EXPERIMENT NUMBER 1
Design of an Eight Bit Latch

INTRODUCTION:

Often times, hardware is not available early in the design process. Although hardware may change as the design evolves, simulating the design early and often usually leads to a much shorter design cycle. By simulating first, problems are found and eliminated more quickly and easily than if you went straight to hardware testing. As a result, most problems are worked out prior to implementation in hardware and you will be in a better position to understand and solve those problems that do occur. These concepts will be introduced in this lab, where you will develop and test hardware.

In this lab, you will develop an 8-bit latch and will integrate it with hardware on the XS40 board. An 8-bit latch is made up of eight 1-bit latches, where each latch stores a single-bit value. Your 8-bit latch will store values written to the seven-segment display. The latch will be created in Design Architect and will be targeted for implementation with Xilinx FPGA parts. You will first test this design through simulation with Quicksim and then, once you are certain the design works, will download it to the XS40 board for final hardware verification.

OBJECTIVES:

1. Design and implement a simple 8-bit parallel port using an FPGA
3. Design simple hardware-models for cosimulation.
4. Introduce hardware simulation with Mentor Graphics QuickSim Pro.
5. Familiarization with the XS40 Board

REFERENCES:

Mentor Graphics Tutorials: http://www.ece.umr.edu/tutorials/
Appendix D, XS40 Schematic: http://www.ece.umr.edu/courses/cpe214/schematics.pdf

MATERIALS REQUIRED:

- Mentor Graphics software
- Simulation model of the XS40 board: http://www.ece.umr.edu/courses/cpe214/dist/equipment_intro.tar
- XS40 simulation board
- XS40 schematic: http://www.ece.umr.edu/courses/cpe214/schematics.pdf
- Xess Corporation: http://www.xess.com
- Windows-based computer with an unused parallel port
- Ftp program
**BACKGROUND:**

At the heart of an 8-bit D-latch are eight D latches (Figure 1). A D latch will store the value of the input (D) on the falling edge of a clock (G), and will keep this value on the output (Q) until the next rising edge of the clock.

![D-type Latch](Figure 1)

A D-latch passes the value of D through to its output Q while G is high. When G is low, the output Q does not change, regardless of how G changes. A microprocessor can use this feature to demultiplex its address and data lines. For example, a microprocessor may wish to write a value to a seven-segment display. Data is only valid on the processor’s data bus for a short period of time so we need to provide a way to hold the data while it is being used to drive the display. The 8051 microcontroller provides four built-in I/O ports for this purpose. Four are sufficient for very small applications but most systems require more than this so the 8051 allows for expansion of its I/O ports. A simple eight-bit latch can be used to provide an extra output port for our microcontroller.

In this lab, you will test each bit of your latch using a 3 to 8 bit decoder. By driving the 3-bit input of the decoder with a binary count (i.e. counting from 0 to 7), you will cycle the decoder’s output through a seven bit test pattern for the seven-segment display using only three bits. In the hardware verification stage, the inputs to the decoder and the clock input to the latch will be controlled from the parallel port on the PC.

Your design will be realized in hardware using the XS40 board by the Xess Corporation (Figure 2). The XS40 board contains several features to rapidly produce prototypes of hardware designs. Additional details will be given in lab 3. The schematic for the XS40 board ([link](http://www.ece.umr.edu/courses/cpe214/schematics.pdf)) shows how the chips are connected. This schematic will be very useful to us in our experiments. The left side of the sheet shows the signal connections to “J3” (‘J’ stands for jack or jumper). Connector J3 consists of all the pins that run up and down the two sides of the XS40 board. These pins are made so the board can be plugged right into an experimenter’s breadboard, but they also make good probe points for an oscilloscope or logic analyzer. Connector J2 is the VGA connector. Connector J1 is the parallel port connector to the PC. You can see the chips these jumpers connect to by following the signal names given to the wires. For example, J3-67 (connector J3, pin 67) is connected to a wire labeled XCBUS67. Looking at the schematic, the same wire, XCBUS67, also connects to P17 (port 1 bit 7) of the 8031 (lower left hand corner of the schematic) and to pin 67 of the XC4005XL FPGA (upper right hand corner).
**PRELIMINARY:**

- Review the tutorials on Design Architect and QuickSim listed in the references.
- Sketch the design of an 8-bit latch before coming to lab. Use an LD latch like that shown in Figure 1 and Figure 3.
- Sketch both the input waveforms to the 3 to 8 decoder you will use to test your design and the resulting output waveforms you expect to see during simulation.

