Ternary Computing Testbed 3-Trit Computer Architecture

Jeff Connelly

Computer Engineering Department

August 29th, 2008

with contributions from Chirag Patel and Antonio Chavez

Advised by Professor Phillip Nico
California Polytechnic State University of San Luis Obispo

Table of Contents

1. Introduction

As the computing industry advances, researchers are discovering new and innovative ways to increase computer processor performance, rather than strictly adhering to the traditional technique of increasing chip density. Some techniques, such as multi-core processors, build upon well-established binary circuit systems, while other technologies such as quantum computing require a radical rethinking of the foundations of computer science. Our team researched a middle-ground by building a functional computer based on base 3 or *trinary* (synonymous with *ternary*) logic. Trinary computers are digital as are binary computers, but are not as extreme departure from classical computing technologies as quantum computers. Trinary technology is relatively unexplored territory in the computer architecture field.

The trinary numbering system itself has numerous interesting properties that make building trinary-based digital circuits an attractive proposition from an efficiency perspective (**2.1**). Although trinary logic is significantly more complex than binary logic, the addition of the third logic state opens up exciting new possibilities (appendix **A**). Additionally, from trinary naturally follows "balanced trinary arithmetic", a numbering system using digits -1, 0, and 1. Esteemed Computer Scientist Donald Knuth describes balanced trinary as "the prettiest number system of all"[1] for its elegant arithmetic properties (appendix **B**).

Nonetheless, trinary computer systems are not a new idea. In as early as 1959, trinary computers have been built, simulated, or imagined (appendix **C**), to varying degrees of completeness and practicality. In this project, our team set out to design and build a complete trinary computer that is capable of running a simple game (section **3**). I began designing logic gates at the transistor level, and then built upon the gates to design memory and control circuitry (appendix **D**). Layered further on top, I implemented and designed high-level architectural components (section **4**) including an instruction memory/decoder, registers, a control unit, and an Arithmetic Logic Unit. In the end, the components culminate in a processor that can execute programmed instructions in a 3-trit assembly language (section **4.5.7**). A compiler for the high-level programming language which compiles into the trinary assembly language is part of the Ternary Computing Testbed project, but out of the scope of this report. Although I also helped with the physical construction of the computer, this report focuses on my primary contribution to trinary computing: the implementation of a practical Ternary Computing Architecture. Our hope is to inspire a change in the foundation of the computer architecture industry which will lead to more efficient computer systems.

In the end, I achieved the goal of designing a complete trinary computer capable of running a simple numerical guessing game. The transistor-level architecture design successfully simulates in the LTspice circuit simulation program and behaves as expected.

1.1. Method

Our practices were inspired by Shimon Schocken's workshop on computer construction, often offered as a college course titled "From NAND to Tetris", taught along with a book written by the instructor[2]. Schocken's course is composed of several units, ascending from low-level logic gates and architecture to high-level application programming concepts. We followed a similar bottom-up approach, beginning with trinary logic, trinary arithmetic, sequential logic and computer architecture, up to writing a assembly-language game running

on a trinary computer.

1.2. Plan

We, the Ternary Computing Testbed team, met weekly with our advisor to discuss our progress and our plans. Each of us posted weekly status reports online[3] along with weekly individual status reports[4].

In addition, we posted our research findings on a publicly-viewable wiki[5].

This interdisciplinary senior project built on our Honors research project started in Winter 2008, applying what we have learned in our efforts to research existing research in trinary, in order to construct a functional trinary computer system.

1.3. Team and Individual Responsibilities

The Ternary Computing Testbed team is composed of Jeff Connelly, Antonio Chavez, and Chirag Patel. You are reading Jeff's report, which also includes documentation of Chirag's efforts since he is not doing a senior project. Antonio's report will be submitted as a separate senior project report, although references to it will be made in this report when appropriate.

1.3.1. Jeff Connelly

Jeff's tasks include designing and simulating the complete trinary architecture to be constructed by Chirag, writing software for the trinary computer, and helping Chirag construct and test circuits.

Deliverables:

- A wiki documenting our research and progress[5]
- Transistor-level SPICE simulation of complete 3-trit CPU architecture circuits and associated components, running a simple game (section **4.5.7**)
- An assembly-language implementation of a simple game to run on the trinary computer

1.3.2. Antonio Chavez

Antonio's tasks include developing software to support construction of the trinary computer, instruction-level simulators of a simple and extended architecture, and a high-level language compiler for a custom trinary computer architecture.

Deliverables:

- Several utility tools[6] were developed to support our work:
  - Arbitrary trinary expression evaluator
  - Expression creator
  - Radix converter
- Instruction-level CPU simulators[7]
  - Simulator of 3-trit architecture that Chirag and Jeff build.
  - Simulator of an expanded trinary architecture, designed by Antonio
- Compiler[8]

- Design a language and implement a compiler for a high-level language using trinary.

While Antonio is part of the team effort, he will submit a separate Senior Project report since his work significantly diverges from ours.

1.3.3. Chirag Patel

Chirag's tasks include physically building and testing trinary circuits for logic gates, storage elements, control units, and all other circuitry.

Deliverables:

- A physical trinary computer running a simple game

While this work is not Chirag's senior project, relevant portions of his progress will be covered in this report as we worked together extensively and our work is related in that Jeff's simulated architectures are being physically built in hardware by Chirag.

2. Background Theory

2.1. Base 3

Base 3 is any weighted numbering system that uses three digits. Familiarity with positional numbering systems[9], including binary and decimal, is assumed throughout this document.

Before delving into the technical details, a word on terminology is in order. Base 3 is traditionally known as *ternary*. In this report, I use the term *trinary* over *ternary* or *tertiary*, as a homage to Steve Grubb's Trinary.cc web site. Merriam-Webster defines ternary as "*Composed of three or arranged in threes, having the base three*." while trinary is defined as "*Consisting of three parts of proceeding by threes; ternary*.", and tertiary as "*Third in place, order, degree, or rank*". Although they are nearly synonyms, *tertiary* would be more appropriate if base 2 was called *secondary*. This naming ambiguity also arises in programming languages such as C and Perl which have a *ternary* or *trinary* operator, often spelled "?:". Although some in the community prefer the term *ternary*[10] Larry Wall, the creator of Perl, in Apocalypse 3[11] as well as in *Programming Perl*[12], uses the term *trinary*. In this paper, I prefer *trinary* as well, although *ternary* will occasionally be used in contexts where it is common.

In the following sections, I will compare the benefits and disadvantages of trinary to several alternatives.

2.1.1. Compared to Analog

The earliest known computer is the *Antikythera mechanism*, an analog mechanical computer designed to calculate astronomical positions, found in Greece and dating to *circa* 100 BC[13]. In more recent times (*circa* 1943), the TDC Mark III electromechanical analog computer was used in U.S. submarines during World War II to aim torpedoes[14]:

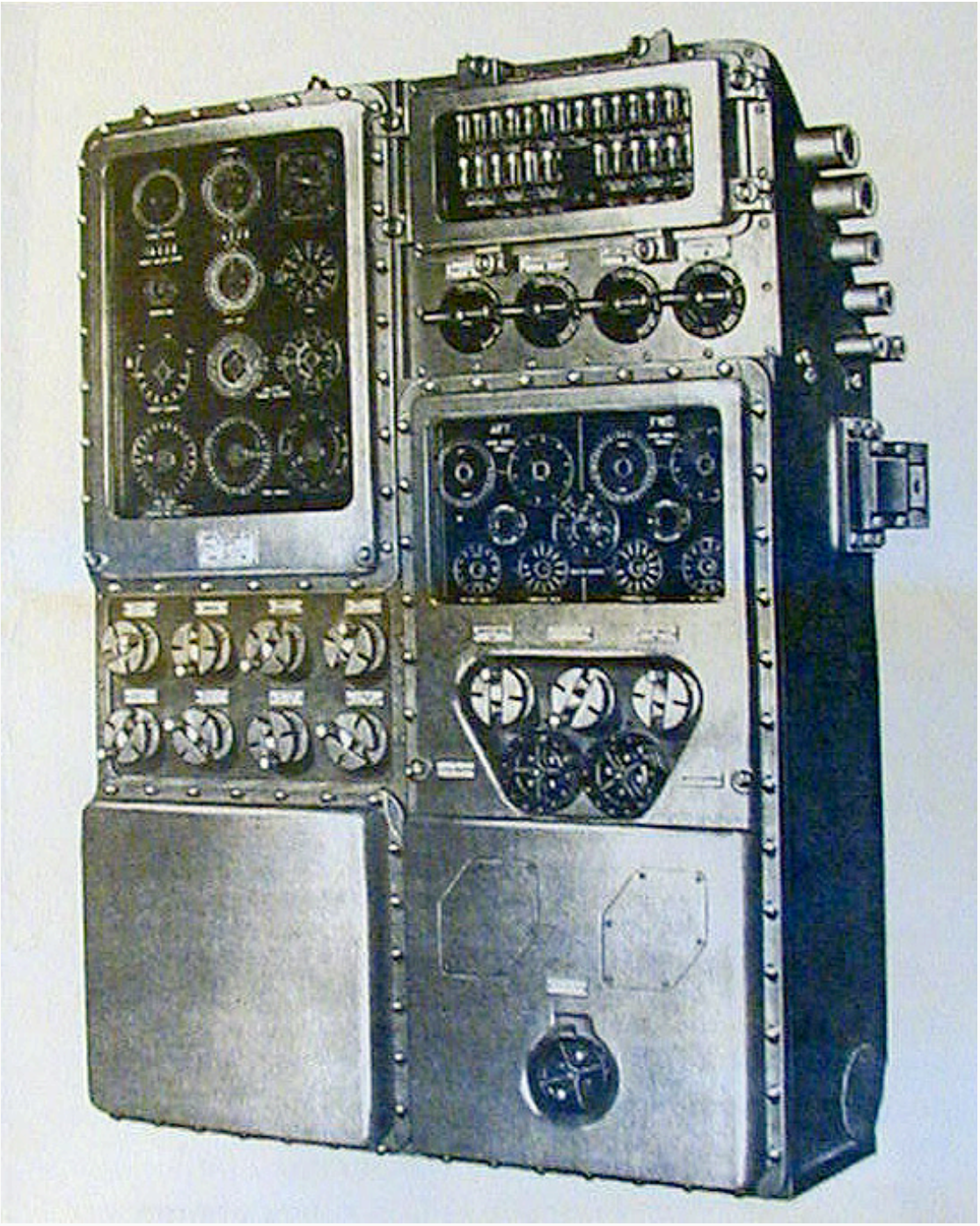


Figure 1. TDC Mark III, 1943[14]

Electrically, operational amplifiers can be used in an analog computer to perform integration, differentiation, root extraction, multiplication/division, logarithm, and anti-log operations[15]. Although analog computers played an important historical role[16], they have the significant disadvantage that noise can unpredictably affect the results of a computation[17]. Following the development of digital computers, analog computing quickly fell out of mainstream usage[14]. Although trinary computers are not immune to noise, they are significantly more resistant than analog computers because discrete voltage levels are used. Any voltage within a quantifiable range is accepted as signifying a given logic state (section **D.3.1**).

2.1.2. Compared to Digital

A *digital computers* is defined as storing data in terms of discrete states and having execution proceed in

discrete steps from one state to the next[18]. Early digital computers used ten voltages, that is, base 10 or *decimal*. Atanasoff[19] came up with the idea in the 1930s of using two voltage levels, or *binary*:

> Atanasoff was thinking about computers. There were already mechanical and analog computers. But Atanasoff thought there might be better methods of computing. He drove from dry Iowa to a bar over the Illinois line, drank three Scotch and waters, and had a *Eureka!* moment. "That's when he figured out he could do everything in base 2," Gustafson says. Base 2 is digital. It's 1s and 0s. Previous computers worked in base 10. "He jotted on a cocktail napkin all the basic principles of modern computing."

In 1938, Claude E. Shannon published his master's thesis describing how the *true* and *false* notions of George Boole's (1847—1854) *Boolean Algebra* could be mapped to the two logic levels of a binary digital computer[20]. The rest, as they say, is history. Digital binary computers are the most prevalent computing technology available today, by far.

Digital computers have the advantage of computational accuracy over analog computers. The two discrete voltage levels allow for some variation due to noise or other environmental factors, without changing the outcome of a calculation. Barring a significant disturbance (such as *cosmic ray* interference[21]), digital computers perform accurate calculations. Like base 10 and 2, base 3 is digital and therefore benefits from the properties of having discrete voltage levels.

2.1.3. Compared to Base *e*

Given a set of assumptions outlined in [22], base *e* is the most efficient base for representing arbitrary numbers. If one measures cost as the radix (base, or *r*) times the number of digits ("width", or *w*), on the grounds that greater widths require proportionally more circuitry, and higher radices require proportionally more complex circuitry. That is, a 16-digit number will require twice the amount of circuitry as a 8-digit number; additionally, the assumption is that base 4 (for example) requires twice as complex circuitry as base 2, and base 3 requires 3/2 or 1.5 times as complex circuitry as base 2, for an abstract definition of complexity. These assumptions are revisited in section **5**.

3 is the closest integer to *e* (2.718…)—closer than 2—therefore, the reasoning is that base 3 is more efficient than base 2 when used to build digital systems.

The following figure from [22] shows how base *e* occupies the local minimum of a graph plotting cost (as previously defined), for numbers of several magnitudes:

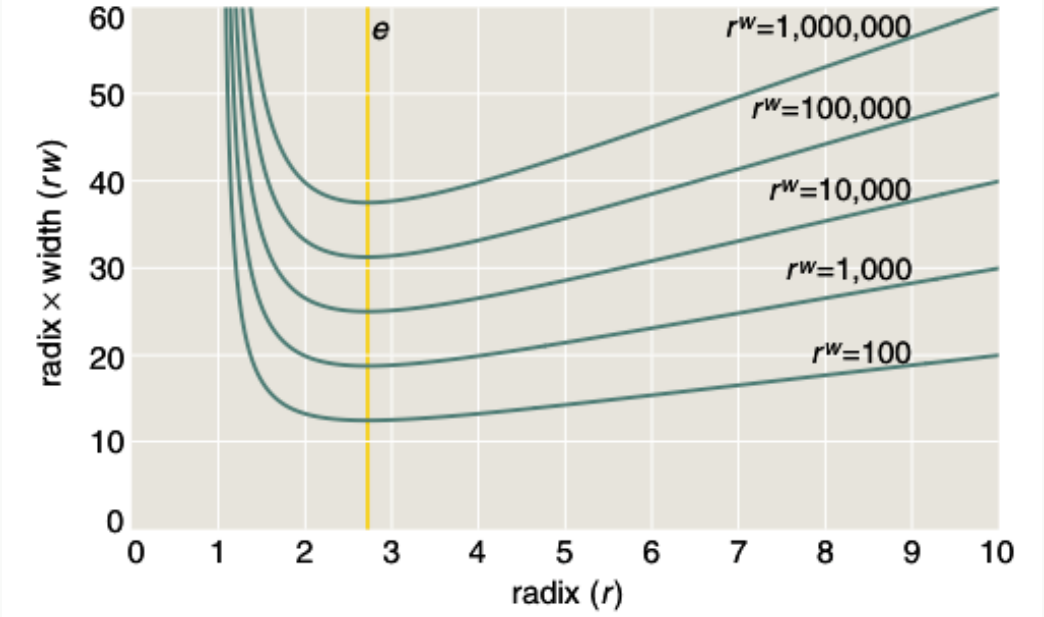


Figure 2. "Most economical radix for a numbering system is (about 2.718) when economy is measured as the product of the radix and the width, or number of digits, needed to express a given range of values. Here both the radix and the width are treated as continuous variables."[22]

Additionally, base 3 is the most efficient integer base. As shown below, the cost of 3 closely follows that of base *e*, whereas base 2 is significantly more costly in many instances:

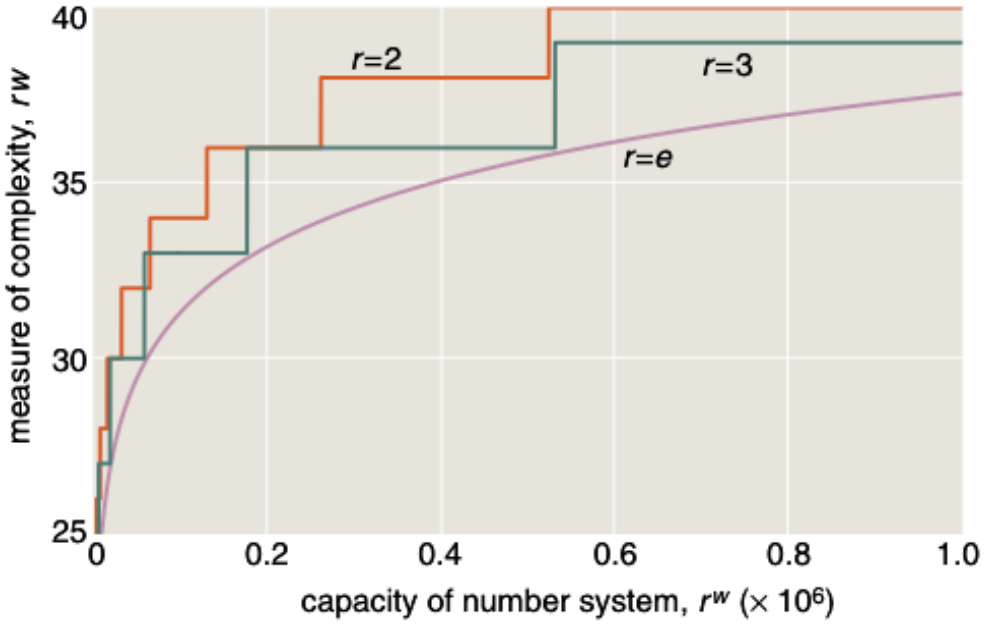


Figure 3. "[The] most economical integer radix is almost always 3, the integer closest to e. If the capacity of a numbering system is r^w, and the cost of a representation is *rw*, then r=3 is the best integer radix for all but a finite set of capacities. Specifically, ternary is inferior to binary only for 8,487 values of *rw*; ternary is superior for infinitely many values."[22]

A. Srivastava and K. Venkatapathy[23] have found that multi-valued logic allows for significantly increased performance, largely because of the reduced number of interconnects. Srivastava also came to the same conclusions as in *Third Base*—base 3 is more efficient than base 2—assuming that cost is measured as radix times width. The relevant portion of the paper is quoted in full below:

The performance of two levels (binary logic) is limited due to interconnect which occupies large area on a VLSI chip. In a VLSI circuit, approximately 70 percent of the area is devoted to interconnection, 20 percent to insulation, and 10 percent to devices[24]. One can achieve a more cost effective way of utilizing interconnections by using a larger set of signals over the same area in multiple-valued logic (MVL) circuits. This also solves the problem of pinout (the limit to the amount of data that can enter and exit a chip). Commercially multiple-valued logic circuits have made an appearance with the four-valued read-only memory (ROM) which Intel used in the control store of its 8087 numeric coprocessor[24]. Hitachi has introduced into the market a 16-valued mass memory with a high storage capacity. Kameyama et al.[25] reported a 32 x 32 bit signed digit (SD) multiplier implementation using MVL circuits realized in current-mode CMOS technology. The chip area and power dissipation of MVL multiplier implementation reduced to half that of the fastest conventional binary realization of the same multiplier.

The main draw back in multiple valued logic circuits is that their design techniques are more complex than the binary logic circuits[26]. The implementation of MVL circuits have ranged through integrated injection logic, emitter coupled logic, CMOS and n-MOS technologies and charge-coupled devices. In this work, the design of ternary-valued logic circuits have been explored over other ternary-valued logic due to the following reasoning. In a numerical system, the number N is given by $N = R^d$ where R< is the radix and d is the necessary number of digits up to the next highest integer value where necessary. If the cost or complexity *C* in any system is assumed to be proportional to R x D[27], then $C = k(R \times d) = k[R(\ln N/\ln R)]$ where *k* is some constant. Differentiating with respect to *R* will show that for a minimum cost *C*, *R* should be equal to *e*(2.718). Since in practice *R* must be an integer, this suggests that *R* = 3(ternary) would be more economical than *R* = 2(binary)[28]

Srivastava and Venkatapathy were not the only ones to come to these conclusions. Dhande and Ingole[29] have also found that base 3 is the most efficient radix for switching circuits, because of the following reasons:[30]

- Base 3 reduce the interconnections required to implement logic functions.
- Base 3 therefore reduces chip area.
- Base 3 allows more information to be transmitted over a given set of lines.
- Base 3 has a lower memory requirement for a given data length.
- Serial operations can be carried out at a higher speed[31][32]

The advantages of base 3 have been confirmed in digital memories, communications components, and the field of digital signal processing[33]. Our research extends the concept of base 3 to the field of computer architecture.

2.1.4 Trits, Tribbles, and Trytes

Before we delve too deeply into trinary computing systems, additional terminology definitions are in order. In the binary world, bits, nibbles, and bytes are household names[34]. As for trinary, the analogous names for base 3 have not been standardized.

Analogous to bits, *trits* are base 3 digits. The term *tert* is also occasionally used, but it will not be used in this paper.

The *TriINTERCAL programming language* defines unsigned 10-trit (0 to 59048) and 20-trit words. The ranges of the 10- and 20-trit numbers are remarkably close to their 16- and 32-bit counterparts. 16 bits store as much as 16*(log(2)/log(3)) ≈ 10.0949 trits, and 32 bits store as much as 32*(log(2)/log(3)) ≈ 20.1898 trits. Following

the pattern, 64 bits are about 40.3795 trits. However, base 2 word sizes are almost always powers of 2. Therefore, I suggest using powers of base 3 for word sizes, grouping the trits as follows:

Table 1. Trit Grouping Names

Trits (base 3)	Digits, base 9	Digits, base 27	Max. (decimal, $3^{trits} - 1$)	Name	Description
1	1/3	1/27	2	trit	Relatively well-established.
2	1	2/3	8	nit	One base-9 digit.
3	3/2	1	26	tribble	Half of a tryte, one base-27 digit.
6		2	728	tryte	Analogous to a byte.
9			19,682	*not defined*	*not defined*
27			7,625,597,484,986	*not defined*	*not defined*

There has been much informal discussion about trinary digit groupings on Slashdot[35], rather than peer-reviewed journals, but I believe these make the most sense based on extrapolating the terminology used for binary (at least, one person[36] agreed). In the architecture I designed (section **4**), the natural *word size* that all operations operate on is 3 trits, or one tribble.

In binary, the two states often correspond to 0 and 1, or *true* and *false*. As discussed in depth later in this document (appendix **A** and **B**), the three states in trinary can be defined as the following:

Table 2. Trinary Digits

Set	Name / comments
{0,1,2}	Unbalanced Trinary
{0,1/2,1}	Fractional Unbalanced Trinary
{-1,0,1}	Balanced Trinary
{F,?,T}	Unknown-State Logic
{T,F,T}	Trinary Coded Binary

The most common trinary digit mappings are {0,1,2} (unbalanced) or {-1,0,1} (balanced). Of these, {-1,0,1} can be defined as {F,?,T} where *?* is unknown (simultaneously *T* and *F*)—this is hereby termed "Unknown-State Logic" (USL) and the logical properties of USL are covered in section **A.7**. The set {0,1/2,1} is mentioned by Merrill[37] but is not covered here, and it can be thought of as simply {0,1,2} (unbalanced) with half the logic level quantities. Lastly, the set {T,F,T} strictly maps trinary digits to binary, and it is expected to be useful for interfacing with binary systems[38].

We chose to use balanced trinary when possible, because of its obvious mapping to electrical voltages: -1 negative, 0 neutral, 1 positive. It is useful to represent -1 as a single digit so it lines up properly in fixed-width text. There are several conventions that have been defined:

- Merrill[37] used T for -1. "T" is like 1 with a negative sign on top of it, but it unfortunately could be too easily mistaken for "True".
- Setun used *i* for -1[39]. This is what will be used when no special formatting is possible. *i* is also used to represent the square-root of -1, so there is some pre-existing convention here.
- Knuth[1] uses 1 with an overline. This is the convention I have adapted in this document: $\overline{1}$. I developed

an extension for the wiki we used to automatically overbar the *i* symbols if they are enclosed within the <trits> tag.

For details on how arithmetic operations can be performed on balanced trinary numbers, refer to section **B.1**.

2.1.5. Base 9 and 27

As in binary computing where one *octal* digit is 3 bits and one *hexadecimal* digit is 4 bits, in trinary computing it is useful to define conventional bases to compactly represent a sequence of trits.

If $a^n = b$, then one base b digit holds n base-a digits. Since $3^2 = 9$, one *ninary* (base 9) digit represents 2 trits. Groups of 2 trits can be converted directly to a ninary digit and vice versa as follows:

Table 3. Ternary and Nonary, Unbalanced

Base 3	Base 9
00	0
01	1
02	2
10	3
11	4
12	5
20	6
21	7
22	8

Table 4. Ternary and Nonary, Balanced

Base 3	Base 9
$1\overline{1}$	$\overline{4}$, ④
$\overline{1}0$	$\overline{3}$, ③
$\overline{1}1$	$\overline{2}$, ②
$0\overline{1}$	$\overline{1}$, ①
00	0
01	1
$1\overline{1}$	2
10	3
11	4

For example, using the table above, one can determine that $2021_3 = 67_9$, or in a balanced numerical system: $11\overline{1}1 = 4\overline{2}_9$. Unbalanced base 9 uses digits [0..8]. In the table above, an overbar is used over negative digits.

Setun[39] pioneered the use of upside-down 4,3,2,1 and right-side up 0,1,2,3,4 for balanced base-9, but the Unicode character standard does not define such symbols[40] so they cannot be easily represented on a modern computer system. Therefore, I chose single-character representations of the negative digits 1-4 to be the digits *circled*, shown above, starting from Unicode character U+2460.

Although one ninary digit represents two trits, greater compactness is desirable for writing longer sequences of trits. Since, $3^3 = 27$, in base 27 (which unfortunately lacks a catchy name), three trits represent one base-27 digit. *Unbalanced base 27* uses the characters [0..9] in addition to [A..S], excluding I and O to avoid confusion between 0 and 1, as follows:

Table 5. Unbalanced Ternary and 27-uary

3	000	001	002	010	011	012	020	021	022	100	101	102	110	
27	0	1	2	3	4	5	6	7	8	9	A	B	C	
3	111	112	120	121	122	200	201	202	210	211	212	220	221	222
27	D	E	F	G	H	J	K	L	M	N	P	Q	R	S

In *balanced base-27*, digit values range from -13 to +13 and use symbols 0 through 9, A through D (table not shown). Negative digits are represented with overbars over the digits.

One base-27 digit is equivalant to exactly 1.5 = 3/2 = log27(9) base-9 digits.

Any power-of-3 radix higher than 27 is impractical for human usage due to the large number of glyphs that would be required (81 for 3^4), but base 27 is highly practical and useful for compact trit representation.

2.1.6. Text

For text on conventional 8-bit binary systems, 7-bit ASCII is a common choice. The 8^{th} bit is variously used for additional characters in "extended ASCII" character sets, of which there are too many conflicting incompatible standards, or to indicate that a character code point greater than 7 bits is approaching (as with UTF-8, a Unicode encoding). With trinary, we are not limited by 8 bits, and 8 digits is rather small. It therefore makes sense to choose a character size big enough for Unicode characters, making Unicode the textual standard rather than ASCII. Unicode uses 21-bit code points up to 2,097,151 to define a repertoire of more than 100,000 standard characters. The Internet Request for Comments document #4042[41], released on April 1st of 2005, defines Unicode Transformation Formats UTF-9 and UTF-18 that would be suitable for usage on an advanced trinary computer. All of the encodings are able to map the full set of characters using varying methods.

2.2. Logic and Arithmetic

There is much to be said about the logical and arithmetic aspects of a trinary-based computer system. Please refer to Appendix **A** and **B** for a detailed analysis.

3. Application Description

As described in the introduction, the end goal of this project is to build a trinary computer running a simple game. This section discusses the high-level aspects of simple gaming applications that were developed.

3.1. Christmas Lights Game

The most trivial "game" that could be developed using a limited computer architecture is what I call the *Christmas lights game*. Although this game does not involve competition, it can provide amusement to young children and therefore is classified as a game.

In this game, the user is treated to a series of several multi-color LEDs. The LEDs sequence through a series of colors, as the instructions advance. The user can change the programming to experiment with an endless variation of Christmas light sequences, limited only the creativity of the programmer.

While simplistic, this game illustrates the programmability of the computer architecture. It is intended for use

on a physical implementation of the Trinary Computer Architecture, also known as *TCA0* (section **4.5.4**), as part of Chirag's tasks. This game has been successfully simulated in a transistor-level design using the LTspice circuit simulator, as detailed in section **4.5.4**.

3.2. Guessing Game

A slightly more sophisticated game is the classic *number guessing game*. In this game, the player inputs a number, and is told whether he or she is too high, too low, or just right.

The peripherals required by this game are:

- A multi-color LED indicating the result of guessing
- A 3-trit array of switches to enter your guess

The high-level procedure of this game is as follows:

- Loop:
  - On power up, the system stores a secret number in a register (the number to guess)
  - Loop:
    - Compare input register from switches to the register that has the correct number
    - Check status trit
      - Status trit $\bar{1}$, too low.
      - Status trit 0, got it right. Break out of inner loop and re-initialize the secret value.
      - Status trit 1, too high.

The programming details of this game are explained later in the architecture section, where the architecture that is able to run this game (known as *TCA2*) is introduced. The "status trit" is wired to a multi-color LED that gives feedback on the guess to the user.

This game demonstrates user input, register storage, branching, and arithmetic. In order to compare the user input to the secret number, the computer system has to subtract the two values and check whether the result is negative, zero, or positive, indicating that the guess was too low, correct, or too high. If the guess is incorrect, the program immediately loops back to the comparison. However, if the guess is incorrect, the program loops back to the very beginning of the program and re-loads the secret number. The secret number is a fixed part of the software, but can be changed by reprogramming the software on the fly; although because of this looping construct, changing the secret value will not take affect until after the user has correctly guessed the previous value. This was done in order to demonstrate conditional branching.

In summary, this game demonstrates programmability, arithmetic computations, input/output, and conditional branching.

This program was successfully demonstrated on a transistor-level LTspice circuit simulation, as detailed in section **4.5.7**.

4. Architecture Description

I designed several related Trinary Computer Architectures of varying complexity:

- **TCA2** is a complete 3-trit system, implementing compare, branch, and load instructions. It can successfully run a "guessing game program" in a transistor-level LTspice simulation as well as in a CPU instruction-level software simulation (see section **4.5.7**).
- **TCA1** is an old prototype architecture obsoleted by TCA0 and TCA2.
- **TCA0** is a simplified proof-of-concept architecture, that has been simulated and is intended to be easily physically built in hardware. It only implements a load instruction. For overall architecture of TCA0, see section **4.5.4**.

Antonio Chavez designed a third, extended, architecture, TCA3, to be simulated only at the CPU instruction level (rather than the transistor-level), exploring higher-level trinary concepts. Antonio's architecture is covered in his own separate senior project paper and will not be discussed further.

4.1. Power Supply

All electrical computers require a source of electrical power to operate. Modern personal computers often have a power supply that outputs several voltages, but most of the current is drawn through a +5 V rail. The motherboard often steps down the voltage even further to power the processor, in order to reduce power consumption[42]. In either case, the processor power supply is one single voltage.

In our trinary computer, we used a dual-rail voltage supply of positive and negative voltages with equal magnitudes. Two supplies were connected back-to-back to provide +5 V, 0 V, and -5 V voltages, corresponding to logic 1, 0, and $\overline{1}$. For simulation purposes, I designed a component, known as `tpower` in the git repository, to provide this functionality:

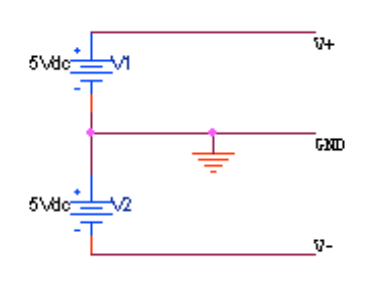


Figure 4. Trinary Power Supply

Alternatively, a suitable power supply can be constructed by supplying a single 10 V voltage, tying the negative side of the 10 V supply to the negative rail, and using a 1/2 voltage divider for ground. In our trinary computer, ground does have its uses, although it is not used as frequently as the positive and negative supply; nonetheless, we did not choose to implement the power supply this way for reasons of simplicity.

Within our circuitry, we used the node names `$G_Vdd` and `$G_Vss` to refer to the +5 V and -5 V rails, respectively. The "$G_" prefix informs the circuit simulation software we used (LTspice) that the nodes are "global", in that they traverse subcomponent hierarchies[43]. Doing this allows us to use the same power supplies for all electrical components, without having to wire power lines to each component within the simulation.

Logic levels are relative to ground for balanced trinary, but they can also be read relative to $G_Vss to convert to unbalanced trinary:

Table 6. Relative Voltages

From	To	Logic Level
GND	V-	$\overline{1}$, balanced
GND	GND	0, balanced
GND	V+	1, balanced
V-	V-	0, unbalanced
V-	GND	1, unbalanced
V-	V+	2, unbalanced
V+	V-	2, inverted unbalanced
V+	GND	1, inverted unbalanced
V+	V+	0, inverted unbalanced

To convert balanced to unbalanced, 1 is added. The alternate system of converting between balanced and unbalanced, replacing $\overline{1}$ with 2, as suggested in the TriINTERCAL manual[44], was not used as it changes the meaning of the truth tables. For simplicity, we exclusively used balanced trinary within this computer architecture.

In the labs on campus, obtaining a steady +5 V and -5 V is easy using the Agilent DC power supply equipment, which was sufficient for our testing. However, to make the computer stand-alone, additional circuitry is needed to regulate the voltage from a battery or AC mains to the desired voltages.

We purchased[45] a handful of AA batteries intended to build the DC power supply, but due to time constraints we did not design a power circuit, instead preferring to use the available lab voltage supply equipment. However, a future task (beyond the scope of this senior project) could be to design, build, test, and integrate such a power supply with the rest of the computer system. I researched several ideas of how to best accomplish this:

- Use an LM317 adjustable voltage regulator chip with appropriate resistors to supply a 5 V output voltage with up to anywhere from about 37 V input or lower.
    - Alternatively, use an LM7805 for a fixed 5 volt output, but without the flexibility to change the voltage later if we need to.
- Use a diode bridge rectifier on the input of the LM voltage regulator, to ensure that the polarity is correct.
- Connect the power from an AC wall outlet using almost any wall wart, using any connector.
    - Power connectors are all different voltages, and some have a negative shield while some have a positive one. The regulator and diode bridge combination makes almost any old "wall wart" power supply acceptable. The power could also come from batteries.
- Alternatively, purchase 5 V wall wart power supplies and their appropriate connectors.
- An alternative power supply: 10 volts, with a voltage divider for 5 V to connect to ground. An All About Circuits posting[46][47] has some ideas using zener diodes to make a positive and negative voltage regulator.

4.2. Instruction Memory

To simplify the design, the system has separate instruction and register memories. Instruction memory is ideally a bank of N triple-throw switches[48] for specific switches; I call this "SWROM" for switch-based

read-only-memory), with poles connected to 1, 0, and $\overline{1}$, labeled as such:

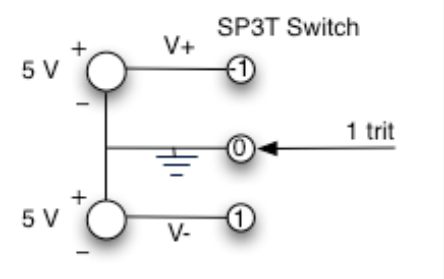


Figure 5. One Trit of Trinary Switch-ROM

The programmer can flip these switches to load a machine-language program into the computer. In the 3x3 SWROM, trits are grouped into 3-trit words, arranged horizontally, allowing 3 words to be programmed into the machine. However, due to time constraints in selecting a proper switch, our "switch ROM" instead is operated by plugging in wires into the appropriate holes of a breadboard.

This memory with three addresses, addressing three trits each for a total of 9 trits, is constructed as follows:

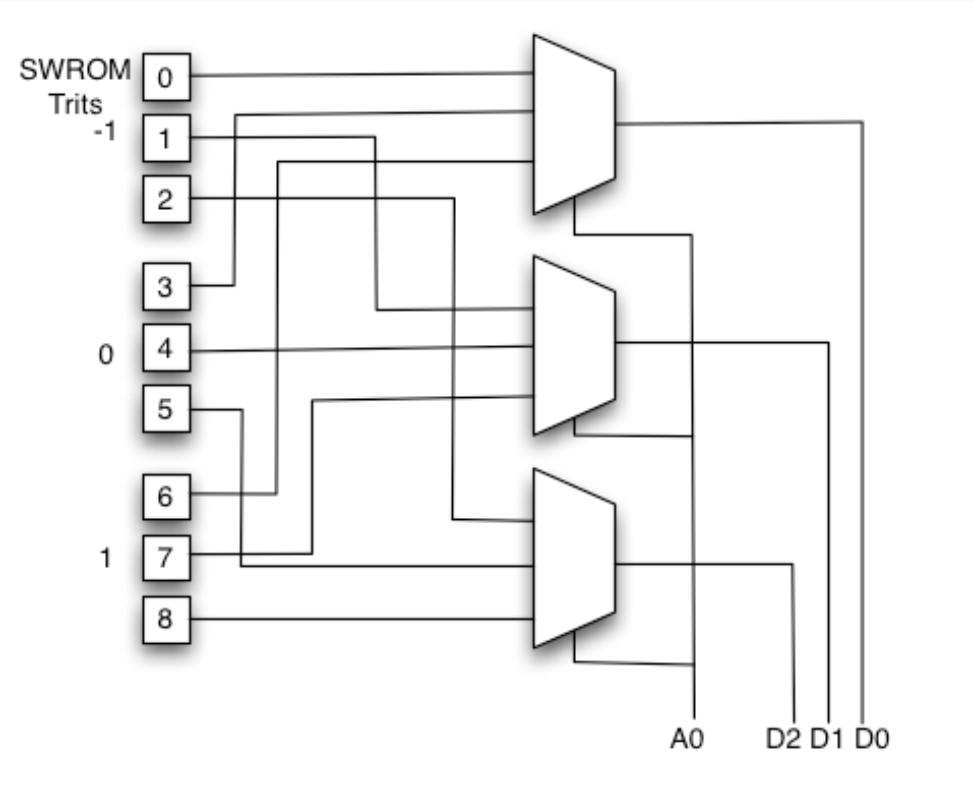


Figure 6. 3x3 SWROM address decoder. The address (A0) determines which triplet of memory cells will be read onto the data lines D2, D1, and D0.

As an example, the following SWROM is loaded with a sequence of trits corresponding to an old version of the assembled guessing game program:

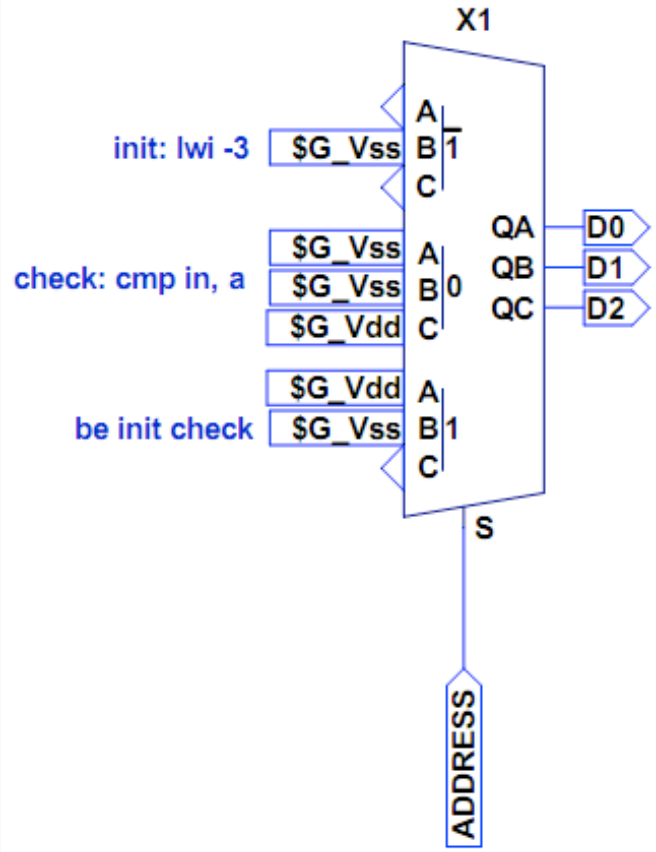


Figure 7. 3x3 SWROM loaded with guess.t trits.

The SWROM behaves as expected during simulation when the address line is given several addresses to read from:

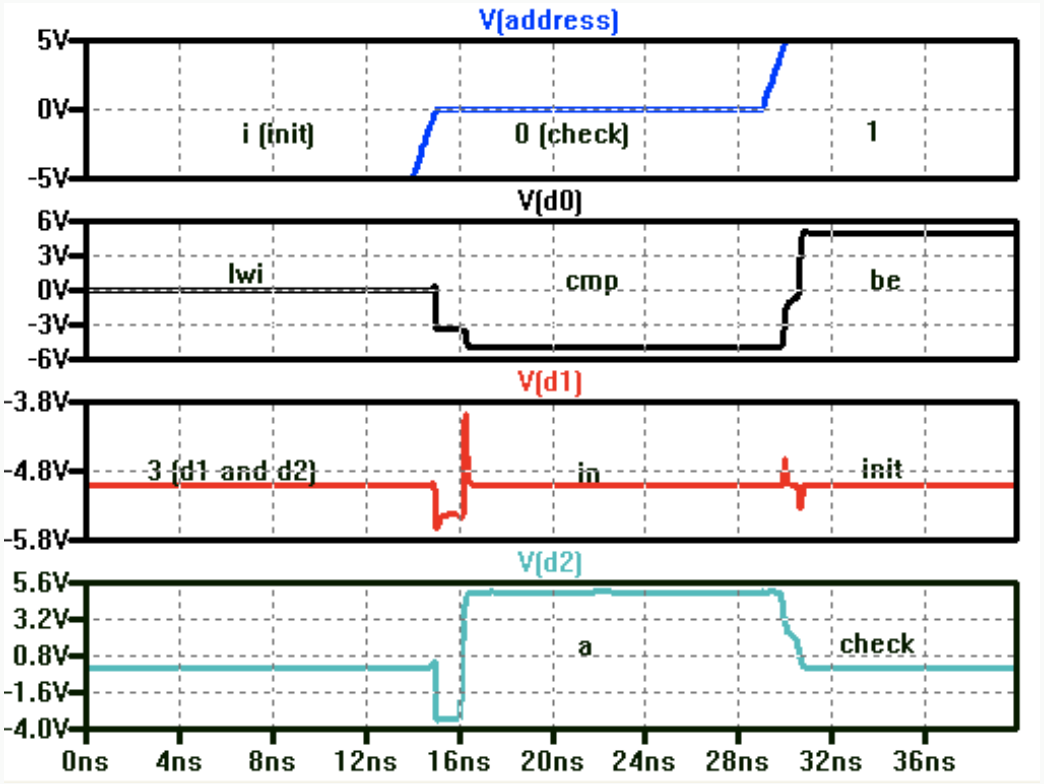


Figure 8. Timing diagram of 3x3 SWROM loaded with guess.t

In the timing diagram above, the first address requested is $\overline{1}$, which outputs trits for the `lwi -3` instruction, as shown. Address 0 outputs `cmp in, a` and address 1 outputs `be init, check`. "init" and "check" are labels that refer to addresses $\overline{1}$ and 0, respectively. Note that since this timing diagram was made, PC has been made to start at 0, and the instructions and labels have shifted around appropriately.

The completed SWROM can be simulated at the transistor level (as it is above), but it takes a few seconds to complete. For faster simulation, I developed `swrom-fast`. This component is a behavioral model that uses B-sources in LTspice, which describe the voltage mathematically rather than using electrical components. It simulates nearly instantaneously, and produces the following timing diagram:

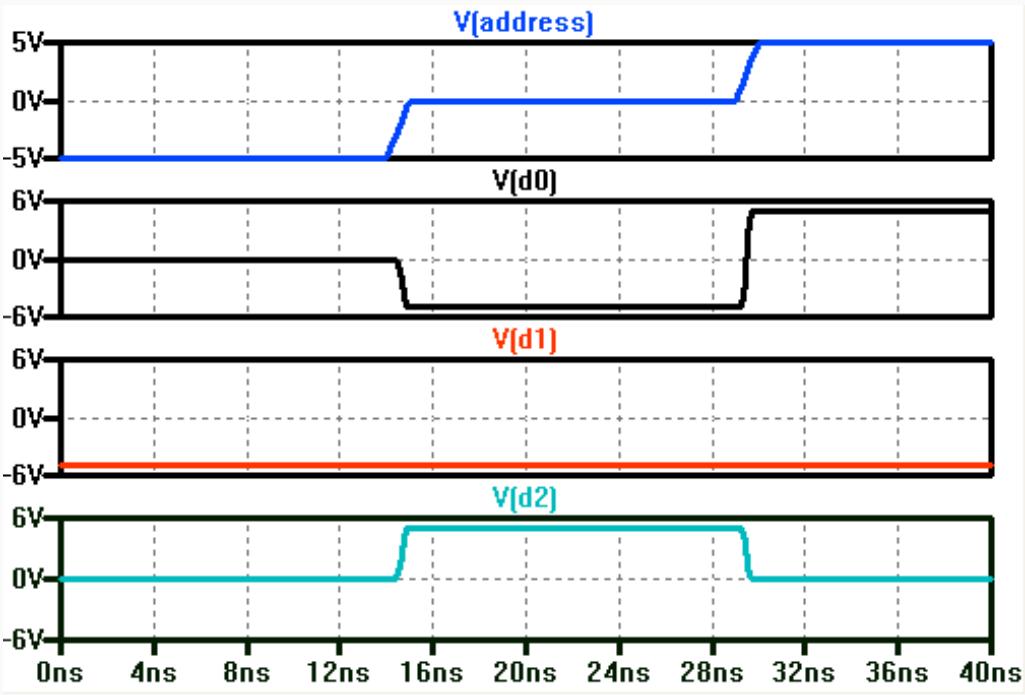


Figure 9. Behavioral model simulation of 3x3 SWROM loaded with guess.t. An ideal model. This is not realistic.

swrom-fast can load arbitrary assembly programs, compiled into an .sp file with `asm/asm.py`.

4.2.1. Experimental Results

Although we designed and constructed a switch ROM printed circuit board, due to time constraints were not able to have a working board with triple-pole switches soldered on to it for easy reprogramming. As stated above, instead the computer can be programmed by manually plugging in wire corresponding to each data line into the appropriate holes of the breadboard: $G_Vdd, ground, and $G_Vss for 1, 0, and $\overline{1}$.

4.3. Program Counter

The *program counter*, or PC, is a 1-trit rising-edge triggered master-slave D-type tri-flop register (refer to section **D.5.6** for detailed information on this component) initialized to 0. The PC cycles through 0, 1, $\overline{1}$ using the cycle up gate. Note that PC does *not* start up at $\overline{1}$ and sequence through $\overline{1}$, 0, 1, but rather 0, 1, $\overline{1}$. Program execution starts at 0 instead of $\overline{1}$ because registers start at 0 on power-up, and it would require additional

hardware to initialize it to $\overline{1}$ (although asynchronous resets would be particularly useful to tie to a global RESET signal, such a feature is out of the scope of this project).

4.4. Clock Generator

TCA2 requires a two-phase clock, with signals named FETCH and EXECUTE, each the inverted version of each other.

The `clock_gen-fast` component includes an LTspice PULSE voltage source with a fixed period and a 50% duty cycle for EXECUTE, and an inverter to generate FETCH. It is so named because it simulates quickly since the signals are generated by the simulator using mathematical expressions, rather than electrical models of physical components.

`clock_gen` contains a circuit with the NE555 timer integrated circuit to generate a similar pulse to the faster behavioral model. The period and duty cycle do not exactly match `clock_gen-fast` but it is close enough to allow for visually similar simulation results to `clock_gen-fast.asc`. The output is a square wave from -6 V to +6 V. It does not stop intermediately at 0 V as one might expect. The period is slightly more than 10 μs:

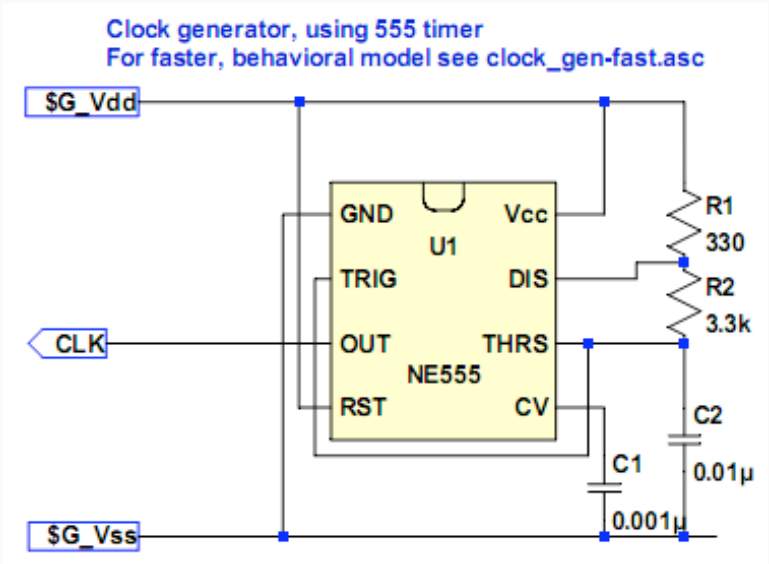


Figure 10. 555-based clock generator (clock_gen.asc).

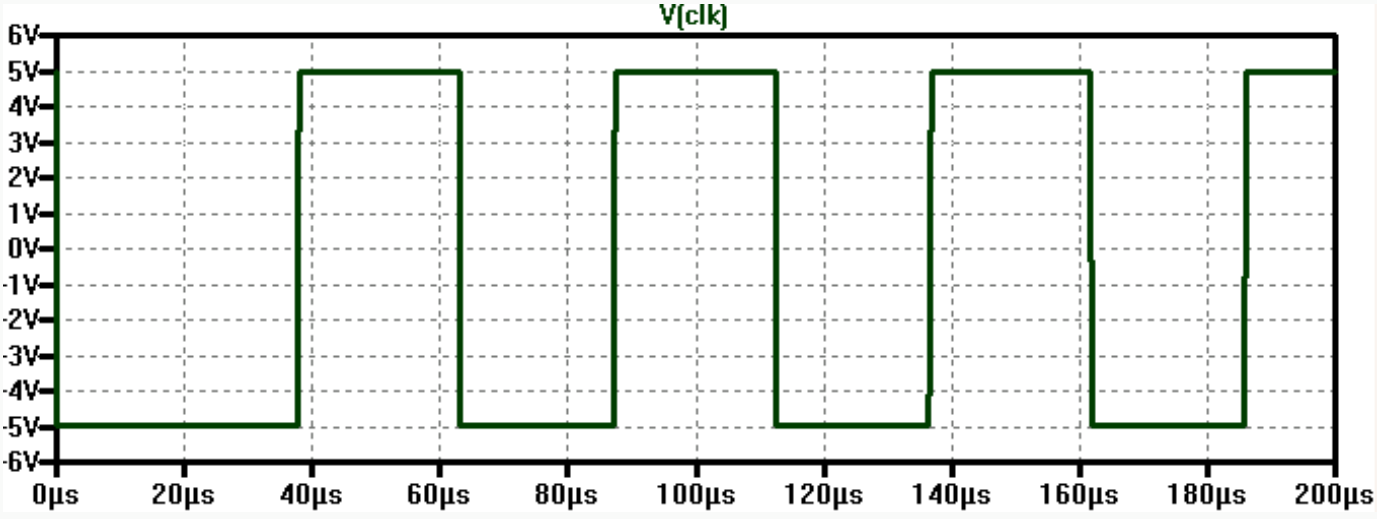


Figure 11. Timing diagram of clock generator.

