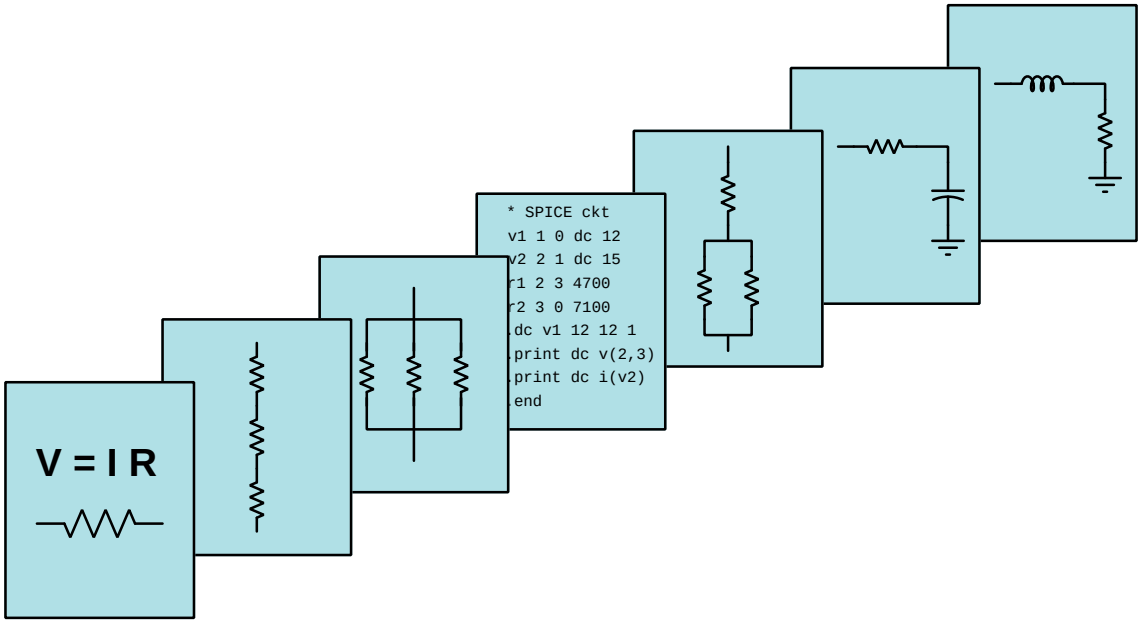
Modular Electronics Learning (ModEL) project



Digital Circuit Diagnostic Principles

© 2020-2025 by Tony R. Kuphaldt – under the terms and conditions of the Creative Commons Attribution 4.0 International Public License

Last update = 19 November 2025

This is a copyrighted work, but licensed under the Creative Commons Attribution 4.0 International Public License. A copy of this license is found in the last Appendix of this document. Alternatively, you may visit `http://creativecommons.org/licenses/by/4.0/` or send a letter to Creative Commons: 171 Second Street, Suite 300, San Francisco, California, 94105, USA. The terms and conditions of this license allow for free copying, distribution, and/or modification of all licensed works by the general public.

Contents

1 Introduction **3**
- 1.1 Recommendations for students . . . 3
- 1.2 Challenging concepts related to digital diagnostic tools . . . 5
- 1.3 Recommendations for instructors . . . 6

2 Case Tutorial **7**
- 2.1 Diagnostic visualization tools . . . 8

3 Tutorial **13**
- 3.1 Digital logic levels . . . 14
- 3.2 Multimeters . . . 16
- 3.3 Logic probes and pulsers . . . 19
- 3.4 Breakout boxes . . . 21
- 3.5 Oscilloscopes . . . 22
- 3.6 Logic analyzers . . . 25
- 3.7 Assembled board testing . . . 26
- 3.8 JTAG . . . 29

4 Derivations and Technical References **33**
- 4.1 TTL logic levels . . . 34
- 4.2 CMOS logic levels . . . 35
- 4.3 Digital pulse criteria . . . 36

5 Questions **41**
- 5.1 Conceptual reasoning . . . 45
  - 5.1.1 Reading outline and reflections . . . 46
  - 5.1.2 Foundational concepts . . . 47
  - 5.1.3 Logic pulser usage . . . 48
  - 5.1.4 Reverse-engineering techniques . . . 49
- 5.2 Quantitative reasoning . . . 50
  - 5.2.1 Miscellaneous physical constants . . . 51
  - 5.2.2 Introduction to spreadsheets . . . 52
  - 5.2.3 Simple logic probe circuit . . . 55
  - 5.2.4 Pulldown resistor sizing . . . 57

5.3 Diagnostic reasoning . . . 59
5.3.1 Logic probe and pulser . . . 60
5.3.2 Identifying possible faults . . . 61
5.3.3 Input switch settings . . . 62
5.3.4 Logic probe limitations . . . 63

A Problem-Solving Strategies **65**

B Instructional philosophy **67**
B.1 First principles of learning . . . 68
B.2 Proven strategies for instructors . . . 69
B.3 Proven strategies for students . . . 71
B.4 Design of these learning modules . . . 72

C Tools used **75**

D Creative Commons License **79**

E References **87**

F Version history **89**

Index **90**

Chapter 1

Introduction

1.1 Recommendations for students

Diagnostic testing of digital electronic circuits requires an array of tools, each one with its own purpose and its own limitations. This module introduces the reader to a range of tools from the simple to the sophisticated.

Important concepts related to digital circuit diagnosis include **opens** versus **shorts**, **logic voltage levels**, **four-wire resistance measurement**, **high** versus **low** logic states, **serial** data, **time-domain** signal measurement, **eye diagrams**, **soldered connections**, and **Design For Testing**.

Here are some good questions to ask of yourself while studying this subject:

- How might an experiment be designed and conducted to measure the electrical resistance of a component using the Kelvin four-wire method? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- How might an experiment be designed and conducted to measure the accepted voltage levels for "high" and "low" states at the input of a logic gate? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- How might an experiment be designed and conducted to measure the threshold voltage levels for a Schmitt trigger logic gate? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- How might an experiment be designed and conducted to measure the propagation delay of a logic gate? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- How do DC voltage values relate to digital logic states for different types of digital circuits?
- What does a logic probe do that a multimeter cannot?

- What does a multimeter do that a logic probe cannot?
- How does four-wire resistance measurement work?
- How are logic pulsers used in conjunction with logic probes?
- How do ohmmeters function to sense the resistance of external devices?
- Why is resistance measurement a useful tool for discerning the integrity of electrical connections on a printed circuit board?
- What special features are added to oscilloscopes to make them more useful for digital signal testing?
- What does a logic analyzer do that an oscilloscope cannot?
- What does an oscilloscope do that a logic analyzer cannot?
- What does "persistence" mode do in a digital oscilloscope, and why is this useful?
- What is a "digital phosphor" oscilloscope, and what is it used for?
- What types of manufacturing defects are observable using an X-ray camera system?
- What is the basic principle of JTAG, and how does it make board testing more efficient?
- What purposes may JTAG be used for other than PCB fault diagnosis?

1.2 Challenging concepts related to digital diagnostic tools

The following list cites concepts related to this module's topic that are easily misunderstood, along with suggestions for properly understanding them:

- **Sourcing versus Sinking output currents** – a common misconception is that since the output of a logic gate is called "output" it must mean that current only ever *exits* that terminal. This is untrue. All that "output" actually signifies is the fact that the gate is outputting *information* consisting of voltage values measured between that terminal and ground. Sometimes the assertion of a "low" (zero-voltage) logical state requires that the gate actually draw current *in* through its "output" terminal!

- **Gates require DC power** – since logic gates are really just transistor amplifier circuits, those transistors require constant DC voltage applied between their power terminals to function properly. Gates are *not* powered through their input terminals – these terminals only receive *information* in the form of "high" and "low" logic states, and ideally pass negligible current.

- **Pullup and pulldown resistors** – in digital circuits, resistors are often used to provide a secure logic state when an input device (such as a switch) goes to a high-impedance (open) mode. Students often have difficulty figuring out exactly where these resistors should go in a circuit. The most common mistake I've seen is to place one of these "pullup" or "pulldown" resistors in *series* with a gate input, which will accomplish absolutely nothing. The "trick" to getting this placement right, if you can call it a trick at all, is to literally follow the word "pullup" or "pulldown". A *pullup* resistor pulls the logic state of a wire up to the positive supply rail, and so must connect between the gate input and +V. A *pulldown* resistor pulls the logic state of a wire down to ground potential, and so must connect between the gate input and ground. In either case, the resistor provides a sure path to the opposite power rail that the input device connects to when active (closed).

1.3 Recommendations for instructors

This section lists realistic student learning outcomes supported by the content of the module as well as suggested means of assessing (measuring) student learning. The outcomes state what learners should be able to do, and the assessments are specific challenges to prove students have learned.

- **Outcome** – Demonstrate effective technical reading and writing

  Assessment – Students present their outlines of this module's instructional chapters (e.g. Case Tutorial, Tutorial, Historical References, etc.) ideally as an entry to a larger Journal document chronicling their learning. These outlines should exhibit good-faith effort at summarizing major concepts explained in the text.

  Assessment – Students present their "Conceptual Toolbox" Theory Journal entries listing important concepts applied in the Tutorial text(s). A "Conceptual Toolbox" serves as inspiration when solving problems, each concept suggesting a potential problem-solving action. Challenge students to collaboratively generate a Conceptual Toolbox for the day's topic during the class session and then use it when solving problems together, thus fostering the habit of reasoning from principles rather than recalling procedures!

- **Outcome** – Apply foundational circuit concepts to the proper sizing of pullup and pulldown resistors

  Assessment – Calculate appropriate resistor values for use as pulldown resistors in a digital logic gate circuit given different types of logic gates (e.g. TTL versus CMOS); e.g. pose problems in the form of the "Pulldown resistor sizing" Quantitative Reasoning question.

- **Outcome** – Diagnose a faulted logic circuit

  Assessment – Determine the probability of various component faults in a combinational logic gate circuit given symptoms and measured values; e.g. pose problems in the form of the "Identifying possible faults" Diagnostic Reasoning question.

Chapter 2

Case Tutorial

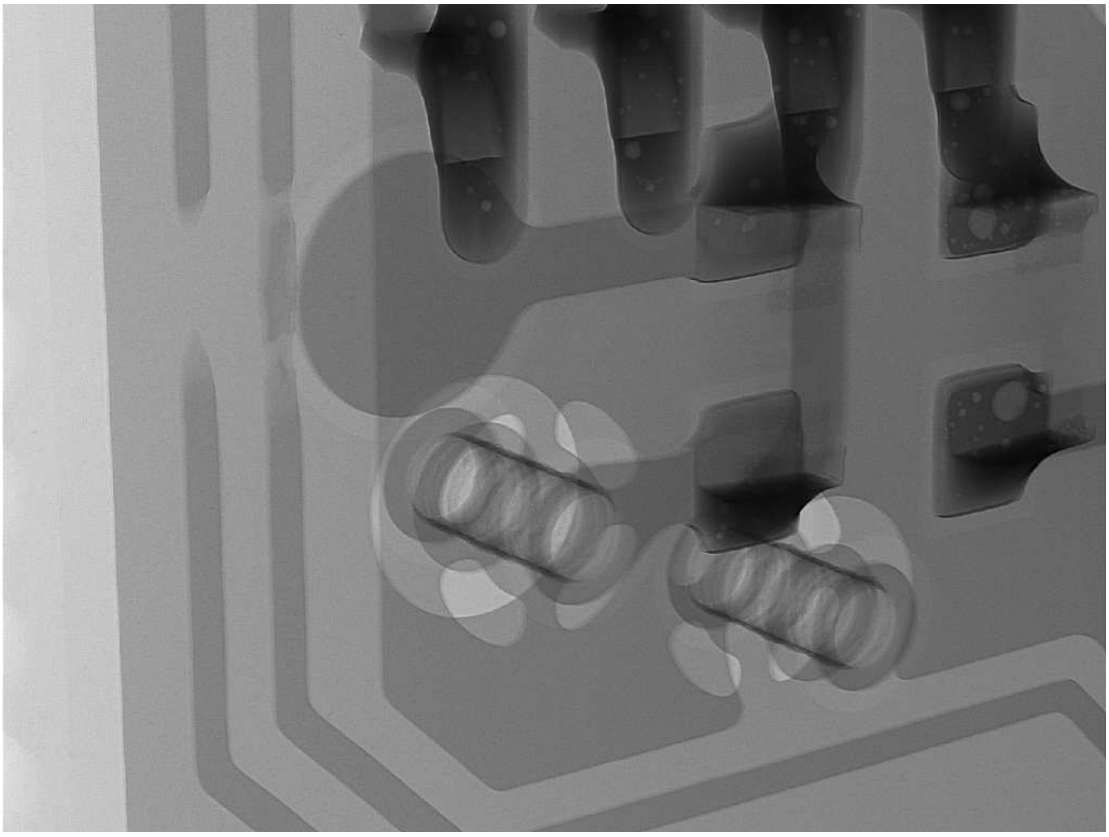
The idea behind a *Case Tutorial* is to explore new concepts by way of example. In this chapter you will read less presentation of theory compared to other Tutorial chapters, but by close observation and comparison of the given examples be able to discern patterns and principles much the same way as a scientific experimenter. Hopefully you will find these cases illuminating, and a good supplement to text-based tutorials.

These examples also serve well as challenges following your reading of the other Tutorial(s) in this module – can you explain *why* the circuits behave as they do?

2.1 Diagnostic visualization tools

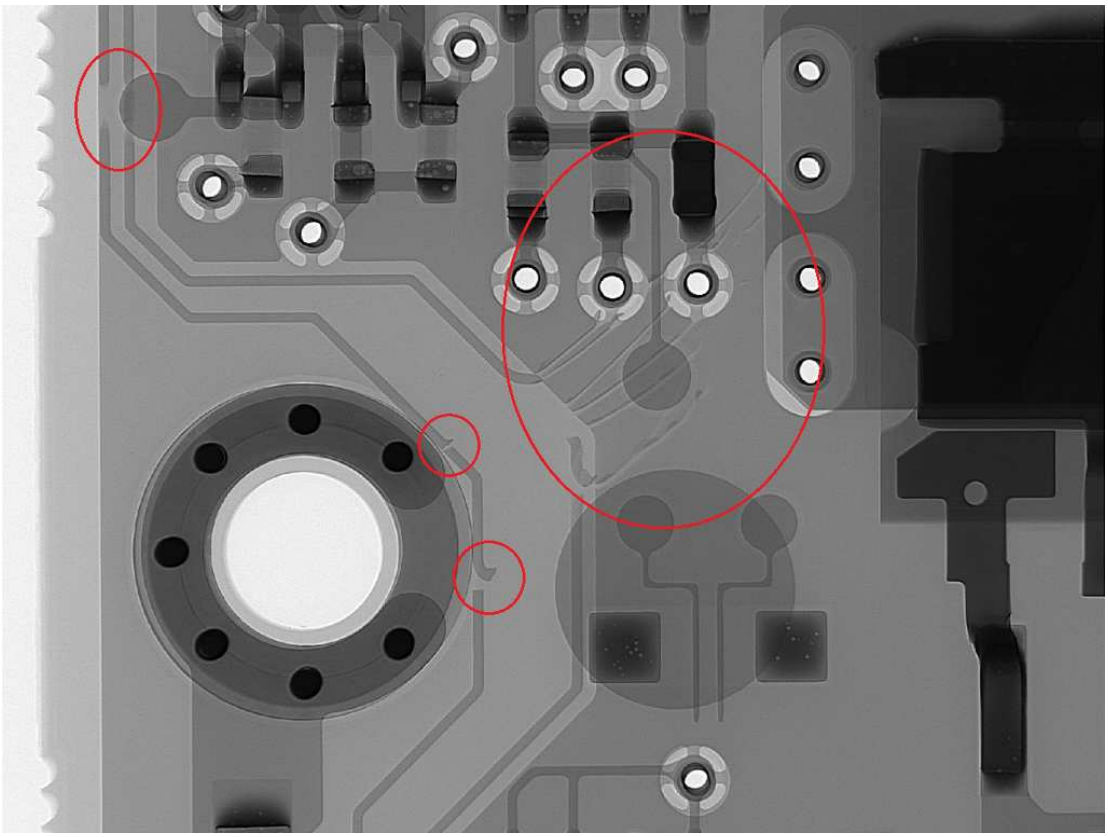
The following photographs[1] show examples of diagnostic visualization tools such as infra-red thermography and X-ray photography applied to the detection of faults on printed circuit boards (PCBs) and within integrated circuits (ICs).

Here we see an X-ray photograph of a PCB showing two broken traces resulting from defective board manufacturing where portions of the two traces were mistakenly etched away:

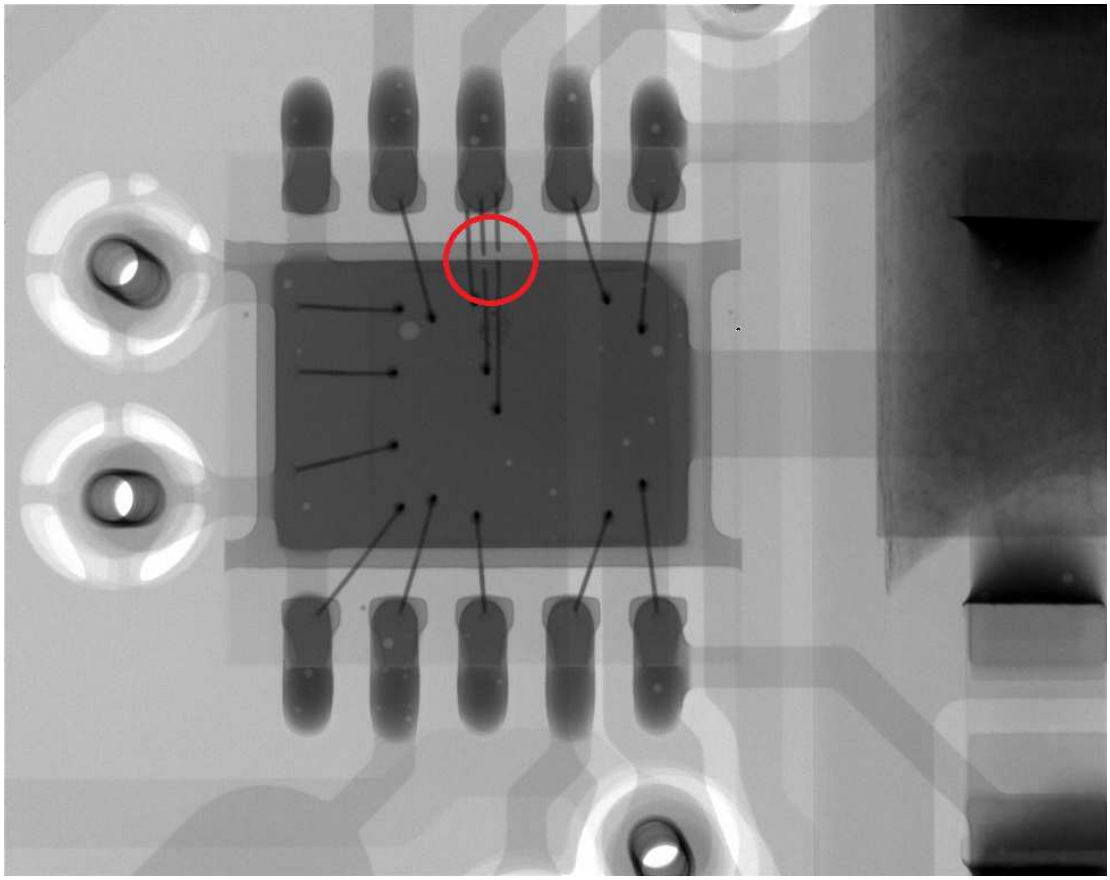


[1] I am indebted to the work of Brian Sharpes at Schweitzer Engineering Laboratories (SEL) for these images which he collected over a span of fifteen years.