**PROCEDURE:**

**Summary:** Create an 8-bit latch in Design Architect. Add the latch to the XS40 board schematic to control output to the seven-segment display. Control the inputs of the latch by adding a 3-to-8 decoder connected to the parallel port. Simulate your design with QuickSim. Compile your design for a Xilinx FPGA. Verify your design in hardware with the XS40 board.

1. Create a directory for your lab using the Unix command mkdir (type “mkdir lab1” to create a directory called lab1 – for more information on any Unix command, type the command “man command”). Download the file [http://www.ece.umr.edu/courses/cpe214/dist/equipment_intro.tar](http://www.ece.umr.edu/courses/cpe214/dist/equipment_intro.tar) and place it in your lab directory. This single tarfile contains a top level schematic of the xc4005 on the XS40 board with many of the fixed pins already defined for you. To extract these smaller files, untar equipment_intro.tar using the command:

   ```
   tar -xvf equipment_intro.tar
   ```

   You should now see several files and directories in your lab directory.

2. We will now begin to use Design Architect (DA). DA is a schematic capture tool from Mentor Graphics. It uses what is called the ‘Falcon Framework’ in order to provide a common look and feel to all of Mentor’s electronic design automation tools. Before using Falcon Framework based tools like DA and QuickSim, you must always set your working directory (MGC_WD). It’s also
a good idea to define a user’s library in order to avoid creating hard pathnames in your design files. To set your working directory to lab1, type the command:

```
setenv MGC WD 'pwd'
```

while in your lab1 directory. This command is also aliased to “swd”. Mentor Graphics will check the shell variable MGC_WD to find the working directory. Notice the use of the reverse quotes ‘ ‘. This tells the Unix shell to insert the results of the Unix pwd command here in place of the text “pwd”. To set your user library, type:

```
setenv USERLIB 'pwd'
```

while in your lab directory. This command is aliased to “sul”. You must use these commands each time you login before doing any work with Mentor’s Falcon Framework tools.

3. Open Design Architect using the command “da” at the Unix prompt. Create a new sheet called 8bit_latch by clicking the open sheet button on the right of the design architect window. The open sheet dialog box will appear after clicking this button. Since we are creating this design for the first time, you must type the component name in. Enter your component name (8bit_latch) in the dialog box. Do not change the sheet name. The next time you open your schematic you can locate it by using the Navigator option.

4. Enter the 8-bit latch design you created in the preliminary. Use Xilinx X series parts, found under the menu item: Libraries--Xilinx--XC4000X. Figure 3 shows a 2-bit latch you can use as a guide.

**NOTE:** Always use parts from the same library in your design. If you mix parts from different libraries, your design may not compile to hardware when you are done.

If you have not used Design Architect before or need a refresher, here are the steps that were used to create the 2-bit latch:

a) **Choose and place the latch:** Choose Xilinx parts from the menu item Libraries—Xilinx Libraries. Choose parts for the XC4000X FPGA from the menu at the right of the window. Search by type and select latches. Select the latch ld. Once chosen, a ghost image of the part appears in the schematic window and can be moved with the mouse. Move it to the desired location and place it there by left-clicking the mouse.

![Figure 3 Two-bit latch](image)

If you need further assistance, please consult the Mentor Graphics documentation or reach out to the support team.
b) **Zoom out** with a short lower left to upper right ‘stroke’ by dragging the mouse from the lower left to the upper right of the window while pressing the middle mouse button. You can also **Zoom in** by pressing the middle mouse button and dragging the mouse from the upper left to the lower right of the area you would like to zoom in on.

c) **Copy the part.** Hit F2 to unselect all possible selections (NOTE: the operation of each function key is shown at the bottom of the window. Look at the middle of the function key description, between F6 and F7, where you see a column of characters “sca”. This column indicates what each row of descriptions means: The operation on the top row is performed when the function key is hit alone, the second row when hitting Shift and the function key, the third for Control-function, and the last for alt-function). Select (just) the flip-flop by placing the mouse pointer over it and clicking the left mouse button. Hold down ctrl-MMB (middle mouse button) while moving the mouse to create a copy of the selected item(s). Release the middle mouse button to place the copy after you have it in the desired position. If you have difficulty copying the part using this method, you can also copy the part using the mouse menu (right click with part selected, then select Copy>Selected).

d) **Add a wire** for the clock. Press and release F3 to add a wire. Click the left mouse button at the location you want to start the wire, in this case at the end of the latch G pin. Click the left mouse button once each time you want to put a bend in the wire, twice to end the wire. Press Esc to quit making wires. If you make a mistake, press F2 to unselect all, select the wire you wish to delete, and then hit the del key to delete it. Position the clock wire as shown in Figure 3. Note that when you end a wire on top of another wire, a connection between the wires is made automatically. If you wish to make a connection between crossing wires, select the wires and hit shift-F6 to connect all.