Tthe 555-timer is too complex to simulate within the full TCA2 architecture in a reasonable timeframe (we left it running for 48 hours and it did not progress past the initialize phase), so `clock_gen-fast` was used in that architecture. Since TCA0 is a simpler architecture, `clock_gen` is integrated to provide a more accurate simulation.

4.5. Processor

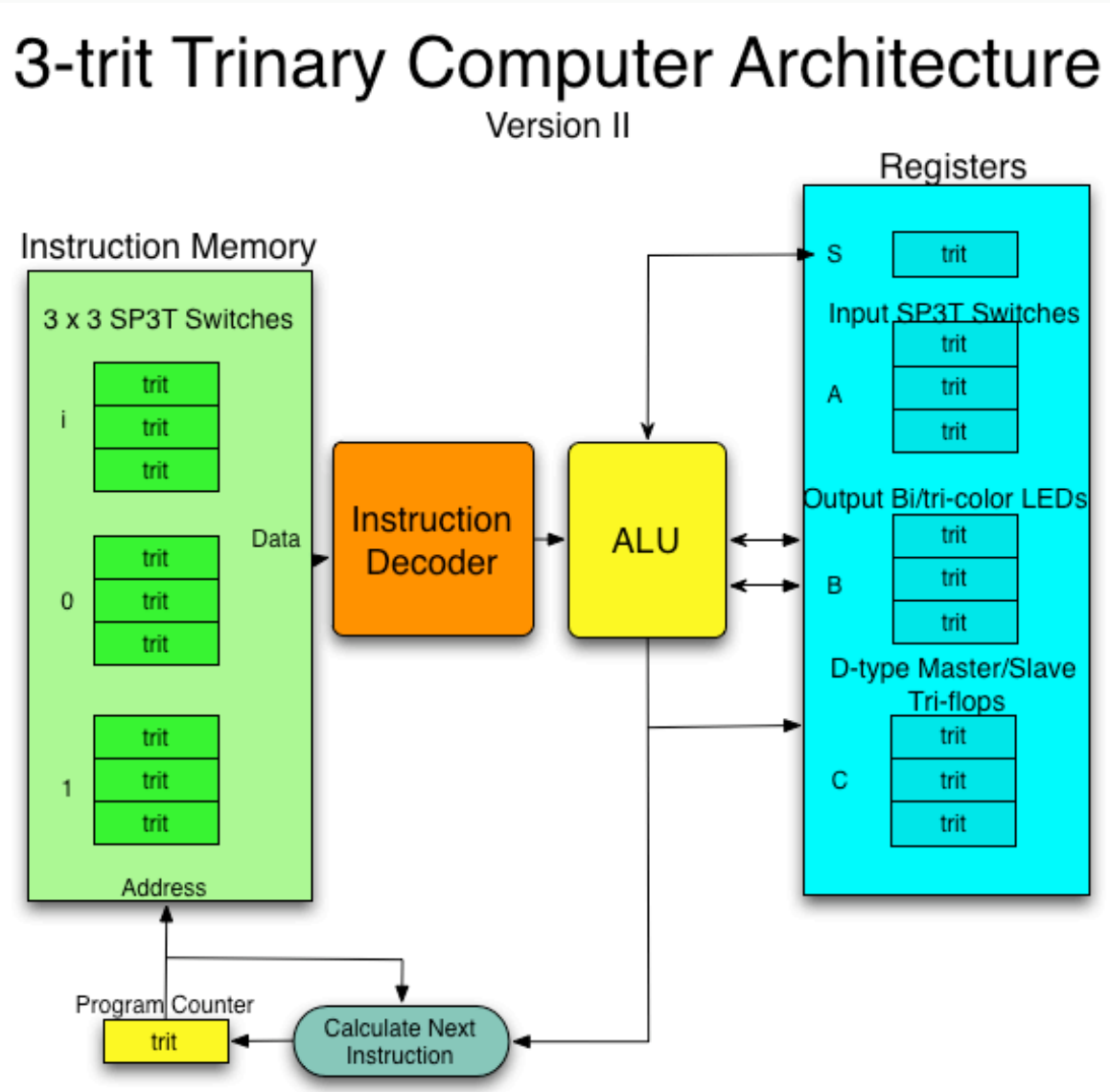


Figure 12. High-level block diagram of 3-trit trinary computer architecture 2, in color (TCA2). Older diagrams[49] are available.

The basic design is a trinary ROM configured by an array of switches, connected to an address decoder. The program counter initializes to 0 on startup and increments with each cycle. The PC is sent, as the address, to the memory bank, and output of the memory is fed to an instruction decoder that controls the proper signal outputs to execute the instruction.

4.5.1. Registers

TCA2 has a set of three registers, addressed by a single trit value:

- $\overline{1}$ - input register IN, wired to 3 switches
- 0 - output register OUT, wired to 3 LEDs
- 1 - general-purpose accumulator register A, 3 trits, latches, also wired to LEDs

We decided on having three registers since three is the least number that can be represented using one trit. Additionally, the status trit, *S*, holds the numerical sign of the last operation. It is set by the CMP instruction and indirectly accessed by the BE branch instruction.

TCA0 only has the general-purpose register: the accumulator, known simply as "A".

4.5.2. Input and Output

We researched a myriad of LEDs and switches to use for I/O[50]. Triple-throw switches were originally planned for TCA2 input, but in TCA0 inputs can be manually wired to positive, ground, and negative by connecting the input wire to the appropriate pins on the breadboard.

For output, we purchased 2-, 3-, and 6-pin bi-color and tri-color LEDs for experimentation. During testing, an oscilloscope can be used to view the output, however, visual output is desirable. In a binary digital system, an active LED is often used to indicate a 1, while 0 is indicated by off. In a trinary digital system, either three colors (red, green, blue) can be used to indicate each of the three states, or two states (red-orange, green) can be represented by two colors and the third state (0) by off.

A 3-pin bi-color LED may appear to be the ideal solution, but the one we purchased had a common cathode and separate anodes, making additional circuitry necessary to translate a trinary voltage level to the LED output:

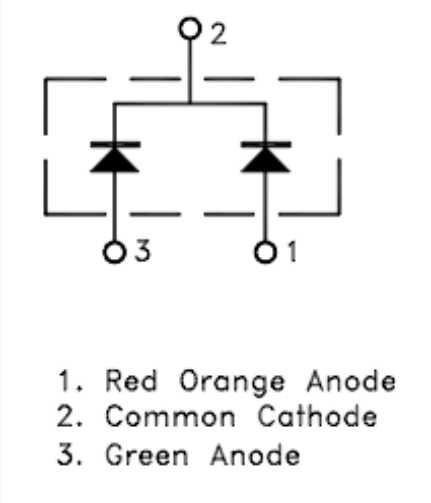


Figure 13. 3-pin LED Pinout[51].

The 6-pin LED offers three colors and a promising pinout:

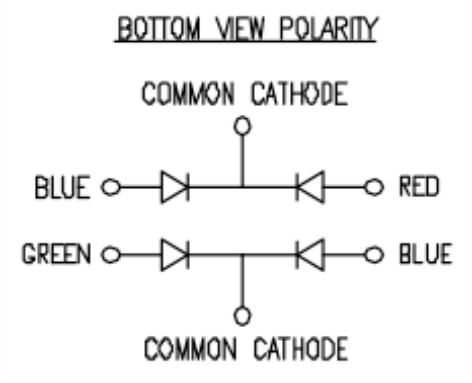


Figure 14. 6-pin RGB LED Pinout[52].

The bottom pair could be used as follows. Blue would be grounded, the common cathode would be left floating, and the green terminal would be used for input. A zero input would leave both LEDs off, a positive input would forward-bias the green LED turning it on, while a negative input would forward-bias the blue LED, turning it on. However, implementation is complicated by the maximum forward voltages required by the LEDs. Green has a maximum of 2.5 V, 2.1 V typical, while blue has a maximum of 4.2 V and 3.65 V typical. A 2:1 resistive voltage divider could be used to bring the nominal 5 V down to 2.5 V, but it is not known whether the blue LED will function at a low 2.5 V. Additional, complicated circuitry would also be needed to use all three colors in this expensive, 6-pin LED.

A third option is a humble 2-pin LED, which offers the perfect reverse-parallel configuration:

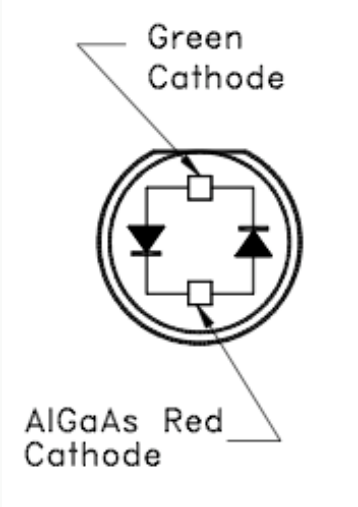


Figure 15. 2-pin LED Pinout[53].

The red has a 40 mA absolute maximum forward current, while the green is 30 mA. Forward voltage for red is 1.8 V typical, 2.4 max, and green 2.1 V typical and 2.6 V max, at 20 mA forward current. Therefore a voltage of ±2.1 V and a current of 20 mA will cause the LED to function as expected:

Table 7. Bicolor LED Output Colors

Logic Input	Color Output
$\overline{1}$	Green
0	(off)
1	Red

According to a current limiting resistor calculator[54], for a 5 V supply and a 2.1 V drop across the LED along with a desired 20 mA current, the current limiting resistor should be 150 Ω.

In summary, to build an LED output circuit, one would connect the logic input signal to a 150 Ω resistor in series with the 2-pin bi-color LED. We built such a circuit and confirmed that it correctly lit the LED as each logic level input was applied.

4.5.3. 3-Trit Instruction Set

Although a *One Instruction Set Computer* (OISC) with a "subtract (balanced ternary) and branch if negative" (subneg) operation is Turing complete and could be used to implement any program[55], for simplicity and ease of debugging we instead decided to implement three separate opcodes for each of the operations, as follows:

- $\overline{1}$xy - *cmp*, compare register *x* with register *y*, store status trit in *S*
- 0xx - *lwi*, load immediate value *xx* into register 1 (A, accumulator)
- 1xy - *be*, branch to immediate address *x* if S = 0 (previous comparison indicated the two values were equal), otherwise branch to immediate address *y*

The motivation for using precisely the above instructions is that a trivial guessing game can be written as follows (this file is available in the git source code repository as `asm/guess.t`):

```
init:  lwi -3         ; random number to guess
check: cmp in, a      ; did they guess right?
       be init, check ; re-initialize if correct, loop if not
```

Using the assembler in `asm/asm.py`, this program assembles to the tritstream `asm/guess.3`:

```
1010T0TT1
```

The tritstream dumps the contents of memory at addresses $\overline{1}$, 0, and 1. The program counter begins at 0, so the first instruction, `lwi -3`, assembles to $\overline{1}0\overline{1}$ in the middle of this stream, at address 0, label `init`. The next instruction at address 1, label `check`, is `cmp in, a` which assembles to $\overline{11}1$. The last instruction wraps PC around to $\overline{1}$ and assembles `be init, check` to 101, at the beginning of the tritstream. Hence, this 3-trit instruction set allows for implementation of a guessing game as required.

4.5.4. LWI Instruction Example (also known as TCA0)

In implementing the architecture, I took the approach of building it incrementally, beginning with the load-word-immediate instruction. This instruction is the easiest to implement because it merely activates a clock signal on the A register, to load it with the last two trits of the instruction. Because it only supports this one instruction, this architecture is known as TCA0. A schematic of the architecture is as follows:

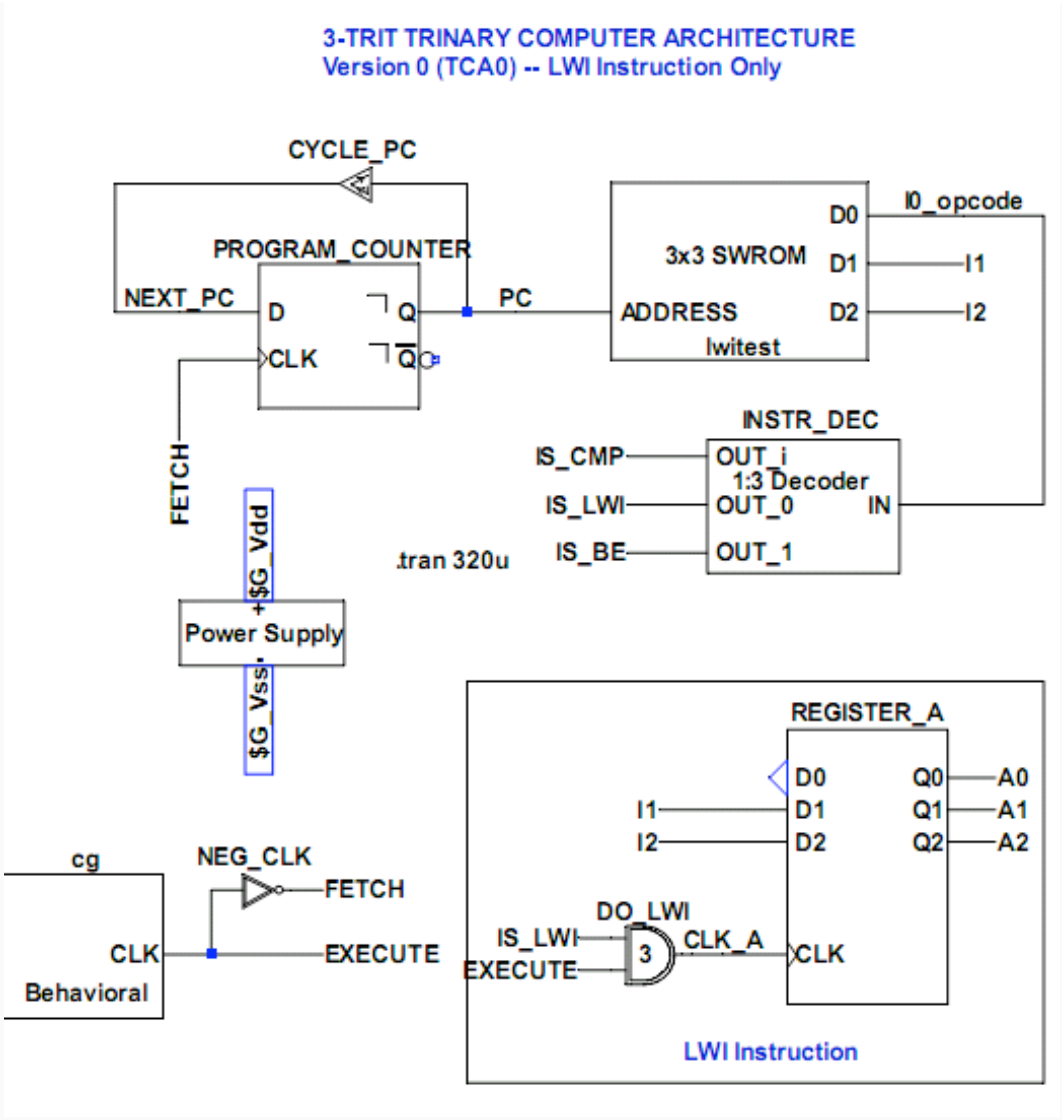


Figure 16. `main_lwitest.asc`**, an architecture that supports the LWI instruction.**

Each of the components shown above have been designed ultimately from transistors, as described in detail in appendix **E**.

The computer operates as follows. The clock generator (section **4.4**) emits a regular series of pulses. In the beginning, the PC register (a rising-edge triggered master-slave tri-flop, see section **D.5.7**) starts at 0, which reads address 0 from the switch ROM. The SWROM is loaded with the following code assembled from the source file`asm/lwitest.t`:

```
lwi -3
lwi -2
lwi 0
```

Therefore, address 0 causes the SWROM to output the signals for `lwi -3` (of which the machine code is $0\overline{11}$). Hence, the instruction signals are at the following logic levels:

Table 8. Logic Levels of TCA0 Instruction Signals Shortly After Power-On

Signal	Logic Level
I0_opcode	0
I1	$\overline{1}$
I2	$\overline{1}$

The I0_opcode feeds into the *instruction decoder*, a *3:1 decoder* (section **D.8**) that translates the opcode into a set of control signals. In this case, I0_opcode is 0, corresponding to the lwi instruction, so the control signals are set as follows:

Table 9. Logic Levels of TCA0 Control Signals Shortly After Power-On

Signal	Logic Level
IS_CMP	$\overline{1}$
IS_LWI	1
IS_BE	$\overline{1}$
EXECUTE	1

As shown above, the instruction decoder outputs $\overline{1}$ for inactive signals, and 1 for active signals. Since the instruction is lwi, IS_LWI is active. This signal feeds into the DO_LWI TAND gate (section **D.4.4**), thus causing CLK_A to be 1 since EXECUTE is also 1 (at power-up). Therefore, the REGISTER_A register loads the values of I1 and I2, $\overline{1}$ and $\overline{1}$ respectively, into the A register. The *lwi -3* instruction has completed, and -3 ($\overline{11}$) has been loaded into the A register.

At the next clock cycle, FETCH is 1 and EXECUTE goes to 0. The rising edge of FETCH causes the program counter register to load the contents of NEXT_PC. NEXT_PC is the output of the program counter when connected to a *cycle up gate* (section **D.3.4**) that increments the 1-trit number. This causes PC to increment from 0 to 1, and the instruction at this address (lwi -2) is executed as EXECUTE goes to 1. In the next cycle, PC is incremented again and wraps around to $\overline{1}$, and the lwi 0 instruction executes. A timing diagram has the details:

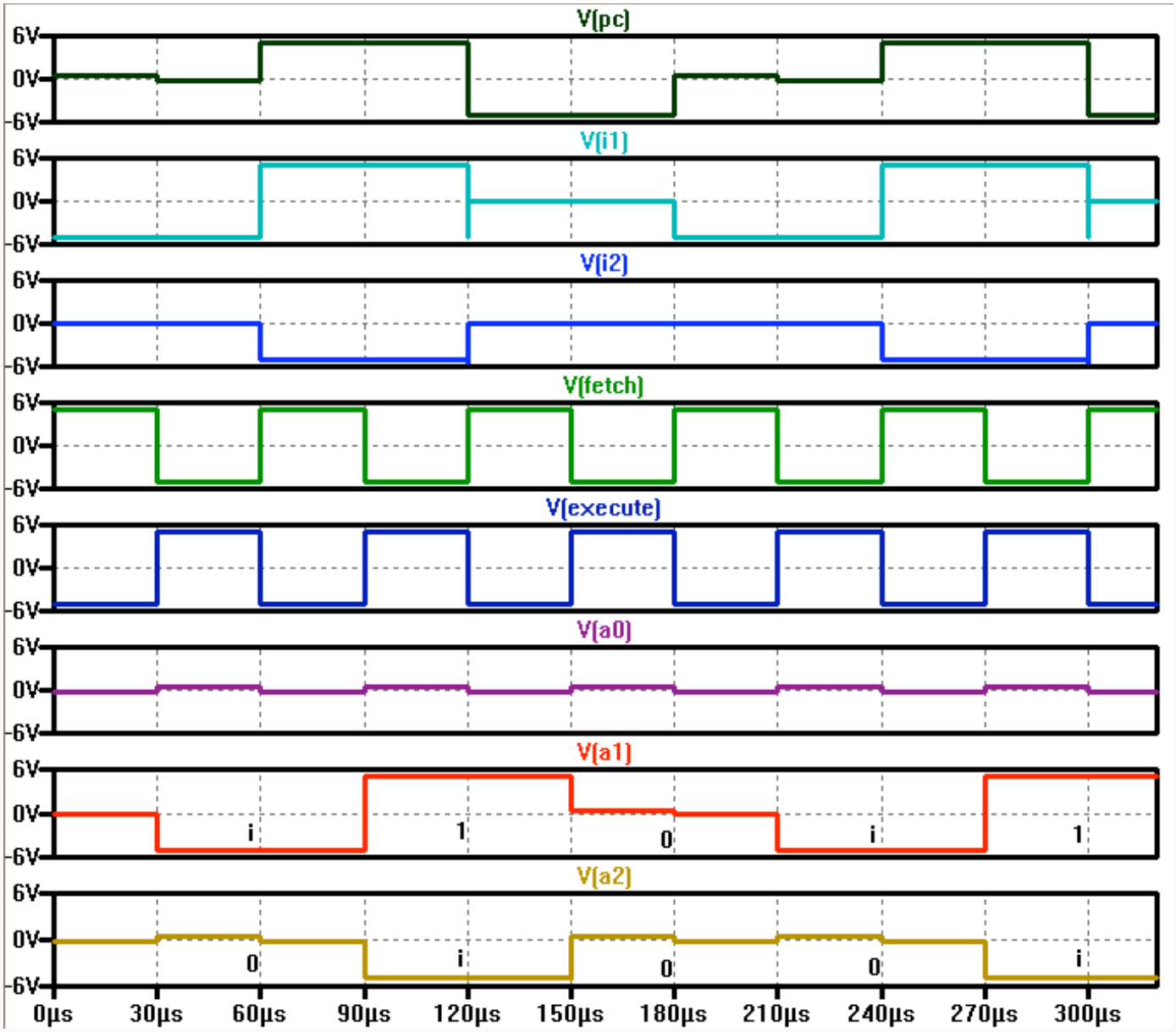


Figure 17. Timing diagram of architecture demonstrating LWI instruction, loading -3, 2, then 0 in a loop ($\bar{1}0$, $1\bar{1}$, 00), from lwitest.t, running `main_lwitest.asc`.

The last three plots show the contents of the accumulator ("A" register). Note that `lwi` only specifies the lower two trits to load; the upper trit is hardwired to 0 in this example.

The above timing diagram uses a SPICE voltage pulse for the clock generator, but because TCA0 is much simpler than TCA2, it is possible to simulate using the clock generator built using the 555 IC. Execution begins at PC = 1 instead of 0, and the frequency is slightly different, but the operation of the instructions is the same:

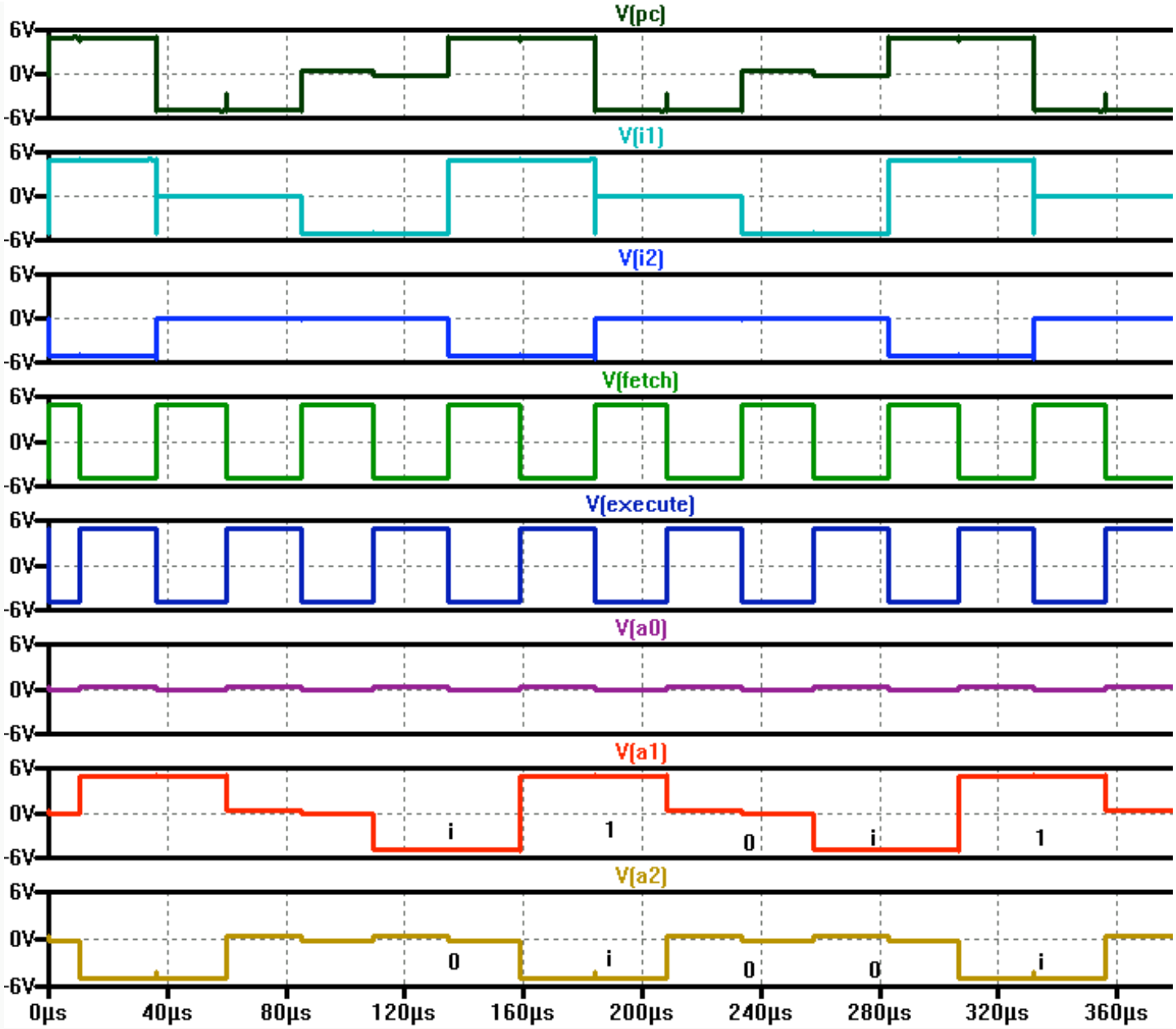


Figure 18. Timing diagram of architecture as above, except using a clock generator based on a 555 timer circuit.

This architecture, although limited, is sufficient to execute a "Christmas lights game", where the player determines the sequence of colors to emit. The output of the A register, as controlled by the `lwi` instruction, is wired to a series of multi-color LEDs. By changing the programming, the player can experiment with a variety of light sequences. While the game would be more exciting with a larger number of LEDs (ideally, enough to string up a medium-size Douglas-fir) than the A register can represent, this game demonstrates the programmability of the TCA0 architecture.

4.5.5. CMP Instruction

The *compare* instruction sets the status bit based on the comparison of two registers, as follows:

Table 10. Status Trit and Condition Codes

Condition	S
R1 < R2	$\overline{1}$
R1 > R2	1
R1 = R2	0

In order to do so, an arithmetic *subtraction* operation must be performed. To do so, an *arithmetic logic unit* is used.

4.5.5.1. ALU

The ALU is a large component consisting of an inverter, 4-trit ripple-carry adder, and a sign detector. As an example, the following inputs A and B cause the following outputs S to occur:

Table 11. ALU Test Cases / Examples

Time (ns)	A	B	A	B	Difference	Difference	S	Meaning
0	000	000	0	0	0000	0	0	=
10	001	001	1	1	0000	0	0	=
20	0$\overline{1}$1	0$\overline{1}$1	-2	-2	0000	0	0	=
30	1$\overline{1}$1	0$\overline{1}$1	7	-2	0100	9	1	>
40	1$\overline{1}$1	$\overline{11}$1	7	-11	1$\overline{1}$00	18	1	>
50	$\overline{11}$1	$\overline{11}$1	-11	-11	0000	0	0	=
60	$\overline{111}$	$\overline{11}$1	-13	-11	00$\overline{1}$1	-2	$\overline{1}$	<
70	$\overline{111}$	$\overline{111}$	-13	-13	0000	0	0	=
80	$\overline{1}$0$\overline{1}$	0$\overline{1}$0	-10	-3	0$\overline{1}$1$\overline{1}$	-7	$\overline{1}$	<

The ALU is built from these operations, in sequence:

- **Negation**: a bank of Simple Ternary Inverters (section **D.3.2**) on the R2 input.
- **Addition**: implemented using an 4-trit ripple carry adder (section **B.1.6**). Since R2 is inverted, this is equivalent to R1 - R2 (subtraction).
- **Sign checking**: the S trit is set to the most-significant non-zero trit of the difference, or 0 if it is zero, using a *sign detector circuit* (section **B.1.2**).

A schematic of the ALU is as follows:

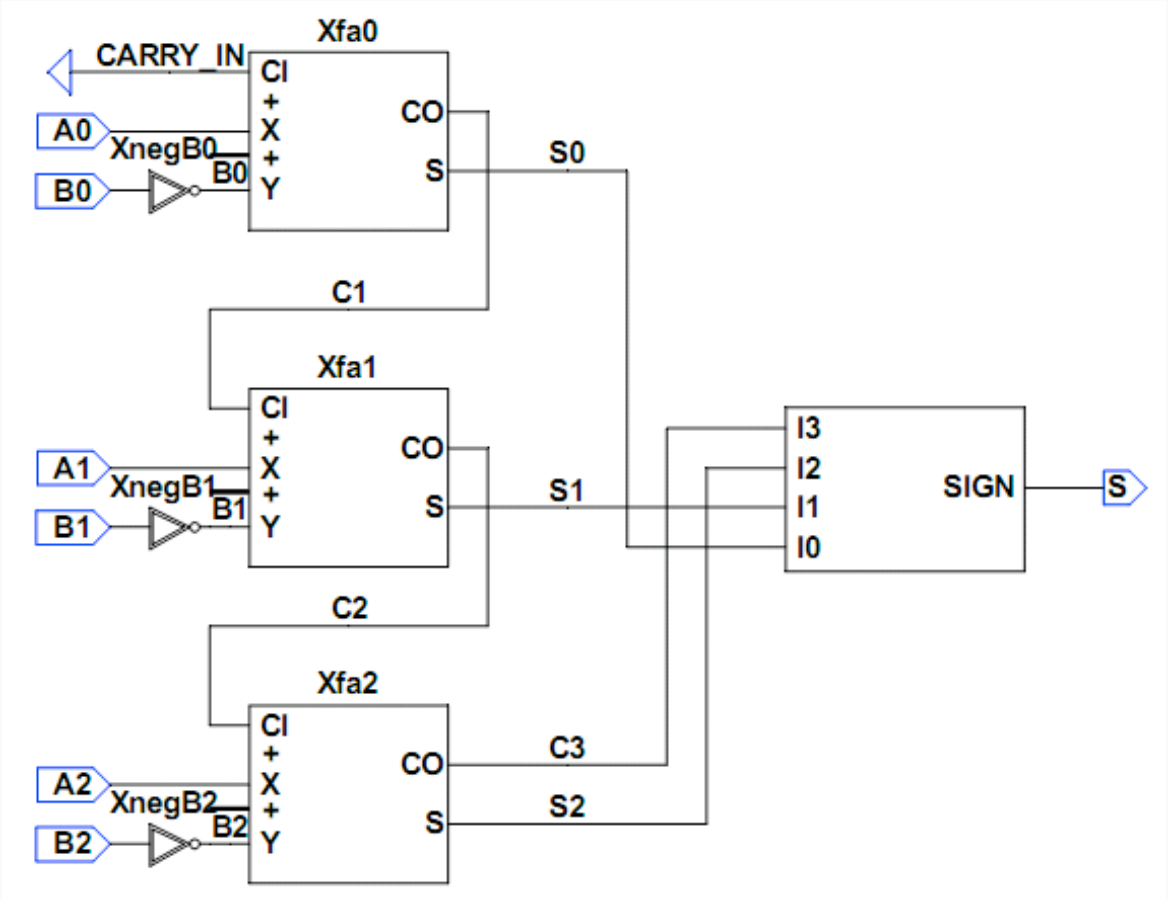


Figure 19. Arithmetic Logic Unit for TCA2.

The timing diagram matches the behavioral model, but it is executed at an order of magnitude slower to allow the carries in the ripple carry adder to propagate:

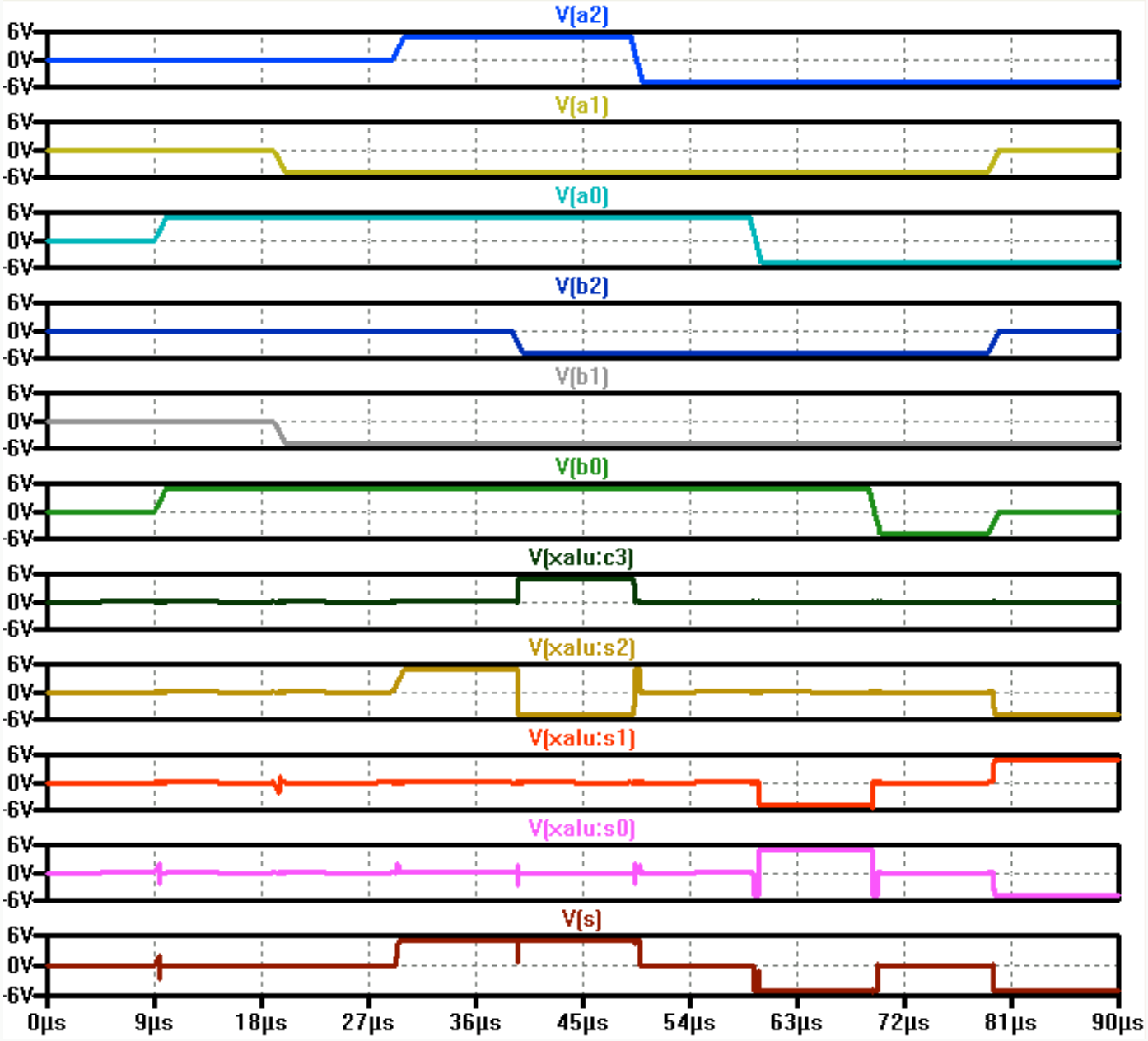


Figure 20. Timing diagram for `alu`, showing several cases.

I also implemented a behavioral model of the ALU, quickly computing the comparison result using a mathematical expression, for comparison purposes. The timing diagram below shows the timing diagram ALU (excluding input signals):

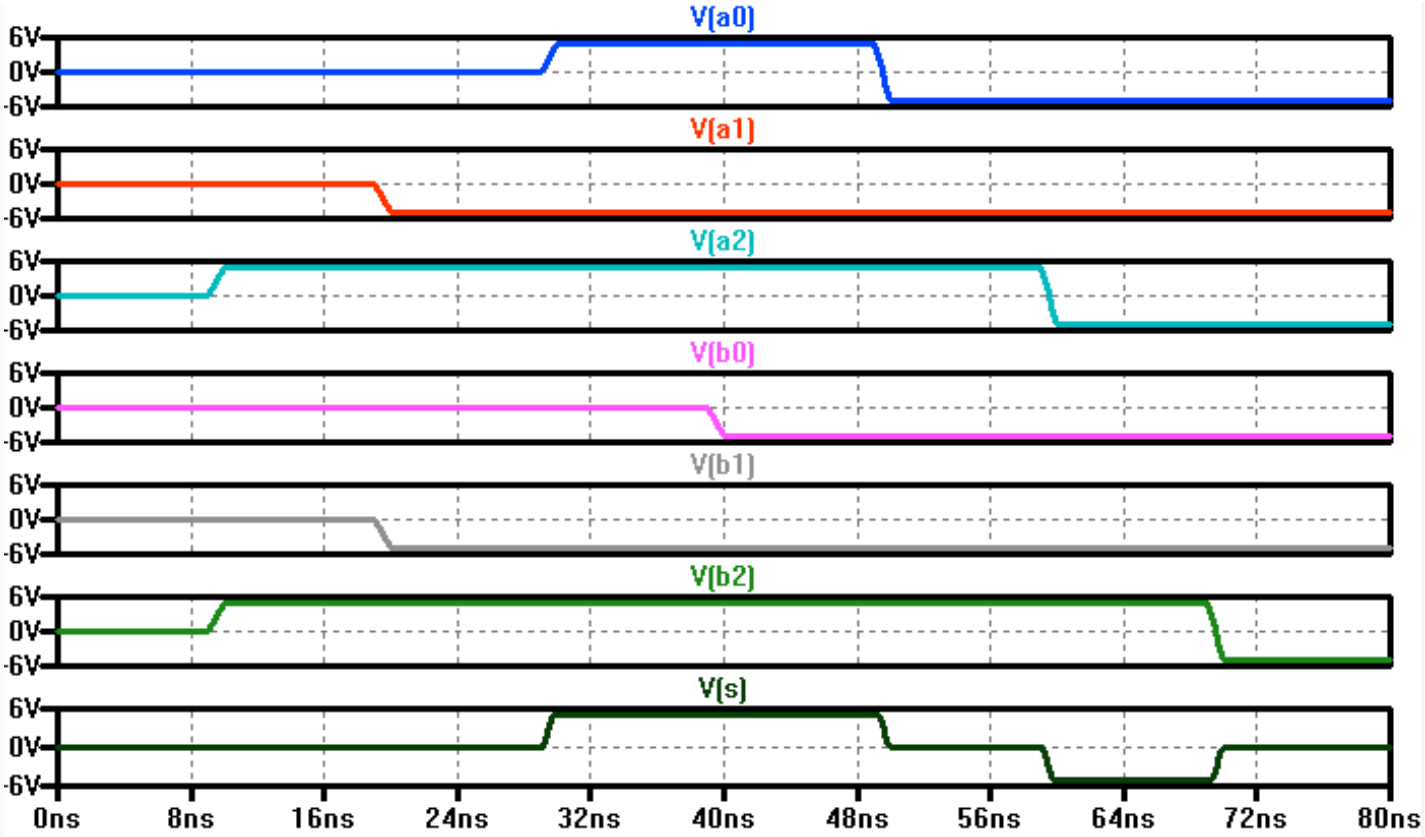


Figure 21. Timing diagram of `alu-fast`, with certain test cases.

4.5.5.2. CMP Instruction Example

To integrate the ALU with the remainder of the computing architecture, I designed the following circuitry:

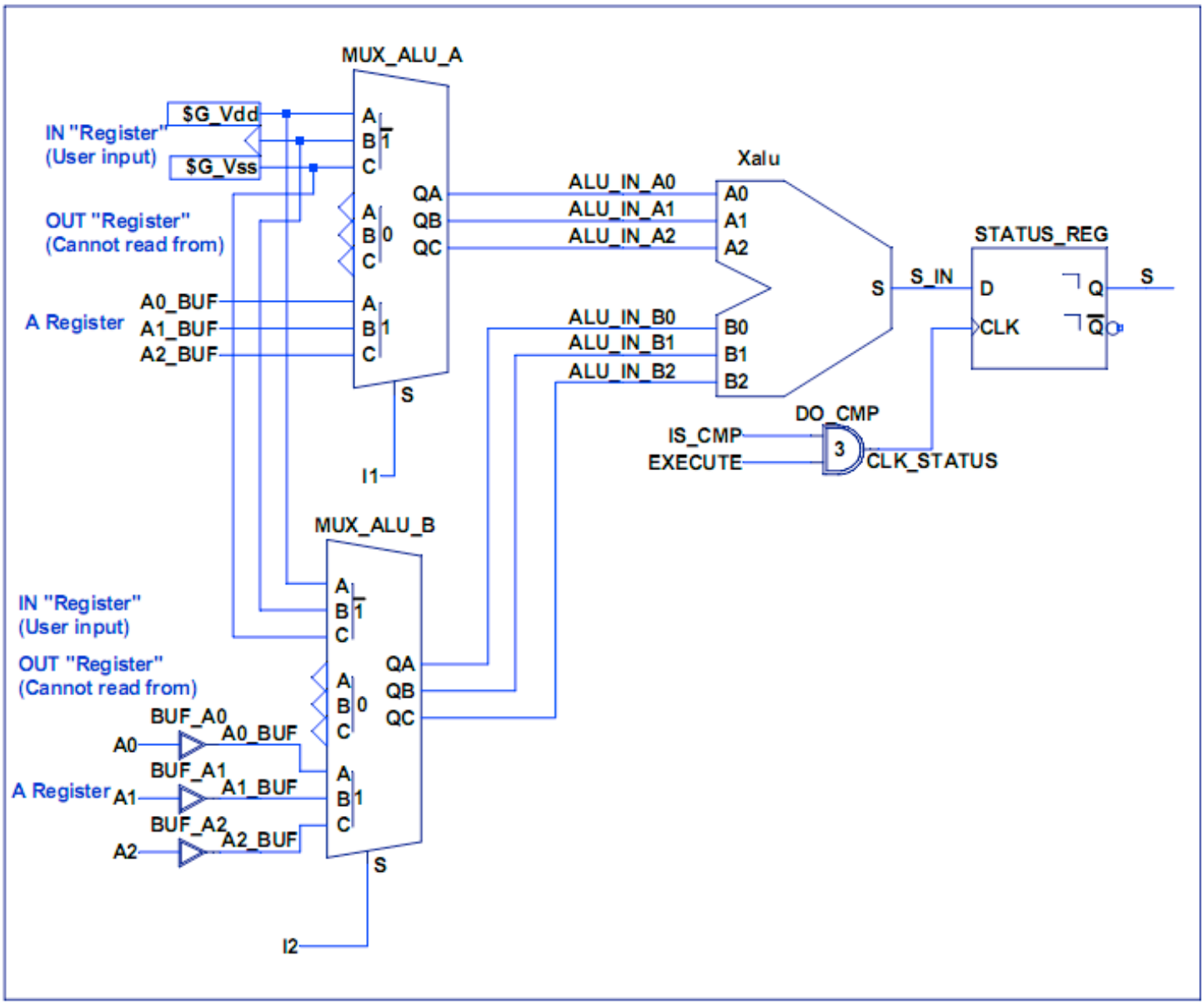


Figure 22. Part of the architecture that handles the CMP (compare) instruction.

The multiplexers (section **D.8**) select which inputs to compare, and the contents are then fed to the ALU. The sign result is sent to the status register, clocked in during the EXECUTE phase if the current instruction is a compare instruction, in a manner similar to how the lwi instruction clocks in input. The above circuitry can be dropped in in-place into TCA0 to add the CMP instruction, since the IS_CMP control signal is available.

An assembly program named `asm/cmptest.t` was written to test this architecture. It is as follows:

```
; Test cmp (compare) instruction and lwi (load word immediate)
lwi -3          ; load A with 0i0
cmp in, a       ; compare A to IN (probably 10i)
cmp a, in       ; now S should be opposite
```

As an example, this assembly program was ran on the architecture with the IN register hardwired to $10\overline{1}$ (8). The timing diagram is as follows:

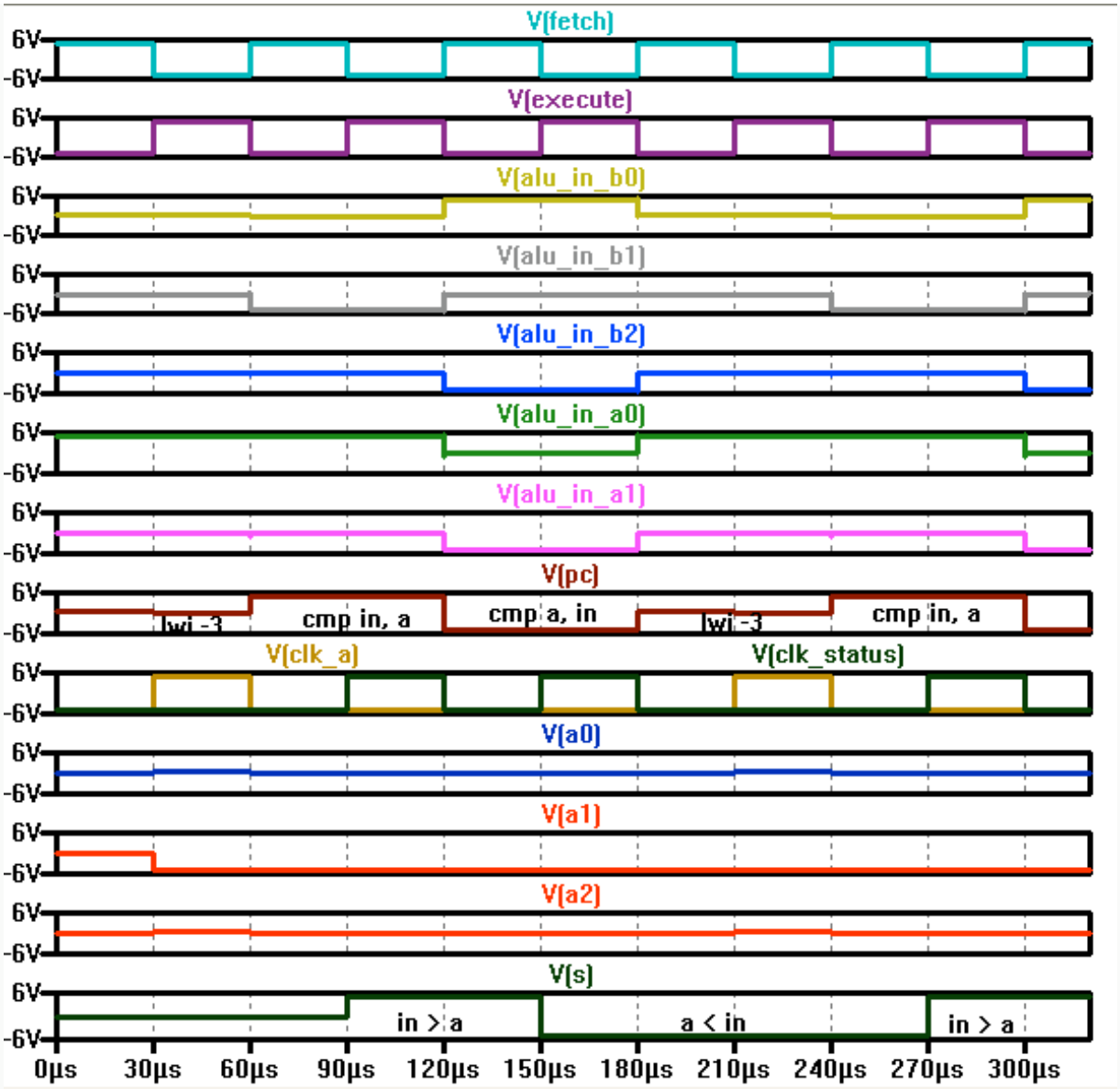


Figure 23. Timing diagram of running cmptest.3, with IN = 10$\bar{1}$ (8).

First A is loaded with 0$\bar{1}$1 (-3), then IN is compared to A. Since 8 > -3, the status trit, S, is set to 1 for "greater than".

Next, the opposite comparison is performed, compare A to IN. Since -3 < 8, S is set to $\bar{1}$ for "less than". The status trit remains unchanged during the next execution of `lwi`, but is set to 1 on the next `cmp in, a`.

Two major changes were needed to the architecture to support this instruction:

- All registers were changed from Mouftah's master-slave tri-flop (section **D.5.6**) to a custom edge-triggered tri-flop design (section **D.5.7**).
- Accumulator outputs were buffered (see section **D.3.2**) before connecting to the 9:3 multiplexers. Without buffering, the accumulator register would undergo undesired changes as the output feeds back to the input from the multiplexers. This can happen because the multiplexers (section **D.8**) operate using *transmission gates*—bidirectional CMOS switches.

4.5.6. BE Instruction

The final instruction to implement for a complete system is *branch-if-equal*. This instruction takes the machine code format `1xy`. If the status trit of the system is 0 (meaning the last comparison instruction resulted in "equal"), the instruction causes the system to jump to the first immediate address given (x, available in the I1 signal), otherwise to jump to the second immediate address given (y, I2).

Changes to NEXT_PC signal are necessary to support this instruction. NEXT_PC is now connected to the output of a 3:1 multiplexer, rather than the output of a cycle up gate (which incremented the program counter by one on every FETCH cycle). The MUX_PC multiplexer allows for the IS_BE control signal to switch between the jump address JUMP_ADDR, and the next address in the incrementing sequence, PC_PLUS_1. Additionally, JUMP_ADDR is connected to the output of another multiplexer, JUMP_MUX, that selects the address to jump to based on the result of the status trit. Essentially, the BE instruction changes what is loaded into PC next on the FETCH rising edge:

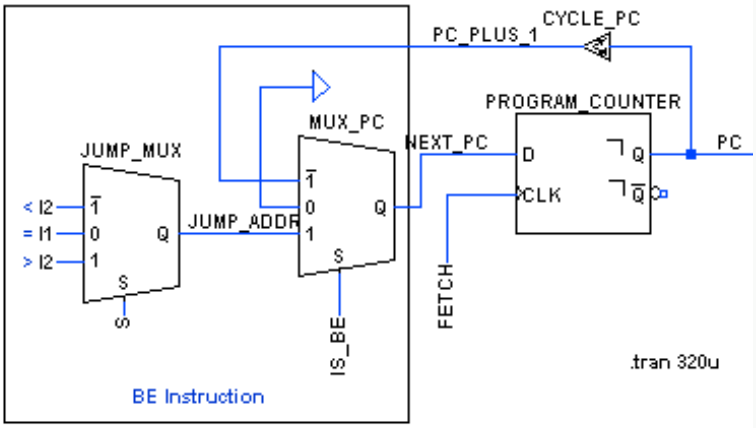


Figure 24. Part of the architecture that supports the BE instruction.