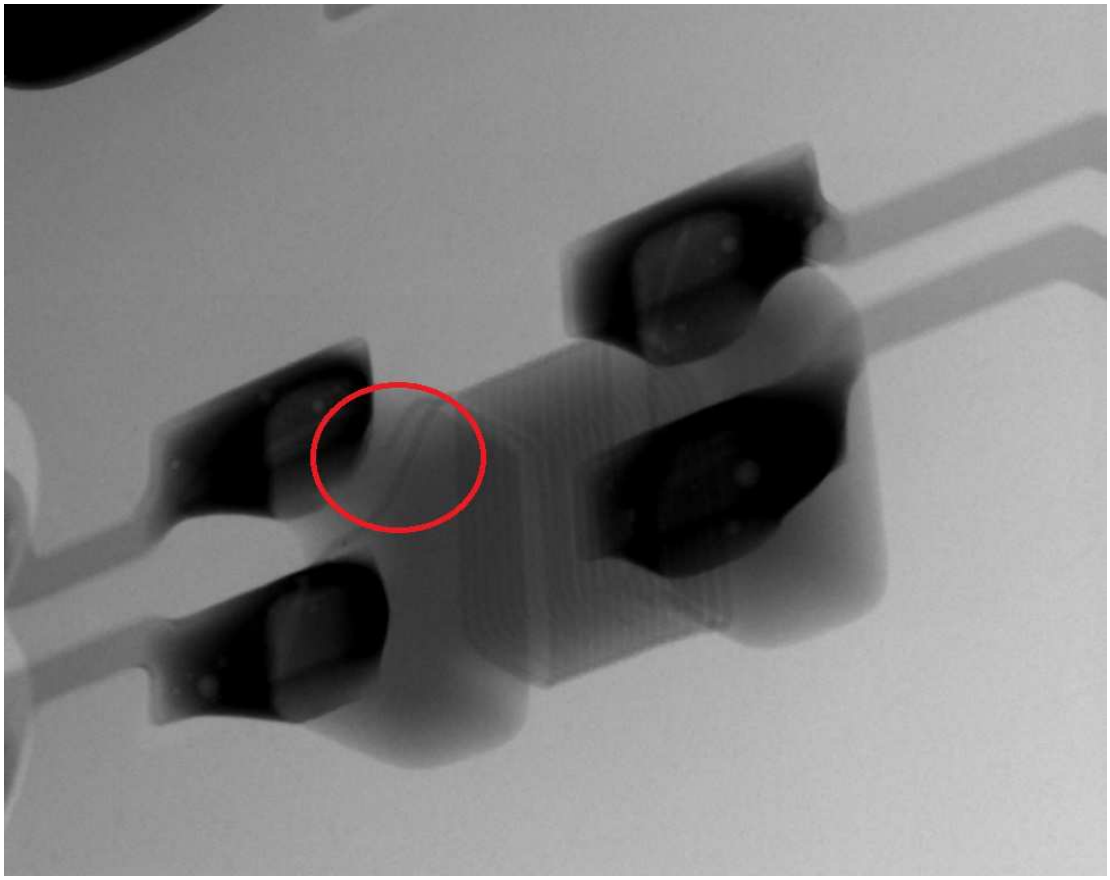
A wider view of the same PCB shows trace failures cause by physical damage to the board in addition to those traces mal-formed at the time of manufacture:



This X-ray photograph shows broken lead wires *inside* of an IC between one of the external terminals and two different points on the IC's internal silicon die:

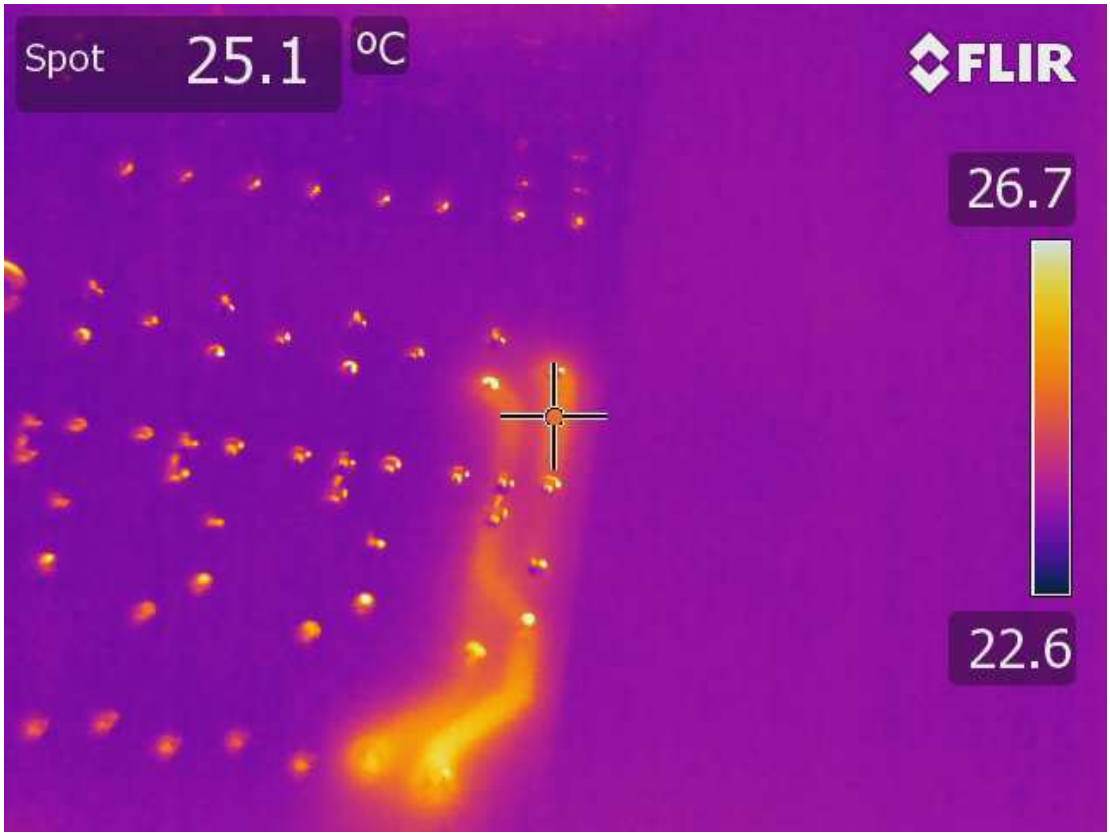


Another example of an internal component fault revealed by X-ray photograph is in this image of a *common-mode balun*[2] soldered to a PCB. These devices consist of two closely-coupled inductors (i.e. a 1:1 transformer) designed to present a high impedance to common-mode currents along a transmission line while presenting negligible impedance to differential currents along the same line. Packaged as a surface-mount device (SMD), we see one broken lead wire at its left end not connecting with the upper-left external terminal as it should:



[2] Also known as a *current balun*. The term "balun" refers to Balanced-Unbalanced, referring to a common application of this device: namely, to convert a balanced signal into an unbalanced signal or vice-versa.

In the following infra-red image we see two overheated traces on a powered PCB, clearly distinguished from the others by their brighter coloring:

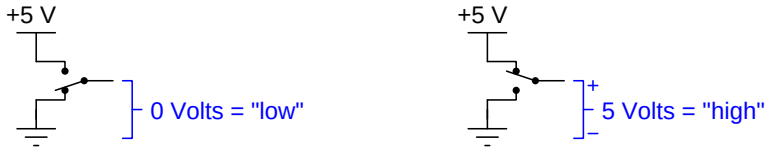


Chapter 3

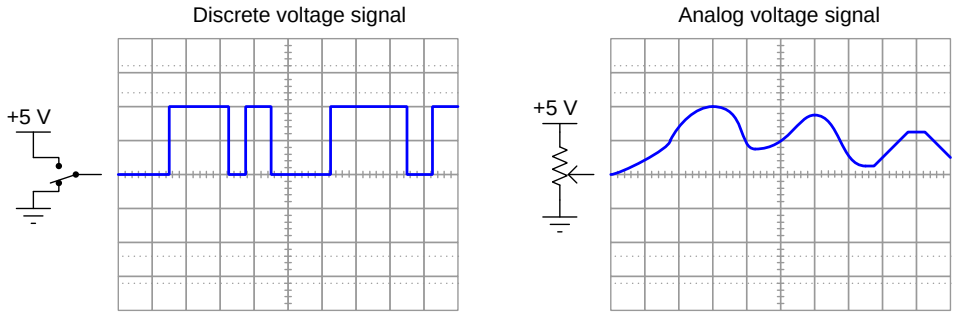
Tutorial

3.1 Digital logic levels

The essence of digital signaling in electronic circuits is the recognition of voltage signals as being either "high" or "low". In the case of a digital circuit operating from a 5 Volt DC power source, an ideal "low" signal is 0 Volts measured with reference to ground while an ideal "high" signal is +5 Volts measured with reference to ground:



When contrasted against analog voltage signals, the distinction is clear: digital signals only have two possible states, while analog signals may attain an infinite number of possible voltage values between full source voltage and zero. We see this contrast in the following illustration showing the high/low dichotomy of a digital signal on the left versus the infinitely-variable range of an analog signal on the right, both signals plotted over time using an oscilloscope[1]:



In real digital electronic circuits, however, "high" signals are rarely equal to DC supply voltage and "low" signals are rarely equal to zero Volts. Instead, we find "high" signals arbitrarily *close* to full DC supply voltage and "low" signals *close* to zero. This practical truth raises a set of important questions for us: *how low may a voltage signal sag and still be considered "high" in the digital sense*, and *how high may a voltage signal rise and still be considered "low" in a digital sense?* In answer to these questions, every class of digital electronic circuits (e.g. a series of logic gates, denoted by part number) must have specified ranges for valid "high" and "low" states, and those specifications must be respected by all manufacturers of digital logic components in order for those components to reliably send and receive signals with other components.

These specifications take the form of four voltage requirements for semiconductor logic gates: V_{OH} (minimum voltage output for a "high" state), V_{OL} (maximum voltage output for a "low" state), V_{IH} (minimum voltage accepted as a "high" input state), and V_{IL} (maximum voltage accepted as a "low" input state).

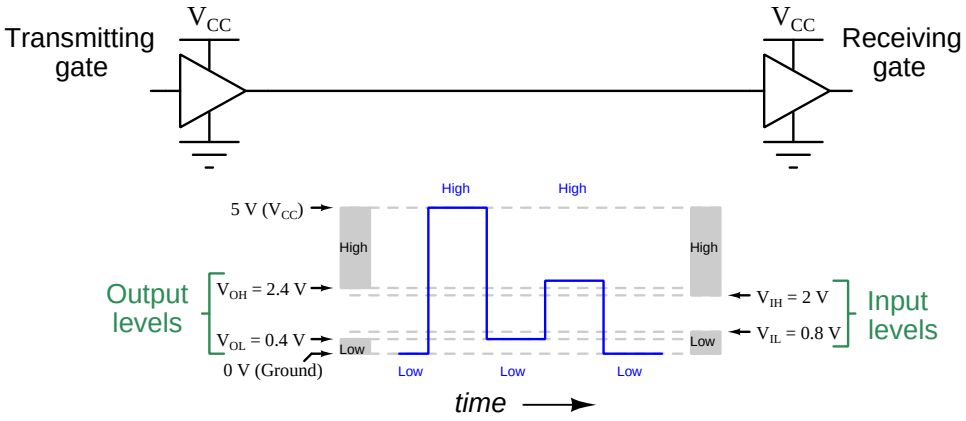
[1] Oscilloscopes are nothing more than graphing voltmeters, showing voltage on the vertical axis and time on the horizontal.

A good illustrative example of logic level specifications for us to explore is the legacy 7400-series *Transistor-to-Transistor Logic* (or *TTL*) logic gates using bipolar transistors internally, designed to operate from the power provided by a regulated 5 Volt DC power source. Shown below is a table declaring the DC voltage levels standardized as "high" and "low" for this family of logic gates:

Logic state	**Guaranteed output**	**Acceptable input**	**Noise margin**
High	V_{OH} = 2.4 Volts min.	V_{IH} = 2.0 Volts min.	2.4 − 2.0 = 0.4 Volts
Low	V_{OL} = 0.4 Volts max.	V_{IL} = 0.8 Volts max.	0.8 − 0.4 = 0.4 Volts

Note how the guaranteed output voltage levels for "high" and "low" differ slightly from the acceptable input voltage levels for "high" and "low" states. The purpose of having two sets of voltage specifications is so that any TTL gate should output a signal that is "stronger" than necessary for any other TTL gate to receive; that is to say, a "high" signal output by a TTL logic gate should exceed the minimum acceptable voltage required by any TTL logic gate receiving that signal at its input, and a "low" signal output by a TTL gate should be closer to ground potential than the maximum acceptable voltage of any other TTL gate receiving that signal.

An illustration shown below graphically presents these minimum and maximum voltage values between transmitting gate and receiving gate:



Note the 0.4 Volt gap between the minimum "high" output level and the minimum "high" input level of the TTL gates, and a similar 0.4 Volt gap between the gates' maximum "low" output level versus the maximum "low" input level. This 0.4 Volt gap is known as the *noise margin* of the TTL logic family, because it represents the amount of additional electrical noise that might be superimposed on the logic signal voltages without falling outside the acceptable input specifications.

Logic level voltages are obviously important for digital logic circuits as they allow manufacturers to produce integrated-circuit digital devices that operate well with each other. Failure to heed these specifications is one reason why logic gates may fail to communicate with each other in any new digital circuit design. For example, if transmitting gates' V_{OH} and/or V_{OL} ratings are incompatible with receiving gates' V_{IH} and/or V_{IL} ratings.

3.2 Multimeters

A basic multimeter set to the DC voltage-sensing function will suffice as a logic-level indicator, connecting the black lead of the meter to the digital circuit's ground terminal and using the red lead to probe test points. The "high" or "low" logic state of the voltage signal may then be interpreted according to the minimum and maximum values specified for the particular logic "family". High/low threshold values for standard TTL and CMOS operating on a 5 Volt supply voltage are shown here:

5-Volt TTL

Logic state	Guaranteed output	Acceptable input	Noise margin
High	V_{OH} = 2.4 Volts min.	V_{IH} = 2.0 Volts min.	2.4 − 2.0 = 0.4 Volts
Low	V_{OL} = 0.4 Volts max.	V_{IL} = 0.8 Volts max.	0.8 − 0.4 = 0.4 Volts

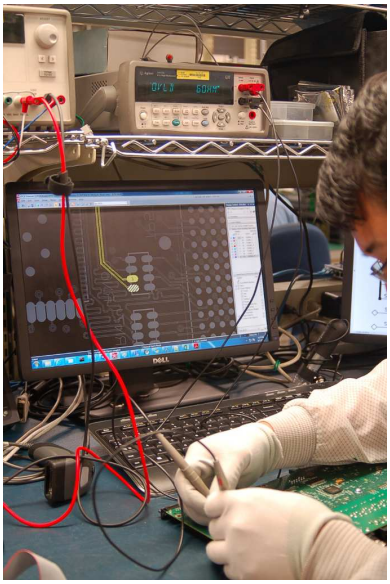
5-Volt CMOS

Logic state	Guaranteed output	Acceptable input	Noise margin
High	V_{OH} = 4.44 Volts min.	V_{IH} = 3.5 Volts min.	4.44 − 3.5 = 0.94 Volts
Low	V_{OL} = 0.5 Volts max.	V_{IL} = 1.5 Volts max.	1.5 − 0.5 = 1.0 Volt

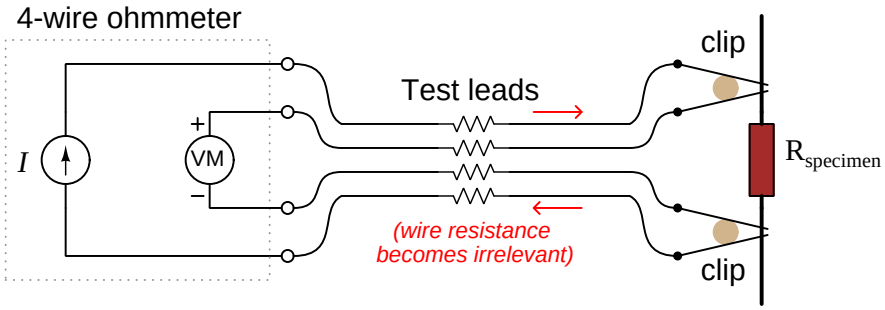
When interpreting DC voltage levels as logic states, one must use the acceptable *input* threshold values (V_{IH} and V_{IL}) to determine how the signal will be interpreted by any receiving logic gates.

Another multimeter function useful for digital circuit diagnosis is *resistance measurement*, to locate "open" or "shorted" faults interfering with digital signals. An "open" fault, of course, will register as an abnormally high resistance while a "shorted" fault will register as an abnormally low resistance. Such measurements, like all resistance checks, must be performed with the circuit de-energized so as to not interfere with the multimeter's internal source.

The following photograph shows an electronics technician using an ohmmeter to check continuity between two copper traces on a printed circuit board (PCB), looking for a "shorted" fault between those traces:



During this particular troubleshooting session, the meter was being used to pinpoint the precise location of the shorted fault in order to determine exactly what sort of manufacturing defect caused it. In order to do this, the technician used special meter probes with two test leads each, with the multimeter set for *four-wire* resistance measurement. This is known as the *Kelvin four-wire technique*, which uses two of the wires to conduct a small excitation current (from a current source inside the multimeter) and the other two wires to sense voltage drop across the specimen being measured:

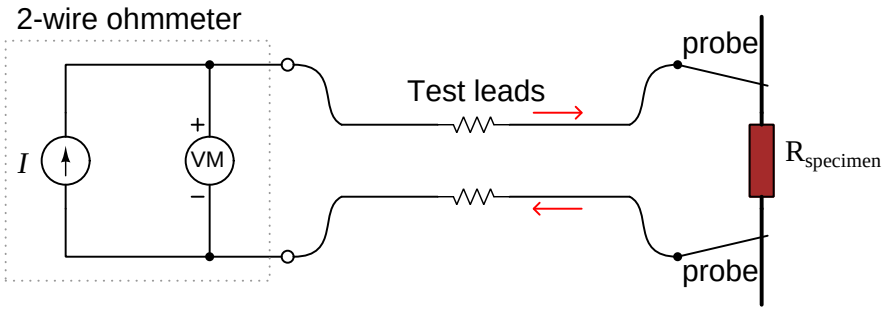


$$R_{specimen} = \frac{\text{Voltmeter indication}}{\text{Current source}}$$

The advantage of this four-wire technique is that test lead resistance becomes irrelevant, allowing the user to make highly precise measurements of resistance free from errors caused by the natural

resistance within the meter's test leads. This enables measurement of extremely small[2] resistance values. In this case that sensitivity was useful to determine *where* along the PCB the two traces were shorted together, as multiple resistance measurements could be taken at different pairs of points on the PCB: the pair with the lowest resistance measurement must be closest to the shorted fault.

By contrast, standard two-wire resistance measurement necessarily includes the test lead resistance along with the specimen's resistance value, resulting in a positive measurement error (i.e. the ohmmeter "thinks" the specimen has more resistance than it actually does):



$$R_{specimen} + R_{leads} = \frac{\text{Voltmeter indication}}{\text{Current source}}$$

[2] In this particular troubleshooting session the measured resistance of the fault was a fraction of an Ohm, about 0.2 Ω. Locating trace-to-trace shorted faults on a PCB is easier than one might think, owing to the relatively high amount of trace resistance per unit distance than for a typical insulated wire. Most PCB signal traces are quite narrow, which means their cross-sectional area is small and therefore they have much more resistance per centimeter of length than most any solid or stranded copper wire.

3.3 Logic probes and pulsers

Multimeters are not intended for the measurement of rapidly pulsing signals, and so are quite limited as diagnostic tools for most digital logic circuits. An alternative to the voltmeter is a device called a *logic probe*, designed to reveal the high/low state of a digital signal by means of two LEDs rather than a numerical display of DC voltage. Signals rapidly oscillating between "high" and "low" states will illuminate both LEDs, making the presence of pulsations evident where they may not be so using a voltmeter.

