UART (UNIVERSAL ASYNCRONOUS RECEIVER & TRANSMITTER)

A 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.

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 I '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.

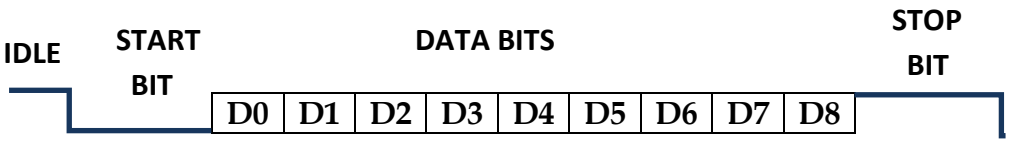

Figure: Transmission with 8 data bits, no parity, and 1 stop bit.

The transmission is started from LSB of the data word. **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.

Design of a UART with a 19,200 baud rate, 8 data bits, 1 stop bit, and no parity bit.

1. 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. The oversampling scheme is used to estimate the middle points of transmitted bits and retrieve these points accordingly.

Oversampling

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 n more times to obtain the stop bits.

The oversampling scheme 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 &. The subsequent data bit retrievals are off by at most & from the middle point as well. Because of the oversampling, the baud rate can be only a small fraction of the system clock rate, and thus this scheme is not appropriate for a high data rate.

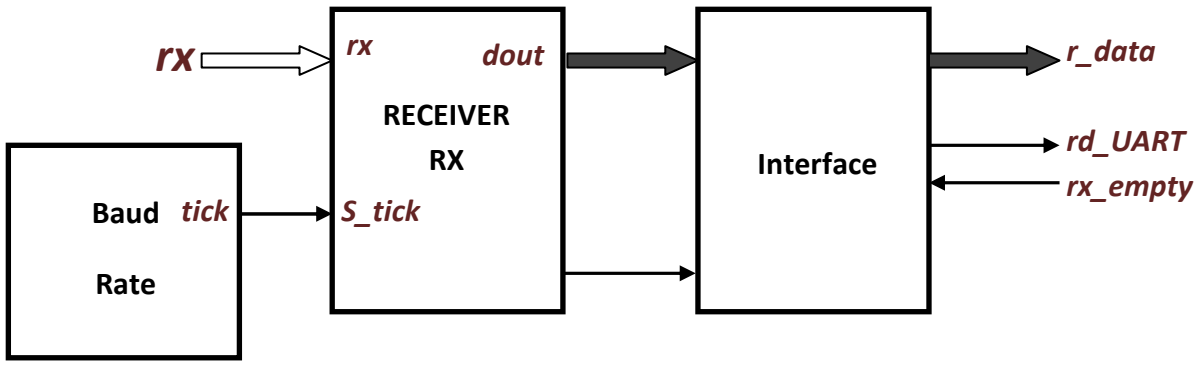

Figure: The conceptual block diagram of a UART receiving subsystem .

UART Receiving Subsystem consists of three major components:

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

UART receiver

The DBIT constant indicates the number of data bits, and the SB-TICK constant indicates the number of ticks needed for the stop bits, which is 16, 24, and 32 for 1, 1.5, and 2 stop bits, respectively. DBIT and SB-TICK are assigned to 8 and 16 in this design.

The chart follows oversampling and includes three major states, **start, data,** and **stop**, which represent the processing of the start bit, data bits, and stop bit. The s-tick signal is the enable tick from the baud rate generator and there are 16 ticks in a bit interval. Note that the FSMD stays in the same state unless the s-tick signal is asserted. There are two counters, represented by the s and n registers. The s register keeps track of the number of sampling ticks and counts to 7 in the start state, to 15 in the data state, and to SB-TICK in the stop state. The n register keeps track of the number of data bits received in the data state. The retrieved bits are shifted into and reassembled in the b register. A status signal, rx-done-tick, is included. It is asserted for one clock cycle after the receiving process is completed.

