AC 2011-519: TEACHING GROUND-FLOOR DIGITAL CIRCUITS TO PRE-ENGINEERING STUDENTS

Christopher R. Carroll, University of Minnesota Duluth

CHRISTOPHER R. CARROLL earned academic degrees at Georgia Tech and Caltech. He is Associate Professor of Electrical and Computer Engineering at the University of Minnesota Duluth. His interests are digital systems and microprocessor applications, especially as they relate to educational environments.

Teaching Ground-Floor Digital Circuits
to Pre-Engineering Students

Abstract

Digital circuits pervade many applications in all engineering disciplines today. Digital circuit basics are easy to introduce early in a pre-engineering curriculum because there are no math or other technical prerequisites, and because the topic sounds glamorous to students. Presented here is a lab instrument that serves well for teaching basic, "ground-floor," digital circuits to students who have no engineering background. Also included is a teaching strategy that uses this instrument to present digital circuits in an uncomplicated and non-intimidating way. This material is suitable for high-school students, or even middle school students, and could be used in pre-engineering courses such as Project Lead the Way's Digital Electronics course.

The secret to presenting digital circuits successfully to students who have no technical background is to avoid references to electricity or computers and stick purely with the 1's and 0's. In such a setting, digital circuits are just implementations of mathematical expressions. There are no "volts" or "bytes" or anything that might confuse the simplicity of 1's and 0's. By treating digital circuits simply as implementations of mathematical expressions, and treating wires as just pencil lines that connect logical elements in drawings, students can learn to design and build digital circuits comfortably.

This paper provides applications of a lab station design that has been disclosed in an earlier ASEE paper[1], and discusses techniques for using that station in teaching digital circuits to students who have no technical background.

The Digisplay Instrument

The Digisplay lab station, disclosed in an earlier ASEE paper[1], is shown in Figure 1. Digisplay is a contraction of "Digital display." It is intentionally bare-bones-simple in its design, so that using it is intuitive for students. There are no settings or adjustments that need to be made by the student to use the Digisplay instrument. Just connect it to the circuit under test and see results.

Digisplay is intended to test the two foundational types of digital circuit designs, combinational circuits and synchronous sequential circuits. *Combinational* circuits are those without memory, so that circuit output variables are functions only of what the input variables are right now. *Sequential* circuits include memory, so that circuit output variables can be functions of the past history of values on input variables, as well as their current values. *Synchronous* sequential design imposes strict restrictions on design techniques, to avoid difficulties with awkward timing problems or other circuit glitches that are clumsy to explain to students. These restrictions basically are that there is a *single* system signal called the *clock*, generated by Digisplay, and any changes that occur in the circuit happen *only* in response to one of the edges of that clock signal, in this case the rising edge (0 to 1 transition) of the signal, referred to here as the clock "*tick*."

