University of Illinois ECE 385

GENERAL GUIDE

I. LAB KITS
- Equipment list
- IC packages
- Using data sheets
- IC list
- TTL characteristics
  - Power
  - Logic levels
  - Loading

II. LAB DOCUMENTATION
- Overview
- Written description
- Component layout
- Logical diagram
- Standard device symbols

III. DESIGN TECHNIQUES FOR TTL
- General Considerations
- Delays and Glitches
- Use of NAND (NOR) Gates

IV. LAB TECHNIQUES
- Circuit assembly
- Wiring techniques
- Debugging techniques
- Light Emitting Diodes (LEDs)
- Toggle Switches
- Contact Bounce
- FPGA Clock Output and Switchbox

I. LAB KITS

Equipment List

The lab kits are checked out in a box to each group contain:

1) A protoboard

2) No. 22 wire kit in assorted colors and lengths.

3) Discrete components (resistors, capacitors, LEDs, switches).

4) About fifty assorted integrated circuit chips or packages.

Additional components checked out to each **individual** contain:

5) Urbana board and programming cable

Note that the breadboard and wiring kit may have been provided as part of a previous course (e.g., ECE 110 or ECE 210) – therefore you are responsible for making sure your equipment is in working order, or you will need to purchase another breadboard and wiring kit. Components 3-5 are provided exclusively for this course and will be checked out to you.

Integrated Circuit (IC) Packages

The ICs in the kit will be like the ones shown in Figure 1. Since the pins on newly manufactured IC packages are bowed outward for use with automatic insertion equipment, it may be necessary to gently bend the pins so they will go into an individual socket easily. This is best done by pushing each row of pins inward on a flat surface.

Consider the IC chip shown in Figure 1. The number SN7400N on the chip indicates the following: The SN is a prefix used by Texas Instruments, Inc., the 74 indicates the chip is from the 74-hundred family of Transistor-Transistor Logic (TTL), the 00 indicates the exact type of circuit. The suffix N gives information on the packaging (e.g., N for Plastic DIP, J for Ceramic DIP, DIP stands for Dual In-line Package) as well as the connection between the internal logic circuits and the external pins.

On some chips a few letters may appear between the family digits and the type digits, e.g., SN74LS00 and SN74S00. These letters are used to indicate that the chip differs electrically (but not logically) from standard TTL chips (e.g., LS stands for Low power Schottky, S for higher speed Schottky). Schottky means that the inputs of the gate have Schottky triggers, which provides some hysteresis to reduce the noise sensitivity. The digits 7523 are a manufacturing date and indicate the IC package was made in the 23rd week of '75 (an antique given the current iteration of this document!) The 54-hundred family provides the same functions as 74-hundred family, but with mil-spec extended operating temperature range and radiation hardening.

Some kits additionally have HC or ACT chips instead of TTL chips. HC standards for high-speed CMOS, whereas ACT stands for advanced CMOS technology. These two logic standards are compatible with standard TTL chips but are based off CMOS (complementary metal-oxide semiconductor) technology (similar to what is in modern integrated circuits) to be even lower power than "low-power" TTL. In addition, HC and ACT families are compatible with 3.3V logic levels and are capable of being powered by 3.3V, which is important when interfacing with modern devices. However, a designer must be aware of the potential issues when interfacing between different logic levels (we'll discuss this later in this document).

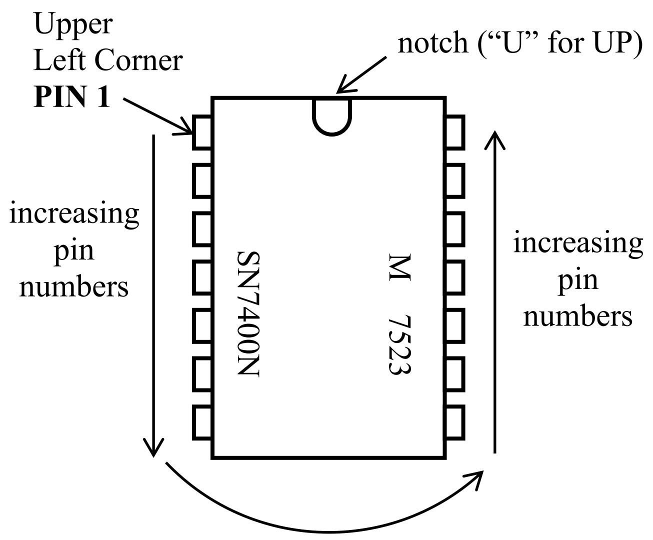
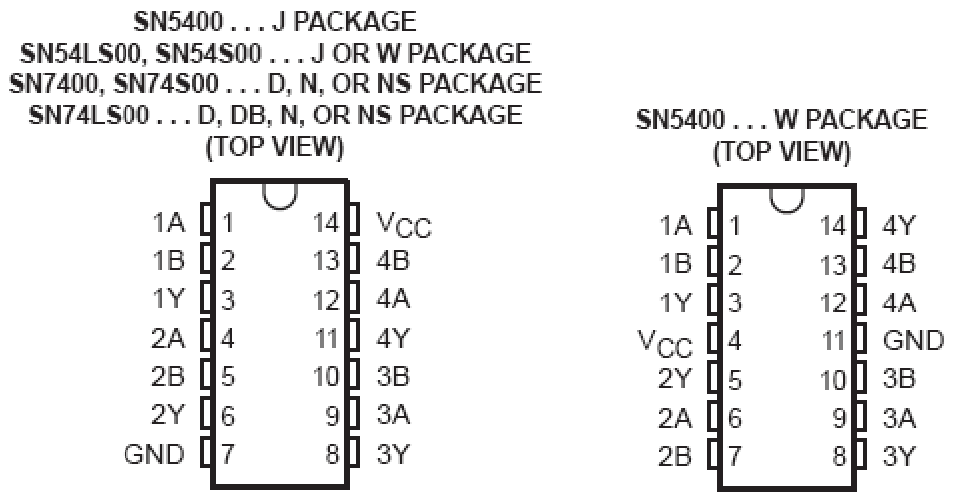


Figure 1 - 7400 IC

54/74 FAMILIES OF COMPATIBLE TTL CIRCUITS



These devices contain four independent 2-input NAND gates. The devices perform the Boolean function $Y = \overline{A \bullet B}$ or $Y = \overline{A} + \overline{B}$ in positive logic.

Figure 2 - Data Sheet for 54/7400

<u>Using Data Sheets</u>

A part of the data sheet for the 7400 package is reproduced from the Texas Instruments website (www.ti.com). The data sheet states that the chip contains four independent 2-input NAND gates, and gives Boolean function in variables A, B and Y. Each NAND is identified on the pins using these same variables with a distinct prefix. For example, 3A, 3B and 3Y. The data sheet also indicates that pin 7 must be grounded and pin 14 must be connected to power (V_{cc}). You will be able to find the data sheets for the chips we will be using on our course website (under Lab 1).

A standard lab kit of IC's consists of the following:

Quantity	Designator	IC Description
4	7400	Quad 2-input NAND
4	7402	Quad 2-input NOR
4	7404	Hex Inverter
4	7410	Triple 3-input NAND
4	7420	Double 4-input NAND
4	7427	Triple 3-input NOR
6	7474	Dual D Positive Edge Triggered Flip Flop
2	7485	4-bit Magnitude Comparator
4	7486	Quad 2-input Exclusive-OR
4	74109N	Dual JK Flip-Flops
4	74151N	8 to 1 Multiplexer
4	74153N	Dual 4 to 1 Multiplexer
4	74157N	Quad 2 to 1 Multiplexer
2	74161N	4-bit Synchronous Up-Down Counter
2	74163E	4-bit Synchronous Binary Counter
6	74194A/E	4-bit Bidirectional Universal Shift Register
6	74195E	4-bit Bidirectional Universal Shift Register
2	SN74LS279	Quad S-R Latch

Note that logic chips may be from the ACT or HC families and may have CD or SN prefixes, which used to designate the manufacturer: CD for RCA/Harris, SN for TI, but which has since become meaningless as the market consolidated.

