

**EXPERIMENT 2
FUNDAMENTALS**

1. Goals :

Learn how to develop a 2-to-1 multiplexer (MUX) during which the following are introduced :

- > schematic design on Xilinx software,
- > how to simulate and verify logic circuits,
- > the Digilent XLA FPGA board, and
- > how to test circuits on FPGA chips

2. Lab Work :

Follow first two of three major digital product development cycles given below. These two development cycles are the **development cycle on computers**, and the **development cycle on breadboards with FPGA chips**. The three development cycles will later be described in detail in a handout titled “*Digital Product Development.*”

2.1 Development Cycle on Paper/Computers :

The development cycle on computers consists of three steps :

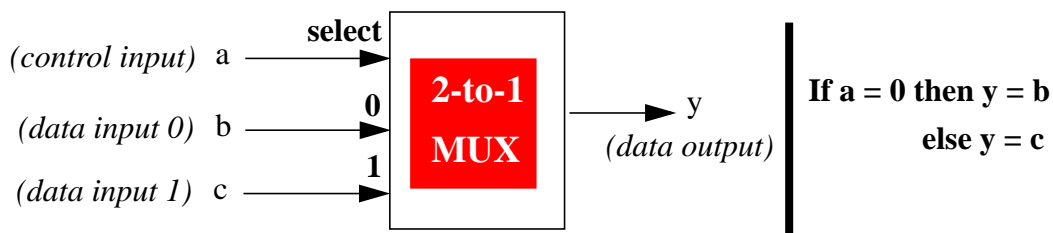
- > logic design
 - > input/output relationship
 - > implementation (the schematic)
- > test
- > modify

The logic design consists of obtaining the *precise input-output relationship* and then the *implementation*. The input/output relationship means we know exactly what happens when. The implementation means we get the full schematic. The test is the *simulation* of the circuit on computers and modification is the *change of the schematic* based on test results. As discussed in the previous experiment, if the circuit is simple one completely designs it on paper. **There is no need to do block partitioning.** For the 2-to-1 MUX, it is the case. So, we design it on paper completely and then move the design to the computer.

2.1.1. Logic Design :

2.1.1.1. The Input-Output Relationship :

The digital circuit is viewed as a black box with only its inputs and outputs considered. Then, the outputs are related to the inputs. A 2-to-1 MUX has three inputs and one output. The black box view and input-output relationship of the MUX are as follows :



Input a is the select input (control input) which selects one of the other two inputs (data inputs). The selected data input is connected to output y. Thus, at any moment, the output is equal to one of the two data inputs.

In order to understand the black box view and the input-output relationship better, inputs b and c are labeled as 0 and 1, respectively. It is because, the output is equal to b when a is 0 and equal to input c when a is 1. In fact, labeling

MUX inputs with numbers starting at 0 is a custom in data manuals.

The **textual** input/output relationship of the 2-to-1 MUX is then :

Output y is equal to b when a is 0 and equal to c when a is 1.

We can restate the textual relationship the following way as well :

Output y is 1 when a is 0 **AND** b is 1 **OR** when a is 1 **AND** c is 1.

This can be restated as :

Output y is 1 when a is **NOT** 1 **AND** b is 1 **OR** when a is 1 **AND** c is 1.

Another way to describe the input-output relationship is by means of a truth table that shows the value of the output for each input combination :

a	b	c	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

2.1.1.2. The implementation :

The implementation is about determining the components and their wiring based on given product goals : speed, cost, power consumption, reliability, size, weight, etc. For the sake of simplicity, throughout the semester, we will aim at a minimal implementation, meaning minimal number of components in the circuit.

In order to determine the components, first, the library of components (gates, flip-flops and Xilinx Design Blocks, XDBs) is searched to see if there is any component that immediately implements this circuit (the textual input-output relationship/the truth table). We realize that the library does contain a 2-to-1 MUX XDB, named **M2_1**. But, for the sake of getting more experience with gates and their wiring, we decide to implement the multiplexer by using gates, i.e. a **gate network**. So, we will **not** use that XDB.

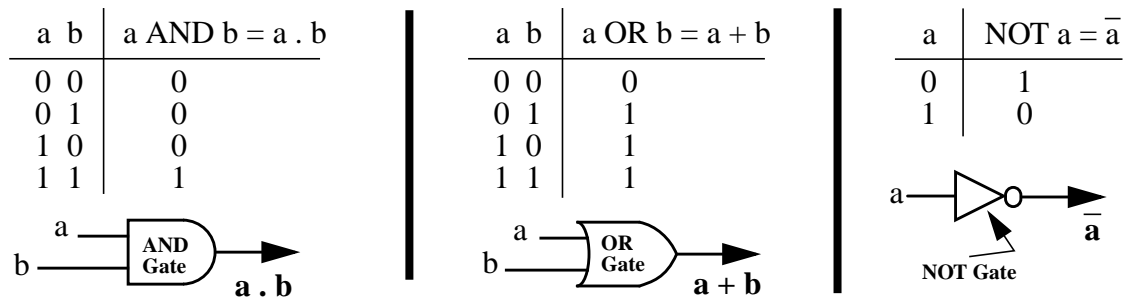
The gate network implements the input/output relationship (the textual one and also the truth table). It is **one** gate network, because we know there is just **one** output. How can we get the gate network ? How can we determine the gates and their wiring ? First, we are given the same three types of gates (digital electronic circuits) : AND, OR and NOT. Second, as we did with the car alarm circuit, we can start with the textual description of the multiplexer :

Output y is 1 when a is **NOT** 1 **AND** b is 1 **OR** when a is 1 **AND** c is 1.

