➢ Introduction

Universal asynchronous receiver and transmitter (UART) is a circuit that sends parallel data through a serial line. UARTs are frequently used in conjunction with the EIA (Electronic Industries Alliance) RS-232 standard, which specifies the electrical, mechanical, functional, and procedural characteristics of two data communication equipment. Because the voltage level defined in RS-232 is different from that of FPGA I/O, a voltage converter chip is needed between a serial port and an FPGA's I/O pins.

The S3 board has a RS-232 port with the standard nine-pin connector. The board contains the necessary voltage converter chip and configures the various RS-232's control signals to automatically generate acknowledgment for the PC's serial port. A standard straight-through serial cable can be used to connect the S3 board and PC's serial port. The S3 board basically handles the RS-232 standard and we only need to concentrate on the design of the UART circuit.

A UART includes a transmitter and a receiver. The transmitter is essentially a special shift register that loads data in parallel and then shifts it out bit by bit at a specific rate. The receiver, on the other hand, shifts in data bit by bit and then reassembles the data. The serial line is '1' when it is idle. The transmission starts with a *start bit*, which is '0', followed by *data bits* and an optional *parity bit*, and ends with *stop bits*, which are '1'. The number of data bits can be 6, 7, or 8. The optional parity bit is used for error detection. For odd parity, it is set to '0' when the data bits have an odd number of 1's. For even parity, it is set to '0' when the data bits have an even number of 1's. The number of stop bits can be 1, 1.5, or 2.

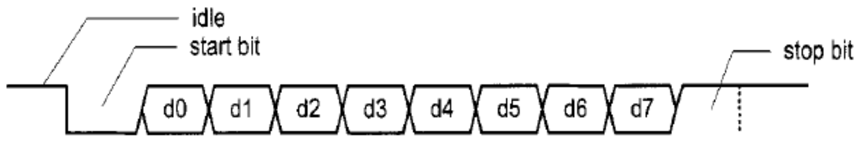

Figure 7.1 Transmission of a byte.

The transmission with 8 data bits, no parity, and 1 stop bit is shown in Figure 7.1. Note that the LSB of the data word is transmitted first.

No clock information is conveyed through the serial line. Before the transmission starts, the transmitter and receiver must agree on a set of parameters in advance, which include the baud rate (i.e., number of bits per second), the number of data bits and stop bits, and use of the parity bit. The commonly used baud rates are 2400, 4800, 9600, and 19,200 bauds.

➢ UART Receiving Subsystem

Since no clock information is conveyed from the transmitted signal, the receiver can retrieve the data bits only by using the predetermined parameters. We use an *oversampling scheme* to estimate the middle points of transmitted bits and then retrieve them at these points accordingly.

- Oversampling Procedure:

The most commonly used sampling rate is 16 times the baud rate, which means that each serial bit is sampled 16 times. Assume that the communication uses N data bits and M stop bits. The oversampling scheme works as follows:

1. Wait until the incoming signal becomes '0', the beginning of the start bit, and then start the sampling tick counter.
2. When the counter reaches 7, the incoming signal reaches the middle point of the start bit. Clear the counter to 0 and restart.
3. When the counter reaches 15, the incoming signal progresses for one bit and reaches the middle of the first data bit. Retrieve its value, shift it into a register, and restart the counter.
4. Repeat step 3 $N-1$ more times to retrieve the remaining data bits.
5. If the optional parity bit is used, repeat step 3 one time to obtain the parity bit.
6. Repeat step 3 M more times to obtain the stop bits.

The oversampling scheme basically performs the function of a clock signal. Instead of using the rising edge to indicate when the input signal is valid, it utilizes sampling ticks to estimate the middle point of each bit. While the receiver has no information about the exact onset time of the start bit, the estimation can be off by at most $\frac{1}{16}$. The subsequent data bit retrievals are off by at most $\frac{1}{16}$ from the middle point as well. Because of the oversampling, the baud rate can only be a small fraction of the system clock rate, and thus this scheme is not appropriate for a high data rate.

