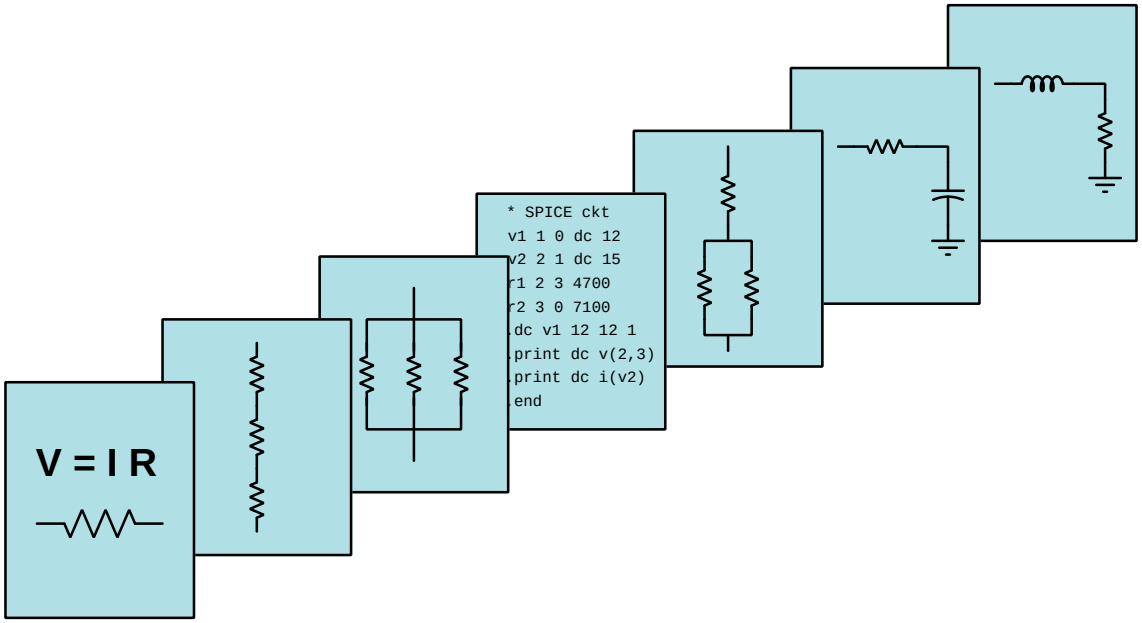
Modular Electronics Learning (ModEL) project



Digital Memory

Last update = 16 January 2025

Contents

Chapter 1

Introduction

1.1 Recommendations for students

One of the principal advantages of *digital* data is its relative ease of storage. Unlike analog data which must find expression in some physical form capable of the same infinitesimal resolution, digital data may be stored in any medium capable of retaining discrete states. The storage of digital data allows for the existence of computer *files*, digital audio recordings, digital movie recordings, with all the intrinsic advantages (and disadvantages!) of digital. The advantages of digital data storage vastly outweigh the advantages of analog for most applications. The general term we use for any form of digital data storage is *memory*, although this term is most often applied to integrated circuits intended to store digital data.

Important concepts related to digital memory include the representation of data in **discrete** form, digital **codes**, binary **counting**, data **volatility**, modes of data **access** (e.g. sequential), **latch** and **multivibrator** circuits, **addressing**, **tri-state** logic, **timing diagrams**, propagation **delay**, **set-up** and **hold** times, **decoding**, memory **expansion**, different memory **technologies**, and **electroscopes**.

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

- How might an experiment be designed and conducted to demontrate how to read and write solid-state RAM memory? What hypotheses (i.e. predictions) might you pose for that experiment, and what result(s) would either support or disprove those hypotheses?
- How might an experiment be designed and conducted to measure the minimum necessary set-up time for a flip-flop? 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 minimum necessary hold time for a flip-flop? What hypothesis (i.e. prediction) might you pose for that experiment, and what result(s) would either support or disprove that hypothesis?
- What property(ies) of a device or system is/are necessary for it to function as a memory element?

- What is the distinction between *reading* from memory versus *writing* to it?
- What is the distinction between *address* versus *data* for any particular memory device?
- How may we tell if a memory device is volatile or non-volatile?
- How may we tell if a memory device is sequential- or random-access?
- Why are tri-state logic gates necessary within certain types of memory circuits?
- What timing requirements may be necessary for reliably reading and writing memory data?
- Why do memory timing diagrams sometimes show data states that are both high and low at the same time?
- Why is the duration of the address enable (AE) line for a memory circuit different from the duration of the stable data on the I/O line(s)?
- How do we "expand" a memory array from smaller memory elements?
- What is the purpose of the various "enable" lines on a memory IC?
- What are some of the various sub-types of "read-only" (ROM) memory technologies?
- Where might we use specific types of digital memory technologies (e.g. PROM versus EEPROM, RAM versus magnetic disk, etc.)?

1.2 Challenging concepts related to digital memory

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

- **Reading hex dumps** – arrays of hexadecimal values showing the contents of a memory module are often referred to as *hex dumps*, and properly interpreting their contents is a skill in itself. Just know that the addresses go in incrementing order as the hex dump is read from left to right, top to bottom, just like reading an English-written document. The "offset" address given in the left-most column simply shows the beginning address value for that row of hex data, each successive column in the hex dump array moving toward the right reflecting an increment in address value from that starting "offset value".

- **Set-up and Hold times** – any digital circuit using timing pulses will have set-up and hold-time requirements, which are *minimum-time specifications* that all input pulse signals must meet in order for the circuit to reliably work. Always remember that these specified times are *minimum*, which means it's perfectly okay to exceed them but not safe to time the signals for less.

- **Memory expansion** – combining multiple memory ICs together to form larger, more practical memory arrays is an important concept in digital systems. Data-word expansion is easy enough to grasp, but address expansion requires more components to implement and tends to cause confusion for students new to the topic.

- **Tri-state gate outputs** – logic gates with tri-state output capability are commonly used to allow multiple digital devices to output data to common "bus" lines, only one of those devices being enabled at any given time. Tri-state outputs essentially disconnect the output of the logic gate from its external output terminal when in "Hi-Z" or *high-impedance* mode. When enabled, a tri-state output acts like the output of any regular logic gate, able to sink or source current.

The *Historical References* chapter contains multiple sections describing memory technologies, some legacy and some still popular. These are useful for applying concepts taught in the Tutorial, as well as for historical context. The *Case Tutorial* chapter also contains sections showing realistic examples of how multiple memory ICs may be interconnected to form memory arrays with wider data words, or with larger address spaces.

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.

- **Outcome** – Apply the concept of memory expansion to practical memory circuits

  Assessment – Predict logical states on address lines necessary to select a given address in an expanded memory array; e.g. pose problems in the form of the "Addressing an expanded memory module" Quantitative Reasoning question.

- **Outcome** – Design an expanded memory array circuit

  Assessment – Sketch a schematic diagram of a circuit employing multiple static RAM or ROM ICs to form a memory array with more data bits; e.g. pose problems in the form of the "Doubling data width" Quantitative Reasoning question.

  Assessment – Sketch a schematic diagram of a circuit employing multiple static RAM or ROM ICs to form a memory array with more addresses; e.g. pose problems in the form of the "Doubling address width" Quantitative Reasoning question.

- **Outcome** – Apply the concept of checksums to digital data

  Assessment – Use hash algorithms to validate the integrity of data files provided by the instructor or by other students, comparing hashes from files claimed to be identical.

- **Outcome** – Program an EPROM or NVRAM memory device with a specified data set

  Assessment – Use a PROM programming tool to load a hex dump style table into the memory IC.

  Assessment – Connect switches to the address and data inputs of a static memory IC to program its contents manually.

  Assessment – Locate static RAM datasheets and properly interpret some of the information contained in those documents including bus width, number of addresses, total memory capacity, timing diagrams showing set-up and hold time minimums, etc.

  Assessment – Locate dynamic RAM datasheets and properly interpret some of the information contained in those documents including bus width, number of addresses, total memory capacity, timing diagrams showing set-up and hold time minimums, etc.

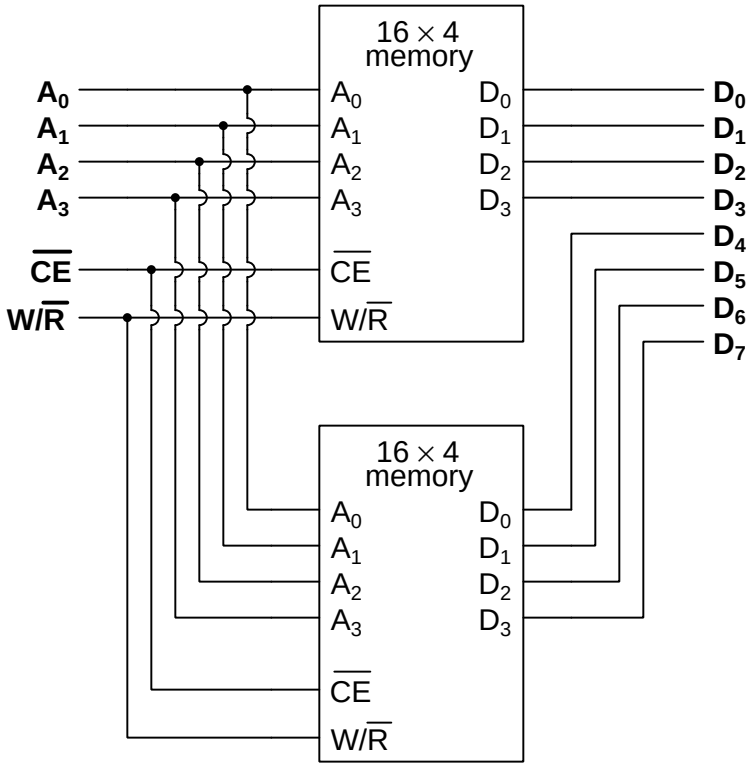
Chapter 2

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.

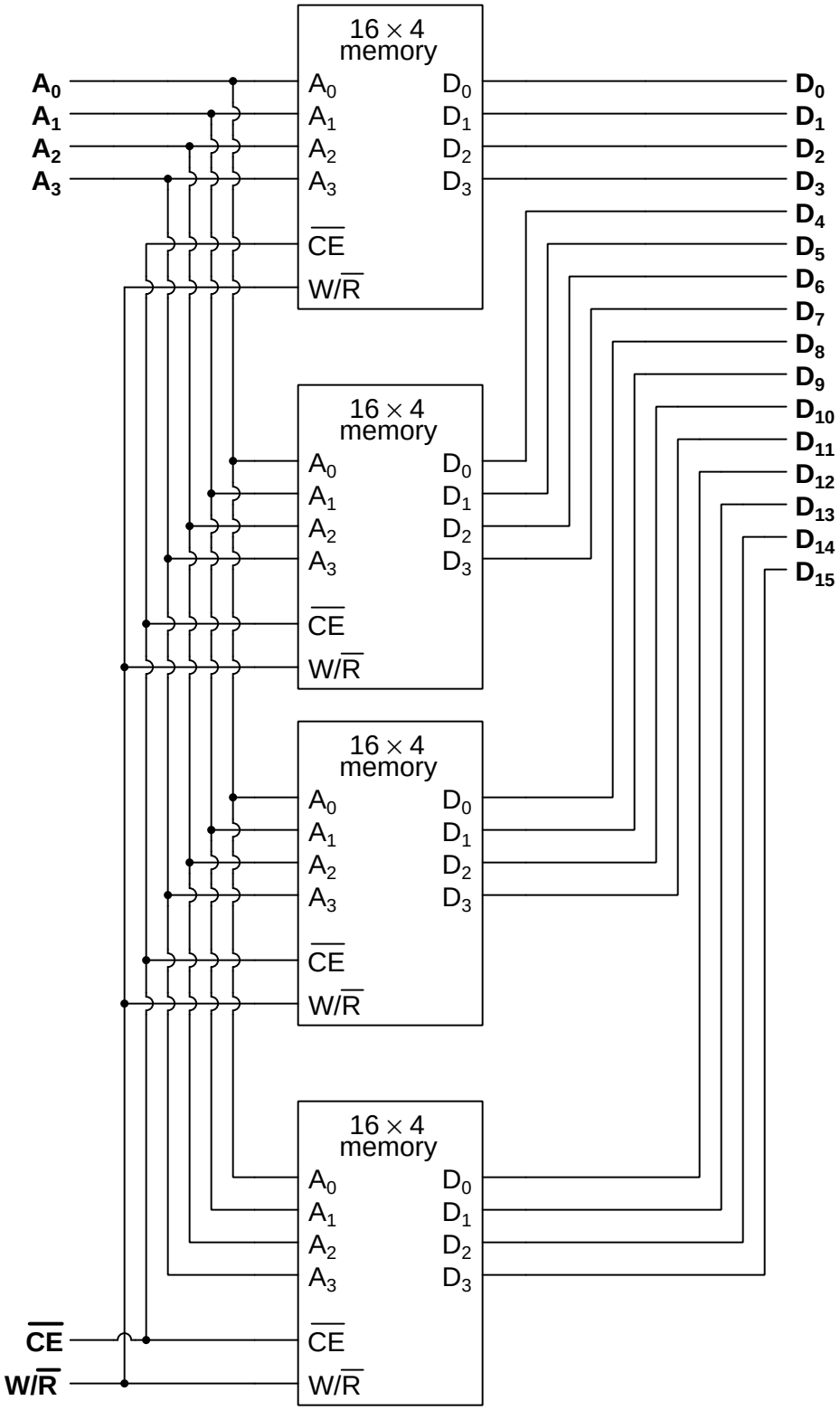
These examples also serve well as challenges following your reading of the other Tutorial(s) in this module – can you explain *why* the circuits behave as they do?

2.1 Example: 16x8 array from 16x4 memory ICs



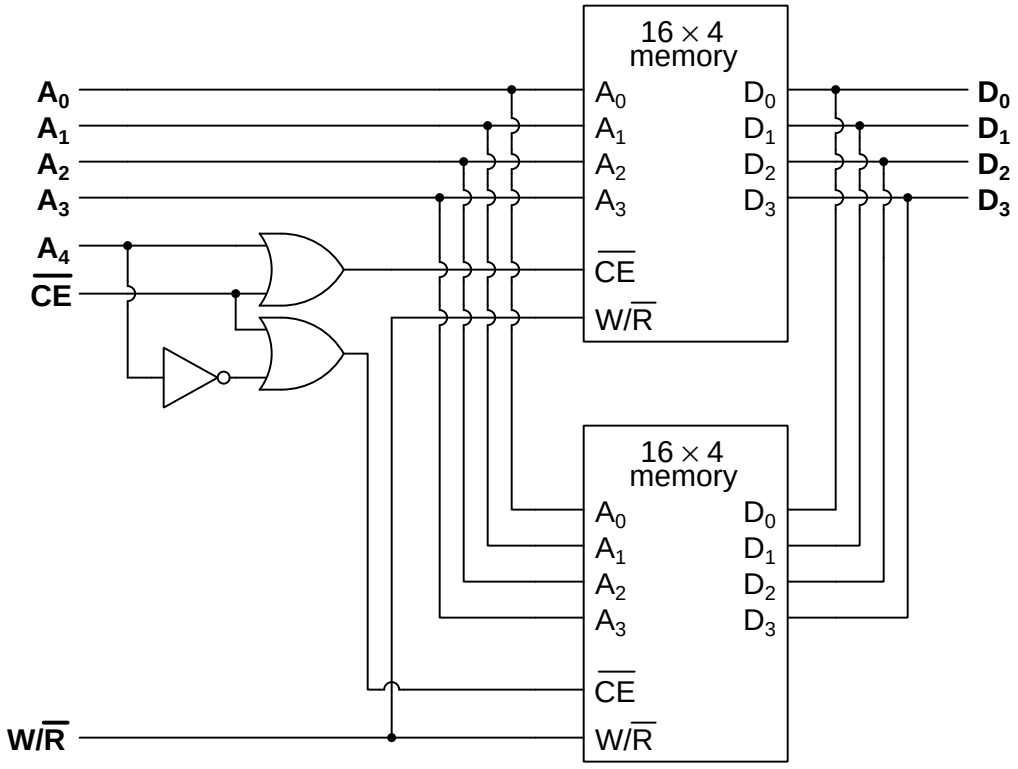
- Each memory ICs contains sixteen addressable memory cells, each cell storing a four-bit word
- The entire array contains sixteen addressable memory cells, each cell storing an eight-bit word
- Both memory ICs access their respective memory cells simultaneously
- Both memory ICs enable and disable simultaneously by command of the $\overline{CE}$ input signal
- Both memory ICs write or read simultaneously by command of the $W\overline{R}$ input signal

2.2 Example: 16x16 array from 16x4 memory ICs



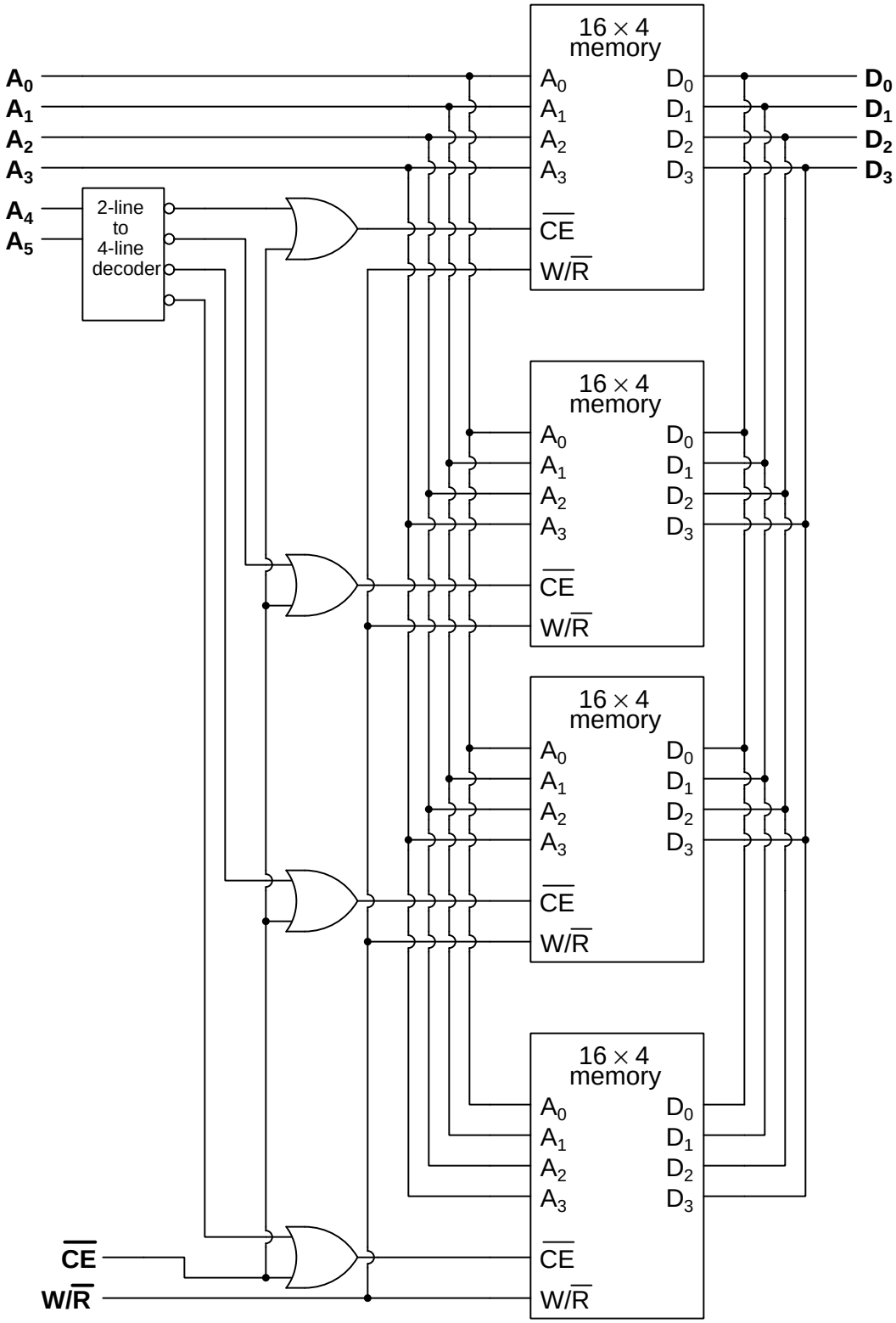
- Each memory ICs contains sixteen addressable memory cells, each cell storing a four-bit word
- The entire array contains sixteen addressable memory cells, each cell storing a sixteen-bit word
- Both memory ICs access their respective memory cells simultaneously
- Both memory ICs enable and disable simultaneously by command of the $\overline{CE}$ input signal
- Both memory ICs write or read simultaneously by command of the $W\overline{R}$ input signal

2.3 Example: 32x4 array from 16x4 memory ICs



- Each memory ICs contains sixteen addressable memory cells, each cell storing a four-bit word
- The entire array contains thirty-two addressable memory cells, each cell storing a four-bit word
- The upper IC accesses its memory cells for the first 16 addresses `0b00000` through `0b01111`
- The lower IC accesses its memory cells for the second 16 addresses `0b10000` through `0b11111`
- Both memory ICs enable and disable simultaneously by command of the $\overline{CE}$ input signal
- Both memory ICs write or read simultaneously by command of the $W/\overline{R}$ input signal

2.4 Example: 64x4 array from 16x4 memory ICs



- Each memory ICs contains sixteen addressable memory cells, each cell storing a four-bit word
- The entire array contains sixty-four addressable memory cells, each cell storing a four-bit word
- Each IC accesses its memory cells for one-quarter of the address space, i.e.:

  One IC for the first 16 addresses `0b000000` through `0b001111`

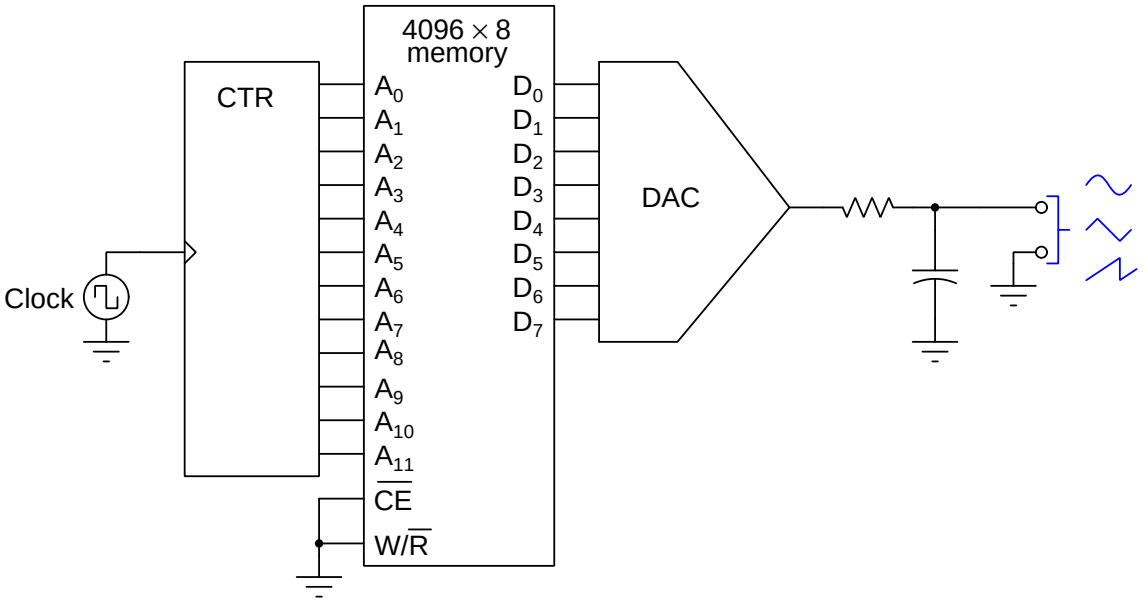
  Another for the next 16 addresses `0b010000` through `0b011111`

  Another for the next 16 addresses `0b100000` through `0b101111`

  The last for last 16 addresses `0b110000` through `0b111111`
- Both memory ICs enable and disable simultaneously by command of the $\overline{CE}$ input signal
- Both memory ICs write or read simultaneously by command of the $W/\overline{R}$ input signal

2.5 Example: AWG circuit using nonvolatile memory IC

A very useful piece of test equipment for electronics work is an *arbitrary waveform generator*, or *AWG*, designed to produce an AC or pulsing DC voltage signal with adjustable frequency. A parallel memory IC may be used as the basis for such a circuit, by programming the memory with numerical values corresponding to different levels of voltage in the desired waveform over time, and then using a clock-driven counter circuit to address the memory IC. The memory IC's output data then drives a digital-to-analog converter:



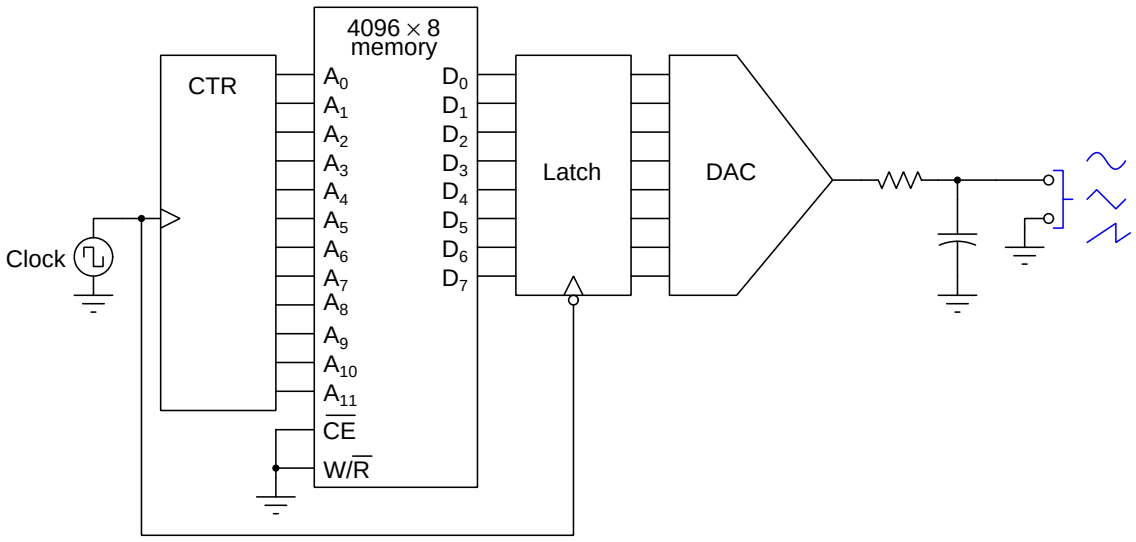
In the example shown here where we use a 12-bit counter to drive a 4096 × 8 memory IC, the arbitrary waveform is stored as a series of 4096 (2^{12}) numerical values in the memory IC which means the output waveform will have a frequency that is $\frac{1}{4096}$ that of the clock signal. The clock may be as simple as a 555 timer circuit, and should be adjustable-frequency. With 8 bits of data driving the DAC, the waveform's amplitude may be divided up into 256 (2^8) increments which establishes its voltage resolution. The result is a waveform that actually has a stair-step profile, but may be easily "smoothed" with some low-pass filtering.