4.5.6.1. BE Instruction Example

For fast simulation, a new architecture, available as `main_jmptest.asc`, was designed to only implement the LWI and BE instructions. A simple test program was written as follows:

```
start: be end, end
skipped: lwi 4
end: lwi -4
```

The second instruction (at address 1) is skipped over by the first instruction (at address 0), jumping to the last instruction (address $\overline{1}$). The accumulator is loaded with 4 instead of -4, and PC cycles from 0, 1, 0, 1, skipping over $\overline{1}$:

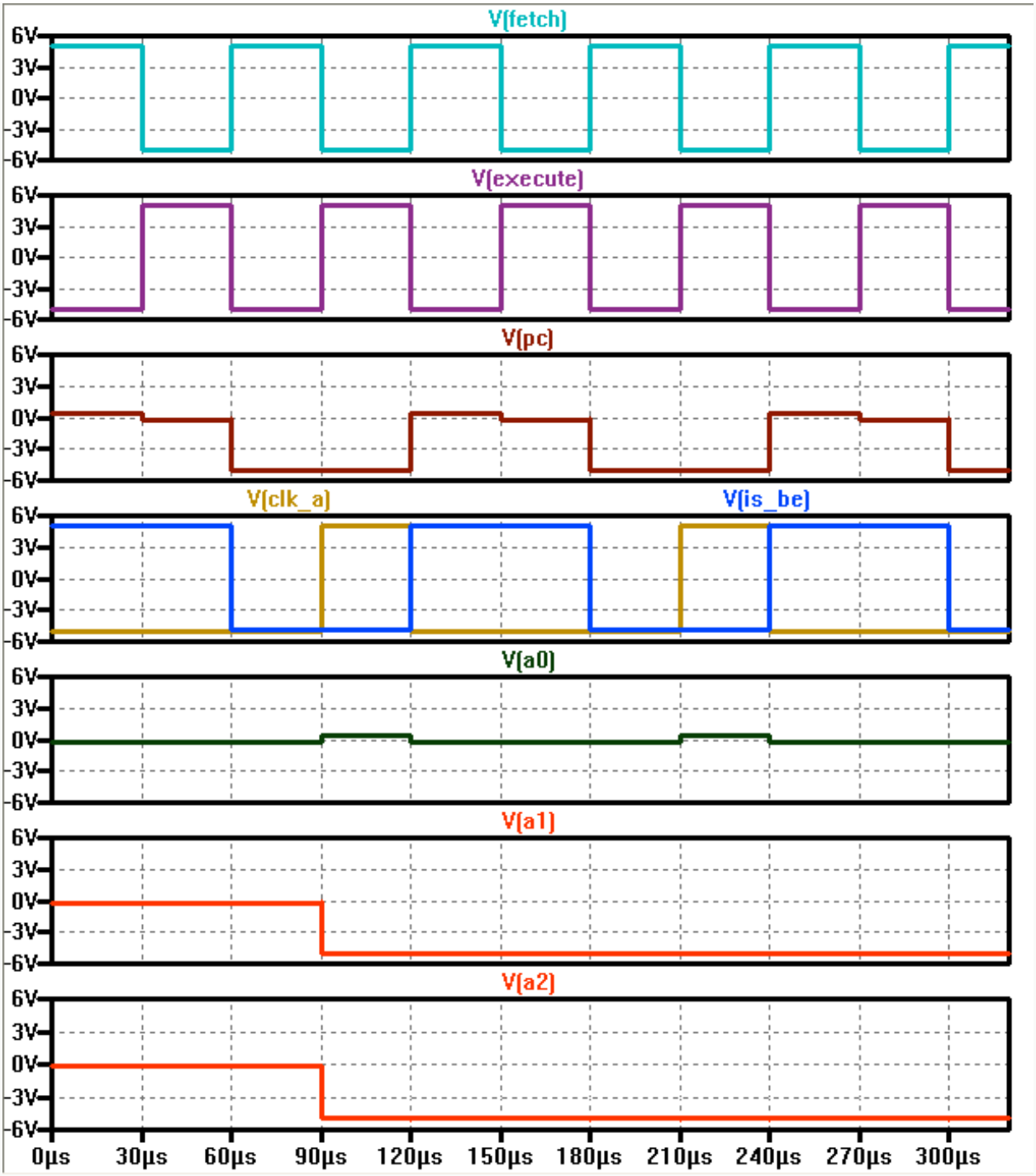


Figure 25. Example timing diagram of using the BE instruction unconditionally.

4.5.7. Guessing Game Program

Putting it all together, the Trinary Computer Architecture v2 (TCA2), supporting the LWI, CMP, and BE instructions to run the guessing game, is as follows:

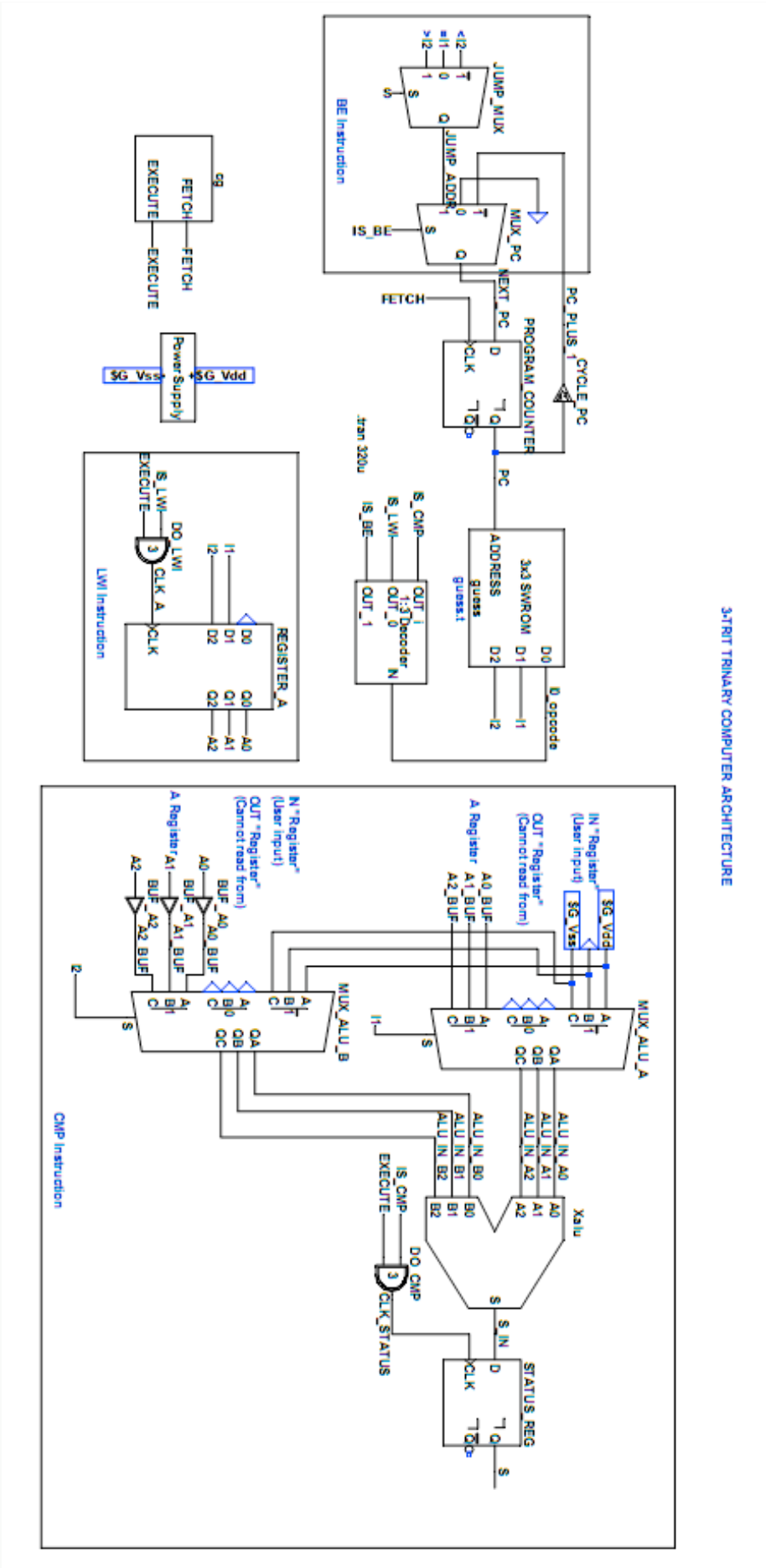


Figure 26. 3-trit Trinary Computer Architecture v2.0, with guessing game loaded.

This circuit is available as `main.asc` in the git repository, circuits directory[56] for simulation. Note that full simulation can take quite a while because the complete system is modeled at the transistor-level. On a 2.4 GHz Intel Core 2 Duo with 2 GB 667 MHz DDR2 SDRAM (MacBook Pro running Windows XP in VMWare), main.asc takes about 30 minutes to simulate.

This complete architecture requires 161 chips: 42 x CD4016, 119 x CD4007, in addition to 340 resistors. For four transistors per CD4007 (two complementary MOSFET pairs are used), 6 per transmission gate (two per gate, but four for the two inverters) times 4 per chip = 1,484 transistors total, about 2/3 of the Intel 4004 processor[57] with 2,300 transistors. However, the TCA2 is significantly less complex than the 4004.

In this example, 8 is guessed ($10\overline{1}$, hardcoded on the user input line) when -3 ($0\overline{1}0$) is correct. 8 is greater, so the status line is high. $8 \neq -3$, so the program keeps looping over `cmp in, a`, `be init, check`:

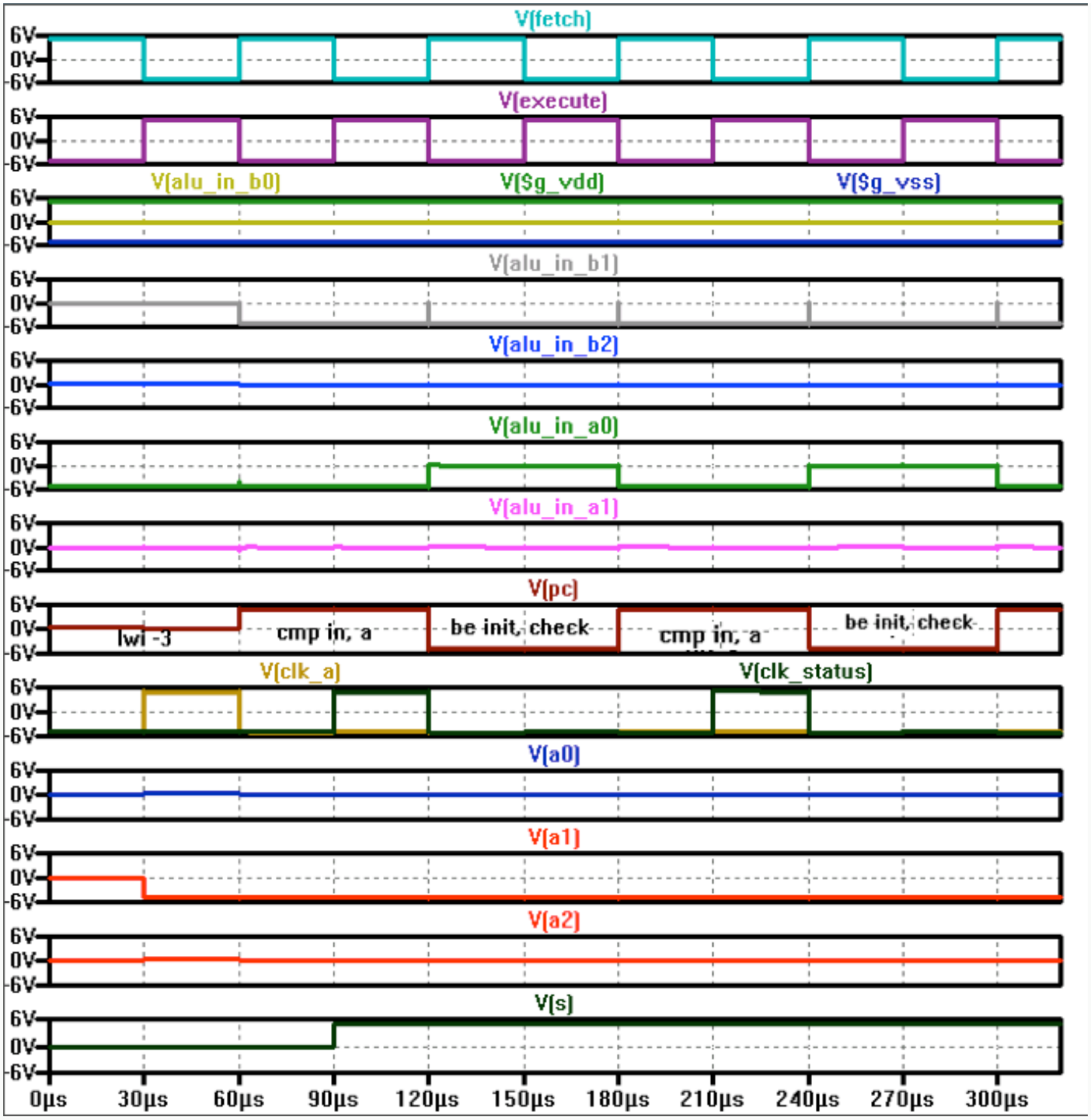


Figure 27. Guessing 8 when the correct number is -3.

This time IN is set to -10, $\overline{1}0\overline{1}$ when -3 is correct. -10 is less, so the status line goes low, and keeps looping:

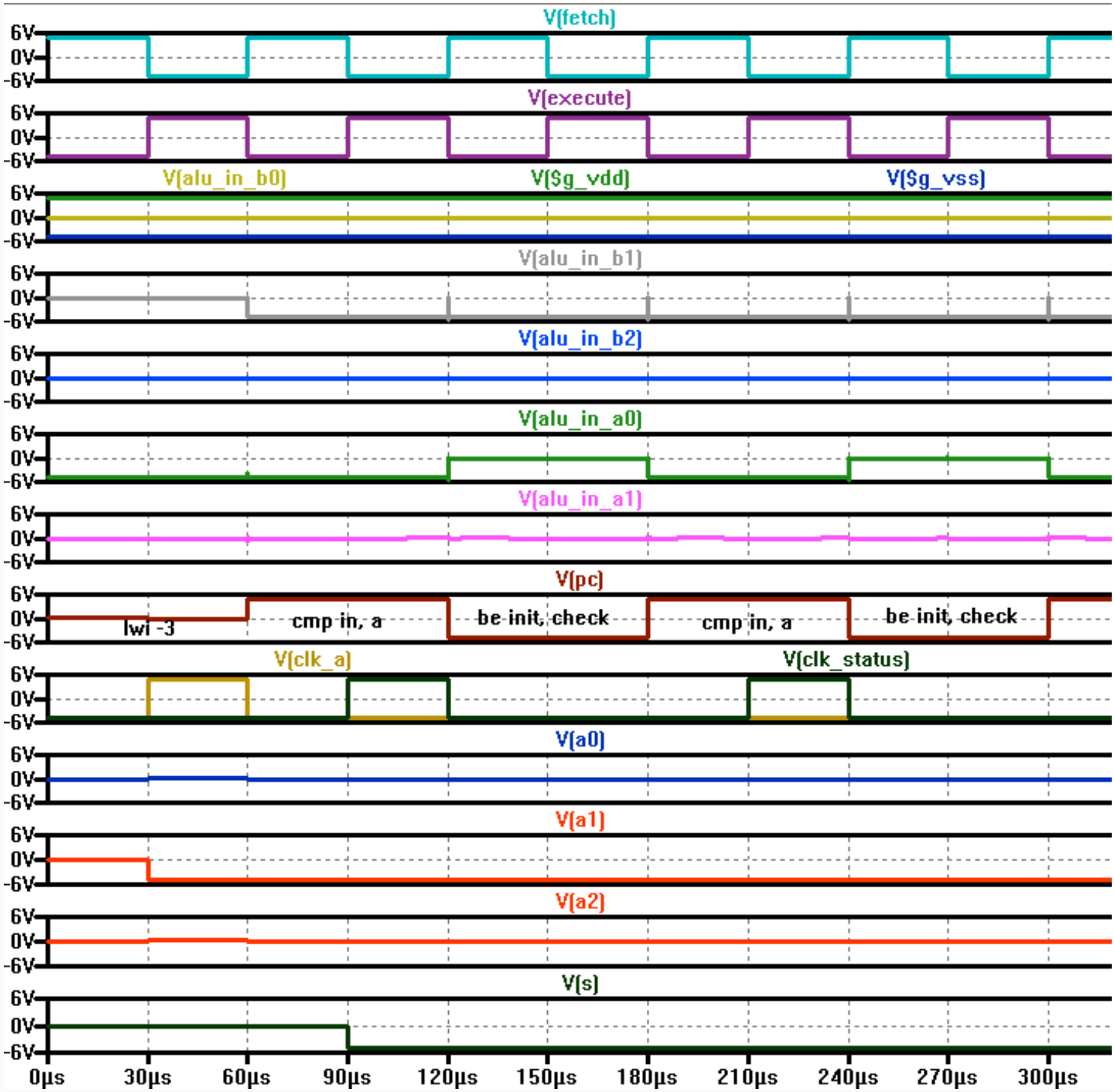


Figure 28. Guessing -10 when the correct number is -3.

Lastly, when IN is -3, the status trit is 0 since IN = A and the guess is correct:

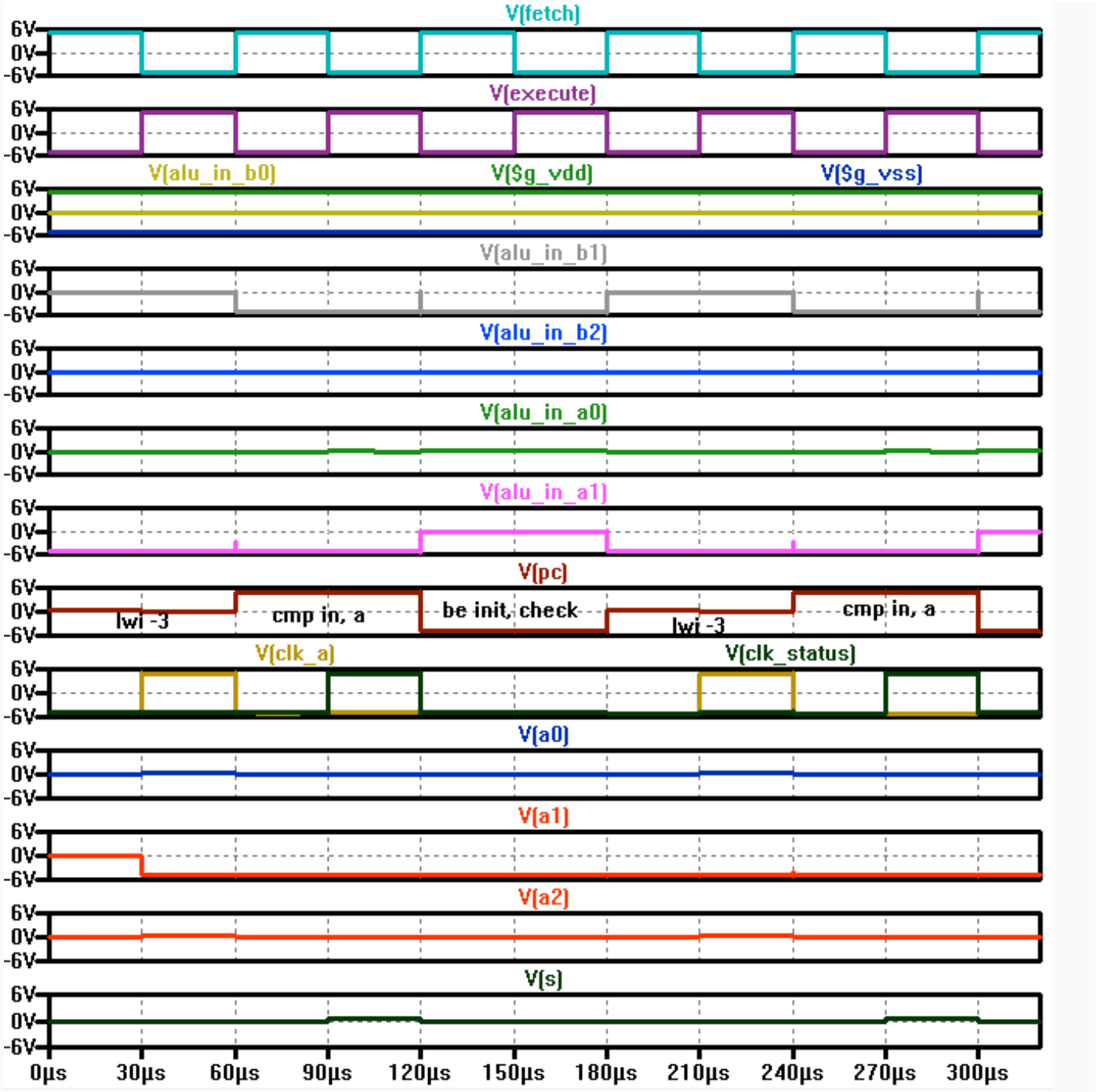


Figure 29. Guessing -3 which is correct.

The branch goes back to lwi and reloads A (which, in this example, has not changed). The guessing game is functional on the trinary computer architecture.

5. Evaluation

The primary advantage of trinary, as discussed in section **2.1**, is that fewer wires are necessary to represent the same range of numerical values. The number of wires per value is a quantifiable metric that can be used to compare the efficiency of binary and trinary systems.

The following table shows how the various quantities in the system are more compactly represented in trinary

than in binary:

Table 12. Evaluation of 3-trit Trinary Architecture vs. Binary

Quantity	States	Trinary Trits	Binary Bits	Count if Binary	States Wasted	% States Wasted	% Wires Increased
Machine Operation Codes	3 (lwi, be, cmp)	1	2	4	1	25%	50%
Register Address	3 (in, out, a)	1	2	4	1	25%	50%
Status code	3 (<, =, >)	1	2	4	1	25%	50%

The table above requires some explanation. As an example, the processor *status code* is required to represent three states: less than, equal to, or greater than. In trinary, only one wire is required to represent the three states. In binary, two bits are required, since one bit represents only two states. A fractional bit count can be calculated by log(3) / log(2) as 1.58 bits, but fractional bits cannot be realized as fractions of a wire, needless to say. Therefore, in binary the one extra state is wasted since it is not used. One state out of four possible states that can be represented with two bits is a 25% waste of representable states. Not only does a binary representation of the status code waste 25% of the available states, but it requires 50% more wires than in trinary (two wires versus one). For the given architecture, trinary is clearly superior for these reasons.

Trinary can be evaluated based on part count as well. The complete TCA2 architecture requires 161 chips: 42 x CD4016, 119 x CD4007, in addition to 340 resistors. For four transistors per CD4007 (two complementary MOSFET pairs are used), 6 per transmission gate (two per gate, but four for the two inverters) times 4 per chip, tis amounts to 1,484 transistors total, about 2/3 of the Intel 4004 processor[57] with 2,300 transistors. However, the TCA2 is significantly less complex than the 4004, so it cannot be directly compared. For a meaningful comparison, an analogous binary architecture would need to be designed and compared.

Furthermore, the trinary architecture could be compared on power usage. However, without an analogous binary architecture, this cannot be done.

The 3-trit architecture design itself was completed successfully. Not only was a transistor-level simulation completed, but a second simplified architecture was designed to run a second game. As intended, I designed an architecture to run a guessing game, but also a reduced architecture to run a Christmas lights game.

Recall that this project was part of the overall Ternary Computing Testbed effort, which also included tasks to design a compiler for an extended trinary architecture, as well as physical implementation of the computing architecture. A physical implementation of TCA0 was successfully built without the register and instruction decoder.

6. Conclusion and Future Directions

In conclusion, a functional transistor-level SPICE simulation of a simple 3-trit trinary architecture running a simple game was successfully simulated. The computer was built from the ground up using MOSFET transistor models to construct the logic gates, and the logic gates in turn were used to design higher-level architectural components. Lastly, a guessing game was written in assembly and loaded on the computer, where it was simulated and behaved as expected.

The deliverables of my project have been finished, but a future project could explore several additional aspects of trinary computing, including: comparing the power usage and cost of trinary and binary computers per bit, characterizing and optimizing the performance of trinary logic gates, and designing trinary VLSI integrated circuits.

7. Works Cited

1. D.E. Knuth, The Art of Computer Programming - Volume 2: Seminumerical Algorithms, pp. 207-208. Addison-Wesley, 3rd ed., 1998. ISBN 0-201-89684-2. Available: http://jeff.tk/wiki/Image:Knuth-TaoCPVol2-pg207%2C8.pdf
2. The Elements of Computing Systems: Building a Modern Computer from First Principles by Noam Nisan and Shimon Schocken (MIT Press, 2005).
3. Connelly, Jeff. Jeff.tk - Trinary/Meetings. Available: http://jeff.tk/wiki/Trinary/Meetings
4. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary/Status. Available: http://jeff.tk/wiki/Trinary/Status
5. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary. Available: http://jeff.tk/wiki/Trinary
6. Chavez, Antonio and Connelly, Jeff. Trinary/Tools - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Tools
7. Chavez, Antonio and Connelly, Jeff. Trinary/CPU Simulation - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/CPU_Simulation
8. Chavez, Antonio. Trinary/Compiler - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Compiler
9. Swanson, William. *Introduction to Binary Numbers*. 2002. Available: http://www.swansontec.com/sbinary.htm
10. davido, Perl Monks. *Ternary operator (there's no Trinary operator)*. Available: http://www.perlmonks.org/?node_id=562203
11. Wall, Larry. perl.com: Apocalypse 3. Available: http://www.perl.com/pub/a/2001/10/02/apocalypse3.html?page=6
12. Wall, Larry. et. al. *Programming Perl*, 3rd. Edition. ISBN: 978-0596000271 Section 3.16: Conditional Operator.
13. The Antikythera Mechanism Research Project. Available: http://www.antikythera-mechanism.gr/project/overview
14. Lexikon. *Analog Computers*. Available: http://www.computermuseum.li/Testpage/AnalogComputers.htm
15. National Semiconductor, Application Note 31. September 2002. *Op Amp Circuit Collection*. Available: http://www.national.com/an/AN/AN-31.pdf
16. Goldstrasz, Thomas et. al. *Computers During World War Two*. Available: http://waste.informatik.hu-berlin.de/Diplom/WW2/default_e.html
17. Bains, Sunny. *Analog computer trumps Turing model*. EE Times. 11/03/1998. Available: http://www.eetimes.com/story/OEG19981103S0017
18. Principia Cybernetica Web: *Digital Computer*. Available: http://pespmc1.vub.ac.be/ASC/DIGITA_COMPU.html
19. Maney, Kevin. USA Today, September 1997. *Debate Stirs Over Origins of Computers*. Available: http://www.scl.ameslab.gov/ABC/Articles/Debate9-97.html
20. Bebop's BYTES Back. *Claude Shannon's master's Thesis*. Available: http://www.maxmon.com/1938ad.htm
21. Hannah, Eric. *United States Patent 7309866: Cosmic ray detectors for integrated circuit chips*. Available: http://tinyurl.com/3ysdmk
22. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://dx.doi.org/10.1511/2001.40.3268 and mirrored at http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
23. A. Srivastava and K. Venkatapathy, “Design and Implementation of a Low Power Ternary Full Adder,”

VLSI Design, vol. 4, no. 1, pp. 75-81, 1996. doi:10.1155/1996/94696. Available: http://jeff.tk/wiki/Image:Design_and_Implementation_of_a_Low_Power_Ternary_Full_Adder.pdf
24. J.T. Butler, Multiple-Valued Logic in VLSI, IEEE Computer Society Press Technology Series, Los Alamitos, California, 1991.
25. A.K. Jain, M.H. Abd-E1-Barr and R.J. Bolton, "A new structure for CMOS realization of MVL functions," International Journal of Electronics, vol. 74, no. 2, pp. 251-263, 1993.
26. S.L. Hurst, "Two decades of multiple valued logic--an invited tutorial," in Proceedings of IEEE International Symposium on Multiple-Valued Logic, p. 164, May 1988.
27. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
28. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
29. A. P. Dhande and V. T. Ingole. Design And Implementation Of 2 Bit Ternary ALU Slice. SETIT 2005, 3rd International Conference: Science of Electronic, Technologies of Information and Telecommunications. March 17-21, 2005, Tunisia. Available: http://jeff.tk/wiki/Image:Dhande%2C_Ingole_-_Design_and_Implementation_of_a_2_Bit_Ternary_ALU_Slice.pdf
30. P.C.Balla & A.Antoniou "low power dissipation MOS ternary logic family" IEEE journal on solid state circuits Vol. Sc-19 no-5, P.739-749, October 1984.
31. D.I.porat "Three valued digital system" Proc.IEE Vol.116, No6, P.947-955, June 1969.
32. K.C.Smith "The prospects of multivalued logic technology & application view " IEEE transaction on computer, Vol.-C -30, P-619-627 September 1981.
33. Chung-Yu-Wu"Design & application of pipelined dynamic CMOS ternary logic & simple ternary differential logic" IEEE journal on solid state circuits Vol.28, No-8, August 1993.
34. CS150. Berkeley EECS. *Bits, Bytes, Nibbles, and Words: Some definitions*. Available: http://inst.eecs.berkeley.edu/~cs150/sp98/lectures/week6_2/tsld002.htm
35. Slashdot. *Ternary Computing Revisited*. Available: http://slashdot.org/comments.pl?sid=23934
36. Sloppy. Slashdot | Ternary Computing Revisited. Monday November 19 2001. *Trits?* Available: http://slashdot.org/comments.pl?sid=23934&cid=2585807
37. Merrill, Roy D. *Ternary Logic in Digital Computers*. January 1965. Avaialble: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
38. Bowles, Gary-alexander. US Patent #5498980 Ternary/binary converter circuit (Publication Date: 03/12/1996). Available: http://www.freepatentsonline.com/5498980.html
39. Setun' W. H. Ware, S. N. Alexander, N. M. Astrahan, H. H. Goode, M. Rubinoff, P. Armer, L. Bers, H.d. Huskey, "Soviet computer technology - 1959," Communications of the ACM, pp. 149-150, 1960.. Available: http://jeff.tk/wiki/Image:Communications_of_the_ACM_-_Soviet_Computer_Technology_-_1959.pdf
40. Faden, David. Reverse Fad Productions: Flip. Available: http://www.revfad.com/flip.html
41. Crispin, M. Panda Programing. 1 April 2005. Network Working Group, Request for Comments: 4042. *UTF-9 and UTF-18 Efficient Transformation Formats of Unicode*. Available: http://www.ietf.org/rfc/rfc4042.txt RFC 4042
42. Aspinwall, Jim. eCoustics. *Hacking CPU Voltage to Speed Up Your PC*. Available: http://forum.ecoustics.com/bbs/messages/34579/147079.html
43. Engelhardt, Mike. *LTspice/SwitcherCAD III User's Manual*. Available: http://ltspice.linear.com/software/scad3.pdf
44. Howell, Louis and Raymond, Eric S. Available: http://jeff.tk/wiki/Trinary/Logic#TriINTERCAL_Manual:_5.5.2.1_UNARY_LOGICAL_OPERATORS
45. Connelly, Jeff. Trinary/Parts - Jeff.tk - First Purchase. Available: http://jeff.tk/wiki/Trinary/Parts#Shopping_List:_First_Purchase
46. All About Circuits. Producing negative supply rails - Urgent - All About Circuits. Available: http://forum.allaboutcircuits.com/showthread.php?t=10415
47. All About Circuits. negative supply - All About Circuits Available: http://forum.allaboutcircuits.com/showthread.php?t=876

48. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
49. Connelly, Jeff. Trinary Computer Architecture - Older Diagrams. Available: Image:Proposed Architecture 2.png, Image:Proposed architecture 1.png
50. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
51. Lite-On Electronics Inc. Part No. LTL-30EHJ. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-30EHJ.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1057-ND
52. Lumex. T-5mm LED, 6 leaded, multi-colored, 636 nm AlInGoP Red/574 nm, AlInGoP Green BiColor, 470 nm Ultra Super Blue, Water Color Lens Datasheet. Part #SSL-LX5099SIUBSUGB. Available: http://rocky.digikey.com/weblib/Lumex/Web%20Data/SSL-LX5099SIUBSUGB1.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=67-1829-ND
53. Lite-On Electronics Inc. Part No. LTL-293SJW. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-293SJW.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1038-ND
54. Quicktar. Current limiting resistor calculator for LEDs. Available: http://www.quickar.com/noqbestledcalc.htm
55. Eigenratios of Self-Interpreters: The Mark II OISC Self-Interpreter. Available: http://eigenratios.blogspot.com/2006/09/mark-ii-oisc-self-interpreter.html
56. Connelly, Jeff. Public Git Hosting - trinary.git/tree - circuits/. Available: [http://repo.or.cz/w/trinary.git?a=tree;f=circuits
57. Carothers D. Christopher. Evolution of Intel Microprocessors: 1971 to 2007. Available: http://www.cs.rpi.edu/~chrisc/COURSES/CSCI-4250/SPRING-2004/slides/cpu.pdf

Appendix A: Logic

A.1. Unicode Symbols

Writing about trinary requires defining new symbols to represent the trinary operations. 7-bit ASCII exists and is widely supported, but only supports a very limited subset of all written characters. Switching to a base-3 number system offers the opportunity to also switch to Unicode as the standard text encoding, since there is plenty of space in a sufficient number of trits to store all Unicode code points, and more importantly, no number of trits will exactly encode the same number of 7-bit ASCII characters. Thus, a trinary computer can standardize on a larger character size than 8-bits, easily allowing for Unicode support.

Here is a list all Unicode characters we use here, or are used elsewhere, or could potentially be used:

Table A.1. Symbols for Trinary Logic

Code Point	Name	Character	ASCII Character	Usage
U+2229	INTERSECTION	∩	]	Trinary unary gate: rotate down
U+222a	UNION	∪	[	Trinary unary gate: rotate up
U+2197	NORTH EAST ARROW	↗	{	Trinary unary gate: shift up
U+2198	SOUTH EAST ARROW	↘	}	Trinary unary gate: shift down
U+0305	COMBINING OVERLINE	◌̅	~	Trinary unary gate: invert (example: x^-, note: in this wiki can also use <ob> tag)
U+00ac	NOT SIGN	¬	/	Trinary unary gate: forward diode (Option-L on Mac)
U+2310	REVERSED NOT SIGN	⌐	\	Trinary unary gate: reverse diode
U+2206	INCREMENT	∆	+	Trinary unary gate: positive trinary inverter
U+2207	NABLA	∇	-	Trinary unary gate: negative trinary inverter
U+2518	BOX DRAWINGS LIGHT UP AND LEFT	┘	+	Trinary unary gate: positive trinary inverter
U+2514	BOX DRAWINGS LIGHT UP AND RIGHT	└	-	Trinary unary gate: negative trinary inverter
U+2319	TURNED NOT SIGN	⌙	-	Trinary unary gate: negative trinary inverter (note: Unicode has no "reversed turned not sign" for a positive inverter)
U+2688	HOT SPRINGS	♨	*	Trinary unary gate: $x^0 = \Delta(x + \overline{x})$
U+2227	LOGICAL AND	∧	@	Trinary unary gate: TAND
U+2228	LOGICAL OR	∨	!	Trinary unary gate: TOR
U+22BC	LOGICAL NAND	⊼	#	Trinary unary gate: TNAND
U+22BD	LOGICAL NOR	⊽	$	Trinary unary gate: TNOR
U+2193	DOWNWARDS ARROW	↓	^	Trinary dyadic gate: minimum

U+2191	UPWARDS ARROW	↑	%	Trinary dyadic gate: maximum
U+21d1	UPWARDS DOUBLE ARROW	⇑	&	Trinary dyadic gate: exclusive maximum
U+2192	RIGHTWARDS ARROW	→		Trinary dyadic gate: mean
U+2261	IDENTICAL TO	≡	=	Trinary dyadic gate: magnitude
U+2460	CIRCLED DIGIT ONE	①		Base 9 digit: -1
U+2461	CIRCLED DIGIT TWO	②		Base 9 digit: -2
U+2462	CIRCLED DIGIT THREE	③		Base 9 digit: -3
U+2463	CIRCLED DIGIT FOUR	④		Base 9 digit: -4
U+0031	DIGIT 1		1	Trinary unary gate: constant 1 ("111")
U+0030	DIGIT 0		0	Trinary unary gate: constant 0 ("000")
U+0069	LOWERCASE LETTER I		i	Trinary unary gate: constant i ("iii")
U+0042	UPPERCASE LETTER B		B	Trinary unary gate: buffer
U+003C	OPENING ANGLE BRACKET		<	Trinary unary gate: swap i/0
U+003E	CLOSING ANGLE BRACKET		>	Trinary unary gate: swap 0/1

The method by which the above Unicode characters can be typed varies from platform to platform, so I compiled a quick reference of tips to type the characters on Mac OS X and Windows[58]. Now that the notation is out of the way, I'll discuss trinary logic.

A.2. Ternary Functions

Boolean (binary) Algebra was invented in 1854 by George Boole (1815-1864). It is well known, whole books have been written on it, and algorithms have been developed using it. This section attempts to help the world fully understand Trinary Algebra, so it can be used more extensively. Note that in general, there are $(3^3)^n$ possible *n*-input trinary functions. I will begin with discussing unary (1-input) functions, followed by dyadic (2-input) functions, and lastly I'll cover troolean algebra and alternative logic systems.

A.3. Unary Functions

Unary functions are those which take one input, and output one output. The quintessential binary unary function is NOT, also known as *invert*:

Table A.2.
Binary NOT
Truth Table

Input	Output
0	1
1	0

Often, the binary NOT function is written with an overbar; for example, "not A" is $\overline{A}$. The NOT function is identified by the function number 10—that is, the series of bits which forms if the right-hand side of the truth table is written down, from top to bottom. In all, binary has $2^2 = 4$ possible unary functions:

- Function number 00 - always output 0 (constant 0)
- Function number 01 - output the same value as the input (*buffer*)
- Function number 10 - invert the input (NOT, invert)
- Function number 11 - always output 1 (constant 0)

Similarly, trinary has $3^3 = 27$ unary functions, more than six times that of binary. To make sense of this large number of functions, they can be divided into *constant*, *one-to-one*, and *many-to-one* functions.

A.3.1. Constant Functions

Constant functions always output the same value as the input. Trinary has three constant unary functions:

Table A.3.
Constant Functions

F#	Description
$\overline{111}$	always output $\overline{1}$
000	always output 0
111	always output 1

A.3.2. One-to-one Functions

One-to-one functions are those where input maps to exactly one output. They all have $\overline{1}$, 0, and 1 in their output, meaning that they are *reversible*: the original input can be derived from the output, by taking the inverse of the output.

First, a note about the symbols used in this document. Grubb[59] and Mouftah[60] define an incompatible set of symbols and terminology for trinary logic.

The one-to-one unary trinary functions, in both Grubb and Mouftah notation, are as follows:

Table A.4. One-to-one Functions

F#	Grubb's Name	Diff: $\bar{1}01$	Inverse	Grubb-based Expression	Mouftah-based Expression/Name	What we call it
$\bar{1}01$	buffer		$\bar{1}01$	A	A	Buffer
$\bar{1}10$	swap 0/1	'\	$\bar{1}10$	$\cup\bar{A}$		Swap 0/1
$0\bar{1}1$	swap $\bar{1}$/0	/'	$0\bar{1}1$	$\cap\bar{A}$		Swap $\bar{1}$/0
$01\bar{1}$	rotate up	///	$1\bar{1}0$	$\cap A$	$(\llcorner\bar{A} \barwedge 0)\barwedge(\lrcorner\bar{A})$ - inverse cycling gate	Cycle up
$1\bar{1}0$	rotate down	\\\	$01\bar{1}$	$\cup A$	$(\lrcorner\bar{A} \barvee 0)\barvee(\llcorner\bar{A})$ - cycling gate	Cycle down
$10\bar{1}$	swap $\bar{1}$/1, invert	\/	$10\bar{1}$	$\bar{A}$	$\bar{A}$	Simple Ternary Inverter

The "Diff" column points to the trit's original position. ' means the trit stayed in the same place, / points left, \ right. The inverses of these functions are themselves except for the rotates.

Grubb uses the term "rotate up/down", while Mouftah uses "cycle" and "inverse cycle". We use a mix of both terminologies for maximum clarity: "cycle up" and "cycle down". The numerical value of the trits is increased as they are cycled up, or decreased as they cycle down, as the name suggests. Furthermore, the term "rotate" (or roll) is also commonly used in binary computers to mean shifting of bits within a register left or right, wiring the bit shifted out into the bit shifted in. For example, the x86 ISA has ROR and ROL instructions. These are not in any way related to these logic functions, so the terms "rotate up" and "rotate down" are confusing/misleading. For this reason, in this document I exclusively use the terms "cycle up" and "cycle down" from this point on.

As for Grubb's symbols, $\cap$ is the well-established symbol for intersection, and $\cup$ is the well-established symbol for union. The established meanings severely conflict with Grubb's intended usage as "cycle up" and "cycle down". For this reason, Mouftah's notation will be used whenever possible, although it uses the symbols $\lrcorner$, $\llcorner$, $\barwedge$, and $\barvee$ which have not yet been explaned . Mouftah[60] only defines a symbol for one one-to-one unary function: the simple ternary inverter, as shown above. The other one-to-one unary functions can be derived from functions which will be introduced later in this document.

A.3.3. Many-to-One Functions

The most plentiful trinary unary functions have only two values of trits in their function table; one is repeated. These have the effect of replacing the trit with a specified value if it is another specified value, otherwise defaulting to the third specified value. The shift functions are useful for coercing trits into bits, used with Trinary Coded Binary, but with the help of the swap, rotate, and invert operators can create any of the *many-to-one* functions.

Table A.5. Many-to-One Functions

F#	ITE	Grubb-based Expression	Mouftah-based Expression	Mouftah/Yoeli-based Name
$\overline{1}10$	$10\overline{1}$	$\searrow A$ = Shift Down	$\neg\lrcorner\overline{A}$	
$\overline{1}11$	$11\overline{1}$	$\cap\nearrow\overline{A}$	$\sqcup A = \overline{\lrcorner\overline{A}} = \lrcorner\overline{A} = \lrcorner\lrcorner A$	x^1 (Yoeli), decode-1
$\overline{1}0\overline{1}$	$0\overline{1}\overline{1}$	$\searrow\cap A$		
$\overline{1}00$	$\overline{1}10$	$\searrow\nearrow A$	$\lceil A$	Reverse Diode
$\overline{1}1\overline{1}$	$01\overline{1}$	$\cap\nearrow\cup\overline{A}$	$\lrcorner(A + \overline{A})$	x^0 (Yoeli), decode-0
$\overline{1}11$	$\overline{1}11$	$\cup\searrow\overline{A}$	$\lrcorner\llcorner A = \lrcorner\overline{A} = \llcorner\llcorner A$	
$0\overline{1}\overline{1}$	$\overline{1}0\overline{1}$	$\searrow\overline{A}$	$\lceil\llcorner A$	Earthed Negative Ternary Inverter
$0\overline{1}0$	$0\overline{1}0$	$\cup\nearrow\cup\overline{A}$		
$00\overline{1}$	$10\overline{1}$	$\cup\nearrow\overline{A}$	$\lceil\llcorner A = \lceil\overline{A}$	
001	110	$\nearrow\searrow A$	$\neg A$	Forward Diode
010	010	$\cap\searrow\cap A$		
011	$\overline{1}01$	$\nearrow A$ = Shift Up	$\overline{\lceil\llcorner A}$	
$1\overline{1}\overline{1}$	$\overline{1}1\overline{1}$	$\cap\nearrow A$	$\llcorner A$	Negative Ternary Inverter (Mouftah), $x^{\overline{1}}$ (Yoeli), decode-$\overline{1}$
$1\overline{1}1$	$0\overline{1}1$	$\cup\searrow\cap A$		
100	$\overline{1}10$	$\cap\searrow\overline{A}$	$\neg\overline{A}$	
101	001	$\nearrow\cup\overline{A}$		
$11\overline{1}$	$1\overline{1}1$	$\cup\searrow A$	$\lrcorner A$	Positive Ternary Inverter
110	101	$\nearrow\overline{A}$	$\neg\lrcorner A$	Earthed Positive Ternary Inverter

The first trit of the if-then-else field is the value to compare the input with; if the input is this value the second value of the ITE field is output, else the third value is. For example, function 011 has an ITE of $\overline{1}01$, so the process goes like this: a) if input is i, output is 0 b) else, output is 1.

Another way to categorize many-to-one unary gates is by what two input values are logically considered equivalent, as far as the output value is concerned. If inputs 0 and 1 are treated equivalently, then the function number has the form *xyy*, where *x* is a trit value different than *y*. All types:

Table A.6. Many-to-one unary trinary gates: input equivalences

Input Values Considered Equivalent	Form of function number	Notes and examples
0 and 1	xyy	Reverse diode
$\overline{1}$ and 0	xxy	Forward diode, most binary gates act this way (treat negative input as zero).
$\overline{1}$ and 1	xyx	The elusive "symmetric gate", cannot be produced exclusively from Mouftah's *unary* gates (requires a dyadic gate)

These are from Trinary.cc. Mouftah[61], on page 125, defines these fundamental unary operators instead:

Table A.7. Mouftah's Many-to-One Unary Ternary Operators

Mouftah's Name	Expression	F#	Balanced F#	Trinary.cc Equivalent Expression	Suggested Expression for Balanced Trinary
Negative Ternary Inverter (NTI)	$\overline{x^0}$	200	$1\overline{1}\overline{1}$	∩↗x	∇x or ∟x or ⌴x
Positive Ternary Inverter (PTI)	$\overline{x^2}$	220	$11\overline{1}$	∪↘x	Δx or ⌟x
Forward Diode (FD)	$x^{\neg}$	112	001	↗↘x	¬x
Reverse Diode (RD)	$x^{\lceil}$	011	$\overline{1}00$	↘↗x	⌐x

Interestingly, the dyadic preference functions (see below) consist of these functions (especially FD and RD) more often than the Trinary.cc functions. They are also easier to construct, as described in the implementation section. It is not known whether they are more fundamental.

Research in trinary minimization [62] [63] [64] [65] often uses these functions, described by Mouftah[60], page 124, adjusted for balanced trinary:

- $x^{\underline{n}}$ = {if x is n, then 1, else $\overline{1}$}, hence, all these functions have an if-then-else number of $n1\overline{1}$.
- $x^{\overline{1}}$ = ∟x
- x^0 = ⌟(x + $\overline{x}$)
- x^1 = ⌟⌟x

Table A.8. Unary Functions Commonly Used in Ternary Logic Minimization

Notation	If-then-else	F#	Mouftah-based Equivalent
$x^{\overline{1}}$	$\overline{1}1\overline{1}$	$1\overline{1}\overline{1}$	∟x
x^0	$01\overline{1}$	$\overline{1}1\overline{1}$	⌟(x + $\overline{x}$)
x^1	$11\overline{1}$	$\overline{1}\overline{1}1$	⌟⌟x

Note that x^0 requires the dyadic function, MAX, covered in later sections. This means that the four many-to-one unary functions ⌐, ¬, ⌟ , ∟, plus the one-to-one function $\overline{x}$, are not alone sufficient to construct all trinary unary functions. Along with MIN and MAX, they are, but not all unary functions can be constructed by those five unary functions alone.

Mouftah also includes schematics for earthed positive/negative ternary inverters (EPTI and ENTI):

Table A.9. Earthed Positive/Negative Ternary Inverters, Balanced (based on Mouftah)

Name	Notation	F#	Equivalent
EPTI	E(⌟ x)	110	¬⌟A
ENTI	E(∟x)	$0\overline{1}\overline{1}$	⌐∟A

These are equivalent to positive/negative inverters followed by forward/reverse diodes, respectively, as Mouftah explains: "It is obvious that using the ternary earthed circuits one may save the use of FD and RD

circuits. "

Comparisons:

Trinary.cc's many-to-one unary functions, shift up and shift down, are named such that confusion between bitwise shifting left and right may result. For example, x86 has SAL, SAR, SHL, SHR instructions for arithmetic and logical shifts left and right (multiplication and division by powers of two). Trinary.cc's "shift up" and "shift down" are in no way related.

The $\overline{x^0}$, $\overline{x^1}$, $\overline{x^2}$ expressions have the positive property that they all look like inverters, since they are all overbars. However, {0,1,2} are not used in balanced trinary, so this must change for our purposes. The most straight-forward choice, $\overline{x^1}$, $\overline{x^0}$, $\overline{x^1}$, is a possibility but the problem is that this kind of formatting is difficult to represent in pure text files using Unicode characters. Unicode does have plenty of combining characters that may be usable for this purpose. For example, x^{-i} is purely text. Luckily Unicode has a superscript i, but no superscript 1. Unicode does have superscript + and - signs, which look like this: x^{-+} and x^{--}, respectively, for positive and negative ternary inverters. However, combining characters may be difficult to process, and Unicode superscript are quite small and difficult to read. A third alternative is using overline, strike-through, and underline for each inverter. Logically, these should represent positive, simple, and negative inverters, but the simple inverter is conventionally an overline, and strike-through could make expressions unreadable. For these reasons I propose ⌞ and ⌟ for negative and positive inverters, and although they do not extend over the characters, they suffer from none of the other problems presented here.

Table A.10. Rejected Symbols

Code Point	Name	Character	Usage
U+0336	COMBINING LONG STROKE OVERLAY	x̶	Trinary unary gate: invert (not used), example: x̶
U+207A	SUPERSCRIPT PLUS SIGN	$^+$	Trinary unary gate: positive ternary inverter, example: x^{-+}
U+207B	SUPERSCRIPT MINUS	$^-$	Trinary unary gate: negative ternary inverter, example: x^{--}
U+2071	SUPERSCRIPT LATIN SMALL LETTER I	^{i}	Trinary unary gate: negative ternary inverter, example: x^{-i}
U+0332	COMBINING LOW LINE	x̲	Trinary unary gate: negative ternary inverter, example: x_

Alternatively, instead of ⌟ for the positive ternary inverter, one could use a line with an upward curve on the end, and instead of ⌞, a line with an upward curve on the beginning, similar to the FD and RD symbols ¬ and ⌐ but pointing upwards. The advantage of these symbols is that they could be written over the text in handwritten trinary logic functions, but written in prefix form when typed up. Although Unicode has a turned not sign ⌙, but unfortunately, I haven't been able to find any inverted turned not sign. However, Unicode does have the box-drawing characters ⌞ and ⌟ which could work.