e) **Add a portin** for the clock. The portin part is the flag connected to the clock port in Figure 3. A similar flag, the portout part, is connected to at the output bus Q. The portin part is found from the Libraries--Xilinx--XC4000X--By Type menu under the heading “io”. In the parts window, click the right mouse button and select the menu option “Show Scroll Bars”. Scroll down to the portin part, select it, and place it at the end of the clock wire. Be sure to use the connection point (a small purple diamond on the symbol) to attach wires. The connection point on the portin is on the pointed end while the connection point for portout’s is on the square end.

f) **Change the name of the net.** Connected wires, such as the ones connecting the clock, are also called nets. Rename the portin name from “NET” to “G” by placing the cursor over the word NET, hitting shift-F7 to change the text value, and typing “G” in the dialog box under New Value.

g) **Add bus rippers.** The 2-bit D input and Q output are connected using a bus – a bundled group of individual wires. To connect to a bus requires a special part called a bus ripper. The bus ripper is available in the XC4000X library, under type “logic”, and is called “rip”. Go to the library and select it. The fat end connects to the bus, the skinny end to an individual wire. Connect a bus ripper to both the D inputs. You cannot connect connect the ripper to the Q output in its default position, since the fat end is positioned toward the flip-flop. You can change the ripper’s orientation using the menu that appeared at the bottom of the screen when you selected the rip. From the flip menu item, choose “horizontal” and hit OK. The rip is now flipped horizontally from its original position. Place a bus ripper at each of the Q outputs.
h) **Add the buses.** Press the Shift-F3 key to add a bus. Busses are added similar to wires. Add a bus both to the D input and to the Q output as shown in Figure 3 (or as needed for an 8-bit design). Short down or up strokes can also be used to start the add wire or add bus process as well.

i) **Add the portin/portout for the bus.** The portin/portout for the bus is added the same way as for the clock (steps e and f), except that there is an important difference in the name. Name the D-input net “D(1:0)” and the Q-output net “Q(1:0)”. The addition “(1:0)” specifies that the port connects to a 2-bit net, where the wire named “1” will be the high bit and the wire named “0” will be the low bit. An 8-bit net might be named “blah(7:0)”, where wire “7” is the high bit.

j) **Name the wires connecting to the bus.** By naming the wires ripped from the bus, you specify which wire is being ripped. Hit F2 to unselect all. With the mouse in the schematic window, press the right mouse button and from the menu select Select Area--Property. Draw a box around the rips for the D-input bus. Note that the “R”’s associated with each rip have turned white, indicating you have now selected this property of the rip. Hit the right mouse button and from the menu choose “Sequence Text”. Hit OK on the resulting menu. This will sequence the rip names from 0 to 1 or, if 8 rips are selected, from 0 to 7. Unselect all and do the same for the Q-output bus.

5. Add a comment box in the lower right corner of your schematic. Type the command ‘add tex’ and type a name for your schematic. Place the text in the lower right corner. Repeat this step to add your name and date. Type the ‘add rec’ command and draw a box around your comment text. Labeling your schematics is a good idea and this adds a more professional touch than a handwritten label on the hardcopy.

6. Check and save the sheet. Check the sheet from the Check--Sheet menu at the top of the window. If the sheet checks OK, close the check-sheet output window and save results, otherwise go back to your schematic and correct any errors. Unconnected wires are a common problem. You can ‘cross select’ items in the Check-Sheet report and view them in the schematic window if you need help finding the location of an error or warning. To save your file, use the File--Save Sheet menu.

7. Print a copy of your latch schematic from the File--Print Sheet menu item.

8. Create a symbol for your 8-bit latch. A symbol can be created using the menu item Miscellaneous--Generate Symbol. The symbol name is typed in the two text boxes under the title “Place Symbol In”. Call the symbol you create “8bit_latch” (that should be the default). Check (with defaults) and save your symbol when you are finished. The symbol created for the 2-bit latch is shown in Figure 4. As you can see from Figure 4, both D and Q are busses, while G is a single-bit line. Print a copy of your latch symbol.

![Figure 4 Two bit latch Symbol](image-url)
9. Close both the symbol sheet and the schematic sheet you just created. Open the xc4005 schematic sheet using the Navigator option in the open sheet dialog box. It should be in your working directory.