DISCRETE COMPONENTS AND TOOLS

Quantity	Type
4	10 Light emitting diode DIP (single package)
10	330 Ω resistors (orange-orange-brown)
10	1 KΩ resistors (brown-black-red **or** brown-black-black-brown)
10	1uF capacitor
4	8 SPST switch DIP (single package)

2	4 SPDT switch DIP (single package)

Logic Gate Characteristics

Power:TTL IC's (74, 74F, 74LS, 74ALS) require a 5V power supply. CMOS ICs (74HC/T, 74AC/T) may run on either 5V or 3.3V. Each IC package has two power pins, one to be connected to GND (0 V) and one to be connected to V_{cc} (+5 V/ +3.3V). All IC's used in the laboratory should be mounted with the indentation or notch, which is located on one end of the IC package, positioned up (away from you). When an IC is positioned in this fashion (pin 1 in the upper left corner and the highest numbered pin in the upper right), most IC's will have their GND connection in the lower left corner and their V_{cc} connection in the upper right corner. There are a few exceptions, however, so always check the data sheet. Your Urbana board provides 3.3V from the 'PMOD connector' when powered through USB, be sure to connect both power (3.3V) and ground from your Urbana board to the protoboard using breadboard wires.

Logic Levels: Under the positive logic convention, low voltage inputs and outputs (from 0.0V to about 0.7 V) are interpreted as logical 0 and high voltage (from about 2 V to 5 V) as logical 1. To place logical 0 on an input, connect it to GND. To place logical 1 on an input, connect it to V_{cc} through a 1kΩ 1/4 W resistor. Although it is possible to connect a logical input directly to V_{cc}, having a 1kΩ resistor will prevent the destruction of the device if you misread the datasheet or miscount the pin number. One such resistor can generally be used to pull as many as eight inputs high. In addition, this pull-up resistor will limit input current during transients (e.g. rapid switching). A gate input which is disconnected is generally considered an unknown logic level. **Some** gates have weak internal pull-up resistors which will cause disconnected inputs to read as logical 1. However, if your circuit contains floating inputs, they are apt to be sensitive to noise**, therefore you should not rely on this behavior.** Unused clock, mode, clear and preset pins are particularly sensitive and **must** be tied to the appropriate logic level for reliable operation.

The Noise Immunity of a gate is defined as the maximum amplitude of a positive-going (noise) pulse added to the nominal logic "0" voltage level or a negative-going (noise) pulse added to the logic "1" voltage level at the input of a gate which does not cause the output of that gate to change its logic value. The nominal logic "0" and "1" voltage levels can be determined by connecting several inverter gates (e.g., 7404) in series and observing the voltage levels at the output of the last inverter when the input to the first inverter is at

GND and at +5v. What is the advantage of a larger noise immunity? Why is the last inverter observed rather than simply the first? Given a graph of output voltage (VOUT) vs. input voltage (VIN) for an inverter, how would you calculate the noise immunity for the inverter? See the following figure.

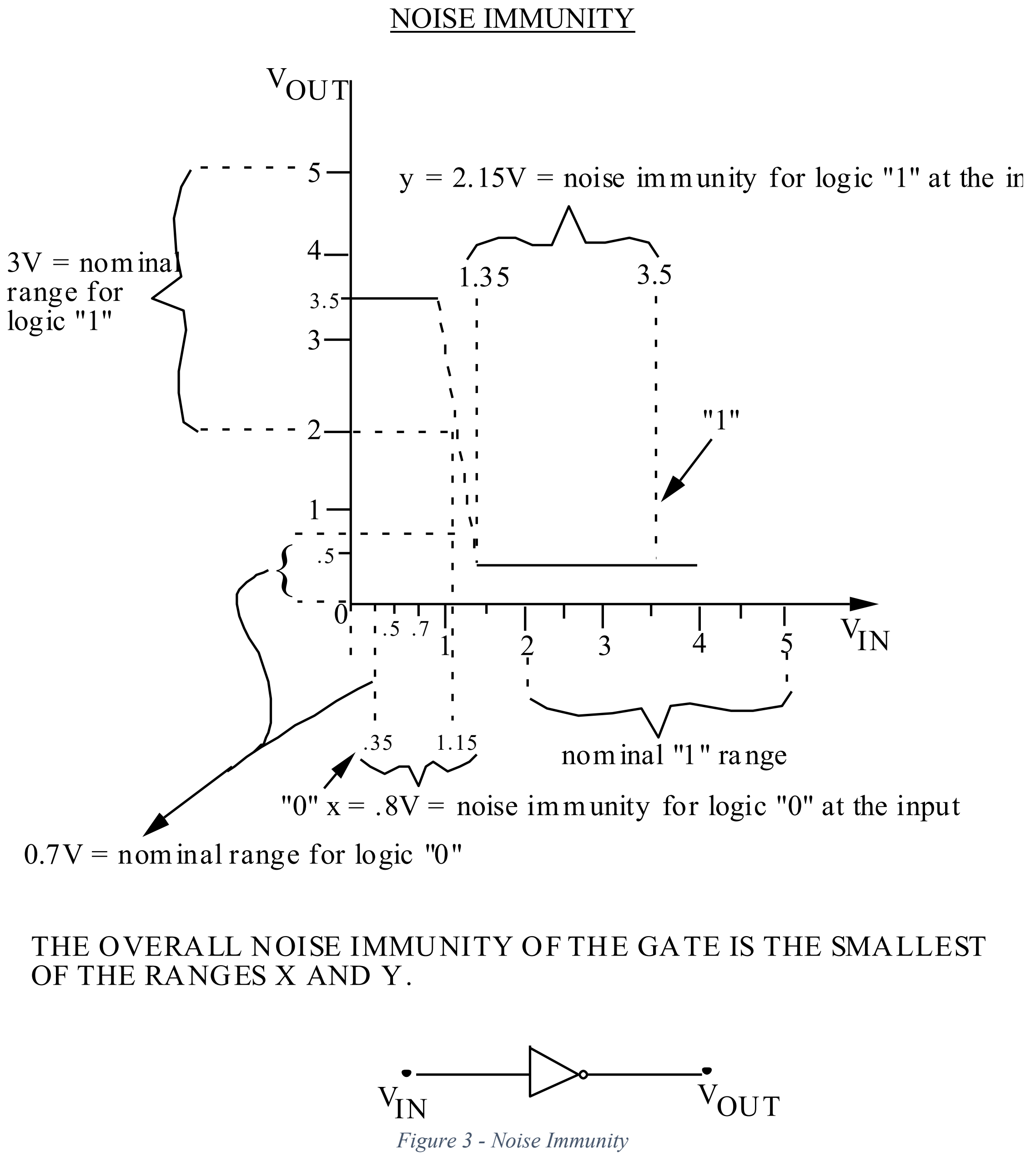
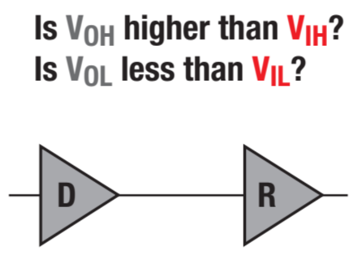
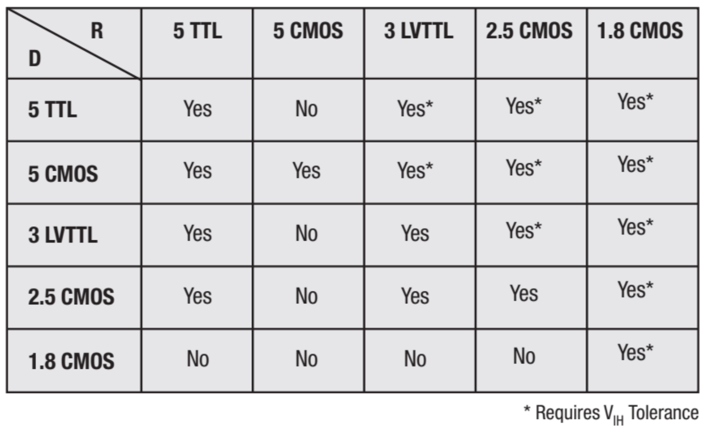

NOISE IMMUNITY
VOUT
5
4
3.5
3
2
1
.5
0
y = 2.15V = noise immunity for logic "1" at the in
1.35
3.5
3V = nominal range for logic "1"
"1"
.5
.7
1
2
3
4
5
VIN
.35
1.15
nominal "1" range
"0" x = .8V = noise immunity for logic "0" at the input
0.7V = nominal range for logic "0"
THE OVERALL NOISE IMMUNITY OF THE GATE IS THE SMALLEST OF THE RANGES X AND Y.
VIN
VOUT

Figure 3 - Noise Immunity

In the case of a mix between 5V and 3.3V logic, in general, 3.3V logic may be used to **drive** a 5V TTL input, but a 3.3V logic input **may not be 5V tolerant**. In particular, the 3.3V **logic inputs of your FPGA board are not 5V tolerant**, but a 3.3V output from the

FPGA board will be read correctly by your logic, whether it is powered by 3.3V or 5V. Note that students may wish to power the TTL designs via the Urbana FPGA board. This is an important distinction for digital system designers, and is summarized by this handy table courtesy of TI:


Is V_{OH} higher than V_{IH}?
Is V_{OL} less than V_{IL}?
D
R



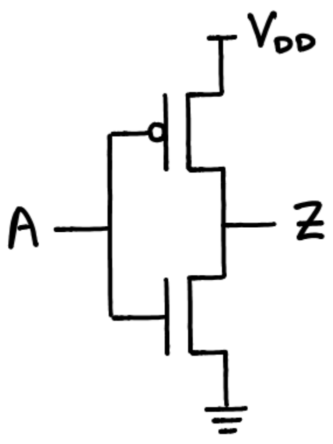
R / D	5 TTL	5 CMOS	3 LVTTL	2.5 CMOS	1.8 CMOS
5 TTL	Yes	No	Yes*	Yes*	Yes*
5 CMOS	Yes	Yes	Yes*	Yes*	Yes*
3 LVTTL	Yes	No	Yes	Yes*	Yes*
2.5 CMOS	Yes	No	Yes	Yes	Yes*
1.8 CMOS	No	No	No	No	Yes*

* Requires V_{IH} Tolerance

Figure 4 - Logic Level Compatibility (Courtesy of Texas Instruments)

Fan Out: Most TTL gate-outputs will drive a maximum of ten standard TTL inputs or ten "unit-loads" (i.e., they have a maximum fan-out of 10). Connection to more than ten gate-inputs may overload the output and produce unpredictable results. A few non-standard TTL outputs are not able to drive as many as ten unit-loads and a few non-standard TTL inputs are more than one unit-load. Check the datasheet about the fan-out if your design exhibits strange behavior - especially problematic are some counters and registers.

Decoupling: While TTL logic is robust to switching noise, CMOS logic requires decoupling to function reliably. It is good practice (and indeed often necessary) to place a 1uF or similar capacitor on each device connected between the power pin and ground. Recall that you saw the CMOS inverter circuit from the ECE 110 notes (courtesy of Prof. Schmitz). Why is a capacitor necessary close to each chip? Hint: what happens inside the CMOS circuit when it switches? Note that this is true even for complex ICs, your Urbana board has a myriad of decoupling capacitors underneath the main FPGA chip (the Spartan 7 chip in the middle of the board).



CMOS Inverter – ECE 110

A. <u>THE PROTOBOARD</u>

The protoboard (see Fig. 5) is a board on which all discrete logic circuits will be built. Power can be accessed by connecting two cables to two binding posts on the protoboard, +3.3V (Red) and GND (Black).

The socket strips are used to mount and interconnect components. Three socket strips for components and five bus strips which are useful for bussing commonly used signals such as power, ground, or clocks, are mounted vertically. Each strip is simply an array of holes with spring clips underneath them. Stripped #22-gauge solid wire, provided in the lab, must be used for interconnections in the Protoboard. **Be sure that the portion of the wire inserted is straight**. **If wire heavier than #20 gauge is forced into the holes, it will permanently damage the spring clips**. Any components with heavy gauge leads, e.g., large capacitors, resistors, crystal oscillators, etc. must be soldered into IC headers or soldered onto #22 solid hookup wire. In general, you will not need to do this in ECE 385.

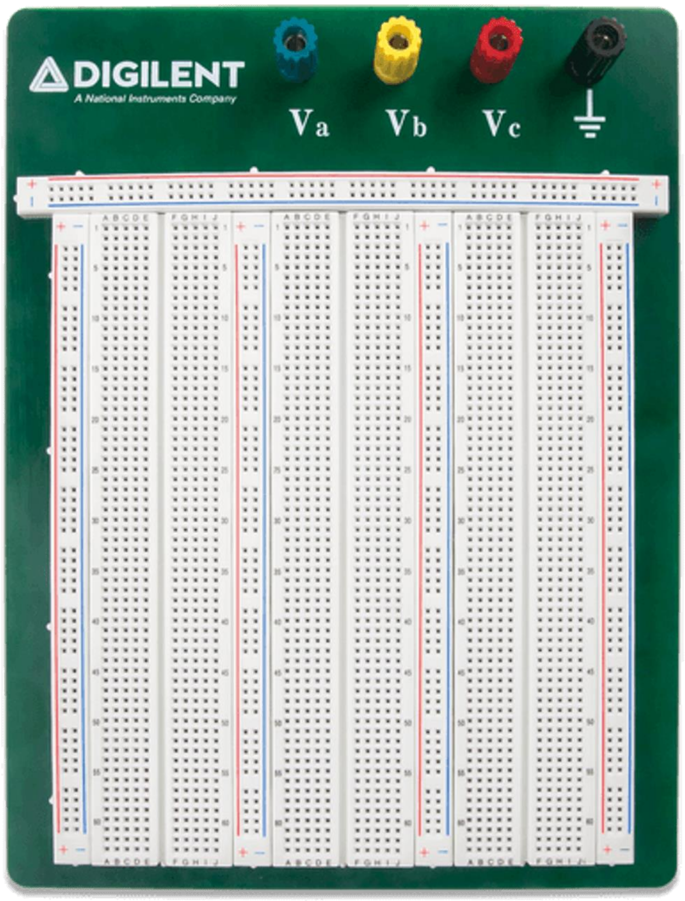
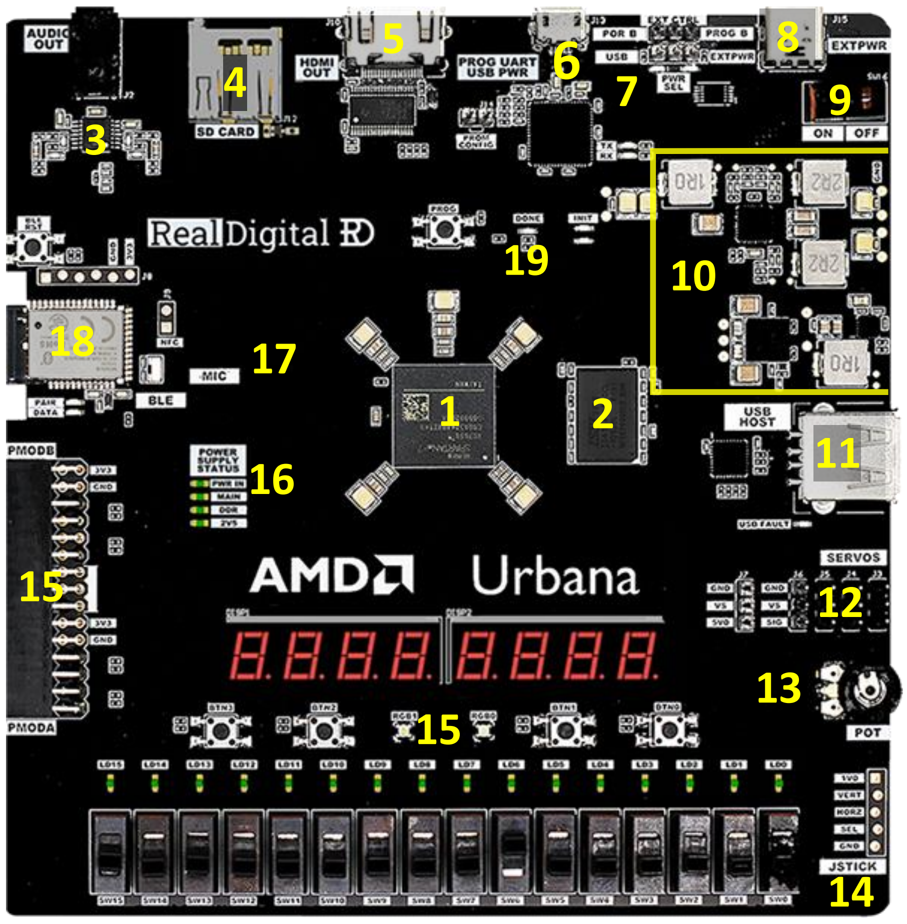


Figure 5 - Breadboard

D. Real Digital Urbana FPGA BOARD

Included with the lab kits at check-out time is a Real Digital Urbana board. This board should also come with a standard USB Type A- to micro-B cable (you must provide an adapter if your computer does not have traditional USB Type A ports – e.g., computers which only have USB-C).



