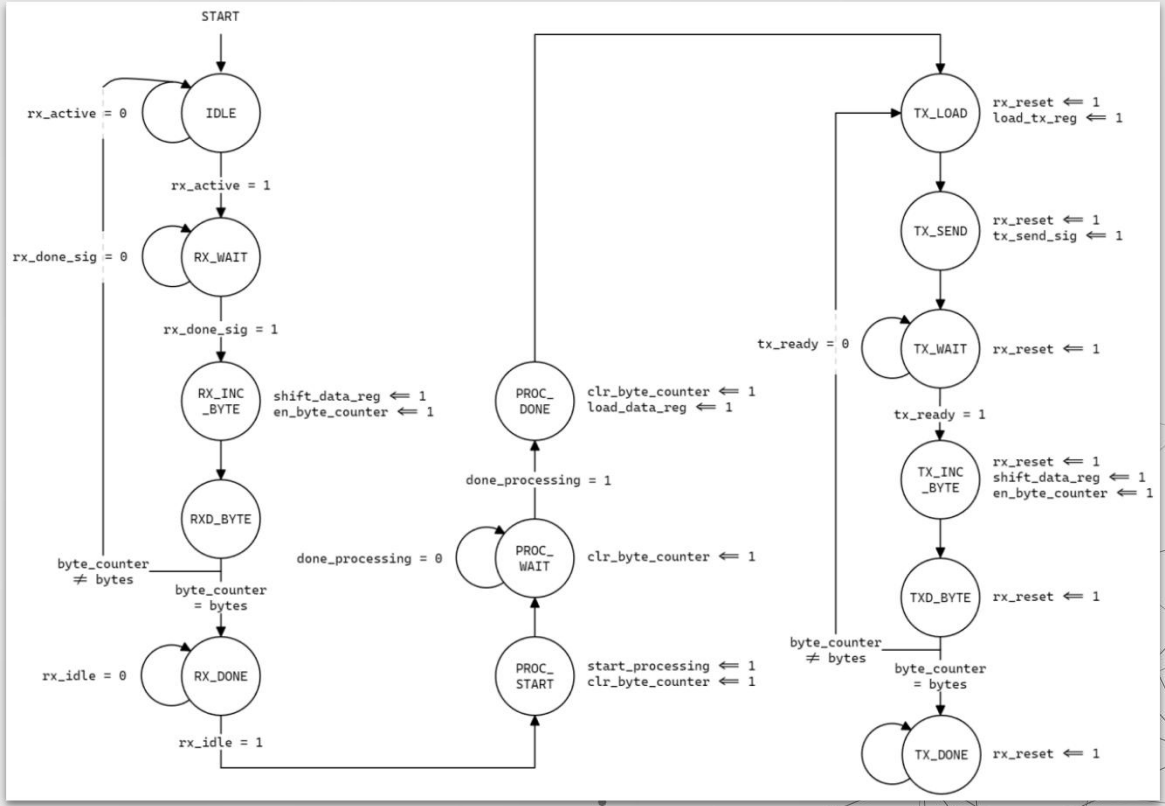
TRANSCEIVER MODULE

- The transceiver submodule has three stages:
  1. Receiving
  2. Processing
  3. Transmitting
- In the receiving stage, it will wait until a start bit is received, then prepare to receive one byte.
- The hardware will accept a predetermined number of bytes before continuing to the processing stage.
- During the processing stage, which is typically very fast, the transceiver will transfer the data in parallel to the hardware (ex. 2LPE) and await a done signal from the hardware.
- At the transmitting stage, the hardware will loop through and



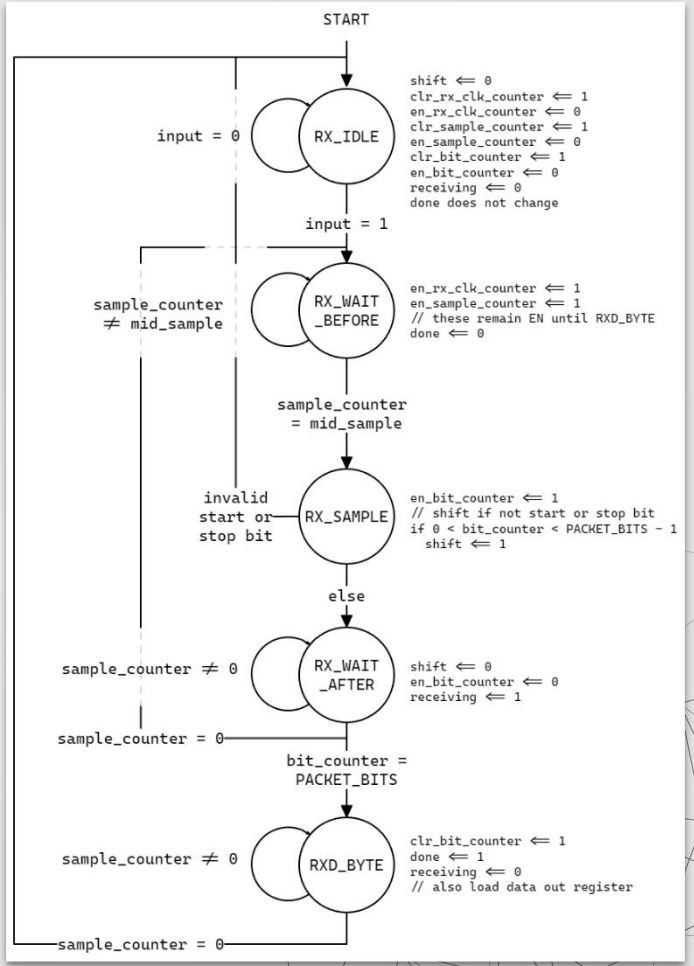
TRANSCEIVER MODULE (CONTINUED)

- Note: you do not need to memorize this diagram, only to have an idea of what is happening inside the transceiver.
- You will notice that this state machine has only byte-level granularity; it does not handle individual bits.
  - Each byte is handled by the actual receiver and transmitter submodules.
  - The transceiver module is more like a manager for both of them.



RECEIVER SUBMODULE

- The receiver has an input (from the RX pin).
- It reads this input four times per second, and samples it at the middle to ensure the result is stable.
    - This helps to ensure timing, since UART is asynchronous.
- Again note that this submodule handles one byte per full loop.



TRANSMITTER SUBMODULE

- The transmitter has an output (the TX pin).
- It also has a buffer (`bit_buffer`) which the state machine writes to synchronously.
    - The output takes the value of this buffer asynchronously.
- Like the receiver, this submodule handles one byte per full loop.