CprE 583 (Fall 2011)
MP1: Testing tools & basic VHDL (Echo serial port)
(Tentatively due Friday: 9/16, Midnight)

0. Overview

Figure 1 shows the high-level structure of the MP1 setup. You will first generate a hardware configuration (bitfile) for the FPGA without modifying the VHDL, and download the bitfile to the FPGA. You will then test that the design works by using a Linux application call "minicom". Minicom is a simple application that lets you send/receive data over the serial (UART) port. If everything is setup correctly, then when you type in the mincom window, you will see what you type. If not, then you will likely just see a blank screen as you type.

In part III you will be making a small modification to the VHDL to change lower case letters into upper case letters. (10 pts)

In part IV you will be asked to understand how the core component of MP1 (mmu_uart_top) works at it interface (i.e. how to use it). You will also be asked to draw the structural schematic of MP1 (Note: in the schematic you will treat mmu_uart_top as a black box). (30 pts)

In part V you will design a simple component that delays a byte received by 5 clock cycles, then echoes it back, as opposed to how the design currently echoes a byte back as soon as it receives a byte. You will make an component to replace the echo component. I suggest using a delay shift register (See slides 54 – 58 of lecture 2). (20 pts)

In part VI you will replace the echo component with a component that echoes back what is received, unless it receives consecutive numbers (between 0-4), in which case it will ADDITIONALLY send back the sum of the two numbers. You will first be asked to draw out the structural schematic of your component, and then you will need to write the VHDL that described your component. **Note: Do <u>NOT</u> use a state machine!!** (30 pts)

In part VII you will explain how to change the baud rate of the UART module (10pts)

Part VIII is BONUS: Modify your component (schematic and VHDL) from part VI to handle consecutive numbers from 0-9. The main difference from part VI is that the sum can now be a 2 digit number. (+5pts)

Part IX explains what to turn in.

Part X has some FAQs. Make sure to read 1 and 2, which describe how to save and load datasets (.dat) and wave format (.do) files. And 3 is a reminder to exit minicom when you are done with the hardware. VERY important! If you do not exit you will block others from using minicom. Email me if this occurs.

