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

EECS 150 J. Wawrzynek and N. Weaver
Fall 1997 Later revisions by R. Fearing and N. Walker

Checkpoint 1
Serial Transmitter

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 sender 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.1 schematic.

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

3 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 T_{in} pin to pin 37 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 T_{out} pin will hook up to the serial port through the top left jumper cable.

4 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 to be 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 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.

To send data over a serial line such as this is comparatively easy, one just builds a parallel to serial converter as seen early on in class. The only minor complications are insuring that you first shift out

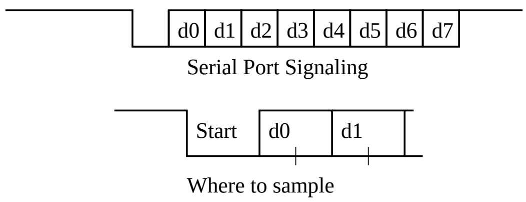

Figure 1: Bits on a serial line

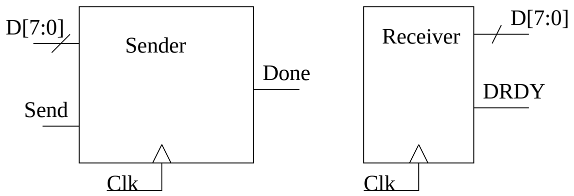

Figure 2: Components for your UART

a 0 to start, and afterwards that you keep outputting 1s. 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 sender component should take a 16 MHz clock, an 8 bit data bus, and a send signal as input. You might want to have it produce a done signal as output, but you don't have to. When send goes high for 1 clock cycle, the sender captures the data on it's input and then shifts it out to the computer.

For the sender 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 16.0 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 sender and this signal should be output to pin 8 on the board for testing purposes. The data input of the sender should be the DIP switches, and when the spare button is pressed, it should send the contents of the dip switches over the serial port to the computer. A small test program is provided which runs on the computer and monitors the serial port. To use this program, double click on the ptest icon and once the program displays a menu, hit "r" to tell the program to receive data. The values entered into the DIP switches should be displayed on the PC screen by the program after the spare button is pressed.

[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

5 Checkoffs

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

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

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

4. Sender 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-