As before, the words in capital and bold face imply the gates we have to use : one NOT gate, two AND gates and one OR gate. Next, how do we determine the gate connections ? The text has two sentences, forming a compound statement by means of the word OR. The first sentence has an AND connective relating input a to input b. Also, input a is in the negative form. In the second sentence connective AND is used to relate input a to c. In terms of truth-functional calculus where we concentrate on input and output values equal to 1, the expression for the multiplexer is :

$$y = ((\bar{a}) \wedge b) \vee (a \wedge c)$$

Note that for digital circuit design, and especially for complex circuits, we use another field of Mathematics, **Switching Theory**, to derive expressions. We are currently studying Switching Theory during lectures where the three gates are defined as follows :

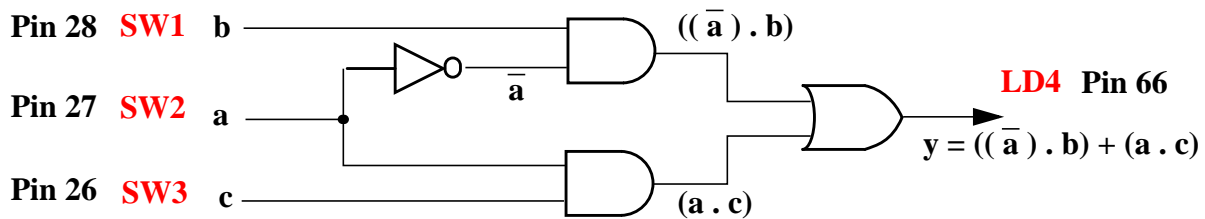


The dot symbol is for the AND gate, the plus symbol is for the OR gate and the overline is for the NOT gate. There are other names used for the NOT gate, including “**Inverter**” and “**Complement**.” The Xilinx software calls it “**INV**,” meaning inverter.”

Now, the Switching Algebra expression that precisely describes the 2-to-1 multiplexer circuit :

$$y = ((\bar{a}) \cdot b) + (a \cdot c)$$

The gate network for the 2-to-1 multiplexer is then as follows :



We will later see how one can convert the switching expression to a gate network in more detail. The reason why this expression is correct (satisfies the textual description and the truth table) can be shown by assigning the two possible values to input a :

- 1) When $a = 0 \Rightarrow y = ((\bar{0}) \cdot b) + (0 \cdot c) = ((\bar{0}) \cdot b) + (0 \cdot c) = (1 \cdot b) + (0) = b + 0 = b$
- 2) When $a = 1 \Rightarrow y = ((\bar{1}) \cdot b) + (1 \cdot c) = ((\bar{1}) \cdot b) + (1 \cdot c) = (0 \cdot b) + (c) = 0 + c = c$

Clearly, the switching expression does satisfy (implements) the input-output relationship. Note that Switching Theory allows us to derive minimal expressions (minimal gate networks or minimal hardware) starting with a truth table, not a textual relationship. This process which takes less time will be introduced later.

We are done with the design of the multiplexer on paper and now ready to move the design to the computer. Note that for large circuits, we do not do the design on paper. Only a few initial design steps are done on paper : the black box is partitioned into smaller and smaller (sub)blocks (**block partitioning**). Then, the component determination and wiring are carried out on computers. We will practice this frequently later in the semester.

2.1.1.2.1. Circuit Design on the Computer (Xilinx Software) :

Follow the steps given below to develop the circuit. Note that throughout the semester, the steps to go through will be given where the “bullet” symbol “>” shows a new step. Key presses and mouse selections are shown in bold.

Also, throughout the semester, the handouts will be prepared by assuming that students use their **S drive** as their project storage space. If students are not able to access the S drive, for example, in a place where no network connection is possible, they should use the default Xilinx project directory which has the path “...\\Fndtn\\active\\projects\\...” Later, students should move their project to the S drive when they establish the connection.

Another convention to remember is that when we say “click the mouse,” we mean clicking the **left** mouse button. The right mouse click will be explicitly mentioned.

First task before starting a new project :

In order to keep our designs organized, we need to have them stored with respect to their experiment number. For each experiment, we will have a directory on the S drive. Therefore, for the current experiment, we will do the following :

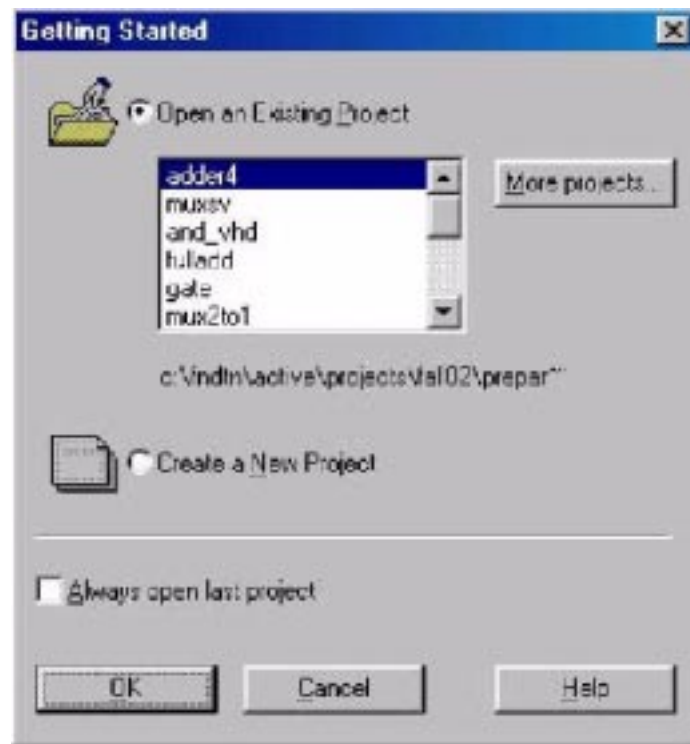
- By using “**My Computer**” create a new directory named “**exp2**” under cs2204.