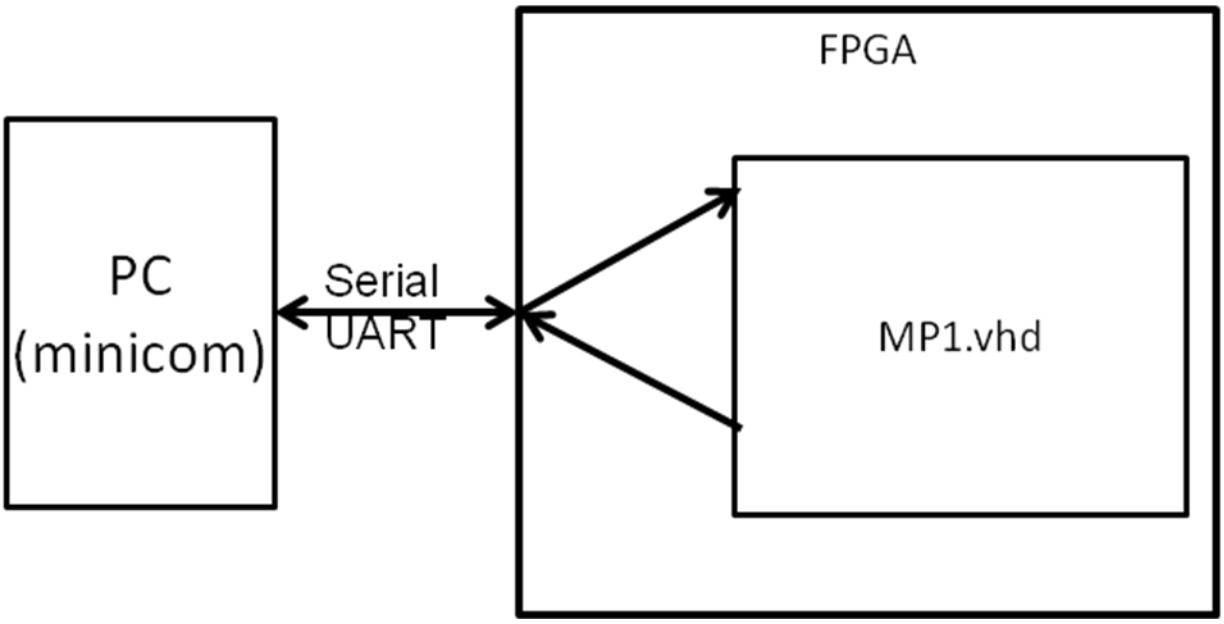

Figure 1

I. Download and file locations

1. Running the tools: 90% of the time you should not need real hardware
    a) Local students
        i. Any Linux machine in Coover 1212, 2046, 2048, 2050
        ii. When testing a design on the FPGA (i.e. you have a bitfile to download to the FPGA): use one of the two machines in Coover 2050 that have a FX70 FPGA attached. They will be labeled
    b) Distance students use NX to log into
        i. research-3.ece.iastate.edu, research-5.ece.iastate.edu, research-6.ece.iastate.edu, xilinx-1.ece.iastate.edu, xilinx-2.ece.iastate.edu, xilinx-3.iastate.edu (for a full list: http://www.it.eng.iastate.edu/remote
        ii. When testing a design on the FPGA (i.e. you have a bitfile ready for download): use xilinx-2 or xilinx-3 (**xilinx-1 should be up soon)**
            1. Note machines such as research-3,5,6 are likely much faster than xilinx-1,2,3. So use these until a bitfile is ready to test.

2. Make a directory to store your work. For example
    a) mkdir cpre583_MP1
    b) cd cpre583_MP1
    c) If you are new to Linux, then go through these pages or see tool overview ppt
        i. http://www.reallylinux.com/docs/basic.shtml
        ii. http://www.reallylinux.com/docs/direct.html

3. Download MP1.tar.gz (or .zip) from http://class.ece.iastate.edu/cpre583 into you ISU home directory (also referred to as your ISU U: drive)

4. Download Xilinx_12_4_src.txt from this link as well (Note this file should already be in zip file in the same directory as the ise project)

5. Uncompress the assignment
    a) gunzip MP1.tar.gz (or unzip MP1.zip)
    b) tar -xvf MP1.tar (skip this if using the .zip file)
    c) cd MP1

6. ISE project is located here: MP1/MP1.xise
    a) Set up environment
        i. Make sure you are using a bash shell. "echo $SHELL" to check. If you are not then type "bash" to get into a bash shell
        ii. Note: you can set your default shell to bash at https://weblogin.iastate.edu/cgi-bin/index.cgi (Manage User -> View/Edit your Linux login shell -> select /bin/bash
        iii. source ./Xilinx_12_$_src.txt (assuming this file in your current directory)
        iv. Note: You MUST run this source command for each terminal window you open.
    b) To open project: ise MP1.xise &

7. You should be able to edit your VHDL, and build your bitfile from ISE

8. Viewing the Hardware (Distance Students for testing on the FPGA). Note: for MP1 you will not need to see the physical FPGA.
    a) http://xilinxcam1.ece.iastate.edu: to see LEDs on xilinx-1 FPGA board
    b) http://xilinxcam3.ece.iastate.edu: to see LEDs on xilinx-2 FPGA board
    c) http://xilinxcam5.ece.iastate.edu: to see LEDs on xilinx-3 FPGA board
    d) username: xilinxuser
    e) password: drjones
    f) Depending on what plugin your browser has, select activeX or Java.
    g) Distance student hardware access (i.e. you have a bitfile to test)
        i. Use NX to login into xilinx-2.ece.iastate.edu or xilinx-3.ece.iastate.edu
        ii. Note using NX is highly recommended. ssh -X will be WAY too slow
    h) NX download locations
        i. For Windows: http://www.nomachine.com/download-client-windows.php
        ii. For Linux: http://www.nomachine.com/download-client-linux.php
        iii. For MAC OS: http://www.nomachine.com/download-client-macosx.php
        iv. For Solaris: http://www.nomachine.com/download-client-solaris.php