1	Spartan-7 XC7S50-CSGA324 FPGA	10	Pow er supplies and control
2	1Gbit DDR3 memory (x16)	11	USB host connector and controller
3	Audio output circuit and 1/8" jack	12	Servo motor connectors
4	SD card socket	13	10K Potentiometer (connected to ADC)
5	HDMI source buffer and connector	14	Joystick connector
6	USB pow er/prog input and FTDI controller	15	GPIO (buttons, sw itches, LEDs)
7	Pow er and reset control select	16	Pow er supply status indicators
8	Optional external pow er jack (USBC)	17	PDM digital microphone
9	Main pow er sw itch	18	FPGA programming status and control

Figure 6 – Real Digital "Urbana" FPGA Board

III. LAB DOCUMENTATION

Overview

Precise and uniform documentation is an indispensable part of digital design. The documentation is simply a presentable record of your design procedure supplemented with a written description of the operation of the circuit. From your documentation, someone unfamiliar with the experiment should be able to understand what your circuit does and should be able to wire up and test your circuit without reference to any other handbooks or literature. Good documentation is essential for correct circuit-assembly and for circuit-examination by an instructor. **The documentation should reflect what was done in the lab and should not be considered an independent portion of the course.**

By default, a complete lab report should consist of an **Introduction** which states the purpose of the lab, **Documentation** for each circuit, **Answers** to the questions asked in the lab manual, and a **Conclusion** to wrap up the things learned.

Basic **Documentation** for a digital circuit designed should consist of

(1) A written description of the operation of the circuit and a block diagram

(2) Karnaugh maps, State Diagrams and Tables, and Boolean Equations

(3) A Logic Diagrams (preliminary AND-OR and final NAND implementations)

(4) Component Layout

Though the four parts have been listed in the correct order for a reasonable derivation of a completed design, the actual design process will probably proceed in a different order. For simple designs it is probably most convenient to draw a partial logic diagram, determine a component layout, complete the logic diagram, and finally supply some description of the circuit operation. More complicated designs will require completing some of the descriptive material (e.g., signal definitions, state transition diagrams) before a logic diagram can be started. Complicated designs will probably require moving back and forth between the design levels several times (e.g., the logic diagram may suggest a better set of signal definitions).

Written Description with Block Diagram

The ability to describe a circuit in a manner which is organized, precise, clear, and easily understood is a very important part of digital design. The amount and type of description needed will depend upon the complexity of the circuit. But almost always, a high-level functional block diagram is an integral part of any written description. The high-level block diagram should be the foundation of this section of your lab documentation but may not be complete by itself. Further elaboration (sub-diagrams or written descriptions) will likely be required.

Karnaugh Maps, State Transition Diagrams and Tables

Here you present your derivation of logic using formal techniques learnt in ECE 120. Often there are several alternative solutions. You should present your rationale for choosing an implementation.

Component-layout

A component layout is used to indicate which types of IC will be needed, where they will be mounted, which pins require power connections, where the input signals originate, and where output signals are monitored. This information is needed for circuit construction and is essential for circuit debugging when one needs to find the physical location of a given logic signal (net) on the logic diagram. Students may also use a physical breadboard drawing tool (such as Fritzing) to create this diagram.

A sample component layout is given as follows. Each switch and LED to be used should be labeled with the symbolic name for the associated digital signal (you do not need to draw switches or LEDs). The three IC socket strips are denoted A to C from left to right. IC packages are numbered 1, 2, 3… from top to bottom. (In general, seven ICs will fit from top to bottom, but differing package sizes may change this.) Thus, C2 denotes the second IC down on the third socket-strip from the left. Note that you should not need to use more than 21 ICs in a single design for this course. The component layout sheet should also show the power and ground connections for each chip. Using the standard orientation (notch up) most chips will have the ground connection in the "lower left corner" and VCC in the "upper right corner." However, you should always check the data sheet for power and ground pin numbers since some chips are non-standard. Note that internal signals are shown in smaller text (B1.3, B2.2 etc..) while inputs and outputs (A, B, and Z) are labeled in large text. Note that we label the internal signals, but do not draw them as wires.

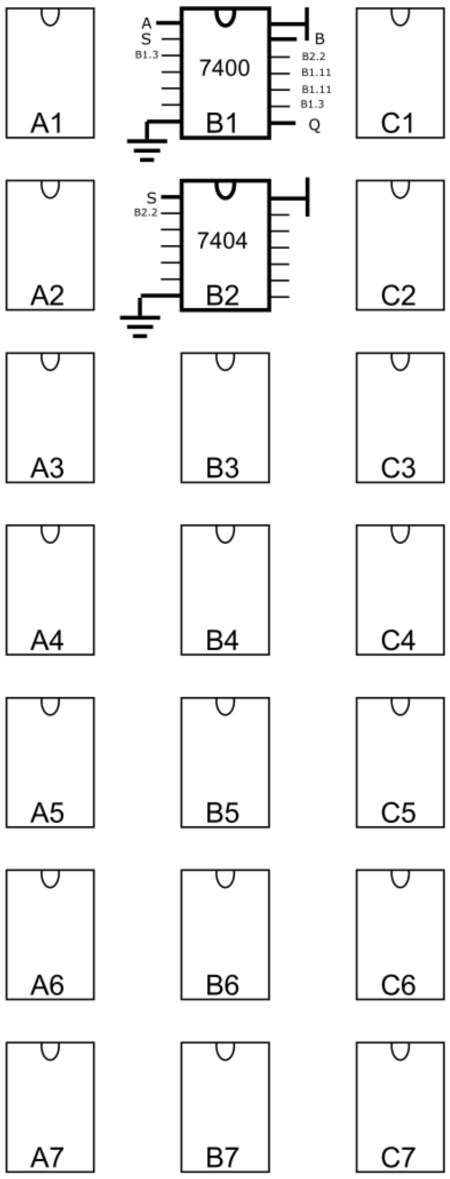


Figure 8 - Example Manual Component Layout

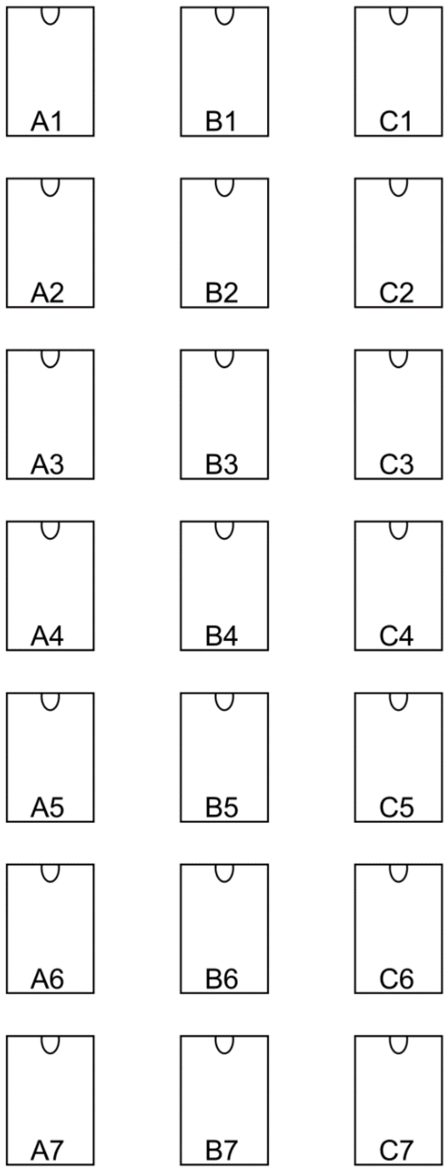


Figure 7 - Blank Manual Component Layout

Below is the same circuit, except as Breadboard sketch in Fritzing.

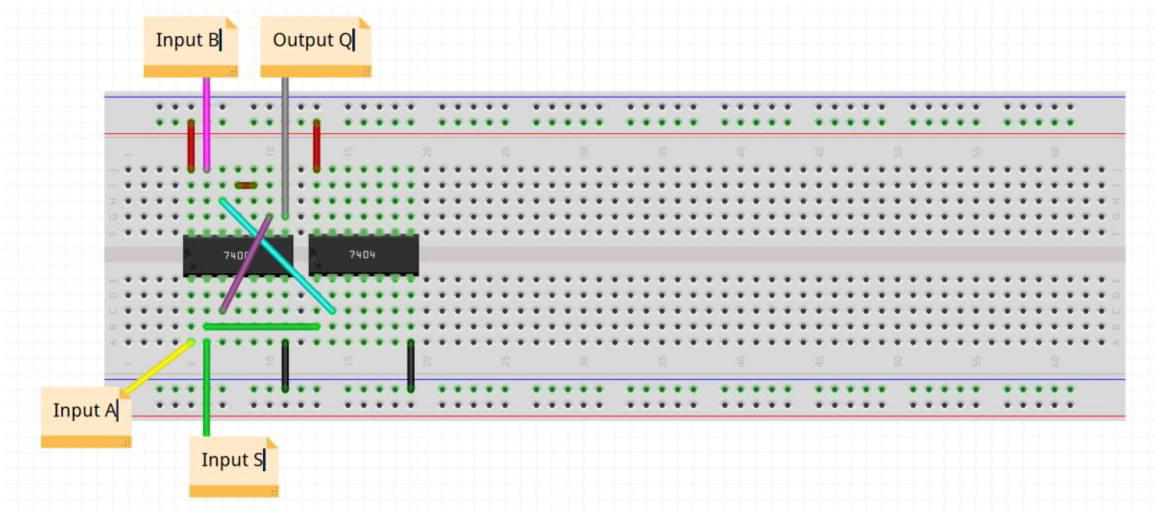


Figure 9 - Fritzing Component Layout

Note that the internal wires should be drawn instead of labeled, as the issue of overlapping wires is less of a concern when multiple colors may be used. As your circuits get more complicated, you may want to use colors to indicate a type of signal (e.g., MUX selects or data bits). In addition, inputs and outputs are simply labeled with a 'note', switches and LEDs do not explicitly need to be drawn (although those parts are available in Fritzing if you so desire).

