University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

EECS 150 Spring 2000

Lab 7
UART Design

1 Objective

In this lab you will design, simulate, implement, and test the receiver portion of a *universal asynchronous receiver / transmitter* circuit (UART). This circuit will be useful later for your course project, but more importantly, through its design you will gain more experience with the FPGA device. In particular, you will have a chance to use some "MSI" level library components.

2 Functional Specification

UARTs are used for communication between two devices. For instance, UARTs are used to connect terminals to computers. UARTs are also used in systems using MIDI. They provide a means to send data with a minimum of wires. The data is sent bit-serially, and no clock signal is sent along with it. The primary function of a UART is parallel-to-serial conversion when transmitting, and serial-to-parallel conversion when receiving. The fact that a clock is not transmitted with the data complicates the design of a UART. The two systems (sender and receiver) have separate, unsynchronized, clock signals. Although the two clocks will have the same frequency, they will not have the same phase. Part of a UART's function, and the tricky part, is to "sample" the serial input at just the right time to reliably capture the bit stream. This is done by using a high-speed clock to sample the bit stream multiple times per data bit.

In our application the bit transfer rate, or *baud rate*, is 31.25kHz. This rate is the standard for MIDI devices and will be useful later for the course project. Your FPGA board is equipped with a 16MHz crystal oscillator. A 31.25kHz clock signal could be generated by dividing the crystal oscillator frequency by 512. We do not actually need that frequency for this lab. The frequency we need is 8 times the MIDI baud rate: $8 \times (31.25\text{kHz}) = 250\text{kHz}$.

Figure 1 shows a functional block diagram of the UART receiver that you will design. Bit-serial data is received on the `SERIAL-IN.H` input. When one byte of data has been received, it is output to the `D` output bus, and the output control signal `DRDY.H` is asserted for one clock period. The block is clocked with a frequency 8 times the baud rate, in this case 250kHz. The reset signal `RESET.H` and output enable `OE.H` are optional.

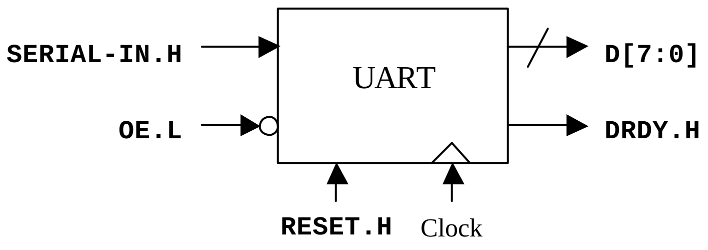

Figure 1: UART symbol.

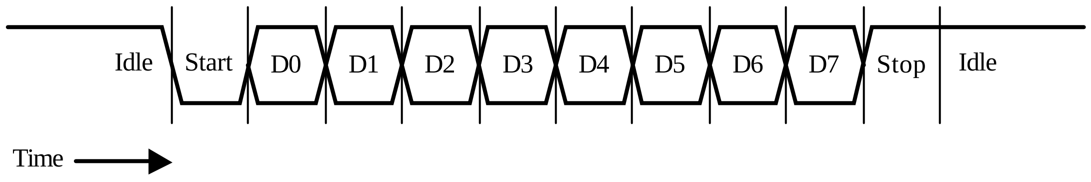

Figure 2: A MIDI transmission character.

Data is transferred one byte at a time to the receiver using the format shown in Figure 2. The transmission character is composed of an 8-bit data byte, sent LSB first, preceded by a start bit (`LOW`) and followed by a stop bit (`HIGH`). When no character is being transmitted, the line is idle (`HIGH`). The line need not go idle between characters, as it is possible for the start bit of a transmission to immediately follow the stop bit of the previous transmission.

3 Theory of Operation

This section describes the internal operation of the UART receiver that you will design. There are many possible detailed designs – we will not present them all here, but we will give you some ideas.