A.3.4. Symmetric Unary Functions

What I call *symmetric* unary functions are those with a function number of the form *xyx*. That is, the output is

the same for $\overline{1}$ and 1 input, but different for 0 input. Additional unary gates can be applied to a symmetric unary gate to produce additional symmetric unary gates.

Mouftah expresses the symmetric function $\overline{1}1\overline{1}$ as $\lrcorner (x + \overline{x})$, where + is the MAX gate, a dyadic function. This might not be optimal, but no simple symmetric unary gate has been found.

Symmetric functions are important in multiplexers.

A.3.5. Basic Unary Functions

Mouftah implemented a separate set of unary gates, which are easier to implement than Grubb's:

Table A.11. Mouftah's Unary Ternary Operators

Mouftah's Name	Expression	F#	Balanced F#	Trinary.cc Equivalent Expression	Suggested Expression for Balanced Trinary
Negative Ternary Inverter (NTI)	$\overline{x^0}$	200	$1\overline{11}$	∩↗x	↳x or ↳x or ⌴x
Simple Ternary Inverter (STI) Note: this is one-to-one	$\overline{x^1}$	210	$10\overline{1}$	$\overline{x}$	$\overline{x}$
Positive Ternary Inverter (PTI)	$\overline{x^2}$	220	$11\overline{1}$	∪↘x	⌟x or ⌟x
Forward Diode (FD)	$x^{\neg}$	112	001	↗↘x	¬x
Reverse Diode (RD)	$x^{\lceil}$	011	$\overline{1}00$	↘↗x	⌐x

But the MAX function needs to be implemented in conjunction with these unary functions in order to make all possible unary gates.

Grubb's unary gates are complete in themselves, in that any of the 27 possible unary gates can be created using any of these:

Table A.12. Basic Unary Functions

F#	Expression	Name	Comments
$\overline{11}0$	↘A	Shift Down	(ITE=$10\overline{1}$)
011	↗A	Shift Up	(ITE=$\overline{1}01$)
$01\overline{1}$	∩A	Rotate Up	///
$1\overline{1}0$	∪A	Rotate Down	\\\
$10\overline{1}$	$\overline{A}$	Invert (swap $\overline{1}$/1)	\'/

Years ago, I developed a Unary Quick Reference[66] sheet with these functions. However, Grubb's circuits are much more complicated than Mouftah's, so we used Mouftah's instead.

A.3.6. Unary Overbar Notation Explained

In binary, the only logically-useful unary gate is NOT, written with an overbar: $\overline{A}$.

In trinary, there are many unary gates, but I find it useful to preserve the overbar notation, to indicate that the operator is unary, and to easily allow nesting. The same $\overline{A}$ notation refers to a Simple Trinary Inverter, or simply an inverter. But additional notation is needed for the additional unary gates.

Mouftah introduced ¬ and ⌐ for forward and reverse diode, respectively. Unary Overbar Notation takes this further. Rules:

- An overbar with a tip angled *downward* always indicates a *diode* operation: ¬, ⌐
- An overbar with a tip angled *upward* indicates inversion operations: ⌟ , ⌞
- An angle at the *end* of the overbar indicates a *forward/positive* operation: ¬, ⌟
- An angle at the *beginning* of the overbar indicates a *negative/reverse* operation: ⌐, ⌞

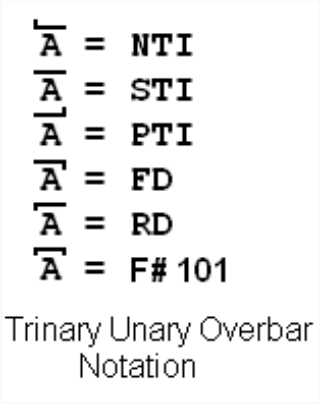


Figure A.1. Overbar unary trinary logic gate convention. Downward=diode, upward=inverter, end=positive/forward, beginning=negative/reverse.

This notation is primarily for handwritten logic equations, or if you want to take the time to typeset it properly on a computer. All of the operators can also be instead written in front of the operator, in prefix form (this differs from Mouftah, where the FD and RD gates are written in suffix form). The bare overbar is written as a slash: /A.

A.4. Dyadic Functions

To avoid confusion with the binary number system, I've chosen to use the term *dyadic* to refer to two-input functions, as opposed to binary. This practice was borrowed from Randall Hyde's Art of Assembly Language[67].

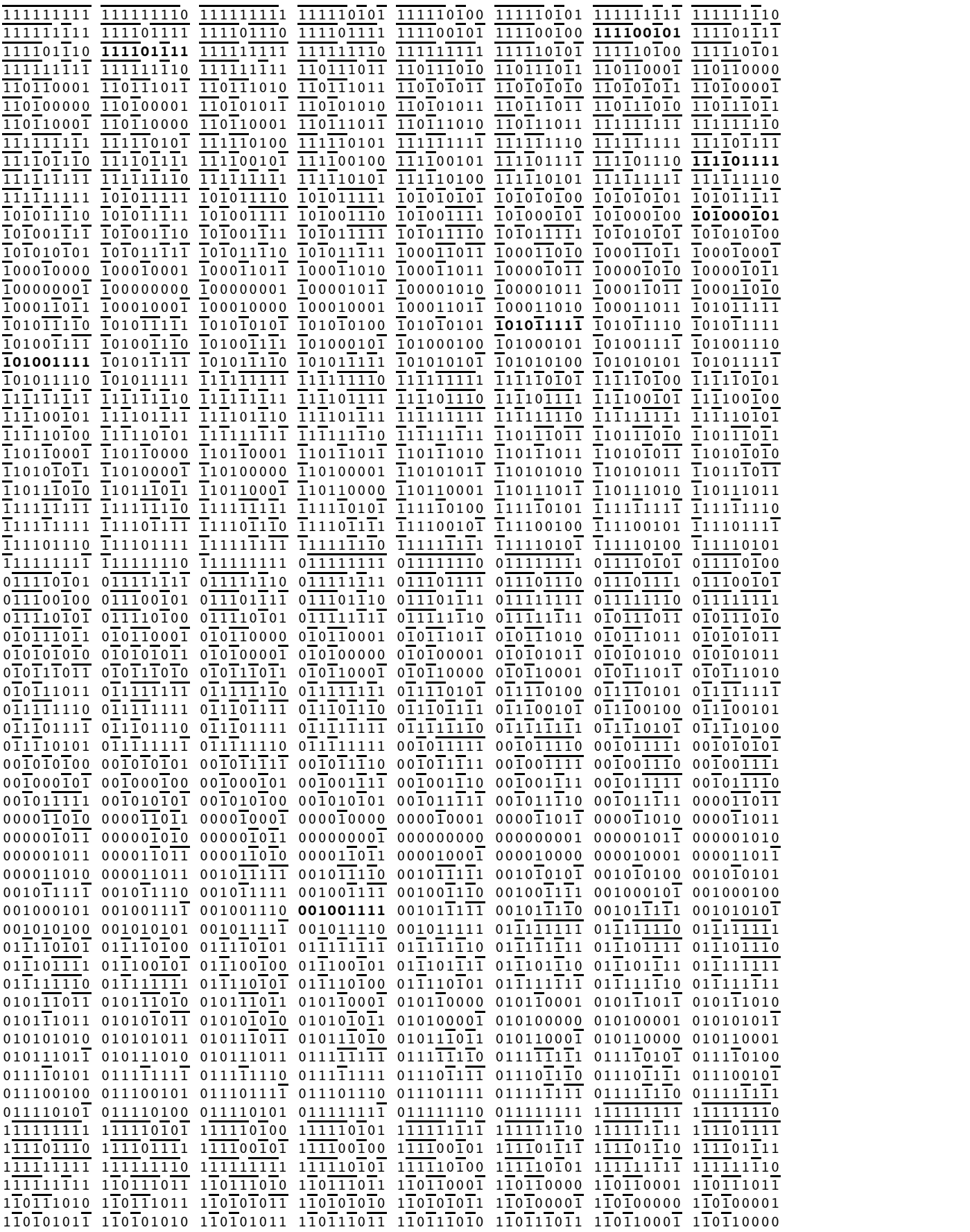
It is no problem to enumerate all the unary functions, because there are only 3^3 of them. Not so with trinary dyadic functions. There are $(3^3)^3$ = 19,683 possible trinary dyadic functions. I've chosen to not list them all in this document; most of wouldn't even be useful or could be derived from more basic building-block functions.

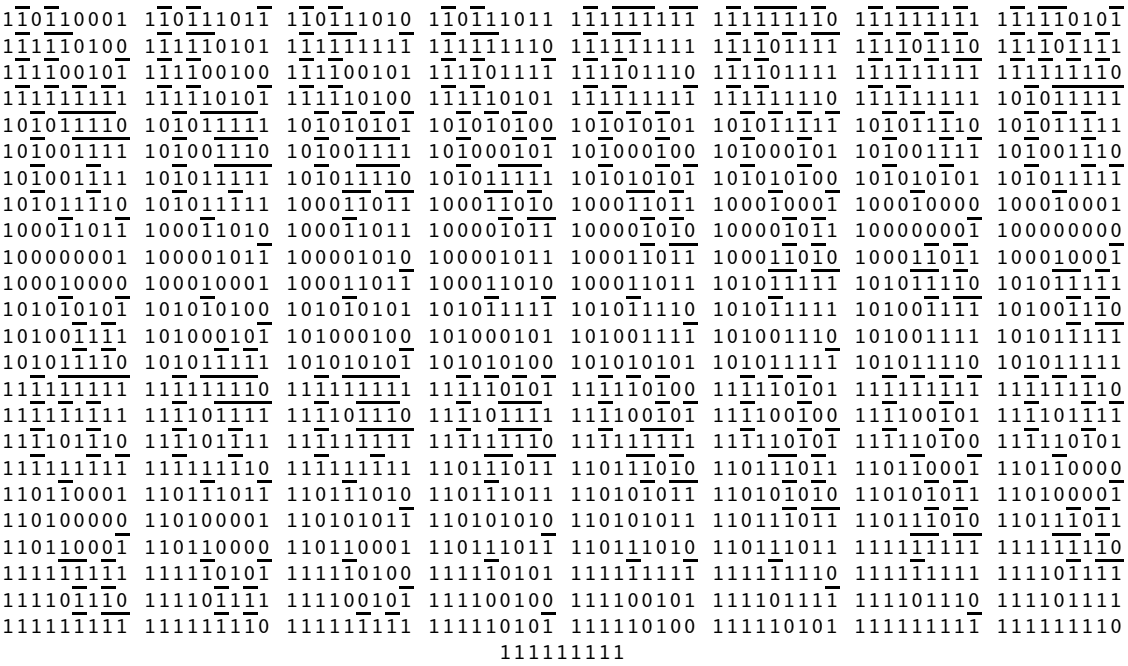
A.4.1. Commutativity

Functions are *commutative* if the order of their inputs do not matter: for an operator ☮, A ☮ $B = B$ ☮ A. If a function isn't commutative, it is probably not worth our time in the search for basic dyadic trinary functions. So we can downsize all the possible functions to a mere 729 commutative functions, less than 4% of the possible functions.

All commutative binary trinary function numbers:

Total bi-tri: 19683
Commutative: 729 (3.7037037037037%)





Bold functions have a name.

A.4.2. Preference Functions

Preference/choice functions where brought to my attention by the TriINTERCAL manual[68]—Trinary dialect of the INTERCAL programming language. Although TriINTERCAL itself is obviously a joke, the logic behind it is not. Or is it?) , specifically section 5.5.2.1:

> Let's start with AND and OR. To begin with, these can be considered "choice" or "preference" operators, as they always return one of their operands. AND can be described as wanting to return 0, but returning 1 if it is given no other choice, i.e., if both operands are 1. Similarly, OR wants to return 1 but returns 0 if that is its only choice. From this it is immediately apparent that each operator has an identity element that "always loses", and a dominator element that "always wins". AND and OR are commutative and associative, and each distributes over the other. They are also symmetric with each other, in the sense that AND looks like OR and OR looks like AND when the roles of 0 and 1 are interchanged (De Morgan's Laws). This symmetry property seems to be a key element to the idea that these are logical, rather than arithmetic, operators. In a three-valued logic we would similarly expect a three- way symmetry among the three values 0, 1 and 2 and the three operators AND, OR and (of course) BUT. The following tritwise operations have all the desired properties: OR returns the greater of its two operands. That is, it returns 2 if it can get it, else it tries to return 1, and it returns 0 only if both operands are 0. AND wants to return 0, will return 2 if it can't get 0, and returns 1 only if forced. BUT wants 1, will take 0, and tries to avoid 2. The equivalents to De Morgan's Laws apply to rotations of the three elements, e.g., 0 -> 1, 1 -> 2, 2 -> 0. Each operator distributes over exactly one other operator, so the property "X distributes over Y" is not transitive. The question of which way this distributivity ring goes around is left as an exercise for the student. In TriINTERCAL programs the '@' (whirlpool) symbol denotes the unary tritwise BUT operation. You can think of the whirlpool as drawing values preferentially towards the central value 1. Alternatively, you can think of it as drawing your soul and your sanity inexorably down... On the other hand, maybe it's best you NOT think of it that way. A few comments about how these operators can be used. OR acts like a tritwise maximum operation. AND can be used with tritmasks. 0's in a mask wipe out the corresponding elements in the other operand, while 1's let the corresponding elements pass through unchanged. 2's in a mask

consolidate the values of nonzero elements, as both 1's and 2's in the other operand yield 2's in the output. BUT can be used to create "partial tritmasks". 0's in a mask let BUT eliminate 2's from the other operand while leaving other values unchanged. Of course, the symmetry property guarantees that the operators don't really behave differently from each other in any fundamental way; the apparent differences come from the intuitive view that a 0 trit is "not set" while a 1 or 2 trit is "set".

To summarize:

- OR prefers $1, 0, \overline{1}$ - this is the max operator
- AND prefers $\overline{1}, 1, 0$
- BUT prefers $0, \overline{1}, 1$

An operator's truth table can easily be derived from its preferences:

Table A.13. Preference Truth Tables

Input	Preferences (pref-nnn)					
AB	**$\overline{1}01$**	**$\overline{1}10$**	**$0\overline{1}1$**	**$1\overline{1}0$**	**$01\overline{1}$**	**$10\overline{1}$**
$\overline{1}\overline{1}$	$\overline{1}$	$\overline{1}$	$\overline{1}$	$\overline{1}$	$\overline{1}$	$\overline{1}$
$\overline{1}0$	$\overline{1}$	$\overline{1}$	0	$\overline{1}$	0	0
$\overline{1}1$	$\overline{1}$	$\overline{1}$	$\overline{1}$	1	1	1
$0\overline{1}$	$\overline{1}$	$\overline{1}$	0	$\overline{1}$	0	0
00	0	0	0	0	0	0
01	0	1	0	1	0	1
$1\overline{1}$	$\overline{1}$	$\overline{1}$	$\overline{1}$	1	1	1
10	0	1	0	1	0	1
11	1	1	1	1	1	1

Notice how A ☮ A = A, where ☮ is a preference/choice function (preference functions satisfy the *idempotent* property). The operator has no other choice but to return A.

pref-$\overline{1}01$ is the minimum function while pref-$10\overline{1}$ is the maximum; this is quite obvious as they prefer the highest or lowest trit. By extrapolating, all of the preference functions can be found:

Table A.14. Preference Functions

Pref	Function #	Name(s)	Comments
$\overline{1}01$	$\overline{1}\overline{1}\overline{1}$,$\overline{1}00$,$\overline{1}01$	pref-$\overline{1}01$	TAND, MIN
$\overline{1}10$	$\overline{1}\overline{1}\overline{1}$,$\overline{1}01$,$\overline{1}11$	pref-$\overline{1}10$	TriINTERCAL-AND (according to TriINTERCAL, but pref-$\overline{1}01$ makes more sense as the AND analog)
$0\overline{1}1$	$\overline{1}01$,000,$\overline{1}01$	pref-$0\overline{1}1$	TriINTERCAL-BUT. The "but" gate is similar to, but not quite "and".
$01\overline{1}$	$\overline{1}01$,000,111	pref-$01\overline{1}$	
$1\overline{1}0$	$\overline{1}\overline{1}\overline{1}$,$\overline{1}01$,111	pref-$1\overline{1}0$	
$10\overline{1}$	$\overline{1}01$,001,111	pref-$10\overline{1}$	TOR, MAX, TriINTERCAL-OR

Now to assign them symbols...

Note that pref-$\overline{1}01$ makes more sense for trinary AND, since it is the MIN function, analogous to in binary. This was discussed in Trinary/Meeting_Notes_20080108.

Also note, TriINTERCAL suggests that one can convert from balanced to unbalanced by replacing $\overline{1}$ with 2. This has the desirable property of preserving the value of 0 and 1 between both balanced and unbalanced trinary, but we decided to not use this scheme because the conversion is non-trivial in hardware. If, instead, balanced trits are converted to unbalanced trits by adding 1, then an unbalanced logic level can be obtained by measuring relative to the negative rail, V-, rather than ground (which would give you balanced trinary), as justified in Trinary/Architecture#Power Supply.

A.4.3. Tritmasks

Unary gates exist within binary gates, no matter the base. Binary trinary gates have unary trinary gates within them, just as binary boolean gates[69] are composed of unary boolean gates.

If a binary trinary function's truth table is written in three groups of three, each group will be a unary function. Group i is the unary function which operates on A if B is $\overline{1}$, group 0 is the unary function which operates on A if B is 0, etc.

```
 1̄   0   1
abc,def,ghi = unary operations on A if B=1̄,0,1
```

Knowing this, unary functions can be decomposed to see how they operate when used as tritmasks:

Table A.15. Decomposing Trinary Dyadic Functions into Unary Functions

Function#	$\overline{1}$	0	1	name(s) and symbol(s)
		Mouftah's		
$\overline{111}$,$\overline{1}00$,$\overline{1}01$	$\overline{1}$	⌐A	A	∧ TAND (minimum, prefer $\overline{1}01$)
111,100,10$\overline{1}$	1	¬$\overline{A}$	$\overline{A}$	⊼ TNAND
$\overline{1}01$,001,111	A	¬A	1	∨ TOR (maximum, prefer 10$\overline{1}$)
10$\overline{1}$,00$\overline{1}$,$\overline{111}$	$\overline{A}$	⌐$\overline{A}$	$\overline{1}$	⊽ TNOR
		Grubb's and Other Preference Functions		
$\overline{111}$,$\overline{1}00$,$\overline{1}01$	$\overline{1}$	↘↗A	A	↓ Minimum - prefer $\overline{1}01$
$\overline{111}$,$\overline{1}01$,111	∩↗$\overline{A}$	A	1	prefer 1$\overline{1}$0
$\overline{1}01$,000,111	A	0	1	prefer 01$\overline{1}$
$\overline{1}01$,001,111	A	↗↘A	1	↑ Maximum - prefer 10$\overline{1}$ ("trinary or")
$\overline{1}01$,0$\overline{1}$1,11$\overline{1}$	A	∩$\overline{A}$	∪↘A	⇑ Exclusive Max
$\overline{111}$,$\overline{1}01$,$\overline{1}11$	$\overline{1}$	A	∩↘$\overline{A}$	prefer $\overline{1}$10 ("trinary and")
$\overline{1}0\overline{1}$,000,$\overline{1}01$	↘∪A	0	A	prefer 0$\overline{1}$1 ("trinary but")

Minimum sets output to $\overline{1}$ if mask is $\overline{1}$, lets it pass through if mask is 1, and if the mask is 0 then $\overline{1}$ and 0 pass through, but 1 is changed to a 0. Similarly, maximum lets the input pass through if the mask is $\overline{1}$, sets the

output to 1 if the mask is 1, and lets 0,1 pass through but sets $\overline{1}$ to 0 if the mask is 0.

A trit exclusive max'd with 0, and then exclusive max'd with 0 again gives the original value, just as (XOR A,0) XOR 0 = A. This works because the unary function $\cap\overline{B}$ is called when one of the inputs is 0. Since it is being called on its own output, (XMAX A,0) XMAX 0 = $\cap\cap\overline{\overline{B}}$ = $\cap\cup$B = B.

BUT, as explained in the TriINTERCAL manual, eliminates 1's while leaving other values unchanged, if the mask contains a $\overline{1}$. $\overline{1}0\overline{1}=\overline{\cup B}$ causes 1 to be mapped to a $\overline{1}$. 0's in the input always output 0's, while 1's output the other operand.

A.4.4. Named Functions

Most functions here where taken from Trinary.cc - Binary Operations[70].

The $\overline{1}$, 0, and 1 columns give the unary function that the dyadic function as like when the respective value is the other input. For example, the unary function for MIN when the other input is 0 is "↘↗A". That means, you can get that unary function by entering a 0 into one of the inputs of the MIN function, and tying the other input to the input (in this case, B).

Table A.16. Named Dyadic Functions

Function#	i	0	1	name(s)
$\overline{111}00\overline{1}01$	i	↘↗A	B	↓ Minimum - prefer $\overline{1}01$
$\overline{1}01001111$	B	↗↘A	1	↑ Maximum - prefer $10\overline{1}$ ("trinary or")
$\overline{1}010\overline{1}111\overline{1}$	B	$\cap\overline{B}$	$\cup$↘B	⇑ Exclusive Max
$\overline{1}0\overline{1}010\overline{1}0\overline{1}$	↘∩B	∩↘∩B	↘∩B	→ Mean
$011\overline{1}01\overline{11}0$	↗B	B	↘B	≡ Magnitude
$\overline{111}01\overline{1}11$				prefer $\overline{1}10$ ("trinary and")
$\overline{1}0\overline{1}000\overline{1}01$				prefer $0\overline{1}1$ ("trinary but")
$\overline{1}01000111$				prefer $01\overline{1}$
$\overline{11}\overline{1}01111$				prefer $1\overline{1}0$

A.4.5. Completeness

The most logically important dyadic trinary functions are MIN and MAX. From those, and a suitable collection of unary gates, every function can be created. From Mouftah's paper[60]:

- "DeMorgan holds for ternary logic when the three types of inverters are used."
- Theorem: Any ternary function may be generated by means of the dyadic function MAX and the unary functions STI, NTI, PTI, FD, and RD. (Universal gates)
- Dyadic gates: TOR = x + y = MAX(x,y), TAND = x * y = MIN(x,y)

  It has been proven by Halpern and Yoeli[71] that an algebra composed of the MAX, the MIN and three unary operators $\overline{x}$, $x^{\overline{1}0}$, and x^{x} with the constant 1 function is a functionally complete system. Since the set of operators presented here are equivalent to those of the algebra operators presented here are equivalent to those of the algebra due to Halpern and Yoeli except the constant 1 which can be substituted by the FD or RD operator depending on the relationships given by the equations

(26.a [¬x = x + 1]) and (26.b [⌐x = x • 1]), therefore the theorem holds.

Still from Mouftah: Moreover, it has been shown in [72] how these ternary operators realize algebras due to Post[73], Rosser and Turquette[74], Yeoli and Rosenfeld[75], Vaca[76] and Mine et al[77], which have all proven to be functionally complete.

A.5. Troolean Algebra Theorems

According to Rick Sobie on 2006-03-10[78], "a troolean operator is a term I just invented." Nynaeve[79] wrote about troolean return values in Microsoft code (GetMessage returns 0, non-zero, or -1; Managed C++'s TriBool class uses 0, -1, and 2) Troolean is a neologism based on Boolean (after George Bool), meaning 3-state.

According to Mouftah[60] where + is MAX (TOR) and * is MIN (TAND), "It can be seen that theorems and laws presented above are nearly the same as those of the Boolea algebra with a little generalization. But there will be some differences due to the existence of three types of complements in the algebra presented here. This is shown clearly in the following theorems[...]". The following table is based on [80], with Mouftah's trinary formulas substituted in:

Table A.17. Troolean Algebra Theorems

Property	TOR	TAND
Idempotent	A + A = A	AA = A
Commutative	A + B = B + A	AB = BA
Associative	(A + B) + C = A + (B + C)	(AB)C = A(BC)
Absorptive	A + (AB) = A (AB) + (A⌋ B) = A	A(A + B) = A (A + B)(A + ⌊B) = A
Distributive	A + (B + C) = (A + B)(A + C)	A(B + C) = AB + AC
Complements	A + ⌋A = 1	A * ⌊A = $\overline{1}$
Identities	A + $\overline{1}$ = A A + 1 = 1	A * 1 = A A * $\overline{1}$ = $\overline{1}$
Involution		//A = A
Consensus? (not verified)	AB + $\overline{A}$C + BC = AB + $\overline{A}$C	(A + B)($\overline{A}$ + C)(B + C) = (A + B)($\overline{A}$ + C)
Consensus	A + (⌋ AB) = A + B (C + A)(C + ⌊A + B) = (C + A)(C + B) (AB) + (⌋ AC) + (BC) = (AB) + (⌋ AC)	A(⌊A + B) = AB (CA) + (C⌋ AB) = (CA) + (CB) (A + B)(⌊A + C)(B + C) = (A + B)(⌊A + C)
DeMorgan*	⌊(A + B) = ⌊A ⌊B $\overline{A + B}$ = $\overline{A}$ * $\overline{B}$ ⌋ (A + B) = ⌋ A⌋ B	⌊(AB) = ⌊A + ⌊B $\overline{AB}$ = $\overline{A}$ + $\overline{B}$ ⌋ (AB) = ⌋ A + ⌋ B

* As shown, DeMorgan's law applies for all three types of inverters. The inverter overbar notation would more clearly show these properties.

Mouftah also covers:

- Relationships that interrelate any inverter with the two others.
- Relationships that govern manipulations of the FD and RD operators

which is not yet listed here.

A.6. Unknown-State Logic

Unknown-State Logic is boolean logic with the addition of the ? state. ? is unknown, which means T or F since those are the only values in boolean logic.

Table A.18. Trinary Codings

Unbalanced	Balanced	Unknown-State Logic
0	$\overline{1}$	F
1	0	? = TF
2	1	T

The addition of the ? state enables *short-circuiting* to be used easily. *Short-circuiting* is a technique used in evaluating expressions, where the second operand expression is not evaluated if the result would evaluate to the same value regardless of the second operand. For example, a common idiom in the Perl programming language is "open file or die". If the "open file" expression evaluates to true, the "die" expression (which aborts the program with an error message) is not evaluated, since true OR any value is always true. However, if the "open file" expression evaluates to false, the "die" expression must be evaluated, because false OR any value can either be true or false. With a third, "unknown", state, the behavior is the following: if the "open" file expression evaluates to "unknown", then the "die" expression also must be evaluated.

A.6.1. NOT: Inversion

Tritwise inversion performs the NOT operation.

Table A.19. Unknown-State Logic NOT

A	$\overline{A}$
T	F
?	?
F	T

NOT ? is the same as the combination of NOT T and NOT F. NOT T = F and NOT F = T, so T and F combine to give TF, or ?.

There are 27 possible unary functions, although NOT is the only useful one which follows the rule that ? = TF.

A.6.2. AND, XOR, OR, XNOR, NAND

Table A.20. Unknown-State Logic Dyadic Functions

Inputs		0001	0110	0111	1000	1001	1110
A	**B**	**and**	**xor**	**or**	**nor**	**xnor**	**nand**
F	F	F	F	F	T	T	T
F	?	F	?	?	?	?	T
F	T	F	T	T	F	F	T
?	F	F	?	?	?	?	T
?	?	?	?	?	?	?	?
?	T	?	?	?	?	?	?
T	F	F	T	T	F	F	T
T	?	?	?	?	?	?	?
T	T	T	F	T	F	T	F

Table A.21. Unknown-State Logic Mapping to Balanced Logic and Unary Decomposition

Function	Unknown-State Logic	Unbalanced	Unary Gates		
and	FFF,F??,F?T	$\overline{1}\overline{1}\overline{1}$,$\overline{1}$00,$\overline{1}$01	$\overline{1}$	↘↗A	A
xor	F?T,???,T?F	$\overline{1}$01,000,10$\overline{1}$	A	0	$\overline{A}$
or	F?T,???,T?T	$\overline{1}$01,000,101	A	0	$\cap\overline{A}$
nor	T?F,???,F?F	10$\overline{1}$,000,$\overline{1}$0$\overline{1}$	$\overline{A}$	0	$\cup\overline{A}$
xnor	T?F,???,F?T	10$\overline{1}$,000,$\overline{1}$01	$\overline{A}$	0	A
nand	TTT,T??,T?F	111,100,10$\overline{1}$	1	$\cap\overline{A}$	$\overline{A}$

The table above was filled in by using the definition that ? = TF. For example, F and ? = (F and T)(F and F) = FF = F. Substitute ? for T and F in two expressions and combine them. Because ? is more of an extention to boolean algebra than a whole new system, binary functions are still represented by four bits.

A.7. SQL-like NULL Logic

In SQL, any expression involving the NULL value itself evaluates to NULL. Choosing F = $\overline{1}$, N = 0, and T = 1:

Table A.22. SQL-based NULL Logic Dyadic Functions

Inputs		0001	0110	0111	1000	1001	1110
A	**B**	**and**	**xor**	**or**	**nor**	**xnor**	**nand**
F	F	F	F	F	T	T	T
F	N	N	N	N	N	N	N
F	T	F	T	T	F	F	T
N	F	N	N	N	N	N	N
N	N	N	N	N	N	N	N
N	T	N	N	N	N	N	N
T	F	F	T	T	F	F	T
T	N	N	N	N	N	N	N
T	T	T	F	T	F	T	F

Table A.23. SQL-based NULL Logic

Function	Truth Table
and	FNF,NNN,FNT
xor	FNT,NNN,TNF
or	FNT,NNN,TNT
nor	TNF,NNN,FNF
xnor	TNF,NNN,FNT
nand	TNT,NNN,TNF

A.8. Works Cited

1. D.E. Knuth, The Art of Computer Programming - Volume 2: Seminumerical Algorithms, pp. 207-208. Addison-Wesley, 3rd ed., 1998. ISBN 0-201-89684-2. Available: http://jeff.tk/wiki/Image:Knuth-TaoCPVol2-pg207%2C8.pdf
2. The Elements of Computing Systems: Building a Modern Computer from First Principles by Noam Nisan and Shimon Schocken (MIT Press, 2005).
3. Connelly, Jeff. Jeff.tk - Trinary/Meetings. Available: http://jeff.tk/wiki/Trinary/Meetings
4. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary/Status. Available: http://jeff.tk/wiki/Trinary/Status
5. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary. Available: http://jeff.tk/wiki/Trinary
6. Chavez, Antonio and Connelly, Jeff. Trinary/Tools - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Tools
7. Chavez, Antonio and Connelly, Jeff. Trinary/CPU Simulation - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/CPU_Simulation

8. Chavez, Antonio. Trinary/Compiler - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Compiler
9. Swanson, William. *Introduction to Binary Numbers*. 2002. Available: http://www.swansontec.com/sbinary.htm
10. davido, Perl Monks. *Ternary operator (there's no Trinary operator)*. Available: http://www.perlmonks.org/?node_id=562203
11. Wall, Larry. perl.com: Apocalypse 3. Available: http://www.perl.com/pub/a/2001/10/02/apocalypse3.html?page=6
12. Wall, Larry. et. al. *Programming Perl*, 3rd. Edition. ISBN: 978-0596000271 Section 3.16: Conditional Operator.
13. The Antikythera Mechanism Research Project. Available: http://www.antikythera-mechanism.gr/project/overview
14. Lexikon. *Analog Computers*. Available: http://www.computermuseum.li/Testpage/AnalogComputers.htm
15. National Semiconductor, Application Note 31. September 2002. *Op Amp Circuit Collection*. Available: http://www.national.com/an/AN/AN-31.pdf
16. Goldstrasz, Thomas et. al. *Computers During World War Two*. Available: http://waste.informatik.hu-berlin.de/Diplom/WW2/default_e.html
17. Bains, Sunny. *Analog computer trumps Turing model*. EE Times. 11/03/1998. Available: http://www.eetimes.com/story/OEG19981103S0017
18. Principia Cybernetica Web: *Digital Computer*. Available: http://pespmc1.vub.ac.be/ASC/DIGITA_COMPU.html
19. Maney, Kevin. USA Today, September 1997. *Debate Stirs Over Origins of Computers*. Available: http://www.scl.ameslab.gov/ABC/Articles/Debate9-97.html
20. Bebop's BYTES Back. *Claude Shannon's master's Thesis*. Available: http://www.maxmon.com/1938ad.htm
21. Hannah, Eric. *United States Patent 7309866: Cosmic ray detectors for integrated circuit chips*. Available: http://tinyurl.com/3ysdmk
22. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://dx.doi.org/10.1511/2001.40.3268 and mirrored at http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
23. A. Srivastava and K. Venkatapathy, "Design and Implementation of a Low Power Ternary Full Adder," VLSI Design, vol. 4, no. 1, pp. 75-81, 1996. doi:10.1155/1996/94696. Available: http://jeff.tk/wiki/Image:Design_and_Implementation_of_a_Low_Power_Ternary_Full_Adder.pdf
24. J.T. Butler, Multiple-Valued Logic in VLSI, IEEE Computer Society Press Technology Series, Los Alamitos, California, 1991.
25. A.K. Jain, M.H. Abd-E1-Barr and R.J. Bolton, "A new structure for CMOS realization of MVL functions," International Journal of Electronics, vol. 74, no. 2, pp. 251-263, 1993.
26. S.L. Hurst, "Two decades of multiple valued logic--an invited tutorial," in Proceedings of IEEE International Symposium on Multiple-Valued Logic, p. 164, May 1988.
27. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
28. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
29. A. P. Dhande and V. T. Ingole. Design And Implementation Of 2 Bit Ternary ALU Slice. SETIT 2005, 3rd International Conference: Science of Electronic, Technologies of Information and Telecommunications. March 17-21, 2005, Tunisia. Available: http://jeff.tk/wiki/Image:Dhande%2C_Ingole_-_Design_and_Implementation_of_a_2_Bit_Ternary_ALU_Slice.pdf
30. P.C.Balla & A.Antoniou "low power dissipation MOS ternary logic family" IEEE journal on solid state circuits Vol. Sc-19 no-5, P.739-749, October 1984.
31. D.I.porat "Three valued digital system" Proc.IEE Vol.116, No6, P.947-955, June 1969.
32. K.C.Smith "The prospects of multivalued logic technology & application view " IEEE transaction on computer, Vol.-C -30, P-619-627 September 1981.

33. Chung-Yu-Wu"Design & application of pipelined dynamic CMOS ternary logic & simple ternary differential logic" IEEE journal on solid state circuits Vol.28, No-8, August 1993.
34. CS150. Berkeley EECS. *Bits, Bytes, Nibbles, and Words: Some definitions*. Available: http://inst.eecs.berkeley.edu/~cs150/sp98/lectures/week6_2/tsld002.htm
35. Slashdot. *Ternary Computing Revisited*. Available: http://slashdot.org/comments.pl?sid=23934
36. Sloppy. Slashdot | Ternary Computing Revisited. Monday November 19 2001. *Trits?* Available: http://slashdot.org/comments.pl?sid=23934&cid=2585807
37. Merrill, Roy D. *Ternary Logic in Digital Computers*. January 1965. Avaialble: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
38. Bowles, Gary-alexander. US Patent #5498980 Ternary/binary converter circuit (Publication Date: 03/12/1996). Available: http://www.freepatentsonline.com/5498980.html
39. Setun' W. H. Ware, S. N. Alexander, N. M. Astrahan, H. H. Goode, M. Rubinoff, P. Armer, L. Bers, H.d. Huskey, "Soviet computer technology - 1959," Communications of the ACM, pp. 149-150, 1960.. Available: http://jeff.tk/wiki/Image:Communications_of_the_ACM_-_Soviet_Computer_Technology_-_1959.pdf
40. Faden, David. Reverse Fad Productions: Flip. Available: http://www.revfad.com/flip.html
41. Crispin, M. Panda Programing. 1 April 2005. Network Working Group, Request for Comments: 4042. *UTF-9 and UTF-18 Efficient Transformation Formats of Unicode*. Available: http://www.ietf.org/rfc/rfc4042.txt RFC 4042
42. Aspinwall, Jim. eCoustics. *Hacking CPU Voltage to Speed Up Your PC*. Available: http://forum.ecoustics.com/bbs/messages/34579/147079.html
43. Engelhardt, Mike. *LTspice/SwitcherCAD III User's Manual*. Available: http://ltspice.linear.com/software/scad3.pdf
44. Howell, Louis and Raymond, Eric S. Available: http://jeff.tk/wiki/Trinary/Logic#TriINTERCAL_Manual:_5.5.2.1_UNARY_LOGICAL_OPERATORS
45. Connelly, Jeff. Trinary/Parts - Jeff.tk - First Purchase. Available: http://jeff.tk/wiki/Trinary/Parts#Shopping_List:_First_Purchase
46. All About Circuits. Producing negative supply rails - Urgent - All About Circuits. Available: http://forum.allaboutcircuits.com/showthread.php?t=10415
47. All About Circuits. negative supply - All About Circuits Available: http://forum.allaboutcircuits.com/showthread.php?t=876
48. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
49. Connelly, Jeff. Trinary Computer Architecture - Older Diagrams. Available: Image:Proposed Architecture 2.png, Image:Proposed architecture 1.png
50. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
51. Lite-On Electronics Inc. Part No. LTL-30EHJ. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-30EHJ.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1057-ND
52. Lumex. T-5mm LED, 6 leaded, multi-colored, 636 nm AlInGoP Red/574 nm, AlInGoP Green BiColor, 470 nm Ultra Super Blue, Water Color Lens Datasheet. Part #SSL-LX5099SIUBSUGB. Available: http://rocky.digikey.com/weblib/Lumex/Web%20Data/SSL-LX5099SIUBSUGB1.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=67-1829-ND
53. Lite-On Electronics Inc. Part No. LTL-293SJW. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-293SJW.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1038-ND
54. Quicktar. Current limiting resistor calculator for LEDs. Available: http://www.quickar.com/noqbestledcalc.htm
55. Eigenratios of Self-Interpreters: The Mark II OISC Self-Interpreter. Available: http://eigenratios.blogspot.com/2006/09/mark-ii-oisc-self-interpreter.html
56. Connelly, Jeff. Public Git Hosting - trinary.git/tree - circuits/. Available: [http://repo.or.cz/w/trinary.git?a=tree;f=circuits
57. Carothers D. Christopher. Evolution of Intel Microprocessors: 1971 to 2007. Available:

http://www.cs.rpi.edu/~chrisc/COURSES/CSCI-4250/SPRING-2004/slides/cpu.pdf
58. Connelly, Jeff. Jeff.tk - Trinary/Symbols/Tips. Available: http://jeff.tk/wiki/Trinary/Symbols/Tips
59. Grubb, Steve. Trinary.cc. Available: http://www.trinary.cc/
60. H. T. Mouftah May 1976 Proceedings of the sixth international symposium on Multiple-valued logic. *A study on the implementation of three-valued logic*. Available: http://jeff.tk/wiki/Image:P123-mouftah_Study_on_the_Implementation_of_Three-valued_Logic.pdf
61. Mouftah, H. T. A study on the implementation of three-valued logic H. T. Mouftah May 1976 Proceedings of the sixth international symposium on Multiple-valued logic. Available: http://jeff.tk/wiki/Image:P123-mouftah_Study_on_the_Implementation_of_Three-valued_Logic.pdf
62. D.I. Porat, "Three-valued digital systems", Proc, lEE, Vol. 116, No. 6, June 1969, pp. 947-954.
63. M. Yoeli and G. Rosenfeld, "Logical design of ternary switching circuits", IEEE Trans. Elect. Comp., Vol. EC-14, February 1965, pp. 19-29.
64. M. Bitran and M.J.O. Strutt, "Minimization of ternary logic and complete set of integrable circuits", Electron. and Commun., AE0, Band 25,No. 8, 1971, pp. 387-392.
65. R.S. Nutter and R.E. Swartwout, "A ternary logic minimization technique", Conference Record of the 1971 Symposium on the Theory and Applications of Multiple-valued Logic Design, May 1971, pp. i12~125.
66. Connelly, Jeff. Trinary/Unary Quick Reference - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Unary_Quick_Reference
67. Hyde, Randall. *Art of Assembly Language*. Available: http://webster.cs.ucr.edu/AoA/DOS/ch01/CH01-2.html
68. Howell, Louis and Raymond, Eric S. *The C-Intercal Supplemental Reference Manual*. 1992-01-18. Available: http://webster.cs.ucr.edu/AoA/DOS/ch01/CH01-2.html
69. Connelly, Jeff. *Unary Gates Inside Binary Gates*. 2001-11-17. Available: http://jeff.tk/bingates/
70. Grubb, Steve. Trinary.cc. Binary Operations. 2001. Available: http://www.trinary.cc/Tutorial/Algebra/Binary.htm
71. I. Halpern and M. Yoeli, "Ternary arithmetic unit", Proc. lEE, Vol. 115, No. i0, October 1968, pp. 1585-1588. Table II : Multiple input ternary operators
72. H.T. Mouftah and I.B. Jordan, "Integrated circuits for ternary logic", Proceedings of the 1974 International Symposium on Multiple-valued Logic, May 1974, pp. 285-302.
73. E.L. Post, "Introduction to a general theory of elementary propositions", Amer. J. Math., Vol. 43, 1921, pp. I~3-185.
74. J.B. Rosser and A.R. Turquette, "Many-valued logics", North-Holland Publishing Co., Amsterdam, 1952.
75. M. Yoeli and G. Rosenfeld, "Logical design of ternary switching circuits", IEEE Trans. Elect. Comp., Vol. EC-14, February 1965, pp. 19-29.
76. R. Vacca, "A three-valued system of logic and its applications to base three digital circuits", Proc. Intern. Conf. Inform. Processing, (UNESCO), June 1959, pp. 407-414.
77. H. Mine, T. Hasegawa, M. Ikeda and T. Shintani, "A construction of ternary logic circuits", Electron. Commun. in Japan, Vol. 51-C, No. 12, pp. 133-140.
78. Sobie, Rick. Troolean operators, Available: http://sci.tech-archive.net/Archive/sci.physics/2006-03/msg00869.html
79. Nynaeve. Blog Archive - The troolean strikes back. Available: http://www.nynaeve.net/?p=87
80. CSE 460 - Spring 2006, Boolean Algebra Definitions, Theorems, and Postulates, Available: http://www.arl.wustl.edu/~lockwood/class/cse460/ba.pdf]

Appendix B: Arithmetic

B.1. Balanced Arithmetic

> *"Ahhh, what an awful dream. Ones and zeroes everywhere...[shudder] and I thought I saw a two."*
> *-- Bender*
> *"It was just a dream, Bender. There's no such thing as two". -- Fry*
> -- Futurama

What Fry says rings true in balanced trinary.

> *"There can only be one"*

Positive and negative one, that is. Lack of being is zero. Balanced trinary uses digits {-1,0,+1}, rather than {0,1,2}. To map unbalanced trinary to balanced, subtract one (TODO: describe substitution and subtraction/addition methods, advantages of each--probably make a new section on unbalanced/balanced conversion). Because the negative sign makes digits longer than 0 or 1, it is often written above the numeral as $\overline{1}$. You can use the Extensions:Trinary MediaWiki extension[81] I wrote to help with this. In ASCII, *i* can be used, and it can be enclosed in a tag to produce the aesthetically pleasing $\overline{1}$.

Unbalanced and balanced conversion chart:

Table B.1. Unbalanced and Balanced Trinary Conversion Chart

Unbalanced	Balanced
0	$\overline{1}$
1	0
2	1

James Allright's Balanced Ternary Web Page[82] includes documentation on arithmetic operations that can be performed on balanced trinary numbers.

TriINTERCAL manual, section 5.4 uses a balanced trinary system where the unbalanced 2 digit is -1 in balanced trinary:

> Note that though TriINTERCAL considers all numbers to be unsigned, nothing prevents the programmer from implementing arithmetic operations that treat their operands as signed. Three's complement is one obvious choice, but balanced ternary notation is also a possibility. This latter is a very pretty and symmetrical system in which all 2 trits are treated as if they had the value -1.

We will not use this system, since changing the 2 to -1 drastically alters the truth tables when converting between balanced and unbalanced. Instead, the conversion chart above will be used.

Knuth covers the history of balanced ternary:

> Representation of numbers in the balanced ternary system is implicitly present in a famous

mathematical puzzle, commonly called "Bachet's problem of weights" although it was already stated by Fibonacci four centuries before Bachet wrote his book, and by Tabari in Persia more than 100 years before Fibonacci[83] Positional number systems with negative digits were invented by J.Colson [84], then forgotten and rediscovered about 100 years later by Sir John Lesie[85] and A. Cauchy[86]. Cauchy pointed out that negative digits make it unnecessary for a person to memorize the multiplication table past 5 x 5. A claim that such number systems were known in India long ago[87] has been refuted by K. S. Shukla[88]. The first true appearance of "pure" balanced ternary notation was in an article by Leon Lalanne[89], who was a designer of mechanical devices for arithmetic. The system was only mentioned rarely for 100 years after Lalanne's paper, until the development of the first electronic computers at the Moore School of Electrical Engineering in 1945-1946; at that time it was given serious consideration along with the binary system as a possible replacement for the decimal system. The complexity of arithmetic circuitry for balanced ternary arithmetic is not much greater than it is for the binary system, and a given number requires only ln 2/ln 3 ≈ 63% as many digit positions for its representation. Discussion of the balanced ternary number system appear in[90] and in [91]. The experimental Russian computer SETUN was based on balanced ternary notation[92], and perhaps the symmetric properties and simple arithmetic of this number system will prove to be quite important some day--when the "flip-flop" is replaced by a "flip-flap-flop".

Further information regarding the history of balanced ternary is available[93].

Mouftah[94] page 137, Table I introduces a unit-distance ternary code and provides equivalent trits for signed (balanced) ternary and decimal, from -4 to +13.

n trits of balanced trinary can represent within $\pm(3^n - 1)/2$. The minus 1 is to account for representing 0 in the middle, and the division by half is for the balanced part.

B.1.1. Negation: Inversion

Numbers are negated by swapping all $\overline{1}$'s with 1 and vice versa, according to Knuth[1]. This is a trivial operation, simpler than forming the two's complement in binary (according to James Allwright on sci.math when announcing the Balanced Ternary System arithmetic package):

Table B.2. Simple Ternary Inverter Truth Table

A	-A
1	$\overline{1}$
0	0
$\overline{1}$	1

This is function $10\overline{1}$ balanced, or 210 unbalanced. $10\overline{1}$ is also known as the invert operator. In balanced trinary notation, logical inversion = arithmetic negation, unlike in binary where (for example) x86 PCs have both NOT and NEG to account for the differences between 2's complement negation and arithmetic negation.

B.1.2. Sign

"The sign of a number is given by its most significant nonzero bit, and in general we can compare any two numbers by reading them from left to right and using lexicographic order, as in the decimal system."[1]

Logically, the sign can be computed by first checking most-significant trit. If it is non-zero, return it. If it is zero, continue on right-ward to the next trit and repeat. If the the last trit is reached, that means they are all zero, so return 0. Note that in trinary, zero is considered to have its own sign, neither positive nor negative, unlike binary where zero may arbitrarily be considered positive or negative, or worse yet there may be both positive and negative zero (or "minus zero"[95]).

Here is a SPICE function (that can be attached to behavioral voltage source in LTspice, 'bv'; requires function definitions as defined in `alu-fast.asc`) to compute the sign of a 3-trit number:

```
; Sign of balanced trinary number (most-significant non-zero trit)
.func tsign(c,b,a){if(isnt_0(c),c,if(isnt_0(b),b,if(isnt_0(a),a,0)))}
```

This can be implemented in hardware using two multiplexers:

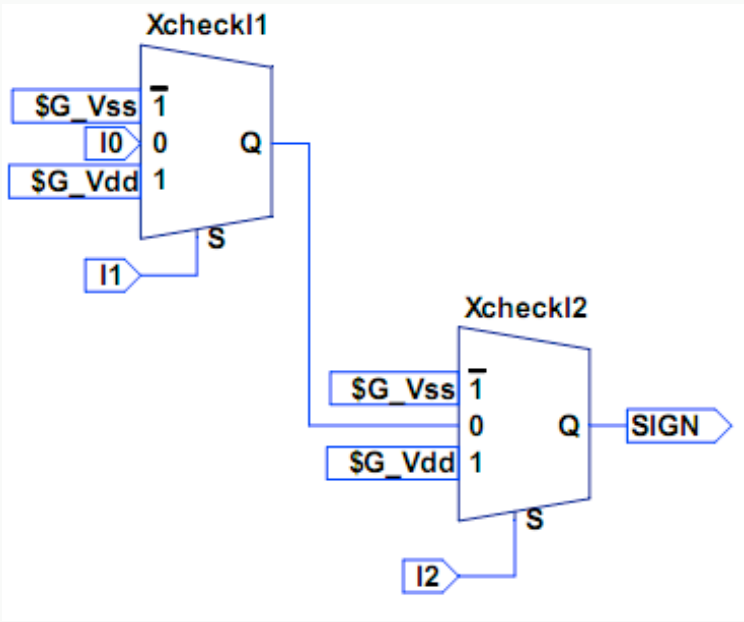


Figure B.1. Circuit to compute the sign ($\bar{1}$, 0, or 1) of a 3-trit balanced trinary number. Custom design by Jeff Connelly.

And it looks like this:

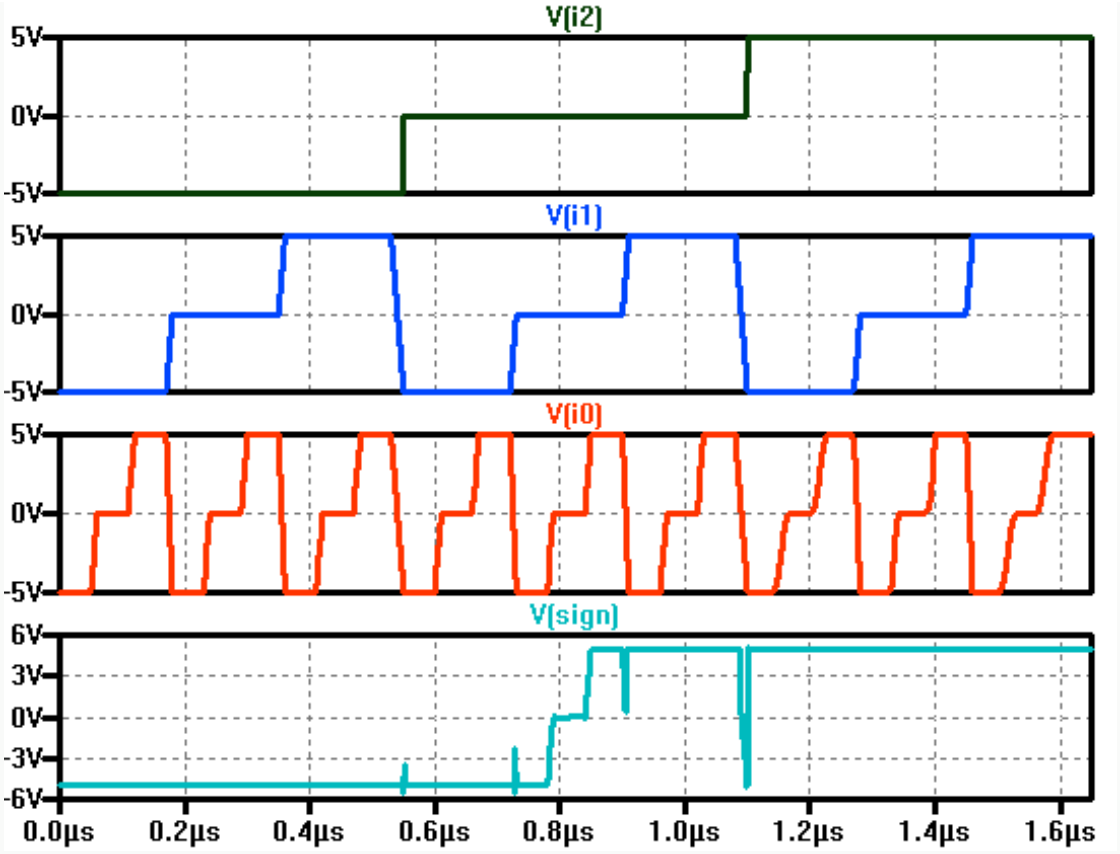


Figure B.2. Timing diagram of circuit to calculate sign of 3-trit balanced trinary number.