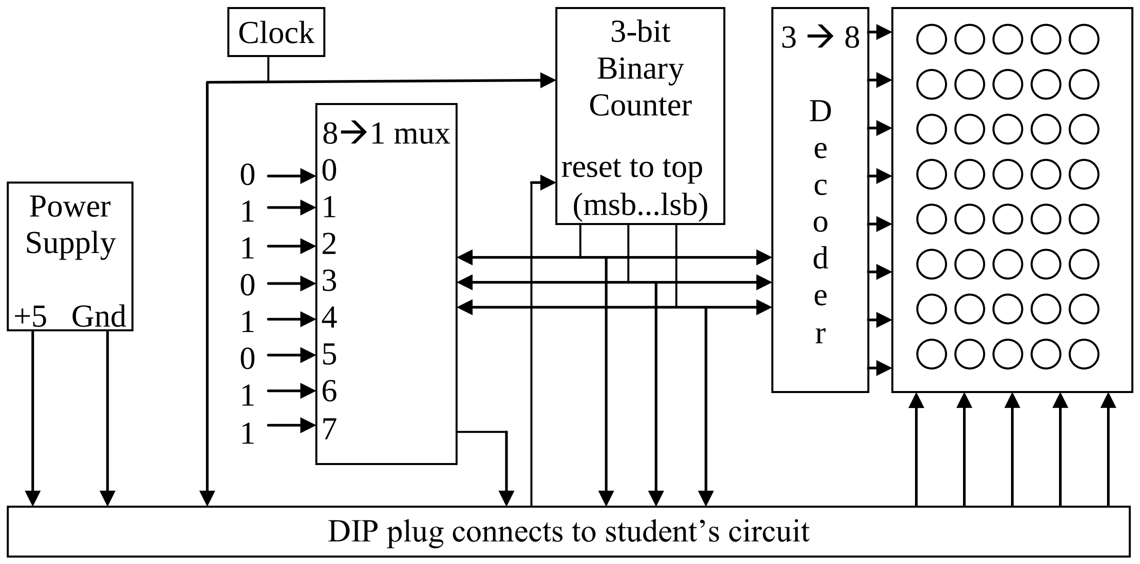

Figure 1. The Digisplay *instrument*

Digisplay tests combinational circuits with up to three input variables by supplying the circuit with every possible combination of 1's and 0's on those variables (eight possibilities) from an internal counter, the "3-bit Binary Counter" shown in Figure 1. Those three input variables are applied to the user's circuit under test through a DIP plug inserted into the user's breadboard. Variables to be displayed, particularly the user's output variables, are conveyed back to Digisplay through the DIP plug for display. Digisplay shows up to five variables, one per column on the 5 x 8 LED matrix display. Each row of the display shows the values of the variables for one set of values on the three signals supplied to the circuit under test. Typically, those input variables would be displayed in the leftmost three columns of the display, with the other two columns used for variables of interest from the student's circuit, resulting in a truth-table display. Figure 2 shows the display for a user's circuit that generates the exclusive-or of the three input variables.

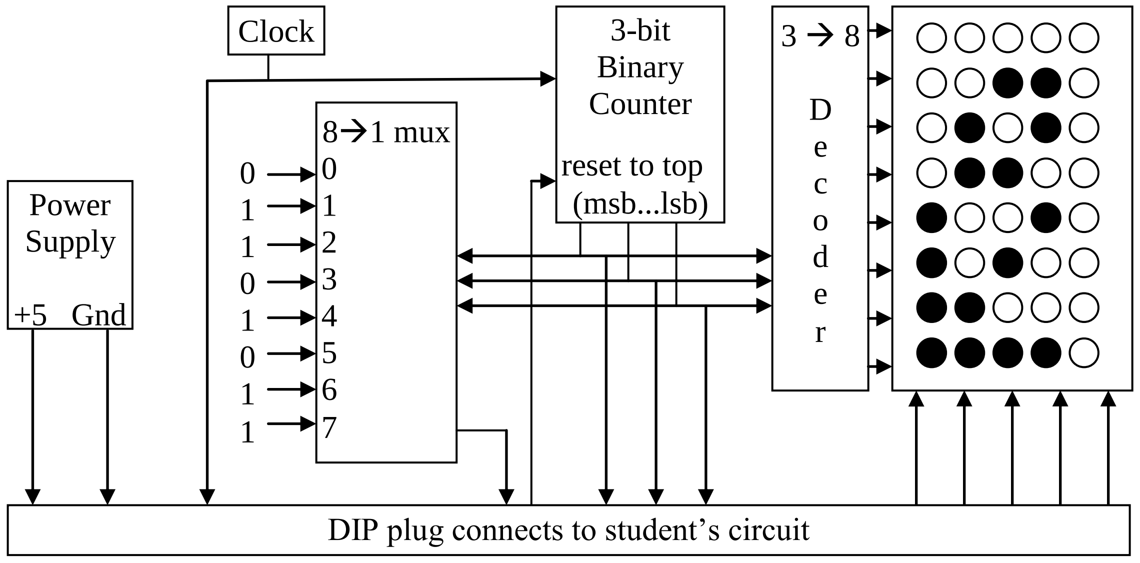

Figure 2. Digisplay *used to test a circuit producing the exclusive-or of three input variables.*

Sequential circuits are tested by supplying the student's circuit under test with the clock signal generated by Digisplay. The internal 3-bit binary counter continues to run, and activates one row of the 5 x 8 LED display per clock cycle. Variables from the student's circuit are shown on columns of the display, and changes in those variables as the clock ticks are presented. In order to synchronize a repetitive display sequence with other than eight states in the sequence, the user's circuit must supply a "reset to top" signal to Digisplay during one of the circuit's states. This returns the 3-bit binary counter to the top row so that the display is synchronized to the operation of the student's circuit.