10. Finish up your design by connecting together the parallel port, a 3-to-8 decoder, your latch, and the seven-segment display. Using the “Choose Symbol” button in the “Schematic Add/Route” palette on the right of the DA window, insert the symbol that you just created into the xc4005 sheet. Insert a 3-to-8 decoder from the Xilinx X series parts menu. As shown in Figure 5, you must hook them up such that a) 4 pins of the parallel port drives the enable bit and the 3 inputs of the decoder, b) 1 pin of the parallel port drives the clock on your latch, c) the output of the decoder drives the 8-bit input of your latch, and d) the output of the latch drives the seven-segment display.

The parallel port is connected through FPGA pins 44-48. Connect them as shown in Table 1. By connecting these pins to the parallel port, we can strobe them from the PC when we perform the hardware verification step of this lab.

<table>
<thead>
<tr>
<th>Pin #</th>
<th>Function</th>
<th>Parallel Port Data Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>A0</td>
<td>D0</td>
</tr>
<tr>
<td>45</td>
<td>A1</td>
<td>D1</td>
</tr>
<tr>
<td>46</td>
<td>A2</td>
<td>D2</td>
</tr>
<tr>
<td>47</td>
<td>E</td>
<td>D3</td>
</tr>
<tr>
<td>48</td>
<td>Clk</td>
<td>D4</td>
</tr>
</tbody>
</table>

The output from your latch needs to go to the seven-segment display, so you can see results during hardware verification. Decide how to connect the latch to the seven-segment display by examining the XS40 board schematic. There is not necessarily a right or wrong way, though you want to understand how you’ve made the connection so you can understand your results later on. Keep in mind that one and only one bit of the decoder output will be active at a time and only when the E input is enabled. Are the decoder bits active high or active low? Is the E input active high or low? The latch outputs latch the decoder outputs only when there is a rising edge on the clock input. Each bit connected to the display will only light a single segment. Does a high at the output of a latch turn on its segment or does a low turn it on?

<table>
<thead>
<tr>
<th>FPGA</th>
<th>7Seg</th>
</tr>
</thead>
<tbody>
<tr>
<td>P19</td>
<td>a</td>
</tr>
<tr>
<td>P23</td>
<td>b</td>
</tr>
<tr>
<td>P26</td>
<td>c</td>
</tr>
<tr>
<td>P25</td>
<td>d</td>
</tr>
<tr>
<td>P24</td>
<td>e</td>
</tr>
<tr>
<td>P18</td>
<td>f</td>
</tr>
<tr>
<td>P20</td>
<td>g</td>
</tr>
</tbody>
</table>
11. When your design is finished, label it, check it, save it, and print out a copy of your schematic (similar to Figure 5). Your next step will be to simulate your design with QuickSim.

12. Close Design Architect (by right clicking on the title bar and choosing Quit or by using the shortcut key combination Alt-Q).

13. Create a design viewpoint that can be used by QuickSim with the Unix command:
   `pld_dve -s xc4005 xc4000xl`

14. Open your design in QuickSim with the Unix command:
   `quicksim xc4005`

15. Simulate and verify your design:
   a) **Open the xc4005 sheet** by choosing File, Open Sheet at the menu at the top of the QuickSim window.
   
   b) **Trace the output.** Mentor can be configured to trace the output to the seven-segment display by selecting the output bus using the left mouse button and then clicking the trace button from the buttons on the right of the QuickSim window.
   
   c) **Create input waveforms.** Add the inputs from the PC, pins 44-48, to the trace as waveforms that can be edited. You need to apply appropriate signals to each of these pins to verify your

---

1 Note that there is a problem with our simulation libraries at UMR. Normally the above command would be correct. However, at UMR you should use xc4000e in place of xc4000xl with the pld_dve command. Ask your instructor if you are unsure of the correct command to use.
design. Three of these lines control the decoder input, one enables the decoder, and one controls the clock.

- Deselect the output bus (if selected) by hitting “F2”. Notice that the function keys functions are given at the bottom of the window.
- Select pins 44-48 in the schematic, click ‘WF Editor’, and click ‘Edit Waveform’. All these signals are now added to the trace listing as force signals.

**Use the force command to enable the decoder.** Identify which pin from the PC is connected to the enable pin of the decoder. Place the mouse cursor over that pin name in the “Trace” window and hit the left mouse button to select it. While the mouse is still over the pin name, press the right mouse button and select Force—Single Value from the window that pops up. Another window will appear which allows you to configure the signal. You can set the value to force the signal to, the time to begin forcing that signal, and the type of force. For this signal, select the appropriate value to force the enable pin to, enter the start time as 0, and press OK without changing any of the other default values. You may force the enable pin to 0 or to 1. Which value is correct?