We will have our multiplexer circuit under this exp2 directory.

Starting the Xilinx software :

- On your PC, start the Xilinx Project Manager by doing one of the two below :
 - Double click on the Xilinx “**Project Manager**” icon,
 - Go through menu selections : **Start** (on the lower left corner of Microsoft Windows) -> **Programs** -> **Xilinx Foundation 4.2** -> **Project Manager**.

You will see “**Getting Started**” window in the foreground and “**Project Manager**” window in the background :

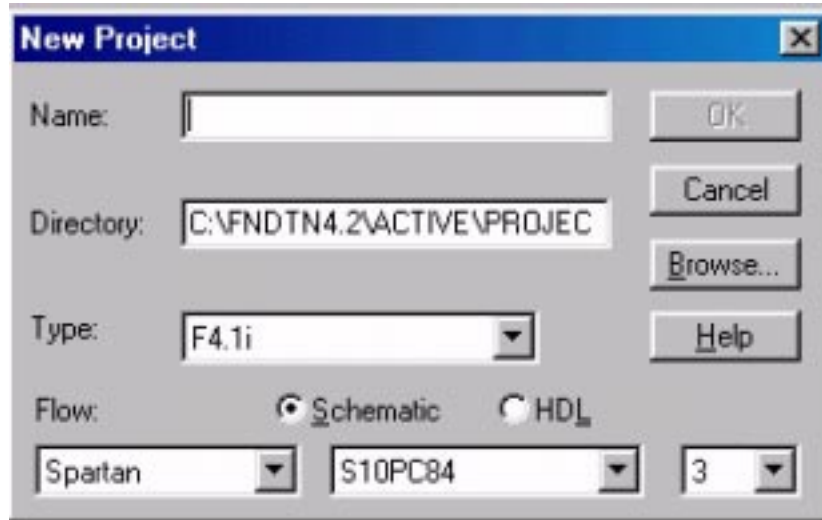


Creating a New Project :

Since we have not designed the project yet, we will start with creating a new project.

- In the “**Getting Started**” dialogue box, click on the radio button with label “**Create a New project.**”
- Click “**OK**”

A dialogue box with the name “**New Project**” will pop up :



- In the dialogue box, enter “**mux2to1**” as the name of your design.

We will change the directory of the project to the “**exp2**” directory on the S drive.

- Click on the “**Browse...**” button and select the project directory (your working directory) as the exp2 directory. All your 2-to-1 multiplexer files will be saved there.
- Click on the radio button with label “**Schematic**” since we will have a schematic design, not an HDL design.

Finally, there are three list boxes on the last row. They are about the FPGA chip specifically. We have to make sure that the choices are “**Spartan**,” “**S10PC84**,” and “**3**.”

- Click on the arrow in the “**Flow**” area and scroll until you see “**Spartan**.” Select it.
- Click on the arrow in the area to the right and scroll down until you see “**S10PC84**.” Select it.
- Click on the arrow in the area to the right and select speed “**3**” for the FPGA.
- Press “**OK**.”

The computer now shows the “**Project Manager**” window with three panels as shown below. The upper left panel displays the project file hierarchy, the upper right one shows the “flow” the project follows until it is completed. Finally, on the bottom, there are notes (actions, error messages and warnings) that the software places as it goes through the phases of the flow. From time to time, students have to read these notes to get a sense of what the software is doing during the circuit development.