If we do not require a very "smooth" wave-shape, we may use fewer address lines and therefore fewer addresses within the memory (this makes fewer horizontal steps in the stair-step output waveform, and also makes the output waveform's frequency greater for the same clock pulse frequency) and/or use fewer data lines (this makes fewer vertical steps in the stair-step output waveform).

Configuring the wave-shape for this AWG is simply a matter of programming different numerical values into the memory IC. A *sawtooth* waveform, for example, consists simply of steadily-incrementing data values written to consecutive addresses, so the 8-bit output begins at `0x00` and

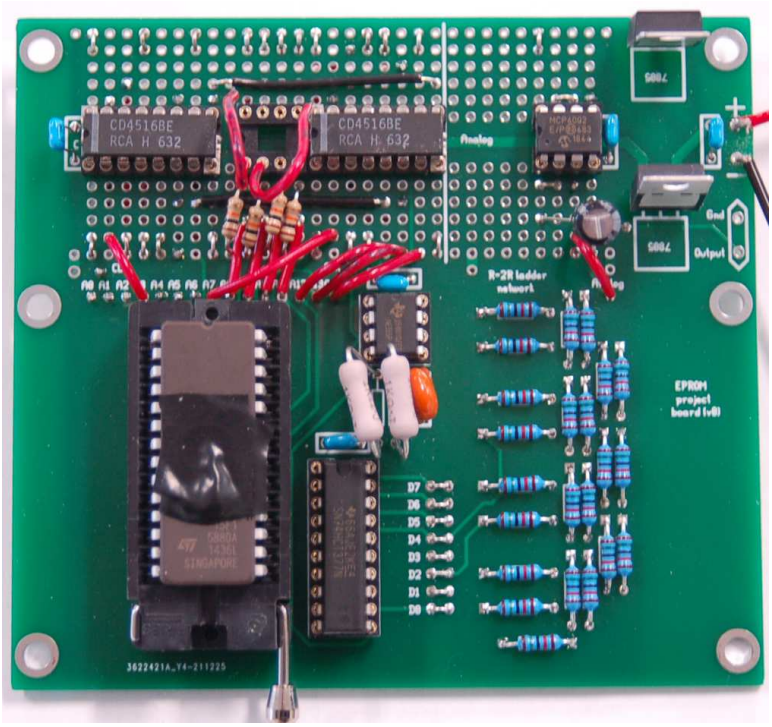
proceeds linearly to `0xFF` over as many time-steps as there are addresses. Such a sawtooth wave example would be a great application for truncating the memory space to the same number of bits as the data, so that each consecutive address will contain the identical value as data (e.g. address `0x00` is data `0x00`, address `0x01` is data `0x02`, etc. A *sine* wave would require the data values to be scaled linearly from the minimum and maximum values of a sine wave (e.g. the data values span 0 through 255 for an 8-bit DAC, corresponding to a sine wave oscillating between -1 and $+1$).

High-frequency AWG applications may require the addition of a data latch between the memory IC and the DAC, to avoid spurious results if the memory IC's data lines do not all settle at their next values simultaneously:



The latch's function is to "hold" the data value constant to the input of the DAC while the memory IC's data lines are transitioning states, and "pass" that new data to the DAC during a period when the counter's output is not changing and therefore not driving any new address values to the memory.

A photograph of a complete AWG built using a 27C256 UVEPROM appears below:



This particular printed circuit board (PCB) has etched traces for all the EPROM terminals, 555 timer circuit, 74HCT377 data latch, and R/2R "ladder" network as the digital-to-analog converter, but a generic "prototyping" area where a student could design and build their own digital counter and analog amplifier circuits.

Chapter 3

Tutorial

The most elementary form of signal is called *discrete*: a signal with only two possible states: *true* and *false*, or 1 and 0, respectively. Such signals may be expressed as electrical voltage levels, optical light intensities, magnetic polarities, physical component positions, subatomic particle "spin" states, and/or any other medium capable of two definite states. Alone, a discrete signal cannot represent anything but a single *bit* of information, but when grouped together a collection of bits may represent many things, for example *binary numbers* or *alphanumeric characters*. This is what we mean by the word *digital*: information represented by means of discrete bit states in *any* medium[1].

Furthermore, any medium capable of retaining a discrete status over some length of time may be employed as a means for the storage of digital information. So long as those discrete states are not easily disturbed, the medium will be able to "remember" the digital data even in the face of corrupting influences such as external noise.

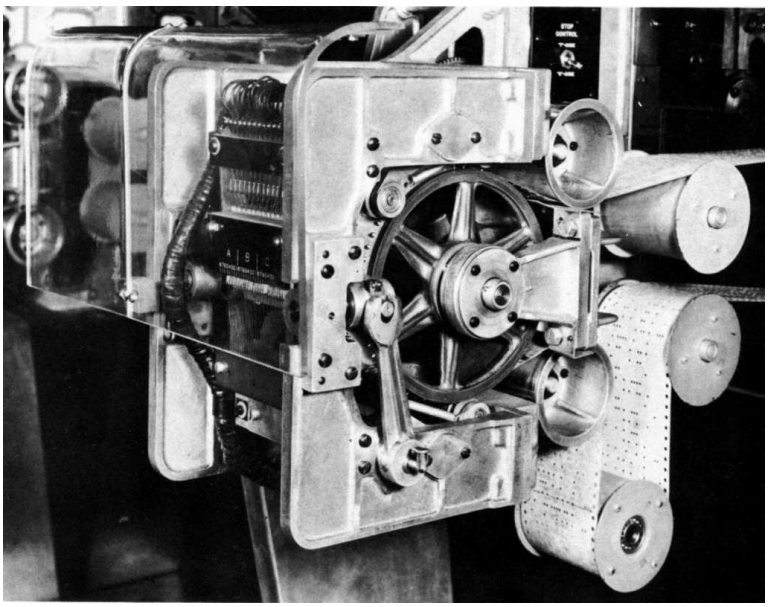
Digital data storage is incredibly useful, and has revolutionized our modern world. We have the ability today to archive and retrieve information in digital form that were unthinkable just a few decades ago, with the most evident form of progress being the ever-shrinking physical volume occupied by this storage.

A general term we use to describe any circuit, mechanism, or other device designed to "remember" information in digital form is *memory*. Many different types of memory technologies exist, and the purpose of this Simplified Tutorial is to introduce you to common principles as some of the more popular forms. We will begin with a form of memory that is considered obsolete but is very easy to understand: *punched paper tape*.

[1] In fact, DNA with its chemical bonds formed between adenine and thymine molecules, and also between guanine and cytosine molecules, is itself a form of digital information storage. These four types of molecules comprise four possible states for each "bit" of a DNA data "word", making DNA coding *quaternary* rather than *binary*.

3.1 Punched paper tape

The earliest digital electrical computers stored data both electrically and mechanically, electrical memory consisting of relay-based latching circuits and mechanical memory consisting of paper tape with small holes punched through. A hole (versus an un-punched area of tape) constituted the discrete states of each section of tape. A photograph of a mechanism designed to translate these holes and non-holes into discrete electrical states appears in the following photograph taken of the *Automatic Sequence Controlled Calculator* built by IBM in the 1940's. The tape with its patterns of punched holes is clearly visible, threaded through the mechanism:



As this tape passed through the mechanism, metal contacts placed on either side of the paper tape would make contact through a hole in the tape and be insulated from each other in the absence of a hole, thus translating the paper's discrete states (hole versus non-hole) into discrete electrical states (continuity versus non-continuity).

A mechanism such as the one pictured is said to *read* the tape: translating the paper holes/non-holes into electrical bits, akin to a human being translating ink markings on paper into word-based thoughts when reading a text. The complement to reading is *writing*, which took the form of a special hole-punching machine in the case of paper tape.

Reading and *writing* are fundamental concepts applicable to any form of memory or data storage system, and are easy to understand in the context of paper tape.

Another general concept associated with digital memory is *access*: the ability to locate data on the medium. In the case of tape, the access is *sequential* in nature because the tape can only move longitudinally. If the computer needed to read (or write) data at some location on the tape far from its present location, its only recourse was to fast-forward or rewind the tape until that location reached the mechanism, meanwhile passing over all the other locations on the tape in sequence.

Some digital memory technologies, by contrast, offer *random access* which means the ability to "jump" to any arbitrary data location on demand without having to pass through all the other data locations in sequential fashion.

An illustration of a punched tape section for the Automatic Sequence Controlled Calculator shows a set of "drive holes" at either edge designed to engage with sprocket pegs within the mechanisms. The smaller holes in the middle section of the tape encode the digital data. Text written beside the tape illustration comment on the meaning of the hole patterns. Thin horizontal lines drawn on the tape serve only as marks to help human readers separate sets of holes from each other, dividing the tape's data into distinct sections[2], each representing a single numerical quantity:

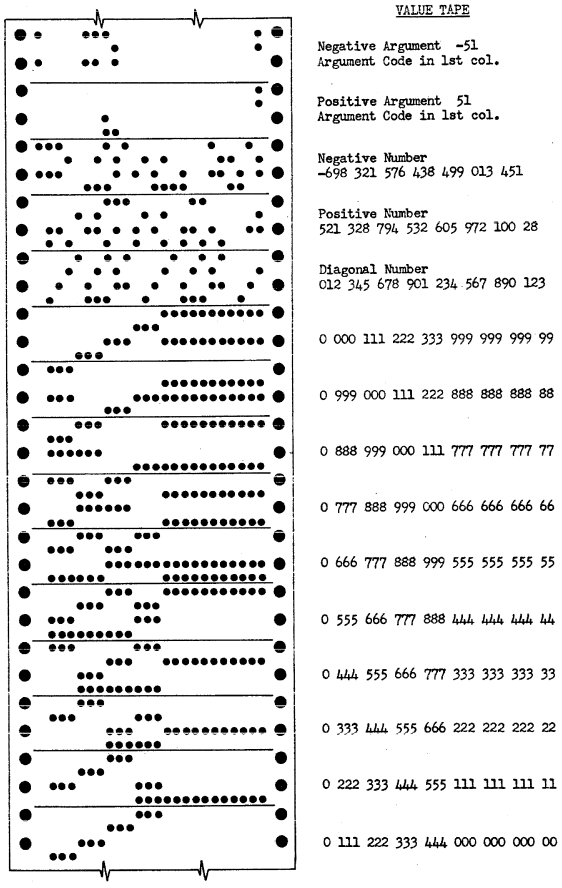


Figure 13

[2]The computer itself had no need for these lines, and in fact was blind to all but the holes. A careful examination of this illustration reveals a consistent placement of the drive holes with each section of tape, so that the computer needed only to align the drive sprocket to these intervals in order to properly orient the rows of holes for reading.

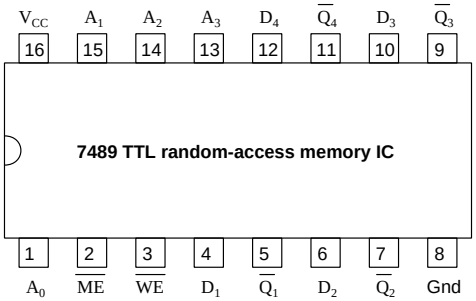
3.2 Digital memory principles

Punched paper tape, of course, is an obsolete technology. However, it serves as a simple example of concepts common to most memory technologies, as we saw with *reading* versus *writing* and *sequential-* versus *random-access*.

Another important concept of digital memory evident in punched paper tape is the distinction between *address* and *data*[3]. "Data" is the information encoded at any particular location in the memory, in the case of this paper tape from the 1940's data being the numerical value represented by any four-row pattern of punched holes. "Address" is the particular location of data along the length of the tape. We could imagine the first set of four data rows at one end of a tape being address number zero, with each successive data group having sequentially-numbered addresses.

Every digital memory is limited by addresses and also by the amount of data resident at each address. In the case of the paper tape example previously shown, the number of addresses is limited only by the length of the tape, while the amount of data per address is fixed at 96 bits[4], equivalent to (a maximum of) 24 decimal digits by the encoding scheme used within the Automatic Sequence Controlled Calculator. A magnetic hard disk (another form of digital memory) is limited in addresses by the maximum number of areas on the disk surface which may be magnetized to represent binary 1 and 0 states.

When we examine semiconductor memory ICs we generally find some pins[5] dedicated as address lines and other pins dedicated as data lines. The legacy model 7489 memory IC pin assignments are as follows, with four address lines labeled A_0 through A_3 and four data lines labeled D_1 through D_4:



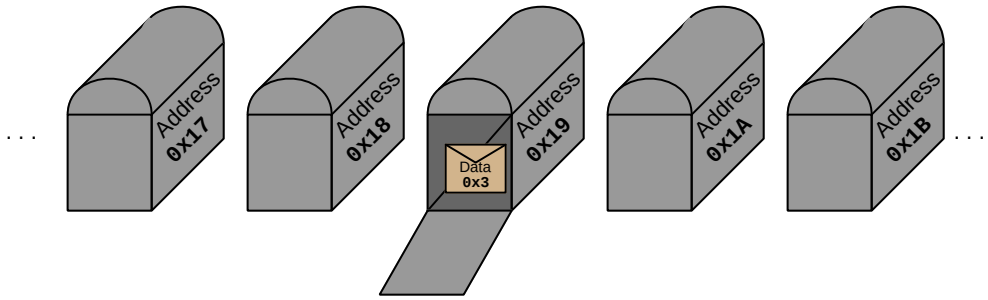
The model 7489 memory IC thus had a total capacity of 64 bits, organized as 16 × 4 bits – sixteen addresses containing four bits of data each. To select the address of interest we set the logic states of the address lines to the binary number desired (e.g. address number five would be selected by making $A_3 = 0$ and $A_2 = 1$ and $A_1 = 0$ and $A_0 = 1$) while the four-bit data word could either be applied to the data pins (with pin 3 set low to enable the IC's *write* mode) or measured on those same pins (with pin 3 set high to disable write mode and enter *read* mode, making the data pins outputs rather than inputs).

[3] Non-computer analogies exist to help understand these two terms. In a postal service, the *address* for any particular mailbox is usually a combination of numbers and words (e.g. 1234 Main Street) while the *data* is the information contained within each mailbox in the forms of letters, photographs, and other items residing inside.

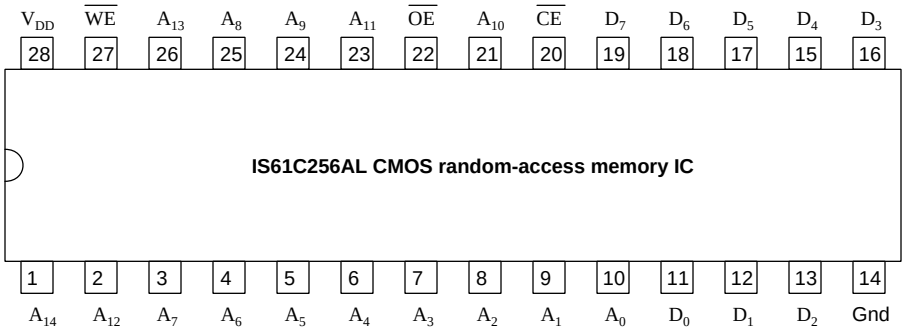
[4] A close examination of the tape reveals that each row contains 24 places for holes, and there are four of these rows per section. $24 \times 4 = 96$.

[5] This is true of so-called *parallel-addressed* memory ICs.

A helpful analogy for digital memory addresses versus data is to consider the function of a postal service *mailbox*, where the box bears a label called the *address* and is ready to contain letters inside of it (*data*). Here we see a row of mailboxes with sequential address labels, one of those mailboxes opened up to reveal data inside of it:



Modern digital memory devices have enormous capacity compared to the legacy 7489 IC. Consider the following IC with an organization of 32k × 8:

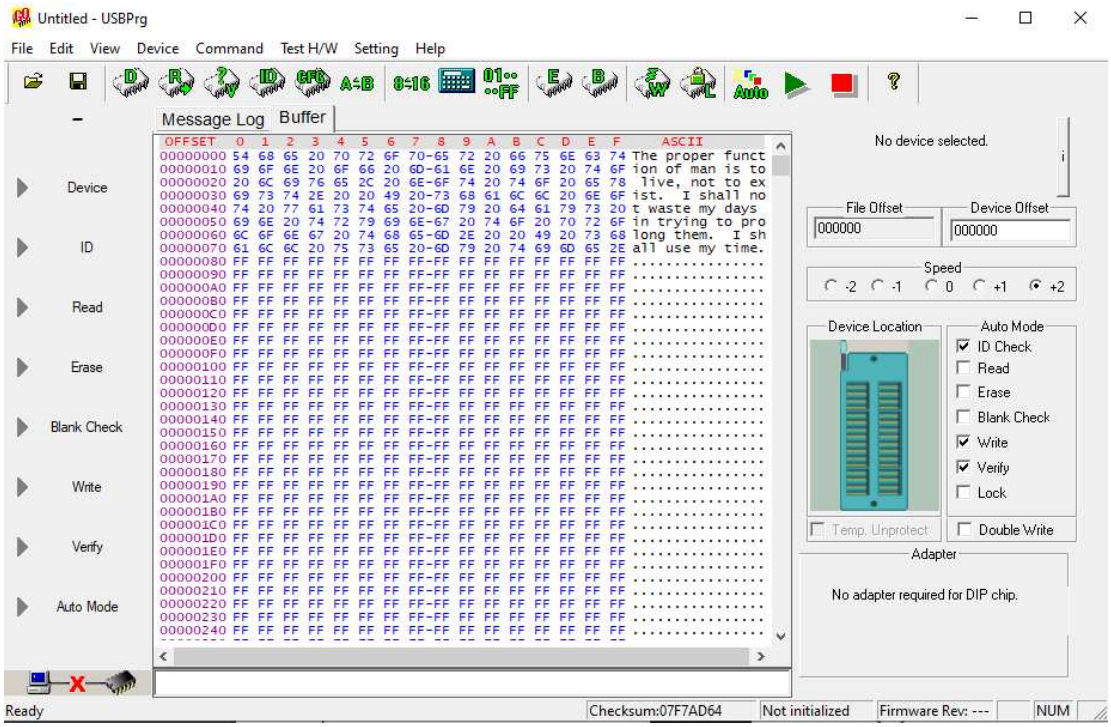


Despite its much larger capacity, this IC functions much the same as the legacy model 7489 with its lowly 16 × 4 organization. We still select the intended address within the IC by asserting "high" and "low" logical states on the address pins (A_0 through A_{14}) while either reading data from or writing data to the data pins (D_0 through D_7). Fifteen address lines represents $2^{15} = 32768$ unique addresses, each of those addresses storing one byte of data accessible via the eight data lines.

The fact that any address within this IC may be accessed simply and immediately by applying the desired "high" and "low" logical states to its address pins makes this device an example of random-access memory, because different addresses may be accessed in any (random) order. Nearly all solid-state memory devices are random-access in nature.

A common form of visual map for addresses and the data stored within a memory device is something called a *hex dump*. This is a listing of digital word values (in hexadecimal) printed left-to-right and top-to-bottom, with a listing of starting addresses on the left-hand side. For example, here is a hex dump shown on the software screen for a memory IC programming device, for a string of text written by the American author Jack London:

> *The proper function of man is to live, not to exist. I shall not waste my days in trying to prolong them. I shall use my time.*

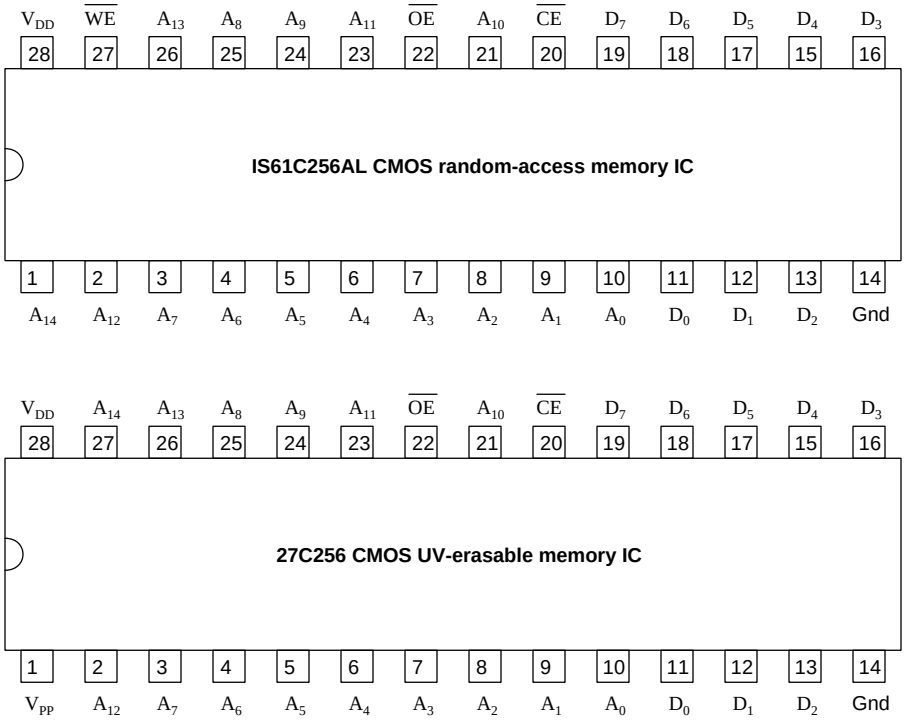


The column of numbers on the left-hand edge of the display labeled "OFFSET" (`00000000` through `00000240`) are the starting addresse values for each row. Each row contains sixteen bytes of data (each byte shown as a two-character hex value), which is why the starting addresses increment by "1" in the sixteen's place[6]. In this example, the data value `0x54` resides in memory address `0x00000000`, the data value `0x68` in address `0x00000001`, the data value `0x65` in address `0x00000003`, the data value `0x74` in address `0x0000000F`, etc. On the far-right side of this display we see the ASCII interpretation of these byte values, which for the digitization of the Jack London quote is perfectly appropriate. Often you will find hex dumps where the ASCII characters are gibberish because the data stored in the device's memory is intended for some purpose other than text.

[6]Remember that the address labels themselves are in hexadecimal as well! Thus, `0x00000000` is zero, `0x00000010` is sixteen, `0x00000020` is thirty-two, etc.