**Use the force command to create a clock.** Identify the pin connected to the clock and select that pin name in the trace window. Click the ‘Stimulus’ button on the right of the window and then ‘Add Clock’. Use a period of 25 ns.

**Assert GlobalSetReset.** For the latch to work properly, you will also need to pulse an internal, active high, signal called GlobalSetReset. The FPGA generates this signal automatically during power-on to reset all latches to a known state. GlobalSetReset can be found three levels deep in your latch design. Open down on successive components until you see this signal. Select it and add it as a waveform that can be edited. Force it with a short positive pulse 5ns in duration that occurs 5ns from the start of simulation. This positive pulse can be created by forcing the line three times, at 0ns, 5ns, and 10ns, in much the same way as you force the enable line to the decoder.

- Force the waveforms at the other decoder inputs. You sketched what these waveforms should look like in the preliminary. You should be able to create the appropriate waveforms in much the same way as you created the clock.

**d) Run your simulation** using the command:

\[
\text{run xxxx}
\]

where XXXX is the length of the simulation in nanoseconds. From the sketch of the input waveforms you made in the preliminary, determine how long your simulation should run to cycle through all your inputs, given that your clock has a period of 25ns. To enter the run command, simply start typing while in the trace or schematic window.

16. Verify the output from your design is correct, by comparing the simulated waveforms to the waveforms you sketched in the preliminary. Be sure that the output only changes after a rising edge on the clock signal. All other times the output should be constant. Print out a copy of your results. Exit Quicksim when you are finished.

Don’t worry if your design fails the first time – that’s part of the reason for simulation. If you have troubles, edit your design and recreate the design viewpoint using pld_dve in a separate command window. Reset the simulation by selecting the QuickSim reset button on the right of the QuickSim window. When a window prompts you to save results, select the button stating that you do not wish to save. Then, re-run the simulation.
17. Now that you are finished with simulation, test your design in hardware using the FPGA on the XS40 board. Make the FPGA bit-file on your Unix workstation with the command:

```
xmake xc4005 xc4000xl xc4005xl-1-pc84
```

This command creates a bit file, xc4005.bit, which can be downloaded into an FPGA.

18. Create a directory on the Windows-based lab computer and place your FPGA bit-file in it. You may copy your bit-file there from a floppy or may download it using an ftp program if your PC is networked to your Unix workstation.

19. Connect the XS40 board to the PC via the parallel port connector.

20. Apply power to the XS40 board through the 9V wall transformer.

21. Use the Windows program GXSLOAD to download your bit-file onto the XS40 board. The bit file may be “dragged” there using the mouse or may be loaded using the menu options.

22. Use the Windows program GXSPORT to toggle the parallel port data lines. In step 11, you connected many of the devices in your design to the parallel port and in step 16 you verified your design through simulation by toggling the voltage level on these parallel port lines. GXSPORT now allows you to toggle the voltage levels for real. To toggle them, select the appropriate line with the mouse and hit “strobe”. Toggle the lines as you did during the QuickSim simulation in step 15, verifying the output at the seven-segment display for all possible combinations of inputs to the 3-to-8-bit decoder\(^2\). Write the results in your lab book. Do you feel comfortable that your design works? Why/why not? Show and explain your results to your lab instructor.

23. Once hardware verification is complete, delete the bit-file and directory you created on the lab computer.

**QUESTIONS:**

1. Hardware simulation is useful for proving hardware in a quick and efficient manner before actually building the hardware prototype. If your design had not worked in hardware but worked in simulation, it was probably because some condition occurred that was not considered by your simulation tool. Outline a process you might use to debug your circuit if it had not worked in the final verification stage. Your outline does not have to be perfect, but should draw on your past engineering experience.

2. The schematic of the XS40 board will be used over and over again during these labs. Using the schematic, identify which J3 pin you would use to connect to port 2 bit 5. Which pin of the FPGA connects to the ALE signal from the 8031? Which J3 pins connect directly to port-pins of the 8031, without also connecting to the FPGA? Which pin is connected to parallel-port pin 4 (not the same as data line D4)?

3. What 2 Unix commands should you always run before starting Falcon Framework based tools?

4. To properly simulate a latch in these labs, you must set an internal control signal. What is that control signal and what value(s) do you set it to?

5. You were not actually able to test the output of bit 7 of your latch in hardware. Sketch a hardware design that would output a unique pattern on the seven-segment display when bit 7 is active, while leaving the outputs for the other seven bits unchanged.

---

\(^2\) If you have trouble and cannot track down the cause in your design, you might consider running GXSTEST to ensure the XS40 board is operating correctly.