A photograph[3] of a logic probe appears below, revealing a metal-tipped probe for sensing the logic state of a terminal, plus red and black alligator clips designed to connect to the circuit under test for powering the probe's internal circuitry and providing a ground reference for the sensed logic signal:

Like many logic probes, the model shown in the photograph offers colored LEDs representing "high" and "low" logic states, as well as audio indication in the form of a beeper. The MEM/PULSE selector switch shows real-time logic states in the "PULSE" position but enables a latch in the "MEM" position to capture and indicate short-duration pulses that would otherwise be too transient to see. Such a simple tool, lacking the sophistication of a multimeter, nevertheless offers diagnostic capability most multimeters would be unable to provide.

[3]This photo is courtesy of *Plusea* who posted it to the photo-sharing website Flickr under the Creative Commons Attribution-only license (version 2).

Logic probes, like multimeters, are *passive*-only test instruments. That is, they are able to measure and display logic states, but they cannot alter logic states within a circuit. The "active" equivalent of a logic probe is the *logic pulser* which closely resembles a logic probe in appearance: typically a hand-held "pencil" shaped device with a metal probe and two clips for power. Logic pulsers work by asserting either a "high" or "low" output state at their probe tip with current sourcing/sinking capability great enough to "override" the output of a logic gate to *force* the logic state to the desired level. In other words, a logic pulser over-powers the output transistor stage of a logic gate to assert the desired voltage level in order to make any receiving gates connected to that same line respond.

Normally it is considered a bad thing to have two or more digital output devices assert conflicting logic states onto the same line, as it results in those devices "fighting" each other, one sinking current and the other sourcing current, with the possibility of one or more of them overheating and sustaining internal damage. Logic pulsers avoid this danger by making their over-riding pulse signals very brief. By limiting the time duration of the over-powering pulse to several microseconds, the pulser will not over-drive the output of any competing device long enough to cause thermal damage.

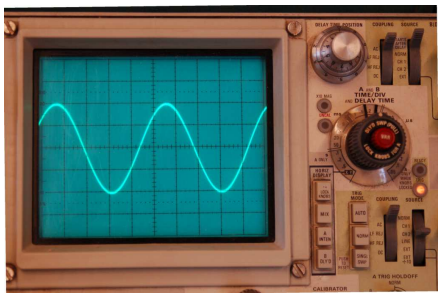
3.4 Breakout boxes

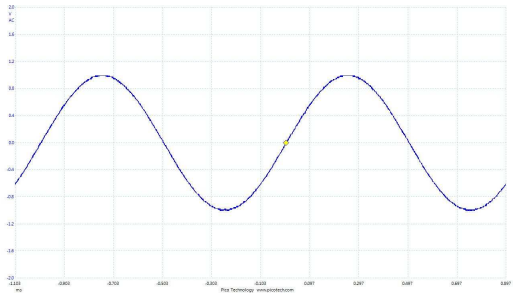
A special-purpose diagnostic tool called a *breakout box* functions as a logic probe designed for use with serial data networks, particularly EIA/TIA-232. Sets of multi-color LEDs indicate "mark" and "space" voltage levels on the data and control lines, enabling one to view the presence of transmitted data as well as hardware handshaking states:

The breakout box shown in this photograph is equipped with male and female 25-pin EIA/TIA-232 connectors, to be inserted in-line with a data cable plugging into a device's serial port. Small switches permit "straight-through" connections from male to female when closed, and when opened those connections are broken. However, female headers positioned along each side of each switch assembly provide means to make custom connections using "jumper" wires. For example, if you wanted to use a breakout box to form a "null modem" connection where transmit and receive lines are swapped, you would open the two switches connecting `TD` to `TD` and `RD` to `RD`, then use jumper wires to connect `TD` to `RD` and `RD` to `TD`. A "Trap" indicator serves to latch transient states, allowing you to see when a momentary pulse occurs that is too brief to view on a real-time LED indicator.

3.5 Oscilloscopes

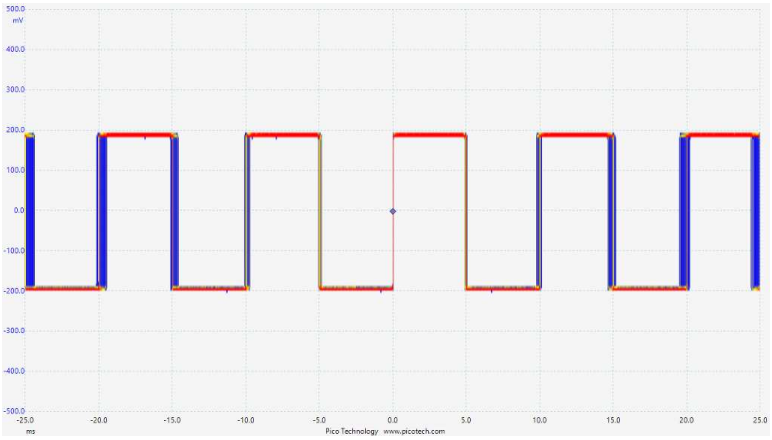
An *oscilloscope* is simply a graphical voltmeter displaying voltage amplitude on the vertical axis of a two-dimensional display and time on the horizontal axis. Below we see photographs of two oscilloscopes displaying a sine-wave-shaped AC voltage signal, the left-hand oscilloscope being a stand-alone analog instrument and the right-hand oscilloscope being a digital model displaying its waveform on the screen of a personal computer:



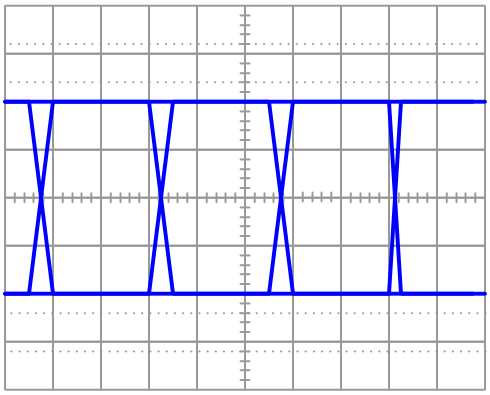


Special oscilloscopes called *Digital phosphor oscilloscopes* are designed to mimic the behavior of old analog oscilloscopes where the relative brightness of the trace represented its frequency of re-occurrence, useful when measuring a repetitive pulse signal to see if successive pulses drift either in amplitude, timing, or both. Legacy analog oscilloscopes with phosphor-based luminescent displays would naturally show a brighter trace where the waveform retraces the same path multiple times, and show a dimmer trace where the waveform only occasionally follows that path. Digital phosphor oscilloscopes mimic that legacy functionality and actually do a step further by *colorizing* the frequency of re-occurrence so that the most frequently retraced paths show as a completely different color compared to seldom-traced paths.

This colorized-occurrence feature is so popular that even some entry-level digital oscilloscopes now offer it. An example is this screenshot from a Picoscope model 2204A where the colorizing function is called *persistence mode*. Red indicates a frequently-traced path, while blue indicates a seldom-traced path. The following screenshot was generated using this model of oscilloscope set to persistence mode, measuring a square wave with a shifting frequency. Note how the waveform is crisp and red near the center of the screen where the triggering cursor is located, but shows more blue regions in the rising and falling edges (shifting position from the trigger point due to the waveform's period changing from sweep to sweep):

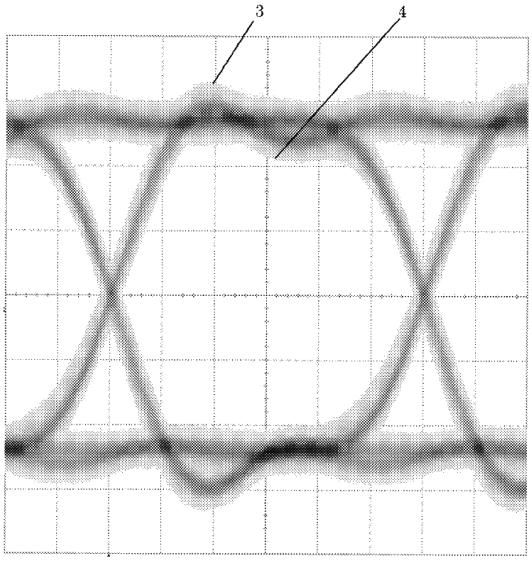


Digital signals in high-speed circuits appear as pulse sequences of high (1) and low (0) states when displayed in the time domain. An alternating sequence of 1 and 0 states would be a simple square wave, but real digital data typically doesn't repeat such simple patterns over and over. Instead, the pattern of high and low states often appears random without any knowledge or context of what those pulses represent in the system at the time of measurement. As such, an oscilloscope configured to repeatedly display a serial digital pulse stream will show something that looks like this:



These are clearly square (or trapezoidal) wave pulses that overlap enough times so that only the high-to-low and low-to-high edges are distinct. At first, this sort of display looks useless because

we really cannot discern individual high or low logic states due to the overlapping of repeated traces. However, when displayed on a digital phosphor oscilloscope, variations in amplitude and timing create what is commonly known as an *eye diagram* which is a good visual representation of uncertainty and therefore of signal integrity. The following image of a greyscale digital phosphor oscillograph was taken from Martin Miller's patent "Noise Analysis To Reveal Jitter And Crosstalk's Effect On Signal Integrity"[4] as part of his explanation of "prior art" (i.e. conventional state-of-the-art) technology for signal integrity analysis:



As with any oscillograph of a digital pulse signal, we can easily identify such imperfections as ringing, overshoot, undershoot, settling time, rise time, and fall time from this image. The added benefit granted by the digital phosphor technology is being able to discern the degree of variation in amplitude and time for successive pulses. For example, the labels 3 and 4 in Miller's image show the highest and lowest amplitude points captured for the "high" portion of the pulse signal over a collection of 5 million sampled pulses, representing a peak-to-peak variance of more than one division on the oscilloscope's vertical axis. Similarly, one could quantify the amount of jitter in this signal stream by measuring the *width* of any rising or falling edge. Identifying the amplitudes and times of these "outlier" signals allows the user to determine worst-case limits for signal integrity in this system[5]. The larger the area inside the "eye" (the "hole"), the greater the signal's integrity.

[4] Miller, Martin T. *US Patent 9,843,402*, "Noise Analysis To Reveal Jitter And Crosstalk's Effect On Signal Integrity", application 4 October 2016, patent granted 12 December 2017. This particular eye diagram results from *5 million* overlaid pulses, the re-occurrence frequency represented by the density of the grey pixels.

[5] A helpful analogy is to picture a city intersection on a snowy day. As vehicles navigate through the intersection, their tires leave tracks in the snow which remain long after each vehicle passes through. After much traffic has passed through the intersection, the culmination of all tracks indicate where most vehicles have driven as well as how far from center-lane the farthest outlying vehicle path was.

3.6 Logic analyzers

A *logic analyzer* is to an oscilloscope what a logic probe is to a voltmeter: an instrument designed to show voltage levels as digital logic states rather than as analog signals. Since each channel of a logic analyzer need only show "high" or "low" rather than precisely resolve the sensed voltage as an analog measurement, it is a simpler instrument to manufacture. This permits logic analyzers to be built with a great many input channels, many more than what is common with an oscilloscope.

A photograph of a legacy Hewlett-Packard logic analyzer appears in this next photograph:

Like an oscilloscope, a logic analyzer displays signals as graphs in the time domain.

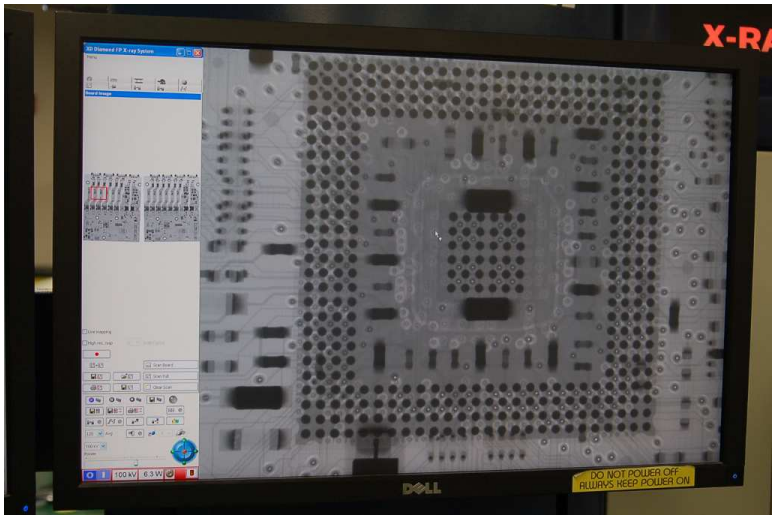
Mixed-signal oscilloscopes blend the capabilities of standard oscilloscopes with logic analyzers, allowing the user to examine digital signals two different ways simultaneously on the same screen. As with logic analyzers, the number of digital input channels is usually quite large, in order to facilitate simultaneous monitoring of multiple bit-lines within a parallel digital bus. Triggering is usually available on either analog or digital channels, allowing the user to synchronize the oscilloscope's display to any arbitrary reference waveform being measured.

3.7 Assembled board testing

Electronics manufacturers must be able to quickly test their finished printed circuit boards (PCBs) in order to check for defects at various stages of the manufacturing process, but especially prior to device packaging and eventual sale to customers.

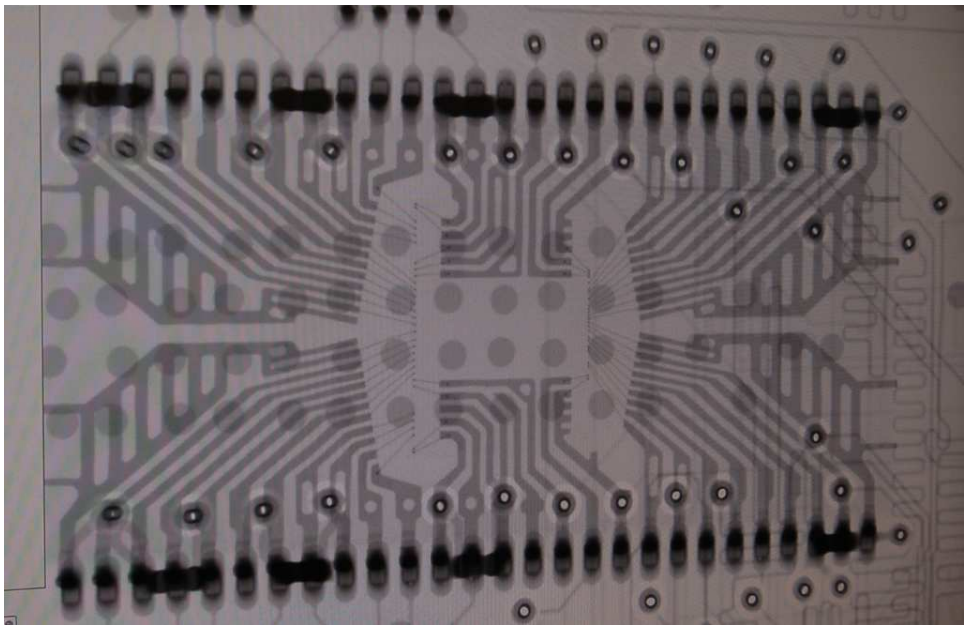
Modern digital circuits constructed on PCBs are often so densely-populated by integrated circuits (ICs, or "chips") that simple visual inspection cannot be relied upon for locating manufacturing and assembly defects such as poor solder joints and misaligned terminals. This is especially true for ICs where the connection terminals lie on the *bottom face* of the chip and are hidden from view by the body of the integrated circuit itself. In such applications an effective means of inspecting such hidden details is to use an *X-ray camera* and source of X-ray radiation to see through the board and placed components. Such a machine is a valuable tool in a high-volume production environment where detection of manufacturing defects must be rapid.

An example of an X-ray camera's view of a *ball grid array* (BGA) style of integrated circuit soldered to a PCB appears below:



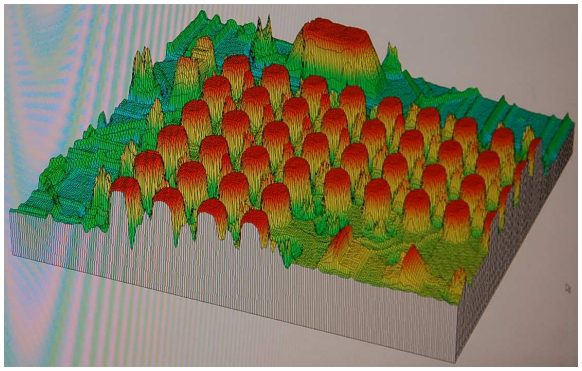
Each dark-grey dot represents a soldered junction between a component terminal and a copper "pad" on the PCB. Certain types of solder defects are evident as non-uniform coloring at the affected dot.

With the appropriate power setting on the X-ray tube, it is possible to "look inside" individual integrated circuits to see the wires connecting the pins to the silicon die inside the chip. An example of this is shown here, where we may see the internal wiring of a DIP (dual-inline package) integrated circuit:



The silicon die for this particular IC is invisible to the X-ray camera at this power setting, but we may see the fine metal wires terminate around its periphery, each of those fine wires connecting to a wider metal trace inside the IC, each trace leading to one of the pins on either side (top or bottom edge) of the chip's exterior.

An advanced feature of some X-ray machines is to render shades of gray as a simulated three-dimensional (and colorized) map, where both color and "height" represent intensity on the grayscale. The following example shows soldered connections on a BGA chip, each solder ball appearing as a red-colored "mesa" on a simulated "landscape" of rendered grayscale image:



As useful as visual and X-ray inspection may be for locating manufacturing defects, there is still a need to electrically test assembled PCBs because some faults cannot be seen. Manually testing densely-populated PCBs for a wide range of possible faults is impractical, and so automated tests exist for this purpose. One such testing method is to fabricate a special board equipped with pointed-tip metal probes designed to simultaneously contact important test points on the assembled PCB, those probes connecting to wires which in turn terminate at some electronic device built to inject test signals and measure responses from the PCB. This is colloquially referred to as a *bed-of-nails* test, referencing the appearance of the metal probes.

A more modern and sophisticated variant of the "bed of nails" test fixture is something called a *flying probe*. This uses robotically-controlled actuators to move one or more probes to different test points on the PCB, injecting and/or measuring signals at those points according to a programmed sequence. The high speed of modern robotic actuators accounts for the "flying" label, and they are impressive to watch in action.

3.8 JTAG

Bed-of-nails and flying probe tests are both electrically-based tests designed to ensure the integrity of soldered connections between components and copper traces on a PCB. These legacy techniques, however, are less useful when the spacing between adjacent pins on ICs is very small because it becomes more difficult to accurately place the probe tips. They are completely useless for ICs such as the Ball Grid Array (BGA) type where most of the IC terminals are sandwiched between the chip and the PCB rather than being positioned around the periphery.

In answer to this challenge, an industry group calling itself the *Joint Test Action Group* formed in the mid-1980's with the purpose of developing a standardized method for testing PCB connections where IC terminals were either too closely-spaced or entirely inaccessible for contact with metal probes. Their work culminated in an IEEE standard (1149.1) in the year 1990, commonly referred to in the industry as *JTAG*. This standard developed a means by which diagnostic capability could be *built into* such ICs so that the chips themselves could serve as the testing fixture.