9. Sharing the Hardware
    a) Everyone should do the following before and after using the FPGAs connected to xilinx-2 or xilinx-3 (Note: Local students should be primarily using the on campus resources)
        i. Check if anyone else is using the ML507.
        ii. If no one responses to your request, then announce you will be using it. Maybe even give a time frame for when you will be done.
        iii. Send an message when you are finished

    b) Two commands that we will use for this are “mesg” and “wall”.

mesg:

mesg y: Allows others to write to you.

mesg n: Disables others from writing to you.

mesg without any options tells you if others can write to you (y/n)

Make sure you type “mesg y” before you start using the HW, so you can tell when others request it. Also type “mesg” to verify that it does return “y”

wall:

wall allows you to broadcast a message to everyone logged into the same computer (e.g. xilinx-2).

Example:

wall Is anyone using the HW?

wall OK, I have not heard from anyone so I am going to use it from 7 – 7:15 pm, I will let you know if I finish earlier.

wall I am finished with the HW.

If I have time to think of, and more importantly implement a better way to share the hardware, then I’ll let the class know. And if anyone has any simple ideas for sharing the HW please let me know.

II. Getting started

1. Before you make any changes, make sure the base design works.
    a. Build the bit file within ISE
        i. In the "View" window (just above Hierarchy window) make sure "Implementation" is selected
        ii. In the "Hierachy window" window, highlight MP1_top
        iii. In the "Processes for MP1_top" window
            1. Right Click: Generate Programming files
            2. Select "rerun all"
            3. After a couple minutes it should generate a FPGA bitfile called MP1_top.bit in the same directory as your project.
    b. Load bitfile to the FPGA
        i. Type "impact &" at the command line prompt were you open ise.
        ii. It will ask if you want the system to create a project for you. Yes or No is fine
        iii. select: Create new project
        iv. Configure Device using boundary scan, Auto connect
        v. Do you want to continue to select configuration files: select Yes
        vi. Bypass until you get to the "fx70" this is you device
            1. Select MP1_top.bit
            2. Open
        vii. Do you want to attach an SPI: select No
        viii. Cancel out of the window that pops up next
        ix. Right click on the "fx70"
        x. Program, OK
        xi. Bitfile should load. Impact will tell you if the bitfile successfully loaded. Also if you pay close attention to xilinxcam5 (for xilinx-3) you will see a GREEN LED trun off, and back on once the bitfile is loaded.
    c. Start minicom in a new terminal window (minicom is simple Linux application for sending data over the serial port)
        i. minicom
        ii. Make sure it is set to 9600 8N1
            1. Ctrl-A Z
            2. Configure Minicom: O
            3. Serial port setup
            4. Bps: E
            5. 9600: E
            6. Enter, Enter
            7. Exit
    d. When you type you should see characters appear in the minicom window. If you do not then you should contact me.
    e. **EXIT Minicom!!!!!** Make sure to exit minicom when you are finished testing on the FPGA. If you do not then you WILL block others from using the serial (UART) port. If you get block out of minicom, then send me an email.

III. Modify the VHDL to convert a-z to (A-Z): (10 pts, 70% sim, 30% hardware)

You will modify MP1.vhd to convert the letters a-z to A-Z when you type on the keyboard. In other words when you type an "a" you will see "A" appear in minicom. This should not take more than about 5 lines of VHDL. Hint: goto http://www.asciitable.com/ and look at the Hex (Hx) encoding of the characters a-z and A-Z.

1. Within ISE open MP1.vhd
    a. In the "Hierarchy" window right click on MP1_top and select "open"

2. Look through MP1.vhd, and try to get a feel for what MP1.vhd is doing. You will be asked later to draw some schematics.