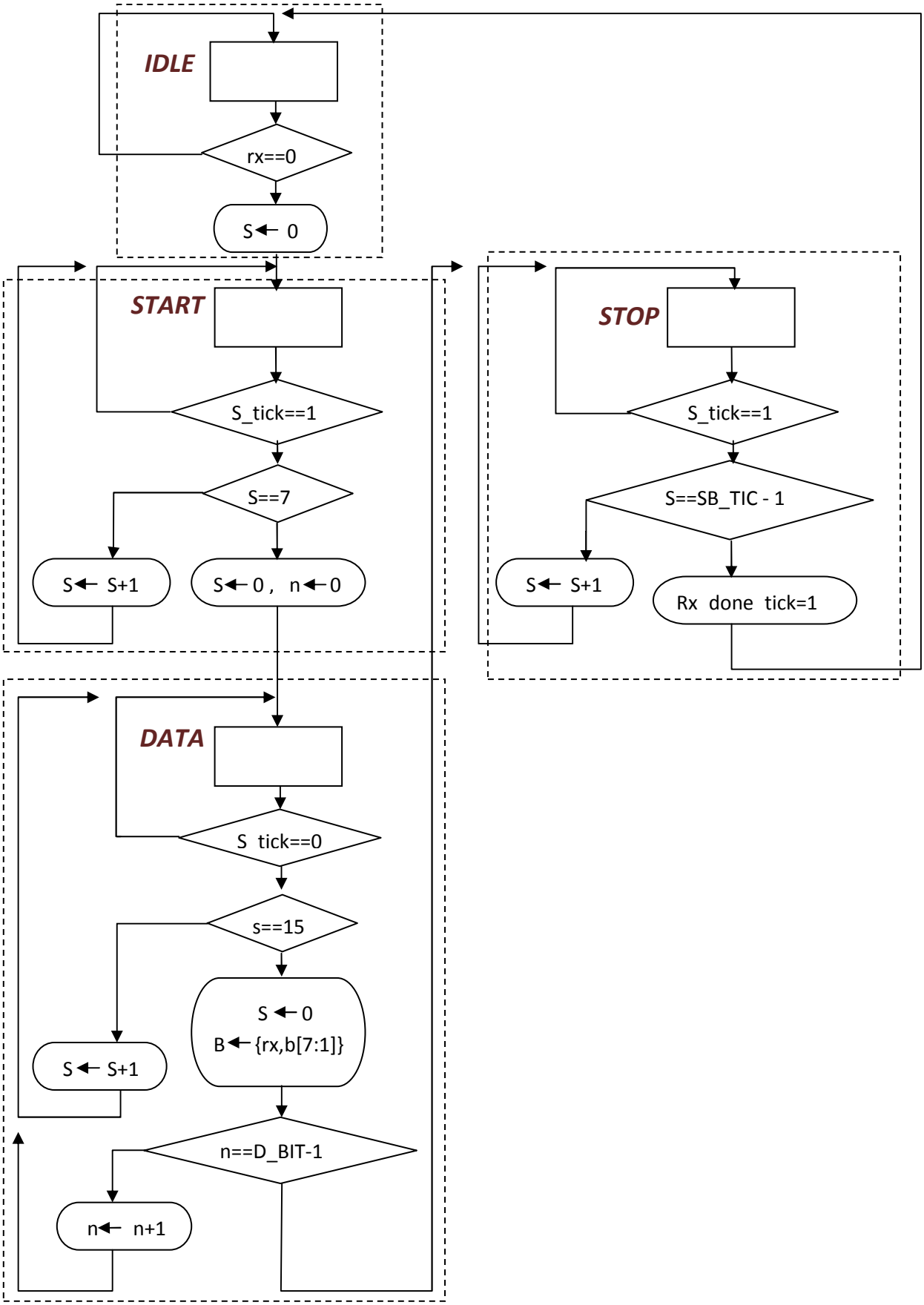
IDLE
rx==0
S ← 0
START
S_tick==1
S==7
S ← S+1
S ← 0 , n ← 0
STOP
S_tick==1
S==SB_TIC - 1
S ← S+1
Rx done tick=1
DATA
S tick==0
s==15
S ← 0
B ← {rx,b[7:1]}
S ← S+1
n==D_BIT-1
n ← n+1

Figure:

Baud rate generator

The baud rate generator generates a sampling signal whose frequency is exactly 16 times the UART's designated baud rate. To avoid creating a new clock domain and violating the synchronous design principle, the sampling signal should function as enable ticks rather than the clock signal to the UART receiver.

For the 19,200 baud rate, the sampling rate has to be 307,200 (i.e., 19,200*16) ticks per second. Assuming that the system clock rate is 50 MHz(if using Basys2 FPGA kit), the baud rate generator needs a mod-163 (i.e., counter = clock frequency/sampling-frequency = (5*10^6)/ 307,200) counter, in which a one-clock-cycle tick is asserted once every 163 clock cycles.

UART Transmitter

The UART transmitter operation is similar to a receiver and it 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.