Yet another general concept associated with memory is *volatility*: the degree to which the bits are able to retain their proper states in the absence of any supporting energy or manipulation. Punched paper tape is considered *nonvolatile* memory because it "remembers" its data indefinitely without any external assistance. By contrast, many of the electrical relays used within the IBM Automatic Sequence Controlled Calculator for temporarily storing data would "forget" that data if de-energized, and would therefore be categorized as *volatile* memory.

For example, the model IS61C256AL memory IC shown previously happens to be volatile. However, the model 27C256 memory IC has the same organization (32k × 8) and nearly the same pin assignments but is nonvolatile[7]:



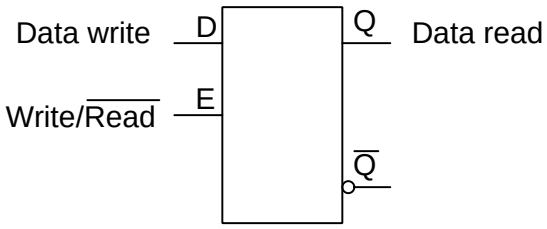
Let us recap these general concepts before continuing:

- **Reading versus Writing** – retrieving data from versus placing data to memory
- **Random-access versus Sequential-access** – being able to arbitrarily jump to a location in memory versus being forced to pass through all locations in sequence
- **Address versus Data** – the location at which information is stored versus the information being stored at a given location in memory
- **Volatile versus Nonvolatile** – requiring power to retain data versus not

[7] The 27C256 IC may be electrically written, but erased only by exposing the silicon die to ultraviolet light.

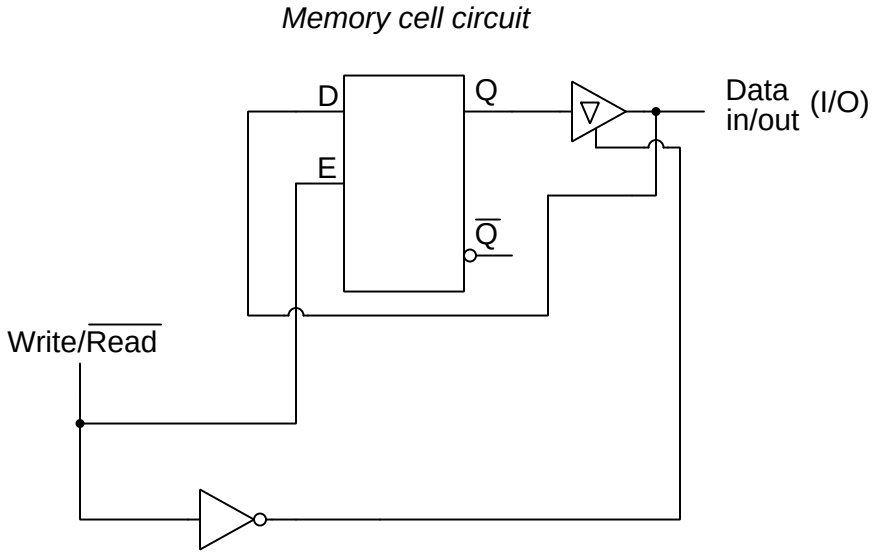
3.3 Electronic memory

A primitive example of volatile electronic memory is the *D latch* logic circuit, capable of storing a single bit of information so long as DC power is maintained to the D-type latch:



A single bit may be *written* to the latch through the D input terminal, as a discrete ground-referenced voltage signal[8]. The stored data may be read at the Q terminal at all times. The "Enable" (E) terminal serves to control when external data is written to the latch or ignored: when high the latch is in *write* mode with any data presented to the latch at terminal D transparently passed on to terminal Q; when low the latch is in *read* mode and it ignores any data input to its D terminal, simply "holding" (i.e. remembering) the previously-written data at its Q output terminal.

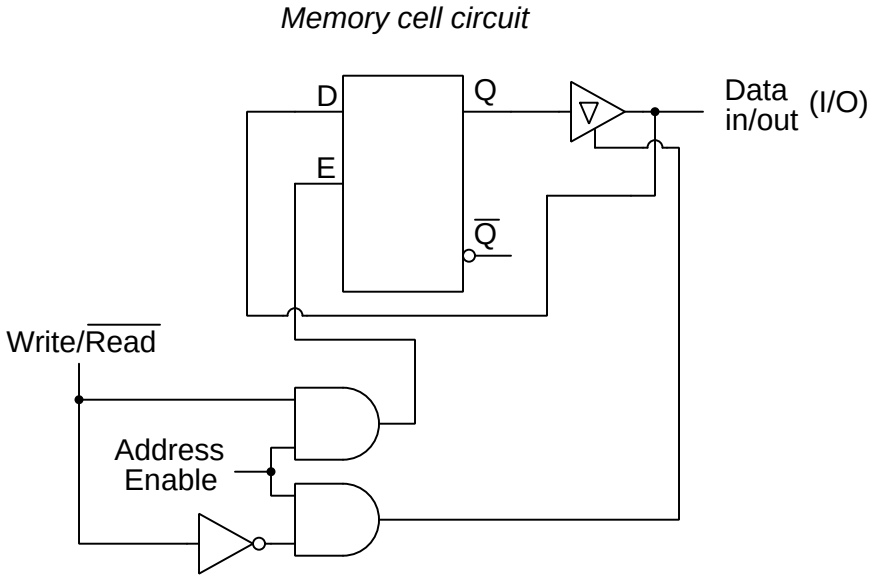
Most semiconductor-based memory circuits re-use the same terminals for alternately reading and writing data. In other words, a single-bit memory circuit will have a solitary "data in/out" terminal serving as an input during write mode and an output during read mode. This functionality is easy to add to our D-type latch using tri-state buffer and inverter gates:



This simple memory circuit has a storage capacity of one bit, and a single address for that bit.

[8] As usual, power supply terminals are omitted from this diagram for simplicity, but are absolutely essential to the proper operation of the circuit.

While this simple circuit works well as a one-bit "memory cell", we will need to find a way to link many of these "cells" together to form memory arrays if we are to ever have memory with significant data-storage capacity. In order to provide multiple addresses for single bits of data, we will need to have multiple memory cells and be able to selectively access one at a time. To do this, we must augment our memory cell design with a master enable/disable input useful for selecting which cell will be read from or written to. Since the purpose of this additional input is for addressing, we will call it the *address enable* line:

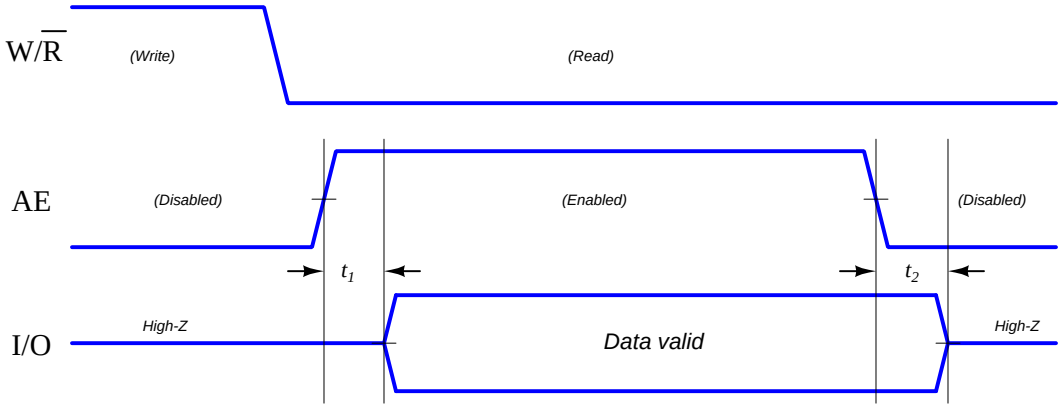


When high, the address enable line allows data to be read from or written to the memory cell. When low, the address enable line forces the outputs of both AND gates to go low, placing the tri-state buffer into its *high-impedance* modes to effectively "disconnect" the Q terminal from the data I/O line, so that the D-type latch cannot assert a logical state onto the I/O line.

Not only does the tri-state buffer allow this memory cell to take data in through the I/O line from some other source when in the Read mode (also with the address enabled, of course) while not sensing its own last-stored state from the latch's Q output, but it also prevents multiple memory cells sharing a common I/O line from accidently asserting opposing logic states on that same line when only one of those memory cells is being addressed.

3.4 Signal timing

Timing diagrams help illustrate the proper use of memory circuits, showing all the set-up and hold times necessary for reliable operation. Below we see a timing diagram for a *read* cycle where we give the memory cell circuit the read command (by making the Write/Read input line *low*) and "address" the memory cell (by making the Address Enable input high):



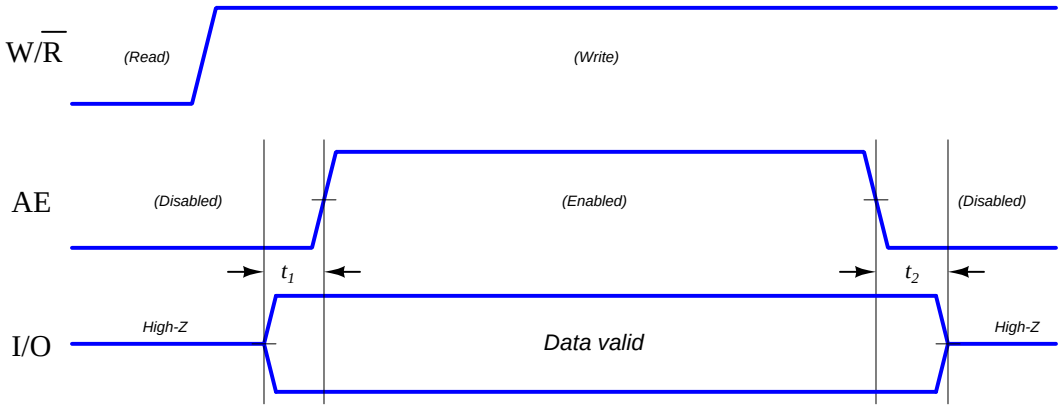
So long as the Address Enable line is disabled (i.e. low) the memory cell's data I/O line will be in its high-impedance mode, as shown in the timing diagram by the line being in the middle rather than firmly high or low[9]. Once the memory cell is enabled by the high Address Enable signal and the buffer's propagation delay time elapses, the D latch's Q output state appears on the I/O line ready to be "read" by any other digital device. Propagation delays (t_1 and t_2) associated with the AND gate and the tri-state buffers make the data's appearance "lag" the Address Enable signal.

The read data's appearance on this timing diagram may look very strange to anyone familiar with combinational logic circuits, as the "Data valid" time period appears to show this signal as being simultaneously high and low. This, of course, is an impossibility, and so the timing diagram must be communicating some other idea to us. Indeed it is, as this "high-and-low" notation simply means the memory cell will output *whatever state is held in the D latch*, be it high or low. Since this depends on data previously "written" to the D latch, and we have not been informed of this circuit's history prior to this read cycle, we have no way of knowing whether the I/O line will assume a high or a low state, *and so we draw both states.*

An important lesson we need to draw from this timing diagram about memory circuits in general is that certain input states are necessary to permit us to read their contents, and also that certain time delays must be anticipated between the assertion of those necessary states and the presentation of the stored data.

[9] This is not to imply that the actual voltage will be mid-way between the +V power supply rail and ground, but rather this is just a way to distinguish the high-impedance mode from a definite low or high state. In reality the data I/O signal's voltage level could be any value during that time. All we can say during the "disabled" time is that the memory cell circuit will not *assert* any logic state to its I/O line, either high or low.

Likewise, when writing data to a memory cell circuit we must be sure to abide by the sequence and timing requirements of the circuit as shown by a timing diagram. Below we see such a diagram showing a *write* cycle where we give the memory cell circuit the write command (by making the Write/Read input line *high*) and "address" the memory cell (by making the Address Enable input high):



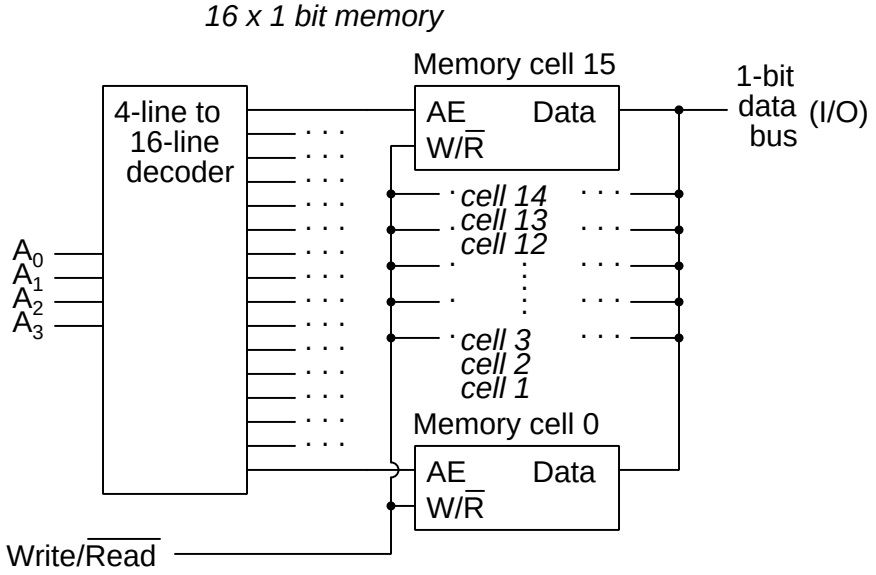
Since this is a *write* cycle and not a *read* cycle, data on the I/O line comes from some other digital device sending a signal to the memory cell. Note how this data must be stable ("valid") *prior* to the enabling of the circuit and remain stable *after* we disable the circuit for certain amounts of time (t_1 set-up time and t_2 hold time, respectively).

An important lesson we need to draw from this timing diagram about memory circuits in general is that data sent to a memory circuit must be stabilized at the I/O pin(s) both before and after the circuit is enabled for a write cycle. Failure to heed these set-up and hold times will result in unreliable operation.

These basic timing principles are true for multi-bit memory arrays as they are for single-bit memory cell circuits. Next, we will explore just how we may combine multiple memory cells together to form these larger arrays.

3.5 Combining memory elements

The following schematic diagram shows how sixteen such memory cells may be interconnected along with a 4-line to 16-line decoder to form a 16 × 1 bit memory array (i.e. sixteen addresses, each one storing one bit of data):

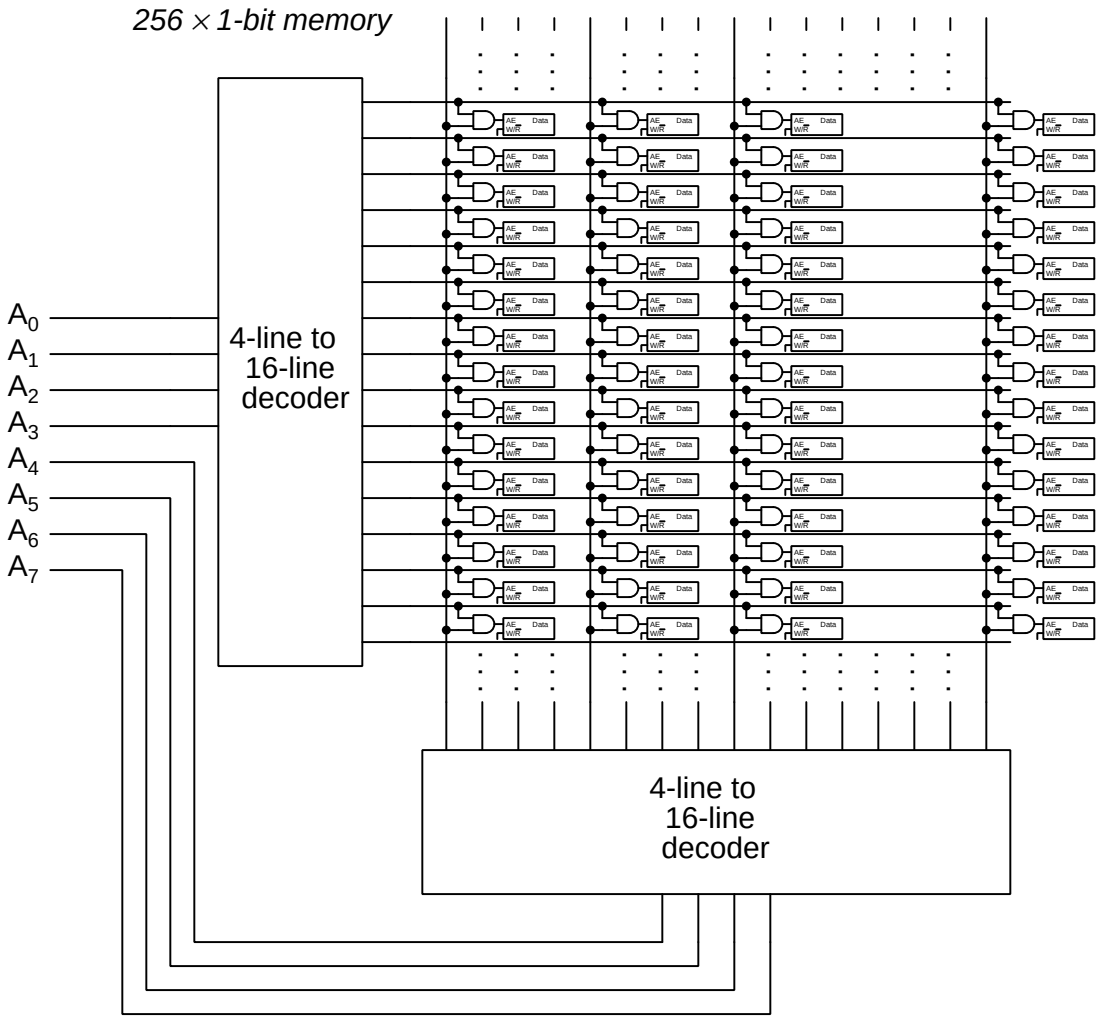


Four address lines (A_0 through A_3) accept a binary-numbered value into the decoder, which in turn selects (enables) which of the sixteen memory cells connects to the data bus for either reading or writing. Note how every one of the memory cells' $W/\overline{R}$ lines connect in common to a single $W/\overline{R}$ line for the 16 × 1 memory array, but with the individually-activated Address Enable (AE) lines only one memory cell may input or output its data at any given moment.

Expanding this memory array to possess a greater number of addresses simply consists of adding address lines and using a larger decoder to select a larger number of memory cells. For example, five address lines would yield 2^5 (thirty-two) memory addresses; eight address lines would yield 2^8 (two hundred and fifty-six) memory addresses; sixteen address lines would give us 2^{16} (65536) addresses.

In order to avoid the complexity of extremely large decoders (e.g. a 16-line to 65536-line decoder!) it is more practical to use pairs of smaller decoders to select memory cells arranged in a two-dimensional grid. For our hypothetical 65536 × 1 memory array, imagine an 8-line to 256-line decoder connected to the first eight address lines (A_0 through A_7) selecting which *row* of memory cells, plus another 8-line to 256-line decoder connected to the last eight address lines (A_8 through A_{15}) selecting which *column* of memory cells to access. Thus, we would have a "grid" of memory cells with 256 rows and 256 columns and a total of 65535 memory cells, each memory cell located at an intersection of one row and one column line. The address enable (AE) input of each memory cell would connect to the output of an AND gate, the AND gate's inputs connecting to the respective row and column lines such that only the cell located at the active row line *and* the active column line would become connected to the data bus for reading or writing.

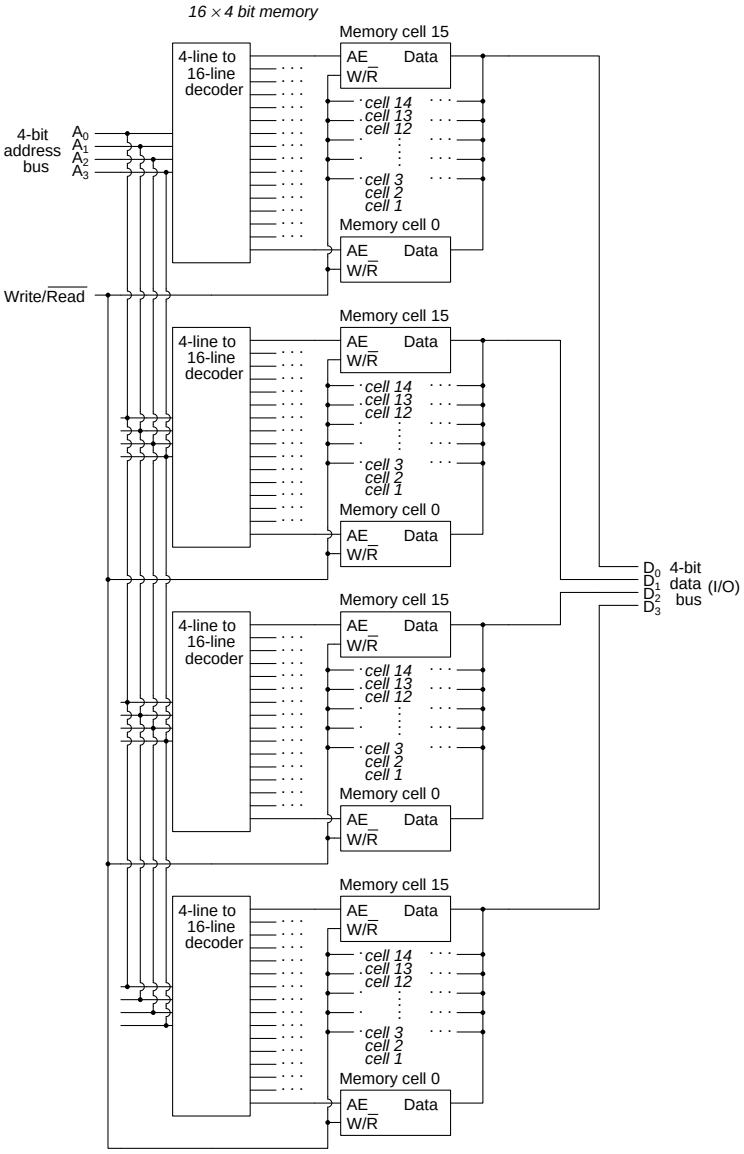
A simplified schematic diagram showing a 16 × 16 "grid" addressing just 256 memory cells appears below[10]. Each memory cell has an AND gate driving its *AE* input, with the two inputs of each AND gate connected to a unique combination of row and column lines:



Thus, the "grid" circuit design allows relatively large memory arrays to be constructed using decoders of modest line count.

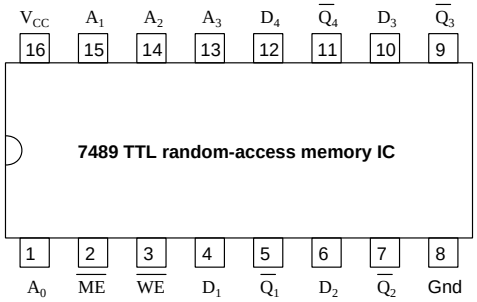
[10] Many wires and memory cells have been omitted from this diagram to avoid clutter, and unconnected wires crossing paths are simply overlaid on top of one another rather than shown with the semi-circular "jump" symbols as I typically use, simply because the schematic would become overwhelmingly complicated if shown in full detail. In fact, the number of memory cells shown here is only *one-quarter* of the 256 you would find in a real 16 × 16 grid memory!

However, no amount of address expansion will make this array store more than one bit per address. In order to expand the number of data bits stored within each address of the array, we must combine the circuits in a different manner. Examine the following 16 × 4 memory array, constructed from four 16 × 1 arrays:



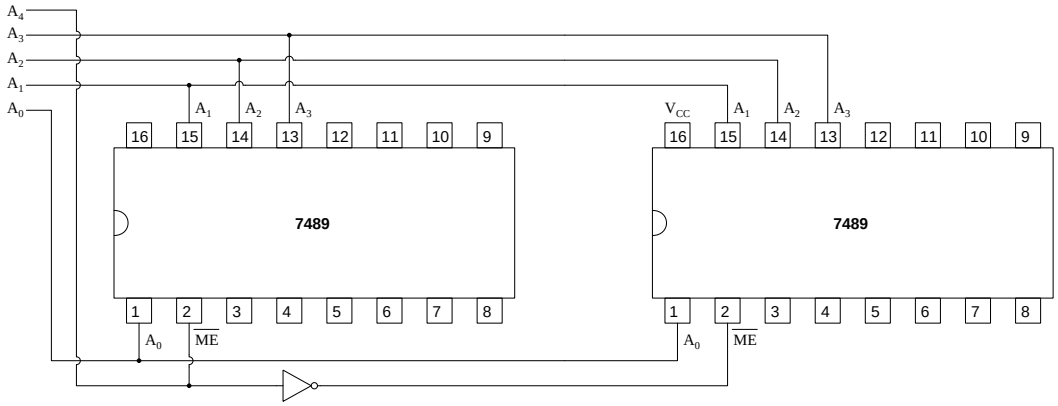
As you can see here, each of the 16 × 1 arrays shares the same address and $R/\overline{W}$ lines, but their respective data lines separately form a 4-bit data I/O bus. Thus, each array stores just one bit of the four-bit data *word* for the larger memory array.