A particularly popular type of sequential circuit often assigned to beginning digital designers is a "sequence detector," which must respond in a specified way whenever a specified sequence of values is observed on an input variable as the clock ticks. Digisplay generates a repetitive sequence of values to be used as an input by the student's circuit using the 8-to-1 multiplexor shown in Figure 1, which chooses one of its inputs to send to the user's circuit for each set of values on the 3-bit binary counter. The sequence can be changed easily by changing the 1's and 0's on the input of the 8-to-1 multiplexor, but this would be the job of the instructor, not the student. Figure 3 shows Digisplay's output for a circuit responding to the "011" input sequence.

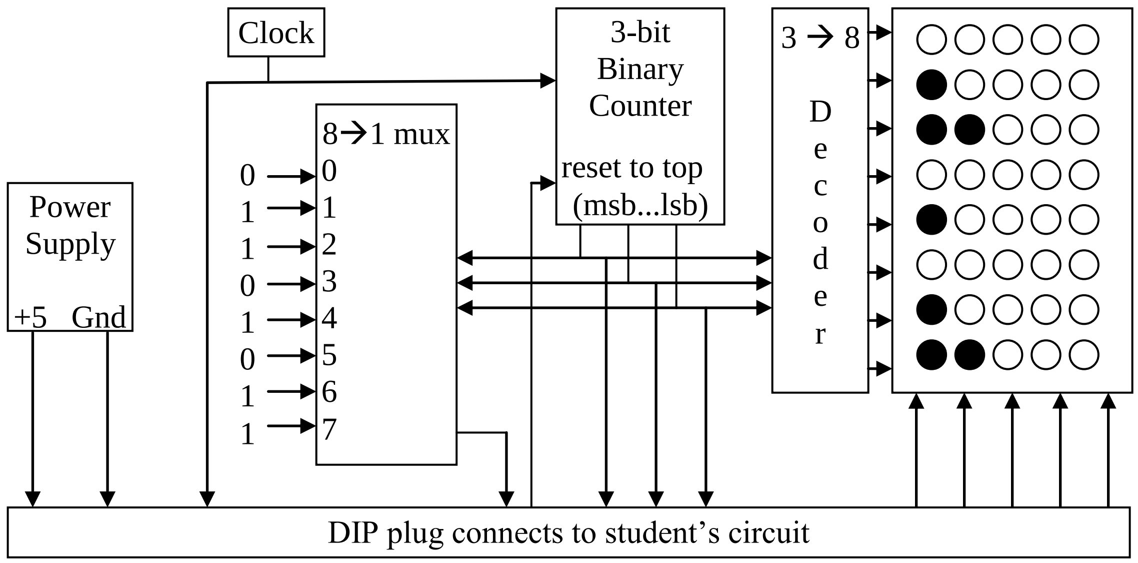

Figure 3. Digisplay used to test a circuit responding to the input sequence "011"

Instructional Secrets

Teaching digital design to students who have no technical background is easy. No prerequisites are required. The topic is totally self contained, and students typically are already interested because of the glamour associated with the word "digital" these days. However, digital circuits can be enormously complex and tricky. The goal of an introductory class must be to inspire interest in students without giving too much information at once. Students must be *dazzled*, but not *intimidated*.

Engineers are always tempted to employ the latest gizmo or most recent technology in their designs. Generally speaking, engineering applications in industry benefit from that approach. Designs using the "leading edge" of technology generally will be well-received by consumers, and will have the longest application lifetime. However, for instruction, fundamentals trump glamour, and engineering instructors must throttle back their temptations when teaching introductory, foundational material such as a first introduction to digital design. Otherwise, students can easily be intimidated by too much complexity, and fail to grasp the fundamentals.