The sign is 0 only with inputs 000, negative for inputs below 0, and positive for inputs above.

A 4-trit sign detector is available in tsign4.asc. This circuit simply takes a 4-trit balanced trinary number, and outputs a trit for the sign.

Chirag's results from testing the sign detector PCB from Trinary/PCBs in lab:

```
I3 =0, I2=0, I1=0, I0=0 --> SIGN =0
I3 =1, I2=1, I1=1, I0=1 --> SIGN =1
I3 =i, I2=i, I1=i, I0=i --> SIGN =i
I3 =1, I2=i, I1=i, I0=i --> SIGN =1
I3 =i, I2=i, I1=i, I0=1 --> SIGN =i
I3 =0, I2=1, I1=1, I0=1 --> SIGN =1
I3 =0, I2=i, I1=i, I0=i --> SIGN =i
```

B.1.3. Even/Odd

According *Third Base*[96], note this is for unbalanced ternary: "A ternary numeral represents an even number if the numeral has an even number of 1s. (The reason is easy to see when you count powers of 3, which are invariably odd.)"

B.1.4. Rounding to Nearest Integer (Truncation)

Compared to binary, balanced trinary has superior round-off properties[97]. Rounding to the nearest integer is equivalent to truncation; "in other words, we can simply delete everything to the right of the radix point."[1] to perform the rounding operation.

Merrill[98] explains the reason for this property. To round $x/3^k = x_{n-1}3^{n-1-k} + x_{n-2}3^{n-2-k} + ... + x_k + ... + x_0 3^{-k}$ to the nearest integer, you can truncate the number to only include the first n-k significant digits, because:

$-1/2 < x_{k-1}3^{-1} + x_{k-2}3^{-2} + ... + x_0 3^{-k} < -1/2$

(Note: we think this should be from -1/2 to +1/2, and that Merrill made a mistake.) In binary, you have to examine the k^{th} least significant digit to determine which direction to round, but in balanced trinary this is not necessary.

B.1.5. Addition: Half-adder

The trinary half-adder truth table is as follows:

Table B.3. Half-Adder Truth Table

A	B	C	S
$\overline{1}$	$\overline{1}$	$\overline{1}$	1
$\overline{1}$	0	0	$\overline{1}$
$\overline{1}$	1	0	0
0	$\overline{1}$	0	$\overline{1}$
0	0	0	0
0	1	0	1
1	$\overline{1}$	0	0
1	0	0	1
1	1	1	1

As an example, the first row is -1 + -1 = -2 = (-1*3^1+1=-3+1=-2)). From the table, the carry and sum output truth tables and formulas can be derived:

- C = $\overline{1}$00,000,001 Balanced
- S = 1$\overline{1}$0,$\overline{1}$01,011 Balanced

- C = ⌐A, 0, ¬A
- S = (┘ $\overline{A}$ ▽ 0)▽(└$\overline{A}$) [cycle down], A, $\overline{⌐ └A}$ [shift up]

A half-adder is available in the repository as `half_adder`. 3:1 multiplexers are used to implement the dyadic functions:

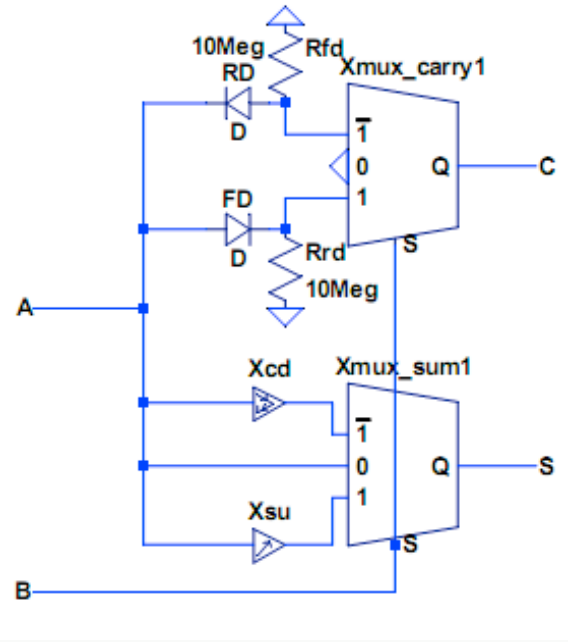


Figure B.3. Schematic of a custom half-adder design.

The design works using the default diode model:

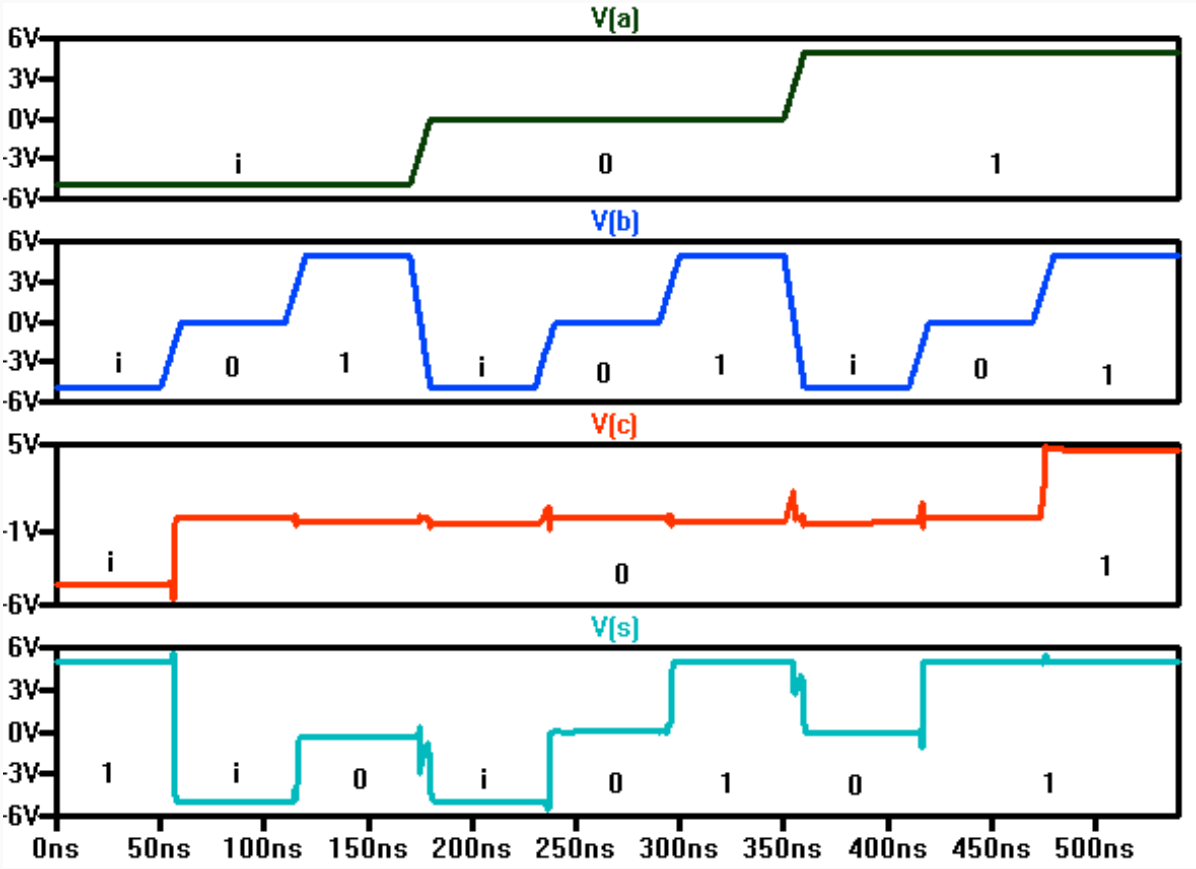


Figure B.4. Timing diagram of trinary half-adder, using FD and RD with the "D" model.

Using 1N4148 diodes for the RD and FD gates breaks the adder where shown:

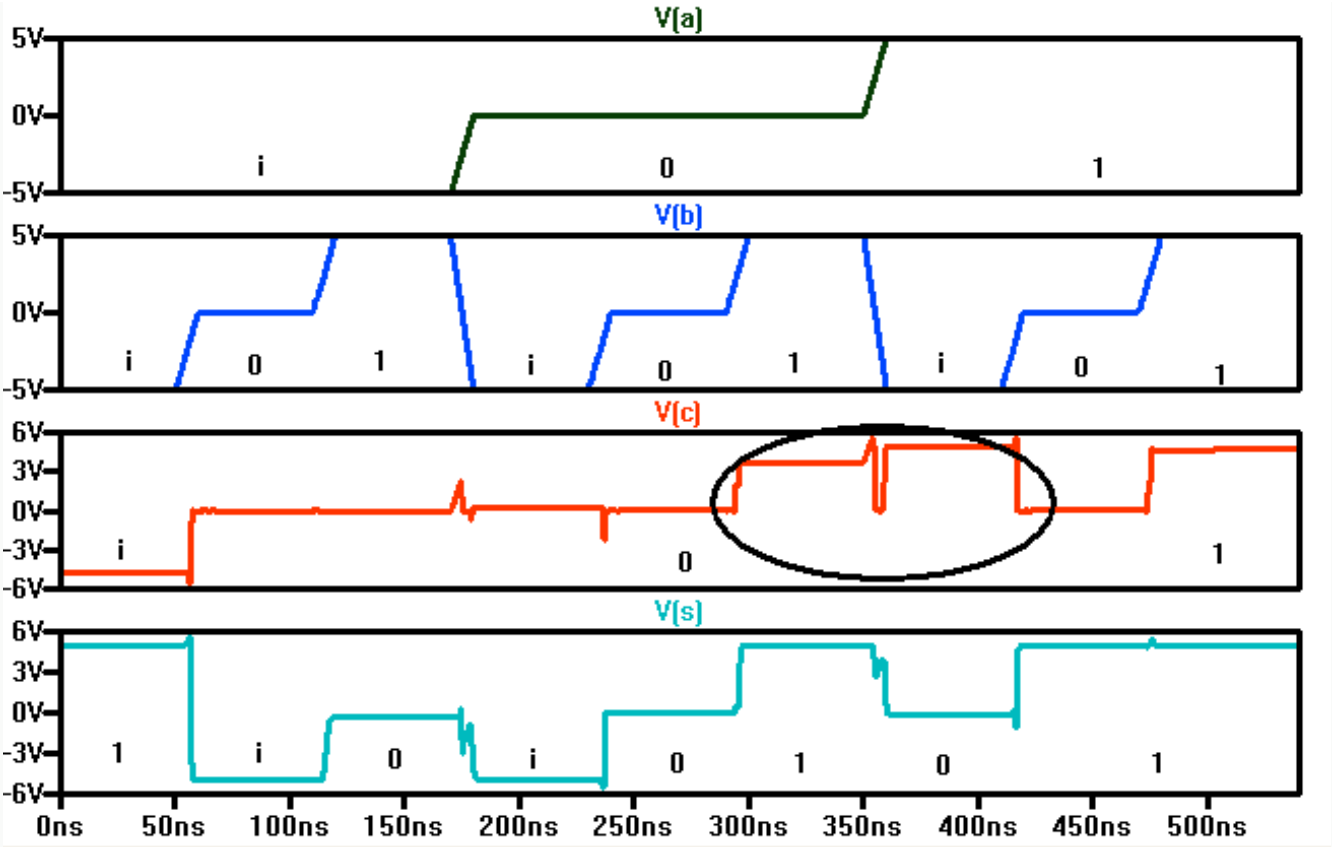


Figure B.5. Timing diagram of adder using 1N4148 diodes. Incorrect outputs circled.

The 1N4448 diode also causes the timing diagram to be incorrect:

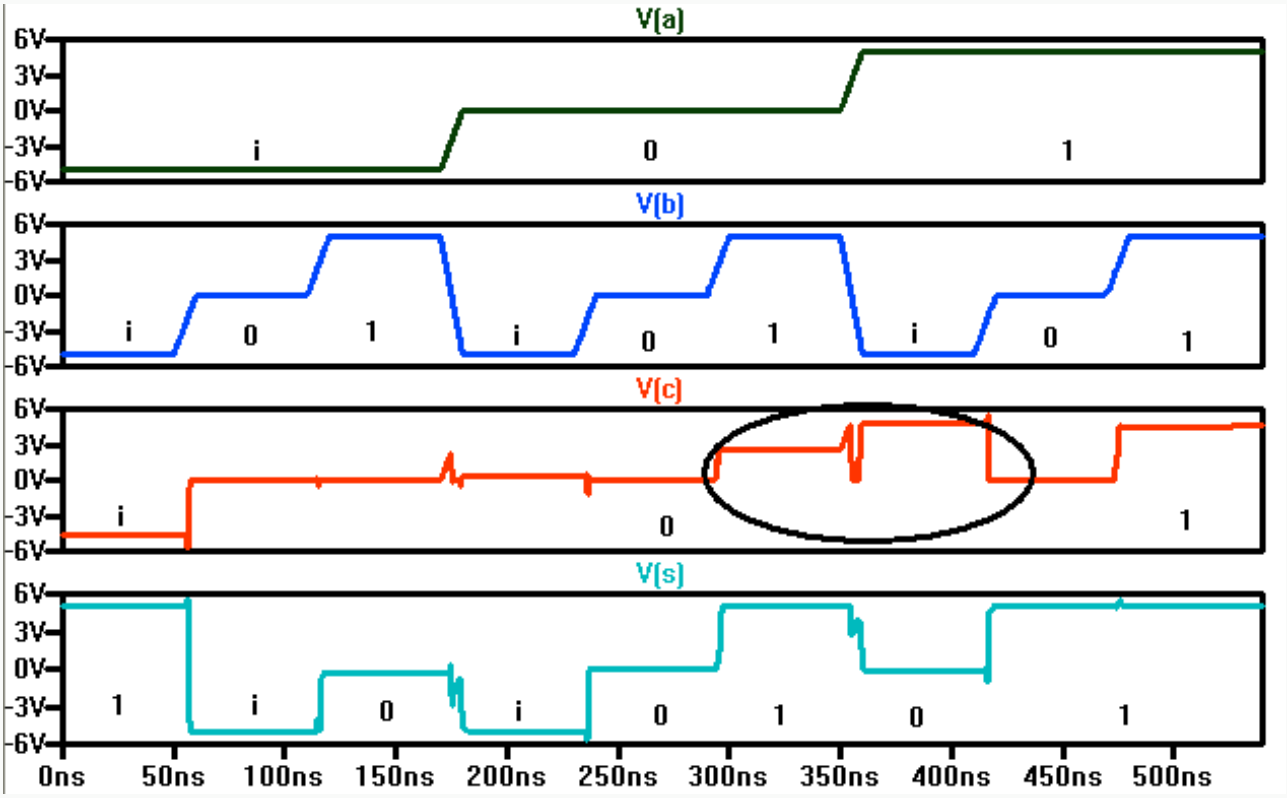


Figure B.6. Timing diagram of adder using 1N4448 diodes. Incorrect outputs circled.

This is what a PCB layout looks like:

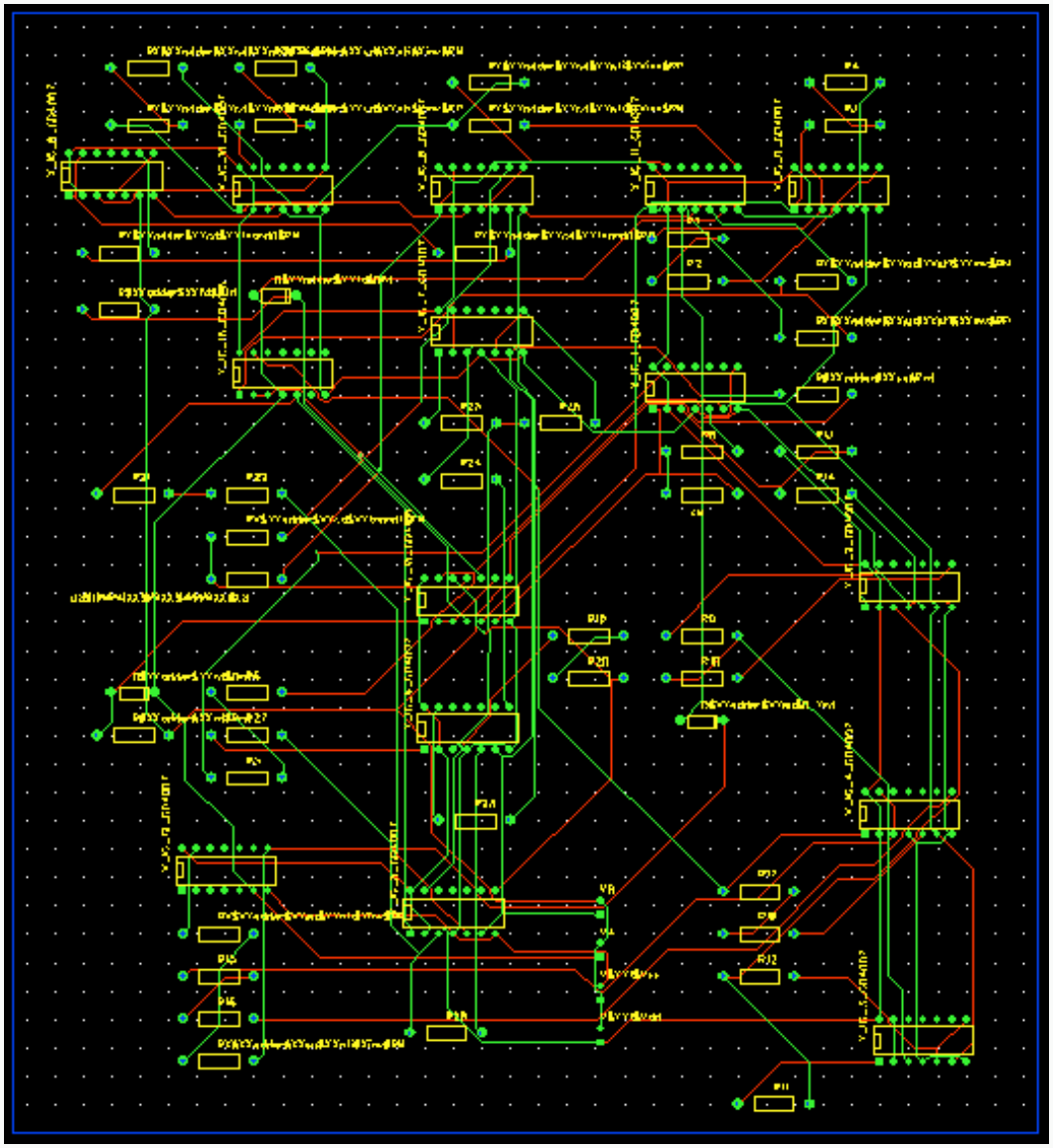


Figure B.7. PCB layout of a 2-trit half adder.

However, we did not use the half adder in our computer system, even where possible. Instead, we exclusively used half adders, both for simplicity (all the adders are full adders) and extendability (if desired, the carry-in trit could be set to the result of the last operation, at some point in the future, instead of 0.)

B.1.6. Addition: Full-Adder

The full-adder adds 3 trits, A, B, and C_{in} (carry in), to produce a sum and a carry out.

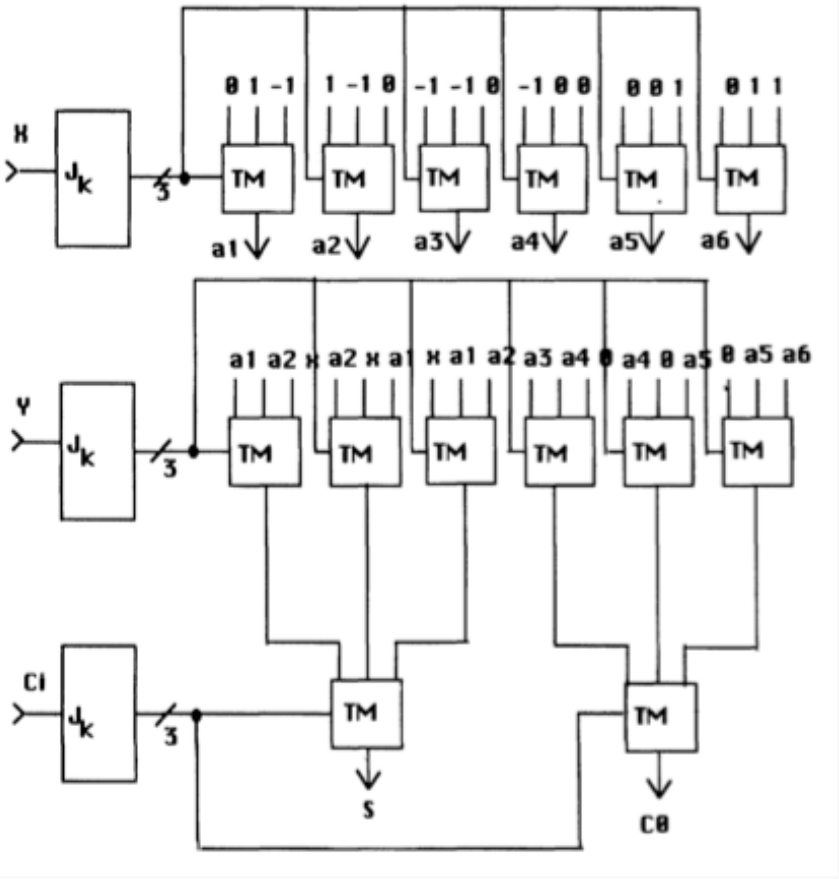
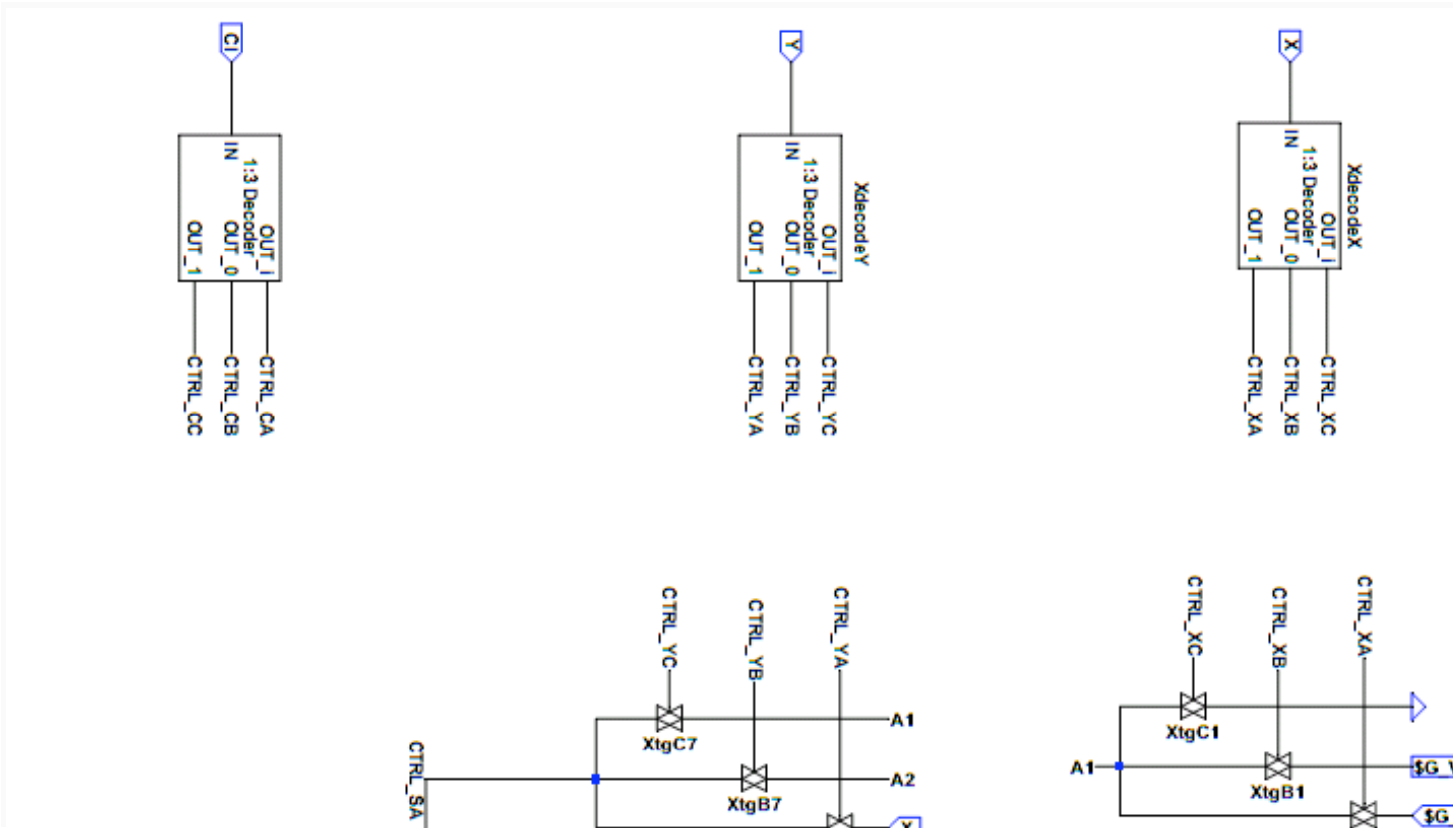


Figure B.8. Ternary Full-Adder, Figure 8 from Image:Design and Implementation of a Low Power Ternary Full Adder.pdf.

The actual schematic of the full adder is:



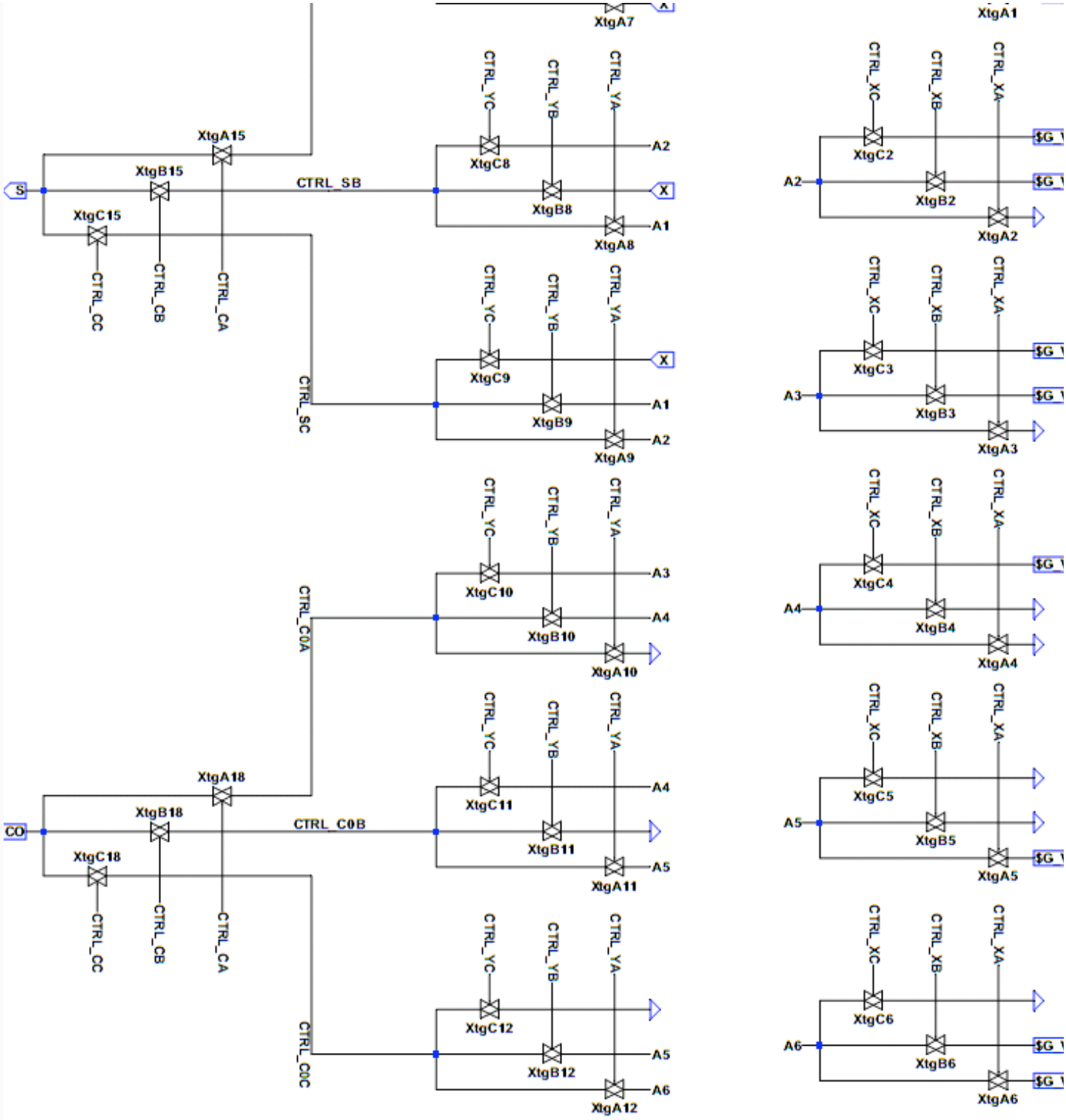


Figure B.9. Full adder schematic.

Timing diagram of the full-adder, implemented as above:

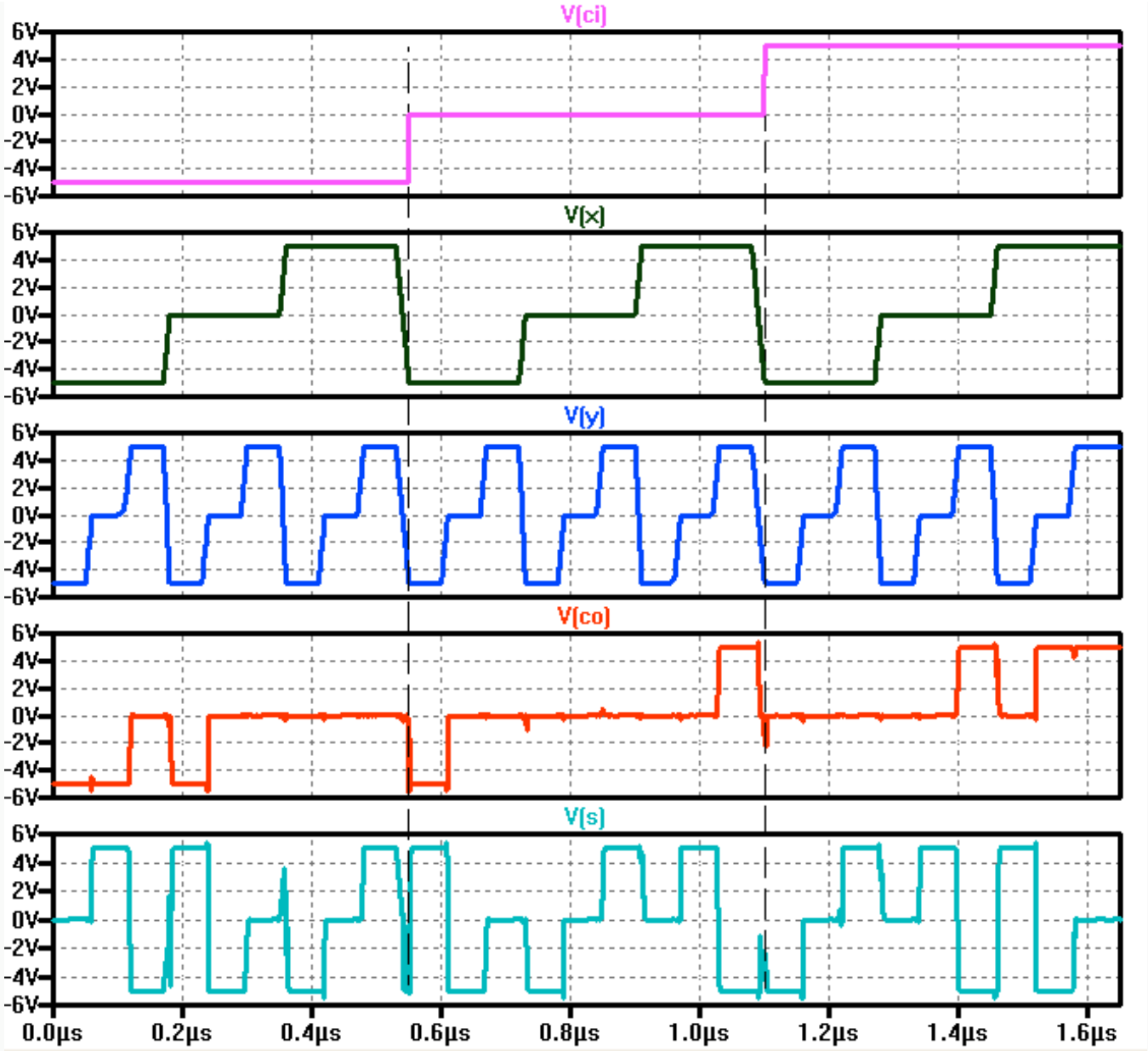


Figure B.10. Timing diagram of full-adder.

Internal Signals

Although the 4-trit ripple-carry adder worked flawlessly in simulation, when constructed on PCB that was not the case. To help isolate the problem, we compared the internal signals from simulation to reality. The internal signals are too large to fit here but are separately available[99] [100] [101]. These signals are primarily useful for debugging, but they also illustrate our process. The following nets are defined in the full adder, connecting to the following parts and pins. A net is a "wire", and the parts and pins are listed as IC_CD4007_1.2, for example, which indicates part IC_CD4007_1, pin number 2.

Table B.4. Nets in the Full Adder

Net	Connections
Power	Vdd.1 IC_CD4007_1.2 IC_CD4007_1.14 IC_CD4007_3.2 IC_CD4007_3.14 IC_CD4007_4.14 IC_CD4016_6.8 IC_CD4016_6.14 IC_CD4016_7.1 IC_CD4016_7.14 IC_CD4016_8.14 IC_CD4016_9.1

Vdd	IC_CD4016_9.11 IC_CD4016_9.14 IC_CD4016_10.4 IC_CD4016_10.14 IC_CD4016_11.14 IC_CD4016_12.14 IC_CD4016_13.14 IC_CD4016_14.14 IC_CD4007_15.2 IC_CD4007_15.14 IC_CD4007_16.14 IC_CD4016_18.14 IC_CD4016_19.14 IC_CD4007_20.2 IC_CD4007_20.14 IC_CD4007_21.2 IC_CD4007_21.14 IC_CD4007_23.14
Vss	Vss.2 IC_CD4007_1.4 IC_CD4007_1.7 IC_CD4007_3.4 IC_CD4007_3.7 IC_CD4007_4.4 IC_CD4007_4.7 IC_CD4016_6.1 IC_CD4016_6.7 IC_CD4016_7.4 IC_CD4016_7.7 IC_CD4016_7.11 IC_CD4016_8.1 IC_CD4016_8.7 IC_CD4016_8.8 IC_CD4016_9.7 IC_CD4016_10.7 IC_CD4016_11.7 IC_CD4016_12.7 IC_CD4016_13.7 IC_CD4016_14.7 IC_CD4007_15.4 IC_CD4007_15.7 IC_CD4007_16.4 IC_CD4007_16.7 IC_CD4016_18.7 IC_CD4016_19.7 IC_CD4007_20.4 IC_CD4007_20.7 IC_CD4007_21.4 IC_CD4007_21.7 IC_CD4007_23.4 IC_CD4007_23.7
0	Vdd.2 Vss.1 VA.2 VB.2 VCI.2 IC_CD4016_6.4 IC_CD4016_6.11 IC_CD4016_7.8 IC_CD4016_8.4 IC_CD4016_8.11 IC_CD4016_9.4 IC_CD4016_9.8 IC_CD4016_10.1 IC_CD4016_12.11 IC_CD4016_14.1 IC_CD4016_14.8
	Inputs
X	VA.1 IC_CD4007_1.6 IC_CD4007_3.6 IC_CD4016_10.8 IC_CD4016_11.11 IC_CD4016_12.4
Y	VB.1 IC_CD4007_21.3
CI	VCI.1 IC_CD4007_20.6 IC_CD4007_21.6
	Outputs
CO	IC_CD4016_18.10 IC_CD4016_19.2 IC_CD4016_19.3
S	IC_CD4016_18.2 IC_CD4016_18.3 IC_CD4016_18.9
	Decoder Outputs from Adder Inputs
XFA$CTRL_XA	IC_MDP1403-12K_2.11 IC_MDP1403-12K_2.12 IC_CD4007_4.6 IC_CD4016_6.12 IC_CD4016_6.13 IC_CD4016_7.6 IC_CD4016_8.5 IC_CD4016_9.12 IC_CD4016_9.13
XFA$CTRL_XB	IC_MDP1403-12K_2.8 IC_MDP1403-12K_5.14 IC_CD4016_6.6 IC_CD4016_7.5 IC_CD4016_8.12 IC_CD4016_8.13 IC_CD4016_9.6 IC_CD4016_10.5
XFA$CTRL_XC	IC_MDP1403-12K_2.6 IC_CD4007_3.8 IC_CD4007_4.3 IC_CD4016_6.5 IC_CD4016_7.12 IC_CD4016_7.13 IC_CD4016_8.6 IC_CD4016_9.5 IC_CD4016_10.13
XFA$CTRL_YA	IC_MDP1403-12K_5.10 IC_MDP1403-12K_5.11 IC_CD4016_10.6 IC_CD4016_11.5 IC_CD4016_12.12 IC_CD4016_12.13 IC_CD4016_13.6 IC_CD4016_14.5 IC_CD4007_16.6
XFA$CTRL_YB	IC_CD4016_11.12 IC_CD4016_11.13 IC_CD4016_12.6 IC_CD4016_13.5 IC_CD4016_14.12 IC_CD4016_14.13 IC_MDP1403-12K_17.13 IC_MDP1403-12K_17.14
XFA$CTRL_YC	IC_MDP1403-12K_5.7 IC_CD4016_10.12 IC_CD4016_11.6 IC_CD4016_12.5 IC_CD4016_13.12 IC_CD4016_13.13 IC_CD4016_14.6 IC_CD4007_15.5 IC_CD4007_16.3
XFA$CTRL_CA	IC_CD4016_18.12 IC_CD4016_18.13 IC_CD4007_21.8 IC_MDP1403-12K_22.1 IC_CD4007_23.3

XFA$CTRL_CB	IC_CD4016_18.6 IC_CD4016_19.5 IC_MDP1403-12K_22.12 IC_MDP1403-12K_22.13
XFA$CTRL_CC	IC_MDP1403-12K_17.9 IC_MDP1403-12K_17.10 IC_CD4016_18.5 IC_CD4016_19.13 IC_CD4007_23.6
	First-Level Transmission Gate Outputs
XFA$A1	IC_CD4016_6.2 IC_CD4016_6.3 IC_CD4016_6.9 IC_CD4016_10.11 IC_CD4016_11.4 IC_CD4016_12.8
XFA$A2	IC_CD4016_6.10 IC_CD4016_7.2 IC_CD4016_7.3 IC_CD4016_11.1 IC_CD4016_11.8 IC_CD4016_12.1
XFA$A3	IC_CD4016_7.9 IC_CD4016_7.10 IC_CD4016_8.2 IC_CD4016_13.1
XFA$A4	IC_CD4016_8.3 IC_CD4016_8.9 IC_CD4016_8.10 IC_CD4016_13.4 IC_CD4016_13.11
XFA$A5	IC_CD4016_9.2 IC_CD4016_9.3 IC_CD4016_9.9 IC_CD4016_13.8 IC_CD4016_14.11
XFA$A6	IC_CD4016_9.10 IC_CD4016_10.2 IC_CD4016_10.3 IC_CD4016_14.4
	Second-level Transmission Gate Outputs
XFA$CTRL_C0A	IC_CD4016_12.10 IC_CD4016_13.2 IC_CD4016_13.3 IC_CD4016_18.11
XFA$CTRL_C0B	IC_CD4016_13.9 IC_CD4016_13.10 IC_CD4016_14.2 IC_CD4016_19.4
XFA$CTRL_C0C	IC_CD4016_14.3 IC_CD4016_14.9 IC_CD4016_14.10 IC_CD4016_19.1
XFA$CTRL_SA	IC_CD4016_10.9 IC_CD4016_10.10 IC_CD4016_11.2 IC_CD4016_18.1
XFA$CTRL_SB	IC_CD4016_11.3 IC_CD4016_11.9 IC_CD4016_11.10 IC_CD4016_18.8
XFA$CTRL_SC	IC_CD4016_12.2 IC_CD4016_12.3 IC_CD4016_12.9 IC_CD4016_18.4
	Deeper Internal Signals
XFA$XX1$IN_pti	IC_MDP1403-12K_17.3 IC_CD4007_20.3 IC_CD4007_20.13
XFA$XX1$XX0nor$NI	IC_CD4007_23.2 IC_CD4007_23.13
XFA$XX1$XX0nor$NN	IC_MDP1403-12K_22.3 IC_CD4007_23.5 IC_CD4007_23.8
XFA$XX1$XX0nor$NP	IC_MDP1403-12K_22.2 IC_CD4007_23.1
XFA$XX1$XX1pti$NTI_Out	IC_MDP1403-12K_17.4 IC_CD4007_20.8
XFA$XX1$XX1pti$STI_Out	IC_MDP1403-12K_17.11 IC_MDP1403-12K_17.12
XFA$XX1$XX1sti$NTI_Out	IC_MDP1403-12K_17.6 IC_CD4007_20.5
XFA$XX1$XX1sti$PTI_Out	IC_MDP1403-12K_17.5 IC_CD4007_20.1
XFA$XX1$XXinti$PTI_Out	IC_MDP1403-12K_17.7 IC_CD4007_21.13
XFA$XX1$XXinti$STI_Out	IC_MDP1403-12K_17.8 IC_MDP1403-12K_22.14
XFA$XXdecodeX$IN_pti	IC_CD4007_1.3 IC_CD4007_1.13 IC_MDP1403-12K_2.1
XFA$XXdecodeX$XX0nor$NI	IC_CD4007_4.2 IC_CD4007_4.13
XFA$XXdecodeX$XX0nor$NN	IC_CD4007_4.5 IC_CD4007_4.8 IC_MDP1403-12K_5.1
XFA$XXdecodeX$XX0nor$NP	IC_MDP1403-12K_2.7 IC_CD4007_4.1
XFA$XXdecodeX$XX1pti$NTI_Out	IC_CD4007_1.8 IC_MDP1403-12K_2.2
XFA$XXdecodeX$XX1pti$STI_Out	IC_MDP1403-12K_2.13 IC_MDP1403-12K_2.14

XFA$XXdecodeX$XX1sti$NTI_Out	IC_CD4007_1.5 IC_MDP1403-12K_2.4
XFA$XXdecodeX$XX1sti$PTI_Out	IC_CD4007_1.1 IC_MDP1403-12K_2.3
XFA$XXdecodeX$XXinti$PTI_Out	IC_MDP1403-12K_2.5 IC_CD4007_3.13
XFA$XXdecodeX$XXinti$STI_Out	IC_MDP1403-12K_2.9 IC_MDP1403-12K_2.10
XFA$XXdecodeY$IN_pti	IC_CD4007_3.1 IC_MDP1403-12K_5.2 IC_CD4007_15.6
XFA$XXdecodeY$XX0nor$NI	IC_CD4007_16.2 IC_CD4007_16.13
XFA$XXdecodeY$XX0nor$NN	IC_CD4007_16.5 IC_CD4007_16.8 IC_MDP1403-12K_17.2
XFA$XXdecodeY$XX0nor$NP	IC_CD4007_16.1 IC_MDP1403-12K_17.1
XFA$XXdecodeY$XX1pti$NTI_Out	IC_CD4007_3.5 IC_MDP1403-12K_5.3
XFA$XXdecodeY$XX1pti$STI_Out	IC_MDP1403-12K_5.12 IC_MDP1403-12K_5.13
XFA$XXdecodeY$XX1sti$NTI_Out	IC_MDP1403-12K_5.5 IC_CD4007_15.8
XFA$XXdecodeY$XX1sti$PTI_Out	IC_MDP1403-12K_5.4 IC_CD4007_15.13
XFA$XXdecodeY$XXinti$PTI_Out	IC_MDP1403-12K_5.6 IC_CD4007_15.1
XXnegY$NTI_Out	IC_CD4007_21.5 IC_MDP1403-12K_22.5
XXnegY$PTI_Out	IC_CD4007_21.1 IC_MDP1403-12K_22.4
_Y	IC_CD4007_3.3 IC_CD4007_15.3 IC_MDP1403-12K_22.10 IC_MDP1403-12K_22.11

Note that CI names the control signals differently:

Table B.5.: Full-Adder Control Signals

Signal Input	Control Signal (others are $\overline{1}$)
X = $\overline{1}$	CTRL_XC = 1
X = 0	CTRL_XB = 1
X = 1	CTRL_XA = 1
Y = $\overline{1}$	CTRL_YC = 1
Y = 0	CTRL_YB = 1
Y = 1	CTRL_YA = 1
CI = $\overline{1}$	**CTRL_CA = 1**
CI = 0	**CTRL_CB = 1**
CI = 1	**CTRL_CC = 1**

However, in the schematic the multiplexers are reversed appropriately. The order of inputs on the top of the trinary multiplexers, from left to right, is always $\overline{1}$, 0, 1. Not the actual values, but inputs that will be passed if the signal corresponding to $\overline{1}$, 0, or 1 is high.

Debugging

Debugging steps we took are as follows. Note that IC_CD4016_14.3 means IC #14, which is a CD4016, pin 3.

- Verify CTRL_CA, CTRL_XA, CTRL_YA through CTRL_CC, CTRL_XC, CTRL_YC (see above). Mostly done.
- Verify A1 through A6. These are correct.
- Verify CTRL_SA,SB,SC and CTRL_C0A,B,C.
    1. Set all inputs to 1
    2. CTRL_C0C is IC_CD4016_14.3, should be 1, actually 0

3. CTRL_C0B is IC_CD4016_13.9, should be 1, actually 0
4. CTRL_C0A is IC_CD4016_12.10, should be 0, actually 0
5. CTRL_SC is IC_CD4016_12.2, should be 0, actually 1
6. CTRL_SB is IC_CD4016_11.3, should be $\overline{1}$, actually 0
7. CTRL_SA is IC_CD4016_10.9, should be 1, actually $\overline{1}$

Part 2

- 1. All inputs should be 1
  2. CTRL_YA, which is IC_MDP1403-12K_5.10 (IC #5, pin #10) should be 1, actually $\overline{1}$
  3. CTRL_YC, which is IC_MDP1403-12K_5.7, should be $\overline{1}$, actually 1
  4. xdecodey$IN_PTI, which is IC_CD4007_3.1, should be $\overline{1}$, actually 1

The .net2 file shows what signals map to which chips. The components XFA$XXdecodeY$XX1pti and XFA$XdecodeY$XXinti appear to be broken. An input of Y = 1 is given to both of these components. PTI of 1 is $\overline{1}$, which is the output signal XFA$XXdecodeY$IN_PTI, however, an output of 1 is observed in lab. Additionally, the NTI of 1 is $\overline{1}$, which is XFA$CTRL_YC, but 1 was observed in lab. So the NTI and PTI are not functioning as expected. These components are constructed out of chip #3, #5 and #15, which are suspect.

```
* Chip #3 - CD4007 pinout:
*       1: XFA$XXdecodeY$IN_pti
*       2: $G_Vdd
*       3: _Y
*       4: $G_Vss
*       5: XFA$XXdecodeY$XX1pti$NTI_Out
*       6: X
*       7: $G_Vss
*       8: XFA$CTRL_XC
*       9: NC__36
*       10: NC__37
*       11: NC__38
*       12: NC__39
*       13: XFA$XXdecodeX$XXinti$PTI_Out
*       14: $G_Vdd

* Chip #5 - MDP1403-12K pinout:
*       1: XFA$XXdecodeX$XX0nor$NN
*       2: XFA$XXdecodeY$IN_pti
*       3: XFA$XXdecodeY$XX1pti$NTI_Out
*       4: XFA$XXdecodeY$XX1sti$PTI_Out
*       5: XFA$XXdecodeY$XX1sti$NTI_Out
*       6: XFA$XXdecodeY$XXinti$PTI_Out
*       7: XFA$CTRL_YC
*       8: XFA$XXdecodeY$XXinti$STI_Out
*       9: XFA$XXdecodeY$XXinti$STI_Out
*       10: XFA$CTRL_YA
*       11: XFA$CTRL_YA
*       12: XFA$XXdecodeY$XX1pti$STI_Out
*       13: XFA$XXdecodeY$XX1pti$STI_Out
*       14: XFA$CTRL_XB

* Chip #15 - CD4007 pinout:
*       1: XFA$XXdecodeY$XXinti$PTI_Out
*       2: $G_Vdd
*       3: _Y
*       4: $G_Vss
*       5: XFA$CTRL_YC
*       6: XFA$XXdecodeY$IN_pti
*       7: $G_Vss
*       8: XFA$XXdecodeY$XX1sti$NTI_Out
*       9: NC__204
*       10: NC__205
*       11: NC__206
*       12: NC__207
```

```
*       13: XFA$XXdecodeY$XX1sti$PTI_Out
*       14: $G_Vdd
```

All these chips are in the upper-left corner of the board. After analyzing the truth table, the PTI and NTI instead have function numbers $\overline{1}11$ and $\overline{11}1$ instead of $11\overline{1}$ and $1\overline{11}$. For the STI output of the PTI and NTI, $\overline{1}01$ (follows the input). For NTI output of the PTI, $\overline{11}1$, and for PTI output of NTI, $\overline{1}11$. Short pin 3 with across, and PTI output of PTI is 011.