The legacy model 7489 memory IC used a very similar form of 16 × 4 organization scheme:



Notable differences between the 7489 and our 16 × 4 memory circuit design include separate input and output data busses (D_0 through D_4 are inputs accepting data to be written, while $\overline{Q_0}$ through $\overline{Q_3}$ are outputs where data could be read from the memory) as well as a master *memory enable* ($\overline{ME}$, alternatively known as a *chip enable*, or $\overline{CE}$) line designed to enable or disable the entire integrated circuit by placing the data pins in high-impedance (tri-state) mode.

The "master enable" line is especially useful for combining multiple memory ICs to form even larger memory arrays. Consider the following example, where two 7489 ICs are combined to form a single 32 × 4 memory (showing only the address line connections for simplicity – the other terminals are all respectively paralleled):



A fifth address bit (A_4) selects which of the two 7489 memory ICs to enable: when low, the left-hand 7489 enables for reading and writing; when high, the right-hand 7489 enables. Thus, one memory IC handles the first sixteen addresses (`00000` through `01111`) while the second IC handles the next sixteen addresses (`10000` through `11111`).

This same strategy could be expanded as far as one likes, combining memory arrays to form even larger arrays, with each new address line doubling the number of memory addresses. We may even incorporate decoder ICs to simplify this expansion: all additional address lines become inputs to the decoder, while the decoder's output lines selectively enable one memory IC at a time. A 3-line to 8-line decoder, for example, would permit us to combine eight memory ICs together for an eight-fold expansion of the address space.

Before proceeding further, let us review our list of general memory concepts and see how they apply to these latch-based forms of digital memory:

- **Reading versus Writing** – controlled by the discrete state of a single $R/\overline{W}$ input terminal, which in turn controls which of the internal tri-state buffers pass data and which are in high-impedance mode
- **Address versus Data** – addressing is controlled by the digital states of a four-bit bus, while data is read and written by a separate four-bit bus
- **Random-access versus Sequential-access** – since addressing is entirely controlled by the binary value represented by the four-bit address bus, it is possible to "jump" to any arbitrary address desired without having to pass sequentially through all the others lying in between
- **Volatile versus Nonvolatile** – since latch circuits are based on interconnected semiconductor logic gates, and logic gates require the DC power to function, this memory is entirely volatile

3.6 Types of memory elements

Many digital applications require electronic forms of memory that are nonvolatile. An example of this is the *BIOS* of a personal computer, retaining important information the microprocessor needs to know immediately upon start-up in order to properly utilize the resources available to it (e.g. magnetic hard drives, communication ports, etc.). *Microcontrollers*, which are essentially single-chip computers, need some on-board nonvolatile memory in order to retain the programming which will instruct them what to do upon power-up as well.

If we were to replace the latch circuit at the heart of every memory cell of the previous examples with a hard-wired connection to +V or to Ground (a logical 1 or 0, respectively), maintaining most of the other circuitry required for addressing and multiplexing those hard-written cells, we would have a *read-only memory* useful for the aforementioned purposes. If the memory IC is designed in such a way that it may only be written (programmed) at the place of manufacture, we simply refer to it as a *ROM* (Read-Only Memory). If it is manufactured as a "blank slate" ready to be indelibly[11] programmed once by the end-user, it is known as a *PROM* (Programmable Read-Only Memory). Some PROMs may be bulk-erased, either by an electrical signal or by exposure to ultraviolet light, in which case they are called *EEPROM* (Electrically Erasable) or *UVPROM* (UltraViolet erasable) memories, respectively. Such *EPROM* (Erasable Programmable Read-Only) memories are the basis of digital system *firmware*, where instructions for some digital device may be deliberately altered by the user, but are otherwise only read by that device and not written to.

Unfortunately, read-write electronic memory systems such as those based on latch circuits are typically called *RAM* which of course stands for Random-Access Memory. While this type of memory is certainly random-access, so is every semiconductor-based read-only memory (ROM). So, the label of "RAM" in fact has more to do with the circuits being volatile than with being randomly-accessible.

Multiple technologies exist for RAM. Latch-based memory cells such as those we previously explored create what is known as *static RAM*. One disadvantage of static RAM is the number of transistors necessary to implement each cell. Since most practical RAM ICs contain *thousands* of these memory cells, even modest reductions in transistor count for each cell yield substantial savings in silicon overall. Fewer transistors per cell mean more cells that may be placed on the same area of silicon. Another disadvantage of static RAM, especially when based on TTL logic circuitry, is high power dissipation: thousands of latch circuits drawing current from the power source will dissipate significant quantities of heat and pose an operating limit on battery-powered digital circuits.

An alternative to a transistor-based latch-type memory cells is RAM memory using capacitively-triggered transistors, the logical memory state maintained as an electric charge (or lack of charge) on each capacitor. With a microscopic capacitance serving as the memory element instead of a latch circuit, the number of transistors per cell may be dramatically reduced, and the amount of memory available on a given chip dramatically increased. Also, power dissipation is less than a static RAM memory cell. However, a disadvantage of this form of memory is that the microscopic capacitors dissipate their stored energy rather rapidly due to parasitic loading, and must be periodically "refreshed" to retain their data. It is this transitory nature which earns them the label of *dynamic*

[11] Some PROMs use fusible links in the integrated circuit which may be intentionally "blown" by careful connections to an external power source. When one of these microscopic fuses blows, the memory cell assumes the complement of its factory-default state. Of course, this procedure is non-reversible: once blown, a fuse cannot be "healed" to restore the original logic state.

RAM. Modern dynamic RAM chips have on-board refresh circuitry to handle this for the user, while older dynamic RAMs required the user's application to periodically address certain regions of memory space in order to keep those cells' states refreshed[12].

A more modern semiconductor memory technology blurring the distinction between RAM and ROM is *flash memory*. It is similar in principle to dynamic RAM – storing single bit-states as electrostatic charges – but the physical construction and operating behavior is different. Flash memory cells utilize a special type of MOSFET transistor with a metal "floating gate" electrode suspended within layers of electrically-insulating material adjacent to the channel of the transistor. This floating electrode becomes electrostatically charged or discharged by means of *quantum tunneling*, a process whereby electrons "leap" from one nearby conductive surface to another without ever passing *through* the insulating material separating those conductive surfaces. This tunneling behavior is possible only under certain energy-level conditions, and if those conditions are not met the electrical charges remain isolated on either side of the insulating barrier as one would expect in "classical" physics. Thus, quantum tunneling becomes the sole method of *writing* data to a cell, which means the stored data is nonvolatile. Reading the stored charge (or lack of charge) on the floating gate is as simple as testing the conductivity of the MOSFET's channel, that channel acting as an electroscope (i.e. electric-charge detector) for the floating gate.

Flash memory cells basically consist of single MOSFETs which makes them extremely compact, and due to the strict conditions necessary for quantum tunneling the stored electrostatic charges do not bleed off in the same manner as conventional dynamic RAM. This gives flash memory reliable retention times in the span of *decades* as opposed to *milliseconds* for dynamic RAM cells.

The speed at which flash memory cells may be written and read exceeds current state-of-the-art mechanical memory technologies (e.g. magnetic disks) which is why this technology finds use in *solid-state hard drives* for computers. It is still not as fast as either dynamic or static RAM, though, which is why these latter technologies are still popular.

Another modern nonvolatile memory technology is *ferroelectric*[13], based on substances exhibiting a natural electric polarization (i.e. a permanent electric field), the polarity of that field being reversible by the application of sufficient external voltage. Like flash memory, ferroelectric memory elements are based on MOSFETs activated by this stored electric field. However, ferroelectric memory offers faster data access times and greater durability (more read/write cycles before failure) at the expense of lower physical density (fewer bits stored in a given volume).

[12] A datasheet for the Texas Instruments model TMS 4062 JL/NL 1024 × 1 dynamic RAM says the following about refreshing the cells: "Each cell must be refreshed at least once in every 2-millisecond period by cycling through the lower order row addresses (A0-A4) or by addressing each row at least once in that period. Addressing any row refreshes all 32 cells in that row. The chip-select clock need not be activated during refresh; however, the clock input must be cycled from high to low to high."

[13] Interestingly, the "ferro" root refers to the hysteretic behavior of these substances (i.e. they may be electrically polarized in a manner similar to how ferromagnetic substances may be magnetically polarized), not their chemical composition. That is to say, a "ferroelectric" need not contain any iron.

Reviewing the many (and confusing) labels used to describe semiconductor memory, all of which happen to be randomly-accessible:

- **RAM** – *Random-Access Memory*, technically a misnomer because all semiconductor memory is random-access. What this acronym really distinguishes is the memory's *volatility*, as data is lost or corrupted when de-energized. Typically very fast read/write times.
- **ROM** – *Read-Only Memory*. In its purest form, this is a *mask ROM* (MROM) where the data must be programmed into the IC at the time of manufacture, that data completely unalterable by the user.
- **PROM** – *Programmable Read-Only Memory*. Literally interpreted, this is an oxymoron (a contradiction in terms). It refers to a memory IC that may be programmed by the user, but afterwards either cannot be altered or requires substantial effort to alter. One-Time-Programmable (OTP) PROM ICs often use "fuse" or "anti-fuse"[14] elements to store the data, where blowing the fuse (or alternatively, shorting the anti-fuse) constitutes writing one of the two states. Such writing cannot be un-done, hence the term "one-time programmable".
- **EPROM** – *Erasable Programmable Read-Only Memory*. Literally interpreted, this is another oxymoron, referring to a PROM that may be erased by the user. EPROM re-writing times are much longer than RAM writing times, which is the major distinguishing feature.
- **UVEPROM** – *Electrically Erasable Programmable Read-Only Memory*. An EPROM that may be programmed multiple times via electrical signals.
- **EEPROM** – *Electrically Erasable Programmable Read-Only Memory*. An EPROM that may be erased by exposure to ultraviolet light.
- **NVRAM** – *Non-Volatile RAM*. A RAM IC with a built-in means to retain information when powered down. Some NVRAMs use an internal long-life battery to maintain energization when external power is lost. Others use a slower EPROM technology to copy the RAM's contents at power-down[15], and restore the RAM's contents at power-up. NVRAm provides all the benefits of non-volatility while enjoying the fast access times of standard RAM.

[14] Zener diodes may function as anti-fuses, because overpowering a Zener diode's PN junction will cause it to fail in a shorted state.

[15] An external capacitor is often required for this type of NVRAM, providing just enough energy for the internal data-archiving action in the absence of regular external power.

Chapter 4

Historical References

This chapter is where you will find references to historical texts and technologies related to the module's topic.

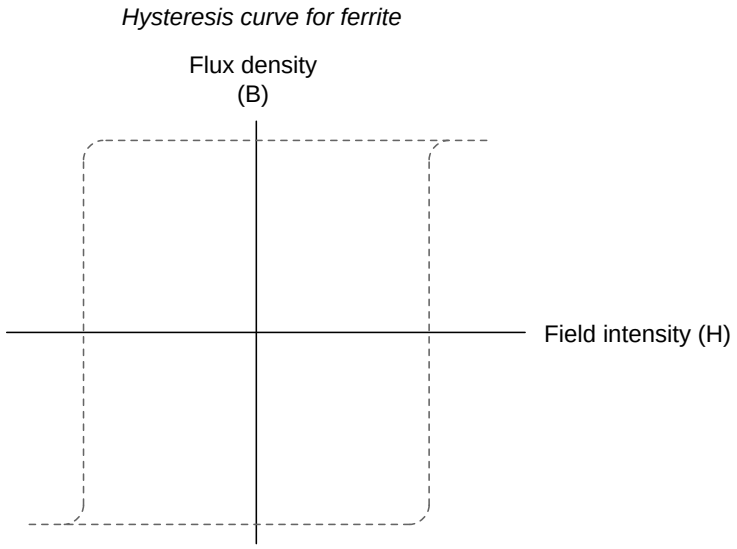
Readers may wonder why historical references might be included in any modern lesson on a subject. Why dwell on old ideas and obsolete technologies? One answer to this question is that the initial discoveries and early applications of scientific principles typically present those principles in forms that are unusually easy to grasp. Anyone who first discovers a new principle must necessarily do so from a perspective of ignorance (i.e. if you truly *discover* something yourself, it means you must have come to that discovery with no prior knowledge of it and no hints from others knowledgeable in it), and in so doing the discoverer lacks any hindsight or advantage that might have otherwise come from a more advanced perspective. Thus, discoverers are forced to think and express themselves in less-advanced terms, and this often makes their explanations more readily accessible to others who, like the discoverer, comes to this idea with no prior knowledge. Furthermore, early discoverers often faced the daunting challenge of explaining their new and complex ideas to a naturally skeptical scientific community, and this pressure incentivized clear and compelling communication. As James Clerk Maxwell eloquently stated in the Preface to his book *A Treatise on Electricity and Magnetism* written in 1873,

> It is of great advantage to the student of any subject to read the original memoirs on that subject, for science is always most completely assimilated when it is in its nascent state . . . `[page xi]`

Furthermore, grasping the historical context of technological discoveries is important for understanding how science intersects with culture and civilization, which is ever important because new discoveries and new applications of existing discoveries will always continue to impact our lives. One will often find themselves impressed by the ingenuity of previous generations, and by the high degree of refinement to which now-obsolete technologies were once raised. There is much to learn and much inspiration to be drawn from the technological past, and to the inquisitive mind these historical references are treasures waiting to be (re)-discovered.

4.1 Magnetic core memory

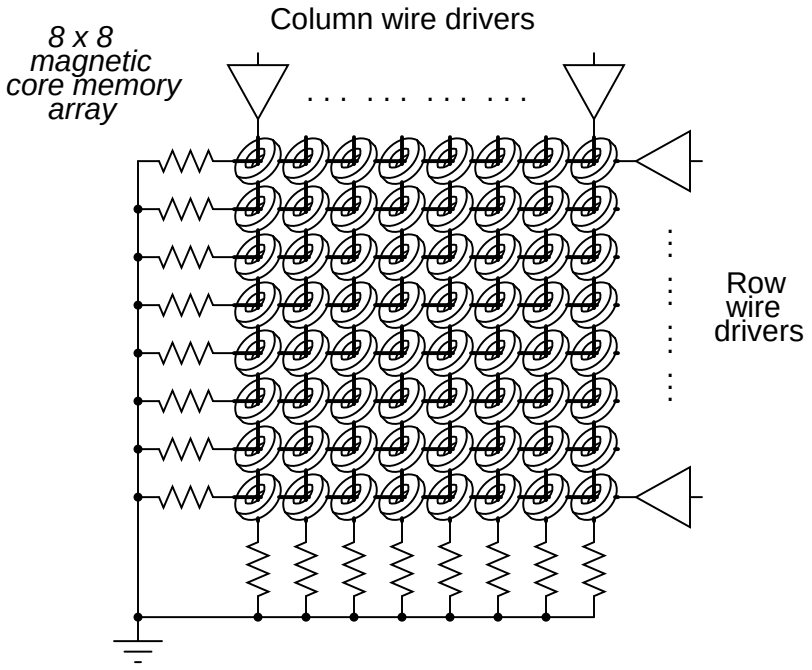
One of the earliest forms of solid-state memory for digital computers was *magnetic core memory*, often referred to simply as *core*. The fundamental principle of operation is the hysteresis curve for *ferrite*, a compound of iron and oxygen. Unlike most ferromagnetic materials, ferrite exhibits an extremely pronounced hysteresis curve which means once magnetized in a particular direction it tends very strongly to retain that magnetic flux. This property of ferrite makes it well-suited for all kinds of magnetic-based digital storage technologies including magnetic disks, magnetic tape, etc.



The field intensity (H) is the "magneto-motive force" (MMF) applied to the ferrite by an external magnetic source such as a current-carrying wire. Flux density (B) represents the number of magnetic domains within the ferrite that have aligned themselves in a common direction with the externally-applied field (H).

When ferrite magnetizes, it tends to do so until saturation which makes it ideal for storing discrete (1 versus 0) states – magnetization in one polarity represents a 1 state, and in the other polarity a 0 state. This single datum bit may be written to the ferrite specimen by a suitably strong H from a nearby current-carrying wire, with the direction of current determining the magnetic polarity and therefore the logical state written to the ferrite.

Core memory differs from disks and tape principally in the fact that the magnetic material does not need to physically move. Instead, tiny rings ("cores") of ferrite are arranged in grids with wires threaded through their centers as shown in the following illustration:



Each core has a "row" wire passing through its center, and also a "column" wire doing the same. Each wire's energized-state current is less than what would be required to saturate the ferrite, and so any ferrite current experiencing the magnetic field intensity (H) from just one wire's current will remain in its present magnetic flux condition (B). However, each wire's current is also regulated to be greater than one-half of the saturation threshold for ferrite, which means if one column wire and one row wire are simultaneously energized with current, the ferrite ring at the intersection of those two wires will experience saturation. Thus, we are able to address individual ferrite cores by column and row coordinates, just like individual RAM or ROM memory cells within large memory arrays are selectively enabled by row/column intersections.

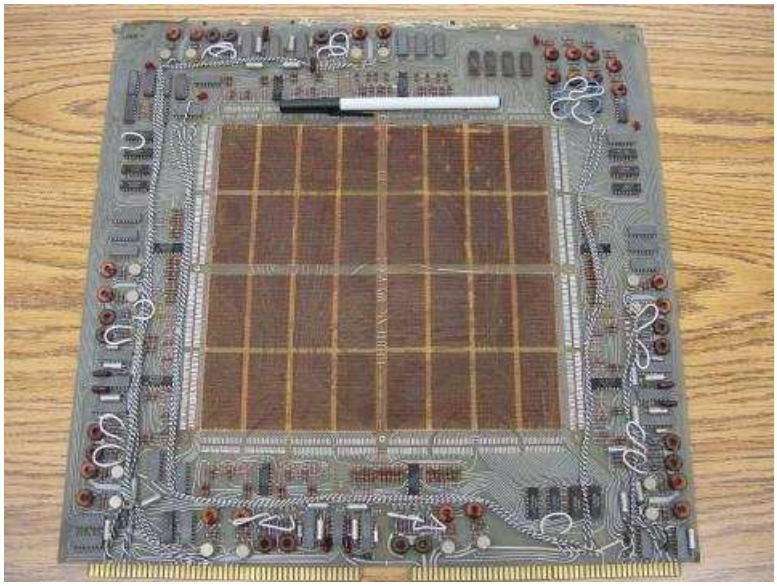
Writing to individual cores is simple, but reading from them is not. The solution developed for reading core states consisted of a single "read" wire threaded through *every* core in the grid. Unlike the row and column wires, the read wire was not energized by any source, but instead connected to the input of an amplifier built to sense a voltage pulse resulting from the sudden change[1] in magnetic flux at a core. So, to read a core, what we do is try *writing* to it and monitor the sense wire to see if anything changed. If we try to write a 1 state and nothing changed, then that bit must have already been a 1; if we try to write a 1 state and sense a change-of-state by means of a voltage pulse across the length of the read wire, then that bit must have been a 0. The memory array's peripheral

[1] This is an application of Faraday's Law of Electromagnetic Induction, $V = N\frac{d\Phi}{dt}$, where ϕ is magnetic flux, related to flux density (B) by cross-sectional area.

circuitry would then have to go back and re-write the original state following a change-of-state, lest the data become corrupted. This process is known as a *destructive read*, and it was an interesting idiosyncrasy of magnetic core memory.

Despite this unique behavior, core memory was once extremely popular for computers. It had no moving parts, it was much faster than any tape or disk-based memory, and most favorably was non-volatile[2].

The following photograph shows a core memory board taken from a 1970's era Data General "Nova" minicomputer. A white ball-point pen with a black cap is placed near the top of the board for size reference. The magnetic cores themselves comprise the rust-colored rectangular areas on the board, with column and row drive circuitry as well as sense circuitry occupying the periphery of the board:

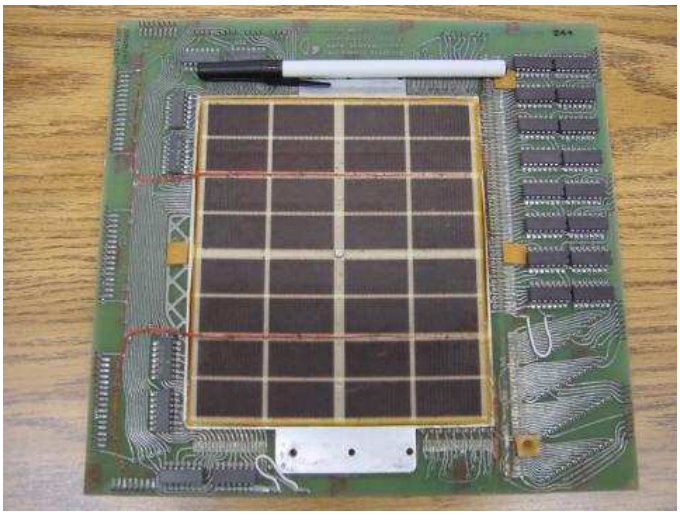


This large printed circuit board represented only *4 kilobytes* of memory, a paltry amount of digital storage by modern standards.

[2]This meant that any computer using core memory in lieu of static or dynamic RAM would enjoy the power of resuming exactly where it left off following a momentary interruption in power!

In this next photograph we see a close-up photograph of the same core memory board. The same ball-point pen remains in the photograph as a size reference. If you look closely, you can see the individual magnetic cores with fine-gauge wires threaded through their centers, the cores themselves tilted diagonally to the vertical column wires and horizontal row wires:

A more "modern" version of core memory appears in this next photograph, using much smaller ferrite rings to achieve greater packaging density and therefore greater memory capacity per unit board area:



This board sports double the memory of the previous core memory board: *8 kilobytes* instead of 4 kbytes, and it does so using a smaller board size as evidenced by the same ball-point pen.

Next we have a close-up photograph of this denser core memory array. My camera's resolution is insufficient to show a single core's shape or orientation as a result of their much smaller size:

4.2 Early magnetic hard disk drives

Any flat surface consisting of ferrite pieces has the ability to be selectively magnetized (i.e. some areas magnetized in one polarity and other areas in the opposite polarity), and such a magnetized surface when passed by a wire coil will induce voltage pulses in that coil with the passing of alternately-polarized areas. This is the basis of both magnetic tape and magnetic disk storage.

Modern magnetic disks use such fine-grained ferrite on their surface, that the wire coils reading and writing data must hover extremely close to the spinning disk's surface, usually closer than the diameter of a grain of dust! For this reason, modern magnetic disk drives are completely *sealed* units assembled in *clean-room* manufacturing facilities to prevent the inclusion of dust within the housing containing these moving parts. However, legacy hard disk drives were crude by comparison, and their disks could be removed for service or even archival in any reasonably clean room.

The following photograph shows such a magnetic disk removed from its drive mechanism, a ball-point pen situated at one edge for size reference. The rust-colored disk is the magnetic surface, while the blue plastic around the periphery is the rim of the protective cover permitting the disk to be taken out of the mechanical drive unit and held in safe keeping:

A complementary blue plastic cover (not shown) fit over the disk, enclosing the entire assembly. This allowed the disk to be safely placed on a shelf, as one might shelve a book, for later re-use.

A common storage capacity for a hard disk of this size was 30 megabytes per side, since the disk had two sides which could be used as media. Some referred to these disks as *Winchester* hard disks, since their 30/30 capacity ratings (30 Mbytes on one side and 30 Mbytes on the other) was reminiscent of the legendary 30/30 Winchester-brand rifle cartridge.

By contrast, a more modern hard disk drive (340 megabytes, still small by modern standards) is shown in this next photograph:

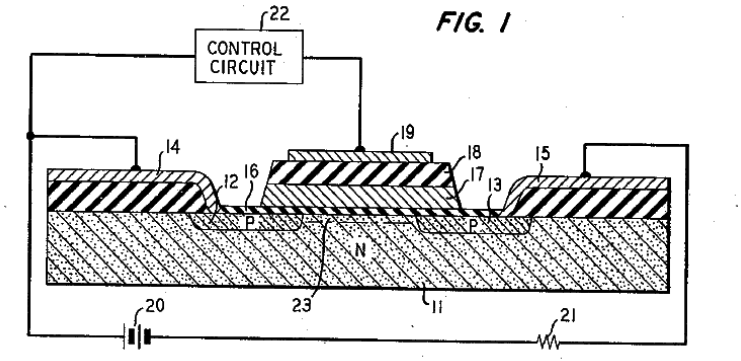
The read/write coil is located at the tip of the lever projecting out over the disk's surface, and is moved in an arc across that spinning surface by means of a small servo motor. In the legacy "Winchester" disk shown previously, the read/write coil and servomechanism was part of a separate unit from the movable disk. Here, the disk is not removable, and all appear as part of a single unit.