The simplest solution for receiving transmission characters is to generate a clock signal at the baud rate, in this case 31.25KHz, and to use it to clock the bits one at a time into a serial-to-parallel converter. A serial-to-parallel converter is simply a shift-register with its internal flip flops connected directly to outputs. Shift registers are available in the Xilinx FPGA component library.

A counter can be used to build a simple controller which counts-off the ten bits of a transmission character (start, 8 data bits, and stop), generates the data ready signal `DRDY.H`, then resets itself. Counters are available in the Xilinx FPGA component library. Have a look at the `CB` and `CC` series library components, *e.g.* `CC16CE`, `CB4CLE` (detailed data sheets are available in the “Xilinx Library Guide,” available from Xilinx's web site as well as our course web page). Some counters in the library are *loadable* with an initial starting value. Non-loadable counters count from 0. Normal operation of a non-loadable counter, after being reset, is to count from 0 to 2^n-1 (1 increment per cycle), then to *wrap around* and start counting at 0 again. Loadable counters are used to count through fewer than 2^n values, by starting at an initial count other than zero and reaching the maximum count sooner. All counters in the library have a *terminal count* (`TC`) output signal which is set high on the cycle when the counter reaches its maximum value 2^n-1.

In this application, the actual `Q` values of the counter need not be used, only the `TC` signal. With the proper input to a loadable counter, the `TC` signal can be made to assert once every 10 clock cycles – exactly what we need for the `DRDY.H` signal. `TC` can also be used to reset the counter by enabling the load of the initial count value into the counter.

Take a look at the data-sheet for the "Serial-in Parallel-out" shift register SR8RE in the Xilinx Library Guide (available from Xilinx's web site as well as our course web page). It is instructive to start by sketching out a simple UART design based on this component and a 4-bit counter. If you have the time, you might want to implement your design in the Xilinx schematic editor and test it out in the lab. You will find is that it has unreliable behavior. Because the receiver clock is out of phase with the sender clock, it is possible for the circuit to capture input bits when they are not at valid logic values, *i.e.* during transitions.

The way we will build a reliable receiver is to start with a high-speed clock signal, in this case 8x the baud-rate clock, 250kHz. This clock is used to super-sample the input waveform. A stage is used to determine the clock period when the start bit is approximately "half-way" transmitted. This can be implemented by 4-bit serial-in parallel-out shift register with its outputs NOR-ed. The circuit generates a high output when it sees the input signal low for four consecutive clock cycles. When this happens, we assume that the input bit is the start bit, by now half-way received (recall that we have 8 250kHz clock periods per bit transmission time). This NOR-ed signal can be used to reset a counter to count off 8 clock cycles. The counter delays until approximately the middle of the first data bit, at which time we sample the input and record it by shifting into a data shift register. This will capture the first data bit in the shift register. We wait another 8 clock cycles, then enable the shift register to capture the second data bit. These steps are repeated for 8 consecutive data bits (64 cycles). You can use a second counter to control the "outer-loop" which counts through the 8 data bits.

While many variations on this basic scheme exist, you should be able to design the receiver with two counters, two shifters (all from the FPGA library), and a simple FSM controller. Remember that in a good design, all the clock inputs to the components come from the same source. Hint: If you wish to slow down the operation of a counter or shifter, use its clock enable input rather than trying to slow down its input clock.

4 Pre-lab

Before coming to lab, design an 8-bit UART receiver. You need not enter the entire design into the Xilinx schematic editor for pre-lab. You must, however, sketch a paper design using components from the Xilinx library.

Design a clock divider circuit to produce a 250kHz signal from the Xilinx 16MHz clock. It suffices to use a counter from the Xilnx library, with an appropriate choice of bit(s) for output. This clock signal should be distributed to your other circuits using a global clock buffer (`BUFG` in the library).