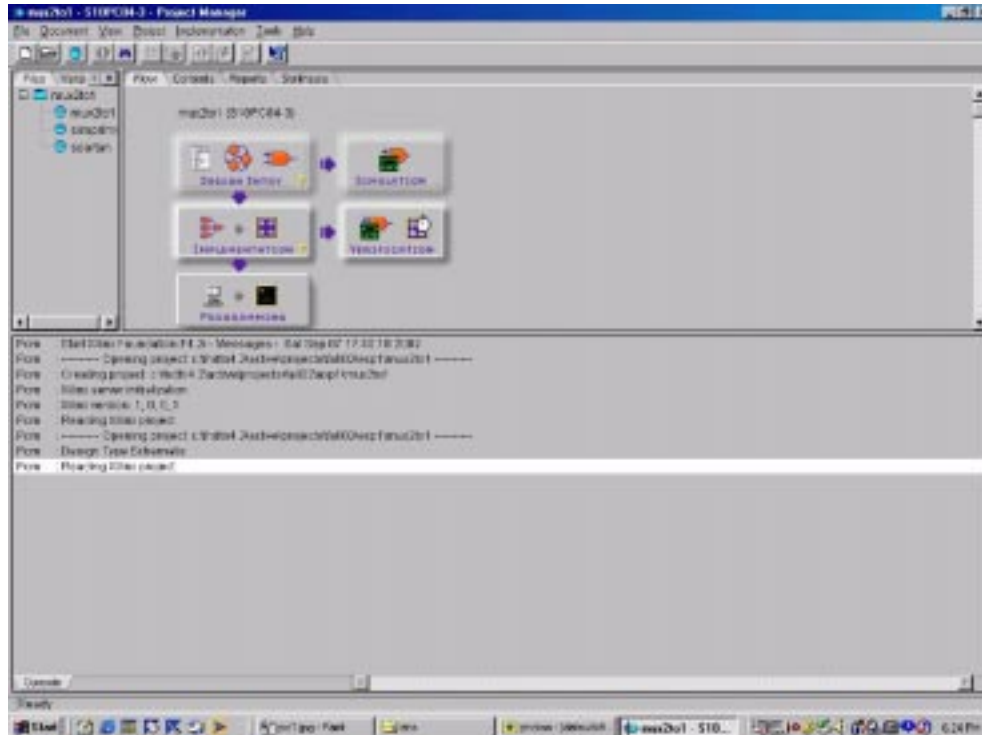
Let’s discuss the three panels in more detail :

○ The upper left panel shows the files in your project and the versions. Right click on one of the listed items, say “**simprims**,” you can get more detailed information about the simsprism “**Library Manager**” window. This manager is not important for us to discuss at this point. We will close it.

○ The upper right panel is the work region. It shows the design in different phases of the flow. For example, one sequence of circuit development steps is the vertical progression as follows :

“Design Entry” -> “Implementation” -> “Programming.” This would be the case if the circuit had no errors, such as all components, connections and pin numbers are correct.

- ❖ The “**Design Entry**” lets you input your design by using either the “**HDL Editor**” or “**FSM Editor**” or the “**Schematic Editor**.” Note that we will use the “**Schematic Editor**” this semester.



- ❖ The “**Implementation**” converts the design (the schematic) to what is called the “**bit file**.” It is the bit file that is downloaded to the FPGA chip. During Implementation, the software places and routes the design components and wires based on the device (FPGA chip) model.
- ❖ The “**Programming**” downloads your design to the FPGA board, i.e. it programs the FPGA chip by downloading the bit file to the chip.

○ Two kinds of testing of the circuit on the computer are provided as you see on the upper right panel : “**Simulation**” and “**Verification**” :

- ❖ “**Simulation**” is used to check if your schematic design implements the required logic. It does not consider the chip that will be used. That is, it assumes **ideal components** are used. Thus, it just indicates if the logic is correct. This simulation is also called “**functional simulation**.”
- ❖ “**Verification**” has two options and so two buttons are provided. The one on the left is “**Timing Simulation**” and the one on the right is “**Timing Analyzer**.” The Timing Simulation is the one we will use this semester and considers the delay information derived in the “Implementation” step, i.e. the Timing Simulation is done after we go through the Implementation step which takes into account the FPGA chip. The Timing Simulation gives you a better sense if the FPGA chip will satisfy the application requirements. It considers the time component of the circuit ! Thus, it can give an idea about the speed of the circuit and can catch timing related errors in the design.

We are now ready to open the Schematic editor to begin the schematic design of the 2-to-1 multiplexer circuit.

The schematic design

➤ Click on the “**Schematic Editor**” button to start the schematic editor.

A blank “design sheet” (window) will be shown. The schematic editor is always in the “**Select and Drag**” mode when it is started.


Our first task on the schematic is entering our names (designers’ names) on the lower right corner of the schematic. The software has already prepared the template there.

- Completely zoom out by selecting “**Display**” -> “**Full Page.**”

You will notice the rectangular area which we will have our information placed in.

- Go through “**File**” -> “**Table Setup...**” You will be shown a dialogue box titled “**Edit Standard Table.**”
- On “**Line1:**” one of the teammates enters his/her name.
- On “**Line2:**” the other teammate enters his/her name.
- On “**Line3:**” the teammates section.
- Click “**OK.**”

In order to place components (such as an AND gate) on the design sheet, we need to change the mode to the “**Symbols**” mode. This can be done in three different ways :

- Click on the “**Symbols toolbox**” icon on the left side of the screen : 
- Pull down the menu “**Mode**” then select “**Symbols,**” or
- Press key “**F3.**”

The “**SC Symbols**” window appears on the right side. This is the library of components we will use this semester. Scroll down the list to become familiar with it :




Let’s place one of the two 2-input AND gates on the sheet :

- Click on the “**AND2**” component inside the window of the library list. Now an “AND2” gate is “attached” to your mouse and you can place it on the design sheet wherever you want.
- Move the mouse to the left and click it in the middle of the sheet.

Now we will give a name to the AND gate. Naming components is helpful for understanding and debugging of the circuit. It is also very critical for documentation. Documentation is necessary for engineers who cooperate on a design and also for those who would be working on the design in the future.

Do one of four steps below to return to the “**Select and Drag mode**” in which the naming will be performed :

- Double click in the small box in the upper left corner of the SC Symbols box, or

- Click on  or
- Press “**Esc**,”(Escape) or
- Go through “**M**ode” -> “**S**elect and **D**rag.”