Logic Diagram

A sample logic diagram is shown in Fig. 10. Each device (gate, flip-flop, etc.) is drawn using the standard logic symbols. Each device is labeled according to its IC location and IC pin numbers. I/O signals are given symbolic titles, written next to their corresponding switches/LEDs/Hex displays. In this case, switches are labeled as SA, while LEDs have been labeled as LA. Signal lines drawn crossing each other are interpreted as being unconnected unless a dot is drawn at their intersection. Should your design require multiple logic diagrams to properly represent, signal lines going to and coming from other diagrams are drawn to enter the diagram at the left and leave the diagram at the right, using standard port-in and port-out notation, and are labeled with the source/destination diagram name and signal name, separated by a slash. As your designs get more complex, logic diagrams can be component level (RTL or Register Transfer Language) rather than gate level. For example, in experiments post-Lab 1, a 2:1 multiplexor can be drawn as a block by itself.

2-TO-1 MULTIPLEXER

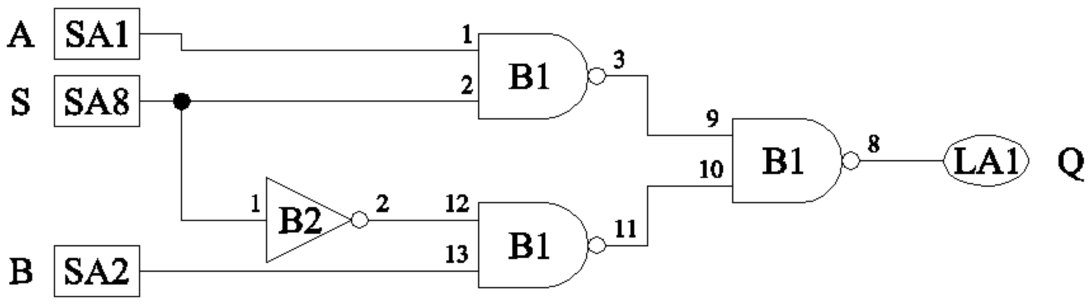


Figure 10 – Logic Diagram (aka Schematic Diagram)

Note that for documentation purposes, TTL labs will require a logic diagram and **one** of the component layout formats (either manual or Fritzing).

IV. DESIGN TECHNIQUES FOR TTL

General Considerations:

Introductory logic design courses such as ECE 120 choose to ignore certain aspects of circuit design to provide a clearer presentation to the student. Technological constraints imposed upon the designer are ignored to emphasize a strong theoretical base for further study. In the laboratory, actual device characteristics must be included in all designs.

In Transistor-Transistor Logic (TTL), a NAND gate can be implemented with fewer transistors than an AND gate. In addition, since any function can be implemented with NAND gates, their use can reduce overall package count and diversity in a design. The ECE 385 student lab kit contains no AND or OR gates; it contains NAND and NOR gates. Design with AND or OR gates is conceptually easier than design with NAND and NOR gates because the design follows intuitively from Boolean algebra equations. A simple technique for converting two level AND/OR circuits to NAND circuits is given later in these notes. You will have to design your circuits considering what actual devices are available in the ECE 385 student lab kit.

Another point to remember when designing with actual devices is that all unused inputs to the device must be tied to some logic level. The decision to tie an input to zero or one is made by checking the input's function on the device data sheet. To tie an input to a logic zero, simply connect it to system ground. To pull an input to a logic one, a 'pull-up' resistor (nominally 1kΩ for 74LSXXX) must be inserted between the input and the +5 volt power supply. Note that a single resistor may be used to hold more than one (up to ten) input at the logic one level.

Delays and Glitches:

An extremely important characteristic of TTL (and other technologies) that is often overlooked by inexperienced designers is that all logic gates have a nonzero propagation delay from input to output. These propagation delays are infinitesimal with respect to the human observer. However, the propagation delay of TTL gates is finite and may lead to the occurrence of glitches in a signal. A glitch is a momentary incorrect output value produced by a circuit during periods when the inputs are changing. Glitches may be detected during the design phase using timing diagrams. Glitches are one of the most important new concepts we will discuss in ECE 385 and have wide ranging implications for digital design in general.

The circuit of Figure 11 should always output a logic one at C as it implements the steady-state logic equation $C = A \oplus \bar{A} = 1$. However, due to the finite delays of the gates used, it may momentarily output a logic zero (a glitch) right after the input A changes. The inverter in the circuit of Figure 11 is from a 7404 chip. The 7404 has a typical propagation delay of 10ns, and a guaranteed maximum delay of 22ns. The exclusive-or gate (7486) has a typical delay of 14ns and a maximum of 30ns. Note that neither of these gates has a guaranteed minimum delay time; therefore, the assumed minimum delay is zero. When preparing a timing diagram always use the worst-case propagation delays, not the typical!

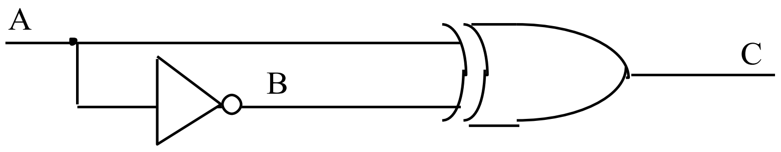


Figure 11 - Propagation Delay Example Circuit

A timing diagram for the circuit of Figure 11 is given in Figure 12. At time T = 0, the input A changes from a zero to a one. The output of the inverter (B) is unknown from time T = 0 until time T = 22 ns. The value of B probably changes somewhere around 10ns (the typical delay time), but the output cannot be guaranteed until T = 22ns. Since B is an input to the XOR gate, its output cannot be guaranteed until 30ns (maximum propagation delay) after the value of B is known. The cross-hashed areas in the timing diagram are the times when the signal value is unknown. The XOR output, C, is unknown for 52 ns after the input A changes. During those 52 ns, the output may be at logic zero or logic one. If it becomes a logic zero at any instance during these 52ns, then the circuit is said to produce a glitch.

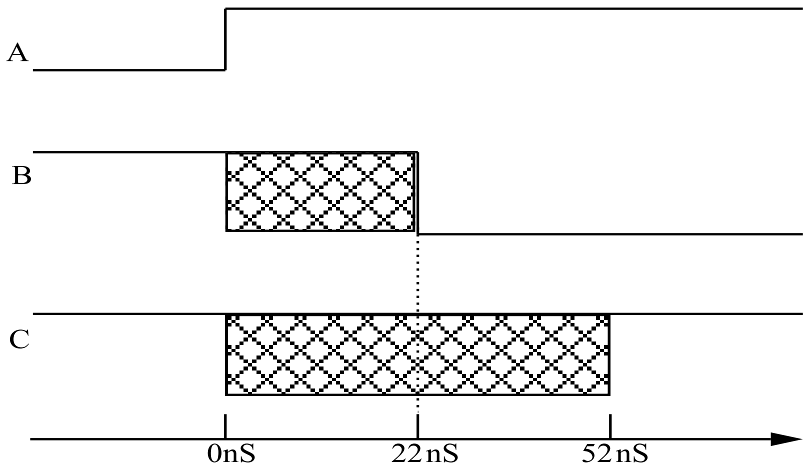
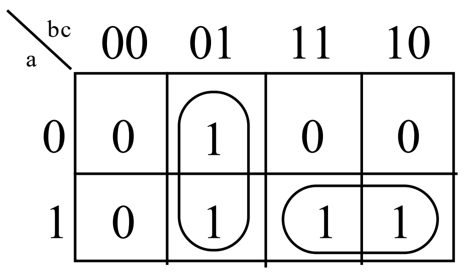


Figure 12 - Propagation Delay Example

Use of NAND (NOR) Gates:

A Karnaugh map of a function is given in Figure 13. To implement this function directly in standard 7400 series TTL components would require three chips: a 7404 hex inverter, a 7408 quad 2-input AND, and a 7432 quad 2-input OR. A logic diagram of the circuit is shown in Figure 13.



a \ bc	00	01	11	10
0	0	1	0	0
1	0	1	1	1

Figure 13 - Karnaugh map of 2-to-1 Multiplexer Function

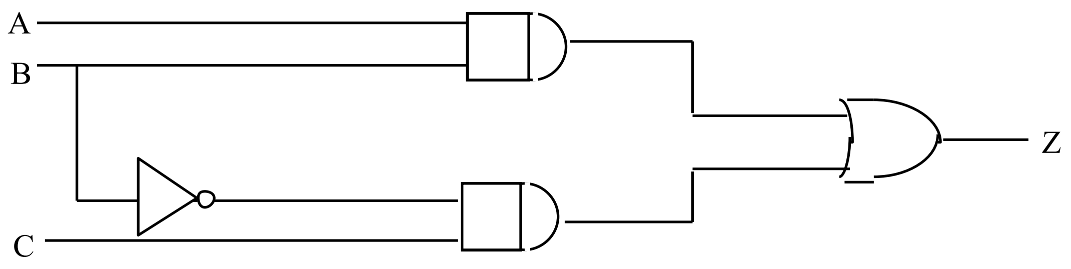


Figure 14 - Sum of Products implemented with 2-level AND-OR Logic

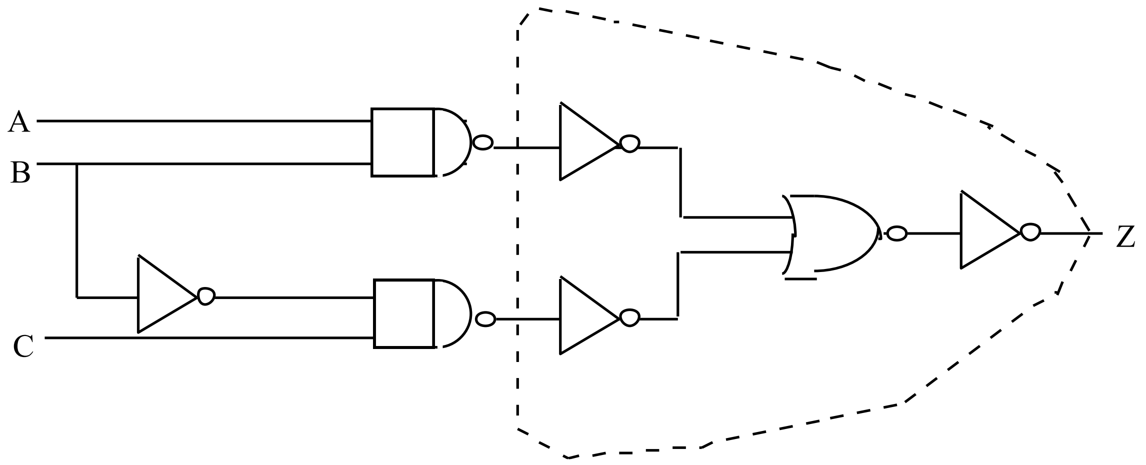


Figure 15 - Transforming AND-OR to NAND-NAND

Since the ECE 385 lab kit does not contain 7408's or 7432's, it is necessary to modify the circuit to one that uses only NANDs, NORs, and inverters. A direct modification is shown in Figure 15. Each AND gate has been replaced by a NAND gate and an inverter, and each OR gate has been replaced by a NOR and an inverter. Notice the four circled gates in Figure 15: a NOR gate with inverted inputs and inverted output. These four gates can be replaced by a single NAND gate. The resulting circuit is shown in Figure 16. In general: **any two level AND/OR circuit can be replaced by a two level NAND/NAND circuit by** simply replacing each AND and OR gate with a NAND gate.

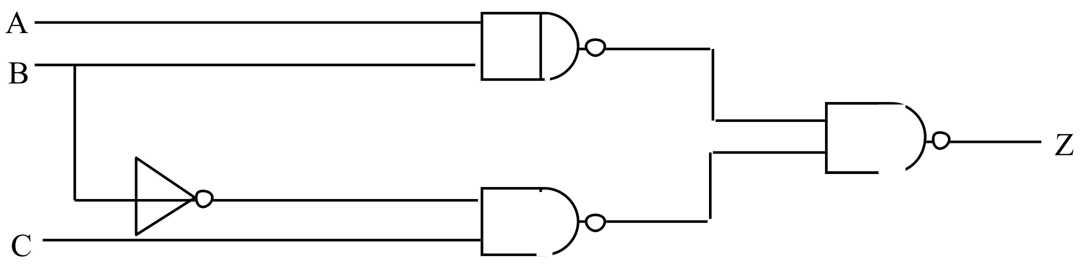


Figure 16 - Final implementation using NAND-NAND

A slight modification of the circuit of Figure 16 is given in Figure 17. Notice that this circuit requires only one 7400, whereas the original design (Figure 14) required three chips to implement. In this example, the conversion to an all-NAND implementation reduced the package count by two thirds.

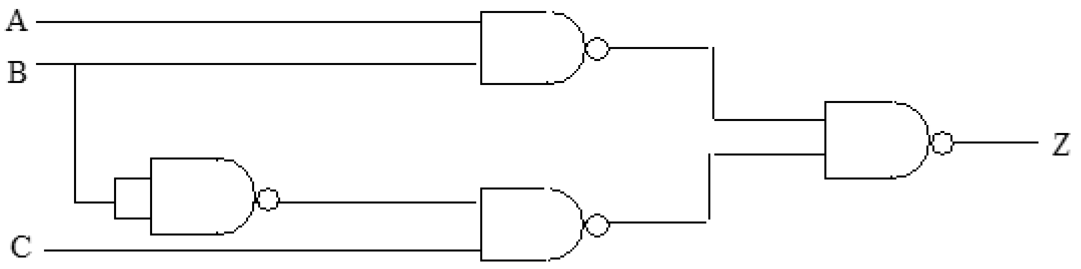


Figure 17 - Single chip implementation using one 7400 Quad 2-input NANDs.

IV. LAB TECHNIQUES

Circuit Assembly

Circuits should be assembled in the following manner (with the power supply switched OFF):

(1) Carefully align and insert IC's in their designed positions on the protoboard socket-strips according to your component-layout. Some of the IC's provided have been pre-socketed for their protection and are to remain socketed. Each IC should be positioned with its indentation or notch up (away from you).

(2) Establish a GND and a V_{cc} bus along the active socket-strips. Make power connections to all the IC's using black wires for GND connections and red wires for V_{cc} connections. It is essential that power connections be made properly or damage to the ICs, protoboard, and the power supply will result. Check your work!

(3) Make other wiring connections and check them off on your logic-diagram as they are made. Only No. 22 wire and 1/4 W resistors should be used in making connections on the panels. Heavier wire will damage the spring clips in the socket causing faulty contacts in the future.

(4) Label (with symbolic names) all switches, pushbuttons, and indicator lamps used. Labels are made by printing on removable gummed labels.

Wiring Techniques

Only No. 22 wire and 1/4 W resistors should be used in making electrical connections on the protoboard socket strips. The wire should have approximately 1/4" of insulation removed from the end. If more insulation is removed, the exposed metal wire will often contact an exposed neighboring wire, resulting in unreliable operation at best. When inserting or removing wires, push or pull in a vertical direction (i.e., don't wiggle the wires or they will break off in the socket). If new wire is cut, cut it to a convenient length and strip 1/4" of insulation from each end.

Select wire colors by function where possible to facilitate organization. However, it is best to use red and only red for +VCC wires; and black for GND. Where the output of a gate goes to several different places, debugging and circuit modification is much easier if you do not "daisy chain."

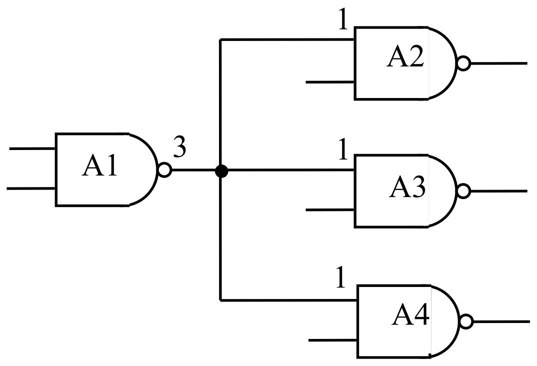


Figure 18 - Proper Fan-out Technique

That is, one end of each of the three wires making the electrical connections between gates should go to the socket strip connection points at pin 3 of chip A1. Don't run a wire from 3A1 (pin 3 chip A1) to 1A2 (pin 1 chip A2), a second wire from 1A2 to 1A3, and a third wire from 1A3 to 1A4.

