

ECE 551 Design Challenge-Designing a UART

What is a UART?

A universal asynchronous receiver transmitter (UART) is basically a parallel to serial data transmitter and a serial to parallel data receiver. The asynchronous part refers to the fact that the clock for the UART does not have to be synchronized to either the transmitting or receiving systems' clocks.

Why is a UART necessary for a computer modem?

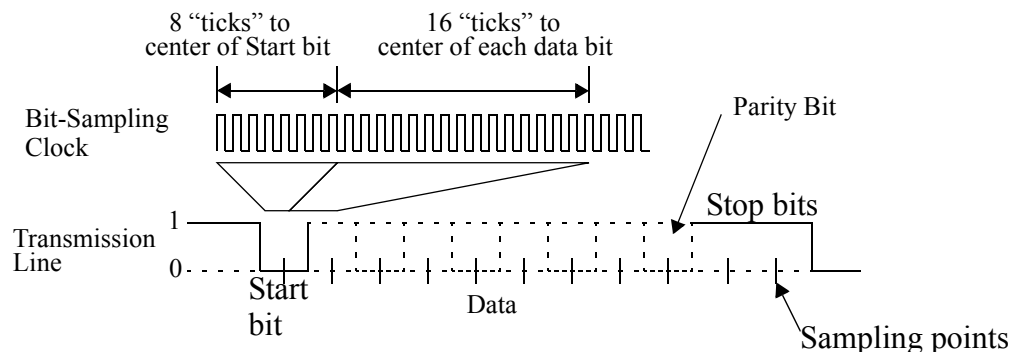
A modem is composed of two basic parts, the UART and the analog modulator-demodulator, which is where the term (MoDem) comes from. During transmission of data the UART is responsible for taking the data from the host machine and turning it into a serial bit stream that can be sent by the modem over the phone line. During reception of data the UART takes the serial data from the modem and converts it into parallel data that can be used by the host machine.

How does a UART work?

Standard UARTs transmit and receive data in 11-bit packets, of which 8-bits are for the data, there is one bit at the beginning of the packet called the Start bit, which is always a logic 0, and 2 bits at the end called the Stop bits, which are always a logic 1.

As a Receiver

The two basic parts of the UART receiver are a bit-sampling clock, which runs at least 16 times faster than the bit rate, and a bit counter. The idle state of the transmission line is at a logic high level. When data is being sent the transmission line drops because of the start bit. The UART senses this change and counts off 8 increments of the bit-sampling clock to find the middle of the start bit (assuming this clock is running 16 times faster than the bit-rate). Then the bit-sampling clock is reset and every time the bit-sampling clock reaches a count of 16 the transmission line is sampled and the bit counter is incremented. When all eight bits have been collected, the UART looks for the stop bits and then sends the data to the host machine.



As a Transmitter

The UART Transmitter is composed of three basic parts: 1) a parallel-load shift register, 2) a transmission line flip-flop, and 3) logic that generates a "register-empty" flag. Data is loaded in

parallel fashion into the shift register, and then shifted out serially to the Line flip-flop, which holds the transmitted data for one clock cycle. Remember you must also append the Start and Stop bits to the data in the shift register. As bits leave the shift register, 0's are shifted into the "empty" slots of the shift-register. The flag logic should detect when the shift register has all zeros (i.e. the Stop bits have left the shift register), and set a flag that the transmission is complete. Upon completion of a successful transmission, the shift register should then load in the next byte of data.

Specifications for Your Design

Minimum Requirements

Your challenge is to design a basic UART that is capable of transmitting and receiving data at 300, 600, and 900 bps. Each packet should include 8-bits of data, 1 Start bit, and 2 Stop bits. Both the transmitting and receiving registers should be at least double-buffered. The mask layout must fit into an area of 1700x900 um. You can assume that you have an external system clock running at 1.8432 MHz, but any division of this clock must be done internal to your design.

Optional Stuff

Here are some optional add ons to the UART that makes it more practical to use. Make sure you get the basic UART design nailed down before you add on these components:

- Triple, Quadruple, or higher levels of buffering
- Parity checking and setting (for transmitting 7-bit or smaller codes)
- Reception error detection
- RS232 Standard control signals (CTS, RTS, ...)