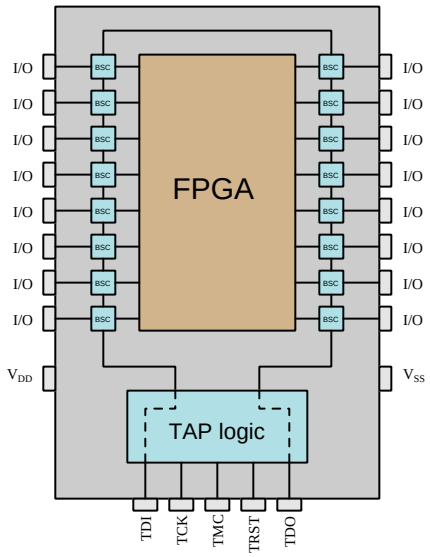
JTAG is part of a larger design philosophy known as *Design For Test* (*DFT*)[6] which simply means that the electronic product and/or its constituent components are designed to facilitate convenient testing, not merely to fulfill their essential functions. A very simple example of "design for test" is to intentionally place exposed test points on a PCB enabling technicians to easily make contact with multimeter and/or oscilloscope probes, in contrast to a board design lacking such test points which would require extra work to reliably attach test probes. JTAG extends this concept to the pins within digital ICs, building-in testing capability to the IC itself so that the final assembly may be more easily validated and diagnosed.

The basic concept of JTAG-compliant ICs is that these digital devices are built with extra circuitry called *boundary scan cells* positioned between the IC's external pins and the functional circuitry of the device. Each cell is able to transparently pass digital data between its respective pin and the digital logic, which is its normal mode of operation, or it has the ability to read or write data to be passed along to other boundary scan cells in serial shift-register fashion for analysis outside of the circuit board.

[6] This is closely related to a concept I like to call *Design For Maintenance* which is where a product is designed to be easily maintained and repaired by the eventual owner or service personnel. Anyone who had had to do major mechanical work to a modern automobile is likely familiar with designs that were *not* made with maintenance in mind!

To illustrate, we will examine a simplified diagram of a Field-Programmable Gate Array (FPGA)[7] chip with this additional JTAG circuitry shown. Each input/output (I/O) pin of the FPGA chip has a boundary scan cell (BSC) between it and the FPGA. All of this circuitry, of course, resides on the same wafer of silicon inside the IC, but is shown separately here in order to represent its function:

Field-Programmable Gate Array equipped with JTAG capability



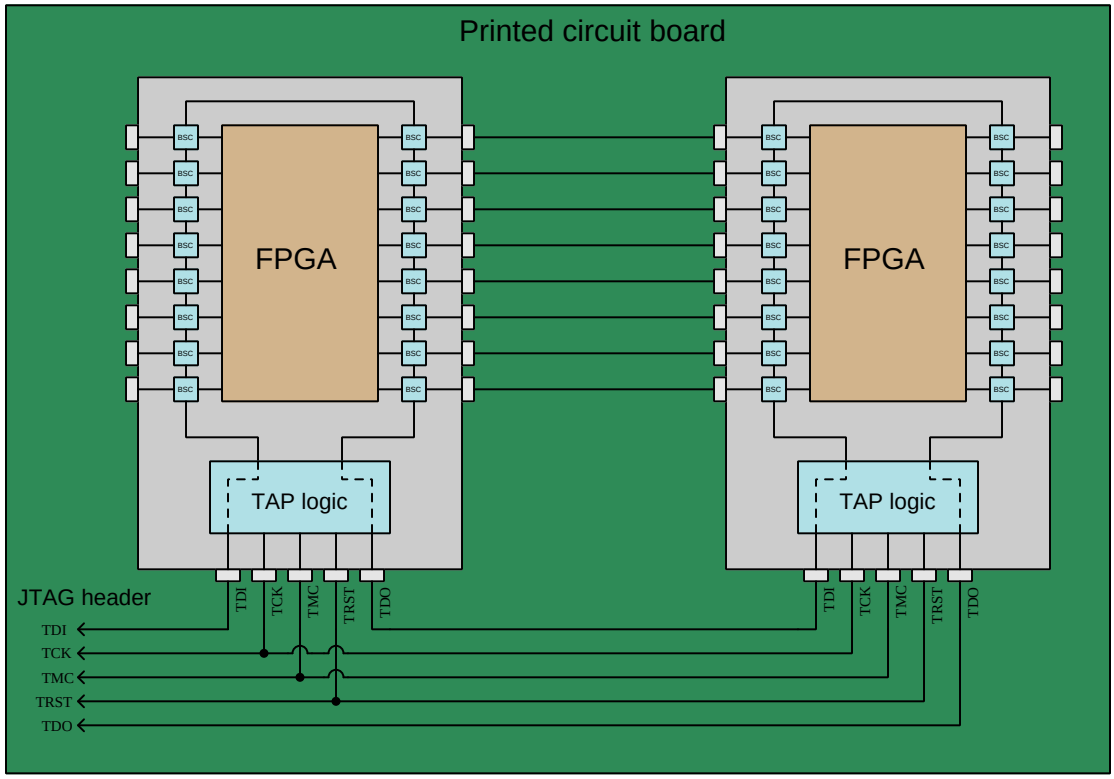
A set of dedicated pins on the IC serve as the JTAG interface. This is called the *Test Access Port* (TAP) by the IEEE 1149.1 standard, and consists of four mandatory pins and one optional pin for controlling the boundary scan cells and intercepting pin data. These TAP pins connect to an external testing device which will generate the necessary signals to activate the JTAG functions.

- `TDI` = Test Data In
- `TDO` = Test Data Out
- `TCK` = Test Clock
- `TMS` = Test Mode Select
- `TRST` = Test Reset (optional)

When performing a boundary scan, for example, serial data output by a testing device enters the JTAG `TDI` pin and is clocked through the device one boundary scan cell at a time in step with the test clock signal (`TCK`) which also comes from the testing device. Eventually these serial bits return to the test device through the `TDO` pin.

[7] FPGAs are configurable logic arrays capable of implementing arbitrary logic functions. They are useful in prototype designs where digital logic may need to be easily modified, and they also find use in systems where a large amount of custom logic must be located in a small area on the PCB. JTAG, of course, is applicable to *any* digital logic IC, but FPGAs were chosen for this example for their arbitrary nature.

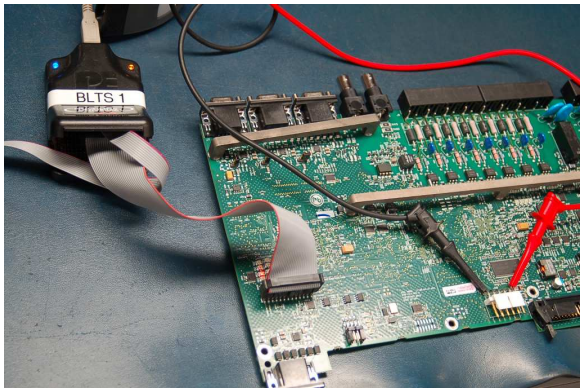
The purpose of a boundary scan test is to ensure proper connections between the JTAG-capable IC and other components on the PCB its data lines connect to. Therefore, to do a proper boundary scan test, we need multiple JTAG-capable components on the board connecting to each other. For example, consider the following pair of interconnected FPGA devices:



As you can see, the `TDO` pin of the left IC connects to the `TDI` pin of the right IC, so that the two form a long shift register "chain" for JTAG testing data. As this test data steps through all the boundary scan cells, it causes the respective output pins on the devices to output the bits given to them, and causes the respective input pins to receive the data and write it to the serial data stream. Therefore, at the end of the test some of the bits in the data stream received at the board's `TDO` pin will represent data conveyed from one device to the other over traces on the PCB connecting those two devices together. If those data bits fail to match the sequence predicted by the testing device, it means a connection problem exists on the PCB and it therefore fails the test.

A single "header" connector located on the PCB provides convenient access for the JTAG testing device to connect to the board, to inject the serial test data stream, to read the received bit states from the "scan chain" formed by those two devices, and also to control the JTAG testing sequence (clock signal, mode signal, reset signal).

In the following photograph you see a PCB connected to a personal computer running JTAG testing software, the interface being a USB-based scanning tool that connects to the board's JTAG header through a ribbon-style cable. A red and black wire pair provide DC power to energize the board's components:



Not only is JTAG useful for performing boundary scan tests between interconnected ICs, but its ability to read and to force data bits on individual device pins also makes JTAG a powerful "debugging" tool to monitor the logic states inside digital components, as well as a viable means of programming writable digital devices such as microcontrollers with on-board Flash memory.

Some manufacturers have opted to incorporate JTAG technology into products lacking the necessary number of pins to provide the standard TAP interface (`TDI`, `TDO`, `TCK`, and `TMS`), by adding serial data converters between an interface using fewer than four pins and an internal TAP. An example of this is found in some of the models of Texas Instruments' MSP430 microcontrollers, which use a two-terminal "Spi-Bi-Wire" (SBW) interface to encode four-wire JTAG signals.

Chapter 4

Derivations and Technical References

This chapter is where you will find mathematical derivations too detailed to include in the tutorial, and/or tables and other technical reference material.

4.1 TTL logic levels

Logic gates need to reliably communicate with each other, the voltage levels output by the "driving" gate being compatible with the input voltage levels of the "receiving" gate. These specifications are given as follows:

- V_{OH} = minimum voltage output by a gate in the "high" (1) state
- V_{OL} = maximum voltage output by a gate in the "low" (0) state
- V_{IH} = minimum voltage received by a gate to be interpreted as a "high" (1) state
- V_{IL} = maximum voltage received by a gate to be interpreted as a "low" (0) state

Classic "LS" family of TTL gate circuits using bipolar transistors operated on a 5 Volt DC power supply, outputting at least 2.4 Volts in the "high" state and 0.4 Volts in the "low" state. The same logic family would accept any input voltage signal greater than 2.0 Volts as a "high" and less than 0.8 Volts as a "low". The amount of *noise margin* is the difference between the guaranteed output voltage levels and the acceptable input voltage levels. Represented in table form for 5-Volt TTL gates:

Logic state	**Guaranteed output**	**Acceptable input**	**Noise margin**
High	$V_{OH} = 2.4$ Volts min.	$V_{IH} = 2.0$ Volts min.	$2.4 - 2.0 = 0.4$ Volts
Low	$V_{OL} = 0.4$ Volts max.	$V_{IL} = 0.8$ Volts max.	$0.8 - 0.4 = 0.4$ Volts

If an ordinary digital logic gate receives an input voltage signal lying somewhere between the threshold values of V_{IH} and V_{IL}, its output state will be unpredictable. The gate may "assume" either a high or a low state, or worse yet may try to process that signal in an analog fashion, generating an output voltage level below V_{OH} and above V_{OL}, thus propagating the problem to the next logic gate(s). For this reason it is very important to design logic circuits to avoid voltage levels below V_{IH} and above V_{IL}.

Classic TTL logic IC part numbers begin with either 54 (military-grade) or 74 (commercial grade), but not all 54- or 74- series ICs are of traditional TTL (bipolar transistor) design. For example, the "HC" series of 54- and 74-numbered ICs utilize MOSFETs rather than bipolar transistors, but are otherwise designed to be pin-for-pin compatible with classic TTL logic ICs. These "high-speed CMOS" ICs are designed to function nearly identically to their bipolar-transistor counterparts, but with much lower current requirements and a slightly wider power supply range (2 to 6 Volts typical). Like classic 4000-series CMOS logic ICs, this means the minimum and maximum acceptable voltage levels for 54HC- or 74HC- series ICs "high" and "low" signals vary with supply voltage. By contrast, since classic TTL digital logic ICs are constrained to a very narrow range of DC power supply voltage (typically 4.75 to 5.25 Volts) their acceptable voltage levels for "high" and "low" signals is correspondingly fixed.

4.2 CMOS logic levels

Logic gates need to reliably communicate with each other, the voltage levels output by the "driving" gate being compatible with the input voltage levels of the "receiving" gate. These specifications are given as follows:

- V_{OH} = minimum voltage output by a gate in the "high" (1) state
- V_{OL} = maximum voltage output by a gate in the "low" (0) state
- V_{IH} = minimum voltage received by a gate to be interpreted as a "high" (1) state
- V_{IL} = maximum voltage received by a gate to be interpreted as a "low" (0) state

Classic "CD" family of CMOS gate circuits using complementary MOSFET transistors operating on a 5 Volt DC power supply has much wider greater noise margins than "LS" series TTL, outputting at least 4.44 Volts in the "high" state and 0.5 Volts in the "low" state, while accepting any input voltage signal greater than 3.5 Volts as a "high" and less than 1.5 Volts as a "low". Represented in table form for 5-Volt CMOS gates:

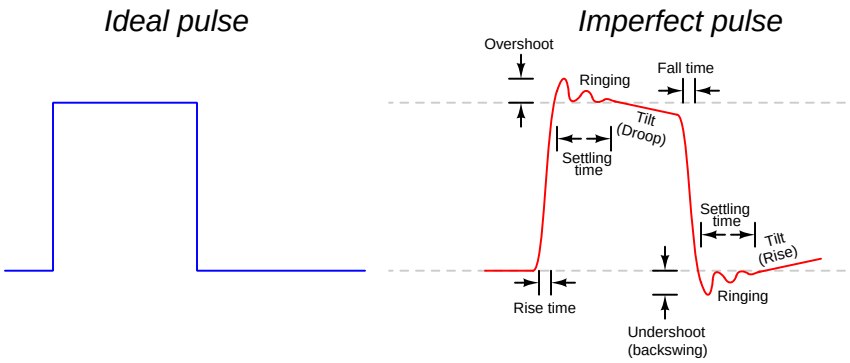
Logic state	**Guaranteed output**	**Acceptable input**	**Noise margin**
High	$V_{OH} = 4.44$ Volts min.	$V_{IH} = 3.5$ Volts min.	$4.44 - 3.5 = 0.94$ Volts
Low	$V_{OL} = 0.5$ Volts max.	$V_{IL} = 1.5$ Volts max.	$1.5 - 0.5 = 1.0$ Volt

CD-family CMOS logic was not limited to a 5 Volt DC power supply as was the LS (TTL) bipolar family of logic gates. Typical DC power supply limits ranged from 3 Volts to 18 Volts, with acceptable input voltage levels varying as a function of power supply voltage. Output voltage levels for a CD-family logic gate also followed power supply voltage, typically within 0.5 Volts of the power supply rails.

If an ordinary digital logic gate receives an input voltage signal lying somewhere between the threshold values of V_{IH} and V_{IL}, its output state will be unpredictable. The gate may "assume" either a high or a low state, or worse yet may try to process that signal in an analog fashion, generating an output voltage level below V_{OH} and above V_{OL}, thus propagating the problem to the next logic gate(s). For this reason it is very important to design logic circuits to avoid voltage levels below V_{IH} and above V_{IL}.

4.3 Digital pulse criteria

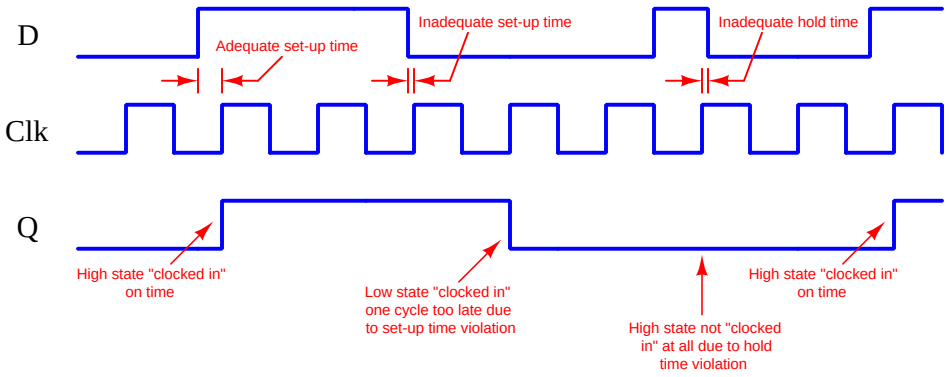
Ideal "pulse" signals have infinitely-steep rise and fall times, level "high" and "low" states well within the specified voltage ranges, and no other imperfections or artifacts. Real pulses always deviate from ideal, often in multiple ways:



Rise and fall times are strongly influenced by parasitic capacitance existing on the signal line with reference to ground, as well as the current-sourcing and current-sinking capability of the logic gate or other device generating the pulse signal. The capacitive "Ohm's Law" formula $I = C\frac{dV}{dt}$ predicts how much current will be necessary to create a linear rate-of-rise or rate-of-fall of voltage for a given capacitance. Note that to achieve infinitely steep rise or fall times an *infinite* amount of current would be necessary to charge/discharge whatever capacitance happens to exist on that signal line!

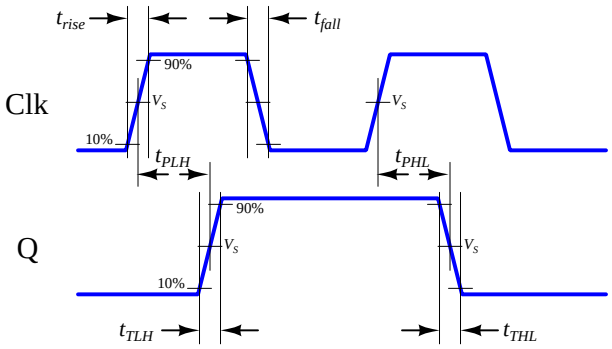
Ringing is caused by *resonance* occurring between parasitic capacitance and parasitic inductance, those two phenomena naturally exchanging energy back and forth with each other to produce the oscillatory waves we call "ringing". Overshoot and undershoot are also the result stored energy within these parasitic L and C circuit properties, and since parasitic capacitance and parasitic inductance can never be fully eliminated from any real circuit it means the effects of over/undershoot and ringing is likewise unavoidable. If you don't see these effects in your pulses, you just aren't viewing them at a fast enough time scale!

Clock-synchronized digital logic circuits such as counters, shift registers, and microprocessors require their input signals to be at stable states immediately before and immediately after the clock pulse arrives. For example, the following timing diagram shows input and output states for a D-type flip-flop circuit (positive-edge triggered), showing the effects of some signal timing violations:



Datasheets for digital circuits often provide timing diagrams showing criteria related to pulse signal timing and logic states. These diagrams don't typically show ideal square-edged pulses, but rather *trapezoidal* pulse profiles intended to exaggerate realistic features such as rise and fall times, propagation delays, and minimum set-up/hold times. Such diagrams usually confuse students who are accustomed to seeing square-edged pulses in their textbook timing diagrams. This technical reference will show some typical timing diagrams and explain what they represent.

For example, consider this timing diagram for a positive-edge-triggered JK flip-flop having both its J and K inputs tied high so as to maintain the circuit in its "toggle" mode. As such we would expect its output (Q) to change state with every rising edge of the clock pulse:



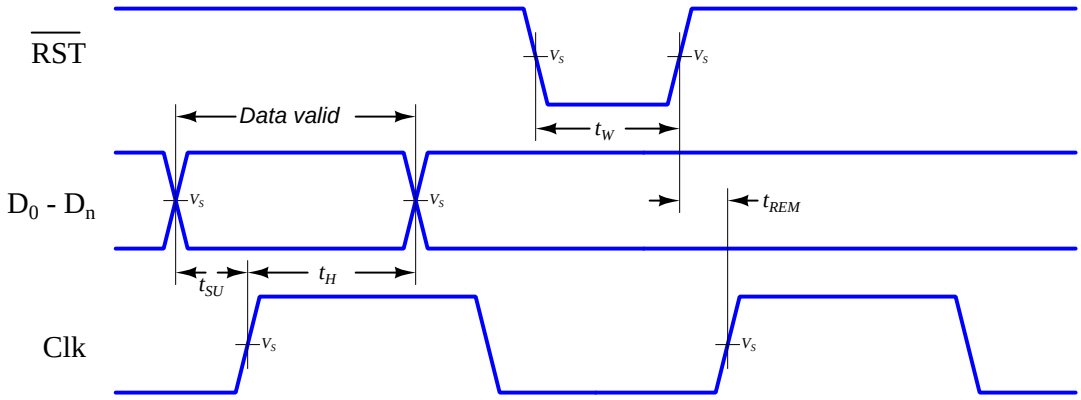
Each of the labels found in this diagram is defined as follows:

- t_{rise} = Rise time of input signal, typically measured from 10% of signal amplitude to 90% of signal amplitude
- t_{fall} = Fall time of input signal, typically measured from 90% of signal amplitude to 10% of signal amplitude
- t_{TLH} = Low-to-High transition time of output signal, typically measured from 10% of signal amplitude to 90% of signal amplitude (the same concept as rise time, but applied to the output signal instead of the input signal)
- t_{THL} = High-to-Low transition time of output signal, typically measured from 90% of signal amplitude to 10% of signal amplitude (the same concept as fall time, but applied to the output signal instead of the input signal)
- t_{PLH} = Propagation delay time of output signal when switching from low to high
- t_{PHL} = Propagation delay time of output signal when switching from high to low
- V_S = Switching threshold voltage, typically defined as 50% of signal amplitude

This timing diagram shows how a digital logic circuit reacts to a single input signal, in this case the clock pulse. Although this example happens to be for a JK flip-flop in toggle mode, the same type of timing diagram with its exaggerated rise/fall times and propagation delays could be applied to any digital logic gate whose output state depended solely on the state of a single input.