Where possible physically route the wires so that you can test an IC's pins and replace it if it is bad, i.e., do not run wires tightly over the top of IC's.

Debugging Techniques

When inserting an IC onto the protoboard socket strip, be certain the pins on the IC are properly aligned. If the pins are not properly aligned, they may fold under when inserted onto the socket strip. The IC will appear to be properly inserted, but the pins which folded under will not make electrical contact. To detect this problem when debugging circuits, always check input logic levels on the exposed IC pins themselves and check output logic levels at one of the connection points on the protoboard socket. Use the oscilloscope, or the corresponding application with an ADALM module. If a hole on the socket strip is found to make a faulty connection, a short stub of wire should be inserted in the hole and clipped off, so that the hole will not inadvertently be used in the future. Always be suspicious of signals which are outside of the nominal range of 3.3V CMOS circuits and signals which appear noisy beyond the operating frequency of the circuit (typically 1 kHz).

Recall that a floating input is logically unknown. All unused light inputs should be grounded, as well as all unused inputs to an IC chip should be grounded or pulled high with a pull-up resistor (1 kΩ, 1/4 watt).

Light Emitting Diodes (LED's)

An LED is a semiconductor diode designed to emit light when forward biased. Since it is a diode, voltage polarity must be observed in its use (reverse voltage of 0 to 5 volts will simply leave the LED off). The logic kits contain red LED's.

LED's are ideal as logic signal indicators because zero current shuts them off and forward current turns them on. A typical indicator circuit is shown below. The current limiting resistor is necessary to prevent the LED from clamping (affecting) the signal voltage.

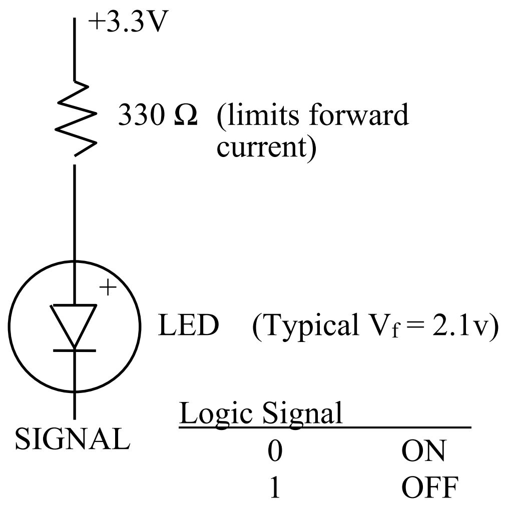


Figure 19 - Connecting LED Correctly

The reason why we typically wire LEDs this way (such that they turn ON when the signal is LOW) is that TTL based chips (74, 74F, 74LS, 74ALS) **sink** significantly more current than they **source**. E.g. the current sink capacity for a 74LS is typically 20 mA (therefore, up to 20mA can enter the output pin while the output voltage is low), but the source current may only be 5mA (that is, only 5mA may exit the output pin while the output voltage is high). CMOS based technology suffers from no such limitation, in fact they can source and sink similar currents (~20mA), however, the convention of having an LED turn on to signify a low voltage was established early on during the 70s and 80s to the point that LEDs are commonly wired this way even today.

Irrespective of which polarity you choose for your LEDs, it is important that each LED has its own resistor. If we have two or more LEDs to monitor several signals, why is it bad practice to share resistors?

LED Polarity

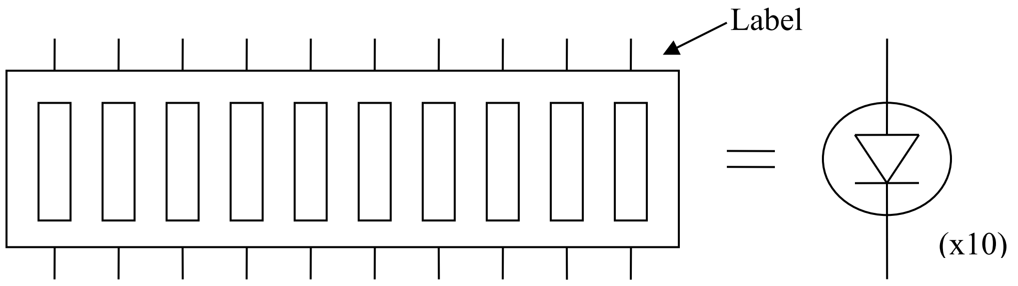


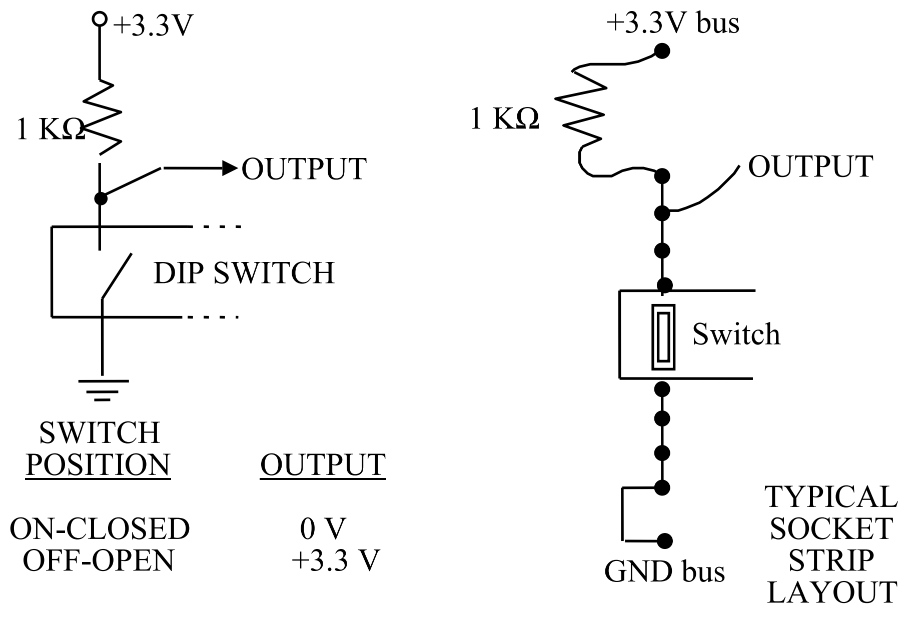
Figure 20 - LED polarity

For some designs it is recommended to use an additional gate (typically an inverter) in between the LED and the rest of the logic – in this case the inverter acts as a buffer. This is because even with the current limiting resistor, sometimes an LED on a signal which connects to many other chips (e.g., the clock) will 'load down' the signal, causing unpredictable behavior for circuits connected to said signal. Keep in mind that if you use an inverter as this buffer, the LEDs will light when the connected signal (input of inverter) is **high** and turn off when the connected signal is **low**.

Toggle Switches (8-switch DIPs)

The DIP switch contains 8 single-pole single-throw switches mounted in a 16 pin DIP. The "ON" position corresponds to a closed switch and the "OFF" position an open switch. Some switches are marked CLOSED AND OPEN.

One use of the DIP switch will be to form a switch "register". A 1-bit slice of such a "register" is shown below (note that the pull up resistor is **required** to prevent the logic input from floating while the switch is open).



SWITCH POSITION	OUTPUT
ON-CLOSED	0 V
OFF-OPEN	+3.3 V

Figure 21 - Switch Connecting

Note that the switches are numbered on each DIP package. It is convenient to refer to them and use them by number.

Switches may be toggled with a well-filed fingernail or a pencil point. Switches of this type (employing contact closures), bounce when they are closed or opened. The logic which receives switch outputs must be designed so that the bounces have no ill effect on its operation.