To help test your UART in lab, design a circuit to display values received by the UART on the Xilinx board's 7-segment display (a pair of LEDs for numeric characters). The circuit should use a data register to record whatever new data is emitted by the UART receiver and output it to the display. In the schematic, use the CS150 library's `NUMLED1` and `NUMLED2` to display the eight bit output signal (each takes a 4-bit input representing a hex digit 0-F). You will need to attach the Lab #3 library from:

```
U:\WVLib\cs150\lab3file\lib\lab3file.lib
```

You may work in pairs and submit a single design per pair of students.

For the prelab, do the following:

1. Read the data sheets for the following Xilinx library components: (available in the *Xilinx Library Guide*, linked from the course web page)
   `CC16CE`, `CB4CLE`, `SR8RE`, `BUFG`

2. Answer pre-lab questions on the check-off sheet

3. Sketch paper designs for the 3 circuits described above
   (UART, clock divider, 7-segment driver). Schematics for these circuits may be drawn before or during lab session.

4. Write a simulation script to simulate the UART receiving a single byte in both of the following cases:
   (a) The line is idle after reset, then a character `0x3E` is transmitted.
   (b) The line is idle, then a character `0xA5` is transmitted.
   Your timing diagram should show clock, `SERIAL-IN.H`, any other control signals, and `DRDY.H`.

5 Lab Assignment

5.1 Schematic Entry and Simulation

Enter your circuits into the Xilinx schematic editor and simulate them as described in the pre-lab. Show your TA the simulation results. Draw your schematics within a `BSHEET` library component, entering both lab partners' names in the title block.

Your schematics should use the following input pins:

- The 16MHz clock will come from pin 13.
- `SERIAL-IN` will come from pin 62.

5.2 Communication Between Two Xilinx Boards

You will physically test your design by having a pair of Xilinx boards communicate over jumper wires. One board will implement a serial transmitter (using a TA-provided schematic) while the other board implements a serial receiver (using your UART design).

Physically connect `PIN 70` of the sender board to the serial input of the receiver board (should be `PIN 62` in your UART design). Also connect the grounds of the two boards together. **Before powering on the boards**, have your TA check-off the connections.

Connect the Xilinx XCHECKER cable to the sender board. Power on the sender board and download into it the TA-provided sender schematic:

```
U:\WVLib\cs150\Lab7UART\SENDER.BIT
```

Reconnect the XCHECKER cable to the receiver board. Power on the receiver board and download into it your UART design.

The sender board uses two inputs. When the `SPARE` button is pushed, the eight DIP switches on `SW5` will be sent on the serial line (surrounded by the start and stop bits). Try several different data values to verify that your UART is working.

Have your TA check-off your working UART. Your TA may select arbitrary values to send down the jumper cable.

6 Extra Credit

Use the Logic Analyzer to show that you are actually sampling in the middle of each bit and that `DRDY` is being asserted.

7 Acknowledgments

Original lab by J. Wawrzynek (Fall 1994). Modifications by N. Weaver and E. Caspi.

Name: ______________________ Name: ______________________ Lab Section: ___

8 Check-offs

Prelab Questions

- Read the data sheets for the following Xilinx library components: (available in the *Xilinx Library Guide*, see course web page)
  CC16CE, CB4CLE, SR8RE, BUFG
- Using the library's counters, how would you design a circuit to count a given number of cycles *N*, then assert a 1-bit "done" output? Draw a block diagram.

  TA: ________________ (5%)

- Why should the 250kHz clock be distributed to the circuit using a BUFG?

  TA: ________________ (5%)

Prelab Design

• UART receiver circuit	TA: ________________	(20%)
• LED driver circuit	TA: ________________	(10%)
• Clock divider circuit	TA: ________________	(10%)
• UART simulation script	TA: ________________	(10%)

Schematics and Simulation

• Working UART simulation	TA: ________________	(25%)

Board-to-Board Communication

• Proper wire connections	TA: ________________	(5%)
• Working serial communication	TA: ________________	(20%)
• Verification using logic analyzer	TA: ________________	(10%, extra credit)

Turned in on time	TA: ________________	($\times$100%)
Turned in 1 week late	TA: ________________	($\times$50%)