Incidentally, the removal of the cover from this hard disk unit necessary for photographing the disk *ruined* it. Unlike the "Winchester" disk shown earlier, modern hard drives must remain sealed throughout their operating lives due to the necessity of a dust-free environment.

4.3 Floating-gate FET transistor patent

A Korean-born researcher working for Bell Laboratories named Dawon Kahng applied for a patent in June of 1967 entitled "Field Effect Semiconductor Apparatus With Memory Involving Entrapment Of Charge Carriers", the patent (number 3,500,142) being granted in March three years later. In Kahng's patent we find a clear description of a special type of MOSFET intended as a digital memory cell. This special MOSFET revolutionized digital memory storage technology, paving the way for "flash" memory, microcontrollers with extensive nonvolatile memory banks, FPGAs, and other modern digital devices.

Figure 1A in Kahng's patent shows a cross-section of the FET:



What makes this MOSFET unique is a "floating" gate consisting of an electrically isolated metal layer (17) sandwiched in between two layers of insulating oxide (16 and 18). Kahng's invention became the foundation of EPROM memory technology, including later "flash" memory technologies. The first paragraph of Kahng's patent tersely describes his invention:

> ABSTRACT OF THE DISCLOSURE – A field effect transistor is provided with a gate electrode assembly comprising a sandwich of a metallic layer between two insulating films of appropriate characteristics for the entrapment of charge carriers in the metallic layer. Such entrapment provides memory whereby an induced electric field can be maintained in the semiconductive element even after the field inducing force is removed. [page 3]

Kahng begins with a brief explanation of digital memory technology including a review of a magnetic-based memory element which his invention supersedes:

> This invention relates to semiconductive apparatus which is characterized by relatively long memory, i.e., apparatus which will persist in a certain induced state for a useful period of time even after the inducing force is removed whereby there is provided for such period a record of the characteristics of the inducing force.
>
> In computers and related apparatus there exists a need for memory elements in which information can be stored temporarily, while remaining accessible for reading, and then readily erased or modified.
>
> Hitherto there have been suggested semiconductive devices having memory in which the memory was achieved by the provision of a ferroelectric element in contact with

> the semiconductive wafer such that the direction of the remanent polarization of the ferroelectric element controlled the state of the device. Typical of such devices are those described in U.S. Patent 2,791,760. An advantage of these devices is that the memory is independent of any sustaining currents so that even if there is a temporary loss of power the stored information is not lost. A serious difficulty that has hampered the exploitation of devices of this kind is that presently available ferroelectrics tend to suffer mechanical damage with use and after suffering such damage are of limited reliability. Also, the inherent speed capability is limited by the domain motion which is relatively slow, and so it is difficult to realize very fast switching.
>
> To this end, the invention provides a semiconductive device with memory, in which the need for a ferroelectric element is obviated by a novel gate assembly which makes use of trapped charge carriers. In particular, viewed from one aspect, the gate assembly typically will comprise a pair of insulating layers sandwiching a metallic layer in which the charges are trapped. Viewed from another aspect, the gate assembly will include a tunnel sandwich diode formed by two conductive element sandwiching an insulating layer sufficiently thin that electron transport or tunneling therethrough is realized under control of applied signal information. [page 3]

Here Kahng first mentions the principle by which his memory cell functions: *tunneling* of electric charge carriers through a thin region of insulating material. This is a *quantum* effect, inexplicable from the perspective of classical physics.

Next, Kahng describes the construction of this special transistor:

> In a typical insulated gate field effect transistor embodiment of the invention, the semiconductive element is a silicon wafer the bulk of which is weakly n-type and which includes a pair of spaced p-type surface zones to which are connected the source and drain electrodes, respectively. The surface portion of the element intermediate such p-type zones is covered with a relatively thin layer of silicon oxide which together with a relatively thick outer layer of zirconium oxide sandwiches a layer of zirconium there between which serves as a floating gate.
>
> A metallic layer covers the zirconium oxide to serve as the outer gate electrode. In operation, a positive voltage pulse applied to the outer gate electrode causes the entrapment of electrons in the zirconium layer for a relatively long time, which results in a p-type skin persisting underneath the silicon oxide and a low resistance path between the source and drain electrodes. The application of a negative pulse instead results in leaving the portion of the element underlying the silicon oxide n-type whereby the resistance between the source and drain electrodes is high.
>
> A variety of embodiments will be described hereinafter including ones in which light may be employed for releasing charges trapped in the floating gate and thereby affecting the resistance between the source and drain electrodes. [page 3]

Features evident in Figure 1 (shown at the beginning of this section) are explained in the following paragraphs:

> With reference now to the drawing, in the memory cell shown in Fig. 1, a monocrystalline semiconductive element, the bulk 11 of which is weakly n-type, also includes spaced, localized, p-type surface zones, 12, 13. Electrodes 14 and 15 make low resistance connections to zones 12 and 13 and serve as the source and drain electrodes, respectively. Overlying the surface between zones 12 and 13 is the gate assembly comprising a relatively thin insulating layer 16 contiguous to the surface, an inner metallic layer 17, a relatively thick outer insulating layer 18, and an outer metallic layer 19. Layer 18 is also chosen to have a higher dielectric constant than layer 16. As seen, the metallic layer 17 is sandwiched between layers 16 and 18, and is insulated from the outer layer 19 and the semiconductive element. It will be convenient to refer to inner layer 17 as the floating gate and outer layer 19 as the terminal gate. It will also be apparent that insulating layer 16 is sandwiched between the semiconductive element 11 and the metal layer 17 to form a "tunnel sandwich diode" because as will be explained hereinafter the operation depends on electron tunneling through layer 16 under control of signal information.
>
> In one embodiment constructed, the element 11 was primarily n-type silicon with a resistivity of about 1 ohm-centimeter. The layer 16 was about 50 angstroms thick and of silicon dioxide thermally grown, the layer 17 was of zirconium and about 1000 angstroms thick, the layer 18 was of zirconium oxide about 1000 angstroms thick, and the layer 19 and the electrodes 14 and 15 were of aluminum. The p-type zones 12 and 13 were spaced apart about 0.5 mil.
>
> A voltage source 20 and load 21 are connected serially between source and drain electrodes 14 and 15. The control voltage source 22 is connected between the source electrode 14 and the terminal gate electrode 19.
>
> For the specific design previously described, a voltage pulse of about fifty volts with a width of 0.5 microsecond was used to store 5×10^{12} electrons per square centimeter on the floating gate, which was adequate to cause the desired silicon surface layer inversion.
>
> In operation, depending on the polarity of the most [page 3]
>
> recent control voltage pulse provided by source 22, the surface portion 23 of the element extending between zones 12 and 13 and underlying the insulating layer 16 is either effectively n- or p-type. In the former case, a high resistance is presented, due to the presence of the reverse-biased p-n junction, to the flow of current in the load and little current flows. In the latter case, a low resistance is presented and a relatively large current flows. Accordingly, bistable operation is permitted. [page 4]

Following this physical description of the transistor, Kahng proceeds to describe its operation from a more theoretical perspective. For this he makes use of energy diagrams, also provided in the patent as figures 2A, 2B, and 2C.

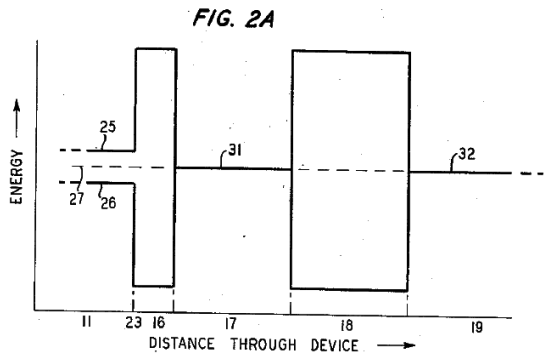
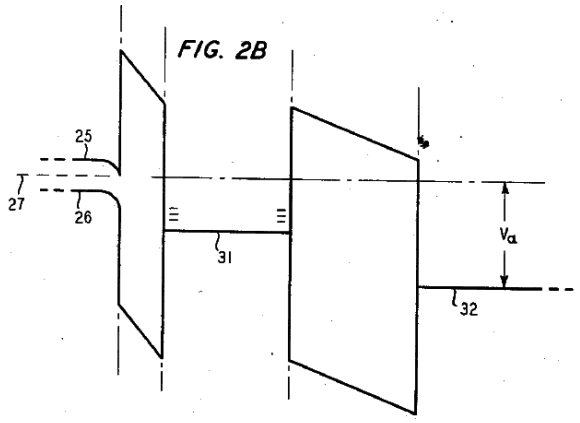
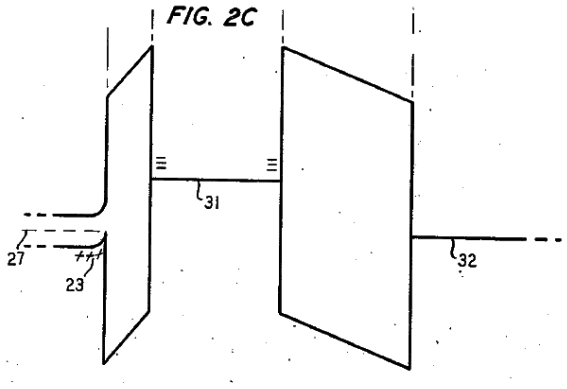


Fig. 2A shows the energy level diagram of the system in the initial quiescent state before any voltage pulse has been applied to electrode 19. In particular, solid line 25 shows the bottom of the conduction band and solid line 26 the top of the valence band as one passes from the n-type bulk of the semiconductive element through surface layer 23 and layers 16, 17, 18 and 19 of the gate assembly. In the insulating layers 16 and 18, lines 25 and 26 are widely spaced, corresponding to the large forbidden energy gap in insulators. In the semiconductor, lines 25 and 26 are relatively closer, corresponding to the narrower band energy gap. The Fermi level is depicted by the broken line 27, which coincides with solid lines 31 and 32 for the metal layers. Thermal equilibrium requires that the Fermi level be at the same energy throughout the system. [page 4]



The presence of a positive voltage V_a on electrode 19 results in the energy level diagram shown in Fig. 2B. The thickness of the insulating layers 16 and 18 and the magnitude of the applied voltage V_a are so chosen that there is a transport of electrons from the semiconductor through layer 16 into metallic layer 17 by a field controlled electron transport mechanism, such as tunneling or internal tunnel-hopping, but there is insignificant exodus of such electrons which are trapped in layer 17. The excess electrons in layer 17 are shown depicted in the usual fashion as negative charges. This condition is met by choosing insulating layers 16 and 18 such that the ratio of dielectric constant ϵ_1 / ϵ_2 (where ϵ_1 and ϵ_2 are the dielectric constants of the two layers, respectively) is small and/or the barrier height into layer 16 is smaller than into layer 18. Additionally, to insure trapping it is important that the thickness of the floating gate layer 17 is thicker than the hot electron range so that the transported electrons are substantially at the Fermi-level 31 of the layer before reaching insulating layer 18. [page 4]



With the removal of the applied bias, the energy level of the system is as depicted in Fig. 2C. The electric fields in the insulating layers 16 and 18 are now too low to permit carrier transport across them; consequently the electrons remain trapped in floating gate 17. In particular, the trapped electrons, which tend to distribute themselves uniformly throughout the layer 17 will result in the layer effectively being at a negative potential. This is depicted in Fig. 2C by the elevation of line 31 relative to that of lines 27 and 32 whose level remains fixed. Accordingly, there will be drawn into underlying surface layer 23 of the semiconductive element positive charges to compensate for the excess electrons in the contiguous metal as shown, and such positive charges will effectively make surface layer 23 p-type. As a consequence, a barrier-free low-resistance path is provided between sources 12 and 13 and significant current flow therebetween and through the load 21 under the influence of source 20 is facilitated. So long as excess electrons are trapped in layer 17, this condition persists. Accordingly, the device remembers the application of a positive pulse to electrode 19 even after the termination of the pulse. `[page 4]`

After this brief discussion of energy band diagrams and electron tunneling, Kahng returns to a more operational description of the process where he elaborates on stored charge quantity and erasure methods:

To store a given amount of charge, one can either increase the applied voltage or increase the charging time by increasing the pulse width, or both, for a given gate structure. Moreover, the lower the barrier height associated with insulating layer 16 the lower the applied voltage needed to achieve a given trapped charge. For example by use of silicon nitride instead of silicon dioxide for layer 16 one can operate with significantly lower applied voltages.

There is a charge loss which is controlled by the dielectric relaxation time of the sandwich structure, and this loss readily can be made quite low. If very long dielectric relaxation times are desired, it will be advantageous to employ organic insulators which typically have longer relaxation times than inorganic insulators.

The trapped electrons can be quickly removed from the metallic layer by the reverse process of applying an appropriate negative pulse to electrode 19. The argument employed above can be used in analogous fashion to establish that such action can

> reduce the number of electrons in the layer 17, whereby there is restored to n-type the underlying surface portion 23. As a consequence, the resistance between zones 12 and 13 becomes high because of the reverse-biased p-n junction in this path and, accordingly, the current in the load is low.
>
> To discharge completely the floating gate, the voltage applied to the terminal gate should be about equal in magnitude and duration although opposite in polarity to the voltage applied for charging.
>
> It should also be apparent that a net positive charge (loss of electrons) can be stored in the floating gate if the discharging voltage applied to the outer gate is appropriately chosen in polarity, magnitude and duration, whereby surface portion 23 is made more strongly n-type.
>
> It should also be apparent that the transport of electrons through the thin insulating layer 16 may be facilitated by irradiation of the thin layer and the contiguous semiconductor with light or other radiant energy coincident with the application of the biasing pulse to increase the temperature or energy of the electrons in the semiconductor surface, effectively lowering the barrier for transport into the floating gate. [page 4]

Toward the end of the patent's text, Kahng explains how these memory cells would likely be organized into memory arrays of useful size:

> It should also be apparent that the invention will find principal utility in arrangement which incorporates a plurality of elements in large matrix arrays to provide increased memory. Typically in such arrays to facilitate accessing, the individual elements are disposed in two dimensional arrays in a coordinate fashion so that by selection of X and Y coordinates an individual element can be selected. One accessing technique feasible would involve: for writing pulsing and appropriate X gate line simultaneously with the appropriate Y source line to energize the element at the corresponding XY coordinate; for reading pulsing the appropriate X source line simultaneously with the appropriate Y drain line to detect the state of the element at the XY coordinate. [page 6]

Chapter 5

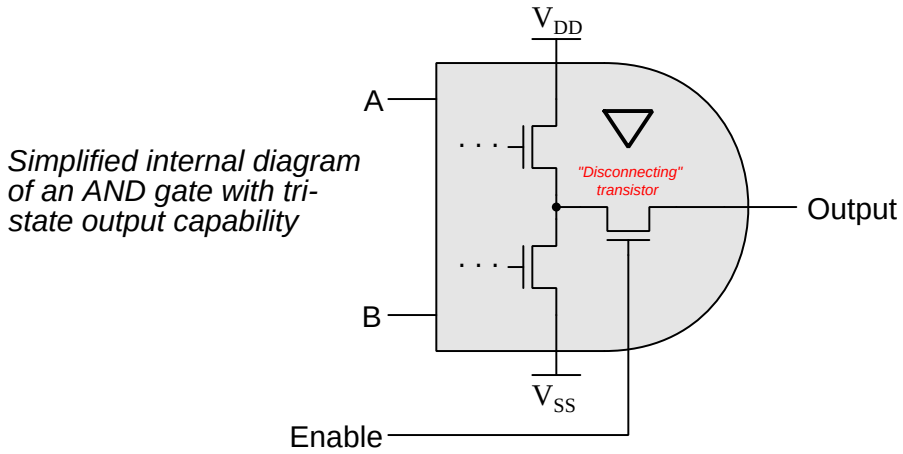
Derivations and Technical References

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

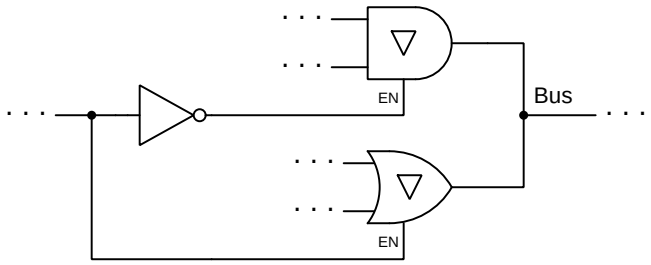
5.1 Tri-state logic gate outputs

When multiple logic devices must assert high or low logic states to a commonly-shared line, it becomes necessary to "disconnect" all but one of those device outputs from that line at any given time, or else multiple devices may attempt to sink *and* source current simultaneously. A practical method to arbitrate access to a common line is to equip those logic devices' outputs with a feature called *tri-state* capability. A logic gate with a tri-state output has an additional input line controlling a transistor "disconnect" between the internal totem-pole transistor stage and the output terminal. When disabled, the gate assumes a *high-impedance* or *high-Z* mode whereby the output terminal "floats" and can no longer source or sink current. When enabled, a tri-state gate behaves like any other logic gate of its functional type.

An inverted triangle symbol denotes a logic gate capable of tri-state operation. An additional "enable" line on the gate is another indication:



In the following example, an AND gate and an OR gate both drive signals to a common line called a *bus*. Their respective enable lines are controlled by a digital signal and its complement, to ensure only one of the two gates will ever be permitted to drive a high or low signal to the bus:



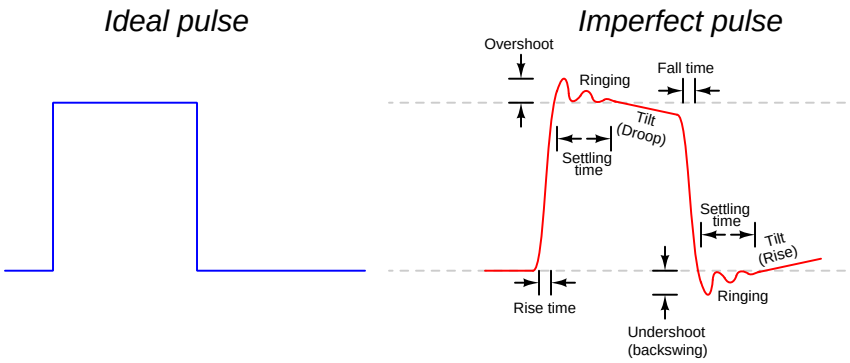
Digital electronic computer designs are largely *bus-oriented*, which means they contain many such "bus" lines shared one at a time by various logic gate outputs, much as a radio channel may be used by one person (talking) at any given time. This is why the potential problem of multiple logic

gates "fighting" one another with opposing logic states is often referred to as *bus contention*: in a poorly-designed system the gates will literally contend with one another for control of the common bus. In complex digital systems where several or more gates connect to a common bus, the strategy employed to select just one of those gates at a time is called *bus arbitration*.

This is the same fundamental problem (and solution!) for having multiple digital memory elements output data to shared lines, as well as having a single memory element selectively write to or read from a common "I/O" line.

5.2 Digital pulse criteria

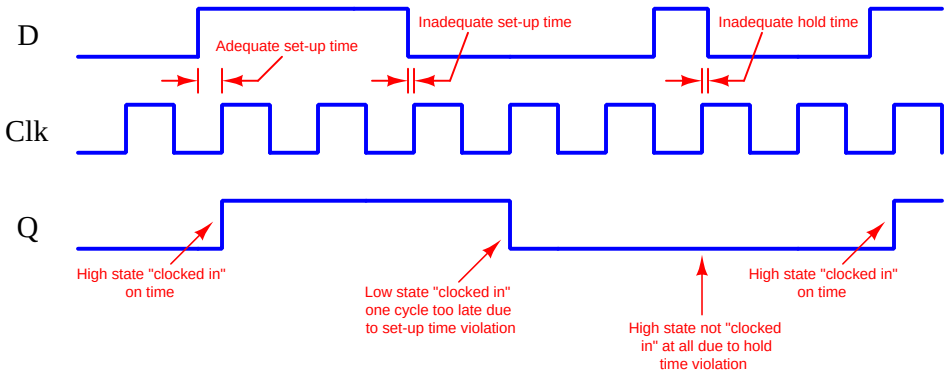
Ideal "pulse" signals have infinitely-steep rise and fall times, level "high" and "low" states well within the specified voltage ranges, and no other imperfections or artifacts. Real pulses always deviate from ideal, often in multiple ways:



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!

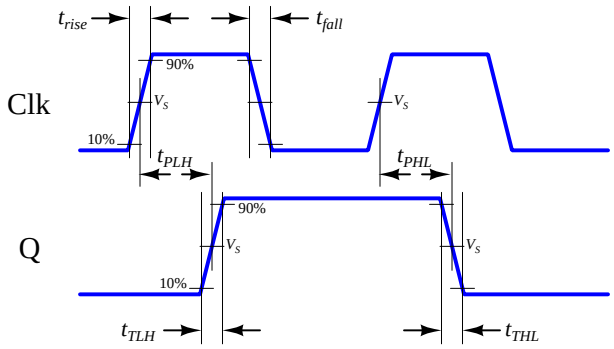
Ringing is caused by *resonance* occurring between parasitic capacitance and parasitic inductance, those two phenomena naturally exchanging energy back and forth with each other to produce the oscillatory waves we call "ringing". Overshoot and undershoot are also the result stored energy within these 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:



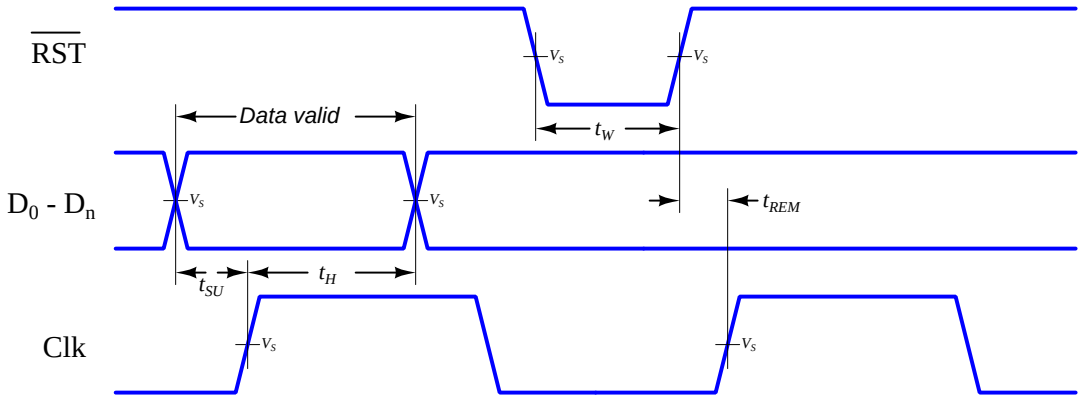
Each of the labels found in this diagram is defined as follows:

- 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 6

Animations

Some concepts are much easier to grasp when seen in *action*. A simple yet effective form of animation suitable to an electronic document such as this is a "flip-book" animation where a set of pages in the document show successive frames of a simple animation. Such "flip-book" animations are designed to be viewed by paging forward (and/or back) with the document-reading software application, watching it frame-by-frame. Unlike video which may be difficult to pause at certain moments, "flip-book" animations lend themselves very well to individual frame viewing.