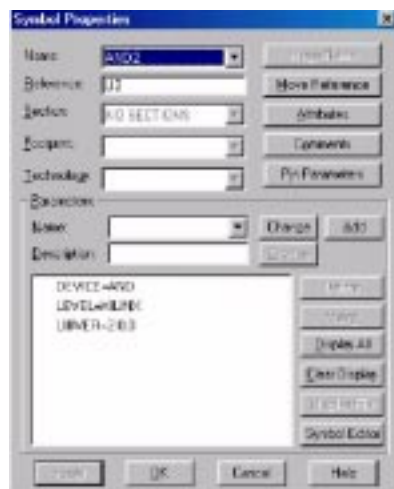
To name the AND gate do the following :

- Click the right mouse button on the AND gate.

A pop up menu will be shown next to the gate.

- Click on the first menu command “**S**ymbol **P**roperties...”

A window named “**S**ymbol **P**roperties” will be shown with a large number of choices :



Locate the second horizontal field on the left : the “**R**eference:” field.

- Replace the default name by entering “**U2**,”
- Click the “**OK**” button.


The name “**U2**” will appear just above the AND gate symbol. It is custom to give names as “**U1**,” “**U2**,” etc. to the gates, FFS and XDBs in schematic diagrams.

Repeat the above steps to pick, place and name components for the remaining gates :

- Place another 2-input AND gate below the first one and name it “**U3**.”
- Place a 2-input OR gate to the right of the AND gates and name it “**U4**.”
 - On the SC Symbols library, a 2-input OR gate is named “**OR2**” which is seen after considerable scrolling. In order to quickly select a component down the list, do the following :
 - Type in “**OR2**” on the bottom empty space of the “**SC Symbols**” window.
- Place a NOT gate (called “**INV**” by Xilinx) to the left of the top AND gate and name it “**U1**.”

Now, we will connect the symbols (components) by wiring them. We need to enter the “**D**raw **W**ires” mode which can be done in three different ways :

- Pull down the menu “**M**ode” then select “**D**raw **W**ires,” or

- Press key “**F4**,” or
- Click on the icon named “Draw wires” on the left edge of the screen : 

Now, to wire the components :

- Click on one side of a gate and move the mouse (as it is stretched by the software) to the appropriate gate and click on the appropriate side to attach the wire.
- To make sure the wire is connecting the two components, switch to the “**Select and Drag**” mode.
- Click on one of the wired components and drag it. If the wire is stretched with it, there is a connection. Do the same thing on the other component. If the wire has a *blue dot* at the touching point of a component, it means the wire is **not** actually making contact with the component.

Wire all the gates to get the gate network shown on page 3. The circuit is ready ! But, in order to download the circuit to the FPGA chip, we need to attach **input buffers** and **output buffers** and also **input pads** and **output pads**. We have to attach these components to be able to access the gates from the external world. *Note that these devices would not be needed if FPGA chips were not used.*

Start with the “SC Symbols” window and continue with the following :

- Connect an “**IBUF**” device to the input of the top AND gate selected from the “**SC Symbols**” window. “**IBUF**” stands for “Input Buffer.” Thus, the position of the IBUF is to the left of the gate.
- Press key “**F2**” to switch to the “**Select and Drag**” mode and name this input buffer “**B_BUF**” as we did for the AND gate.

Repeat this for the other two inputs and name the IBUFs as “**A_BUF**” and “**C_BUF**.”

- Connect an “**IPAD**” device to the input of the top IBUF selected from the “**SC Symbols**” window. “**IPAD**” stands for “Input Pad.” Thus, the position of the IPAD is to the left of B_BUF.
- Press key “**F2**” to switch to the “**Select and Drag**” mode and name this input pad “**B_PAD**.”

Repeat this for the other two inputs, by attaching two IPADs and name them as “**A_PAD**” and “**C_PAD**.”

- Connect an “**OBUF**” device to the output of the OR gate selected from the “**SC Symbols**” window. “**OBUF**” stands for “Output Buffer.” Thus, the position of the OBUF is to the right of the gate.
- Connect an “**OPAD**” device to the output of the OBUF selected from the “**SC Symbols**” window. “**OPAD**” stands for “Output Pad.” Thus, the position of the OPAD is to the right of the OBUF.
- Press key “**F2**” to switch to the “**Select and Drag**” mode and name this output pad “**Y_PAD**.”

We will now name the wires. Note that another name used for wires is “**net**.” Those wires whose values we will check during our simulations **must** be named. It is advisable to name as many other wires as possible for documentation purposes. But, often time is a limiting factor. Also, too many names in the schematic would be confusing. Finding appropriate names becomes hard and typing mistakes are quite likely. Thus, it is left to students’ judgement to decide which other wires should be named. Also, if simulations show errors, you will name more wires to observe their values. During these namings, you need to have a naming convention in order to be consistent and not to be confusing. We will use the convention below :

- Press key “**F2**” to switch to the “**Select and Drag**” mode.
- Right click on the net (wire) connecting the **B_PAD** and the **B_BUF**.
- Select the menu command “**Net Properties...**” and name this net “**B_NP2B**,” where the “**N**” in the name is for “net” and “**P2B**” means “Pad to Buffer.”

Repeat this for the other wires connecting pads and buffers as “**A_NP2B**,” “**C_NP2B**,” and “**Y_NB2P**.”

Next, we have to indicate to the Xilinx software which pin of the FPGA chip is connected to which IPAD or OPAD.