Digisplay is designed using the TTL (Transistor-Transistor-Logic) logic family. The logic family used simply determines the particulars of the electrical implementation of logical devices and defines the physical representations of the logical 1 and 0 values, but all that is totally unimportant when introducing digital circuits for the first time. Any available logic family can be used. TTL was chosen because of its electrical durability and familiarity. It has been available for at least forty years. CMOS (Complementary Metal Oxide Semiconductor), another popular logic family, is susceptible to electrical damage from static charge, and thus does not serve well in an introductory class where electrical properties of the devices are not discussed. As technologies change, other emergent logic families may be preferred over TTL.

Why Not Programmable Logic?

Modern digital designs make use of two different programmable logic structures. One of these is the FPGA (Field Programmable Gate Array) and the other is the CPLD (Complex Programmable Logic Device). Both devices can be configured to implement complex digital systems, sometimes including millions of logic elements. For industrial designs where the goal is to implement the design as cheaply and reliably as possible, FPGAs or CPLDs are the way to go, without question. However, for introductory instruction, they are a poor choice.

Learning digital design using FPGAs or CPLDs is like learning arithmetic using a calculator. Sure, you get the right answer every time, and that's the goal in an industrial setting. However, in an educational setting, the goal is not the answer itself, but *how to produce* the answer. Fundamentals are the key to teaching students how to design digital circuits. Otherwise, students learn only what buttons to push to make the answer appear, without understanding the underpinnings or implications of what they are doing. To design a combinational circuit using an FPGA or a CPLD, one can simply enter the truth table for the desired function(s) and the synthesis software that configures the device figures out the implementation. Where is the design experience there? Nothing is learned by using such tools except how to use the tool.

Pedagogy

Teaching digital circuits to pre-engineering students who have no technical background requires some restraint on the part of the instructor. The topic can be mind-boggling in complexity, but the fundamentals are easy. The task is to convey the fundamentals while giving just hints about where those fundamentals can lead.

Combinational circuits must focus on logic gates, the primitive circuit elements from which all other combinational structures are built. Students must understand AND, OR, NAND, NOR, XOR, XNOR, and NOT gates fully, and must be comfortable in manipulating logic functions for implementation using a variety of different collections of those gates. Description of higher level combinational modules such as decoders, data selectors, arithmetic elements, etc. should not be included, except possibly as gate-level design assignments. There is plenty of opportunity to challenge students in design using just gates.

For example, lab exercises for combinational circuit design typically would specify a function or functions to implement, *and* a set of constraints, or costs, by which to measure the quality of the resulting design. With three variables, there are $2^{2^\wedge 3}$, or 256 different functions available. It is easy to create a circuit that *works* by just replacing the operators in the arithmetic expression for the function with appropriate gates. The resulting circuit can't help but work. However, designing a working circuit under appropriate constraints can add significant challenge to the assignment. For example, possible constraints might include designing the circuit using…

… a minimum number of components, or "chips" (affecting cost).
… a minimum number of gates of any kind (affecting complexity).
… a minimum number of NAND gates only (encouraging flexibility).
… a minimum number of NOR gates only (encouraging flexibility).
… a minimum total of gate inputs on all the gates in the circuit (affecting power).
… gates of any type that have no more than 2 inputs per gate (requiring innovation).

The first constraint listed above provides the most realistic estimate of circuit cost when designing with TTL or other logic families where three or four identical gates are packaged in a single component, or "chip." Using just one gate from a chip in a design costs the same as using all the gates in the chip, so minimizing chip count, which is not necessarily the same as minimizing gate count, is a reasonable strategy for minimizing the cost of the resulting circuit.

Figure 4 shows three different gate-level implementations of the same function,

$$f(a,b,c) = \overline{a}\overline{b}c + \overline{a}bc + a\overline{b}c + ab\overline{c} = \overline{a}c + \overline{b}c + ab\overline{c} = (a+c)(b+c)(\overline{a}+\overline{b}+\overline{c})$$

which is shown above in canonical sum of products, minimal sum of products, and minimal product of sums algebraic forms.

