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

EECS 150 J. Wawrzynek and N. Weaver
Fall 1999 Later revisions by R. Fearing, J. Shih, D. Chinnery, and T. Tuan

Checkpoint 1
Serial Receiver

1 Objectives

For this checkpoint, you will

1. Learn how to operate a serial port
2. Gain more practice in digital design
3. Implement the low level serial interface between the computer and your Xilinx board
4. Test your interface.

2 Prelab

You will need to wire wrap the connections for a Maxim 233 CPP adapter chip, which converts TTL level signals (which the Xilinx uses) to RS232 serial port signaling (which uses +/- 10 volts). You should also design the receiver portion of your UART and the test circuit before coming to class. For inputs and outputs to the Xilinx chip, you should use the u:/wvlib/cs150/uart.sch schematic.

This checkpoint is where you build all the circuits you need to interface with a computer through a serial port.

3 Debugging

To reduce the amount of time it takes to get your design working, you should follow a procedure along the lines of:

1. Design a finite state machine to implement the control logic.
2. Implement the finite state machine, and then simulate this schematic in Foundation to see that it works correctly.
3. Download your design to the Xilinx board. If it doesn't work, use the logic probe and digital inputs on the oscilloscope to observe what is happening. You can also use the control panel in Hardware Debugger to observe signals which are of interest.
4. Ensure you have wired things correctly and ensure that your wires actually make an electrical connection: use the logic probe or oscilloscope to verify the the signal at one end of the wire is the same as at the other end.
5. When things don't work, try to look at what is happening and relate that to what might be going wrong, so you can tell what you need to fix.

In this lab in particular, you can use the oscilloscope to measure and verify that your SHIFT_ENABLE signal (or clock signal used for this) for the receiver is close to 115.2kHz. With this in mind, you should design your circuit so that you can measure the signal: output important signals to pins so that you can observe them with the oscilloscope, and you can also use the LEDs on CR2 and CR1 to observe signals (e.g. observing your state on these LEDs as you step through operation is extremely helpful when debugging).

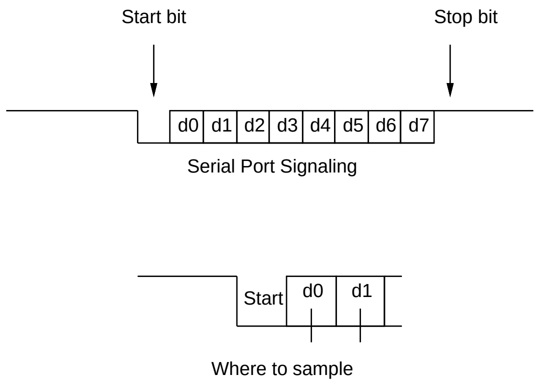

Figure 1: Bits on a serial line

4 The Maxim DC-DC converter

The computer uses an RS232 Serial Port, which uses a $+/-10$ volt signaling scheme (+10 volts="0" & -10 volts="1"). Since the Xilinx operates on a 0 and 5 volt scheme, we need to convert between the two voltages. The Maxim 233 provides a convenient, single package way to convert between the two. The R_{in} pins take RS232 signals as input, and output them on the corresponding R_{out} pins. Similarly, the T_{in} pins take a TTL signal (from the Xilinx) and output on the T_{out} pins an RS232 signal.

In order to wire up your Maxim chip, you should connect a R_{out} pin to pin 9 on your Xilinx. The two C2+ pins should be wired together. The two C2- pins should be wired together. The two V- pins should be wired together. And Vcc and ground should be hooked up normally. The corresponding R_{in} pin will hook up to the serial port through the top middle jumper cable on the nine pin serial cable. Also connect the ground cable on the serial cable (top right) to ground on the Xilinx board. The depression on the Maxim 233 chip corresponds to the top of the Wrap ID (Wrap IDs are on the bottom of the checkoff sheet).

Follow the same coloring scheme for wires as you used to wire the SRAM. This will make it easier to follow what is happening when you have more chips wired on the board, with many signal wires. More colors to code your wires will also simplify things later when you want to follow power signals (you can also color code gnd and Vcc differently), input signals, output signals and control signals.

5 UART

A UART (Universal Asynchronous Receiver/Transmitter) is a device for communicating on a serial port such as RS232. Normally, when there is no data sent, the serial line is high. When data is sent, it is transmitted 8 bits at a time. What first happens is the line goes to 0 for one clock cycle (start bit). Then, on each successive clock cycle the data is sent out 1 bit at a time, starting with the least significant bit. Finally, the data on the serial line goes high for at least one clock cycle (stop bit), and stays high until the next byte is sent. Graphically, this is shown in Figure 1, where DRDY stands for "data ready" – signaling the data is ready to be read or has finished being ready (wait state on the receiver in the computer's serial port).