This operation is called “**binding**” the input and output PADS to the PINs. If you study the picture of the FPGA board or the PINout table of the demo board, you will realize that some FPGA PINS are permanently connected to **toggle switches, push buttons, 7-segment display and LED lights**. What we would like to do is to connect the three IPADs to three toggle switches and the OPAD to a LED light.

A toggle switch generates 1 or 0, depending on the position of its handle. A LED light gives off light when it is applied 1. We arbitrarily decide to use toggle switches named SW1, SW2 and SW3 which are connected to PINs 28, 27 and 26 of the FPGA chip, respectively. That is, we will use PIN 28, PIN 27 and PIN 26 as the a, b and c inputs, respectively. We also decide arbitrarily to use LED light named LD4 to show the output value of the AND gate. This light is connected to PIN 66 of the FPGA chip. Thus, we will use PIN 66 as output y :

- Press “**F2**” to enter the “**Select and Drag**” mode.
- Right click on “**A_PAD**”.
- Select “**Symbol Properties...**”

Locate the “**Parameters:**” section of the Symbol Properties window which is the lower part of the window.

- In the “**Name:**” list box (again inside the “**Parameters:**” panel), select “**LOC,**”
- In the “**Description:**” field type in “**p28.**”
- Click on the “**Add**” button after which “**## LOC=P28**” appears in the box under the “**Description:**” field.
- Press the “**OK**” button. Now this input PAD is bound to PIN 28 of the FPGA chip.

You will see “**LOC=P28**” just below the IPAD.

Bind the other input PADS and the output PAD to PIN 27, PIN 26 and PIN 66, respectively.

The schematic design is over ! We now have to save our design in the LABS domain :

- Go through “**File**” -> “**Save.**”

We then minimize “**Schematic Editor**” and return to the “**Project Manager**” window. You will see the “?” mark next to the name of schematic design file : “**mux2to11.sch.**”

If we summarize the design steps we went through :

- We select the logic components from the SC Symbols window and connect them by using nets (wires). We name these components and wires.
- We connect input and output buffers to each input and output wires then connect input and output PADS to the buffers. We also name them.
- We then bind the PADS to the FPGA pins.

Note that besides explicit documentation that helps a reader, the designer can help more by just simply drawing a schematic that “looks good to the eye.” We call it **beautifying** the circuit. This means components that do similar work form horizontal and vertical lines, line tanglings are minimized and there is enough space between components to identify circuits, subcircuits, and so on.

We are now ready to start the next step of the development cycle on computers : the simulation. This step will show if we have any logic and connection problem in our circuit.

2.1.2. Test via Functional Simulation :

We must have the Project Manager with three panels on the screen. We locate the large “Simulation” button on the upper right panel.

- Click on the “**Simulation**” button to enter the logic simulator. If a pop up menu warns that the “**Schematic netlist is older than...**” click “**Yes.**” Maximize the “**Logic Simulator**” window.

- Go through “**Signal**” -> “**Add Signals...**” to select signals. That is, we will select input signals to assign values to and an output signal to observe its value.

You will see three adjacent panels with names “**Signals Selection,**” “**Chip Selection**” and “**Scan Hierarchy.**”

- In the “**Signals Selection**” panel, double-click on signals named “**A_NP2B,**” “**B_NP2B,**” “**C_NP2B,**” and “**Y_NB2P.**”
- Then close the Signals Selection window, by clicking on the “**Close**” button right below the “**Signals Selection**” panel. Be careful not to close the “**Logic Simulator**” window.

Now the selected signals are listed on the first column. The software requires all input signals be renamed :



- Click on signal “**A_NP2B**” which will be highlighted.
- Make selections “**Signal**” -> “**Add Stimulators...**” A keyboard will be shown on the screen.
- Click key “**q**” on the keyboard. Stimulator “q” is assigned to signal “**A_NP2B.**” This selection of name is completely arbitrary.
- Click on signal “**B_NP2B**” which will be highlighted.
- Click key “**w**” on the keyboard. Stimulator “w” is assigned to signal “**B_NP2B.**”
- Click on signal “**C_NP2B**” which will be highlighted.
- Click key “**e**” on the keyboard. Stimulator “e” is assigned to signal “**C_NP2B.**”
- Close the keyboard by clicking on the “**Close**” button.

We have assigned names to the inputs (stimulators). We will **not** do this for the outputs.

- Click on “**q**” in red color to toggle the logic value of signal “**A_NP2B.**” You will see the short line next to the character “q” moving up and down (toggling) with each press on “q.”

We can now simulate the circuit for any input combination :

Set the logic value of signals “**A_NP2B,**” “**B_NP2B**” and “**C_NP2B**” to all 0, by clicking on “**q,**” “**w**” and “**e,**” appropriately.

- Click the  button located just below the “**View**” menu name on top and watch the waveforms of input and output signals. The output value, “**Y_NB2P**” should show 0. Click the  button several times to comfortably observe that it is 0.

Repeat this simulation for all other input combinations “**001,**” “**010,**” “**011,**” “**100,**” etc. to make sure your design works right.

If all input combinations result in correct outputs, we end the functional simulation step :

- Close “**Logic Simulator**” window to return to the Project Manager.

If the waveform does not match your expected result, you will need to go back to your schematic design to check it. Often, the common problem here is that some wires may not make contact with the components.