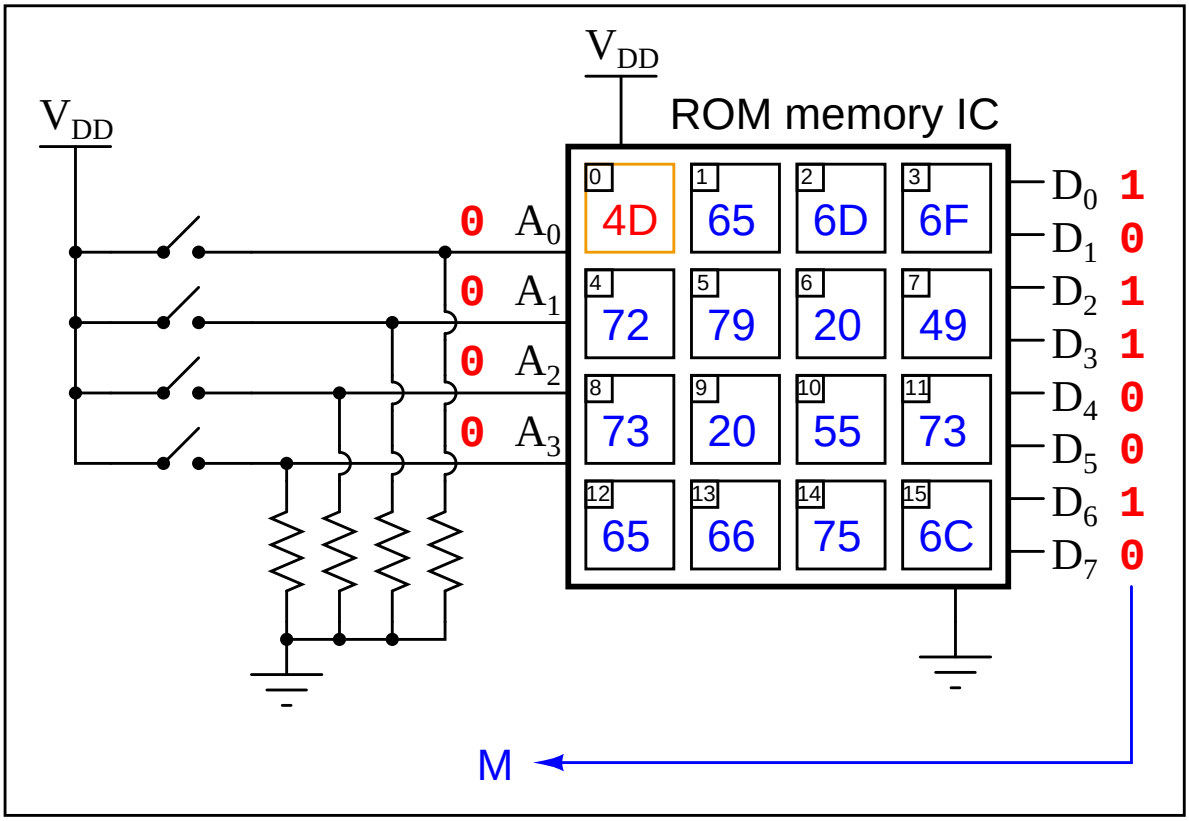
6.1 Animation of 16 × 8 ROM

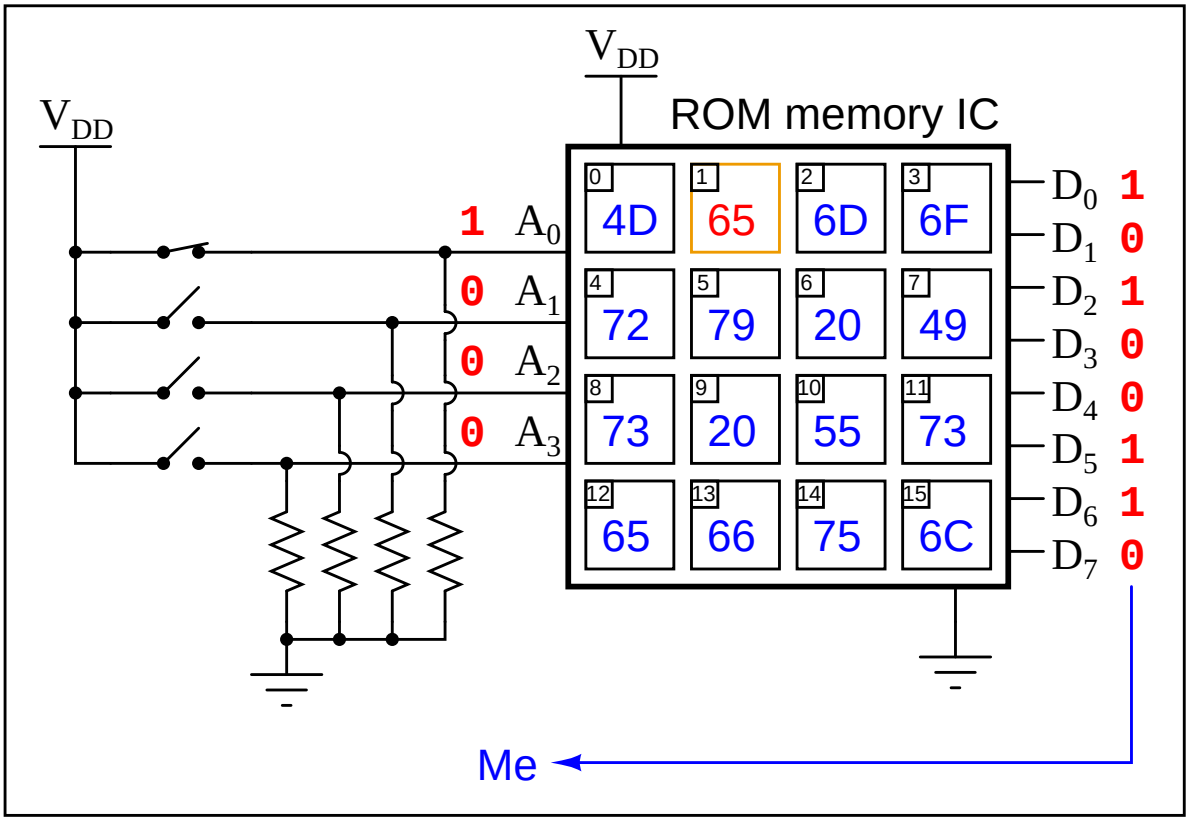
The following animation shows a 16 × 8 ROM[1] integrated circuit used to store ASCII-encoded data. The four address lines are sequenced in binary order, and with each new address sent to the ROM chip it outputs the eight-bit data word stored in that memory cell. The resulting English sentence is spelled out with each new letter read from the ROM chip.

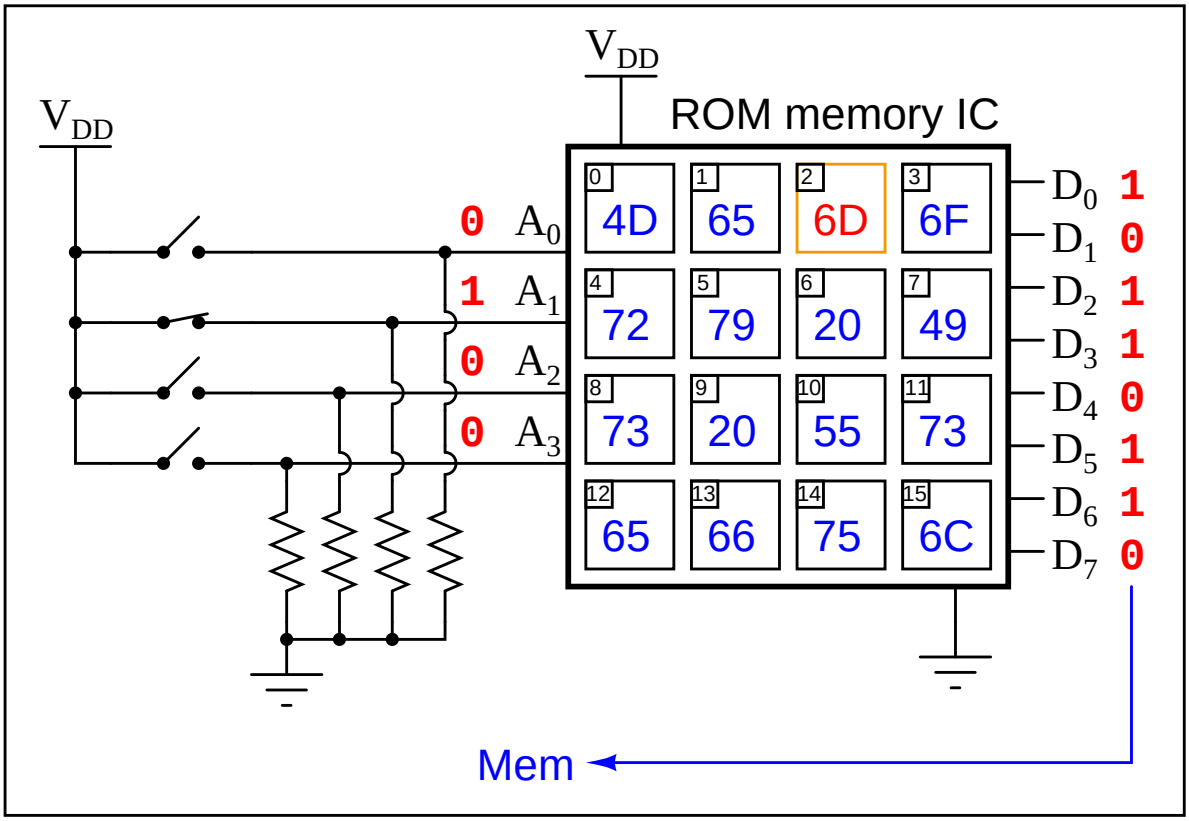
A memory map for this ROM is shown here in standard "hex dump" format:

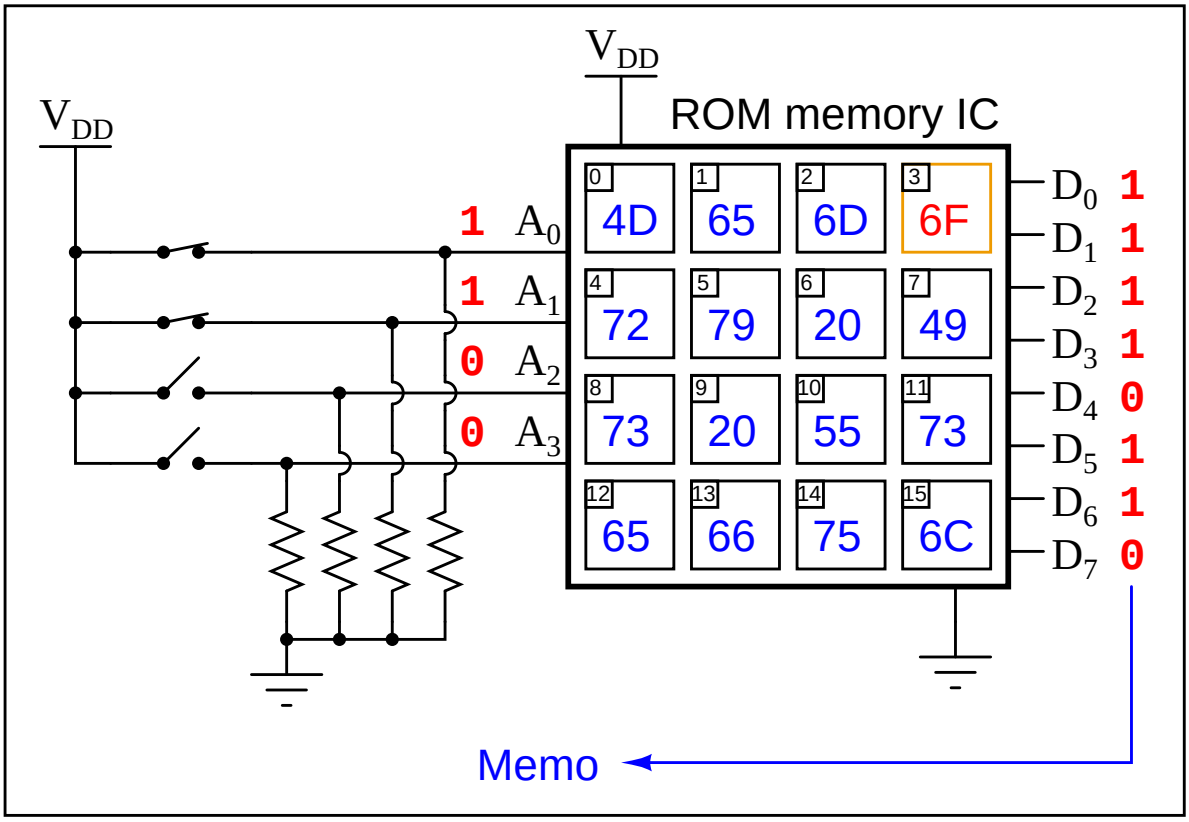
```
   0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
00 4D 65 6D 6F 72 79 20 49 73 20 55 73 65 66 75 6C
```

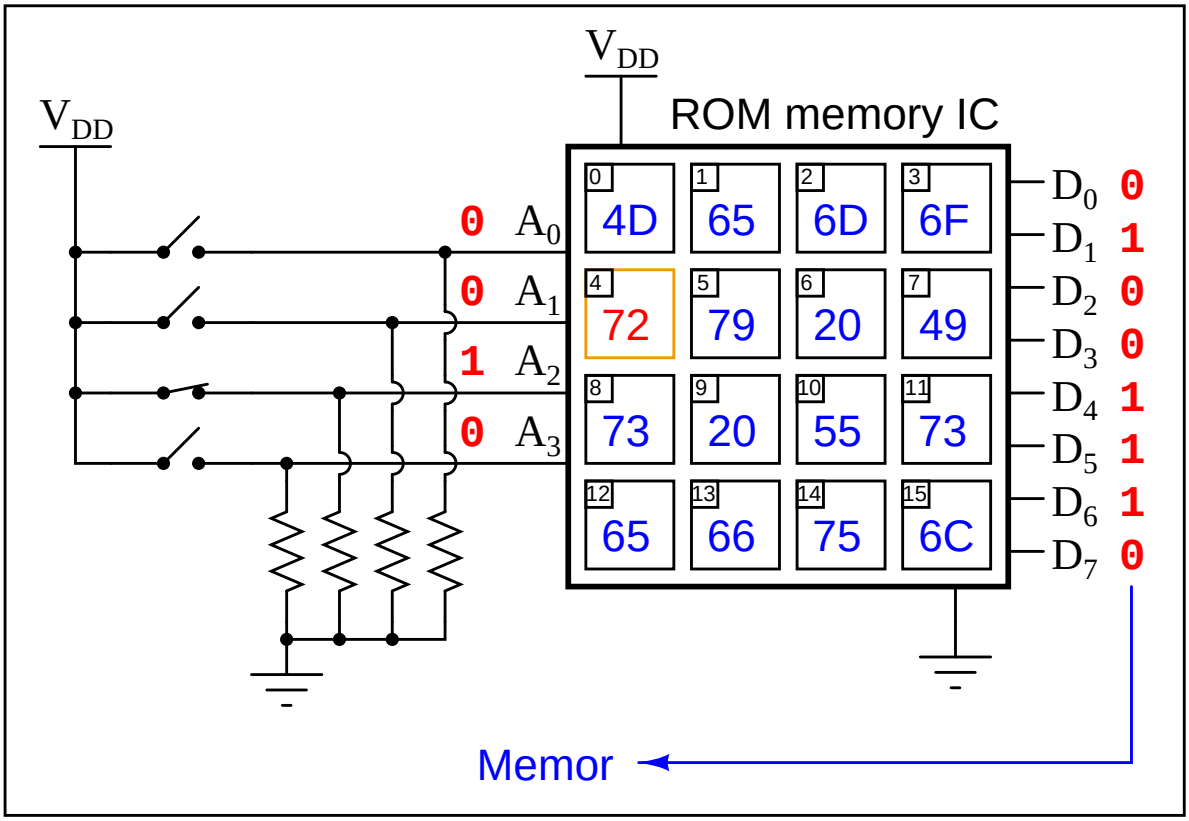
[1] 16 × 8 refers to *sixteen* addresses, each one storing an *eight*-bit word.

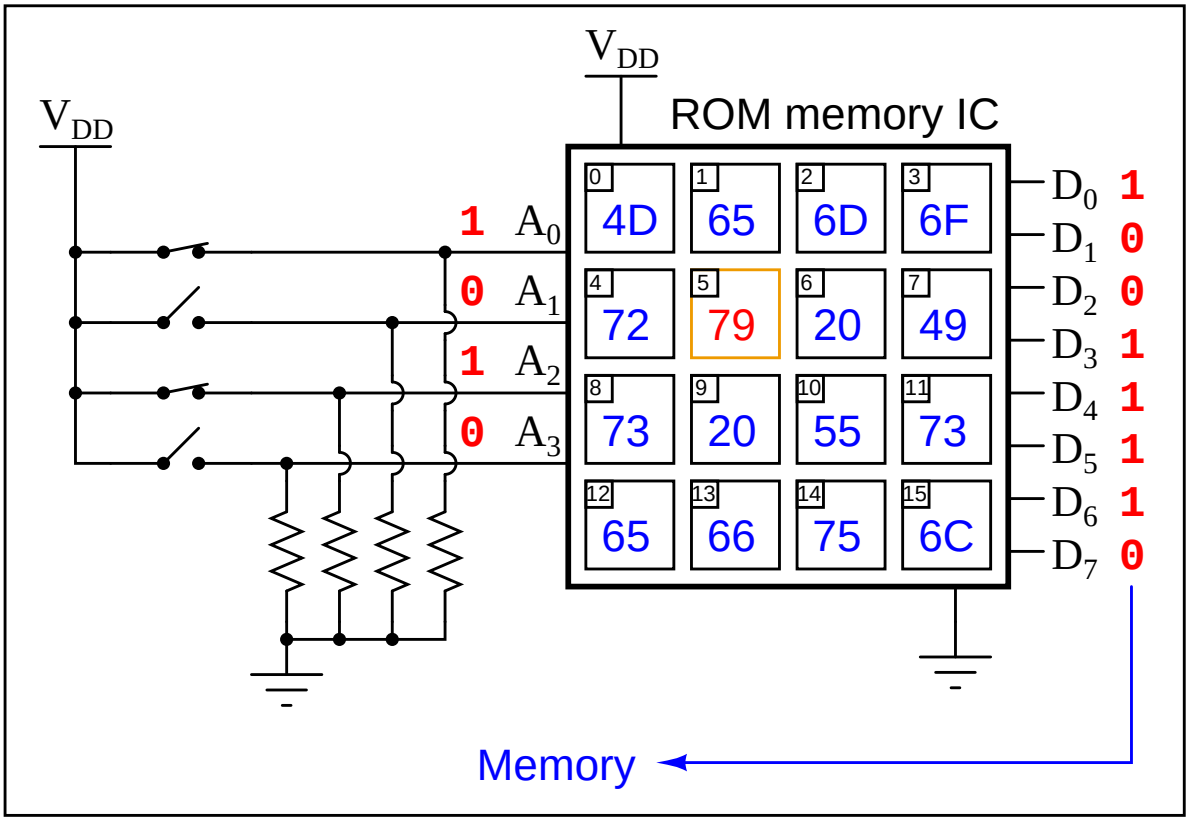

V_DD
V_DD
ROM memory IC
0 A_0
0 A_1
0 A_2
0 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 1
D_1 0
D_2 1
D_3 1
D_4 0
D_5 0
D_6 1
D_7 0
M

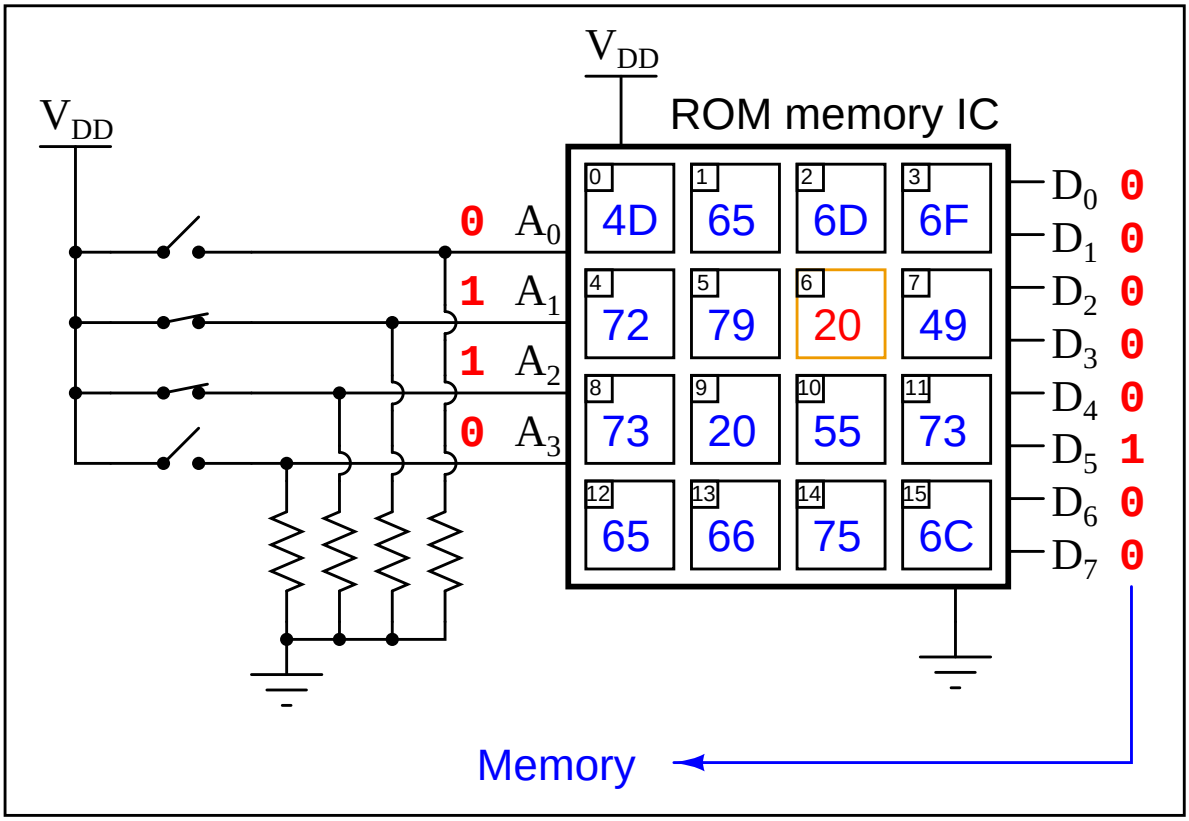

V_DD
V_DD
ROM memory IC
1 A_0
0 A_1
0 A_2
0 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 1
D_1 0
D_2 1
D_3 0
D_4 0
D_5 1
D_6 1
D_7 0
Me

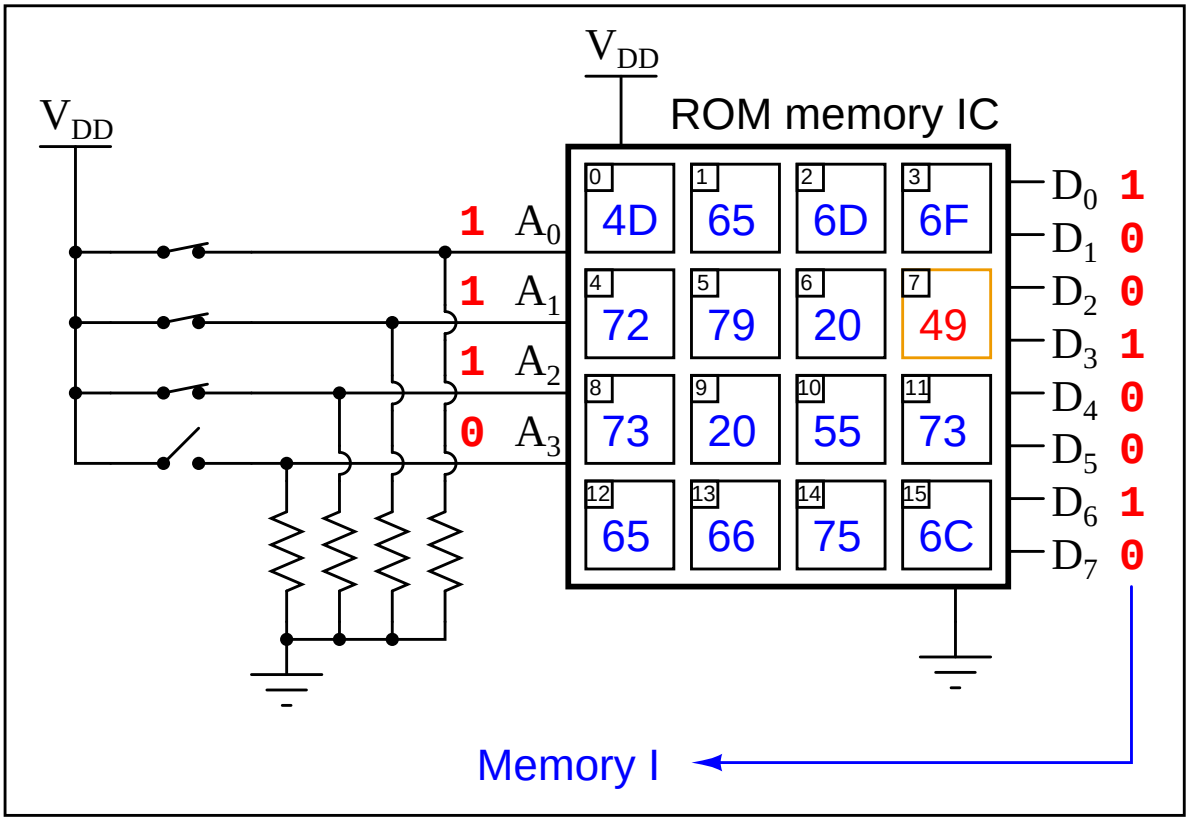

V_DD
V_DD
ROM memory IC
0 A_0
1 A_1
0 A_2
0 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 1
D_1 0
D_2 1
D_3 1
D_4 0
D_5 1
D_6 1
D_7 0
Mem