To receive data over a serial line such as this is comparatively easy, just build a serial to parallel converter as seen early on in class. The only minor complications are detecting for the start and stop bit. The shift registers in the Xilinx library are a useful primitive.

The bit rate for the serial port is 115200 bits per second, or 115200 hertz.

The receiver component should take a 16, 18, or 20 MHz clock and a 1 bit data signal as input. You might want to have it produce a DRDY signal, but you don't have to. When the start bit is detected on the serial input, the receiver shifts in the data on the serial input and output them as an 8 bit byte.

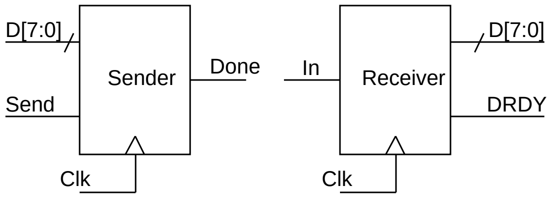

Figure 2: Components for your UART

For the receiver component, most of your logic should consist of shift registers and counters, with some small additional control circuitry. You should think CAREFULLY about your design before implementing it!

In addition, you will be using a 16, 18, or 20 MHz clock from a crystal oscillator tied to pin 13 on the Xilinx test board. You will need to produce a 0.1152 MHz clock or SHIFT_ENABLE signal [1] for your receiver and this signal should be output to pin 66 on the board for testing purposes. A small test program is provided which runs on the computer and transmits data to the serial port.

The program is U:/WVLIB/cs150/ptest.exe. Enter "com2" to read from the com2 serial port, and hit "2" to tell the program to send data. The data should then be received by the UART receiver and displayed on the NUMLEDs as two hexadecimal digits. The correct options are baud rate 115200, timeout 20ms, output hex, and parity 8/n/1.

[1] You can use a counter to divide the clock. You don't need to be perfectly at 0.1152 MHz. 5% is probably close enough.

Name: ______________________ Name: ______________________

Lab Section (Check one)

M: ☐AM ☐PM T: ☐AM ☐PM W: ☐AM ☐PM Th: ☐PM

6 Checkoffs

1. MAXIM wire wrapped (which you did before lab) TA: __________(20%)

2. Simulation of receiving one byte at 115.2 KHz (which you did before lab) TA: __________(25%)

3. Clock Divider working TA: __________(25%)

4. Receiver working TA: __________(30%)

Turned In On Time

TA: __________(full credit (100%))

Turned In One Week Late

TA: __________(1/2 credit (50% x Points))

Reminder: You need the datapath block diagram and state diagram for week 1 design review with your TA! See project spec!

. . . . Cut Here . . . .

Maxim 233

$R2_{out}$ ○	○ $T2_{in}$
$R2_{in}$ ○	○ $T1_{in}$
$T2_{out}$ ○	○ $R1_{out}$
V- ○	○ $R1_{in}$
C2- ○	○ $T1_{out}$
C2+ ○	○ GND
V+ ○	○ VCC
C1- ○	○ C1+
V- ○	○ GND
C2+ ○	○ C2-

Maxim 233

$R2_{out}$ ○	○ $T2_{in}$
$R2_{in}$ ○	○ $T1_{in}$
$T2_{out}$ ○	○ $R1_{out}$
V- ○	○ $R1_{in}$
C2- ○	○ $T1_{out}$
C2+ ○	○ GND
V+ ○	○ VCC
C1- ○	○ C1+
V- ○	○ GND
C2+ ○	○ C2-

Maxim 233

$R2_{out}$ ○	○ $T2_{in}$
$R2_{in}$ ○	○ $T1_{in}$
$T2_{out}$ ○	○ $R1_{out}$
V- ○	○ $R1_{in}$
C2- ○	○ $T1_{out}$
C2+ ○	○ GND
V+ ○	○ VCC
C1- ○	○ C1+
V- ○	○ GND
C2+ ○	○ C2-

Maxim 233

$R2_{out}$ ○	○ $T2_{in}$
$R2_{in}$ ○	○ $T1_{in}$
$T2_{out}$ ○	○ $R1_{out}$
V- ○	○ $R1_{in}$
C2- ○	○ $T1_{out}$
C2+ ○	○ GND
V+ ○	○ VCC
C1- ○	○ C1+
V- ○	○ GND
C2+ ○	○ C2-

Maxim 233

$R2_{out}$ ○	○ $T2_{in}$
$R2_{in}$ ○	○ $T1_{in}$
$T2_{out}$ ○	○ $R1_{out}$
V- ○	○ $R1_{in}$
C2- ○	○ $T1_{out}$
C2+ ○	○ GND
V+ ○	○ VCC
C1- ○	○ C1+
V- ○	○ GND
C2+ ○	○ C2-