On a second board, with just the Y decoder constructed, one of the PTI or NTI circuits fails, but the other works. After extensive debugging, Chirag was unable to fix the problem, and as a result we instead focused on physically building a reduced architecture known as TCA0 (instead of the previously designed and fully simulated TCA2). However, the TCA2 architecture performed well in simulation, as intended as Jeff's deliverable for this project.

B.1.7. Subtraction

Although subtraction can be derived from addition, addition can be derived from subtraction. The 1100 Univac takes advantage of this by what is called a subtractive adder[95]. Here is a subtractor:

Table B.6. Subtraction Truth Table

A	B	C (Borrow)	D (Difference)
$\overline{1}$	$\overline{1}$	0	0
$\overline{1}$	0	0	$\overline{1}$
$\overline{1}$	1	$\overline{1}$	1
0	$\overline{1}$	0	1
0	0	0	0
0	1	0	$\overline{1}$
1	$\overline{1}$	1	$\overline{1}$
1	0	0	1
1	1	0	0

- C=00$\overline{1}$,000,100 Balanced
- D=0$\overline{1}$1,10$\overline{1}$,$\overline{1}$10 Balanced

- C = ⌐ ⌊A, 0, ¬/A
- D = ∩$\overline{A}$ in Grubb notation, not sure in Mouftah-like notation (see Trinary/Logic), $\overline{A}$, A

An additive subtractor can be made from the adder in 3.2.2 and a negation. A subtractive adder can be made by using the subtractor above with a negation.

However, building a subtractor circuit from scratch using Mouftah's gates requires more circuitry than an adder (the logic functions above are more complicated), so we opted to built it from an adder instead.

B.1.8. Multiplication

On sci.math, James Allwright announced a Balanced Ternary System arithmetic package, and noted:

- Addition and multiplication are simple operations, with the addition and multiplication tables not much more complicated than for binary.
- A suprising division algorithm.

B.1.9. Division

LeRoy Eide developed a fast balanced ternary division-by-2 algorithm using "just-in-time subtraction"[102]. The algorithm also works for unbalanced ternary.

James Allwright in his Balanced Ternary System package also developed a division algorithm.

B.2. Unbalanced Arithmetic

In unbalanced arithmetic, the digits {0,1,2} are used as themselves.

B.2.1. Negation: 3's Complement

In base 2, 1's complement is found by performing bitwise inversion. 2's complement is obtained by adding 1 to 1's complement. Trinary is similar. Tritwise inversion gives 2's complement, adding 1 gives 3's complement.

According to a base converter, 42 = 11203. Tritwise inversion of 000011203 = 222211023 = 651810 unsigned, which is the 2's complement. Add one to get unsigned 6519 = -42 signed, known as the 3's complement. Verify this works by performing -42 + 42, or 6519 + 42 = 6561 = 1_0000_00003, truncated equals 0.

B.2.2. Addition

Table B.7. Balanced Trinary Addition Truth Table

A	B	C	S
0	0	0	0
0	1	0	1
0	2	0	2
1	0	0	1
1	1	0	2
1	2	1	0
2	0	0	2
2	1	1	0
2	2	1	1

- A,B = inputs

- C,S = carry, sum
- C = 000,001,011 = 0 \A \/A
- S = 012,120,201 = A]A [A

B.2.3. Subtraction

Subtraction is simply negation via 3's complement followed by addition.

B.3. Works Cited

1. D.E. Knuth, The Art of Computer Programming - Volume 2: Seminumerical Algorithms, pp. 207-208. Addison-Wesley, 3rd ed., 1998. ISBN 0-201-89684-2. Available: http://jeff.tk/wiki/Image:Knuth-TaoCPVol2-pg207%2C8.pdf
2. The Elements of Computing Systems: Building a Modern Computer from First Principles by Noam Nisan and Shimon Schocken (MIT Press, 2005).
3. Connelly, Jeff. Jeff.tk - Trinary/Meetings. Available: http://jeff.tk/wiki/Trinary/Meetings
4. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary/Status. Available: http://jeff.tk/wiki/Trinary/Status
5. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary. Available: http://jeff.tk/wiki/Trinary
6. Chavez, Antonio and Connelly, Jeff. Trinary/Tools - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Tools
7. Chavez, Antonio and Connelly, Jeff. Trinary/CPU Simulation - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/CPU_Simulation
8. Chavez, Antonio. Trinary/Compiler - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Compiler
9. Swanson, William. *Introduction to Binary Numbers*. 2002. Available: http://www.swansontec.com/sbinary.htm
10. davido, Perl Monks. *Ternary operator (there's no Trinary operator)*. Available: http://www.perlmonks.org/?node_id=562203
11. Wall, Larry. perl.com: Apocalypse 3. Available: http://www.perl.com/pub/a/2001/10/02/apocalypse3.html?page=6
12. Wall, Larry. et. al. *Programming Perl*, 3rd. Edition. ISBN: 978-0596000271 Section 3.16: Conditional Operator.
13. The Antikythera Mechanism Research Project. Available: http://www.antikythera-mechanism.gr/project/overview
14. Lexikon. *Analog Computers*. Available: http://www.computermuseum.li/Testpage/AnalogComputers.htm
15. National Semiconductor, Application Note 31. September 2002. *Op Amp Circuit Collection*. Available: http://www.national.com/an/AN/AN-31.pdf
16. Goldstrasz, Thomas et. al. *Computers During World War Two*. Available: http://waste.informatik.hu-berlin.de/Diplom/WW2/default_e.html
17. Bains, Sunny. *Analog computer trumps Turing model*. EE Times. 11/03/1998. Available: http://www.eetimes.com/story/OEG19981103S0017
18. Principia Cybernetica Web: *Digital Computer*. Available: http://pespmc1.vub.ac.be/ASC/DIGITA_COMPU.html
19. Maney, Kevin. USA Today, September 1997. *Debate Stirs Over Origins of Computers*. Available: http://www.scl.ameslab.gov/ABC/Articles/Debate9-97.html
20. Bebop's BYTES Back. *Claude Shannon's master's Thesis*. Available: http://www.maxmon.com/1938ad.htm
21. Hannah, Eric. *United States Patent 7309866: Cosmic ray detectors for integrated circuit chips*. Available: http://tinyurl.com/3ysdmk
22. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://dx.doi.org/10.1511/2001.40.3268 and mirrored at http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf

23. A. Srivastava and K. Venkatapathy, “Design and Implementation of a Low Power Ternary Full Adder,” VLSI Design, vol. 4, no. 1, pp. 75-81, 1996. doi:10.1155/1996/94696. Available: http://jeff.tk/wiki/Image:Design_and_Implementation_of_a_Low_Power_Ternary_Full_Adder.pdf
24. J.T. Butler, Multiple-Valued Logic in VLSI, IEEE Computer Society Press Technology Series, Los Alamitos, California, 1991.
25. A.K. Jain, M.H. Abd-E1-Barr and R.J. Bolton, "A new structure for CMOS realization of MVL functions," International Journal of Electronics, vol. 74, no. 2, pp. 251-263, 1993.
26. S.L. Hurst, "Two decades of multiple valued logic--an invited tutorial," in Proceedings of IEEE International Symposium on Multiple-Valued Logic, p. 164, May 1988.
27. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
28. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
29. A. P. Dhande and V. T. Ingole. Design And Implementation Of 2 Bit Ternary ALU Slice. SETIT 2005, 3rd International Conference: Science of Electronic, Technologies of Information and Telecommunications. March 17-21, 2005, Tunisia. Available: http://jeff.tk/wiki/Image:Dhande%2C_Ingole_-_Design_and_Implementation_of_a_2_Bit_Ternary_ALU_Slice.pdf
30. P.C.Balla & A.Antoniou "low power dissipation MOS ternary logic family" IEEE journal on solid state circuits Vol. Sc-19 no-5, P.739-749, October 1984.
31. D.I.porat "Three valued digital system" Proc.IEE Vol.116, No6, P.947-955, June 1969.
32. K.C.Smith "The prospects of multivalued logic technology & application view " IEEE transaction on computer, Vol.-C -30, P-619-627 September 1981.
33. Chung-Yu-Wu"Design & application of pipelined dynamic CMOS ternary logic & simple ternary differential logic" IEEE journal on solid state circuits Vol.28, No-8, August 1993.
34. CS150. Berkeley EECS. *Bits, Bytes, Nibbles, and Words: Some definitions*. Available: http://inst.eecs.berkeley.edu/~cs150/sp98/lectures/week6_2/tsld002.htm
35. Slashdot. *Ternary Computing Revisited*. Available: http://slashdot.org/comments.pl?sid=23934
36. Sloppy. Slashdot | Ternary Computing Revisited. Monday November 19 2001. *Trits?* Available: http://slashdot.org/comments.pl?sid=23934&cid=2585807
37. Merrill, Roy D. *Ternary Logic in Digital Computers*. January 1965. Avaialble: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
38. Bowles, Gary-alexander. US Patent #5498980 Ternary/binary converter circuit (Publication Date: 03/12/1996). Available: http://www.freepatentsonline.com/5498980.html
39. Setun' W. H. Ware, S. N. Alexander, N. M. Astrahan, H. H. Goode, M. Rubinoff, P. Armer, L. Bers, H.d. Huskey, "Soviet computer technology - 1959," Communications of the ACM, pp. 149-150, 1960.. Available: http://jeff.tk/wiki/Image:Communications_of_the_ACM_-_Soviet_Computer_Technology_-_1959.pdf
40. Faden, David. Reverse Fad Productions: Flip. Available: http://www.revfad.com/flip.html
41. Crispin, M. Panda Programing. 1 April 2005. Network Working Group, Request for Comments: 4042. *UTF-9 and UTF-18 Efficient Transformation Formats of Unicode*. Available: http://www.ietf.org/rfc/rfc4042.txt RFC 4042
42. Aspinwall, Jim. eCoustics. *Hacking CPU Voltage to Speed Up Your PC*. Available: http://forum.ecoustics.com/bbs/messages/34579/147079.html
43. Engelhardt, Mike. *LTspice/SwitcherCAD III User's Manual*. Available: http://ltspice.linear.com/software/scad3.pdf
44. Howell, Louis and Raymond, Eric S. Available: http://jeff.tk/wiki/Trinary/Logic#TriINTERCAL_Manual:_5.5.2.1_UNARY_LOGICAL_OPERATORS
45. Connelly, Jeff. Trinary/Parts - Jeff.tk - First Purchase. Available: http://jeff.tk/wiki/Trinary/Parts#Shopping_List:_First_Purchase
46. All About Circuits. Producing negative supply rails - Urgent - All About Circuits. Available: http://forum.allaboutcircuits.com/showthread.php?t=10415
47. All About Circuits. negative supply - All About Circuits Available:

http://forum.allaboutcircuits.com/showthread.php?t=876
48. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
49. Connelly, Jeff. Trinary Computer Architecture - Older Diagrams. Available: Image:Proposed Architecture 2.png, Image:Proposed architecture 1.png
50. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
51. Lite-On Electronics Inc. Part No. LTL-30EHJ. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-30EHJ.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1057-ND
52. Lumex. T-5mm LED, 6 leaded, multi-colored, 636 nm AlInGoP Red/574 nm, AlInGoP Green BiColor, 470 nm Ultra Super Blue, Water Color Lens Datasheet. Part #SSL-LX5099SIUBSUGB. Available: http://rocky.digikey.com/weblib/Lumex/Web%20Data/SSL-LX5099SIUBSUGB1.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=67-1829-ND
53. Lite-On Electronics Inc. Part No. LTL-293SJW. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-293SJW.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1038-ND
54. Quicktar. Current limiting resistor calculator for LEDs. Available: http://www.quickar.com/noqbestledcalc.htm
55. Eigenratios of Self-Interpreters: The Mark II OISC Self-Interpreter. Available: http://eigenratios.blogspot.com/2006/09/mark-ii-oisc-self-interpreter.html
56. Connelly, Jeff. Public Git Hosting - trinary.git/tree - circuits/. Available: [http://repo.or.cz/w/trinary.git?a=tree;f=circuits
57. Carothers D. Christopher. Evolution of Intel Microprocessors: 1971 to 2007. Available: http://www.cs.rpi.edu/~chrisc/COURSES/CSCI-4250/SPRING-2004/slides/cpu.pdf
58. Connelly, Jeff. Jeff.tk - Trinary/Symbols/Tips. Available: http://jeff.tk/wiki/Trinary/Symbols/Tips
59. Grubb, Steve. Trinary.cc. Available: http://www.trinary.cc/
60. H. T. Mouftah May 1976 Proceedings of the sixth international symposium on Multiple-valued logic. *A study on the implementation of three-valued logic*. Available: http://jeff.tk/wiki/Image:P123-mouftah_Study_on_the_Implementation_of_Three-valued_Logic.pdf
61. Mouftah, H. T. A study on the implementation of three-valued logic H. T. Mouftah May 1976 Proceedings of the sixth international symposium on Multiple-valued logic. Available: http://jeff.tk/wiki/Image:P123-mouftah_Study_on_the_Implementation_of_Three-valued_Logic.pdf
62. D.I. Porat, "Three-valued digital systems", Proc, lEE, Vol. 116, No. 6, June 1969, pp. 947-954.
63. M. Yoeli and G. Rosenfeld, "Logical design of ternary switching circuits", IEEE Trans. Elect. Comp., Vol. EC-14, February 1965, pp. 19-29.
64. M. Bitran and M.J.O. Strutt, "Minimization of ternary logic and complete set of integrable circuits", Electron. and Commun., AE0, Band 25,No. 8, 1971, pp. 387-392.
65. R.S. Nutter and R.E. Swartwout, "A ternary logic minimization technique", Conference Record of the 1971 Symposium on the Theory and Applications of Multiple-valued Logic Design, May 1971, pp. i12~125.
66. Connelly, Jeff. Trinary/Unary Quick Reference - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Unary_Quick_Reference
67. Hyde, Randall. *Art of Assembly Language*. Available: http://webster.cs.ucr.edu/AoA/DOS/ch01/CH01-2.html
68. Howell, Louis and Raymond, Eric S. *The C-Intercal Supplemental Reference Manual*. 1992-01-18. Available: http://webster.cs.ucr.edu/AoA/DOS/ch01/CH01-2.html
69. Connelly, Jeff. *Unary Gates Inside Binary Gates*. 2001-11-17. Available: http://jeff.tk/bingates/
70. Grubb, Steve. Trinary.cc. Binary Operations. 2001. Available: http://www.trinary.cc/Tutorial/Algebra/Binary.htm
71. I. Halpern and M. Yoeli, "Ternary arithmetic unit", Proc. lEE, Vol. 115, No. i0, October 1968, pp. 1585-1588. Table II : Multiple input ternary operators
72. H.T. Mouftah and I.B. Jordan, "Integrated circuits for ternary logic", Proceedings of the 1974 International Symposium on Multiple-valued Logic, May 1974, pp. 285-302.

73. E.L. Post, "Introduction to a general theory of elementary propositions", Amer. J. Math., Vol. 43, 1921, pp. I~3-185.
74. J.B. Rosser and A.R. Turquette, "Many-valued logics", North-Holland Publishing Co., Amsterdam, 1952.
75. M. Yoeli and G. Rosenfeld, "Logical design of ternary switching circuits", IEEE Trans. Elect. Comp., Vol. EC-14, February 1965, pp. 19-29.
76. R. Vacca, "A three-valued system of logic and its applications to base three digital circuits", Proc. Intern. Conf. Inform. Processing, (UNESCO), June 1959, pp. 407-414.
77. H. Mine, T. Hasegawa, M. Ikeda and T. Shintani, "A construction of ternary logic circuits", Electron. Commun. in Japan, Vol. 51-C, No. 12, pp. 133-140.
78. Sobie, Rick. Troolean operators, Available: http://sci.tech-archive.net/Archive/sci.physics/2006-03/msg00869.html
79. Nynaeve. Blog Archive - The troolean strikes back. Available: http://www.nynaeve.net/?p=87
80. CSE 460 - Spring 2006, Boolean Algebra Definitions, Theorems, and Postulates, Available: http://www.arl.wustl.edu/~lockwood/class/cse460/ba.pdf]
81. Connelly, Jeff. Extensions:Trinary MediaWiki Extension. Available: http://jeff.tk/wiki/Extensions:Trinary
82. Allright, James. Balanced Ternary Web Page. Available: http://web.archive.org/web/20050211091401/http://perun.hscs.wmin.ac.uk/~jra/ternary/
83. W. Ahrens, *Mathematische Unterhaltungen und Spiele* **1** (Leipzig: Teubner, 1910), Section 3.4; H. Hermelink, *Janus* **65** (1978), 105-117
84. *Philos. Trans.* **34** (1726) 161-173
85. *The Philosophy of Arithmetic* (Edinburgh: 1817); see pages 33-34, 54, 64-65, 117, 150
86. *Computes Rendus Acad. Sci. Paris* **11** (1840), 789-798
87. J. Bharati, *Vedic Mathematics* (Delhi: Motilal Banarsidass, 1965)
88. *Mathematical Education* **5**, 3 (1989), 129-133
89. *Computes Rendus Acad. Sci. Paris* **11** (1840), 903-905
90. *American Mathematical Monthly* **57** (1950), 90-93
91. *High-speed Computing Devices*, Engineering Research Associates (McGraw-Hill, 1950), 287-289.
92. *Communications of the Association for Computing Machinery* **3** (1960), 149-150
93. Bhattacharjee, Abhijit. *A polar place value number system for computers and life in general*. Available: http://abhijit.info/tristate/tristate.html
94. H.T. Mouftah, K.C. Smith and Z.G. Vranesic Department of Electrical Engineering University of Toronto Toronto, Ontario, Canada. *Ternary Logic In a Positional Control System*. Available: http://jeff.tk/wiki/Image:P135-mouftah_Ternary_Logic_in_a_positional_control_system.pdf
95. Walker, John. Forumilab, August 19, 1996. Minus Zero. Available: http://www.fourmilab.ch/documents/univac/minuszero.html
96. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
97. Ternary computers: part I: motivation for ternary computers, International Symposium on Microarchitecture archive, Conference record of the 5th annual workshop on Microprogramming table of contents, Urbana, Illinois, 1972
98. Merrill, Roy D. *Ternary Logic in Digital Computers*. January 1965. Available: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
99. Connelly, Jeff. Full Adder Timing Diagram - Internal Signals I. Available: http://jeff.tk/wiki/Image:Full_Adder_Timing_Diagram_-_Internal_Signals_I.png
100. Connelly, Jeff. Full Adder Timing Diagram - Internal Signals II. Available: http://jeff.tk/wiki/Image:Full_Adder_Timing_Diagram_-_Internal_Signals_II.png
101. Connelly, Jeff. Full Adder Y Decoder Signals. Available: Full Adder Y Decoder Signals
102. Halleck, John (via email) and Eide, Leroy. Fast BT division-by-2 using "Just-in-Time Subtraction". Available: http://www.dyalog.dk/dfnsdws/n_JitSub.htm

Appendix C: Implementation Methods

The most labor-intensive aspect of building a trinary computer system is the actual hardware implementation.

C.1. Existing Computers

This section summarizes the known trinary computers—real, simulated, or imaginary—in existence.

C.1.1. TERNAC

TERNAC is a software-emulated machine developed at State University of New York by Gideon Frieder in 1973. This system was Mentioned in *Third Base*[103]. Freider published an extensive two-part article detailing the motivation of ternary computers[104] as well as emulation techniques[105].

C.1.2. Trinary

Trinary is the computer rumored to be used by the Tholians in Star Trek. Can anyone verify this?

C.1.3. Dytrax 1000

Dytrax 1000 is an imaginary computer in the science fiction series *The Fall of Binary Symbolism*. This chapter has a mention of an imaginary trinary computer, the Dytrax 1000. However, the web site has disappeared off the face of the Internet and no further information can be found.

C.1.4. TRIPS Processor

COMP203 (1997 Mid-trimester Test) at Victoria University of Wellington mentions the imaginary *TRIPS* processor, which supports ternary arithmetic in 3's complement using 4-trit fixed-width operands.

C.1.5. Setun (Russian: Сетунь)

SETUN[106] is a real, physical trinary computer that built with add and multiply instructions at Moscow State University in the 1950's. However, its trit flip-flap-flops where not genuine, rather two bits wired to have three stable states. The paper notes that "the choice of base-3 was made because it can be shown that in some sense a base of 3 provides the most efficient utilization of equipment", however, a base-4 machine was constructed and 3 of the states were used, since base-3 electronic technology was not available.

Setun had 4000 magnetic cores, 4000 germanium diodes, 100 transistors, 40 vacuum tubes, a 20 kHz clock, and 150 mW transistor dissipation. With 81 words (18 trits/word) of storage, it could execute 27 types of instructions, supporting fixed point arithmetic (the radix point being between the second and third digits from the left), including a normalizing operation, as well as an addition instruction that executed in 180 ms, multiply in 335 ms, and control transfers in 100 ms. The ferrite core memory was 162 x 9-trit words.

Setun is part of the following Soviet computer family from 1951 to 1969:

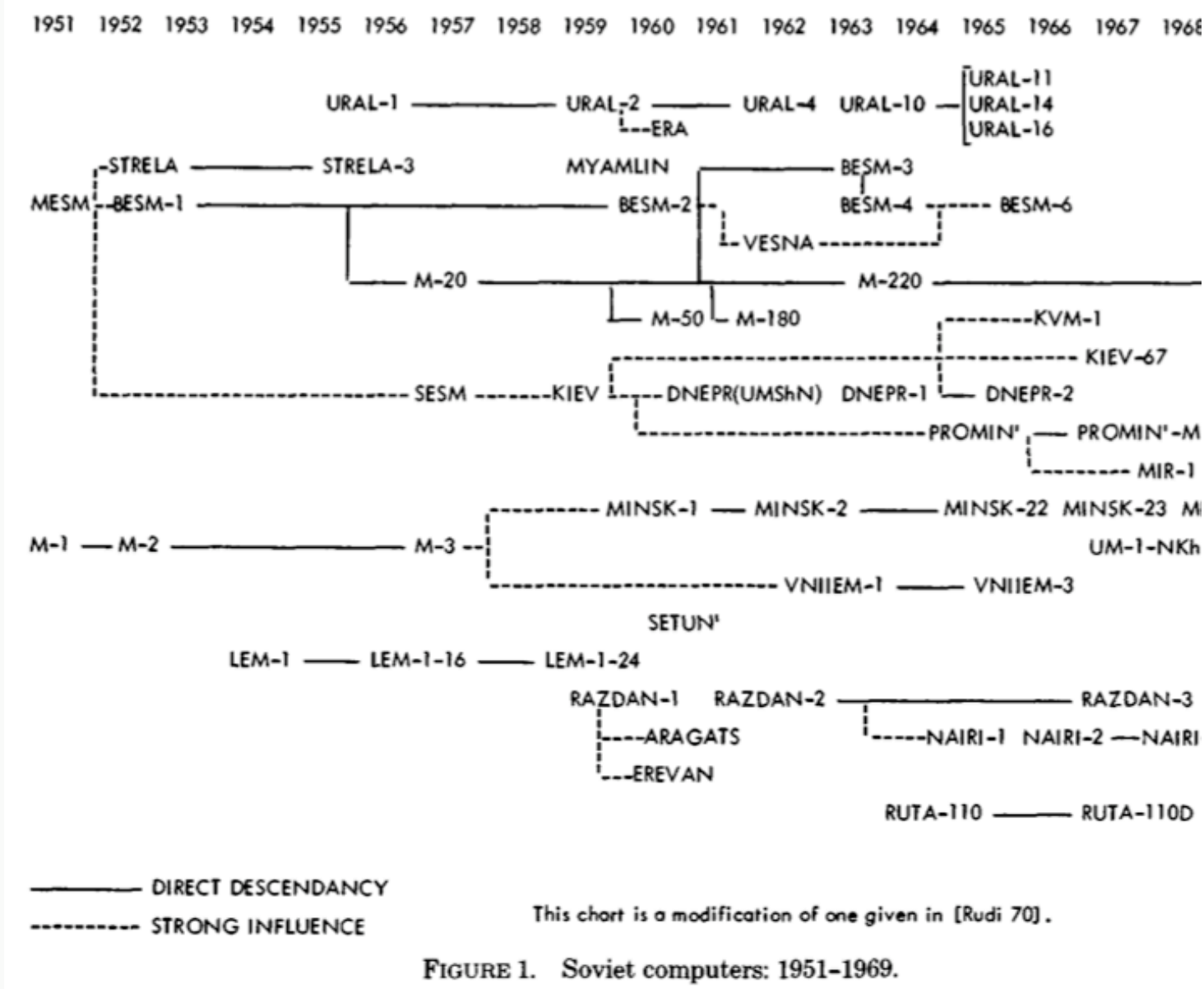


FIGURE 1. Soviet computers: 1951–1969.

Figure C.1. Setun genealogy.

C.1.6. Team R2D2's 64-tert SRAM

Team R2D2[107], composed of Daniel Chillet, Ekue Kinvi-Boh, and Olivier Sentieys "described VHDL models of ternary basic logic and arithmetic cells and of some arithmetic processing units (adder, multiplier, shifter)." in the project "Multiple-Valued Logic architectures and circuits". Excerpt:

> A 64-tert SRAM and a 4-tert adder have been designed and fabricated at UCL. These two circuits represent the very first full-ternary circuit ever fabricated. They have been successfully tested using specifically fabricated test equipments.

C.2. Electrostatic Charge (Capacitors)

Non-polarized capacitors can store a positive or negative charge. Dynamic RAM uses capacitors in a binary fashion—charged or uncharged—to store bits. A trinary dynamic RAM is hypothesized, but has not been

built.

Note that *non-polarized* capacitors must be used, rather than polarized capacitors such as electrolytics. If a negative input voltage is applied to an electrolytic capacitor, it will short out:

> Unlike most capacitors, electrolytic capacitors have a voltage polarity requirement. The correct polarity is indicated on the packaging by a stripe with minus signs and possibly arrowheads, denoting the adjacent terminal that should be more negative than the other. This is necessary because a reverse-bias voltage will destroy the center layer of dielectric material via electrochemical reduction (see Redox reactions). *Without the dielectric material the capacitor will short circuit*, and if the short circuit current is excessive, then the electrolyte will heat up and either leak or cause the capacitor to explode. Modern capacitors have a safety valve, typically either a scored section of the can, or a specially designed end seal to vent the hot gas/liquid, but ruptures can still be dramatic. *Electrolytics can withstand a reverse bias for a short period of time, but they will conduct significant current and not act as a very good capacitor*. Most will survive with no reverse DC bias or with only AC voltage, but circuits should be designed so that there is not a constant reverse bias for any significant amount of time. A constant forward bias is preferable, and will increase the life of the capacitor. Wikipedia (http://en.wikipedia.org/wiki/Electrolytic_capacitors)

C.3. Magnetism

Magnetism naturally has North, South, and unmagnetised states. Materials respond differently[108] depending on if they're diamagnetic, paramagnetic, or ferromagnetic. Diamagnetism is a phenomena all materials inherently experience, but it is very weak. Diamagnetic materials repel both North and South magnetic flux. Ferromagnetism[109] occurs when magnetic domains align, forming a temporary magnet. The magnetization is greater than the applied magnetic field. Paramagnetic materials have magnetization proportional to the strength of the magnetic field applied to it.

C.3.1. Electromagnetism

A relay's coil normally is wound around a ferromagnetic material to increase its strength. The contacts themselves however are for the most part paramagnetic. This means the COM (common) contact is attracted to the NO (normally open) contact if there is any magnetic flux radiating from the coil, no matter the direction.

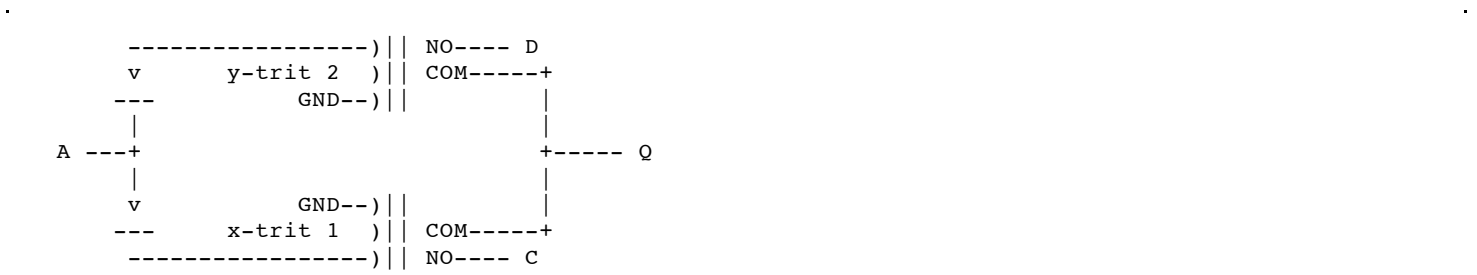
Shown above is a standard relay. In trinary logic, the relay takes advantage of the fact that the coil will be energized when the A input is either a positive or negative voltage (it does not matter which). When the coil is energized, the COM common contact will connect with the NO normally open signal. When the A input is near zero volts, the COM common contact will rest on the NC normally closed signal.

Table C.1.
Electromechanical Relay
Unary Gate

A (Input)	COM (Output)
$\overline{1}$	NO
0	NC
1	NO

The relay can be used to create any *symmetric unary gate*; that is, those with function numbers of the form *xyx*.

To tell negative and positive voltages apart, we can have two relays with diodes to separate the signals in a Dual Diode/Dual Relay configuration:



With dual relays, various unary functions can be implemented including a Simple Ternary Inverter. I've successfully physically constructed the 2D2R configuration, created an inverter and it operated as expected. However, the relay require large voltages and are generally unpleasant to deal with. For a real relay-based computer, see Harry Porter's Relay Computer[110].

C.3.2. Bipolar Relays

In what I call a bipolar relay, the paramagnetic COM contact is replaced by a ferromagnetic temporary magnet.

The COM contact is normally connected to Neutral, but a positive voltage causes it to be connected to North, while negative connects it to South. In this way, the 0, 1, and 2 trits can be detected and substituted with arbitrary values. All 27 unary functions can be created using a single bipolar relay.

```
A Q
0 Neutral
1 North
2 South
```

Bipolar relays can also be used as 1-trit demultiplexers. The input is still the coil, but the trit on COM redirects

to South/Neutral/North depending on the coil. In this way, several unary gates can be created having an input we'll call A, and demultiplexed by an input called B — thus creating a trinary dyadic logic gate.

C.3.3. Core Memory

Magnetic core memory operates by storing magnetic energy in small ferrite rings ("cores"). The most common type, X/Y coincident-current, arranges cores in a grid, and operates by sending half the current required to flip the magnetic field down the row and half down the column. At the intersection, enough current exists to flip the magnetic state. Core is non-volatile but reading is destructive. Reading requires performing a "flip to 0" operation. If the core was originally storing a 1, a small pulse of current will be produced on the sense lines which will be recognized as a 1. If not, it is known that a 0 was stored. Either way, a 0 is stored in the core after being read, so the value has to be rewritten after reading. Writing is performed by doing a "flip to 1" operation if a 1 is to be stored, after doing a read. To store a 0, no flip to 1 operation needs to be performed.

A *trinary core memory* cell could hypothetically operate as follows: instead of 1 and 0 being stored as North and South (or vice versa), 1 could be North, -1 South, and 0 non-magnetized. Writing could increase (from -1 to 0 or 0 to 1) the trit stored, or decrease (from 1 to 0, or 0 to -1) it by pumping current through in the appropriate direction. Reading could be done by writing but observing the sense line for pulse of current, indicating that the value stored was changed.

For example, suppose all memory initially started off non-magnetized (all 0s). To store an arbitrary trit, X:

- Read existing memory value stored at given location into Y (initially will be 0), decreasing its value.
  - If Y = -1, the memory is still -1.
  - If Y = 0, now the memory is -1.
  - If Y = 1, now the memory is 0.
- Check the value to be stored, X:
  - If X = -1:
    - If Y = 1, apply a current to decrease the memory value
    - Otherwise, do nothing because the memory is already -1
  - If X = 0:
    - If Y = -1 or 0, apply current to increase
    - If Y = 1, do nothing
  - If X = 1:
    - If Y = -1 or 0, apply 2 x current to increase from 0 then to 1
    - If Y = 1, apply current to increase

However, ensuring that the magnetic field does not become too strongly -1 or +1 may present serious challenges. The other disadvantages of core memory are the low density and delicacy, which make it uninteresting when building modern computing systems.

C.4. Gravity

The *binary marble adding machine*[111] shows how binary computation is possible using marbles and wood, fueled by gravity. Such a computer could be useful for educational purposes, since the machine's operation and state is clearly visible to the naked eye. Phun[112] is a physics simulation game that could be useful for simulating such a device. However, it is not very practical to do this at a smaller scale (though progress is being made with MEMS—Microelectromechanical systems), so it won't be examined further in this document.

Nonetheless, people have built simple logic gates using dominos [113].

C.5. Rapid Single Flux Quantum

RSFQ logic came up in a thread[114] on Slashdot. Liquor made a post which I'll quote in full, verbatim (including hyperlinks):

> Unfortunately,(RSFQ (Rapid Single Flux Quantum) (http://www.ece.rochester.edu:8080/~sde/research/rsfq/What.is.html) [rochester.edu] circuitry is beyond the scope of SPICE simulations, but this appears to me to be a natural fit to the trinary logic paradigm. Some circuits have already been physically built and tested - and at least one person feels that they lend themselves to tristate logic (http://pavel.physics.sunysb.edu/RSFQ/Research/tri-stable.html) gates [sunysb.edu]. The basic principles are already in the category of proven technology - ever heard of a SQUID sensor? Josephson junctions work equally well for either positive or negative currents - and so do magnetic flux quanta. (But this circuitry has to be the ultimate in low-power computing - you can't get much lower discrete amounts of energy than a single quantum of magnetic flux.)

Josephson junctions, being inherently ternary in nature, allow for building tri-stable RFSQ elements[115]. However because of their relative scarcity, they will not be examined further in this document.

C.6. Cryogenic Storage Devices

In 1965, Merrill[116] describes a cryogenic trinary memory storage cell:

> Cryogenic storage devices have been built to exhibit three stable states using clockwise, counterclockwise and zero persistent loop currents. Such devices have been constructed on the principle of variable loop segment induc- tance using three cryotrons per cell[117] and the constant fluxoid effect using two cryotrons per cell.[118] Both approaches appear to be amenable to batch fabrication techniques and quite feasible when reliable and economical cryostats become available.

Cryostats are used in MRI machines and for biological purposes, to keep helium (typically) liquid while maintaining minimal boil-off. It remains to be seen whether they are economically feasible.

C.7. Light and Other Methods

There is a question whether *light* can be used for trinary computing. An article on a Ternary Optical Computer Architecture[119] was found, although the article was restricted and required a $30 payment to access, so no further details will be provided.

Other methods of computing are possible[120], out of the realm of ordinary experience.

C.8. Electrical Methods

Semiconductor electronics are the most common implementation technique for computers today, by far. A plethora of semiconductor binary logic families have been developed, including TTL, CMOS, and ECL, but these will not be detailed here.

C.8.1. BJT Models

Steve Grubb has an extensive collection of Bipolar Junction Transistor models[121]. However, the circuits are excessively complex:

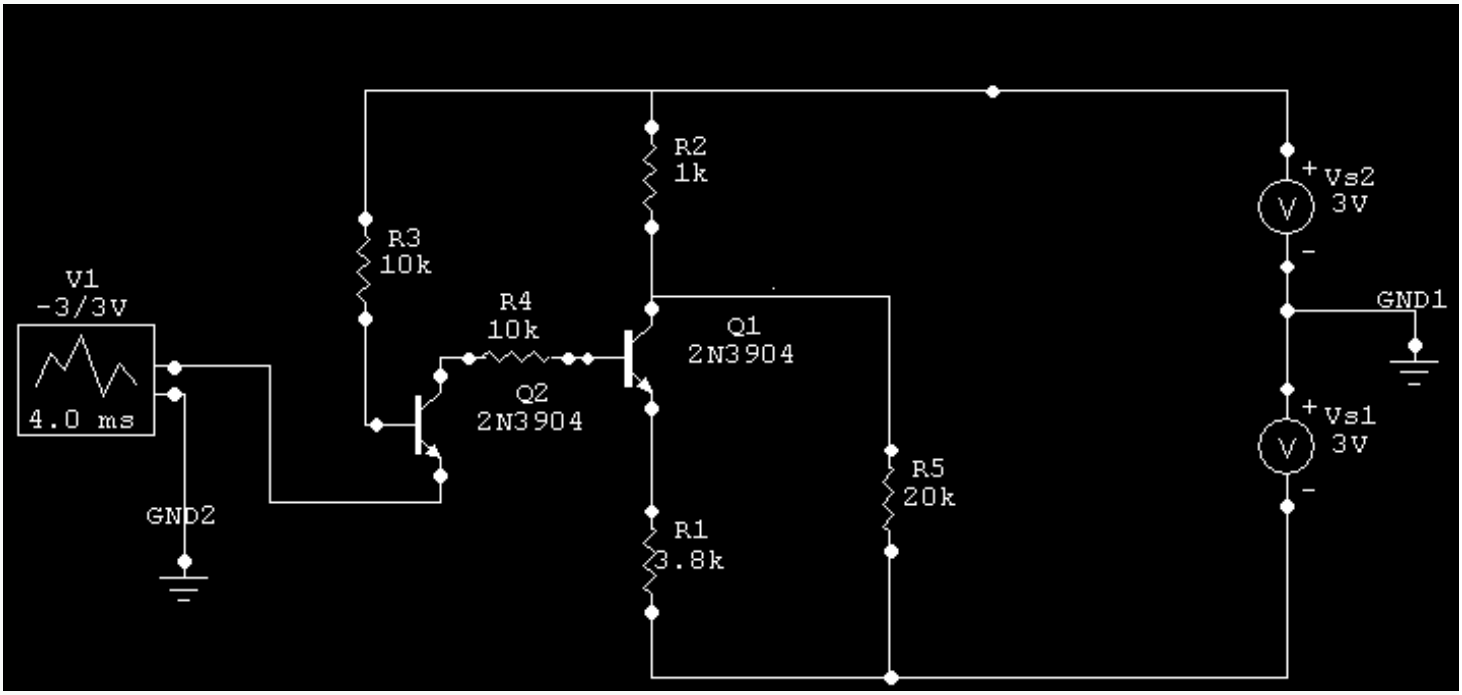


Figure C.2. Grubb's BJT-based trinary inverter circuit.

Due to their complexity, we did not build Grubb's circuits.

C.8.2. CMOS Logic

Mouftah laid out a semiconductor-based trinary logic family, using Complementary Metal Oxide Semiconductors (CMOS). CMOS is a technology where n-channel and p-channel Metal Oxide Field Effect Transistors (MOSFETs) are used in pairs, but Mouftah enhances it using pull-middle resistors to achieve the third state. This is how we implemented the trinary computer, and the details will be discussed in the next section, appendix **D**.

C.9. Works Cited

1. D.E. Knuth, The Art of Computer Programming - Volume 2: Seminumerical Algorithms, pp. 207-208. Addison-Wesley, 3rd ed., 1998. ISBN 0-201-89684-2. Available: http://jeff.tk/wiki/Image:Knuth-TaoCPVol2-pg207%2C8.pdf
2. The Elements of Computing Systems: Building a Modern Computer from First Principles by Noam Nisan and Shimon Schocken (MIT Press, 2005).
3. Connelly, Jeff. Jeff.tk - Trinary/Meetings. Available: http://jeff.tk/wiki/Trinary/Meetings
4. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary/Status. Available: http://jeff.tk/wiki/Trinary/Status
5. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary. Available: http://jeff.tk/wiki/Trinary
6. Chavez, Antonio and Connelly, Jeff. Trinary/Tools - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Tools
7. Chavez, Antonio and Connelly, Jeff. Trinary/CPU Simulation - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/CPU_Simulation
8. Chavez, Antonio. Trinary/Compiler - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Compiler

9. Swanson, William. *Introduction to Binary Numbers*. 2002. Available: http://www.swansontec.com/sbinary.htm
10. davido, Perl Monks. *Ternary operator (there's no Trinary operator)*. Available: http://www.perlmonks.org/?node_id=562203
11. Wall, Larry. perl.com: Apocalypse 3. Available: http://www.perl.com/pub/a/2001/10/02/apocalypse3.html?page=6
12. Wall, Larry. et. al. *Programming Perl*, 3rd. Edition. ISBN: 978-0596000271 Section 3.16: Conditional Operator.
13. The Antikythera Mechanism Research Project. Available: http://www.antikythera-mechanism.gr/project/overview
14. Lexikon. *Analog Computers*. Available: http://www.computermuseum.li/Testpage/AnalogComputers.htm
15. National Semiconductor, Application Note 31. September 2002. *Op Amp Circuit Collection*. Available: http://www.national.com/an/AN/AN-31.pdf
16. Goldstrasz, Thomas et. al. *Computers During World War Two*. Available: http://waste.informatik.hu-berlin.de/Diplom/WW2/default_e.html
17. Bains, Sunny. *Analog computer trumps Turing model*. EE Times. 11/03/1998. Available: http://www.eetimes.com/story/OEG19981103S0017
18. Principia Cybernetica Web: *Digital Computer*. Available: http://pespmc1.vub.ac.be/ASC/DIGITA_COMPU.html
19. Maney, Kevin. USA Today, September 1997. *Debate Stirs Over Origins of Computers*. Available: http://www.scl.ameslab.gov/ABC/Articles/Debate9-97.html
20. Bebop's BYTES Back. *Claude Shannon's master's Thesis*. Available: http://www.maxmon.com/1938ad.htm
21. Hannah, Eric. *United States Patent 7309866: Cosmic ray detectors for integrated circuit chips*. Available: http://tinyurl.com/3ysdmk
22. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://dx.doi.org/10.1511/2001.40.3268 and mirrored at http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
23. A. Srivastava and K. Venkatapathy, “Design and Implementation of a Low Power Ternary Full Adder,” VLSI Design, vol. 4, no. 1, pp. 75-81, 1996. doi:10.1155/1996/94696. Available: http://jeff.tk/wiki/Image:Design_and_Implementation_of_a_Low_Power_Ternary_Full_Adder.pdf
24. J.T. Butler, Multiple-Valued Logic in VLSI, IEEE Computer Society Press Technology Series, Los Alamitos, California, 1991.
25. A.K. Jain, M.H. Abd-E1-Barr and R.J. Bolton, "A new structure for CMOS realization of MVL functions," International Journal of Electronics, vol. 74, no. 2, pp. 251-263, 1993.
26. S.L. Hurst, "Two decades of multiple valued logic--an invited tutorial," in Proceedings of IEEE International Symposium on Multiple-Valued Logic, p. 164, May 1988.
27. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
28. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
29. A. P. Dhande and V. T. Ingole. Design And Implementation Of 2 Bit Ternary ALU Slice. SETIT 2005, 3rd International Conference: Science of Electronic, Technologies of Information and Telecommunications. March 17-21, 2005, Tunisia. Available: http://jeff.tk/wiki/Image:Dhande%2C_Ingole_-_Design_and_Implementation_of_a_2_Bit_Ternary_ALU_Slice.pdf
30. P.C.Balla & A.Antoniou "low power dissipation MOS ternary logic family" IEEE journal on solid state circuits Vol. Sc-19 no-5, P.739-749, October 1984.
31. D.I.porat "Three valued digital system" Proc.IEE Vol.116, No6, P.947-955, June 1969.
32. K.C.Smith "The prospects of multivalued logic technology & application view " IEEE transaction on computer, Vol.-C -30, P-619-627 September 1981.
33. Chung-Yu-Wu"Design & application of pipelined dynamic CMOS ternary logic & simple ternary

differential logic" IEEE journal on solid state circuits Vol.28, No-8, August 1993.
34. CS150. Berkeley EECS. *Bits, Bytes, Nibbles, and Words: Some definitions*. Available: http://inst.eecs.berkeley.edu/~cs150/sp98/lectures/week6_2/tsld002.htm
35. Slashdot. *Ternary Computing Revisited*. Available: http://slashdot.org/comments.pl?sid=23934
36. Sloppy. Slashdot | Ternary Computing Revisited. Monday November 19 2001. *Trits?* Available: http://slashdot.org/comments.pl?sid=23934&cid=2585807
37. Merrill, Roy D. *Ternary Logic in Digital Computers*. January 1965. Avaialble: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
38. Bowles, Gary-alexander. US Patent #5498980 Ternary/binary converter circuit (Publication Date: 03/12/1996). Available: http://www.freepatentsonline.com/5498980.html
39. Setun' W. H. Ware, S. N. Alexander, N. M. Astrahan, H. H. Goode, M. Rubinoff, P. Armer, L. Bers, H.d. Huskey, "Soviet computer technology - 1959," Communications of the ACM, pp. 149-150, 1960.. Available: http://jeff.tk/wiki/Image:Communications_of_the_ACM_-_Soviet_Computer_Technology_-_1959.pdf
40. Faden, David. Reverse Fad Productions: Flip. Available: http://www.revfad.com/flip.html
41. Crispin, M. Panda Programing. 1 April 2005. Network Working Group, Request for Comments: 4042. *UTF-9 and UTF-18 Efficient Transformation Formats of Unicode*. Available: http://www.ietf.org/rfc/rfc4042.txt RFC 4042
42. Aspinwall, Jim. eCoustics. *Hacking CPU Voltage to Speed Up Your PC*. Available: http://forum.ecoustics.com/bbs/messages/34579/147079.html
43. Engelhardt, Mike. *LTspice/SwitcherCAD III User's Manual*. Available: http://ltspice.linear.com/software/scad3.pdf
44. Howell, Louis and Raymond, Eric S. Available: http://jeff.tk/wiki/Trinary/Logic#TriINTERCAL_Manual:_5.5.2.1_UNARY_LOGICAL_OPERATORS
45. Connelly, Jeff. Trinary/Parts - Jeff.tk - First Purchase. Available: http://jeff.tk/wiki/Trinary/Parts#Shopping_List:_First_Purchase
46. All About Circuits. Producing negative supply rails - Urgent - All About Circuits. Available: http://forum.allaboutcircuits.com/showthread.php?t=10415
47. All About Circuits. negative supply - All About Circuits Available: http://forum.allaboutcircuits.com/showthread.php?t=876
48. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
49. Connelly, Jeff. Trinary Computer Architecture - Older Diagrams. Available: Image:Proposed Architecture 2.png, Image:Proposed architecture 1.png
50. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
51. Lite-On Electronics Inc. Part No. LTL-30EHJ. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-30EHJ.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1057-ND
52. Lumex. T-5mm LED, 6 leaded, multi-colored, 636 nm AlInGoP Red/574 nm, AlInGoP Green BiColor, 470 nm Ultra Super Blue, Water Color Lens Datasheet. Part #SSL-LX5099SIUBSUGB. Available: http://rocky.digikey.com/weblib/Lumex/Web%20Data/SSL-LX5099SIUBSUGB1.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=67-1829-ND
53. Lite-On Electronics Inc. Part No. LTL-293SJW. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-293SJW.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1038-ND
54. Quicktar. Current limiting resistor calculator for LEDs. Available: http://www.quickar.com/noqbestledcalc.htm
55. Eigenratios of Self-Interpreters: The Mark II OISC Self-Interpreter. Available: http://eigenratios.blogspot.com/2006/09/mark-ii-oisc-self-interpreter.html
56. Connelly, Jeff. Public Git Hosting - trinary.git/tree - circuits/. Available: [http://repo.or.cz/w/trinary.git?a=tree;f=circuits
57. Carothers D. Christopher. Evolution of Intel Microprocessors: 1971 to 2007. Available: http://www.cs.rpi.edu/~chrisc/COURSES/CSCI-4250/SPRING-2004/slides/cpu.pdf

58. Connelly, Jeff. Jeff.tk - Trinary/Symbols/Tips. Available: http://jeff.tk/wiki/Trinary/Symbols/Tips
59. Grubb, Steve. Trinary.cc. Available: http://www.trinary.cc/
60. H. T. Mouftah May 1976 Proceedings of the sixth international symposium on Multiple-valued logic. *A study on the implementation of three-valued logic*. Available: http://jeff.tk/wiki/Image:P123-mouftah_Study_on_the_Implementation_of_Three-valued_Logic.pdf
61. Mouftah, H. T. A study on the implementation of three-valued logic H. T. Mouftah May 1976 Proceedings of the sixth international symposium on Multiple-valued logic. Available: http://jeff.tk/wiki/Image:P123-mouftah_Study_on_the_Implementation_of_Three-valued_Logic.pdf
62. D.I. Porat, "Three-valued digital systems", Proc, lEE, Vol. 116, No. 6, June 1969, pp. 947-954.
63. M. Yoeli and G. Rosenfeld, "Logical design of ternary switching circuits", IEEE Trans. Elect. Comp., Vol. EC-14, February 1965, pp. 19-29.
64. M. Bitran and M.J.O. Strutt, "Minimization of ternary logic and complete set of integrable circuits", Electron. and Commun., AE0, Band 25,No. 8, 1971, pp. 387-392.
65. R.S. Nutter and R.E. Swartwout, "A ternary logic minimization technique", Conference Record of the 1971 Symposium on the Theory and Applications of Multiple-valued Logic Design, May 1971, pp. i12~125.
66. Connelly, Jeff. Trinary/Unary Quick Reference - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Unary_Quick_Reference
67. Hyde, Randall. *Art of Assembly Language*. Available: http://webster.cs.ucr.edu/AoA/DOS/ch01/CH01-2.html
68. Howell, Louis and Raymond, Eric S. *The C-Intercal Supplemental Reference Manual*. 1992-01-18. Available: http://webster.cs.ucr.edu/AoA/DOS/ch01/CH01-2.html
69. Connelly, Jeff. *Unary Gates Inside Binary Gates*. 2001-11-17. Available: http://jeff.tk/bingates/
70. Grubb, Steve. Trinary.cc. Binary Operations. 2001. Available: http://www.trinary.cc/Tutorial/Algebra/Binary.htm
71. I. Halpern and M. Yoeli, "Ternary arithmetic unit", Proc. lEE, Vol. 115, No. i0, October 1968, pp. 1585-1588. Table II : Multiple input ternary operators
72. H.T. Mouftah and I.B. Jordan, "Integrated circuits for ternary logic", Proceedings of the 1974 International Symposium on Multiple-valued Logic, May 1974, pp. 285-302.
73. E.L. Post, "Introduction to a general theory of elementary propositions", Amer. J. Math., Vol. 43, 1921, pp. I~3-185.
74. J.B. Rosser and A.R. Turquette, "Many-valued logics", North-Holland Publishing Co., Amsterdam, 1952.
75. M. Yoeli and G. Rosenfeld, "Logical design of ternary switching circuits", IEEE Trans. Elect. Comp., Vol. EC-14, February 1965, pp. 19-29.
76. R. Vacca, "A three-valued system of logic and its applications to base three digital circuits", Proc. Intern. Conf. Inform. Processing, (UNESCO), June 1959, pp. 407-414.
77. H. Mine, T. Hasegawa, M. Ikeda and T. Shintani, "A construction of ternary logic circuits", Electron. Commun. in Japan, Vol. 51-C, No. 12, pp. 133-140.
78. Sobie, Rick. Troolean operators, Available: http://sci.tech-archive.net/Archive/sci.physics/2006-03/msg00869.html
79. Nynaeve. Blog Archive - The troolean strikes back. Available: http://www.nynaeve.net/?p=87
80. CSE 460 - Spring 2006, Boolean Algebra Definitions, Theorems, and Postulates, Available: http://www.arl.wustl.edu/~lockwood/class/cse460/ba.pdf]
81. Connelly, Jeff. Extensions:Trinary MediaWiki Extension. Available: http://jeff.tk/wiki/Extensions:Trinary
82. Allright, James. Balanced Ternary Web Page. Available: http://web.archive.org/web/20050211091401/http://perun.hscs.wmin.ac.uk/~jra/ternary/
83. W. Ahrens, *Mathematische Unterhaltungen und Spiele* **1** (Leipzig: Teubner, 1910), Section 3.4; H. Hermelink, *Janus* **65** (1978), 105-117
84. *Philos. Trans.* **34** (1726) 161-173
85. *The Philosophy of Arithmetic* (Edinburgh: 1817); see pages 33-34, 54, 64-65, 117, 150
86. *Computes Rendus Acad. Sci. Paris* **11** (1840), 789-798

87. J. Bharati, *Vedic Mathematics* (Delhi: Motilal Banarsidass, 1965)
88. *Mathematical Education* **5**, 3 (1989), 129-133
89. *Computes Rendus Acad. Sci. Paris* **11** (1840), 903-905
90. *American Mathematical Monthly* **57** (1950), 90-93
91. *High-speed Computing Devices*, Engineering Research Associates (McGraw-Hill, 1950), 287-289.
92. *Communications of the Association for Computing Machinery* **3** (1960), 149-150
93. Bhattacharjee, Abhijit. *A polar place value number system for computers and life in general*. Available: http://abhijit.info/tristate/tristate.html
94. H.T. Mouftah, K.C. Smith and Z.G. Vranesic Department of Electrical Engineering University of Toronto Toronto, Ontario, Canada. *Ternary Logic In a Positional Control System*. Available: http://jeff.tk/wiki/Image:P135-mouftah_Ternary_Logic_in_a_positional_control_system.pdf
95. Walker, John. Forumilab, August 19, 1996. Minus Zero. Available: http://www.fourmilab.ch/documents/univac/minuszero.html
96. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
97. Ternary computers: part I: motivation for ternary computers, International Symposium on Microarchitecture archive, Conference record of the 5th annual workshop on Microprogramming table of contents, Urbana, Illinois, 1972
98. Merrill, Roy D. *Ternary Logic in Digital Computers*. January 1965. Available: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
99. Connelly, Jeff. Full Adder Timing Diagram - Internal Signals I. Available: http://jeff.tk/wiki/Image:Full_Adder_Timing_Diagram_-_Internal_Signals_I.png
100. Connelly, Jeff. Full Adder Timing Diagram - Internal Signals II. Available: http://jeff.tk/wiki/Image:Full_Adder_Timing_Diagram_-_Internal_Signals_II.png
101. Connelly, Jeff. Full Adder Y Decoder Signals. Available: Full Adder Y Decoder Signals
102. Halleck, John (via email) and Eide, Leroy. Fast BT division-by-2 using "Just-in-Time Subtraction". Available: http://www.dyalog.dk/dfnsdws/n_JitSub.htm
103. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
104. Frieder, Gideon. Part 1 - Motivation for Ternary Computers. Available: http://jeff.tk/wiki/Image:P83-frieder_Ternary_Computers_-_Part_1_-_Motivation_for_Ternary_Computers.pdf
105. Frieder, Gideon. Part 2 - Emulation of Ternary Computers. Available: http://jeff.tk/wiki/Image:P86-frieder_-_Ternary_Computers_-_Part_2_-_Emulation_of_Ternary_Computers.pdf
106. Setun' W. H. Ware, S. N. Alexander, N. M. Astrahan, H. H. Goode, M. Rubinoff, P. Armer, L. Bers, H.d. Huskey, "Soviet computer technology - 1959," Communications of the ACM, pp. 149-150, 1960.. Available: http://jeff.tk/wiki/Image:Communications_of_the_ACM_-_Soviet_Computer_Technology_-_1959.pdf
107. Chillet, Daniel et. al. Team: r2d2. Available: http://web.archive.org/web/20060514100747/http://www.inria.fr/rapportsactivite/RA2004/r2d22004/uid51.html
108. Hyperphysics: Magnetic Properties of Solids. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/magpr.html
109. Hyperphysics: Ferromagnetism. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/ferro.html
110. Porter, Harry. Harry Poter's Relay Computer. Last updated 2007-11-15. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/ferro.html
111. Woodgears.ca: Binary marble adding machine. Available: http://woodgears.ca/marbleadd/
112. Phun — 2D Physics Sandbox — Home. Available: http://www.phunland.com/wiki/Home
113. mlittman. Youtube: Logic gates using toys. Available: http://www.youtube.com/watch?v=H-53TVR9EOw
114. Grubb Steve. Slashdot: Ternary Computing Revisited: Re:impractical circuits. 2001-11-19. Available: http://slashdot.org/comments.pl?sid=23934&cid=2586271
115. H. Chan, T. van Duzer, D. Erne. A tri-stable state Josephson device memory cell. Available: http://slashdot.org/comments.pl?sid=23934&cid=2586271
116. Merrill, Roy D. Ternary Logic in Digital Computers. January 1965. Available:

http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
117. A. E. Slade, "A Cryotron Memory Cell," Proc. IRE, Vol. 50, Jan. 62, p. 81.
118. C. F. Kooi, R. D. Merrill and H. H. Nakano, "Superconductive Ternary Information Storage Device," Lockheed Missiles & Space Co., *Research on Automatic* Computer Electronics*, Vol. I, RTD-TDR-63-4173, Oct. 31, 1963, pp. A-36-A-48.*
119. Yi, Jin. Institute of Physics. Ternary Optical Computing Architecture. Available for purchase: http://www.iop.org/EJ/abstract/1402-4896/2005/T118/025/
120. NewScientistTech. Ten weirdest computers, New Scientist, 18:30 11 April 2008 by Duncan Graham-Rowe. Available: http://technology.newscientist.com/article/dn13656-ten-weirdest-computers.html?DCMP=ILC-hmts&nsref=news1_head
121. Grubb, Steve. Trinary.cc: Transistor Models. Available: http://www.trinary.cc/Tutorial/Transistors/Transistors.htm

Appendix D: Circuit Simulation

We tested a plethora of trinary electronic circuits, first using the circuit simulation software LTspice[122] developed by Linear Technologies[123], and then physically building select circuits on breadboard and printed circuit boards.