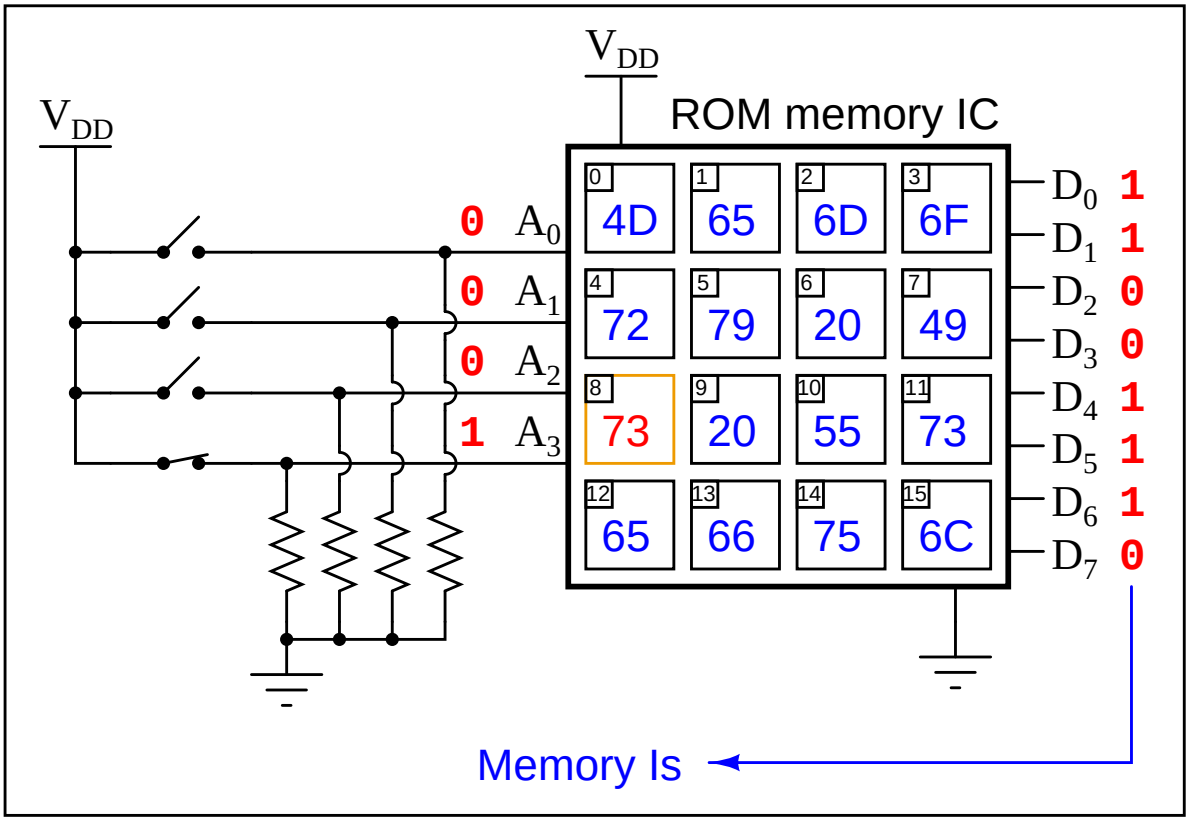

V_DD
V_DD
ROM memory IC
1 A_0
1 A_1
0 A_2
0 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 1
D_1 1
D_2 1
D_3 1
D_4 0
D_5 1
D_6 1
D_7 0
Memo

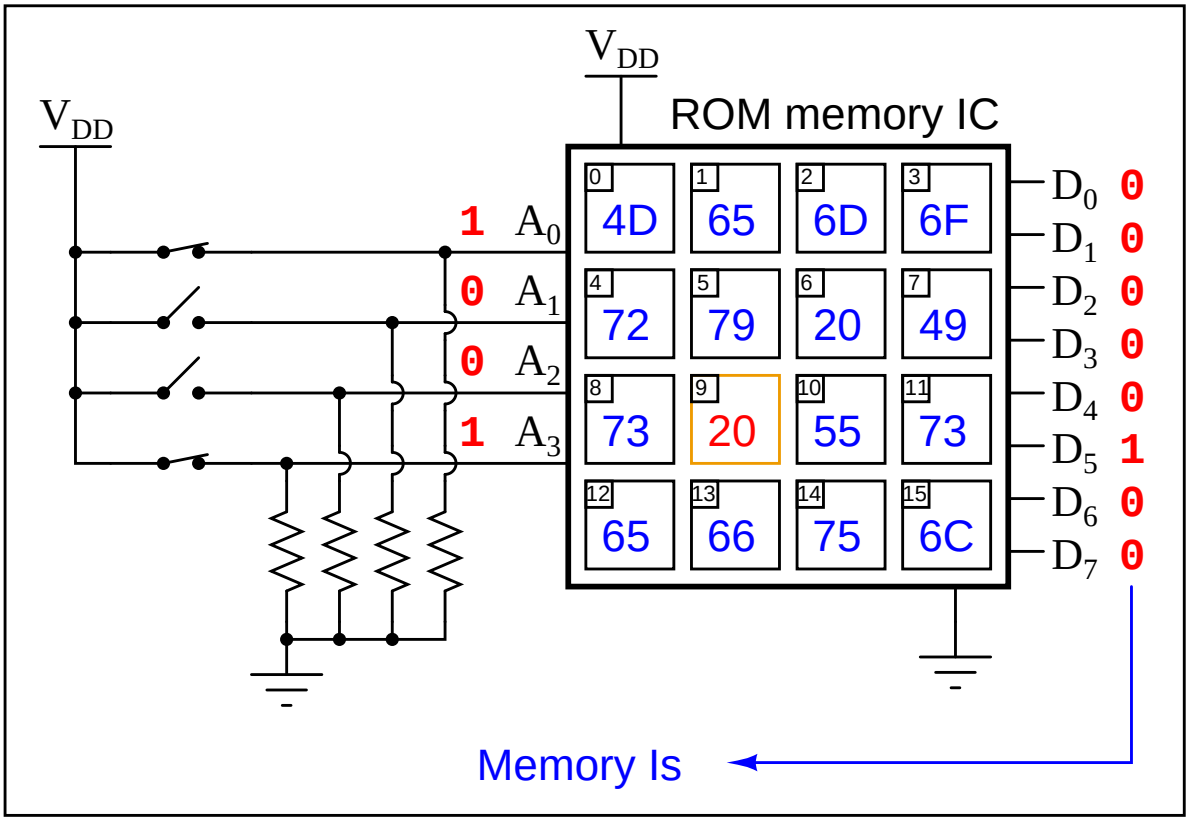

V_DD
V_DD
ROM memory IC
0 A_0
0 A_1
1 A_2
0 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 0
D_1 1
D_2 0
D_3 0
D_4 1
D_5 1
D_6 1
D_7 0
Memor

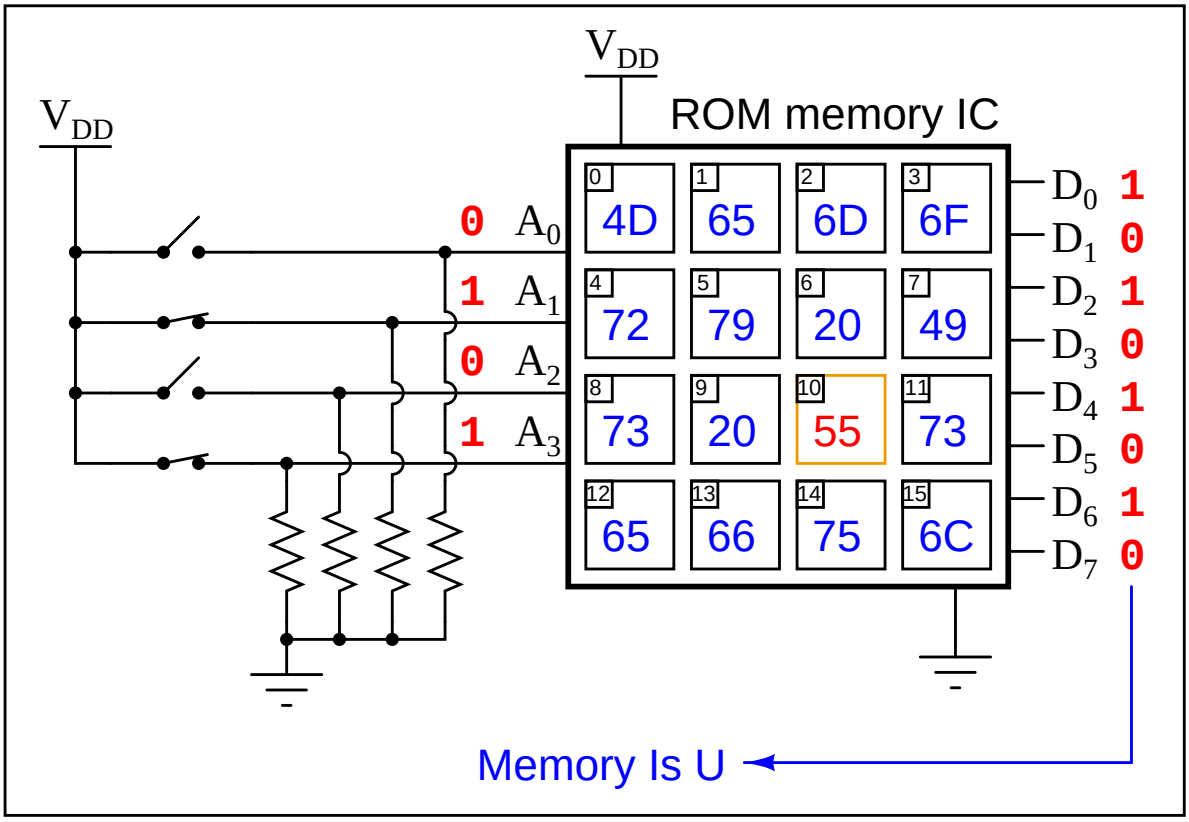

V_DD
V_DD
ROM memory IC
1 A_0
0 A_1
1 A_2
0 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 1
D_1 0
D_2 0
D_3 1
D_4 1
D_5 1
D_6 1
D_7 0
Memory

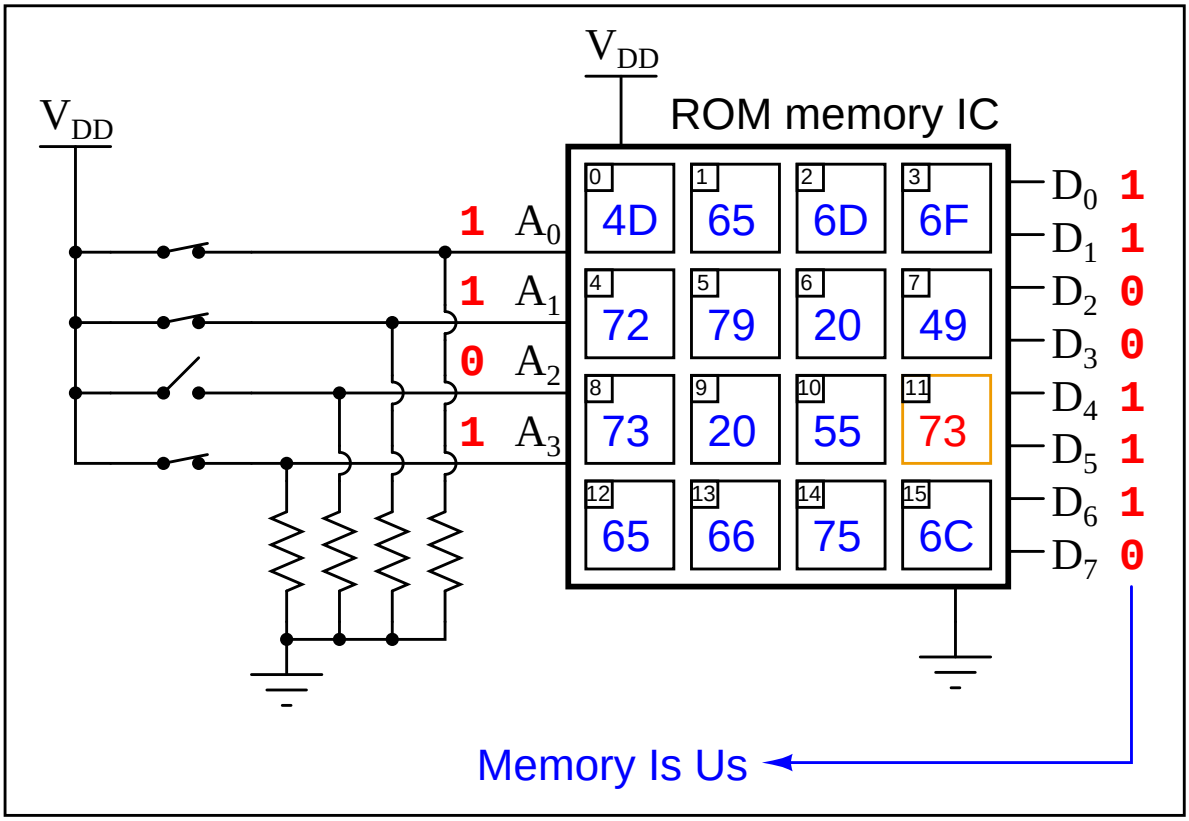

V_DD
V_DD
ROM memory IC
0 A_0
1 A_1
1 A_2
0 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 0
D_1 0
D_2 0
D_3 0
D_4 0
D_5 1
D_6 0
D_7 0
Memory

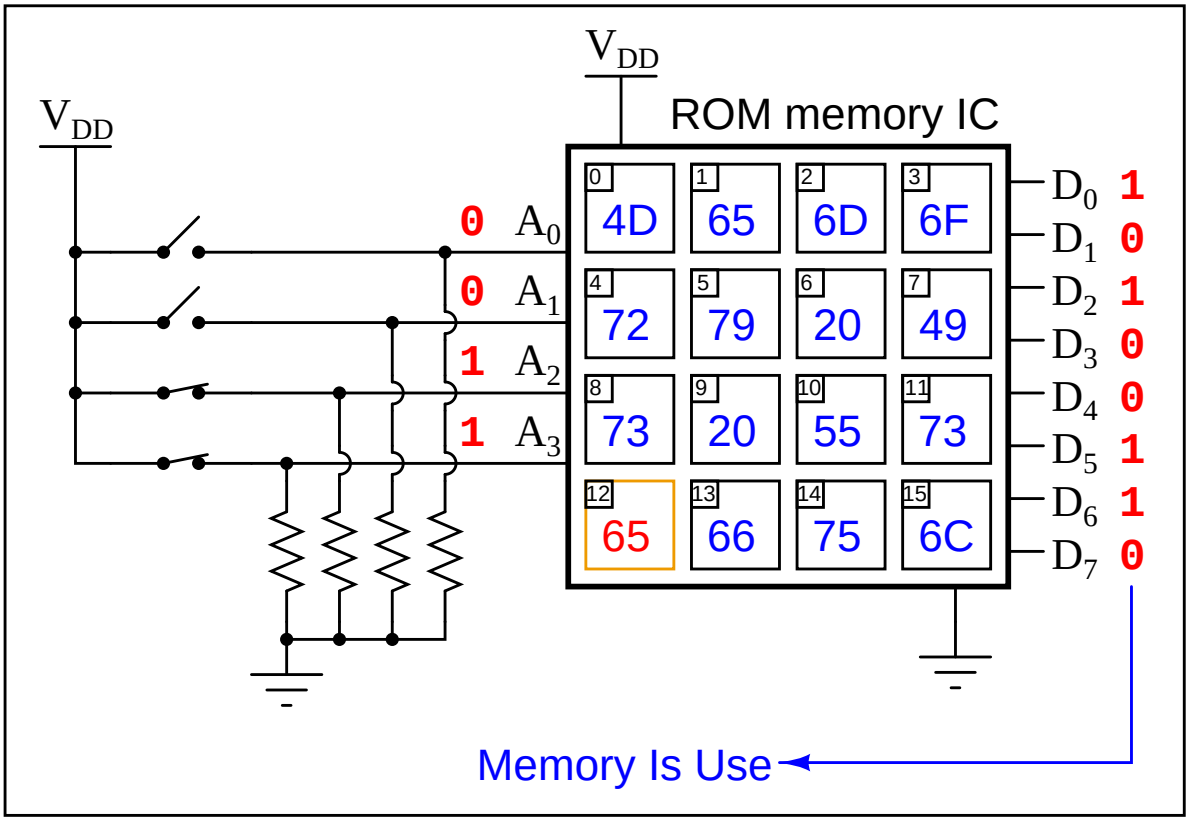

V_DD
V_DD
ROM memory IC
1 A_0
1 A_1
1 A_2
0 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 1
D_1 0
D_2 0
D_3 1
D_4 0
D_5 0
D_6 1
D_7 0
Memory I

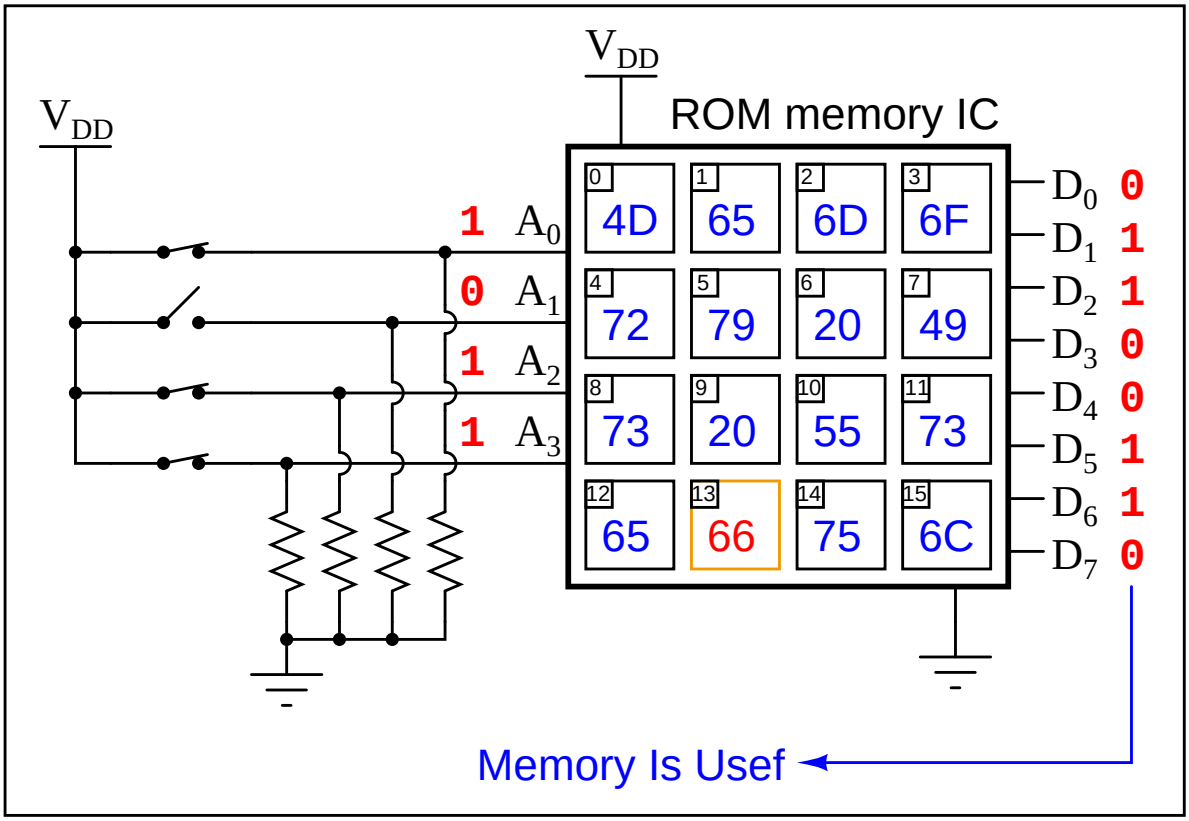

V_DD
V_DD
ROM memory IC
0 A_0
0 A_1
0 A_2
1 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 1
D_1 1
D_2 0
D_3 0
D_4 1
D_5 1
D_6 1
D_7 0
Memory Is

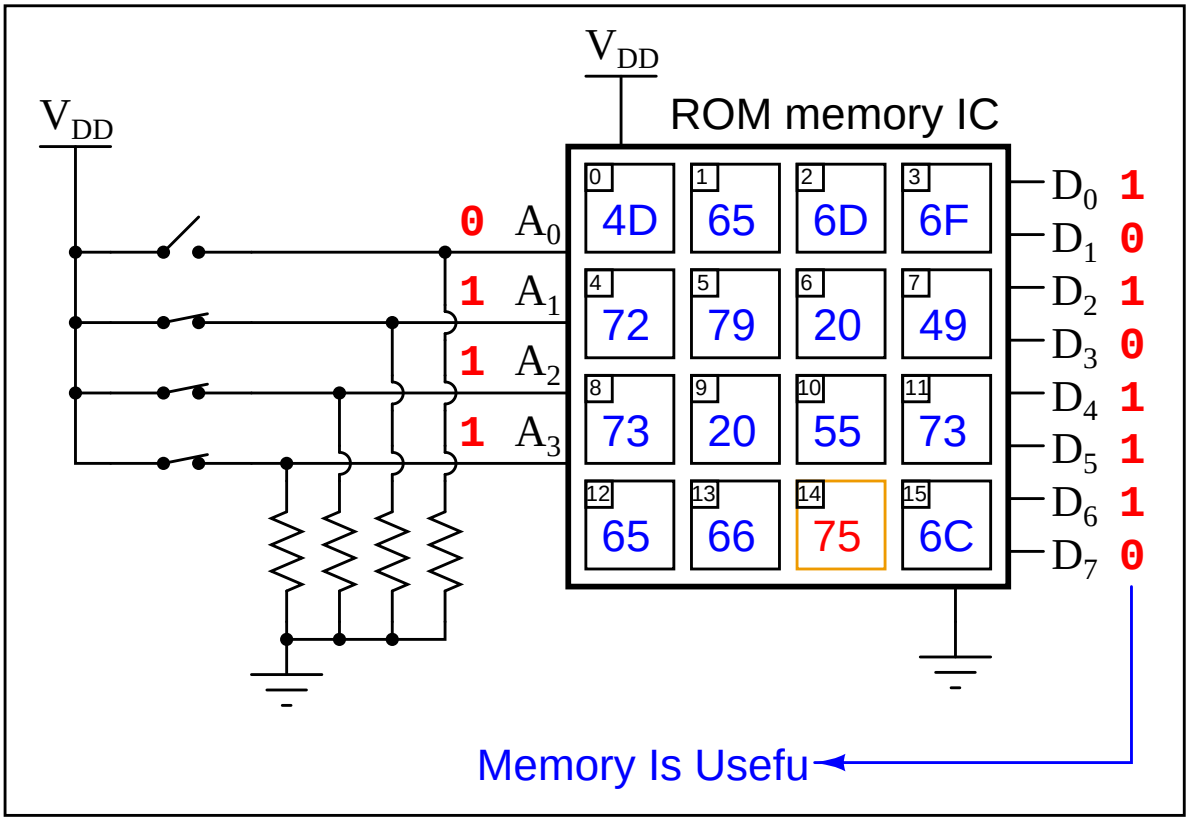

V_DD
V_DD
ROM memory IC
1 A_0
0 A_1
0 A_2
1 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 0
D_1 0
D_2 0
D_3 0
D_4 0
D_5 1
D_6 0
D_7 0
Memory Is

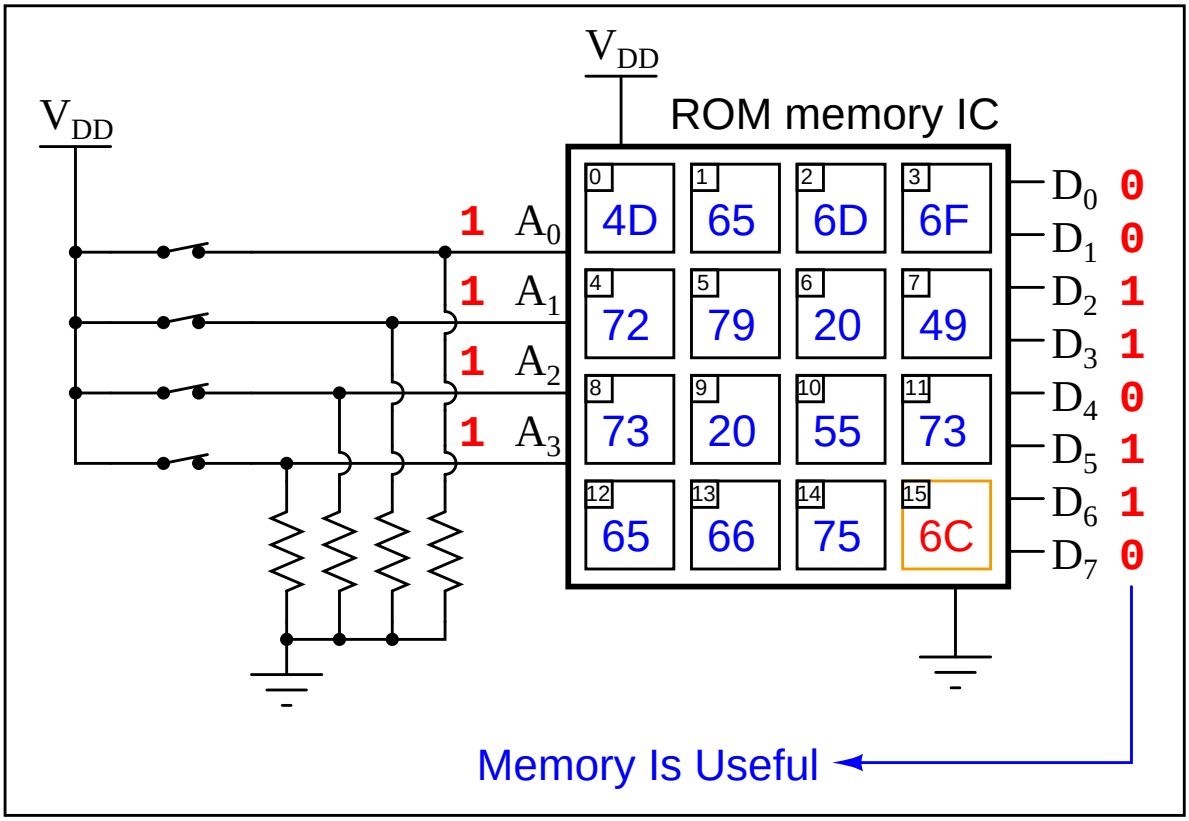

V_DD
V_DD
ROM memory IC
0 A_0
1 A_1
0 A_2
1 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 1
D_1 0
D_2 1
D_3 0
D_4 1
D_5 0
D_6 1
D_7 0
Memory Is U


V_DD
V_DD
ROM memory IC
1 A_0
1 A_1
0 A_2
1 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 1
D_1 1
D_2 0
D_3 0
D_4 1
D_5 1
D_6 1
D_7 0
Memory Is Us


V_DD
V_DD
ROM memory IC
0 A_0
0 A_1
1 A_2
1 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 1
D_1 0
D_2 1
D_3 0
D_4 0
D_5 1
D_6 1
D_7 0
Memory Is Use


V_DD
V_DD
ROM memory IC
1 A_0
0 A_1
1 A_2
1 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 0
D_1 1
D_2 1
D_3 0
D_4 0
D_5 1
D_6 1
D_7 0
Memory Is Usef


V_DD
V_DD
ROM memory IC
0 A_0
1 A_1
1 A_2
1 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 1
D_1 0
D_2 1
D_3 0
D_4 1
D_5 1
D_6 1
D_7 0
Memory Is Usefu


V_DD
V_DD
ROM memory IC
1 A_0
1 A_1
1 A_2
1 A_3
0 4D
1 65
2 6D
3 6F
4 72
5 79
6 20
7 49
8 73
9 20
10 55
11 73
12 65
13 66
14 75
15 6C
D_0 0
D_1 0
D_2 1
D_3 1
D_4 0
D_5 1
D_6 1
D_7 0
Memory Is Useful

Chapter 7

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?