3. Simulate the design. In general you should ALWAYS simulate your VHDL to verify that its logic is correct. In this case you should find that MP1.vhd is echoing data back to the driver.
    a. In the "View" box (just above Hierarchy) select "Behavioral Simulation"
    b. In the "Hierarchy" box highlight MP1_tb (this is the testbench
    c. In the "Process for: MP1_tb" box expand "ModelSim Simulator"
        i. Right click and select "Rerun all"
    d. ModelSim will start up an begin to simulate your design
        i. In the lower left hand corner you can see how long it has simulated for. I have set the default to simulate for 10 ms. (Should take 1 – 3 minutes to complete simulation.
        ii. Once the simulation is finish click in the wave window
        iii. Click Zoom full (this will show the results from t=0 to t=10ms)
    e. Now use your VHDL to figure out which signals to take a closer look at in order to verify if your design (MP1.vhd) is working as expected
    f. Also look at the testbench driver .vhd file (MP1_top_driver) to see if you can use the simulation to help you understand what the VHDL design is doing.
    g. Note: You will need to zoom in A LOT to see the echo occur. If you cannot see individual clock pules, then you are likely not zoomed in enough.

4. Between the lines "Your code below here!!!!" and "Your code above!!!" place your VHDL code to convert a-z to A-Z. (Don't forget to save your changes)

5. Once your VHDL design appears to be correct in simulation, try to build a bitfile to down load to the FPGA (follow the directions from II. Getting Started).

6. If when you type lower case letters in minicom they appear as upper case, then CONGRATUALTIONS!! if this is your first time doing hardware design. Save your bitfile as MP1_part3.bit.

7. Save you .dat, and .do file. Call them MP1_part3.dat and MP1_part3.do

IV. Explore the design in more detail (Making a user spec for the mmu_uart_top component, and drawn a circuit schematic for MP1). (30 pts)

Explore more details of the design. The core of this MP is built on top of a UART design developed by Mihai Munteanu, and is meant to be a very clean and simple implementation of a UART communication core.

You task is to:

1. Draw a black box diagram of the mmu_uart entity labeling the input and output ports
2. Give a timing diagram(s), and accompanying written description of how to use this entity. It should be written such that someone else can use it as a userguide for properly using the mmu_uart component.
3. In order to reverse engineer how this module works you may need to add more signals to your simulation. Here is a quick overview of how to do this: (or see tools over ppt).
    a) In ModeSim in the "sim" window expand the hierarchy of MP1.vhd (uart_hw), and start adding signals to look at.
    b) To add signals to the waveform
        i. Highlight a component in the "sim" window
        ii. Select the signals that pop up in the Objects window (these signals will match what is in the VHDL code)
        iii. Drag and drop these signals into the wave window
        iv. Format the signals appropriately, and add dividers to label groups of signals. Note adding dividers is VERY important for organizing groups of signals. Right click in the wave window for formatting options, and adding divider options.
    c) Save your new wave file. If you do not save you will need to Re-add and reformat your signals if you restart the simulation. It is a good idea to get into the habit of saving you wave file/format often. You will find it is VERY frustrating to do LOTs of formatting, just to have to redo it because you forgot to save.
        i. In ModelSim click in the wave window
        ii. File->save
        iii. Browse
        iv. Select "MP1_wave.do"
        v. Save (if asked if you want to overwrite say "yes")
4. Running simulation for more time.
    a) In the Transcript window (in ModelSim)
    b) run <time> <unit>, For example: run 1 ms
    c) Note: typically after you add more signal to the simulation, you will want to close Modelsim and restart the simulation from ISE.

5. Draw the schematic for MP1
    a) For mmu_uart: treat this component as a blackbox in your schematic
    b) For process_data: Draw out the hardware at the Flip-flop, mux, and gate level
    c) For echo: Draw out the hardware at the Flop-flop, mux, and gate level.
        i. Note: you will notice that one of the processes in echo gives assumes some default assignment. It will be useful for you to recode this so that such signals are explicitly defined for each possible path of execution.

