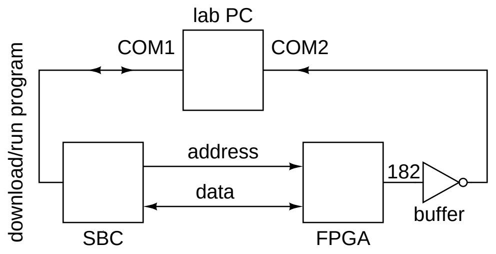
Lab 4 - Serial Output Port

Introduction

In this lab you will design a serial output interface (a "serial port") capable of transmitting 8-bit characters at 9600 bits per second (bps).

You will also write an 8086 assembly-language program to output a string consisting of your name and student number over this serial port. You will download and run this program on the lab SBC. Your program will output the string over your serial port and, if everything works correctly, you will see the string displayed on a second terminal emulator program (a second Hyperterm window) monitoring the lab PC's second serial port (COM2).

The laboratory equipment includes a circuit connected to FPGA pin 182 that the generates the correct serial interface voltages. The output of this circuit is attached via a connector and cable to the "COM2" interface on the rear of the PC:



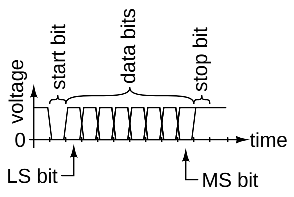
The interface between the CPU and the serial port uses an 8-bit data output port and an 8-bit status input port accessible over the PC-104 bus.

RS-232 Serial Interface

The "RS-232" serial interface is available on most general-purpose microcomputers.

Data is transferred over this serial interface one bit at a time. A zero bit (the "start bit") is sent to indicate the start of the character being sent. This is followed by the 8 bits in the character, from LS to MS bit and a one bit (the "stop" bit). Each bit has the same duration (the "bit period", in this case 1/9600 seconds). An arbitrarily long delay can be inserted between characters.

The following diagram shows how one character is transmitted over a serial interface:

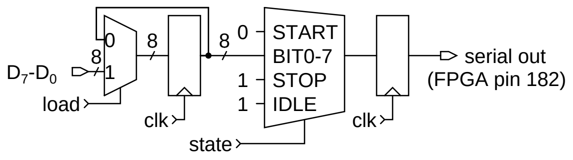


The serial output buffer inverts the signal (in addition to shifting and amplifying it). The signal levels described in this lab are those *before* the inverting buffer.

Hardware Description

All registers are loaded on the rising edge of the 8.333 MHz `clk` signal which is the PC-104 bus clock, SYSCLK (see Lab 2 for signal location). This clock is synchronous with other bus signals – IOW* and IOR* are valid on every rising edge of SYSCLK and the address bus will be valid on every rising edge of SYSCLK where either IOW* or IOR* are asserted.

Transmit Data Register and Serial Data Multiplexer

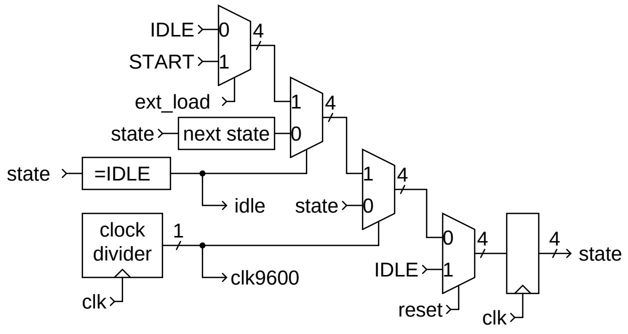


An 8-bit register is loaded with the contents of the data bus when the `load` signal is asserted. Unlike previous labs, this register is loaded synchronously with `clk`, not `IOW*`.

The multiplexer sets the serial data output to be a start bit (L, VHDL '0'), one of the eight data bits in the transmit data register in order from LS to MS bit, or a stop bit (H, VHDL '1'), depending on the value of `state`. When no character is being sent (in the `IDLE` state) the output should be set high (H, VHDL '1').

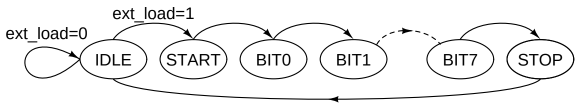
The output of the multiplexer is registered to avoid glitches. This means that the output will be delayed by one clock cycle.

Clock Generator and Controller



The bit clock generator creates the 9600 Hz `clk9600` signal that is used to advance the controller state. Use the clock divider from the previous lab with the appropriate divider value to create a 9600 Hz signal.

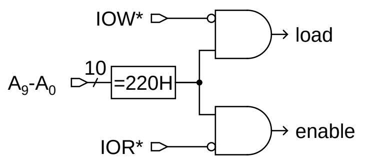
The diagram below shows the state transition diagram for the controller:



The controller always changes state (to the `IDLE` state) when the `reset` input is asserted (these transitions are not shown in the diagram). The `reset` signal is the PC-104 `RESET` bus input.

Otherwise the controller only changes state when the `clk9600` signal is asserted (this condition is also not shown). If the controller is in the `IDLE` state, it advances to the `START` ("start bit") state only if the `ext_load` signal is asserted. Otherwise the controller state advances each time the `clk9600` signal is asserted.