7.1 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.

7.1.1 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.

7.1.2 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.

Discrete signal

Bit

Digital signal

Read versus Write

Sequential versus Random access

Memory address

Memory data

Volatility

Latch behavior

Tri-state logic

Enable

Static versus Dynamic memory

Quantum tunneling

7.1.3 Digital Audio Tape

When Digital Audio Tape (DAT) was first introduced to the American public, it was touted as delivering superior sound quality. Most importantly, this high quality of sound was not supposed to degrade over time like standard (analog) audio cassette tape recordings.

The magnetic media from which DAT was manufactured was basically the same stuff used to make *analog* audio tape. Explain why the encoding of audio data *digitally* on the same media would provide superior resistance to degradation over analog recordings even though the recording media was the same. Also, explain how this is significant to modern digital data storage technologies such as those used to store photographic images and numerical data.

Challenges

- Identify some disadvantages of digitally-recorded audio.
- Identify similarities and differences between DAT and Compact Disk (CD) audio formats.

7.1.4 Random versus sequential access

Determine whether the following recording devices are *random access* or *sequential access*, and discuss the advantage(s) of one type of access over the other:

- DVD (disk) –
- Audio tape cassette –
- CD-ROM (disk) –
- ROM memory chip –
- EPROM memory chip –
- Vinyl phonograph record –
- Video tape cassette –
- Magnetic "hard" drive –
- Paper tape (a long strip of tape with holes punched in it) –
- RAM memory chip –
- Magnetic bubble memory –
- Flash memory –

Challenges

- Which of these technologies rely on moving components?

7.1.5 Static versus dynamic RAM

Explain the difference between *static* RAM ("SRAM") and *dynamic* RAM ("DRAM") memory technologies. Which type of memory technology provides faster access of data, and why? Which type of memory technology provides the greatest storage density, and why?

Challenges

- How is *refreshing* provided for dynamic RAM chips? Is this something taken care of internal to the chip, or must the circuit designer provide external circuitry to refresh the dynamic RAM chip's memory cells?

7.1.6 Flash versus RAM

Flash memory is a nonvolatile memory technology, offering greater density than either SRAM or DRAM, and faster erasure than standard EPROMs. At first, it would seem Flash memory outperforms all other memory types, but it doesn't. What are some of the *disadvantages* of Flash memory, and what kind of applications is it best suited for?

Challenges

- Explain the operating principle of Flash memory cells.

7.1.7 Binary-BCD conversion by memory

Research datasheets for the 74LS184 and 74LS185 integrated circuits, and then explain how read-only memory technology is used to perform the BCD/binary conversion functions.

Challenges

- In the 1990's Intel produced a microprocessor IC called the *Pentium* which contained a "bug" causing occasional incorrect quotients when dividing floating-point numbers. Research the so-called *Pentium FDIV bug* and explain what this has to do with look-up tables.

7.2 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.

7.2.1 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.

7.2.2 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.*

7.2.3 4k × 8 ROM

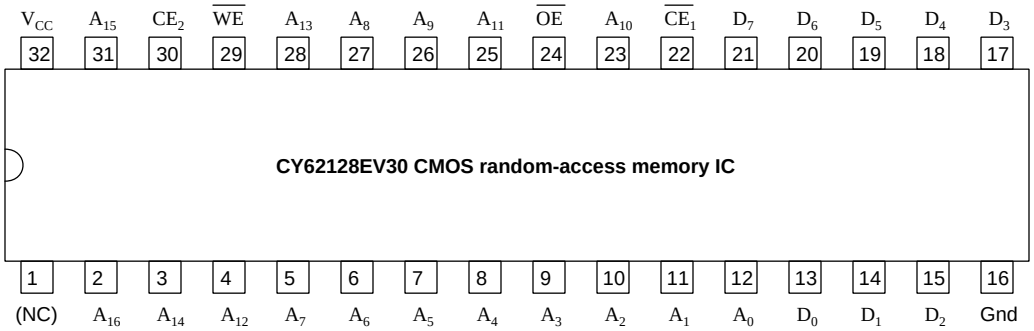
Suppose a particular ROM memory chip is rated at 4k × 8 bits. What, exactly, does this designation mean? How many addresses are there inside this memory chip? How many bits of storage are there, total, in this memory chip? How many address bits are there, and how many data bits are there?

Challenges

- Why are there not exactly 4000 addresses in a "4k" memory chip?

7.2.4 Memory organization

Determine the organization of this RAM IC's memory (i.e. how many addresses, how many bits in the data word) based on an examination of its pin diagram:



Challenges

- Identify how to enable this IC for reading or writing.
- How do we control whether this IC is in "read" mode or in "write" mode?

7.2.5 Storing ASCII text characters

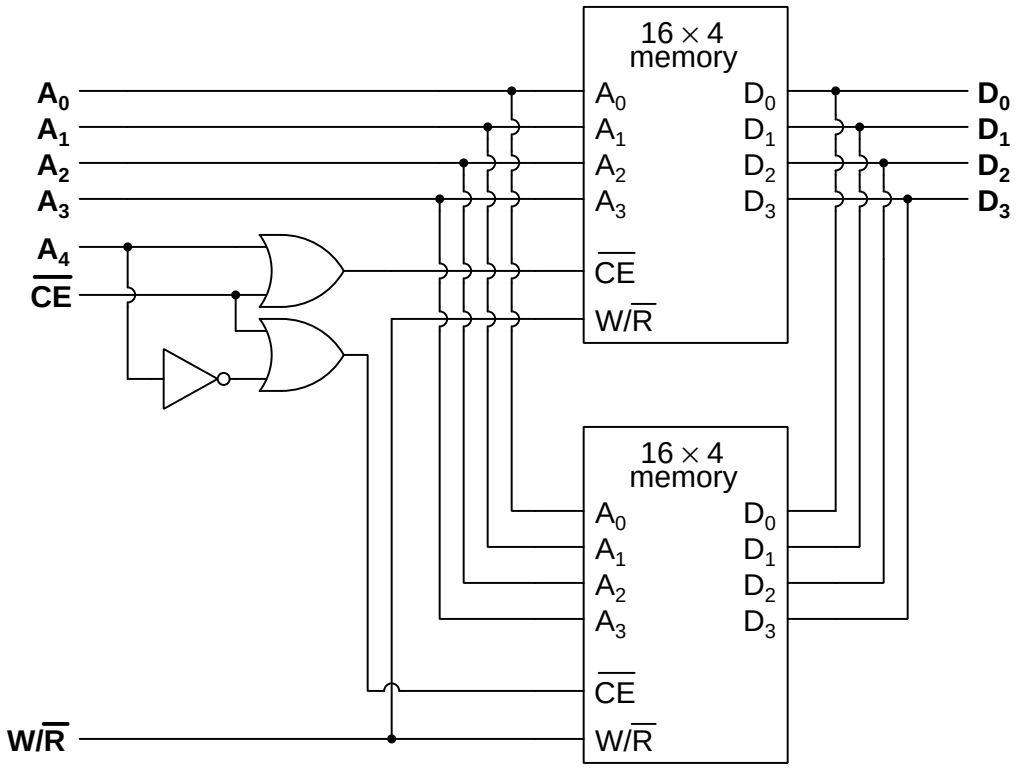
Suppose you need to store a text message in digital memory, consisting of 7500 ASCII characters. What is the most logical memory organization (addresses × data lines) to do this? How many address bits would be needed to store these 7500 characters?

Challenges

- If we needed to store the same number of EBCDIC characters, would it change the memory requirement?

7.2.6 Addressing an expanded memory module

Examine the following schematic diagram for a 32×4 memory array built out of two 16×4 memory ICs:



Now, determine the following:

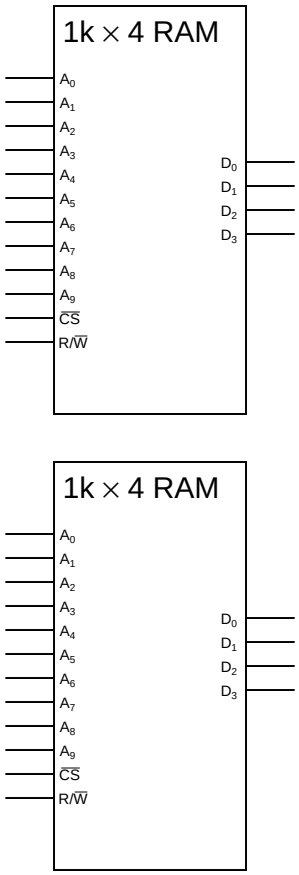
- Bit states for lines A_4 through A_0 to select address 5 within the upper memory IC
- Bit states for lines A_4 through A_0 to select address 10 within the upper memory IC
- Bit states for lines A_4 through A_0 to select address 2 within the upper memory IC
- Bit states for lines A_4 through A_0 to select address 12 within the lower memory IC
- Bit states for lines A_4 through A_0 to select address 3 within the lower memory IC
- Bit states for lines A_4 through A_0 to select address 11 within the lower memory IC

Challenges

- Explain why the OR gates are necessary in this circuit.

7.2.7 Doubling data width

Suppose you need a memory array with 1k × 8 organization, but all you have on hand are 1k × 4 memory chips. Show how you could connect two of them to form the desired array:

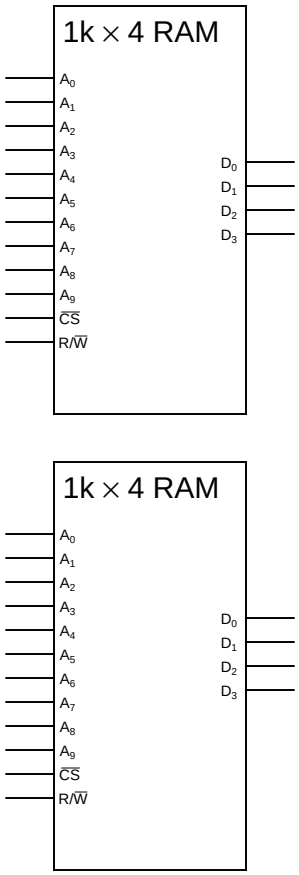


Challenges

- A common mistake made by students when they "expand" the data bus width of a memory array is to parallel the output lines (in the same way that the address lines are shown paralleled here). Why would this be wrong to do? What might happen to the memory chip(s) if their data lines were paralleled.

7.2.8 Doubling address width

Suppose you need a memory array with 2k $\times$ 4 organization, but all you have on hand are 1k $\times$ 4 memory chips. Show how you could connect two of them to form the desired array:



Challenges

- In your solution, identify which of the two memory chips stores the first 1024 addresses, and which stores the next 1024 addresses.

7.2.9 Quadrupling address width

Suppose you need a memory array with 4k × 4 organization, but all you have on hand are 1k × 4 memory chips. Explain how you could build a memory array of this size using multiple 1k × 4 chips.

Challenges

- Devise a solution using only discrete logic gates.

7.2.10 Quadrupling data width

Suppose you need a memory array with 1k × 16 organization, but all you have on hand are 1k × 4 memory chips. Explain how you could build a memory array of this size using multiple 1k × 4 chips.

Challenges

- Explain why this solution requires no additional logic.

7.3 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.

7.3.1 Checksums

After a ROM memory has been programmed with data, it is good to verify that the data now stored is okay, and not corrupted with any errors. A popular method of doing this is to calculate a *checksum* on the stored data, and compare that against the checksum for the original data. If the checksum numbers are identical, chances are there are no corruptions in the stored data.

Explain exactly what checksum is, and how it works as an error-detection strategy.

Challenges

- Checksums are also used to check for errors of digital data transmitted over long cables. Explain how this would be done.

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

"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.

The following principles inform the instructional and assessment philosophies embodied in these learning modules:

- 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[1] 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[2] 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[3] 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

[1] 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.

[2] 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.

[3] 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 characteristic of these learning modules is a lack of step-by-step instructions in the Project and Experiment chapters. Unlike many modern workbooks and laboratory guides where step-by-step instructions are prescribed for each experiment, these modules take the approach that students must learn to closely read the tutorials and apply their own reasoning to identify the appropriate experimental steps. Sometimes these steps are plainly declared in the text, just not as a set of enumerated points. At other times certain steps are implied, an example being assumed competence in test equipment use where the student should not need to be told *again* how to use their multimeter because that was thoroughly explained in previous lessons. In some circumstances no steps are given at all, leaving the entire procedure up to the student.

This lack of prescription is not a flaw, but rather a feature. Close reading and clear thinking are foundational principles of this learning series, and in keeping with this philosophy all activities are designed to *require* those behaviors. Some students may find the lack of prescription frustrating, because it demands more from them than what their previous educational experiences required. This frustration should be interpreted as an unfamiliarity with autonomous thinking, a problem which must be corrected if the student is ever to become a self-directed learner and effective problem-solver. Ultimately, the need for students to read closely and think clearly is more important both in the near-term and far-term than any specific facet of the subject matter at hand. If a student takes longer than expected to complete a module because they are forced to outline, digest, and reason on their own, so be it. The future gains enjoyed by developing this mental discipline will be well worth the additional effort and delay.

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.

This methodology of assigning text-based modules to students for digestion and then using Socratic dialogue to assess progress and hone students' thinking was developed over a period of several years by the author with his Electronics and Instrumentation students at the two-year college level. While decidedly unconventional and sometimes even unsettling for students accustomed to a more passive lecture environment, this instructional philosophy has proven its ability to convey conceptual mastery, foster careful analysis, and enhance employability so much better than lecture that the author refuses to ever teach by lecture again.

Problems which often go undiagnosed in a lecture environment are laid bare in this "inverted" format where students must articulate and logically defend their reasoning. This, too, may be unsettling for students accustomed to lecture sessions where the instructor cannot tell for sure who comprehends and who does not, and this vulnerability necessitates sensitivity on the part of the "inverted" session instructor in order that students never feel discouraged by having their errors exposed. *Everyone* makes mistakes from time to time, and learning is a lifelong process! Part of the instructor's job is to build a culture of learning among the students where errors are not seen as shameful, but rather as opportunities for progress.

To this end, instructors managing courses based on these modules should adhere to the following principles:

- Student questions are always welcome and demand thorough, honest answers. The only type of question an instructor should refuse to answer is one the student should be able to easily answer on their own. Remember, *the fundamental goal of education is for each student to learn to think clearly and independently.* This requires hard work on the part of the student, which no instructor should ever circumvent. Anything done to bypass the student's responsibility to do that hard work ultimately limits that student's potential and thereby does real harm.

- It is not only permissible, but encouraged, to answer a student's question by asking questions in return, these follow-up questions designed to guide the student to reach a correct answer through their own reasoning.

- All student answers demand to be challenged by the instructor and/or by other students. This includes both correct and incorrect answers – the goal is to practice the articulation and defense of one's own reasoning.

- No reading assignment is deemed complete unless and until the student demonstrates their ability to accurately summarize the major points in their own terms. Recitation of the original text is unacceptable. This is why every module contains an "Outline and reflections" question as well as a "Foundational concepts" question in the Conceptual reasoning section, to prompt reflective reading.

- No assigned question is deemed answered unless and until the student demonstrates their ability to consistently and correctly apply the concepts to *variations* of that question. This is why module questions typically contain multiple "Challenges" suggesting different applications of the concept(s) as well as variations on the same theme(s). Instructors are encouraged to devise as many of their own "Challenges" as they are able, in order to have a multitude of ways ready to probe students' understanding.

- No assigned experiment or project is deemed complete unless and until the student demonstrates the task in action. If this cannot be done "live" before the instructor, video-recordings showing the demonstration are acceptable. All relevant safety precautions must be followed, all test equipment must be used correctly, and the student must be able to properly explain all results. The student must also successfully answer all Challenges presented by the instructor for that experiment or project.

Students learning from these modules would do well to abide by the following principles:

- No text should be considered fully and adequately read unless and until you can express every idea *in your own words, using your own examples.*
- You should always articulate your thoughts as you read the text, noting points of agreement, confusion, and epiphanies. Feel free to print the text on paper and then write your notes in the margins. Alternatively, keep a journal for your own reflections as you read. This is truly a helpful tool when digesting complicated concepts.
- Never take the easy path of highlighting or underlining important text. Instead, *summarize* and/or *comment* on the text using your own words. This actively engages your mind, allowing you to more clearly perceive points of confusion or misunderstanding on your own.
- A very helpful strategy when learning new concepts is to place yourself in the role of a teacher, if only as a mental exercise. Either explain what you have recently learned to someone else, or at least *imagine* yourself explaining what you have learned to someone else. The simple act of having to articulate new knowledge and skill forces you to take on a different perspective, and will help reveal weaknesses in your understanding.
- Perform each and every mathematical calculation and thought experiment shown in the text on your own, referring back to the text to see that your results agree. This may seem trivial and unnecessary, but it is critically important to ensuring you actually understand what is presented, especially when the concepts at hand are complicated and easy to misunderstand. Apply this same strategy to become proficient in the use of *circuit simulation software*, checking to see if your simulated results agree with the results shown in the text.
- 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[4] effort, and never give up!

Students interested in using these modules for self-study will also find them beneficial, although the onus of responsibility for thoroughly reading and answering questions will of course lie with that individual alone. If a qualified instructor is not available to challenge students, a workable alternative is for students to form study groups where they challenge[5] one another.

To high standards of education,

Tony R. Kuphaldt

[4] 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.

[5] Avoid the temptation to simply share answers with study partners, as this is really counter-productive to learning. Always bear in mind that the answer to any question is far less important in the long run than the method(s) used to obtain that answer. The goal of education is to empower one's life through the improvement of clear and independent thought, literacy, expression, and various practical skills.

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

"27C256 256k (32 K × 8) CMOS EPROM", document DS11001L, Microchip Technology Incorporated, 1996.

A Manual of Operation for the Automatic Sequence Controlled Calculator, Harvard University Press, Cambridge, Massachusetts, 1946.

"CY62128EV30 MoBL Automotive 1-Mbit (128 K × 8) Static RAM", document 001-65528, Revision E, Cypress Semiconductor Corporation, San Jose, CA, 15 April 2015.

Kahng, Dawon, *US Patent 3,500,142*, "Field Effect Semiconductor Apparatus With Memory Involving Entrapment Of Charge Carriers", application 5 June 1967, patent granted 10 March 1970.

Rohrer, George A. and McMillan, Larry, *US Patent 4,707,897*, "Monolithic Semiconductor Integrated Circuit Ferroelectric Memory Device, and Methods of Fabricating and Utilizing Same", application 24 March 1980, patent granted 24 November 1987.

Savitzky, Stephen R., *Real-Time Microprocessor Systems*, Van Nostrand Reinhold Company, New York, NY, 1985.

"STK14C88 128 K × 8 AutoStore nvSRAM", document 001-52038, Revision F, Cypress Semiconductor Corporation, San Jose, CA, 31 March 2015.

The Semiconductor Memory Data Book, First Edition, document CC-420, Texas Instruments Incorporated, Dallas, TX, 1975.

Villee, Claude A.; Solomon, Eldra Pearl; Martin, Charles E.; Martin, Diana W.; Berg, Linda R.; Davis, P. William; *Biology* Second Edition, Saunders College Publishing, Philadelphia, PN, 1989.

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.

16 January 2025 – minor edits to the Tutorial and also to image_4361 used in one of the Case Tutorial sections.

18-25 August 2024 – divided the Introduction chapter into three sections, one with recommendations for students, one showing challenging concepts related to the module's topics, and one with recommendations for instructors with sample learning outcomes and measures. Also added a Technical Reference section on tri-state logic outputs, and correct some schematic errors in the "Electronic memory" section of the Tutorial chapter.

22 January 2024 – minor corrections to two images (image_3800 and image_3801). Also made minor edits to the wording of the two address-expansion Case Tutoral examples.

29 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.

19 January 2022 – added photograph of EPROM-based arbitrary waveform generator to that Case Tutorial section.

28 November 2021 – added more explanation to the Tutorial on address versus data, and also expanded descriptions of different types of memory elements.

30 August 2021 – added a "hex dump" memory map to the introduction of the Animation of a 16×8 ROM being used to store ASCII characters.

26 August 2021 – fixed typographical error in image_3801 where the "Read" and "Write" text labels were reversed. Also divided Tutorial up into individual sections, and elaborated upon row/column addressing with a new illustration.

9 July 2021 – replaced some TeX-style italicizing markup with LaTeX-style, and also added a Case Tutorial section on using nonvolatile memory to form an AWG (arbitrary waveform generator).

10 May 2021 – commented out or deleted empty chapters.

18 April 2021 – changed all lower-case Greek letter "phi" symbols (ϕ) to upper-case (Φ).

25 January 2021 – minor edits to the Introduction chapter.

27 August 2020 – added Case Tutorial chapter with examples showing memory expansion. Also, corrected a typographical error pointed out by Ty Weich.

25 August 2020 – added a quantitative question on quadrupling the data width of a memory array, and made minor edits to the tutorial to better explain the function of the "master enable" pin on a RAM chip.

23 August 2020 – significantly edited the Introduction chapter to make it more suitable as a pre-study guide and to provide cues useful to instructors leading "inverted" teaching sessions.

5 June 2020 – added timing diagram to Tutorial.

31 May 2020 – added timing diagram to Tutorial.

7 March 2020 – added Technical Reference section on digital pulse criteria.

29 January 2020 – added Foundational Concepts to the list in the Conceptual Reasoning section.

20 January 2020 – added Historical Reference on floating-gate MOSFET memory cells.

19 January 2020 – wrote Tutorial chapter.

16 January 2020 – document first created.

Index