V. Delay the echoing of a byte by 5 clock cycles. (20 pts, 70% sim, 30% hardware)

Now that you understand how the mmu_uart_top works you will make a new module that captures a byte when received, delays it for 5 clk cycles, and then echoes it back to minicom.

1. Draw a block diagram of your entity: label the input and output ports
2. Draw a block diagram of the internal workings of your entity (e.g. show multiplexers, flip-flops, etc.)
3. Give a short written description of how your module operates
4. Implement the component in VHDL
5. Test in simulation
    a. Make sure to save you .dat and .do file to turn in (call them MP1_part5.dat, and MP1_part5.do).
6. Once it works in simulation test in hardware (i.e. check that when you type in miniciom the FPGA still echoes back to you): save your bitfile as MP1_part5.bit.
7. I suggest making use of delay shift registers as discussed in class.

VI. In flight calculations. (30 pts, 70% sim, 30% hardware)

Next you will make a more involved component that does some computation as data is flowing through the system. Make a component that echoes data back, unless it receives consecutive ASCII numbers (‘0’ – ‘4’). When receiving consecutive numbers your module should in addition return the sum of the consecutive numbers. Here is an example of what you would see in Mincom (Note: the characters that are underlined are the additional characters returned by your component).

a b e 3 b 4 v 4 3 7 3 6 a g w 2 h 2 3 5 a 3

1. Draw a block diagram of your entity: label the input and output ports

2. Draw a block diagram of the internal workings of your entity (e.g. show adder blocks, multiplexers, flip-flops, shift registers etc.). **Note: Do NOT use a state machine!!**
3. Give a short written description of how your module operates

4. Implement the component in VHDL
5. Test in simulation
    a. Make sure to save you .dat and .do file to turn in (call them MP1_part6.dat, and MP1_part6.do).
6. Once it works in simulation test in hardware: save your bitfile as MP1_part6.bit

VII. Changing Baud rate (10 pts)

Look through the VHDL and figure out how to change the baudrate from 9600 to 38400. Write a short description of what needs to be changed. Note your circuit runs at 33 MHz.

VIII. Bonus (5 pts)

Extend part VI, so that you can handle adding consecutive numbers from '0' – '9'. The major difference from part VI is that sometime you will need to return two digits for the sum.

1. Show the modified schematic
2. Implement the circuit in VHDL
3. Simulate name your files as MP1_part8.dat, and MP1_part8.do

IX. What to turn in (zip or tar.gz).

Email me a .zip or .tar.gz file with the following

1. Your modified and newly created .vhd files (add part#.vhd to the names)

2. A copy of your wavefile format (.do) and dataset (.dat) for parts 3, 5, and 6 (8 Bonus).
    a. Specify a time stamp and group of signals I should view.

3. Send me your bitfiles. MP1_part3.bit, MP1_part5.bit, MP1_part6.bit (bonus MP1_part8.bit)

4. Your block diagrams, timing diagrams, and written user spec from part 4.

5. On campus students: Demo your working bitfile to me in Cover 2041.

6. Off-campus students: I will run your bitfiles myself

X. Updates and FAQs

1. How to save a simulation/waveform dataset
    a. Dataset
        i. File->Datasets->(highlight vsim.wlf)->Save As->name_file.wlf-> Done
    b. Waveform format
        i. File -> Save Format -> name_file.do

2. How to load a data set and waveform format (Note load dataset first)
    a. Dataset
        i. File Datasets -> Open -> Browse -> (select wanted .wlf file)
    b. Waveform format
        i. File -> Load -> (select wanted .do file)

3. Exit minicom

Make sure to exit minicom when you are finished with the hardware. If you do not then you will block others from using the serial (UART) port for testing.

4. How to unlock the cable

There are two types of locked cable issues:

1. The locked cable will not let you download a bitfile

Two solutions (if neither works reboot the machine, for distance students send an email asking for the machine to be rebooted)

Solution 1:

```
impact -batch
setMode -bscan
cleancablelock
quit
```

Solution 2:

```
xmd
xclean_cablelock
```