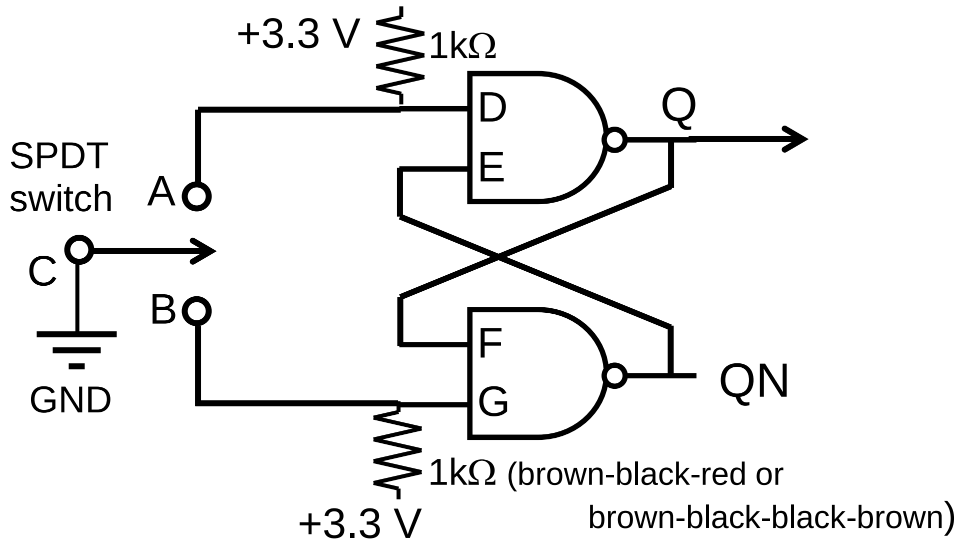
Note: If you choose to use these LEDs and switches, you need not reconnect to use the lab station LEDs and switches when you come to the lab. It may, however, be more convenient to do so.

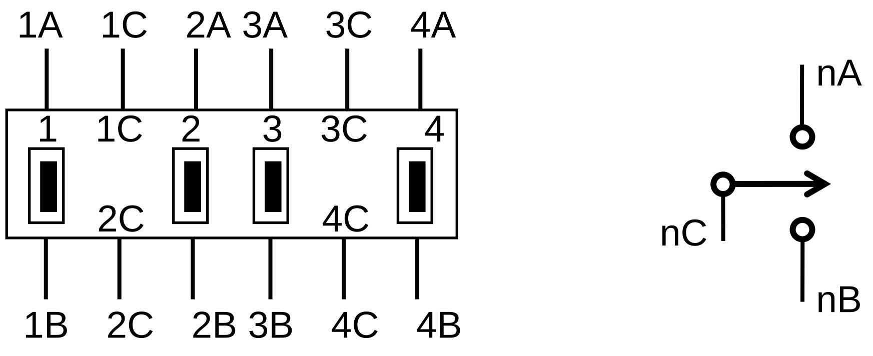
Contact Bounce:

The incredible speed of digital logic causes some interesting problems when interfacing circuits to the slow world around them. For example, when a mechanical switch is closed, the electrical contacts inside the switch are violently slammed together. The force of the impact of the two contacts is often sufficient to cause the contacts to separate momentarily, and then come together. The momentary separation of the electrical contacts is called ‘contact bounce.’ When a switch is flipped, the contacts may bounce several times before finally coming to rest. The contacts will be stable within milliseconds of first contact: essentially zero time to the human observer, but a long period for a digital logic circuit. If a switch is used to clock a counter circuit, the counter may advance several times per flip of the switch.

The ‘de-bouncer’ circuit shown in Figure 22 eliminates problems caused by switch contact bounce by ensuring a clean transition from a logic zero to logic one (or logic one to logic zero) when the switch is thrown. The details of how and why the debouncer circuit works is left as a post-lab exercise. (Hint: The $\overline{S}$ - $\overline{R}$ cross-coupled NAND latch has its output stable when $\overline{S} = \overline{R} = 1$)

Note the use of the pull-up resistors in the de-bouncer circuit. When the switch is in position ‘A,’ the input ‘D’ of the first NAND gate is tied directly to ground (a logic 0). With the switch in position ‘B,’ NAND gate input ‘D’ is not connected to ground but is pulled to a logic 1 by the pull-up resistor. Without the pull-up resistor, input ‘D’ would be left floating (unconnected) when the switch is in position ‘B.’ REMEMBER - All inputs must be held at some logic level!





QUAD SPDT Switch

Figure 22 - Debouncer Circuit

<u>FPGA Clock Output and Switchbox</u>

To facilitate debugging, we have pre-programmed an FPGA bitstream which will turn your Urbana board into a power supply, variable clock generator, and switchbox. The pinout is shown in Figure 23 and uses the PMOD connectors on the bottom left side of the Urbana board. Each of the positions on the PMOD connector will accommodate a standard sized breadboard wire, such as those in your wiring kit.

At the minimum, you will likely need to use the power and ground connectors as well as the Clk connector to supply power to your breadboard and generate the clock for the discrete logic lab(s). All the logic chips you have been provided may be powered by 3.3V. **Make sure that if you are using the power and ground connectors, you are not using any external power supply besides the Urbana board (e.g. you should not simultaneously use the Urbana board with the ADALM power supply).** Note that while the board will provide power (3.3V) so long as the Urbana board is connected to a USB power source and turned on, supplying the clock signal will require you to program the FPGA with the provided switchbox .bit file. Note that you will need to have Vivado installed (follow the directions in the Introduction to Vivado document), but you do not need to make a project. You can directly access the “Hardware manager” and manually load the .bit file which is downloaded from the course materials. You can then program the FPGA, a red LED should start blinking at the rate of 1 Hz, indicating the Clk output.

The default clock (which will appear via the Clk pin in Figure 23) is a 0-3.3V square wave at 1 Hz. You may use BTN0 to toggle the clock frequency between 1 Hz and 1 kHz (note that your eyes cannot perceive a LED blinking at 1 kHz; it will just appear solidly lit but somewhat dimmer). In addition, signals S0-S15 are debounced versions of the Urbana board switches. You may use this in lieu of building your own switches on the breadboard to simplify some of the wiring in the discrete logic lab(s). Note that if the Urbana board turns off when connected to your breadboard, it is likely that you have a short in your circuit design. Typically, this will not damage the Urbana board, but you should disconnect power to your breadboard and do some debugging. The Urbana board should be able to supply at least 200 mA on the 3.3V outputs (labeled 3V3 in Figure 23) which should be sufficient for your discrete logic design(s). You may use any of the 4 3.3V pins and any of the 4 GND pins, as they are all connected together.

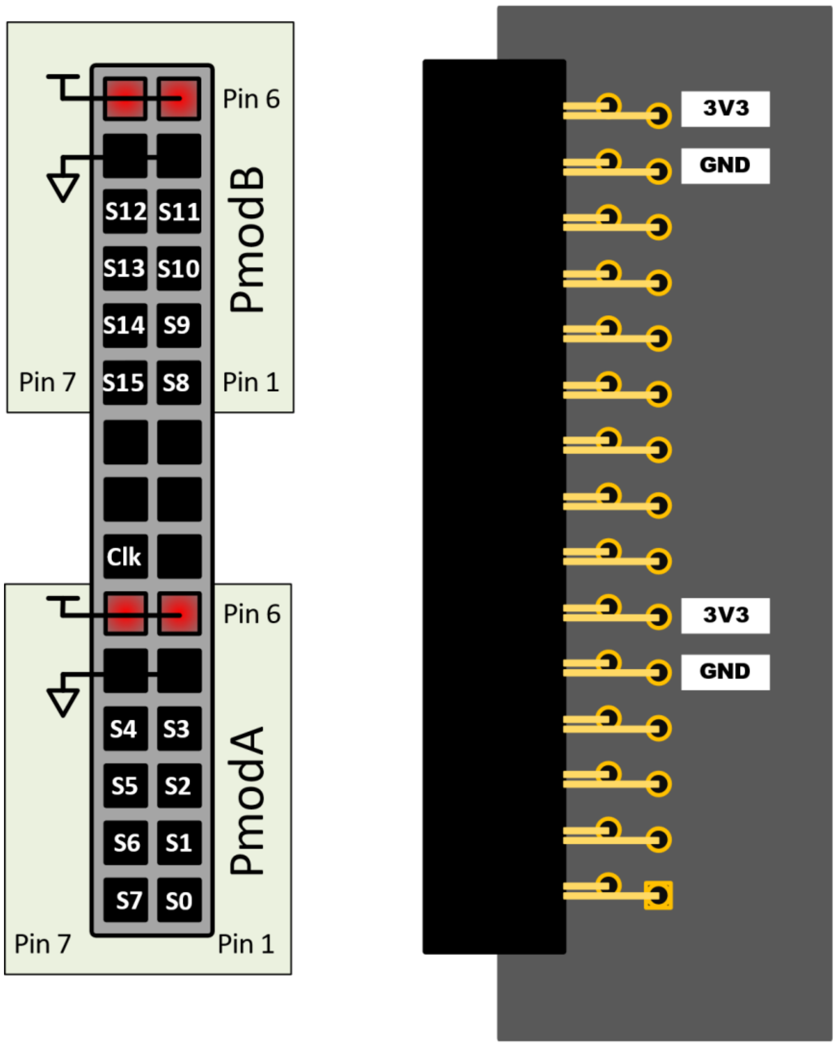


Figure 23 - FPGA Switchbox Connections