For synchronous digital logic circuits where input signals must coordinate with the clock pulse signal in order to be properly accepted by the circuit, we typically find timing diagrams comparing these input states to each other, often without showing the output(s) at all. Instead of showing us how the digital logic circuit will react to an input signal, this sort of timing diagram shows what the digital logic circuit *expects* of its multiple input signals.

The example is shown here for a positive-edge-triggered D register[1] having multiple data lines (D_0 through D_n), one asynchronous[2] reset line ($\overline{RST}$), and one clock input. The arbitrary logic levels of the multiple data lines are shown as a pair of complementary-state pulse waveforms, the only relevant features being the *timing* of the data and not the particular voltage levels of the data signals:



Labels shown in this diagram refer to *minimum* time durations the logic circuit requires for reliable operation:

- t_{SU} = Minimum set-up time before the arrival of the next clock pulse
- t_H = Minimum hold time following the last clock pulse
- t_W = Minimum width (duration) of the asynchronous reset pulse
- t_{REM} = Minimum removal time before the arrival of the next clock pulse

Violations of any of these minimum times may result in unexpected behavior from the logic circuit, and is an all-too-common cause of spurious errors in high-speed digital circuit designs. The assessment of digital pulse signals with regard to reliable circuit operation is generally known as *digital signal integrity*.

[1] In this case, a "D register" is synonymous with multiple D-type flip-flops sharing a common clock input, passing data through from each D input to each corresponding Q output synchronously with each clock pulse.

[2] To review, a *synchronous* input depends on a clock pulse while an *asynchronous* input is able to affect the circuit independent of the clock pulse.

Chapter 5

Questions

This learning module, along with all others in the ModEL collection, is designed to be used in an inverted instructional environment where students independently read[1] the tutorials and attempt to answer questions on their own *prior* to the instructor's interaction with them. In place of lecture[2], the instructor engages with students in Socratic-style dialogue, probing and challenging their understanding of the subject matter through inquiry.

Answers are not provided for questions within this chapter, and this is by design. Solved problems may be found in the Tutorial and Derivation chapters, instead. The goal here is *independence*, and this requires students to be challenged in ways where others cannot think for them. Remember that you always have the tools of *experimentation* and *computer simulation* (e.g. SPICE) to explore concepts!

The following lists contain ideas for Socratic-style questions and challenges. Upon inspection, one will notice a strong theme of *metacognition* within these statements: they are designed to foster a regular habit of examining one's own thoughts as a means toward clearer thinking. As such these sample questions are useful both for instructor-led discussions as well as for self-study.

[1] Technical reading is an essential academic skill for any technical practitioner to possess for the simple reason that the most comprehensive, accurate, and useful information to be found for developing technical competence is in textual form. Technical careers in general are characterized by the need for continuous learning to remain current with standards and technology, and therefore any technical practitioner who cannot read well is handicapped in their professional development. An excellent resource for educators on improving students' reading prowess through intentional effort and strategy is the book textitReading For Understanding – How Reading Apprenticeship Improves Disciplinary Learning in Secondary and College Classrooms by Ruth Schoenbach, Cynthia Greenleaf, and Lynn Murphy.

[2] Lecture is popular as a teaching method because it is easy to implement: any reasonably articulate subject matter expert can talk to students, even with little preparation. However, it is also quite problematic. A good lecture always makes complicated concepts seem easier than they are, which is bad for students because it instills a false sense of confidence in their own understanding; reading and re-articulation requires more cognitive effort and serves to verify comprehension. A culture of teaching-by-lecture fosters a debilitating dependence upon direct personal instruction, whereas the challenges of modern life demand independent and critical thought made possible only by gathering information and perspectives from afar. Information presented in a lecture is ephemeral, easily lost to failures of memory and dictation; text is forever, and may be referenced at any time.

General challenges following tutorial reading

- Summarize as much of the text as you can in one paragraph of your own words. A helpful strategy is to explain ideas as you would for an intelligent child: as simple as you can without compromising too much accuracy.
- Simplify a particular section of the text, for example a paragraph or even a single sentence, so as to capture the same fundamental idea in fewer words.
- Where did the text make the most sense to you? What was it about the text's presentation that made it clear?
- Identify where it might be easy for someone to misunderstand the text, and explain why you think it could be confusing.
- Identify any new concept(s) presented in the text, and explain in your own words.
- Identify any familiar concept(s) such as physical laws or principles applied or referenced in the text.
- Devise a proof of concept experiment demonstrating an important principle, physical law, or technical innovation represented in the text.
- Devise an experiment to disprove a plausible misconception.
- Did the text reveal any misconceptions you might have harbored? If so, describe the misconception(s) and the reason(s) why you now know them to be incorrect.
- Describe any useful problem-solving strategies applied in the text.
- Devise a question of your own to challenge a reader's comprehension of the text.

General follow-up challenges for assigned problems

- Identify where any fundamental laws or principles apply to the solution of this problem, especially before applying any mathematical techniques.
- Devise a thought experiment to explore the characteristics of the problem scenario, applying known laws and principles to mentally model its behavior.
- Describe in detail your own strategy for solving this problem. How did you identify and organized the given information? Did you sketch any diagrams to help frame the problem?
- Is there more than one way to solve this problem? Which method seems best to you?
- Show the work you did in solving this problem, even if the solution is incomplete or incorrect.
- What would you say was the most challenging part of this problem, and why was it so?
- Was any important information missing from the problem which you had to research or recall?
- Was there any extraneous information presented within this problem? If so, what was it and why did it not matter?
- Examine someone else's solution to identify where they applied fundamental laws or principles.
- Simplify the problem from its given form and show how to solve this simpler version of it. Examples include eliminating certain variables or conditions, altering values to simpler (usually whole) numbers, applying a limiting case (i.e. altering a variable to some extreme or ultimate value).
- For quantitative problems, identify the real-world meaning of all intermediate calculations: their units of measurement, where they fit into the scenario at hand. Annotate any diagrams or illustrations with these calculated values.
- For quantitative problems, try approaching it qualitatively instead, thinking in terms of "increase" and "decrease" rather than definite values.
- For qualitative problems, try approaching it quantitatively instead, proposing simple numerical values for the variables.
- Were there any assumptions you made while solving this problem? Would your solution change if one of those assumptions were altered?
- Identify where it would be easy for someone to go astray in attempting to solve this problem.
- Formulate your own problem based on what you learned solving this one.

General follow-up challenges for experiments or projects

- In what way(s) was this experiment or project easy to complete?
- Identify some of the challenges you faced in completing this experiment or project.

- Show how thorough documentation assisted in the completion of this experiment or project.
- Which fundamental laws or principles are key to this system's function?
- Identify any way(s) in which one might obtain false or otherwise misleading measurements from test equipment in this system.
- What will happen if (component X) fails (open/shorted/etc.)?
- What would have to occur to make this system unsafe?

5.1 Conceptual reasoning

These questions are designed to stimulate your analytic and synthetic thinking[3]. In a Socratic discussion with your instructor, the goal is for these questions to prompt an extended dialogue where assumptions are revealed, conclusions are tested, and understanding is sharpened. Your instructor may also pose additional questions based on those assigned, in order to further probe and refine your conceptual understanding.

Questions that follow are presented to challenge and probe your understanding of various concepts presented in the tutorial. These questions are intended to serve as a guide for the Socratic dialogue between yourself and the instructor. Your instructor's task is to ensure you have a sound grasp of these concepts, and the questions contained in this document are merely a means to this end. Your instructor may, at his or her discretion, alter or substitute questions for the benefit of tailoring the discussion to each student's needs. The only absolute requirement is that each student is challenged and assessed at a level equal to or greater than that represented by the documented questions.

It is far more important that you convey your *reasoning* than it is to simply convey a correct answer. For this reason, you should refrain from researching other information sources to answer questions. What matters here is that *you* are doing the thinking. If the answer is incorrect, your instructor will work with you to correct it through proper reasoning. A correct answer without an adequate explanation of how you derived that answer is unacceptable, as it does not aid the learning or assessment process.

You will note a conspicuous lack of answers given for these conceptual questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your conceptual answers, where applicable, is to use circuit simulation software to explore the effects of changes made to circuits. For example, if one of these conceptual questions challenges you to predict the effects of altering some component parameter in a circuit, you may check the validity of your work by simulating that same parameter change within software and seeing if the results agree.

[3] *Analytical* thinking involves the "disassembly" of an idea into its constituent parts, analogous to dissection. *Synthetic* thinking involves the "assembly" of a new idea comprised of multiple concepts, analogous to construction. Both activities are high-level cognitive skills, extremely important for effective problem-solving, necessitating frequent challenge and regular practice to fully develop.

5.1.1 Reading outline and reflections

"Reading maketh a full man; conference a ready man; and writing an exact man" – Francis Bacon

Francis Bacon's advice is a blueprint for effective education: reading provides the learner with knowledge, writing focuses the learner's thoughts, and critical dialogue equips the learner to confidently communicate and apply their learning. Independent acquisition and application of knowledge is a powerful skill, well worth the effort to cultivate. To this end, students should read these educational resources closely, journal their own reflections on the reading, and discuss in detail their findings with classmates and instructor(s). You should be able to do all of the following after reading any instructional text:

☑ Briefly SUMMARIZE THE TEXT in the form of a journal entry documenting your learning as you progress through the course of study. Share this summary in dialogue with your classmates and instructor. Journaling is an excellent self-test of thorough reading because you cannot clearly express what you have not read or did not comprehend.

☑ Demonstrate ACTIVE READING STRATEGIES, including verbalizing your impressions as you read, simplifying long passages to convey the same ideas using fewer words, annotating text and illustrations with your own interpretations, working through mathematical examples shown in the text, cross-referencing passages with relevant illustrations and/or other passages, identifying problem-solving strategies applied by the author, etc. Technical reading is a special case of problem-solving, and so these strategies work precisely because they help solve any problem: paying attention to your own thoughts (metacognition), eliminating unnecessary complexities, identifying what makes sense, paying close attention to details, drawing connections between separated facts, and noting the successful strategies of others.

☑ Identify IMPORTANT THEMES, especially GENERAL LAWS and PRINCIPLES, expounded in the text and express them in the simplest of terms as though you were teaching an intelligent child. This emphasizes connections between related topics and develops your ability to communicate complex ideas to anyone.

☑ Form YOUR OWN QUESTIONS based on the reading, and then pose them to your instructor and classmates for their consideration. Anticipate both correct and incorrect answers, the incorrect answer(s) assuming one or more plausible misconceptions. This helps you view the subject from different perspectives to grasp it more fully.

☑ Devise EXPERIMENTS to test claims presented in the reading, or to disprove misconceptions. Predict possible outcomes of these experiments, and evaluate their meanings: what result(s) would confirm, and what would constitute disproof? Running mental simulations and evaluating results is essential to scientific and diagnostic reasoning.

☑ Specifically identify any points you found CONFUSING. The reason for doing this is to help diagnose misconceptions and overcome barriers to learning.

5.1.2 Foundational concepts

Correct analysis and diagnosis of electric circuits begins with a proper understanding of some basic concepts. The following is a list of some important concepts referenced in this module's full tutorial. Define each of them in your own words, and be prepared to illustrate each of these concepts with a description of a practical example and/or a live demonstration.

Digital signal

Logic state

Sourcing

Sinking

CMOS logic

TTL

Clocking

Open

Short

Kelvin four-wire method

Latch

Timing diagram

Serial versus Parallel data

Time domain

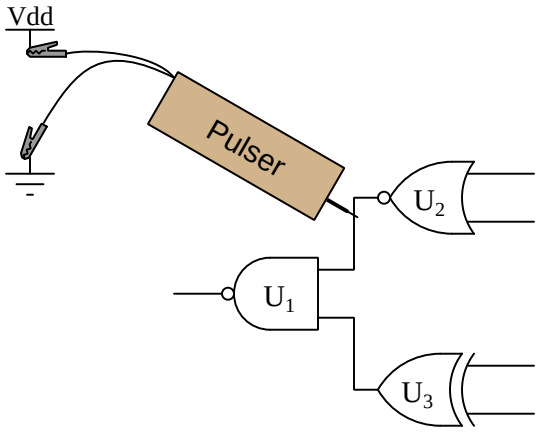
Design For Testing

Boundary scan

Soldering

5.1.3 Logic pulser usage

A technician uses a logic pulser to force the logic state of the wire connecting two gates together:



Which gate, or gates, are we testing by placing the pulser in this position? What other instrument(s) would we have to connect to the circuit (and where?) to complete the test? Why does the logic pulser require a ground connection to do its job in this circuit?

Challenges

- What logic state should the *other* input of the NAND gate need to be in for this test to give a conclusive result? Explain why.

5.1.4 Reverse-engineering techniques

A variety of sophisticated tools exist for manufacturers to examine details inside of the integrated circuits of their competitors, in an effort to copy intellectual property and/or to document vulnerabilities. Once an IC has been "decapsulated" to expose its silicon die to inspection, the following techniques may be employed to read and write digital logic states:

- Emission microscopy (observing light emitted by transistor junctions)
- Laser light manipulation of bit states
- "Nanoprobing" signal lines

Explain how each of these techniques works.

Challenges

- Describe one practical example where a highly skilled group of engineers and technicians might be motivated to employ such techniques.

5.2 Quantitative reasoning

These questions are designed to stimulate your computational thinking. In a Socratic discussion with your instructor, the goal is for these questions to reveal your mathematical approach(es) to problem-solving so that good technique and sound reasoning may be reinforced. Your instructor may also pose additional questions based on those assigned, in order to observe your problem-solving firsthand.

Mental arithmetic and estimations are strongly encouraged for all calculations, because without these abilities you will be unable to readily detect errors caused by calculator misuse (e.g. keystroke errors).

You will note a conspicuous lack of answers given for these quantitative questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. My advice is to use circuit simulation software such as SPICE to check the correctness of quantitative answers. Refer to those learning modules within this collection focusing on SPICE to see worked examples which you may use directly as practice problems for your own study, and/or as templates you may modify to run your own analyses and generate your own practice problems.

Completely worked example problems found in the Tutorial may also serve as "test cases[4]" for gaining proficiency in the use of circuit simulation software, and then once that proficiency is gained you will never need to rely[5] on an answer key!

[4] In other words, set up the circuit simulation software to analyze the same circuit examples found in the Tutorial. If the simulated results match the answers shown in the Tutorial, it confirms the simulation has properly run. If the simulated results disagree with the Tutorial's answers, something has been set up incorrectly in the simulation software. Using every Tutorial as practice in this way will quickly develop proficiency in the use of circuit simulation software.

[5] This approach is perfectly in keeping with the instructional philosophy of these learning modules: *teaching students to be self-sufficient thinkers*. Answer keys can be useful, but it is even more useful to your long-term success to have a set of tools on hand for checking your own work, because once you have left school and are on your own, there will no longer be "answer keys" available for the problems you will have to solve.

5.2.1 Miscellaneous physical constants

Note: constants shown in **bold** type are *exact*, not approximations. Values inside of parentheses show one standard deviation (σ) of uncertainty in the final digits: for example, the magnetic permeability of free space value given as $1.25663706212(19) \times 10^{-6}$ H/m represents a center value (i.e. the location parameter) of $1.25663706212 \times 10^{-6}$ Henrys per meter with one standard deviation of uncertainty equal to $0.0000000000019 \times 10^{-6}$ Henrys per meter.

Avogadro's number (N_A) = **6.02214076** $\times 10^{23}$ **per mole** (mol^{-1})

Boltzmann's constant (k) = **1.380649** $\times 10^{-23}$ **Joules per Kelvin** (J/K)

Electronic charge (e) = **1.602176634** $\times 10^{-19}$ **Coulomb** (C)

Faraday constant (F) = **96,485.33212...** $\times 10^{4}$ **Coulombs per mole** (C/mol)

Magnetic permeability of free space (μ_0) = 1.25663706212(19) $\times 10^{-6}$ Henrys per meter (H/m)

Electric permittivity of free space (ϵ_0) = 8.8541878128(13) $\times 10^{-12}$ Farads per meter (F/m)

Characteristic impedance of free space (Z_0) = 376.730313668(57) Ohms (Ω)

Gravitational constant (G) = 6.67430(15) $\times 10^{-11}$ cubic meters per kilogram-seconds squared ($m^3/kg\text{-}s^2$)

Molar gas constant (R) = **8.314462618... Joules per mole-Kelvin** (J/mol-K) = 0.08205746(14) liters-atmospheres per mole-Kelvin

Planck constant (h) = **6.62607015** $\times 10^{-34}$ **joule-seconds** (J-s)

Stefan-Boltzmann constant (σ) = **5.670374419...** $\times 10^{-8}$ **Watts per square meter-Kelvin**^{4} ($W/m^2 \cdot K^4$)

Speed of light in a vacuum (c) = **299,792,458 meters per second** (m/s) = 186282.4 miles per second (mi/s)

Note: All constants taken from NIST data "Fundamental Physical Constants – Complete Listing", from `http://physics.nist.gov/constants`, National Institute of Standards and Technology (NIST), 2018 CODATA Adjustment.

5.2.2 Introduction to spreadsheets

A powerful computational tool you are encouraged to use in your work is a *spreadsheet*. Available on most personal computers (e.g. Microsoft Excel), *spreadsheet* software performs numerical calculations based on number values and formulae entered into cells of a grid. This grid is typically arranged as lettered columns and numbered rows, with each cell of the grid identified by its column/row coordinates (e.g. cell `B3`, cell `A8`). Each cell may contain a string of text, a number value, or a mathematical formula. The spreadsheet automatically updates the results of all mathematical formulae whenever the entered number values are changed. This means it is possible to set up a spreadsheet to perform a series of calculations on entered data, and those calculations will be re-done by the computer any time the data points are edited in any way.

For example, the following spreadsheet calculates average speed based on entered values of distance traveled and time elapsed:

	A	B	C	D
1	Distance traveled	46.9	Kilometers	
2	Time elapsed	1.18	Hours	
3	Average speed	= B1 / B2	km/h	
4				
5				

Text labels contained in cells `A1` through `A3` and cells `C1` through `C3` exist solely for readability and are not involved in any calculations. Cell `B1` contains a sample distance value while cell `B2` contains a sample time value. The formula for computing speed is contained in cell `B3`. Note how this formula begins with an "equals" symbol (`=`), references the values for distance and speed by lettered column and numbered row coordinates (`B1` and `B2`), and uses a forward slash symbol for division (`/`). The coordinates `B1` and `B2` function as *variables*[6] would in an algebraic formula.

When this spreadsheet is executed, the numerical value `39.74576` will appear in cell `B3` rather than the formula `= B1 / B2`, because 39.74576 is the computed speed value given 46.9 kilometers traveled over a period of 1.18 hours. If a different numerical value for distance is entered into cell `B1` or a different value for time is entered into cell `B2`, cell `B3`'s value will automatically update. All you need to do is set up the given values and any formulae into the spreadsheet, and the computer will do all the calculations for you.

Cell `B3` may be referenced by other formulae in the spreadsheet if desired, since it is a variable just like the given values contained in `B1` and `B2`. This means it is possible to set up an entire chain of calculations, one dependent on the result of another, in order to arrive at a final value. The arrangement of the given data and formulae need not follow any pattern on the grid, which means you may place them anywhere.