Note that we only tested electronic circuit implementations of trinary logic, primarily built using Metal Oxide Semiconductor Field Effect Transistors (MOSFETs), rather than Bipolar Junction Transistors (BJTs) or non-electronic implementations. This section also only covers low-level circuits; high-level architectural components will be discussed in the next section. Lastly, the scope of this project only allowed for experimentation with Small Scale Integration (SSI) integrated circuit techniques, rather than Very Large Scale Integration (VLSI) as would be done when designing a custom integrated circuit[124].

D.1. LTspice Parts Library

I developed a comprehensive LTspice parts library of reusable trinary components for use in constructing the trinary computer. A total of 66 parts were designed in LTspice, along with 33 test circuits to verify individual part operation in simulation. The parts require the SwCADIII-jc.zip library[125], extracted into `C:\Program Files\LTC\SwCAD III` on the computer system where LTspice is installed. This archive includes CD4007 transistor models used for the low-level circuits.

All circuits for the trinary computer are available in the *git repository*[126]. Git is a distributed version control system that we used to keep track of all our changes to the circuits and source code.

D.2. Transmission Gate

First we'll consider a model for a transmission gate. A *transmission gate* simply passes an analog signal from the input to the output, if the control signal is active. The CD4016 transmission gate is a bidirectional analog switch, so it can be used to pass ternary signals in either direction.

There are other analog switches[127] to consider:

- The CD4066BC[128] is pin-for-pin compatible with the CD4016 but has much lower on resistance.
- MAX4610-4612[129] offers higher performance, and in a variety of normally open/closed configurations. But in the end, we purchased the CD4016 since it is the same chip Mouftah used in his designs. A SPICE model could not be located, so I designed a schematic to approximate the internal structure of the circuit:

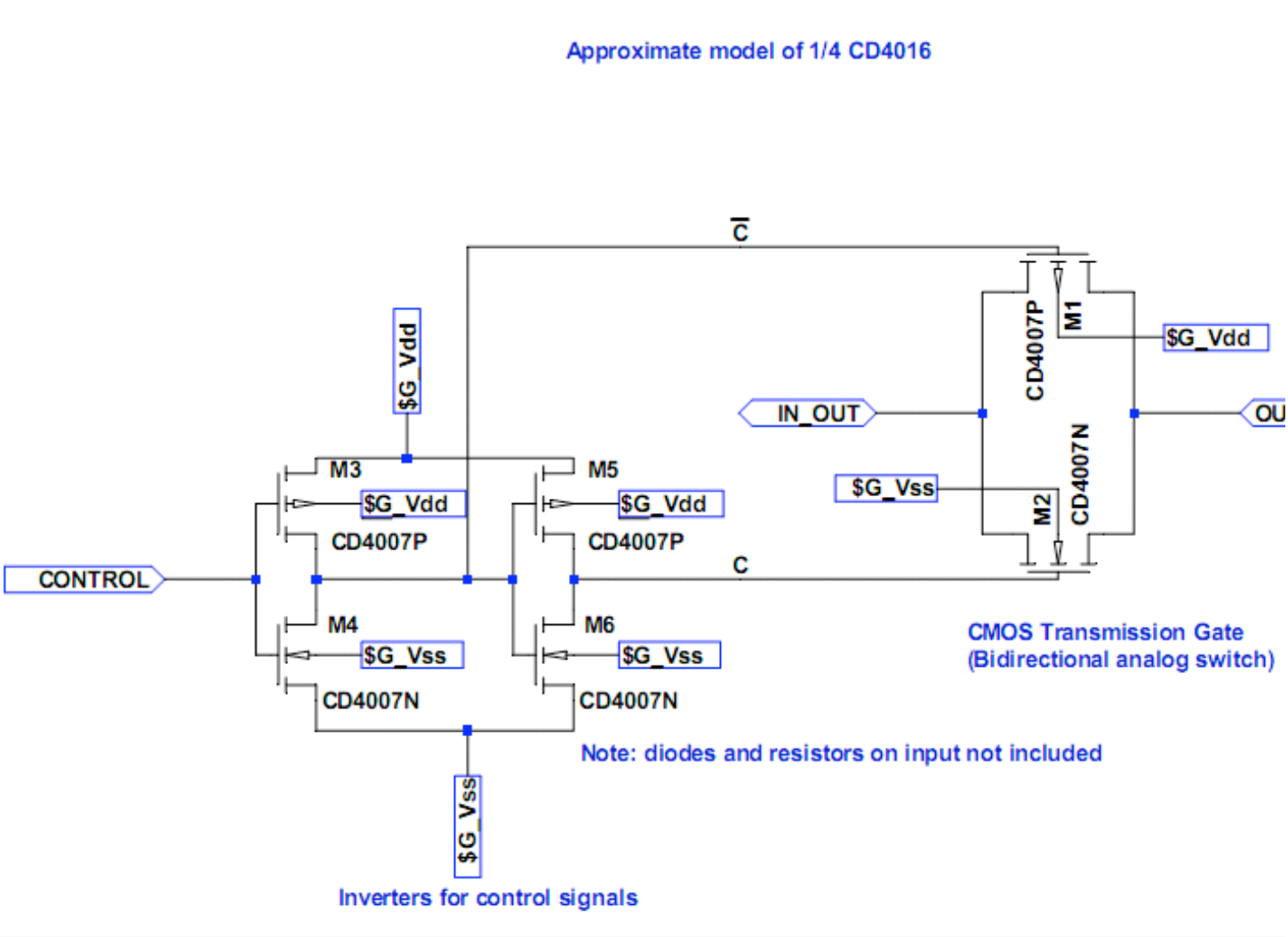


Figure D.1. Schematic for CMOS transmission gate model.

Here is the result of a model using CD4007 transistors:

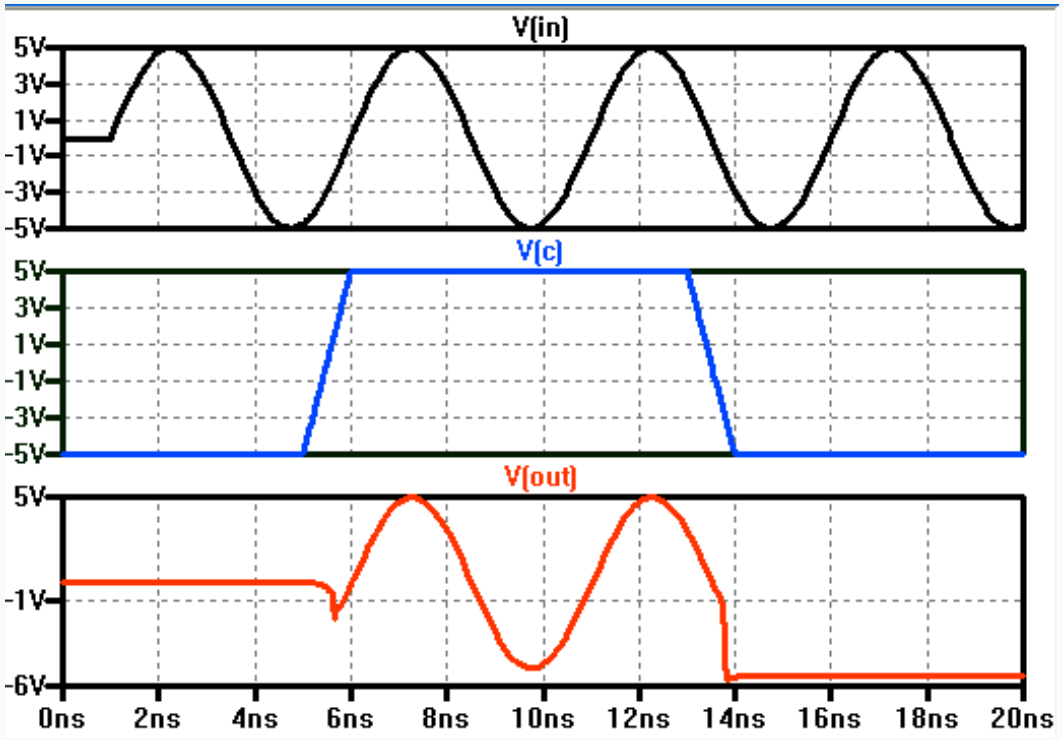


Figure D.2. Timing diagram of CMOS transmission gate with integrated inverters.

The model uses two inverters, like the CD4016 block diagram, and the control signal complement is internally generated.

D.3. Unary Logic Gates

D.3.1. VTC Curve Characteristics

Binary inverters have a Voltage Transfer Characteristic as:

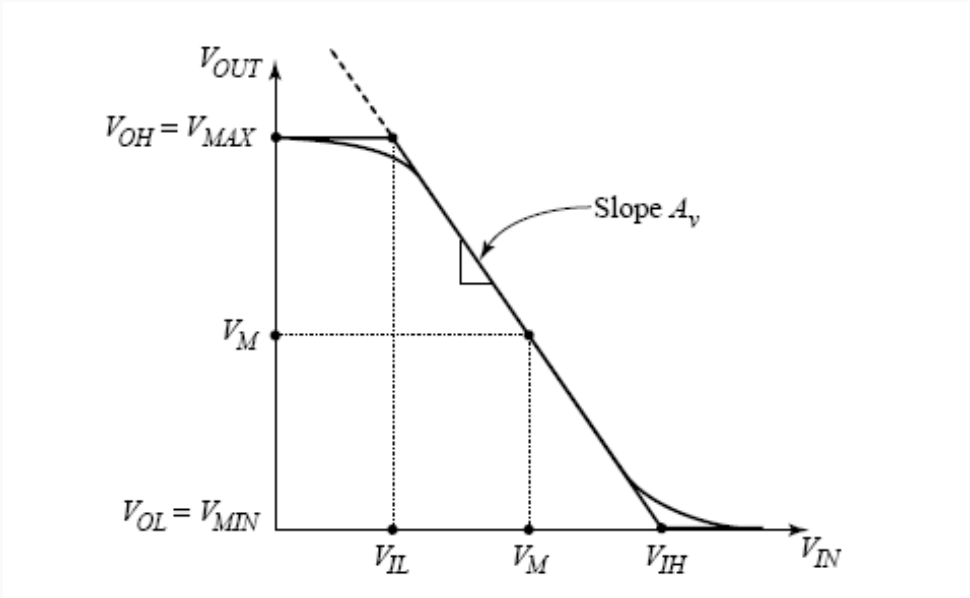


Figure D.3. Voltage Transfer Characteristics Curve
Source: http://inst.eecs.berkeley.edu/~ee105/fa98/lectures_fall_98/093098_lecture16.pdf

For trinary inverters, Chirag Patel designed the following VTC diagram labeled with critical voltages:

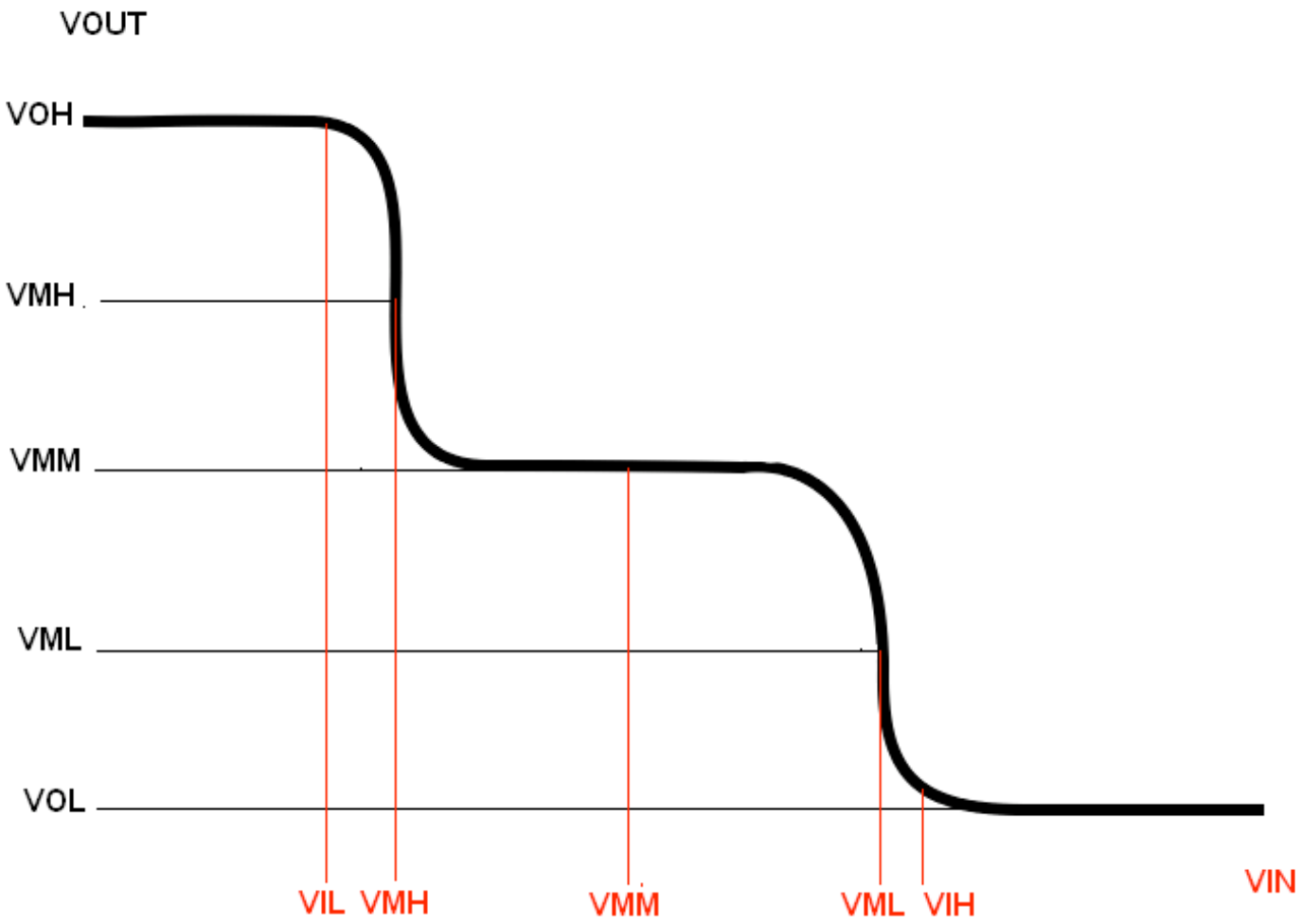


Figure D.4. Trinary Voltage Transfer Characteristic, annotated with critical voltages.

The critical voltages are as follows:

Table D.1. Trinary Voltage Transfer Characteristic

Critical Voltage	Description
V_{OH}	Output high
V_{OM}	Output middle
V_{OL}	Output low
V_{IH}	Input high
V_{IM}	Input middle
V_{IL}	Input low
V_{MH}	Metastable high ($V_{in} = -V_{out}$, $V_{out} < 0$)
V_{MM}	Metastable medium ($V_{in} = V_{out}$)
V_{ML}	Metastable low ($V_{in} = -V_{out}$, $V_{out} > 0$);

The output/input high/low/middle voltages are self-explanatory, but the metastable voltages require some

explanation. In binary, there is only one point on the graph where the input voltage is equal to the output voltage, and this is known as the switching or *metastable voltage*, V_M. In trinary, there are three metastable voltages, where the magnitude of the input voltage is equal to the magnitude of the output voltage, and one or neither of the voltages is negative.

D.3.2. Inverters

The basic inverter circuit is built using an NMOS, PMOS, and two resistors of equal value. Five different inverters can be made with this basic circuit, invented by Mouftah[130]. Please refer to the earlier section on logic in this document for an explanation of the troolean logic behind this circuit.

The basic ternary inverter is as shown:

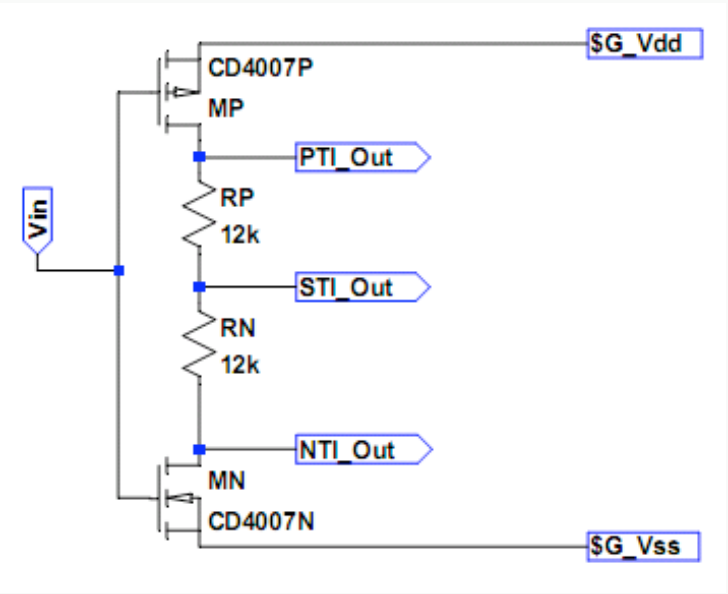


Figure D.5. Ternary inverters: positive, simple, and negative with R = 12 kΩ. A subcircuit with an inverter symbol is based on this circuit. Schematic created in SwCAD.

The resistors are herein termed *pull-middle resistors*, since they pull the output to near zero volts as zero input, analogous to the *pull-up* and *pull-down* resistors in binary circuits. Inverters can be made with pull-middle resistors other than 12 kΩ, although this results in inferior VTCs[131]. The circuit with 12 kΩ pull-middles behaves as follows:

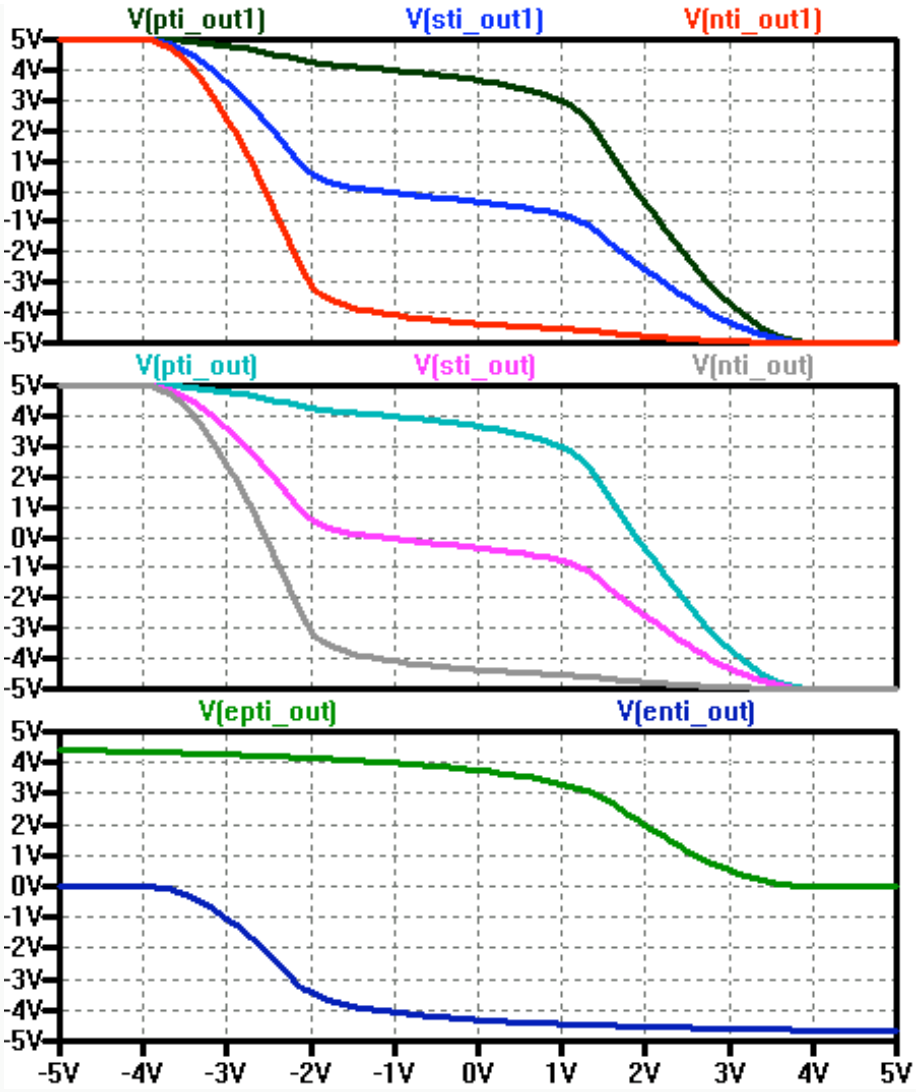
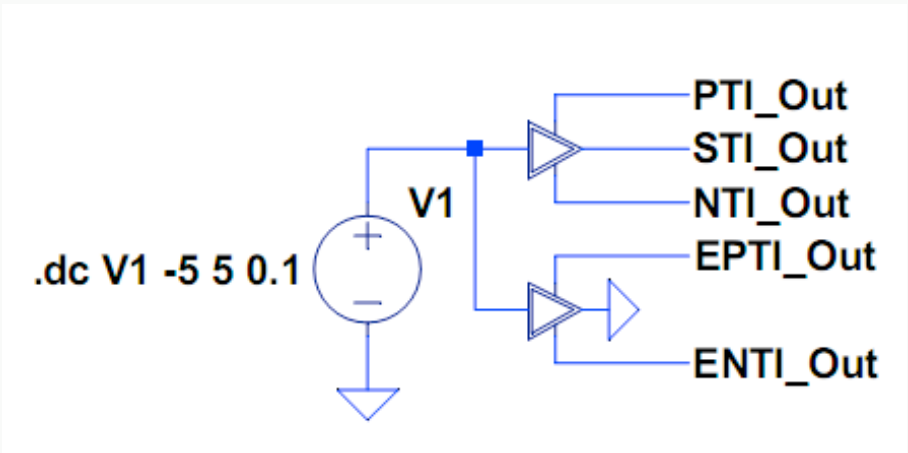


Figure D.6. Voltage Transfer Characteristic of positive, simple, and negative ternary inverters with R = 12 kΩ, simulated with LTspice.

The circuit is tested using this test harness:



Test circuit to demonstrate use of inverters. (Power supply not shown.)

In lab, we built and tested this circuit on breadboard:

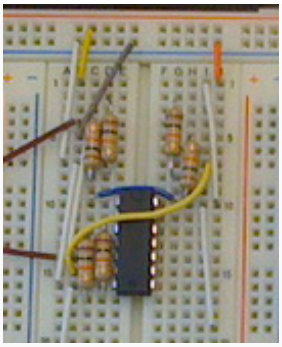


Figure D.7. Basic ternary inverter built on a breadboard.

When tested in lab, each inverter behaved as expected, matching the VTC that was simulated:

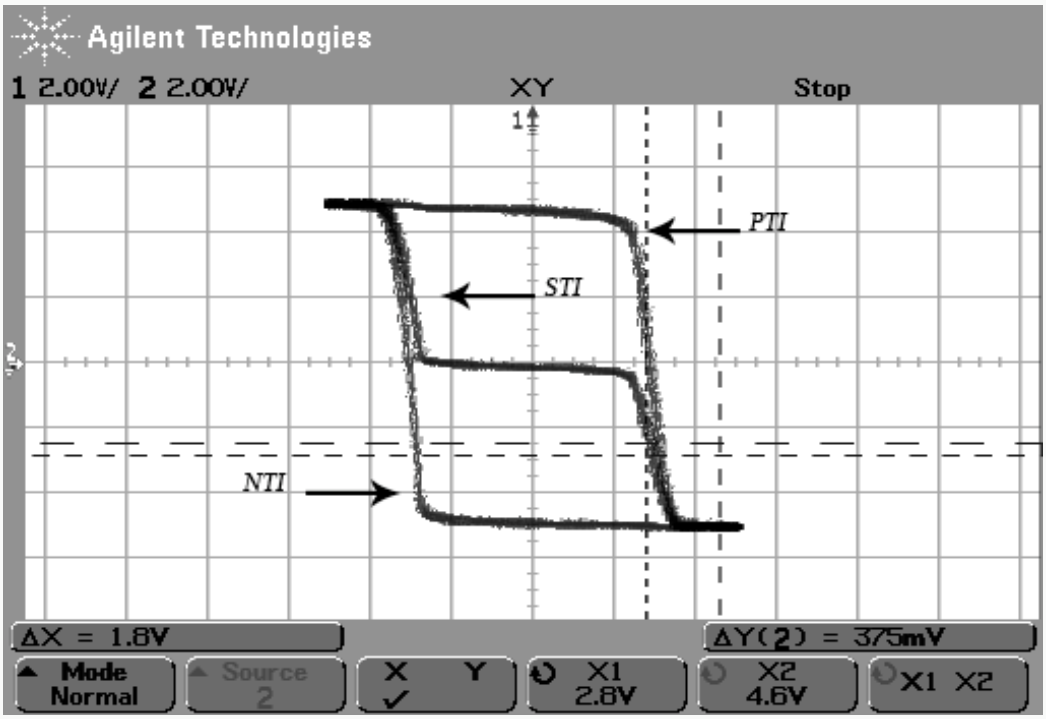


Figure D.8. Voltage Transfer Characteristic for STI, NTI, PTI with R = 12 kΩ

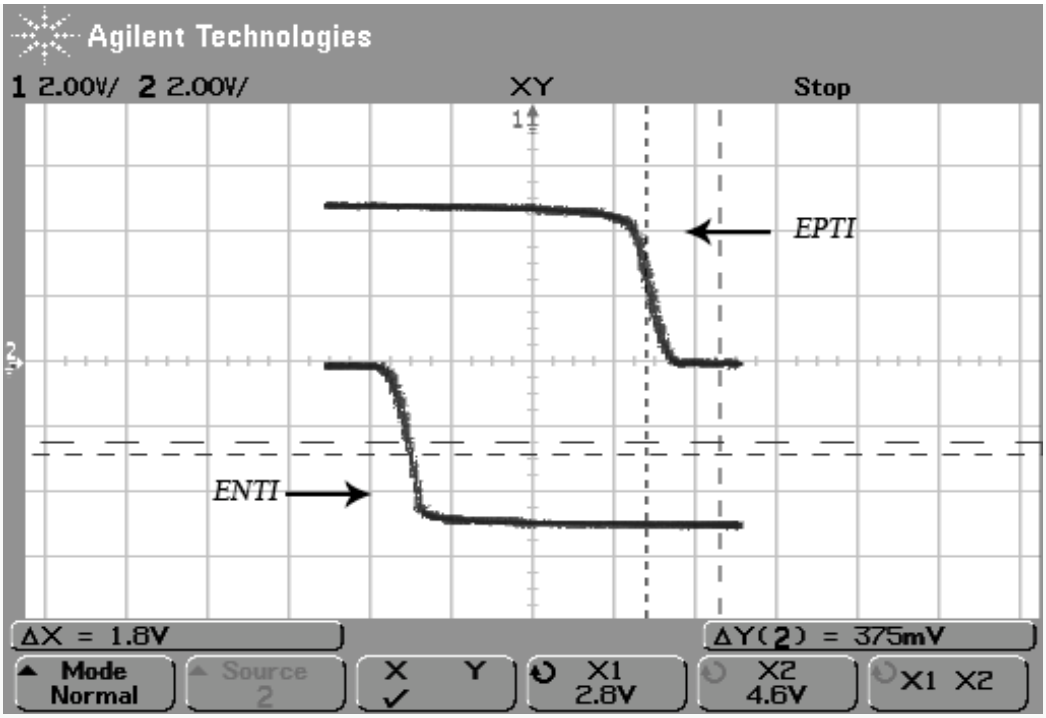


Figure D.9. Voltage Transfer Characteristic for ENTI and EPTI with R = 12 kΩ

D.3.3. Diode Gates

Trinary gates can be constructed using *diodes*. Diodes block current in one direction while allowing it in the other direction, making them ideal devices to use in a system with positive and negative voltages. The following test circuit was used:

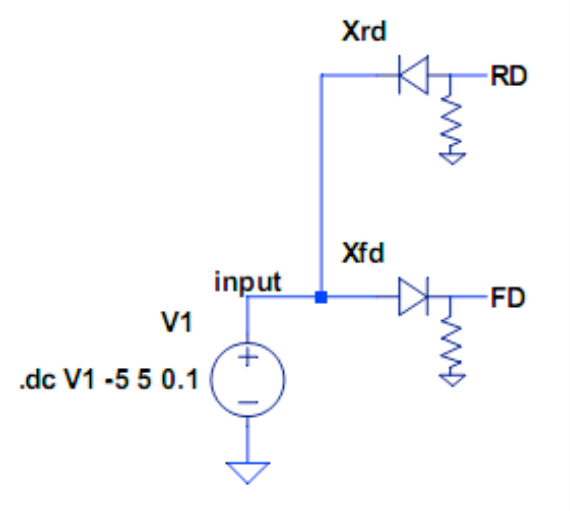


Figure D.10. Forward diode and reverse diode test circuit.

The diode + resistor in the FD (forward diode) gate is one symbol, `fd`, and the diode + resistor in the RD (reverse diode) gate is `rd`. This allows for placing FD and RD gates without having to think about where to add the diode and resistor, but the visual representation in the schematic is still there. The pull-middle resistor is 10 MΩ, and this is what the VTCs look like with SPICE's default diode model:

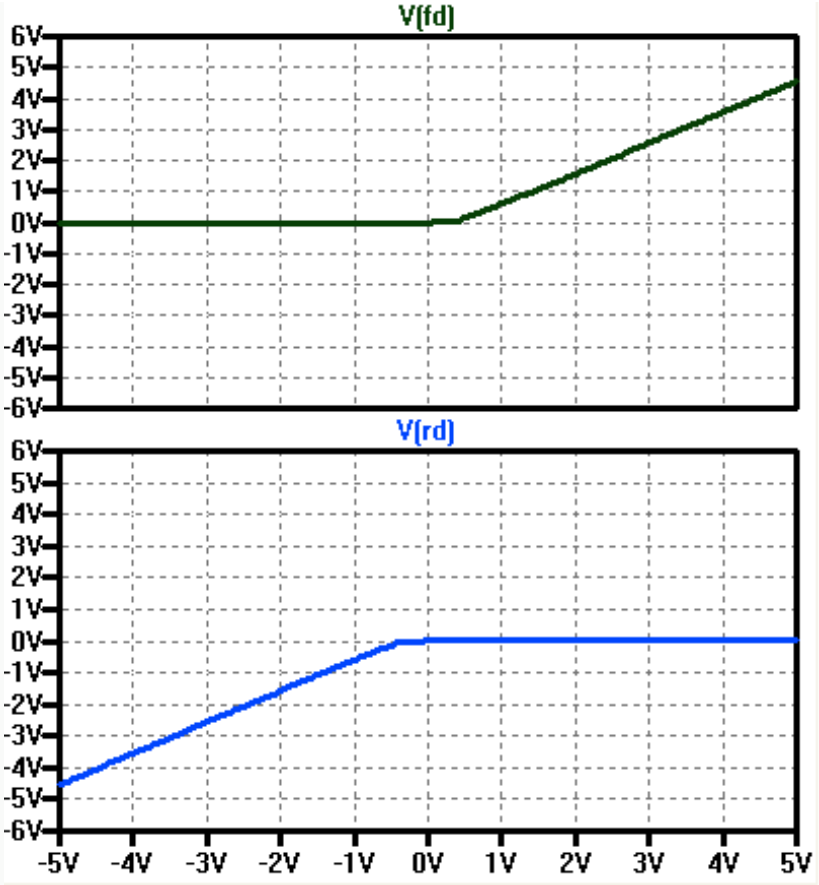


Figure D.11. VTC for RD and FD gates, with default SPICE diode model "D".

Using the 1N4148 model instead results in a very similar VTC:

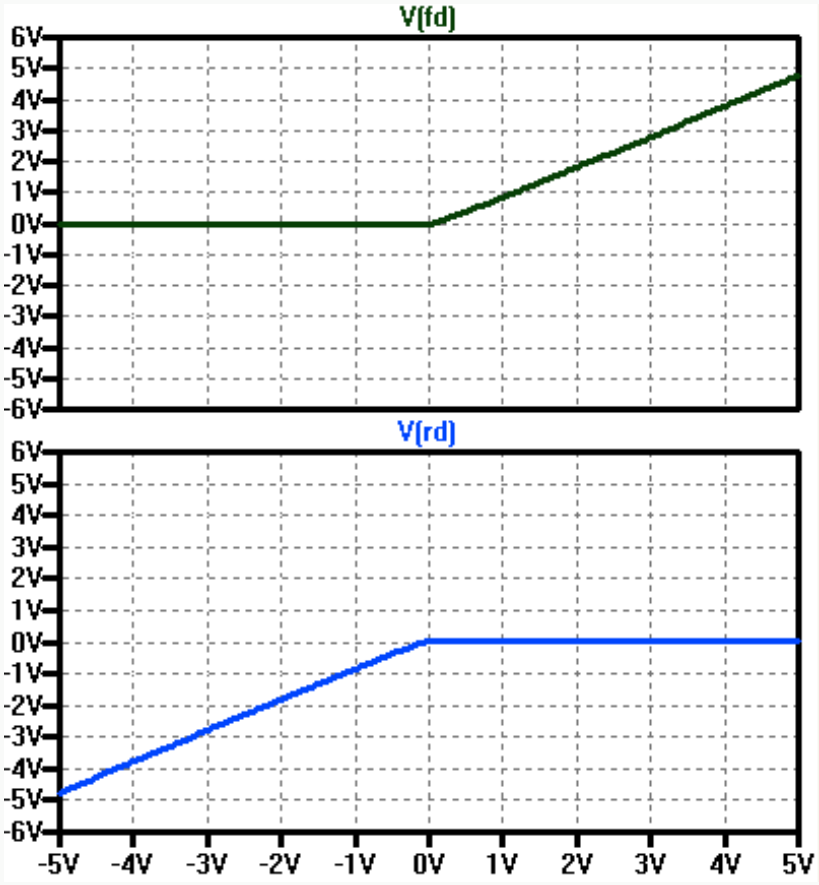


Figure D.12. VTC for RD and FD gates, with 1N4148 diode model.

Using the 1N4448 diode:

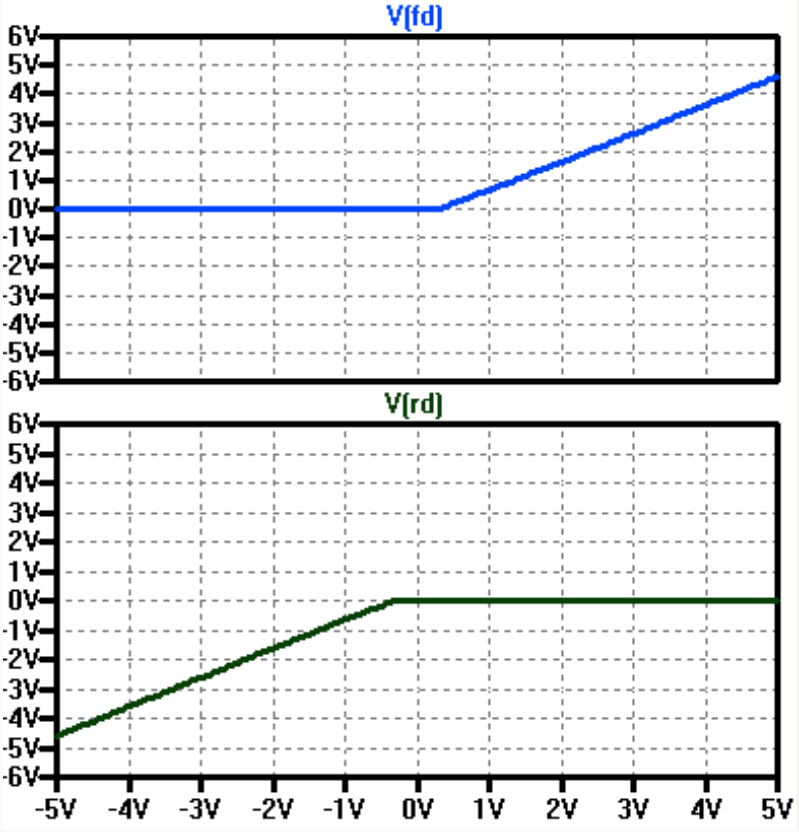


Figure D.13. VTC for RD and FD gates, with 1N4448 diode model.

And using the 1N5817 Schottky diode model does not rectify the signal at all:

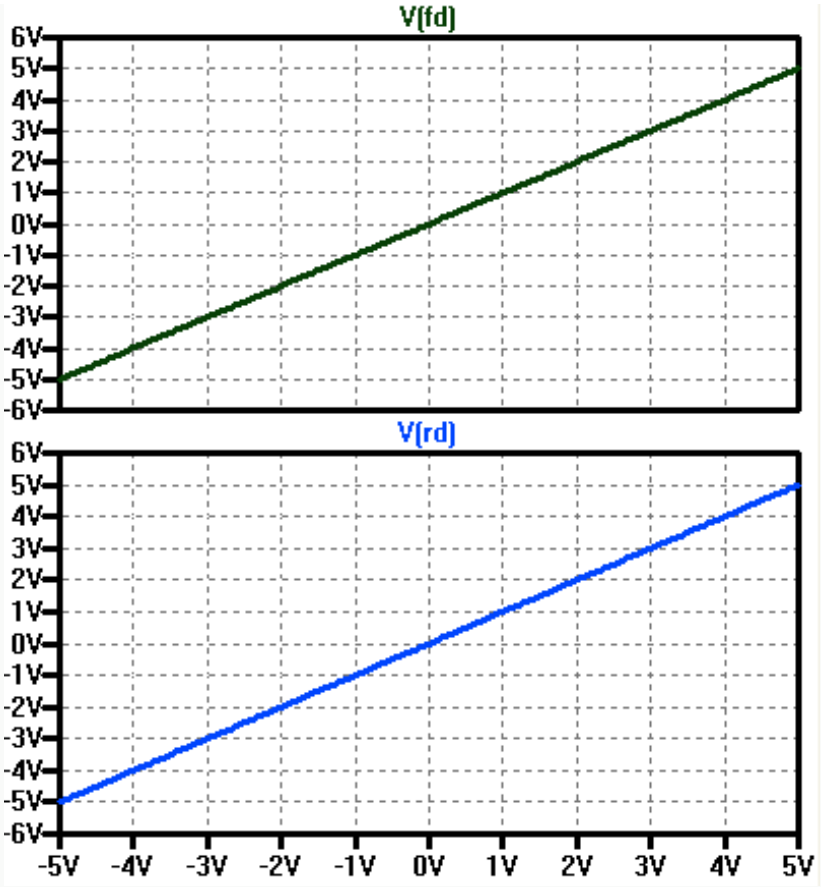


Figure D.14. VTC for RD and FD gates, with 1N5817 Schottky diode model.

In lab, we tested the FD and RD gates built with a 1N4448 High Conductance Fast Diode and 90 kΩ resistor:

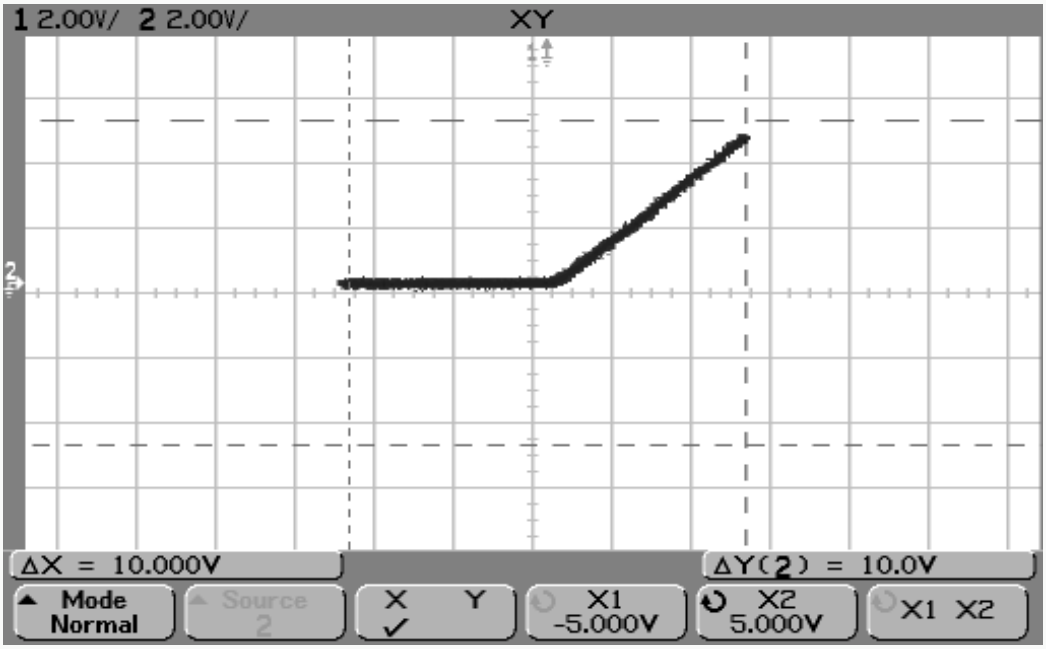


Figure D.15. Forward Diode gate with 1N4448 diode and 90 kohm resistor.

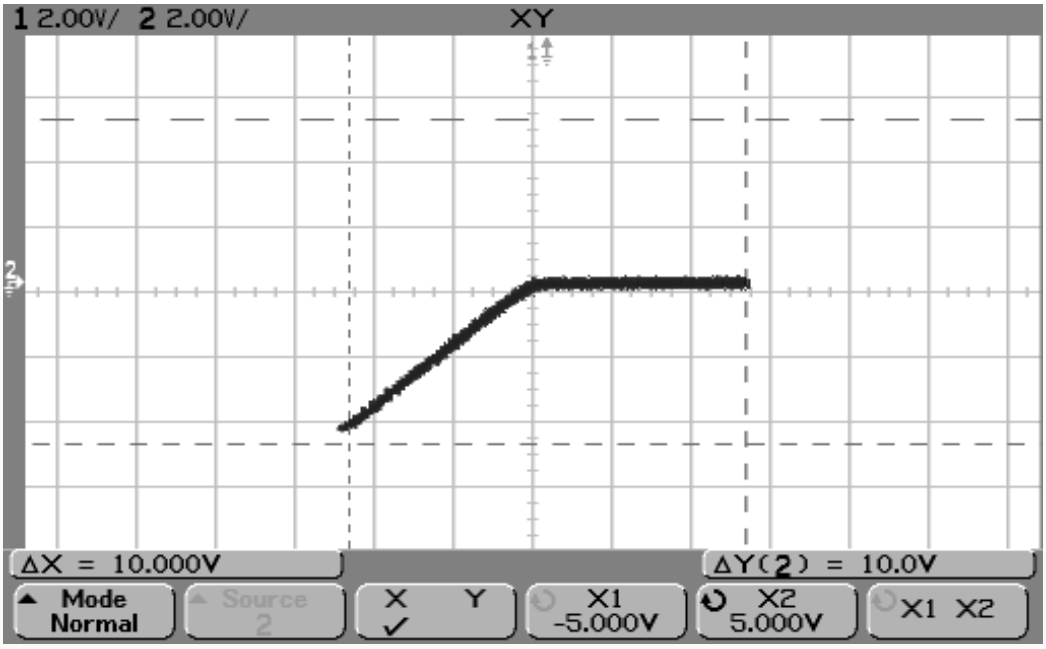


Figure D.16. Reverse Diode gate with 1N4448 diode and 90 kohm resistor.

The results observed in lab match those simulated in lab. However, we did not find the FD and RD gates useful in constructing higher-level trinary architectural components for the computer system, so they will not be discussed further. The fundamental problem is that the diodes, when conducting, have a linear region rather than a sharp change from one stable voltage to another (as with the basic ternary inverters). This characteristic degrades signal quality; additionally, forward and reverse diode gates are not necessary so they will not be used.

D.3.4. Cycling Gates

Cycling and inverse cycling gates were invented by Mouftah[132] As explained in Trinary/Logic#One-to-one_Functions, Grubb calls these "rotate up" and "rotate down", respectively. For clarity, I chose a mix of both terminologies: cycle down and cycle up.