An easy way to see if there is any wiring problem, do the following on the schematic editor window :

- Select “**Options**” -> “**Integrity Test.**”

You would receive a message “**Integrity test passed successfully**” if there is no problem.

Another way to see if there is a wiring error in the circuit is by doing the “**Implementation**” step of the Xilinx software on the Project Manager. Recall that this step is needed to obtain the “**bit file.**” If there are errors, depending on the severity of the error, the Implementation is stopped. Whether the Implementation is stopped or not, messages are

generated in the “**Implementation Log File**” in the form of error messages and warning messages. All error messages have to be resolved. Among the warning messages, “**logical net xyz has multiple drivers**” and “**logical net xyz has no driver**” must be also resolved. Driver means output. A “Multiple Drivers” message means there are two or more outputs with the same name. The “No Driver” message means there is no input signal connected therefore the output cannot be generated. The “**logical net xyz has no load**” warning means the output is not connected to any input. That is, the output is not used. In some cases, some outputs are not needed and so not used. In other cases, by mistake the output is not used and this has to be corrected. The term project at the course web site now has six “no load” warnings which are fine. We do not need these six outputs.

- Starting with the “**Project Manager**” window and by following “**Reports**” -> “**Implementation Log Files**,” one can read the “Implementation Log File” and see where the problems are.

If we summarize the simulation steps we went through :

- We select signals whose values we want to observe.
- We add stimulators to inputs to assign values to them.
- We observe input and output waveforms.

2.2. Development Cycle on Breadboards with FPGAs :

As described in the *Xilinx Implementation and FPGA Programming* handout, go through the following cycles :

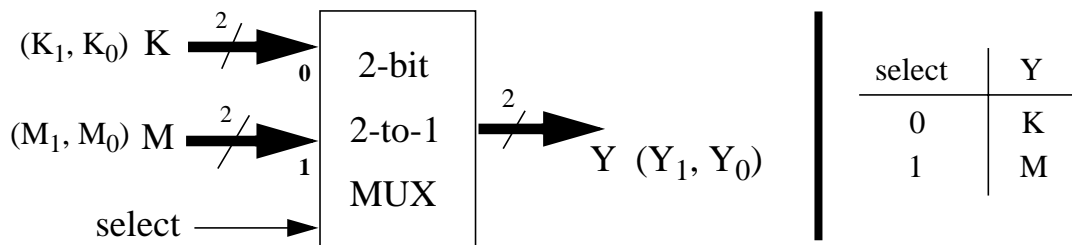
- Do a **Xilinx Implementation** of the design, then perform **timing simulations** and then **Download** the design to the FPGA chip.
- In order to do timing simulations, the simulator is started as described above. The simulator is in the “**Functional**” simulation mode which is shown in an area near the top of the simulator window. Click on the selection arrow and select the “**Timing**” option to change the mode.
- Test the design on the FPGA board. Modify the schematic design if errors are encountered.

EXERCISES :

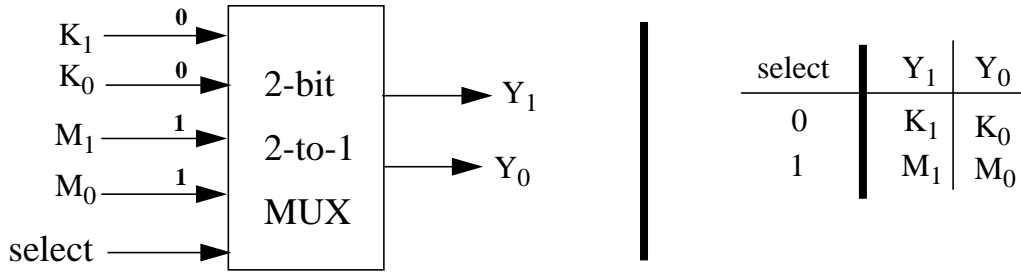
1) Develop a 2-bit 2-to-1 MUX circuit implemented by two gate networks. Name this project “**mux2b2t1.**” Appropriately name the components and wires. Beautify the schematic design. In order to design the 2-bit 2-to-1 MUX, study the logic design of the MUX below :

Input-output relationship :

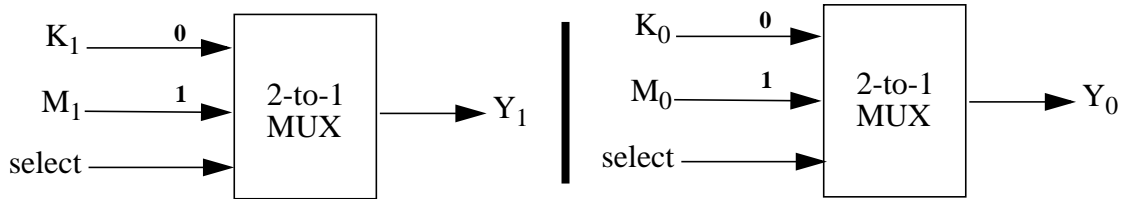
A 2-bit 2-to-1 MUX has 4 data inputs, 1 control input and 2 data outputs. Its black box view and the input-output relationship in the form of an **operation table** are given below :



Since the circuit is complex, with five inputs, we cannot use truth-functional calculus nor a truth table. For example, the truth table of this circuit would need 2^5 rows, 32 rows. It would be hard to obtain the expressions. Thus, one needs to divide the circuit into blocks and design the blocks separately. How can we divide the blocks ? If we consider the inputs and outputs in more detail, we would see the following :