[6] Spreadsheets may also provide means to attach text labels to cells for use as variable names (Microsoft Excel simply calls these labels "names"), but for simple spreadsheets such as those shown here it's usually easier just to use the standard coordinate naming for each cell.

Common[7] arithmetic operations available for your use in a spreadsheet include the following:

- Addition (`+`)
- Subtraction (`-`)
- Multiplication (`*`)
- Division (`/`)
- Powers (`^`)
- Square roots (`sqrt()`)
- Logarithms (`ln()` , `log10()`)

Parentheses may be used to ensure[8] proper order of operations within a complex formula. Consider this example of a spreadsheet implementing the *quadratic formula*, used to solve for roots of a polynomial expression in the form of $ax^2 + bx + c$:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

	A	B
1	x_1	= (-B4 + sqrt((B4^2) - (4*B3*B5))) / (2*B3)
2	x_2	= (-B4 - sqrt((B4^2) - (4*B3*B5))) / (2*B3)
3	a =	9
4	b =	5
5	c =	-2

This example is configured to compute roots[9] of the polynomial $9x^2 + 5x - 2$ because the values of 9, 5, and -2 have been inserted into cells `B3`, `B4`, and `B5`, respectively. Once this spreadsheet has been built, though, it may be used to calculate the roots of *any* second-degree polynomial expression simply by entering the new a, b, and c coefficients into cells `B3` through `B5`. The numerical values appearing in cells `B1` and `B2` will be automatically updated by the computer immediately following any changes made to the coefficients.

[7] Modern spreadsheet software offers a bewildering array of mathematical functions you may use in your computations. I recommend you consult the documentation for your particular spreadsheet for information on operations other than those listed here.

[8] Spreadsheet programs, like text-based programming languages, are designed to follow standard order of operations by default. However, my personal preference is to use parentheses even where strictly unnecessary just to make it clear to any other person viewing the formula what the intended order of operations is.

[9] Reviewing some algebra here, a *root* is a value for x that yields an overall value of zero for the polynomial. For this polynomial ($9x^2 + 5x - 2$) the two roots happen to be $x = 0.269381$ and $x = -0.82494$, with these values displayed in cells `B1` and `B2`, respectively upon execution of the spreadsheet.

Alternatively, one could break up the long quadratic formula into smaller pieces like this:

$$y = \sqrt{b^2 - 4ac} \qquad z = 2a$$

$$x = \frac{-b \pm y}{z}$$

	A	B	C
1	x_1	= (-B4 + C1) / C2	= sqrt((B4^2) - (4*B3*B5))
2	x_2	= (-B4 - C1) / C2	= 2*B3
3	a =	9	
4	b =	5	
5	c =	-2	

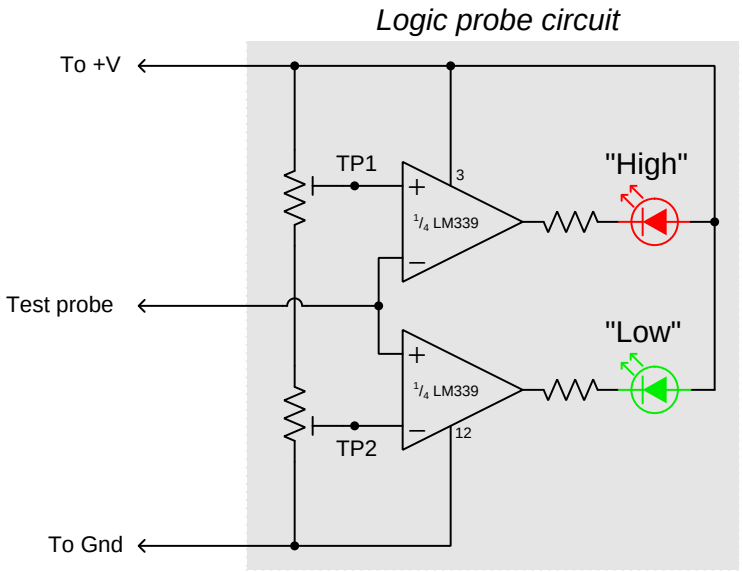
Note how the square-root term (y) is calculated in cell C1, and the denominator term (z) in cell C2. This makes the two final formulae (in cells B1 and B2) simpler to interpret. The positioning of all these cells on the grid is completely arbitrary[10] – all that matters is that they properly reference each other in the formulae.

Spreadsheets are particularly useful for situations where the same set of calculations representing a circuit or other system must be repeated for different initial conditions. The power of a spreadsheet is that it automates what would otherwise be a tedious set of calculations. One specific application of this is to simulate the effects of various components within a circuit failing with abnormal values (e.g. a shorted resistor simulated by making its value nearly zero; an open resistor simulated by making its value extremely large). Another application is analyzing the behavior of a circuit design given new components that are out of specification, and/or aging components experiencing drift over time.

[10] My personal preference is to locate all the "given" data in the upper-left cells of the spreadsheet grid (each data point flanked by a sensible name in the cell to the left and units of measurement in the cell to the right as illustrated in the first distance/time spreadsheet example), sometimes coloring them in order to clearly distinguish which cells contain entered data versus which cells contain computed results from formulae. I like to place all formulae in cells below the given data, and try to arrange them in logical order so that anyone examining my spreadsheet will be able to figure out *how* I constructed a solution. This is a general principle I believe all computer programmers should follow: *document and arrange your code to make it easy for other people to learn from it.*

5.2.3 Simple logic probe circuit

A *logic probe* is a very useful tool for working with digital logic circuits. It indicates "high" and "low" logic states by means of LEDs, giving visual indication only if the voltage levels are appropriate for each state. The following schematic diagram shows a logic probe circuit built using *comparators* which are integrated circuit elements designed to compare two analog voltage signals (each with reference to ground) and drive its output pin either "high" or "low" depending on which of the sensed input voltages is larger:



Identify the purpose of the two potentiometers in this circuit.

When this logic probe circuit is connected to the V_{CC} and V_{EE} power supply terminals of a powered TTL circuit, what voltage levels should test points TP1 and TP2 be adjusted to, in order for the probe to properly indicate "high" and "low" TTL logic states?

When this logic probe circuit is connected to the V_{DD} and V_{SS} power supply terminals of a powered CMOS circuit operating on a 5 Volt DC power supply, what voltage levels should test points TP1 and TP2 be adjusted to, in order for the probe to properly indicate "high" and "low" CMOS logic states?

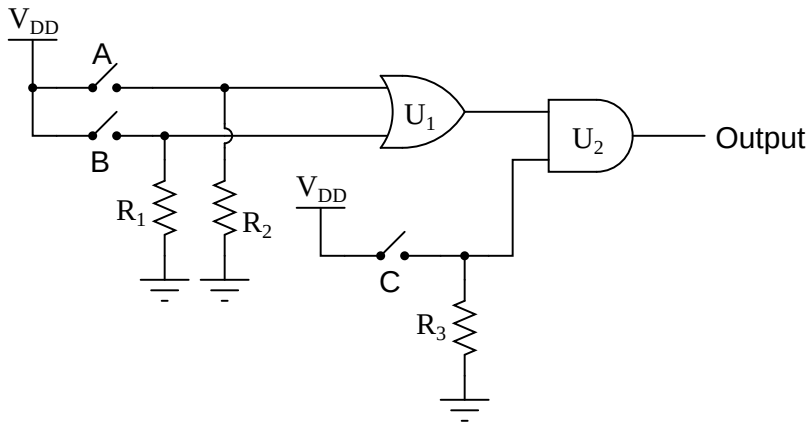
Challenges

- Write a formula for calculating appropriate current-limiting resistor sizes for the two LEDs in this circuit, given the value of $+V$ and the LED forward voltage and current values.
- The logic probe circuit shown is minimal in component count. To make a more practical and reliable probe, one would probably want to have reverse-polarity protection (in case someone

were to accidently connect the probe backward across the power supply) as well as decoupling for immunity against electrical noise. Add whatever necessary components you think there should be in this circuit to provide these features.

5.2.4 Pulldown resistor sizing

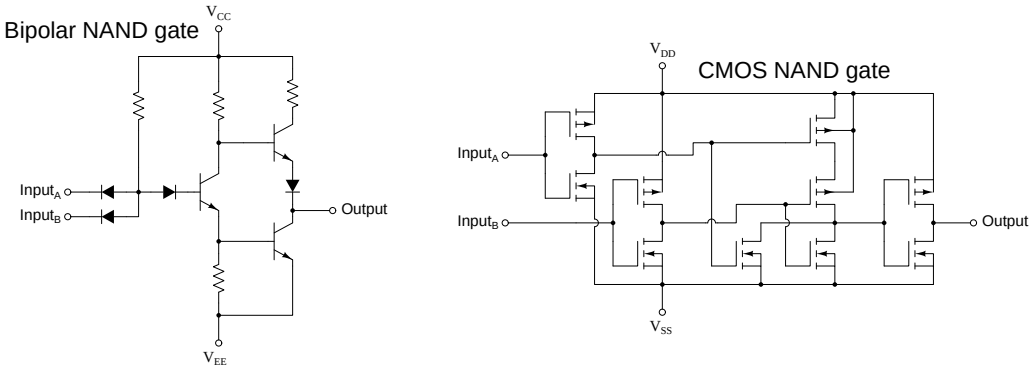
In the following schematic diagram, the pulldown resistor sizes are not shown:



Supposing these logic gates are both CMOS in design, what would be acceptable resistor values to use?

Now supposing these logic gates are both TTL in design, what would be acceptable resistor values to use?

Below is a comparison of typical TTL and MOSFET logic gate internal circuitry. The fact that these show NAND gates is irrelevant to the question of pulldown resistor sizing, as we only need to know what typical *input* circuitry looks like for TTL and for CMOS:



You may find CMOS and TTL logic gate datasheets helpful as sources of internal gate properties!

Challenges

- Identify a good diagnostic instrument to tell whether an installed pulldown resistor is properly sized.

5.3 Diagnostic reasoning

These questions are designed to stimulate your deductive and inductive thinking, where you must apply general principles to specific scenarios (deductive) and also derive conclusions about the failed circuit from specific details (inductive). In a Socratic discussion with your instructor, the goal is for these questions to reinforce your recall and use of general circuit principles and also challenge your ability to integrate multiple symptoms into a sensible explanation of what's wrong in a circuit. Your instructor may also pose additional questions based on those assigned, in order to further challenge and sharpen your diagnostic abilities.

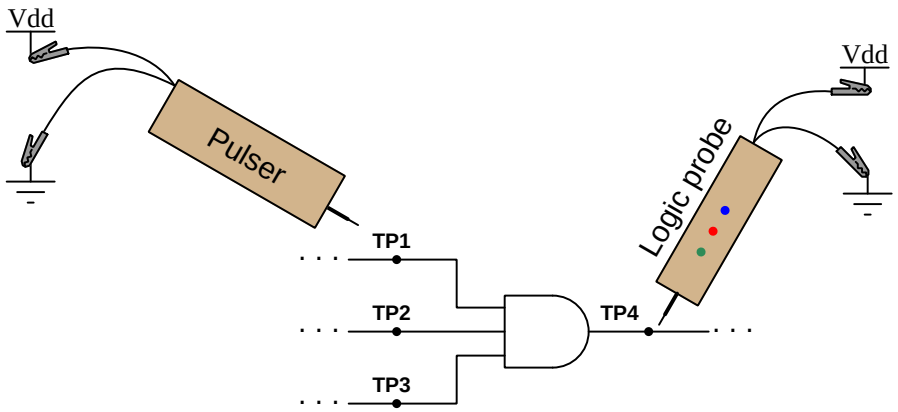
As always, your goal is to fully *explain* your analysis of each problem. Simply obtaining a correct answer is not good enough – you must also demonstrate sound reasoning in order to successfully complete the assignment. Your instructor's responsibility is to probe and challenge your understanding of the relevant principles and analytical processes in order to ensure you have a strong foundation upon which to build further understanding.

You will note a conspicuous lack of answers given for these diagnostic questions. Unlike standard textbooks where answers to every other question are given somewhere toward the back of the book, here in these learning modules students must rely on other means to check their work. The best way by far is to debate the answers with fellow students and also with the instructor during the Socratic dialogue sessions intended to be used with these learning modules. Reasoning through challenging questions with other people is an excellent tool for developing strong reasoning skills.

Another means of checking your diagnostic answers, where applicable, is to use circuit simulation software to explore the effects of faults placed in circuits. For example, if one of these diagnostic questions requires that you predict the effect of an open or a short in a circuit, you may check the validity of your work by simulating that same fault (substituting a very high resistance in place of that component for an open, and substituting a very low resistance for a short) within software and seeing if the results agree.

5.3.1 Logic probe and pulser

A technician decides to check a suspect three-input AND gate using a logic pulser. This particular AND gate is installed in a larger circuit, the rest of which not shown here. She touches the logic pulser to each input of the AND gate, while looking for a pulsing signal at the output with a logic probe:



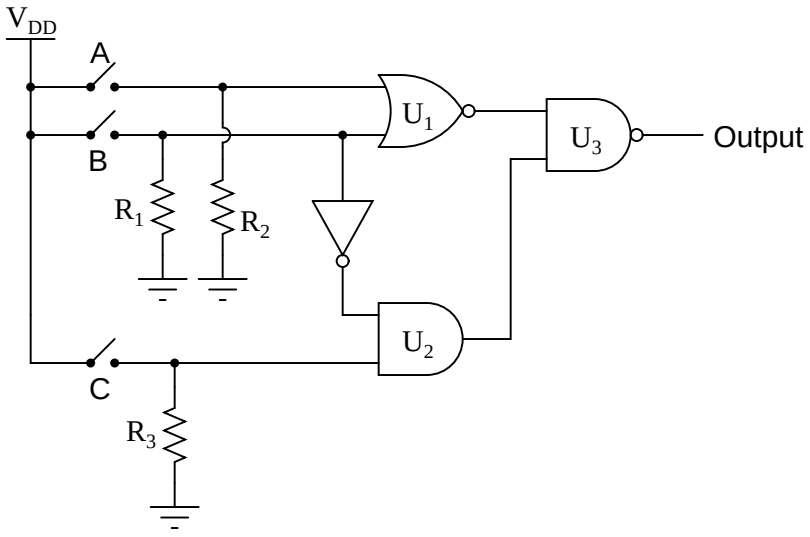
No matter which input test point (TP1, TP2, or TP3) she pulses, though, the output test point (TP4) always reads low. Does this prove the AND gate to be defective? Explain why or why not.

Challenges

- Suppose this were an OR gate rather than an AND gate. Would the same test result be conclusive or inconclusive?

5.3.2 Identifying possible faults

The output of the following gate circuit is always high, no matter what states the input switches are in. Assume that CMOS logic gates are being used here:



Identify which of these possibilities could account for the output always being high:

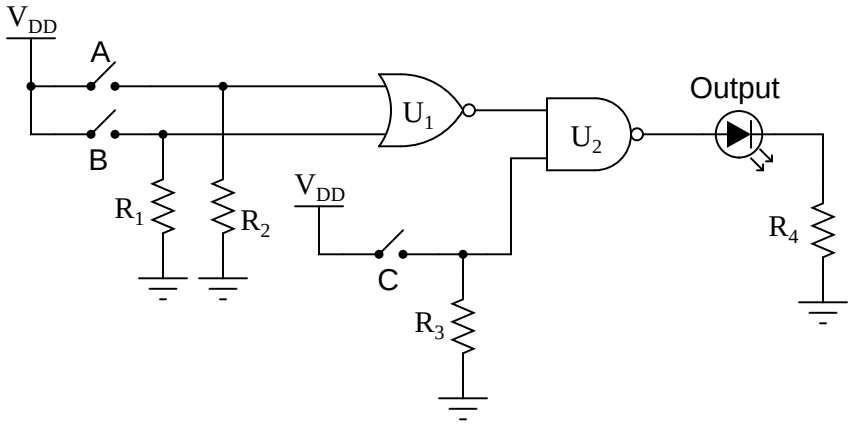
- Output of U_1 stuck in a high state
- Output of U_2 stuck in a high state
- R_1 failed open
- R_2 failed shorted
- R_3 failed shorted
- Switch A failed open
- Switch B failed shorted
- Switch C failed shorted

Challenges

- Identify other possible faults not shown on this list.

5.3.3 Input switch settings

The following gate circuit has a problem:



When tested, it is found that the circuit does not respond in the same manner as its (ideal) truth table predicts. Here is a comparison of the ideal and actual truth tables, as predicted and tested:

A	**B**	**C**	**Output** (ideal)	**Output** (actual)
0	0	0	1	1
0	0	1	0	0
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

The first thing a good electronics technician would do, of course, is set up either a voltmeter or a logic probe and begin testing for logic levels in the circuit to see what is wrong. However, the settings of the input switches are very important as part of the diagnosis. Based on the design of the circuit, and the truth table results shown, in what states (open or closed) would you first set the input switches, and then what logic level would you first test with the logic probe or voltmeter?

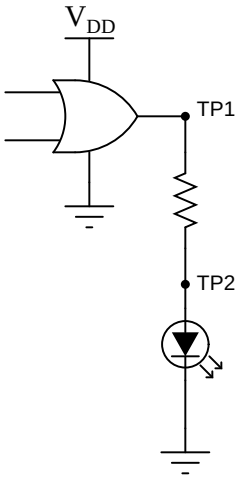
Based on the truth table data, what do you suspect is the fault?

Challenges

- Suppose you were asked to troubleshoot a simple electric lamp circuit using only a voltmeter. The problem is, the lamp does not energize when the switch is closed. Would it be best to take voltage measurements with the switch on or off?

5.3.4 Logic probe limitations

Logic probes are useful tools for troubleshooting digital logic gate circuits, but they certainly have limitations. For instance, in this simple circuit, a logic probe will give correct "high" and "low" readings at test point 1 (TP1), but it will always read "low" (even when the LED is on) at test point 2 (TP2):



Now, obviously the output of the gate is "high" when the LED is on, otherwise it would not illuminate. Why then does a logic probe fail to indicate a high logic state at TP2?

Challenges

- ???.
- ???.
- ???.

Appendix A

Problem-Solving Strategies

The ability to solve complex problems is arguably one of the most valuable skills one can possess, and this skill is particularly important in any science-based discipline.

- Study principles, not procedures. Don't be satisfied with merely knowing how to compute solutions – learn *why* those solutions work.
- Identify what it is you need to solve, identify all relevant data, identify all units of measurement, identify any general principles or formulae linking the given information to the solution, and then identify any "missing pieces" to a solution. Annotate all diagrams with this data.
- Sketch a diagram to help visualize the problem. When building a real system, always devise a plan for that system and analyze its function *before* constructing it.
- Follow the units of measurement and meaning of every calculation. If you are ever performing mathematical calculations as part of a problem-solving procedure, and you find yourself unable to apply each and every intermediate result to some aspect of the problem, it means you don't understand what you are doing. Properly done, every mathematical result should have practical meaning for the problem, and not just be an abstract number. You should be able to identify the proper units of measurement for each and every calculated result, and show where that result fits into the problem.
- Perform "thought experiments" to explore the effects of different conditions for theoretical problems. When troubleshooting real systems, perform *diagnostic tests* rather than visually inspecting for faults, the best diagnostic test being the one giving you the most information about the nature and/or location of the fault with the fewest steps.
- Simplify the problem until the solution becomes obvious, and then use that obvious case as a model to follow in solving the more complex version of the problem.
- Check for exceptions to see if your solution is incorrect or incomplete. A good solution will work for *all* known conditions and criteria. A good example of this is the process of testing scientific hypotheses: the task of a scientist is not to find support for a new idea, but rather to *challenge* that new idea to see if it holds up under a battery of tests. The philosophical

principle of *reductio ad absurdum* (i.e. disproving a general idea by finding a specific case where it fails) is useful here.