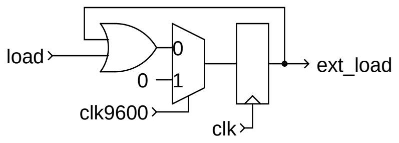
Address Decoder



When IOW* is asserted and the address bus contains 220H the `load` signal is asserted.

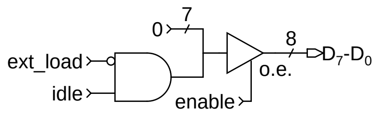
When IOR* is asserted and the address bus contains 220H the `enable` signal is asserted.

Extended `load` Signal



The `load` signal indicates that a new value has been loaded into the data register and that the state machine should commence sending the character. However, since the state only changes when `clk9600` is asserted it is necessary to extend the duration of the `load` signal until `clk9600` is asserted. The `ext_load` signal is this extended `load` signal.

Status Port



When the `enable` signal is asserted, a signal indicating whether the port is idle (finished sending the previous character) or busy is placed on the LS bit of the data bus and the MS 7 bits are set to zero. Otherwise the tri-state bus driver is left in high-impedance mode.

The idle/busy* signal is generated from the `ext_load` (see below) and `idle` signals.

Pre-Lab Assignment

Before the lab you must write, assemble and test an 8086 assembly-language program that does the following:

1. initializes a pointer to the first character of an ASCII string consisting of your name, student number followed by a carriage-return and a line feed
2. returns control to DOS if the current character is zero
3. waits until the status bit indicates the buffer is empty
4. writes the current character to the output port
5. increments the pointer to the next character
6. loops back to step 2

A `C` version of this program would be as follows:

```
main()
{
   char *p, buf[] = "Ed Casas, 12345678\r\n" ;
   for ( p=buf ; *p ; p++ ) {
      while ( inb(0x220) & 0x1 != 1 ) ;
      outb(0x220,*p) ;
   }
}
```

You should come up with a way to test as much of your program as possible. For example, substitute calls to the DOS character input and output functions in place of the port input and output.

You must also design the circuit and test it by simulating its operation. Modify your code to assume a `clk` frequency of 19200 Hz when simulating the operation of your circuit. Your simulation should include a reset cycle, a status read that returns "idle"" status, a data write, and a second status read returning "not idle". This should be followed by at least 22 clock cycles (after the first write) to view the start bit, the data bits and the stop bit being transmitted for the first value written. Then show a status read that returns "done" status again. Verify that the polarity and bit order are correct.

You will be asked to hand in your VHDL code and assembly listings at the start of the lab.

Lab Procedure

Connect the PC-104 address bus, data bus, IOR* and IOW* to the FPGA pins on the interconnect board as described in the previous labs. Make sure the serial interface cable connects the serial interface socket attached to the FPGA board to the PC's second serial interface (COM2) on the rear of the PC.

Double-check your connections and turn on the power.

Compile your VHDL code if you haven't already done so, and configure the FPGA as described in the previous lab.

Assemble and link your assembly code if you haven't already done so. All of your files should be stored in a *subdirectory* of the `c:\max2work` directory.

Run two copies of the Windows Hyperterm program. The first (setup `379com1`) should connect to the SBC as usual and be used to download and run your program. The second terminal emulator (setup `379com2`) should be set for a direct connection to COM2 and a baud rate of 9600 bps so that you can monitor the output of your program. When you run your program you should see your name and student number displayed on the second terminal emulator window.

When your device is working properly ask the TA to check your work. He will make sure your device works as required and ask you one or two questions to verify your understanding of the material.

Report

Submit a short report describing the hardware and software. Include a listing of your assembly-language program, the VHDL code and a printout of the simulation waveforms that demonstrates correct operation of your device. Follow the documentation conventions given on the course web page.

Optional Features

To get bonus marks you can improve the design by adding a control register (at address 221H) that selects the following options:

- different bit rates (e.g. 57600, 38400, ... 300 bps)
- different word lengths (e.g. 5, 6 ... 8 data bits)
- an optional parity bit

- an optional second stop bit

Hints

This is probably the most challenging lab in this course. You must prepare for the lab and understand the material involved or you are unlikely to complete the lab.

Common errors include:

- pin configuration errors (prepare and double-check your ACF file *before* you come to the lab!)
- forgetting to change the clock divider ratio back from the simulation value
- wiring errors
- using the FPGA board's 25.175 MHz clock instead of the PC-104 bus 8.333 MHz clock (wrong characters received due to wrong baud rate)
- not testing for idle state in your program (only the last character will be printed)
- getting the test for the idle state backwards in your program (your program may "hang")
- using the wrong polarity for the start, stop or data bits or sending the data bits in the wrong order (wrong characters received)
- re-using the same name (e.g. using `IDLE` for both a signal name and a constant)
- selecting the wrong device type in Max+PlusII

Your lab is unlikely to work correctly the first time. Use a methodical problem-solving approach. You may want to output the serial data to an accessible FPGA pin (e.g. pin 163, see Lab 2 for its location) in addition to pin 182 so you can view the serial output signal with the logic analyzer.