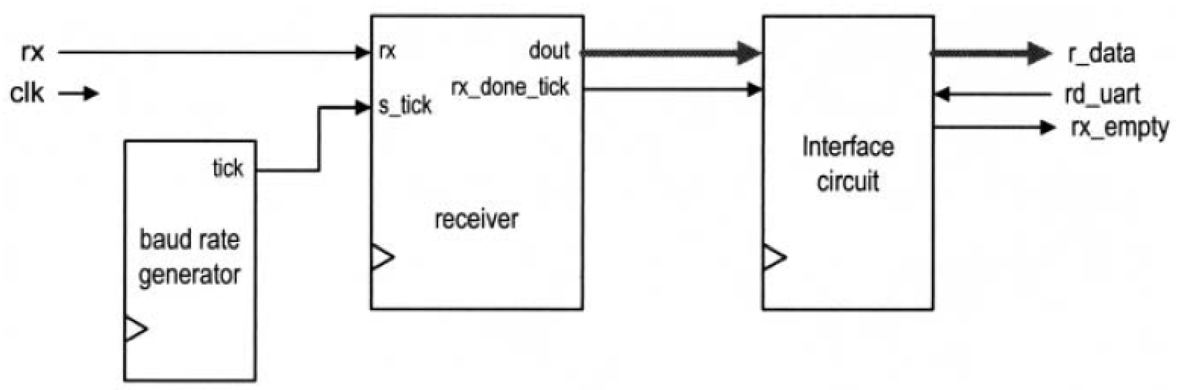

Figure 7.2 Conceptual block diagram of a UART receiving subsystem.

The conceptual block diagram of a UART receiving subsystem is shown in Figure 7.2. It consists of three major components:

- *UART receiver*: the circuit to obtain the data word via oversampling
- *Baud rate generator*: the circuit to generate the sampling ticks
- *Interface circuit*: the circuit that provides buffer and status between the UART receiver and the system that uses the UART

- Baud rate generator

The baud rate generator generates a sampling signal whose frequency is exactly 16 times the UART's designated baud rate.

For the 19,200 baud rate, the sampling rate has to be 307,200 (i.e., 19,200∗16) ticks per second. Since the system clock rate is 50 MHz, the baud rate generator needs a mod-163 (i.e., $\frac{50*10^6}{307200}$) counter, in which the one-clock-cycle tick is asserted once every 163 clock cycles.

- Interface Circuit

In a large system, a UART is usually a peripheral circuit for serial data transfer. The main system checks its status periodically to retrieve and process the received word. The receiver's interface circuit has two functions. First, it provides a mechanism to signal the availability of a *new* word and to prevent the received word from being retrieved multiple times. Second, it can provide buffer space between the receiver and the main system.

➢ UART Transmitting Subsystem

The organization of a UART transmitting subsystem is similar to that of the receiving subsystem. It consists of a UART transmitter, baud rate generator, and interface circuit.

The UART transmitter is essentially a shift register that shifts out data bits at a specific rate. The rate can be controlled by one-clock-cycle enable ticks generated by the baud rate generator. Because no oversampling is involved, the frequency of the ticks is 16 times slower than that of the UART receiver. Instead of introducing a new counter, the UART transmitter usually shares the baud rate generator of the UART receiver and uses an internal counter to keep track of the number of enable ticks. A bit is shifted out every 16 enable ticks.

- Complete UART Core

By combining the receiving and transmitting subsystems, we can construct the complete UART core.

- We need to design the following full-featured UART:

The alternative to the customized UART is to include all features in design and to dynamically configure the UART as needed. Consider a full-featured UART that uses additional input signals to specify the baud rate, type of parity bit, and the numbers of data bits and stop bits. The UART also includes an error signal. In addition to the I/O signals of the `uart_top` design in Listing 7.4, the following signals are required:

- `bd_rate`: 2-bit input signal specifying the baud rate, which can be 1200, 2400, 4800, or 9600 baud
- `d_num`: 1-bit input signal specifying the number of data bits, which can be 7 or 8
- `s_num`: 1-bit input signal specifying the number of stop bits, which can be 1 or 2
- `par`: 2-bit input signal specifying the desired parity scheme, which can be no parity, even parity, or odd parity
- `err`: 3-bit output signal in which the bits indicate the existence of the parity error, frame error, and data overrun error