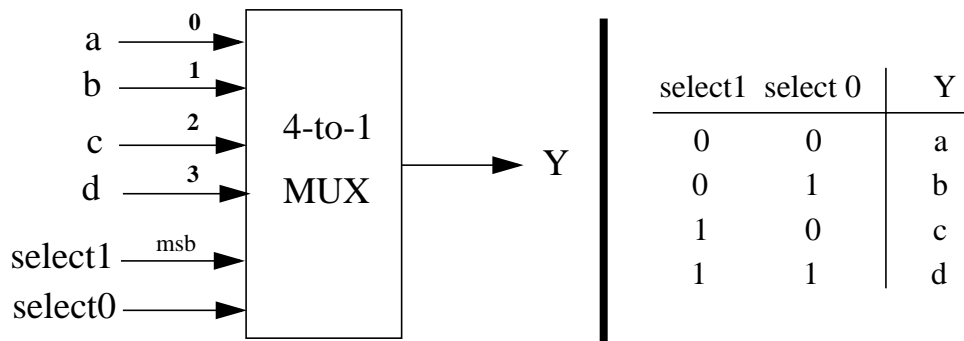
We see that we can partition the circuit into **two** identical blocks. Each block is a 2-to-1 MUX that we have just designed. That is, the MUX we have designed is in fact a 1-bit 2-to-1 MUX, but because of a convention, the “1-bit” part is not mentioned. Then, here are the two blocks of the circuit :



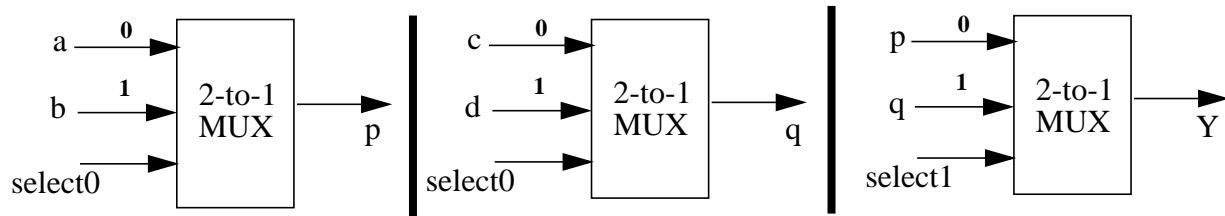
2) Develop a (1-bit) 4-to-1 MUX circuit and name this project “**mux4t1**.” Appropriately name the components and wires. Beautify the schematic design. In order to design the 4-to-1 MUX, study the logic design of the MUX below :

Input-output relationship :

A 4-to-1 MUX has 4 data inputs, 2 control inputs and 1 data output. Its black box view, the input-output relationship in the form of an operation table are given below :



The circuit is complex, with six inputs, we cannot use truth-functional calculus nor a truth table. For example, the truth table of this circuit has 2^6 rows, 64 rows. One needs to divide the circuit into blocks and design the blocks separately. How can we divide the blocks ? We see that we can perform the top two rows of the operation table in parallel with the bottom two rows by using select input, select0 and then we select one out of the two by using select1. Thus, we would use three 2-to-1 MUX circuits designed in this experiment :



3) By using the concepts introduced in Exercises 1 and 2, develop a 2-bit 4-to-1 MUX circuit. Name this project “**mux2b4t1**.” Appropriately name the components and wires. Beautify the schematic design.

CONTACTS :

1) Students can see the professor and teaching adjuncts (TAs), about the lectures, homework, and lab experiments.

2) Professor's contact information :

Room : 114 LC (718) 260-3101 Fax : (718) 260-3609 haldun@photon.poly.edu

Open-door policy to see the professor. If the door is closed, he might be in the lab.

Present in the lab : Mondays (4-6), Wednesdays (3-5) and Fridays (2-4)

3) Below are the contact information and assignments of the TAs :

Nikhil Joshi : 001LC, (718) 260-4011, njoshi01@utopia.poly.edu

- Present in lab session : Tuesday : 3-5:50 (B)

Sapan Shenoy : 102RH, (718) 260-3731, ssheno01@utopia.poly.edu

- Present in lab sessions : Tuesday : 6-8:50 (C), Thursday : 6-8:50 (A)

Jeff Tao : 233LC, (718) 260-3420, jefftao@photon.poly.edu

- Present in lab session : Wednesday : 6-8:50 (D)

Bo Yang : 001LC, (718) 260-4011, yangbo@photon.poly.edu

- Present in lab session : Wednesday : 6-8:50 (D)

- Grading the homework

Peng Yao : 004LC, (718) 260-3975, yaopeng@utopia.poly.edu

- Present in lab sessions : Tuesday : 3-5:50 (B), Thursday : 6-8:50 (A)

4) All handout and lab files are at the course web site : <http://cis.poly.edu/cs2204>

5) Students are asked to give feedback about the announced lab hours in case the hours are not sufficient and/or need to be rescheduled.

6) When short-term problems are encountered in PC labs, students are advised to contact : help@duke.poly.edu or (718) 260- 3123 or go to Room : 337 RH.

For CS2204 lab related issues and to have the lab open, students need to contact the CIS lab supervisor Mr. Keni Yip at (718) 260-3023, keni@poly.edu. His office is 225RH

For longer-term problems in PC labs and **any other matter**, students should not hesitate to contact the professor and the TAs.