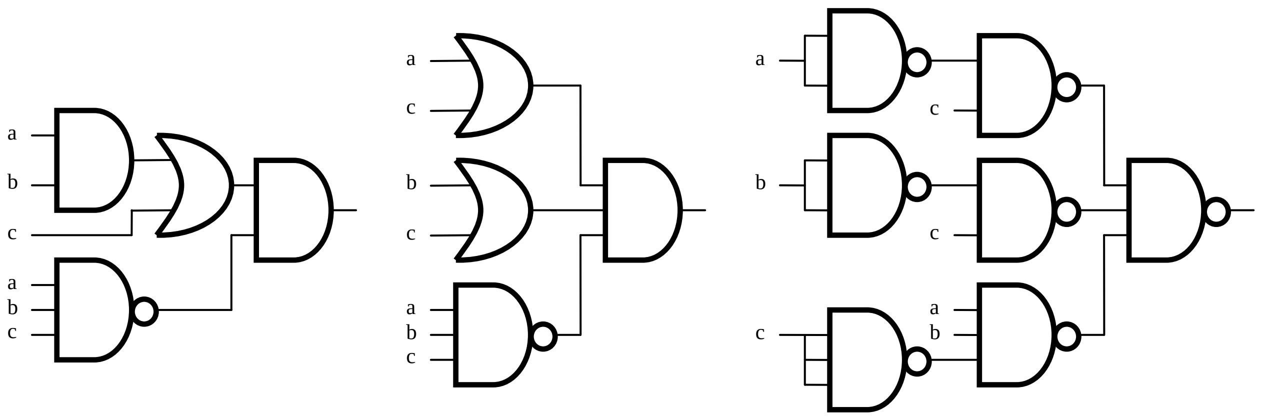

Figure 4. Three different implementations of the same function. Which is "best?"

Designing and implementing this function would be a typical lab assignment in combinational circuits. Which of the three designs for the function shown in Figure 4 is “best?” The answer depends on the design criteria. The left design uses the fewest gates (4) and the fewest gate inputs (9), and therefore is the least complex of these designs, but uses three different types of gates, and so would require three chips for implementation. The middle design uses the same number of gates but more gate inputs (10) than the left design. It still would require three chips for implementation, but in this middle case the worst case path from input to output passes through only two levels of gates, resulting in a design that is probably faster than either of the other two circuits. The design on the right side of Figure 4 clearly is more complex than the other two designs, but because of the way gates are packaged in components, it requires only two chips using standard TTL components, and so is probably the least costly of the three designs. Different design criteria result in different “best” solutions. This is the reason that ABET’s definition of engineering design requires comparison of different alternative solutions to determine the best solution for a given design situation.

Sometimes intuition and experience are required to achieve the best solution. The circuit in Figure 5 generates the same function as Figure 4’s circuits. It would be hard to argue against this as the “best” solution. The goal of teaching this material to students is to make such creative solutions accessible to them.

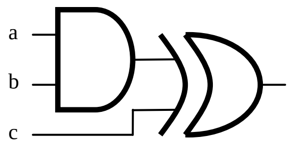

Figure 5. Same function!

Teaching sequential circuits requires introduction of flip-flops. For pre-engineering students, the D flip-flop is adequate, and easy to describe. More complex flip-flops such as JK or others add additional steps to the design process and cause unnecessary complexity. D flip-flops are the only sequential component necessary to design sequential circuits.

A typical sequential circuit that might be assigned is a counter that cycles through the sequence,

$$\begin{tikzcd} XYZ = & 001 \arrow[r] & 010 \arrow[r] & 011 \arrow[r] & 100 \arrow[r] & 101 \arrow[llll, bend left] \end{tikzcd}$$

and repeats this five-state pattern forever as the system clock ticks. Figure 6 shows two different circuits that implement the counter producing the above sequence on variables X, Y, and Z.