- Work "backward" from a hypothetical solution to a new set of given conditions.
- Add quantities to problems that are qualitative in nature, because sometimes a little math helps illuminate the scenario.
- Sketch graphs illustrating how variables relate to each other. These may be quantitative (i.e. with realistic number values) or qualitative (i.e. simply showing increases and decreases).
- Treat quantitative problems as qualitative in order to discern the relative magnitudes and/or directions of change of the relevant variables. For example, try determining what happens if a certain variable were to increase or decrease before attempting to precisely calculate quantities: how will each of the dependent variables respond, by increasing, decreasing, or remaining the same as before?
- Consider limiting cases. This works especially well for qualitative problems where you need to determine which direction a variable will change. Take the given condition and magnify that condition to an extreme degree as a way of simplifying the direction of the system's response.
- Check your work. This means regularly testing your conclusions to see if they make sense. This does *not* mean repeating the same steps originally used to obtain the conclusion(s), but rather to use some other means to check validity. Simply repeating procedures often leads to *repeating the same errors* if any were made, which is why alternative paths are better.

Appendix B

Instructional philosophy

B.1 First principles of learning

- **Anyone can learn anything** given appropriate time, effort, resources, challenges, encouragement, and expectations. Dedicating time and investing effort are the student's responsibility; providing resources, challenges, and encouragement are the teacher's responsibility; high expectations are a responsibility shared by both student and teacher.

- **Transfer is not automatic.** The human mind has a natural tendency to compartmentalize information, which means the process of taking knowledge learned in one context and applying it to another usually does not come easy and therefore should never be taken for granted.

- **Learning is iterative.** The human mind rarely learns anything perfectly on the first attempt. Anticipate mistakes and plan for multiple tries to achieve full understanding, using the lessons of those mistakes as feedback to guide future attempts.

- **Information is absorbed, but understanding is created.** Facts and procedures may be memorized easily enough by repeated exposure, but the ability to reliably apply principles to novel scenarios only comes through intense personal effort. This effort is fundamentally creative in nature: casting new concepts in one's own words, running mental experiments to perceive patterns and trends, etc. These acts of making knowledge "one's own" need not be perfect in order to be effective, as the value lies in the activity and not necessarily the finished product. Depth of understanding and permanence of memory are both directly related to mental effort during the learning process – i.e. *easy learning is shallow and fleeting.*

- **Education trumps training.** There is no such thing as an entirely isolated subject, as all fields of knowledge are connected. Training is narrowly-focused and task-oriented. Education is broad-based and principle-oriented. When preparing for a life-long technical career, education beats training every time.

- **Character matters.** Poor habits are more destructive than deficits of knowledge or skill. This is especially true in collective endeavors, where a team's ability to function depends on trust between its members. Simply put, no one wants an untrustworthy person on their team. An essential component of education then, is character development.

- **People learn to be responsible by bearing responsibility.** An irresponsible person is someone who has never *had* to be responsible for anything that mattered enough to them. Just as anyone can learn anything, anyone can become responsible if the personal cost of irresponsibility becomes high enough.

- **What gets measured, gets done.** Accurate and relevant assessment of learning is key to ensuring all students learn. Therefore, it is imperative to measure what matters.

- **Failure is nothing to fear.** Every human being fails, and fails in multiple ways at multiple times. Eventual success only happens when we don't stop trying.

B.2 Proven strategies for instructors

- Assume every student is capable of learning anything they desire given the proper conditions. Treat them as capable adults by granting real responsibility and avoiding artificial incentives such as merit or demerit points.
- Create a consistent culture of high expectations across the entire program of study. Demonstrate and encourage patience, persistence, and a healthy sense of self-skepticism. Anticipate and de-stigmatize error. Teach respect for the capabilities of others as well as respect for one's own fallibility.
- Replace lecture with "inverted" instruction, where students first encounter new concepts through reading and then spend class time in Socratic dialogue with the instructor exploring those concepts and solving problems individually. There is a world of difference between observing someone solve a problem versus actually solving a problem yourself, and so the point of this form of instruction is to place students in a position where they *cannot* passively observe.
- Require students to read extensively, write about what they learn, and dialogue with you and their peers to sharpen their understanding. Apply Francis Bacon's advice that "reading maketh a full man; conference a ready man; and writing an exact man". These are complementary activities helping students expand their confidence and abilities.
- Use artificial intelligence (AI) to challenge student understanding rather than merely provide information. Find productive ways for AI to critique students' clarity of thought and of expression, for example by employing AI as a Socratic-style interlocutor or as a reviewer of students' journals. Properly applied, AI has the ability to expand student access to critical review well outside the bounds of their instructor's reach.
- Build frequent and rapid feedback into the learning process so that students know at all times how well they are learning, to identify problems early and fix them before they grow. Model the intellectual habit of self-assessing and self-correcting your own understanding (i.e. a cognitive *feedback loop*), encouraging students to do the same.
- Use "mastery" as the standard for every assessment, which means the exam or experiment or project must be done with 100% competence in order to pass. Provide students with multiple opportunity for re-tries (different versions of the assessment every time).
- Require students to devise their own hypotheses and procedures on all experiments, so that the process is truly a scientific one. Have students assess their proposed experimental procedures for risk and devise mitigations for those risks. Let nothing be pre-designed about students' experiments other than a stated task (i.e. what principle the experiment shall test) at the start and a set of demonstrable knowledge and skill objectives at the end.
- Have students build as much of their lab equipment as possible: building power sources, building test assemblies[1], and building complete working systems (no kits!). In order to provide

[1] In the program I teach, every student builds their own "Development Board" consisting of a metal chassis with DIN rail, terminal blocks, and an AC-DC power supply of their own making which functions as a portable lab environment they can use at school as well as take home.

this same "ground-up" experience for every new student, this means either previous students take their creations with them, or the systems get disassembled in preparation for the new students, or the systems grow and evolve with each new student group.

- Incorporate external accountability for you and for your students, continuously improving the curriculum and your instructional methods based on proven results. Have students regularly network with active professionals through participation in advisory committee meetings, service projects, tours, jobshadows, internships, etc. Practical suggestions include requiring students to design and build projects for external clients (e.g. community groups, businesses, different departments within the institution), and also requiring students attend all technical advisory committee meetings and to dialogue with the industry representatives at those meetings.
- Repeatedly explore difficult-to-learn concepts across multiple courses, so that students have multiple opportunities to build their understanding.
- Relate all new concepts, whenever possible, to previous concepts and to relevant physical laws. Challenge each and every student, every day, to *reason* from concept to concept and to explain the logical connections between. Challenge students to verify their conclusions by multiple approaches (e.g. double-checking their work using different methods). Ask *"Why?"* often.
- Maintain detailed records on each student's performance and share these records privately with them. These records should include academic performance as well as professionally relevant behavioral tendencies.
- Hold mandatory "check-in" meetings between all program faculty and each new student during their first term. Offer these to all other students as an option, except for any students continuing to manifest unprofessional behaviors, poor academic performance, or who have some other need for a face-to-face meeting with faculty.
- Address problems while they are small, before they grow larger. This is equally true for tutoring technical concepts as it is for helping students build professional habits.
- Build rigorous quality control into the curriculum to ensure every student masters every important concept, and that the mastery is retained over time. This includes (1) review questions added to every exam to re-assess knowledge taught in previous terms, (2) cumulative exams at the end of every term to re-assess all important concepts back to the very beginning of the program, and (3) review assessments in practical (hands-on) coursework to ensure critically-important skills were indeed taught and are still retained. What you will find by doing this is that it actually boosts retention of students by ensuring that important knowledge gets taught and is retained over long spans of time. In the absence of such quality control, student learning and retention tends to be spotty and this contributes to drop-out and failure rates later in their education.
- Finally, *never rush learning.* Education is not a race. Give your students ample time to digest complex ideas, as you continually remind yourself of just how long it took you to achieve mastery! Long-term retention and the consistently correct application of concepts are always the result of *focused effort over long periods of time* which means there are no shortcuts to learning.

B.3 Proven strategies for students

The single most important piece of advice I have for any student of any subject is to take responsibility for your own development in all areas of life including mental development. Expecting others in your life to entirely guide your own development is a recipe for disappointment. This is just as true for students enrolled in formal learning institutions as it is for auto-didacts pursuing learning entirely on their own. Learning to think in new ways is key to being able to gainfully use information, to make informed decisions about your life, and to best serve those you care about. With this in mind, I offer the following advice to students:

- **Approach all learning as valuable**. No matter what course you take, no matter who you learn from, no matter the subject, there is something useful in every learning experience. If you don't see the value of every new experience, you are not looking closely enough!
- **Continually challenge yourself**. Let other people take shortcuts and find easy answers to easy problems. The purpose of education is to stretch your mind, in order to shape it into a more powerful tool. This doesn't come by taking the path of least resistance. An excellent analogy for an empowering education is productive physical exercise: becoming stronger, more flexible, and more persistent only comes through intense personal effort.
- **Master the use of language**. This includes reading extensively, writing every day, listening closely, and speaking articulately. To a great extent language channels and empowers thought, so the better you are at wielding language the better you will be at grasping abstract concepts and articulating them not only for your benefit but for others as well.
- **Do not limit yourself to the resources given to you**. Read books that are not on the reading list. Run experiments that aren't assigned to you. Form study groups outside of class. Take an entrepreneurial approach to your own education, as though it were a business you were building for your future benefit.
- **Express and share what you learn**. Take every opportunity to teach what you have learned to others, as this will not only help them but will also strengthen your own understanding[2].
- Realize that **no one can give you understanding**, just as no one can give you physical fitness. These both must be *built*.
- **Above all, recognize that learning is hard work, and that a certain level of frustration is unavoidable**. There are times when you will struggle to grasp some of these concepts, and that struggle is a natural thing. Take heart that it will yield with persistent and varied[3] effort, and never give up! That concepts don't immediately come to you is not a sign of something wrong, but rather of something right: that you have found a worthy challenge!

[2] On a personal note, I was surprised to learn just how much my own understanding of electronics and related subjects was strengthened by becoming a teacher. When you are tasked every day with helping other people grasp complex topics, it catalyzes your own learning by giving you powerful incentives to study, to articulate your thoughts, and to reflect deeply on the process of learning.

[3] As the old saying goes, "Insanity is trying the same thing over and over again, expecting different results." If you find yourself stumped by something in the text, you should attempt a different approach. Alter the thought experiment, change the mathematical parameters, do whatever you can to see the problem in a slightly different light, and then the solution will often present itself more readily.

B.4 Design of these learning modules

"The unexamined circuit is not worth energizing" – Socrates (if he had taught electricity)

These learning modules, although useful for self-study, were designed to be used in a formal learning environment where a subject-matter expert challenges students to digest the content and exercise their critical thinking abilities in the answering of questions and in the construction and testing of working circuits. Every effort has been made to embed the following instructional and assessment philosophies within:

- The first goal of education is to enhance clear and independent thought, in order that every student reach their fullest potential in a highly complex and inter-dependent world. Robust reasoning is *always* more important than particulars of any subject matter, because its application is universal.
- Literacy is fundamental to independent learning and thought because text continues to be the most efficient way to communicate complex ideas over space and time. Those who cannot read with ease are limited in their ability to acquire knowledge and perspective.
- Articulate communication is fundamental to work that is complex and interdisciplinary.
- Faulty assumptions and poor reasoning are best corrected through challenge, not presentation. The rhetorical technique of *reductio ad absurdum* (disproving an assertion by exposing an absurdity) works well to discipline student's minds, not only to correct the problem at hand but also to learn how to detect and correct future errors.
- Important principles should be repeatedly explored and widely applied throughout a course of study, not only to reinforce their importance and help ensure their mastery, but also to showcase the interconnectedness and utility of knowledge.

These learning modules were expressly designed to be used in an "inverted" teaching environment[4] where students first read the introductory and tutorial chapters on their own, then individually attempt to answer the questions and construct working circuits according to the experiment and project guidelines. The instructor never lectures, but instead meets regularly with each individual student to review their progress, answer questions, identify misconceptions, and challenge the student to new depths of understanding through further questioning. Regular meetings between instructor and student should resemble a Socratic[5] dialogue, where questions serve as scalpels to dissect topics and expose assumptions. The student passes each module only after consistently demonstrating their ability to logically analyze and correctly apply all major concepts in each question or project/experiment. The instructor must be vigilant in probing each student's understanding to ensure they are truly *reasoning* and not just *memorizing*. This is why "Challenge" points appear throughout, as prompts for students to think deeper about topics and as starting points for instructor queries. Sometimes these challenge points require additional knowledge that hasn't been covered in the series to answer in full. This is okay, as the major purpose of the Challenges is to stimulate analysis and synthesis on the part of each student.

The instructor must possess enough mastery of the subject matter and awareness of students' reasoning to generate their own follow-up questions to practically any student response. Even completely correct answers given by the student should be challenged by the instructor for the purpose of having students practice articulating their thoughts and defending their reasoning. Conceptual errors committed by the student should be exposed and corrected not by direct instruction, but rather by reducing the errors to an absurdity[6] through well-chosen questions and thought experiments posed by the instructor. Becoming proficient at this style of instruction requires time and dedication, but the positive effects on critical thinking for both student and instructor are spectacular.

An inspection of these learning modules reveals certain unique characteristics. One of these is a bias toward thorough explanations in the tutorial chapters. Without a live instructor to explain concepts and applications to students, the text itself must fulfill this role. This philosophy results in lengthier explanations than what you might typically find in a textbook, each step of the reasoning process fully explained, including footnotes addressing common questions and concerns students raise while learning these concepts. Each tutorial seeks to not only explain each major concept in sufficient detail, but also to explain the logic of each concept and how each may be developed

[4] In a traditional teaching environment, students first encounter new information via *lecture* from an expert, and then independently apply that information via *homework*. In an "inverted" course of study, students first encounter new information via *homework*, and then independently apply that information under the scrutiny of an expert. The expert's role in lecture is to simply *explain*, but the expert's role in an inverted session is to *challenge*, *critique*, and if necessary *explain* where gaps in understanding still exist.

[5] Socrates is a figure in ancient Greek philosophy famous for his unflinching style of questioning. Although he authored no texts, he appears as a character in Plato's many writings. The essence of Socratic philosophy is to leave no question unexamined and no point of view unchallenged. While purists may argue a topic such as electric circuits is too narrow for a true Socratic-style dialogue, I would argue that the essential thought processes involved with scientific reasoning on *any* topic are not far removed from the Socratic ideal, and that students of electricity and electronics would do very well to challenge assumptions, pose thought experiments, identify fallacies, and otherwise employ the arsenal of critical thinking skills modeled by Socrates.

[6] This rhetorical technique is known by the Latin phrase *reductio ad absurdum*. The concept is to expose errors by counter-example, since only one solid counter-example is necessary to disprove a universal claim. As an example of this, consider the common misconception among beginning students of electricity that voltage cannot exist without current. One way to apply *reductio ad absurdum* to this statement is to ask how much current passes through a fully-charged battery connected to nothing (i.e. a clear example of voltage existing without current).

from "first principles". Again, this reflects the goal of developing clear and independent thought in students' minds, by showing how clear and logical thought was used to forge each concept. Students benefit from witnessing a model of clear thinking in action, and these tutorials strive to be just that.

Another feature of these learning modules is that they do not treat topics in isolation. Rather, important concepts are introduced early in the series, and appear repeatedly as stepping-stones toward other concepts in subsequent modules. This helps to avoid the "compartmentalization" of knowledge, demonstrating the inter-connectedness of concepts and simultaneously reinforcing them. Each module is fairly complete in itself, reserving the beginning of its tutorial to a review of foundational concepts.

To high standards of education,

Tony R. Kuphaldt

Appendix C

Tools used

I am indebted to the developers of many open-source software applications in the creation of these learning modules. The following is a list of these applications with some commentary on each.

You will notice a theme common to many of these applications: a bias toward *code*. Although I am by no means an expert programmer in any computer language, I understand and appreciate the flexibility offered by code-based applications where the user (you) enters commands into a plain ASCII text file, which the software then reads and processes to create the final output. Code-based computer applications are by their very nature *extensible*, while WYSIWYG (What You See Is What You Get) applications are generally limited to whatever user interface the developer makes for you.

The `GNU/Linux` computer operating system

> There is so much to be said about Linus Torvalds' `Linux` and Richard Stallman's `GNU` project. First, to credit just these two individuals is to fail to do justice to the *mob* of passionate volunteers who contributed to make this amazing software a reality. I first learned of `Linux` back in 1996, and have been using this operating system on my personal computers almost exclusively since then. It is *free*, it is completely *configurable*, and it permits the continued use of highly efficient `Unix` applications and scripting languages (e.g. shell scripts, Makefiles, `sed`, `awk`) developed over many decades. `Linux` not only provided me with a powerful computing platform, but its open design served to inspire my life's work of creating open-source educational resources.

Bram Moolenaar's `Vim` text editor

> Writing code for any code-based computer application requires a *text editor*, which may be thought of as a word processor strictly limited to outputting plain-ASCII text files. Many good text editors exist, and one's choice of text editor seems to be a deeply personal matter within the programming world. I prefer `Vim` because it operates very similarly to `vi` which is ubiquitous on `Unix`/`Linux` operating systems, and because it may be entirely operated via keyboard (i.e. no mouse required) which makes it fast to use.

Donald Knuth's TEX typesetting system

Developed in the late 1970's and early 1980's by computer scientist extraordinaire Donald Knuth to typeset his multi-volume magnum opus *The Art of Computer Programming*, this software allows the production of formatted text for screen-viewing or paper printing, all by writing plain-text code to describe how the formatted text is supposed to appear. TEX is not just a markup language for documents, but it is also a Turing-complete programming language in and of itself, allowing useful algorithms to be created to control the production of documents. Simply put, TEX *is a programmer's approach to word processing*. Since TEX is controlled by code written in a plain-text file, this means anyone may read that plain-text file to see exactly how the document was created. This openness afforded by the code-based nature of TEX makes it relatively easy to learn how other people have created their own TEX documents. By contrast, examining a beautiful document created in a conventional WYSIWYG word processor such as Microsoft `Word` suggests nothing to the reader about *how* that document was created, or what the user might do to create something similar. As Mr. Knuth himself once quipped, conventional word processing applications should be called WYSIAYG (What You See Is *All* You Get).

Leslie Lamport's LATEX extensions to TEX

Like all true programming languages, TEX is inherently extensible. So, years after the release of TEX to the public, Leslie Lamport decided to create a massive extension allowing easier compilation of book-length documents. The result was LATEX, which is the markup language used to create all ModEL module documents. You could say that TEX is to LATEX as `C` is to `C++`. This means it is permissible to use any and all TEX commands within LATEX source code, and it all still works. Some of the features offered by LATEX that would be challenging to implement in TEX include automatic index and table-of-content creation.

Tim Edwards' `Xcircuit` drafting program

This wonderful program is what I use to create all the schematic diagrams and illustrations (but not photographic images or mathematical plots) throughout the ModEL project. It natively outputs PostScript format which is a true vector graphic format (this is why the images do not pixellate when you zoom in for a closer view), and it is so simple to use that I have never had to read the manual! Object libraries are easy to create for `Xcircuit`, being plain-text files using PostScript programming conventions. Over the years I have collected a large set of object libraries useful for drawing electrical and electronic schematics, pictorial diagrams, and other technical illustrations.

`Gimp` graphic image manipulation program

Essentially an open-source clone of Adobe's `PhotoShop`, I use `Gimp` to resize, crop, and convert file formats for all of the photographic images appearing in the ModEL modules. Although `Gimp` does offer its own scripting language (called `Script-Fu`), I have never had occasion to use it. Thus, my utilization of `Gimp` to merely crop, resize, and convert graphic images is akin to using a sword to slice bread.