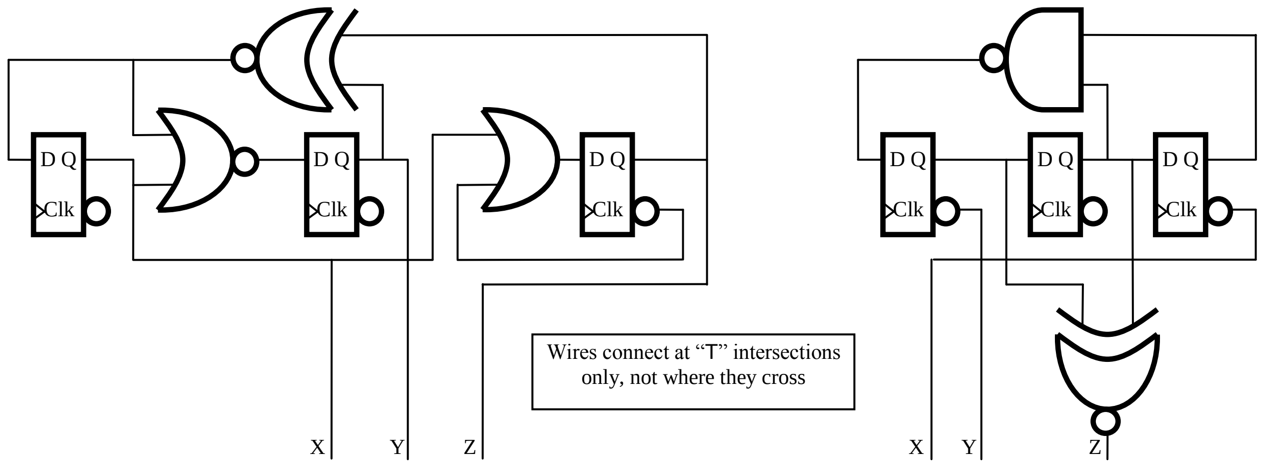
D Q
Clk
D Q
Clk
D Q
Clk
Wires connect at “T” intersections only, not where they cross
X
Y
Z
D Q
Clk
D Q
Clk
D Q
Clk
X
Y
Z

Figure 6. Two different implementations of a counter that cycles through the same sequence

In Figure 6, since the circuit is synchronous, the clock input on the flip-flops is not shown for clarity, because it is known to connect to the single system clock signal. The design on the left of Figure 6 is achieved through the "intuitive" approach where the output variables, X, Y, and Z, are chosen to be the outputs of the three flip-flops. This requires a particular state assignment, that results in the circuit shown to generate the D inputs of the three flip-flops. However, a different choice of state assignment leads to the simpler circuit on the right side of Figure 6. In this circuit, the basic five-state counter (three D flip-flops plus the one NAND gate) cycles through a different sequence of values internally. Additional logic (the XNOR gate) is required to translate that internal sequence to the desired external sequence on X, Y, and Z. The XNOR gate can be replaced with three 2-input NAND gates to minimize the chip count, if needed.

These basic examples of combinational and synchronous sequential lab experiments show the depth of design experience that can be conveyed to pre-engineering students even using just the simplest of logical components. The first solution that comes to mind is not necessarily the best solution. A little thought, intuition, and experience often will lead to solutions not initially envisioned that may be better matches to the design constraints.

Assessment

The tools and techniques for introducing digital circuits to pre-engineering students presented here have not yet been implemented in an actual class setting, so there are no assessment results. However, the author has thirty years of experience in teaching digital circuits to undergraduate college students, and the approaches used in this paper have resulted from interactions with many undergraduate students who have learned digital circuits using similar techniques. The goal of this teaching strategy is to introduce the topic without the need for prerequisites and without intimidating students with unnecessary complexity.

Summary

Teaching digital circuits to pre-engineering students at the "ground floor" of complexity is a rewarding challenge for faculty. The Digisplay instrument makes testing lab designs easy and comfortable for students, and avoids the need for faculty to spend instructional time describing how to use complex and intimidating instrumentation. Faculty must restrain themselves from introducing mind-boggling complexity. Even by restricting design components to just standard gates and D flip-flops, faculty can encourage students to explore alternative design strategies and criteria to achieve design results with a variety of characteristics.

References

1. Carroll, C. R., "Test Equipment for High School Digital Electronics Designs under Project Lead the Way," *Proceedings of the 2007 ASEE North Midwest Section Meeting*, Houghton, MI (2007).

2. Carroll, C. R., "Digital Logic Lab Experiments Using the Chipmonk Instrument," *Proceedings of the 2001 ASEE North Midwest Section Meeting*, Grand Forks, ND (2001).

3. Carroll, C. R., "Portable Input/Output Instrument for Interfacing Student Designs," *1998 ASEE Annual Conference*, Seattle, WA (1998).