`SPICE` circuit simulation program

`SPICE` is to circuit analysis as TEX is to document creation: it is a form of markup language designed to describe a certain object to be processed in plain-ASCII text. When the plain-text "source file" is compiled by the software, it outputs the final result. More modern circuit analysis tools certainly exist, but I prefer `SPICE` for the following reasons: it is *free*, it is *fast*, it is *reliable*, and it is a fantastic tool for *teaching* students of electricity and electronics how to write simple code. I happen to use rather old versions of `SPICE`, version 2g6 being my "go to" application when I only require text-based output. `NGSPICE` (version 26), which is based on Berkeley `SPICE` version 3f5, is used when I require graphical output for such things as time-domain waveforms and Bode plots. In all `SPICE` example netlists I strive to use coding conventions compatible with all `SPICE` versions.

Andrew D. Hwang's `ePiX` mathematical visualization programming library

This amazing project is a `C++` library you may link to any `C/C++` code for the purpose of generating PostScript graphic images of mathematical functions. As a completely free and open-source project, it does all the plotting I would otherwise use a Computer Algebra System (CAS) such as `Mathematica` or `Maple` to do. It should be said that `ePiX` is *not* a Computer Algebra System like `Mathematica` or `Maple`, but merely a mathematical *visualization* tool. In other words, it won't determine integrals for you (you'll have to implement that in your own `C/C++` code!), but it can graph the results, and it does so beautifully. What I really admire about `ePiX` is that it is a `C++` programming library, which means it builds on the existing power and toolset available with that programming language. Mr. Hwang could have probably developed his own stand-alone application for mathematical plotting, but by creating a `C++` library to do the same thing he accomplished something much greater.

`gnuplot` mathematical visualization software

Another open-source tool for mathematical visualization is `gnuplot`. Interestingly, this tool is *not* part of Richard Stallman's GNU project, its name being a coincidence. For this reason the authors prefer "gnu" *not* be capitalized at all to avoid confusion. This is a much "lighter-weight" alternative to a spreadsheet for plotting tabular data, and the fact that it easily outputs directly to an X11 console or a file in a number of different graphical formats (including PostScript) is very helpful. I typically set my `gnuplot` output format to default (X11 on my Linux PC) for quick viewing while I'm developing a visualization, then switch to PostScript file export once the visual is ready to include in the document(s) I'm writing. As with my use of `Gimp` to do rudimentary image editing, my use of `gnuplot` only scratches the surface of its capabilities, but the important points are that it's *free* and that it *works well*.

`Python` programming language

Both Python and C++ find extensive use in these modules as instructional aids and exercises, but I'm listing Python here as a *tool* for myself because I use it almost daily as a *calculator*. If you open a Python interpreter console and type `from math import *` you can type mathematical expressions and have it return results just as you would on a hand calculator. Complex-number (i.e. *phasor*) arithmetic is similarly supported if you include the complex-math library (`from cmath import *`). Examples of this are shown in the Programming References chapter (if included) in each module. Of course, being a fully-featured programming language, Python also supports conditionals, loops, and other structures useful for calculation of quantities. Also, running in a console environment where all entries and returned values show as text in a chronologically-ordered list makes it easy to copy-and-paste those calculations to document exactly how they were performed.

Appendix D

Creative Commons License

Creative Commons Attribution 4.0 International Public License

By exercising the Licensed Rights (defined below), You accept and agree to be bound by the terms and conditions of this Creative Commons Attribution 4.0 International Public License ("Public License"). To the extent this Public License may be interpreted as a contract, You are granted the Licensed Rights in consideration of Your acceptance of these terms and conditions, and the Licensor grants You such rights in consideration of benefits the Licensor receives from making the Licensed Material available under these terms and conditions.

Section 1 – Definitions.

a. **Adapted Material** means material subject to Copyright and Similar Rights that is derived from or based upon the Licensed Material and in which the Licensed Material is translated, altered, arranged, transformed, or otherwise modified in a manner requiring permission under the Copyright and Similar Rights held by the Licensor. For purposes of this Public License, where the Licensed Material is a musical work, performance, or sound recording, Adapted Material is always produced where the Licensed Material is synched in timed relation with a moving image.

b. **Adapter's License** means the license You apply to Your Copyright and Similar Rights in Your contributions to Adapted Material in accordance with the terms and conditions of this Public License.

c. **Copyright and Similar Rights** means copyright and/or similar rights closely related to copyright including, without limitation, performance, broadcast, sound recording, and Sui Generis Database Rights, without regard to how the rights are labeled or categorized. For purposes of this Public License, the rights specified in Section 2(b)(1)-(2) are not Copyright and Similar Rights.

d. **Effective Technological Measures** means those measures that, in the absence of proper authority, may not be circumvented under laws fulfilling obligations under Article 11 of the WIPO Copyright Treaty adopted on December 20, 1996, and/or similar international agreements.

e. **Exceptions and Limitations** means fair use, fair dealing, and/or any other exception or

limitation to Copyright and Similar Rights that applies to Your use of the Licensed Material.

f. **Licensed Material** means the artistic or literary work, database, or other material to which the Licensor applied this Public License.

g. **Licensed Rights** means the rights granted to You subject to the terms and conditions of this Public License, which are limited to all Copyright and Similar Rights that apply to Your use of the Licensed Material and that the Licensor has authority to license.

h. **Licensor** means the individual(s) or entity(ies) granting rights under this Public License.

i. **Share** means to provide material to the public by any means or process that requires permission under the Licensed Rights, such as reproduction, public display, public performance, distribution, dissemination, communication, or importation, and to make material available to the public including in ways that members of the public may access the material from a place and at a time individually chosen by them.

j. **Sui Generis Database Rights** means rights other than copyright resulting from Directive 96/9/EC of the European Parliament and of the Council of 11 March 1996 on the legal protection of databases, as amended and/or succeeded, as well as other essentially equivalent rights anywhere in the world.

k. **You** means the individual or entity exercising the Licensed Rights under this Public License. **Your** has a corresponding meaning.

Section 2 – Scope.

a. License grant.

1. Subject to the terms and conditions of this Public License, the Licensor hereby grants You a worldwide, royalty-free, non-sublicensable, non-exclusive, irrevocable license to exercise the Licensed Rights in the Licensed Material to:

A. reproduce and Share the Licensed Material, in whole or in part; and

B. produce, reproduce, and Share Adapted Material.

2. Exceptions and Limitations. For the avoidance of doubt, where Exceptions and Limitations apply to Your use, this Public License does not apply, and You do not need to comply with its terms and conditions.

3. Term. The term of this Public License is specified in Section 6(a).

4. Media and formats; technical modifications allowed. The Licensor authorizes You to exercise the Licensed Rights in all media and formats whether now known or hereafter created, and to make technical modifications necessary to do so. The Licensor waives and/or agrees not to assert any right or authority to forbid You from making technical modifications necessary to exercise the Licensed Rights, including technical modifications necessary to circumvent Effective Technological Measures.

For purposes of this Public License, simply making modifications authorized by this Section 2(a)(4) never produces Adapted Material.

5. Downstream recipients.

A. Offer from the Licensor – Licensed Material. Every recipient of the Licensed Material automatically receives an offer from the Licensor to exercise the Licensed Rights under the terms and conditions of this Public License.

B. No downstream restrictions. You may not offer or impose any additional or different terms or conditions on, or apply any Effective Technological Measures to, the Licensed Material if doing so restricts exercise of the Licensed Rights by any recipient of the Licensed Material.

6. No endorsement. Nothing in this Public License constitutes or may be construed as permission to assert or imply that You are, or that Your use of the Licensed Material is, connected with, or sponsored, endorsed, or granted official status by, the Licensor or others designated to receive attribution as provided in Section 3(a)(1)(A)(i).

b. Other rights.

1. Moral rights, such as the right of integrity, are not licensed under this Public License, nor are publicity, privacy, and/or other similar personality rights; however, to the extent possible, the Licensor waives and/or agrees not to assert any such rights held by the Licensor to the limited extent necessary to allow You to exercise the Licensed Rights, but not otherwise.

2. Patent and trademark rights are not licensed under this Public License.

3. To the extent possible, the Licensor waives any right to collect royalties from You for the exercise of the Licensed Rights, whether directly or through a collecting society under any voluntary or waivable statutory or compulsory licensing scheme. In all other cases the Licensor expressly reserves any right to collect such royalties.

Section 3 – License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the following conditions.

a. Attribution.

1. If You Share the Licensed Material (including in modified form), You must:

A. retain the following if it is supplied by the Licensor with the Licensed Material:

i. identification of the creator(s) of the Licensed Material and any others designated to receive attribution, in any reasonable manner requested by the Licensor (including by pseudonym if designated);

ii. a copyright notice;

iii. a notice that refers to this Public License;

iv. a notice that refers to the disclaimer of warranties;

v. a URI or hyperlink to the Licensed Material to the extent reasonably practicable;

B. indicate if You modified the Licensed Material and retain an indication of any previous modifications; and

C. indicate the Licensed Material is licensed under this Public License, and include the text of, or the URI or hyperlink to, this Public License.

2. You may satisfy the conditions in Section 3(a)(1) in any reasonable manner based on the medium, means, and context in which You Share the Licensed Material. For example, it may be reasonable to satisfy the conditions by providing a URI or hyperlink to a resource that includes the required information.

3. If requested by the Licensor, You must remove any of the information required by Section 3(a)(1)(A) to the extent reasonably practicable.

4. If You Share Adapted Material You produce, the Adapter's License You apply must not prevent recipients of the Adapted Material from complying with this Public License.

Section 4 – Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that apply to Your use of the Licensed Material:

a. for the avoidance of doubt, Section 2(a)(1) grants You the right to extract, reuse, reproduce, and Share all or a substantial portion of the contents of the database;

b. if You include all or a substantial portion of the database contents in a database in which You have Sui Generis Database Rights, then the database in which You have Sui Generis Database Rights (but not its individual contents) is Adapted Material; and

c. You must comply with the conditions in Section 3(a) if You Share all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not replace Your obligations under this Public License where the Licensed Rights include other Copyright and Similar Rights.

Section 5 – Disclaimer of Warranties and Limitation of Liability.

a. Unless otherwise separately undertaken by the Licensor, to the extent possible, the Licensor offers the Licensed Material as-is and as-available, and makes no representations or warranties of any kind concerning the Licensed Material, whether express, implied, statutory, or other. This includes, without limitation, warranties of title, merchantability, fitness for a particular purpose, non-infringement, absence of latent or other defects, accuracy, or the presence or absence of errors,

whether or not known or discoverable. Where disclaimers of warranties are not allowed in full or in part, this disclaimer may not apply to You.

b. To the extent possible, in no event will the Licensor be liable to You on any legal theory (including, without limitation, negligence) or otherwise for any direct, special, indirect, incidental, consequential, punitive, exemplary, or other losses, costs, expenses, or damages arising out of this Public License or use of the Licensed Material, even if the Licensor has been advised of the possibility of such losses, costs, expenses, or damages. Where a limitation of liability is not allowed in full or in part, this limitation may not apply to You.

c. The disclaimer of warranties and limitation of liability provided above shall be interpreted in a manner that, to the extent possible, most closely approximates an absolute disclaimer and waiver of all liability.

Section 6 – Term and Termination.

a. This Public License applies for the term of the Copyright and Similar Rights licensed here. However, if You fail to comply with this Public License, then Your rights under this Public License terminate automatically.

b. Where Your right to use the Licensed Material has terminated under Section 6(a), it reinstates:

1. automatically as of the date the violation is cured, provided it is cured within 30 days of Your discovery of the violation; or

2. upon express reinstatement by the Licensor.

For the avoidance of doubt, this Section 6(b) does not affect any right the Licensor may have to seek remedies for Your violations of this Public License.

c. For the avoidance of doubt, the Licensor may also offer the Licensed Material under separate terms or conditions or stop distributing the Licensed Material at any time; however, doing so will not terminate this Public License.

d. Sections 1, 5, 6, 7, and 8 survive termination of this Public License.

Section 7 – Other Terms and Conditions.

a. The Licensor shall not be bound by any additional or different terms or conditions communicated by You unless expressly agreed.

b. Any arrangements, understandings, or agreements regarding the Licensed Material not stated herein are separate from and independent of the terms and conditions of this Public License.

Section 8 – Interpretation.

a. For the avoidance of doubt, this Public License does not, and shall not be interpreted to, reduce, limit, restrict, or impose conditions on any use of the Licensed Material that could lawfully

be made without permission under this Public License.

b. To the extent possible, if any provision of this Public License is deemed unenforceable, it shall be automatically reformed to the minimum extent necessary to make it enforceable. If the provision cannot be reformed, it shall be severed from this Public License without affecting the enforceability of the remaining terms and conditions.

c. No term or condition of this Public License will be waived and no failure to comply consented to unless expressly agreed to by the Licensor.

d. Nothing in this Public License constitutes or may be interpreted as a limitation upon, or waiver of, any privileges and immunities that apply to the Licensor or You, including from the legal processes of any jurisdiction or authority.

Creative Commons is not a party to its public licenses. Notwithstanding, Creative Commons may elect to apply one of its public licenses to material it publishes and in those instances will be considered the "Licensor." Except for the limited purpose of indicating that material is shared under a Creative Commons public license or as otherwise permitted by the Creative Commons policies published at `creativecommons.org/policies`, Creative Commons does not authorize the use of the trademark "Creative Commons" or any other trademark or logo of Creative Commons without its prior written consent including, without limitation, in connection with any unauthorized modifications to any of its public licenses or any other arrangements, understandings, or agreements concerning use of licensed material. For the avoidance of doubt, this paragraph does not form part of the public licenses.

Creative Commons may be contacted at `creativecommons.org`.

Appendix E

References

"Guide to embedded security", document SWPB020D, Texas Instruments Incorporated, Dallas, TX, 2019.

"JTAG Tutorial", Corelis Incorporated, Cerritos, CA.

Miller, Martin T. *US Patent 9,843,402*, "Noise Analysis To Reveal Jitter And Crosstalk's Effect On Signal Integrity", application 4 October 2016, patent granted 12 December 2017.

"MSP430 Programming With the JTAG Interface", document SLAU320AI, Texas Instruments Incorporated, Dallas, TX, 2020.

"RSR Logic Probes and Pulsers" datasheet, part number 01LP610, Electronix Express.

"Understanding Data Eye Diagram Methodology for Analyzing High Speed Digital Signals" Application Note AND9075/D revision 1, ON Semiconductor, Semiconductor Components Industries LLC, Denver, CO, June 2015.

"What is JTAG? – and how can I make use of it?", Cambridge, UK.

Appendix F

Version history

This is a list showing all significant additions, corrections, and other edits made to this learning module. Each entry is referenced by calendar date in reverse chronological order (newest version first), which appears on the front cover of every learning module for easy reference. Any contributors to this open-source document are listed here as well.

19 November 2025 – added some instructor notes, and corrected a typographical error courtesy of Wyatt Gleason. Also added more explanatory text to the "Logic probe and pulser" question.

21 April 2025 – minor edits to the Tutorial.

20 November 2024 – re-named module "Digital Circuit Diagnostic Principles" and added some new Tutorial sections on logic states and voltage levels. Also made some minor edits to other sections of the Tutorial.

6 November 2024 – divided the Introduction chapter into two sections, one for students and one for instructors, and added content to the instructor section recommending learning outcomes and measures.

23 April 2024 – added more instructor notes as well as elaborated on a Tutorial footnote about the relatively high resistance-per-length of typical PCB signal traces. Also added some explanatory content to the "Pulldown resistor sizing" Quantitative Reasoning question, and corrected a couple of typographical errors.

13 December 2023 – added more rigor to the Quantitative Reasoning question "Pulldown resistor sizing". Also added some more instructor notes in answers to other questions.

29 November 2023 – minor edits to the Tutorial.

28 April 2023 – added illustration and comments about two-wire resistance measurement as a contrast to four-wire.

28 November 2022 – placed questions at the top of the itemized list in the Introduction chapter

prompting students to devise experiments related to the tutorial content.

23 July 2022 – added a Case Tutorial chapter, with a section showing examples of diagnostic visualization tools for circuit diagnosis.

26 April 2022 – added more questions to the Introduction chapter.

1 December 2021 – corrected a typo (CMNOS should have been CMOS) in the "Simple logic probe circuit" Quantitative Reasoning question.

9 May 2021 – commented out or deleted empty chapters.

8 April 2021 – edited image_4044 to comment more explicitly on the irrelevance of wire resistance.

1 December 2020 – added commentary about comparators to the "Simple logic probe circuit" question, as well as minor additions to the Tutorial.

13-14 September 2020 – added more content to the Tutorial chapter.

30 August 2020 – document first created.

Index

Adding quantities to a qualitative problem, 66
Annotating diagrams, 65
Asynchronous, 39

Ball grid array, 26
Balun, 11
Bed-of-nails test, 28
BGA, 26
Boundary scan cell, 29
Breakout box, 21

Capacitance, parasitic, 36
Chain, JTAG, 31
Checking for exceptions, 66
Checking your work, 66
CMOS, 35
CMOS digital logic, 16
Code, computer, 75
Common-mode balun, 11
Current balun, 11

Design For Test, 29
DFT, 29
Digital phosphor oscilloscope, 22
Digital signal integrity, 39
Dimensional analysis, 65
DPO, 22

Edwards, Tim, 76
Eye diagram, 24

Fall time, 38
Family, logic gate, 34, 35

Graph values to solve a problem, 66
Greenleaf, Cynthia, 41

Hold time, 37
How to teach with these modules, 73
Hwang, Andrew D., 77

IC, 8
Identify given data, 65
Identify relevant principles, 65
IEEE standard 1149.1, 29
Inductance, parasitic, 36
Integrated circuit, 8
Integrity, signal, 39
Intermediate results, 65
Inverted instruction, 73

Joint Test Action Group, 29
JTAG, 29
JTAG chain, 31

Kelvin four-wire method, 17
Knuth, Donald, 76

Lamport, Leslie, 76
Limiting cases, 66
Logic analyzer, 25
Logic gate family, 34, 35
Logic level, 16
Logic probe, 19

Margin, noise, 34, 35
Metacognition, 46
Miller, Martin, 24
Mixed-signal oscilloscope, 25
Moolenaar, Bram, 75
MSO, 25
Multimeter, 16
Murphy, Lynn, 41

Noise margin, 15, 34, 35

Open-source, 75
Oscilloscope, 14, 22
Oscilloscope, digital phosphor, 22
Oscilloscope, mixed-signal, 25

Parasitic capacitance, 36
Parasitic inductance, 36
PCB, 8
Persistence mode, 23
Positive edge triggering, 38
Power supply rail, 35
Printed circuit board, 8
Problem-solving: annotate diagrams, 65
Problem-solving: check for exceptions, 66
Problem-solving: checking work, 66
Problem-solving: dimensional analysis, 65
Problem-solving: graph values, 66
Problem-solving: identify given data, 65
Problem-solving: identify relevant principles, 65
Problem-solving: interpret intermediate results, 65
Problem-solving: limiting cases, 66
Problem-solving: qualitative to quantitative, 66
Problem-solving: quantitative to qualitative, 66
Problem-solving: reductio ad absurdum, 66
Problem-solving: simplify the system, 65
Problem-solving: thought experiment, 65
Problem-solving: track units of measurement, 65
Problem-solving: visually represent the system, 65
Problem-solving: work in reverse, 66
Propagation delay, 38

Qualitatively approaching a quantitative problem, 66

Rail, power supply, 35
Reading Apprenticeship, 41
Reductio ad absurdum, 66, 72, 73
Register, 39
Resonance, 36
Rise time, 38

SBW, 32
Schoenbach, Ruth, 41
Scientific method, 46
Set-up time, 37
Signal integrity, 39
Simplifying a system, 65
SMD, 11
Socrates, 72
Socratic dialogue, 73
Spi-Bi-Wire, 32
SPICE, 41
Stallman, Richard, 75
Surface mount device, 11
Synchronous, 39

TAP, 30
Test Access Port, JTAG, 30
Thought experiment, 65
Time domain, 25
Toggle mode, 38
Torvalds, Linus, 75
Transition time, 38
TTL, 34
TTL digital logic, 16

Units of measurement, 65

Visualizing a system, 65

Work in reverse to solve a problem, 66
WYSIWYG, 75, 76