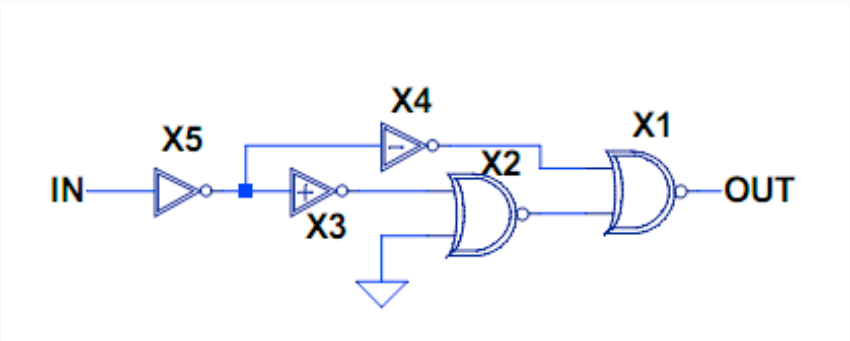


Figure D.17. Cycle up gate, function $01\bar{1}$, from Mouftah[132]

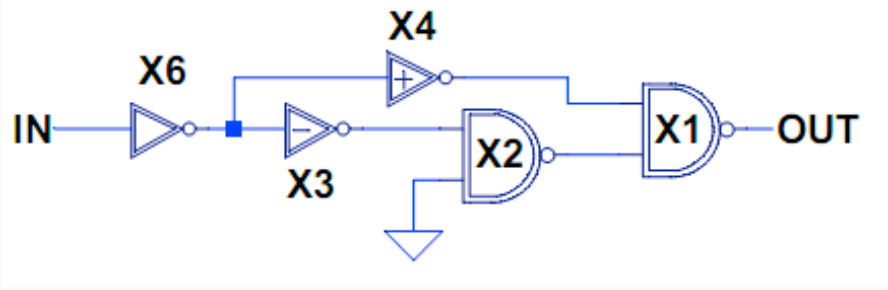


Figure D.18. Cycling down gate, function $1\bar{1}0$, from Mouftah[132]

The gates simulate as follows:

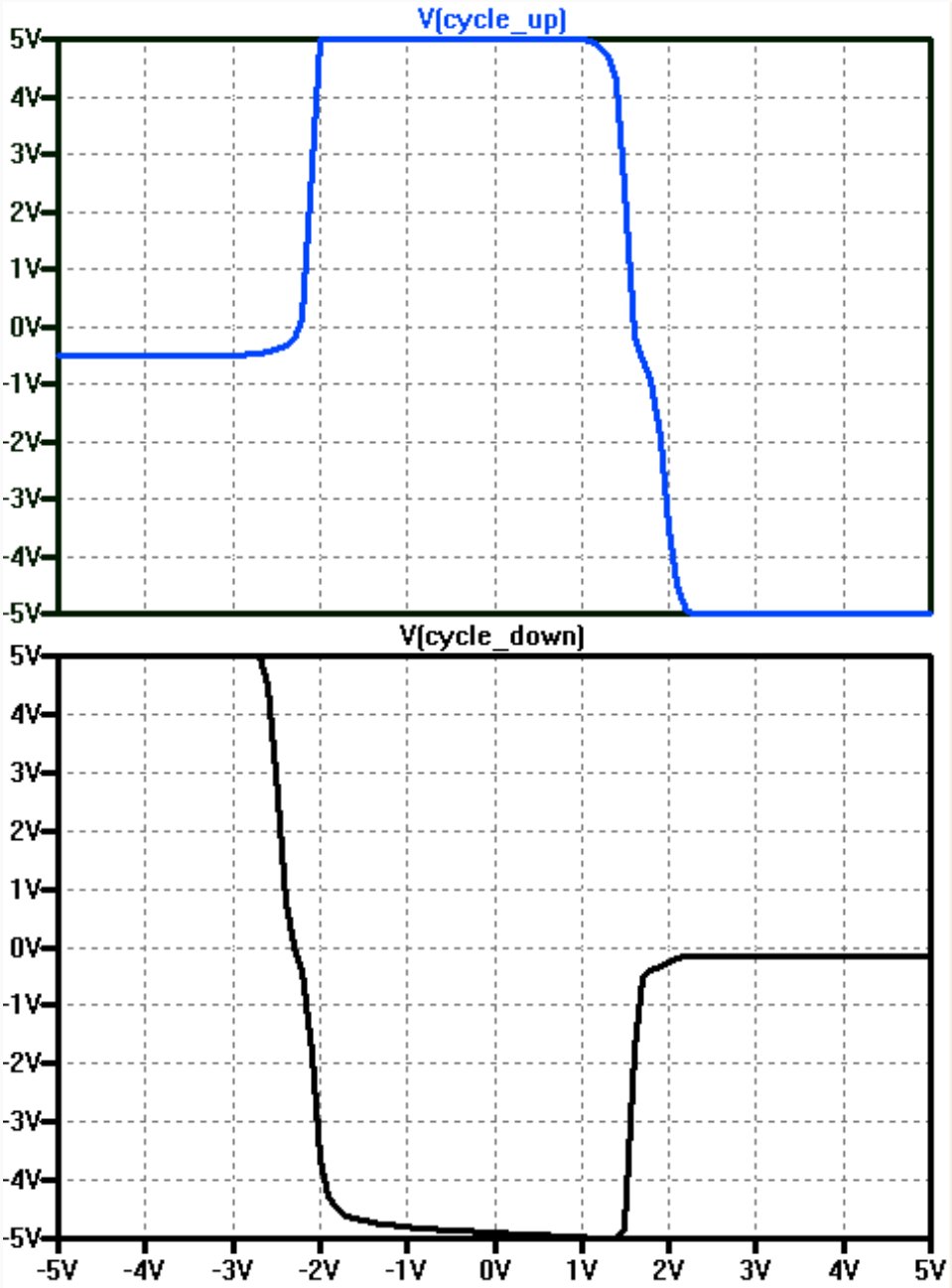


Figure D.19. Cycle up and cycle down Voltage Transfer Characteristics.

The cycle down gate functions as expected:

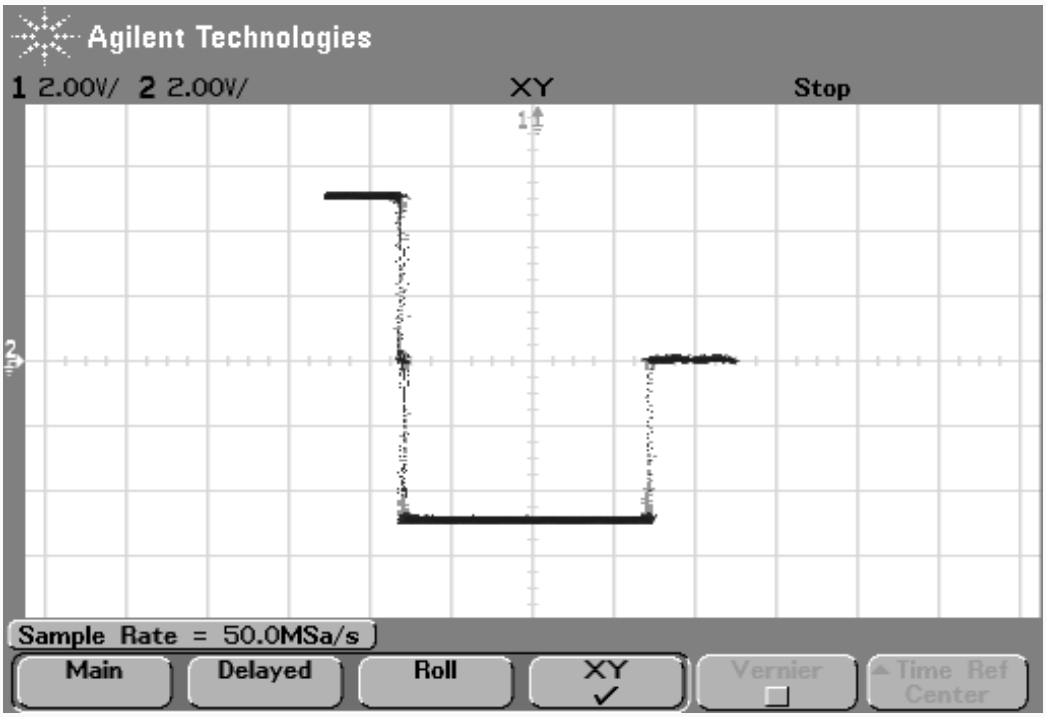


Figure D.20. Cycle down gate VTC as observed in lab.

If the input is connected to the output, the cycling down gate oscillates at a high frequency, though it does not reach each logic level:

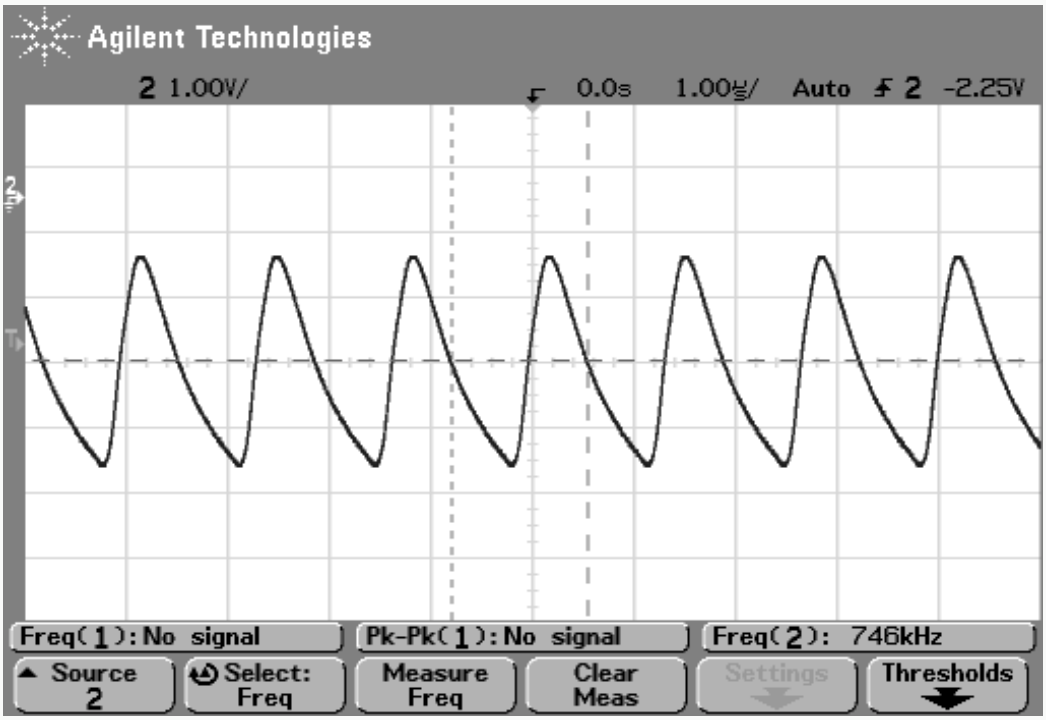


Figure D.21. Cycle down gate with input connected to output.

This indicates a switching speed limit in the devices we used.

Lastly, we built and tested a cycle up gate in lab that behaved as expected:

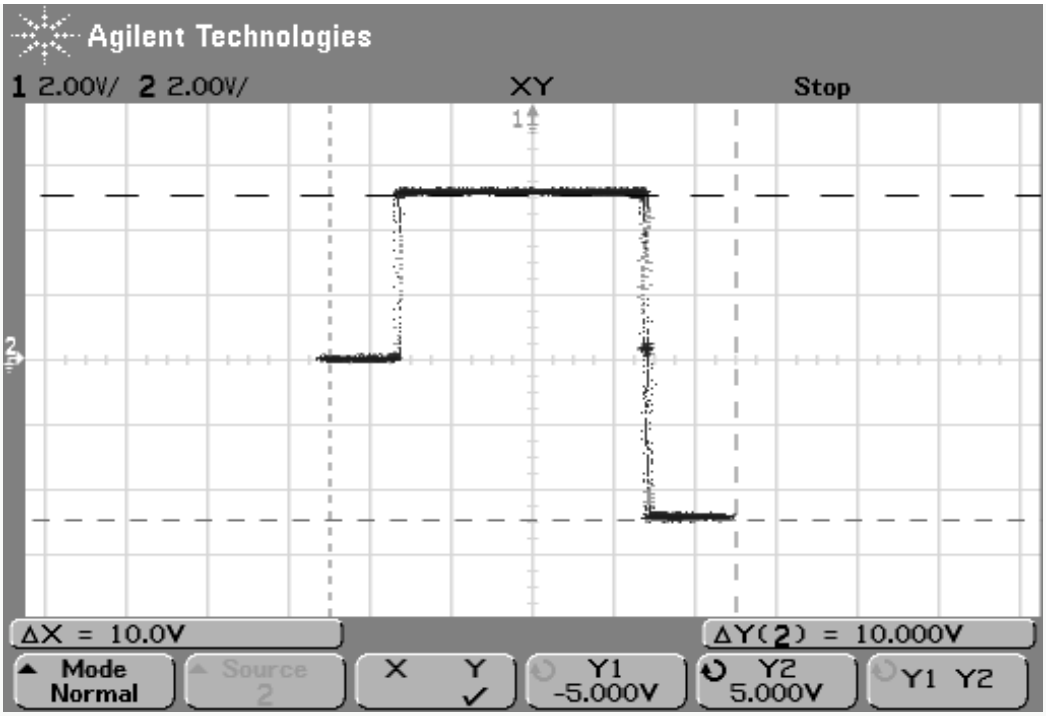


Figure D.22. Cycle up gate VTC as observed in lab.

D.3.5. Shift Gates

Grubb invented the "shift up" and "shift down" gates (the "shift" terminology is not to be confused with "bit shifting"), which essentially add or subtract 1 from the input, without wrapping around like the cycling gates (which Grubb called "rotate" gates). Although Grubb used BJTs to make these gates, they can be implemented with Mouftah's inverters and diode gates instead.

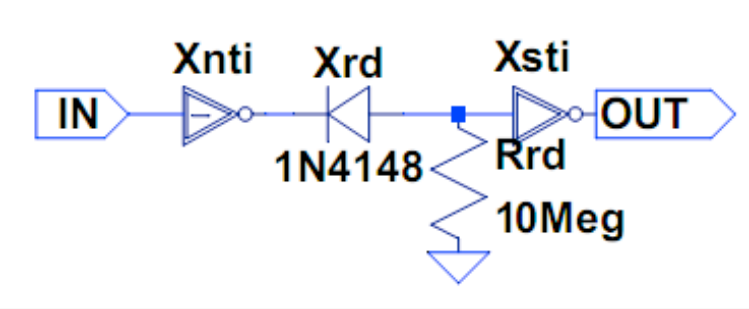


Figure D.23. Schematic for Grubb's shift up gate implemented using Mouftah's unary gates.

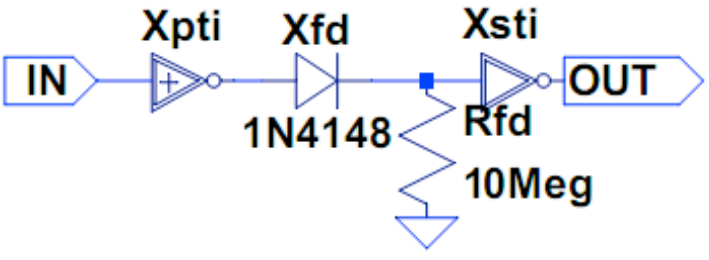


Figure D.24. Schematic for Grubb's shift down gate implemented using Mouftah's unary gates.

We only simulated these gates in LTspice:

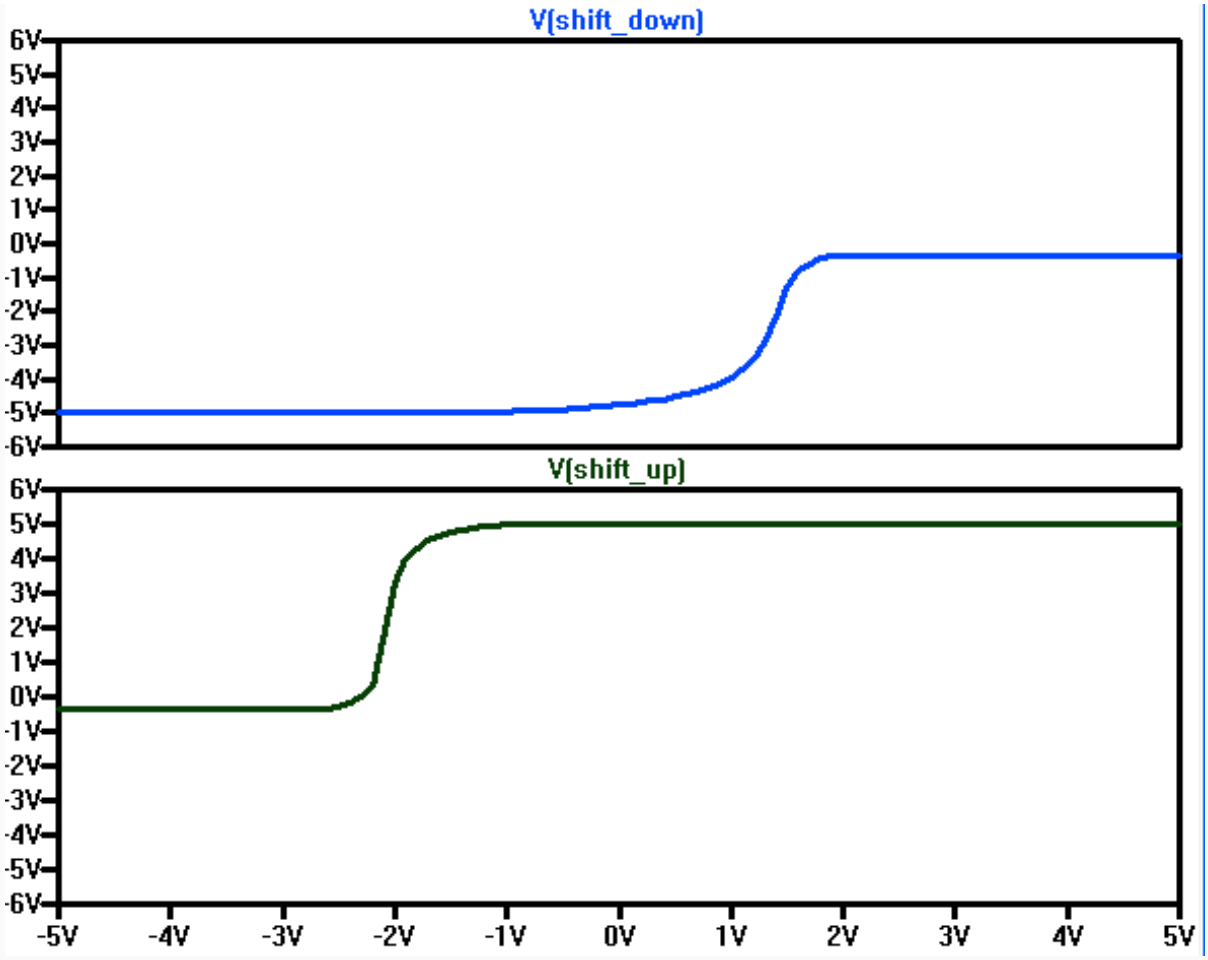


Figure D.25. VTC of shift up and down gates.

The shift up and down gates were not physically built because a trinary computer can be built without them.

D.4. Dyadic Logic Gates

D.4.1. TNAND

Invented by Mouftah[133], the TNAND is constructed similarly to a binary NAND gate: NMOS in series, PMOS in parallel, as follows:

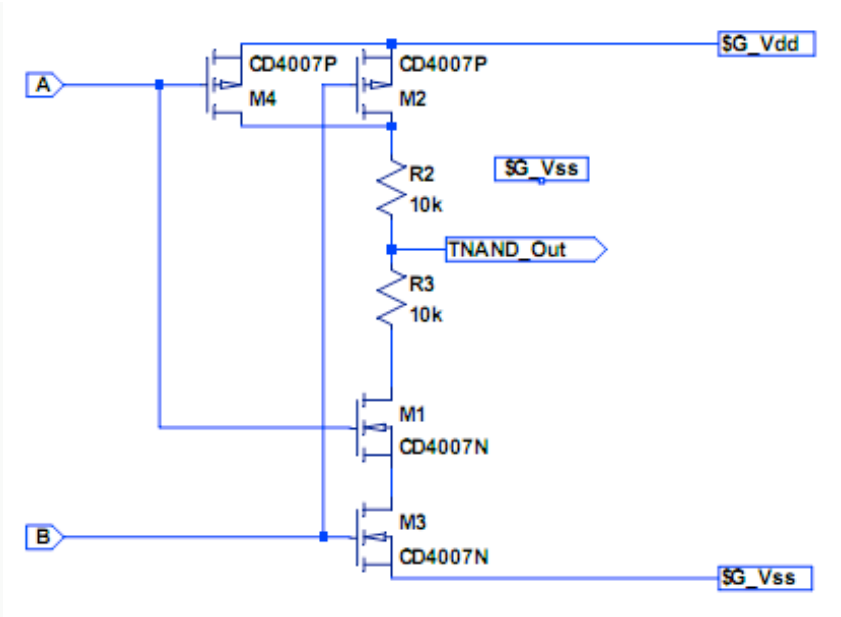


Figure D.26. TNAND Schematic in SwCAD III

.

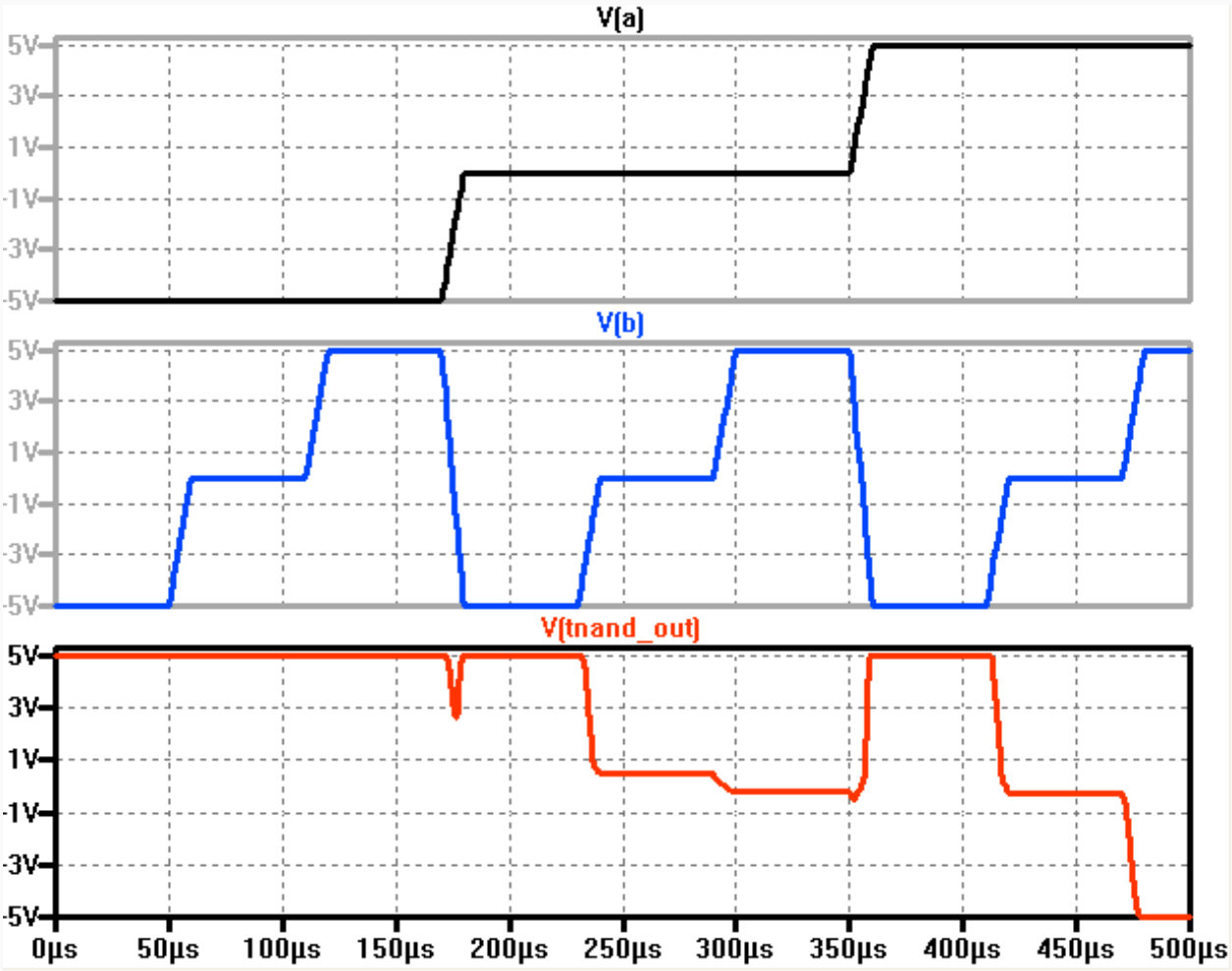


Figure D.27. Transient response of TNAND in LTSpice.

Three voltage transfer characteristics can be produced by hardwiring one of the inputs to a fixed value:

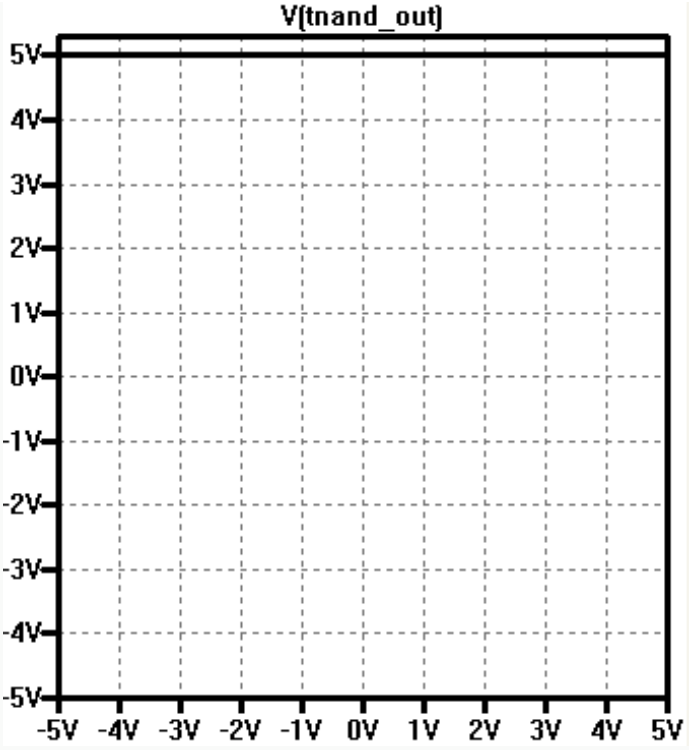


Figure D.28. TNAND VTC with A=-5 V, created in LTSpice.

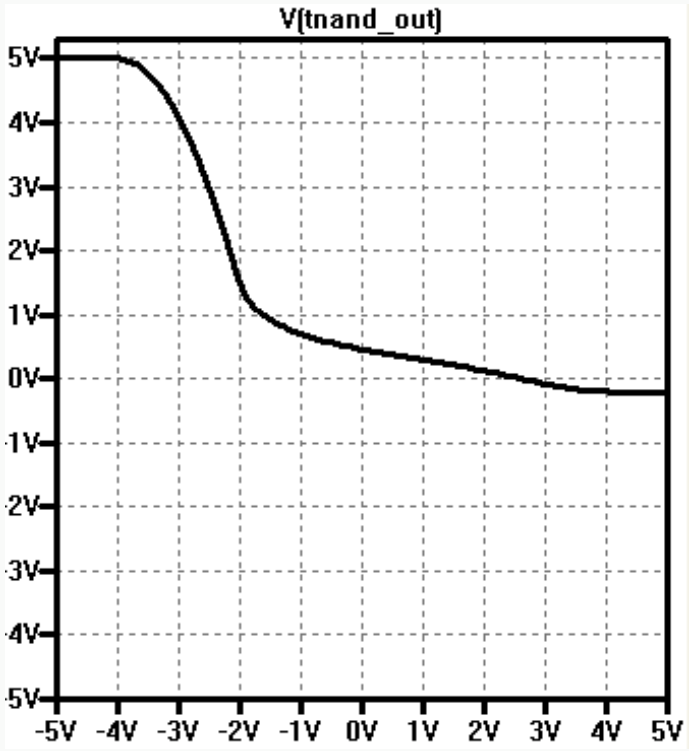


Figure D.29. TNAND VTC with A=0 V, created in LTSpice.

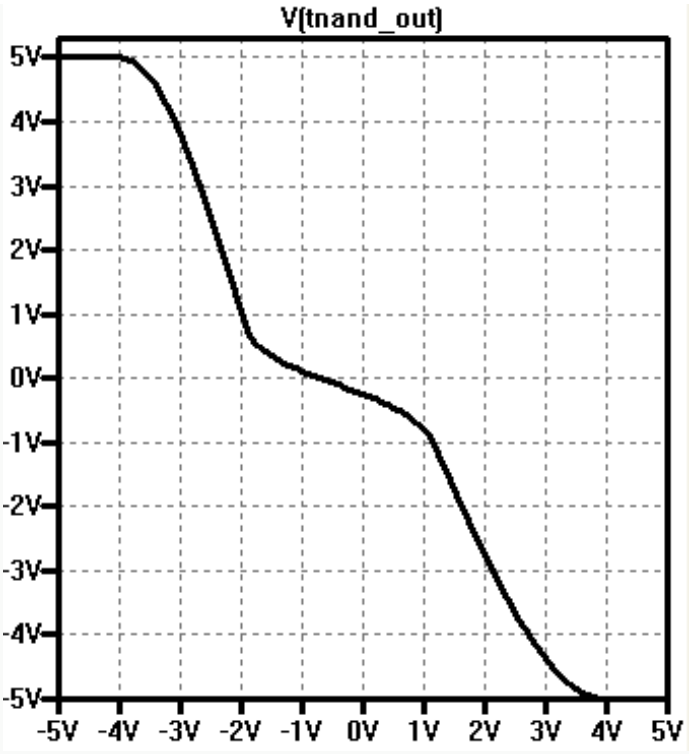


Figure D.30. TNAND VTC with A=5 V, created in LTSpice.

In lab, our observations matched the simulations:

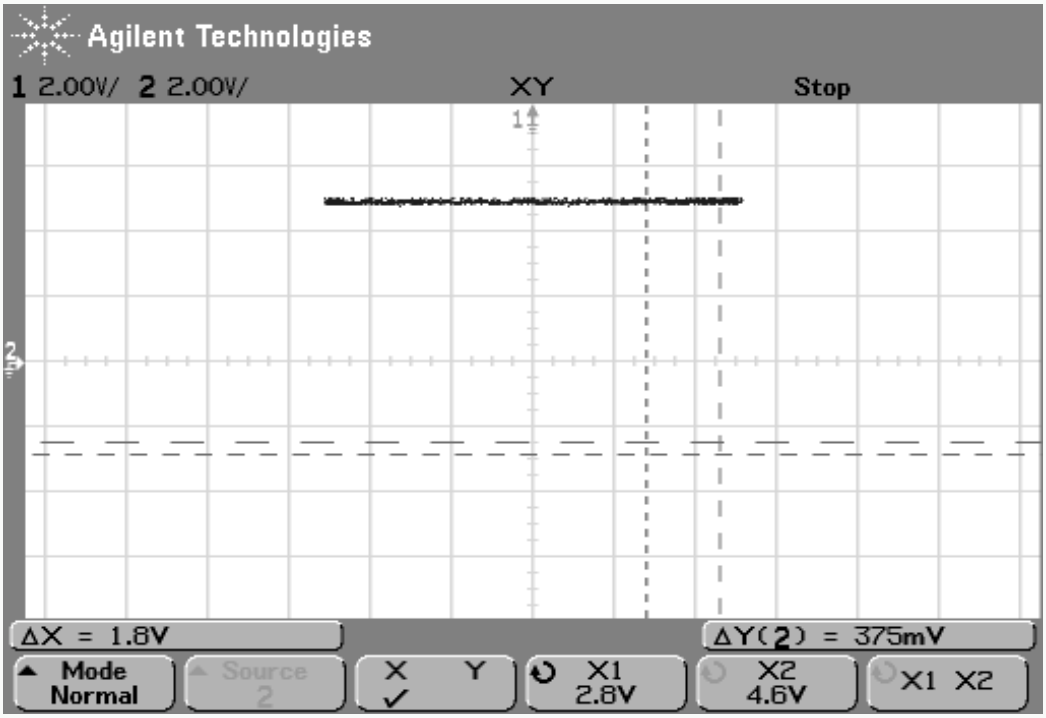


Figure D.31. Voltage Transfer Characteristic for TNAND with R = 12 kΩ, B = -5 V.

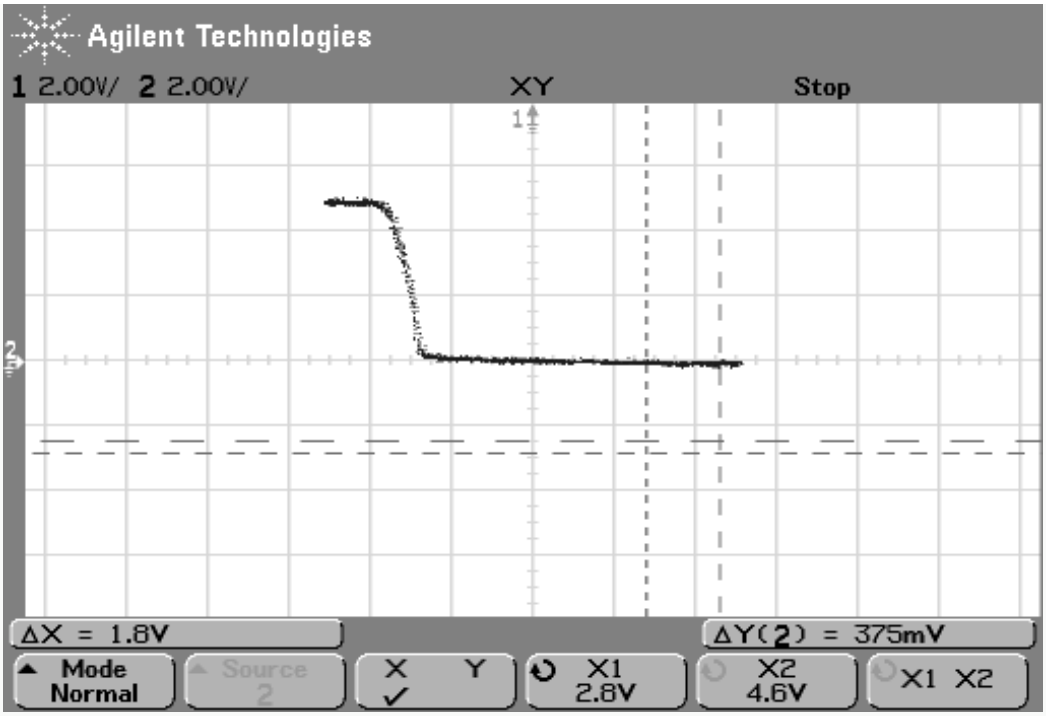


Figure D.32. Voltage Transfer Characteristic for TNAND with R = 12 kΩ, B = 0 V.

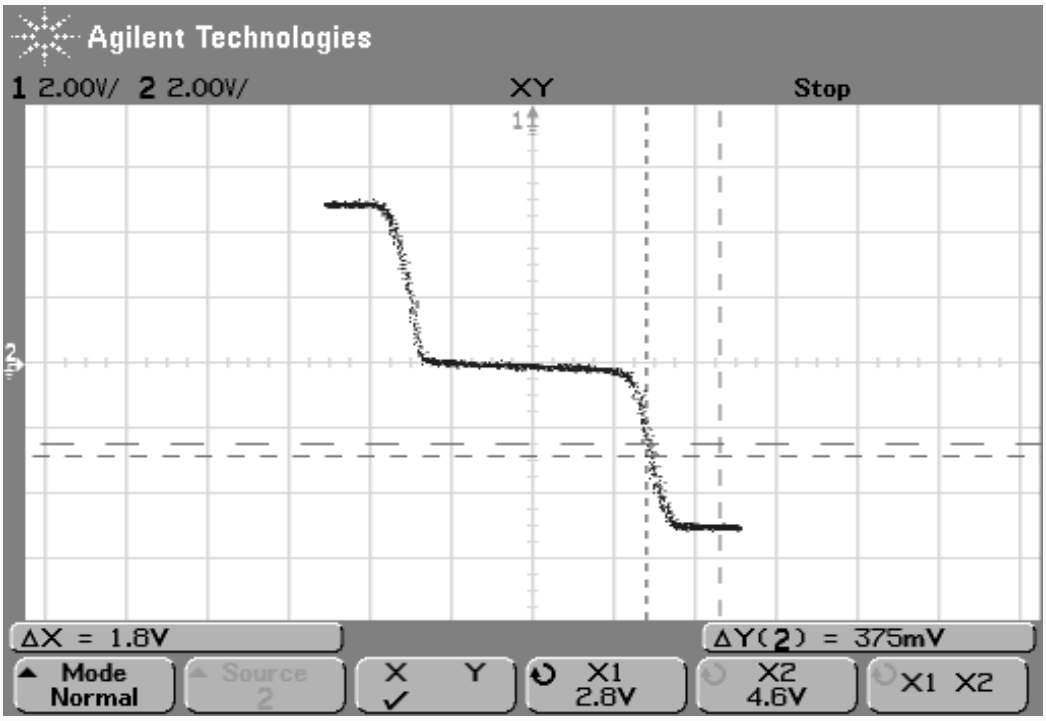


Figure D.33. Voltage Transfer Characteristic for TNAND with R = 12 kΩ, B = 5 V.

D.4.2. TNOR

Also invented by Mouftah, the TNOR gate is constructed by wiring NMOS in parallel, PMOS in series:

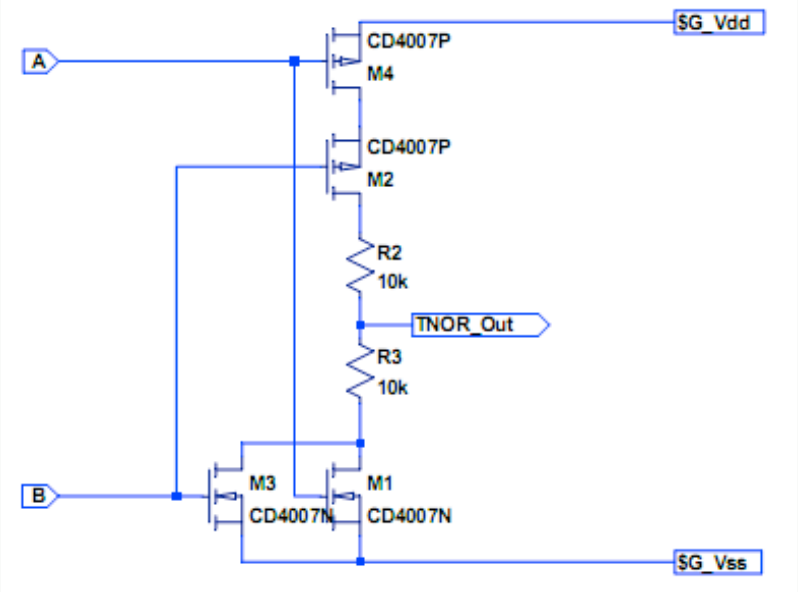


Figure D.34. TNOR schematic made in SwCAD.

When simulated in LTspice, the results are as follows:

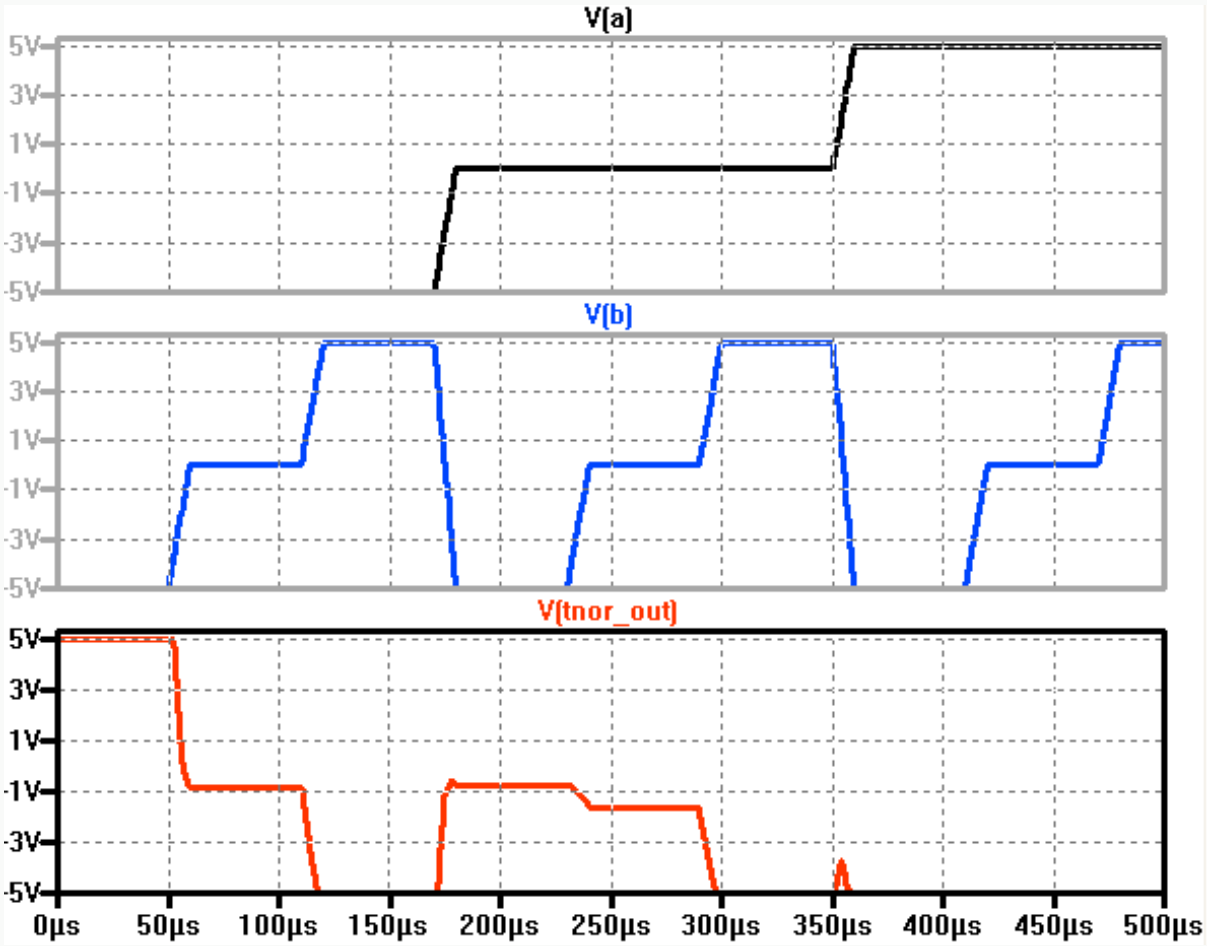


Figure D.35. TNOR transient response from LTSpice.

The voltage transfer characteristics are as follows:

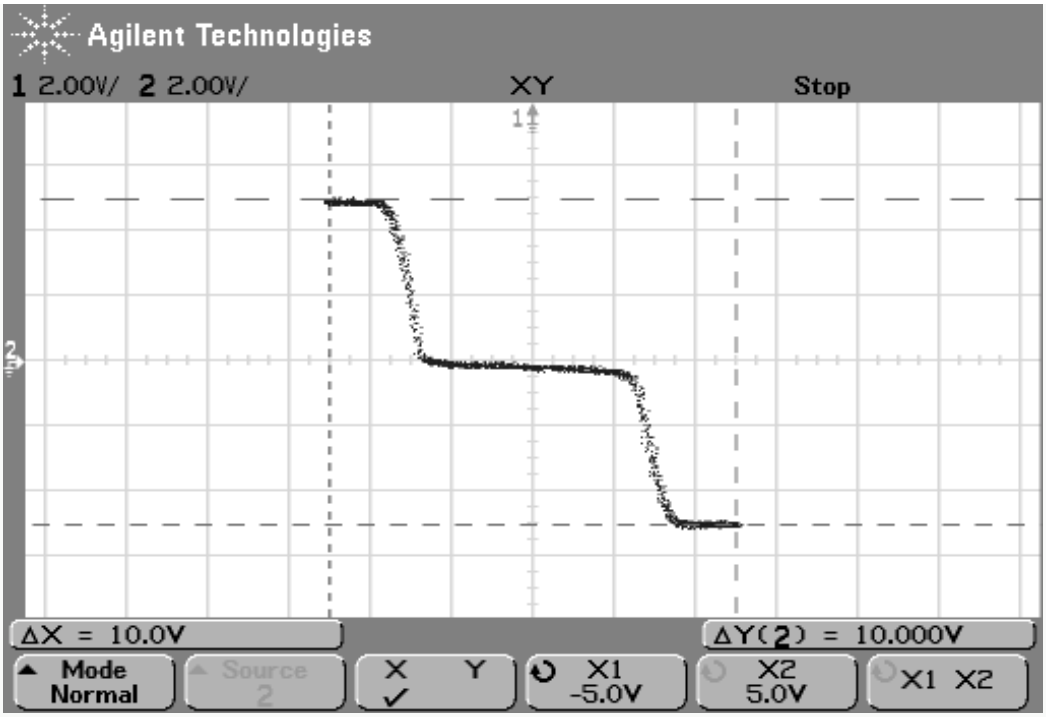


Figure D.36. Voltage Transfer Characteristic for TNOR with R = 12 kΩ, B = -5 V

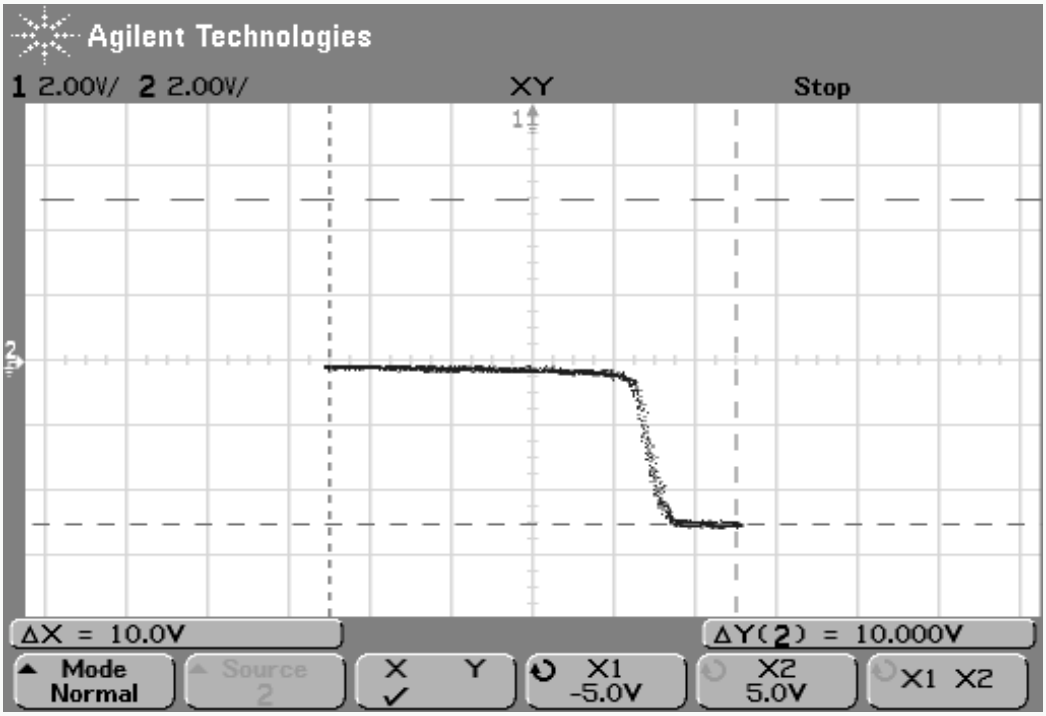


Figure D.37. Voltage Transfer Characteristic Curve for TNOR with R = 12 kΩ

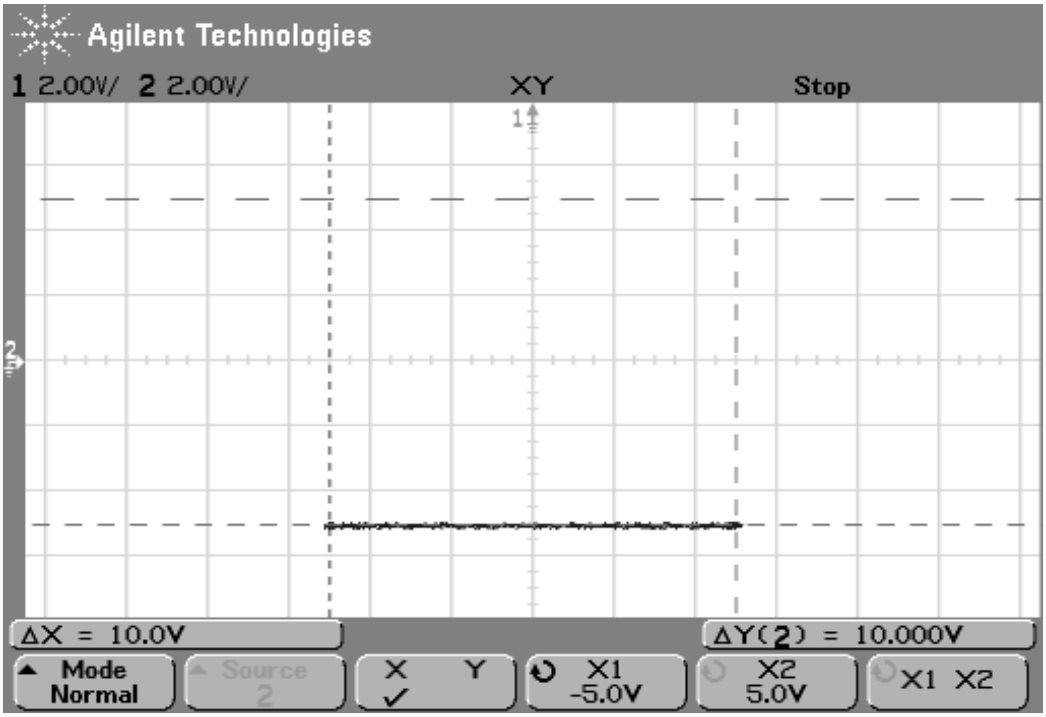


Figure D.38. Voltage Transfer Characteristic for TNOR with R = 12 kΩ, B = 5 V

From the VTCs, one can derive the critical voltages for TNOR with A = sine(-5 to 5 V), B = -5 V, +/- 5 V supply, as follows:

Table D.2. Critical Voltages for TNOR

Quantity	Value (V)
V_{OH}	4.959 to 4.915
V_{OM}	0.125 to -0.375
V_{OL}	-4.875 to -5.062
V_{IH}	3.3 to 5.0
V_{IM}	2.4 to -2.8
V_{IL}	-5.1 to -3.5
V_{MH}	3.219 ($V_{in} = -V_{out}$, $V_{out} < 0$)
V_{MM}	-0.125 ($V_{in} = V_{out}$)
V_{ML}	2.820 ($V_{in} = -V_{out}$, $V_{out} > 0$);

D.4.3. TOR

The TOR gate is TNOR followed by a Simple Ternary Inverter.

We only tested this in lab, but the experimental results matched what we expected:

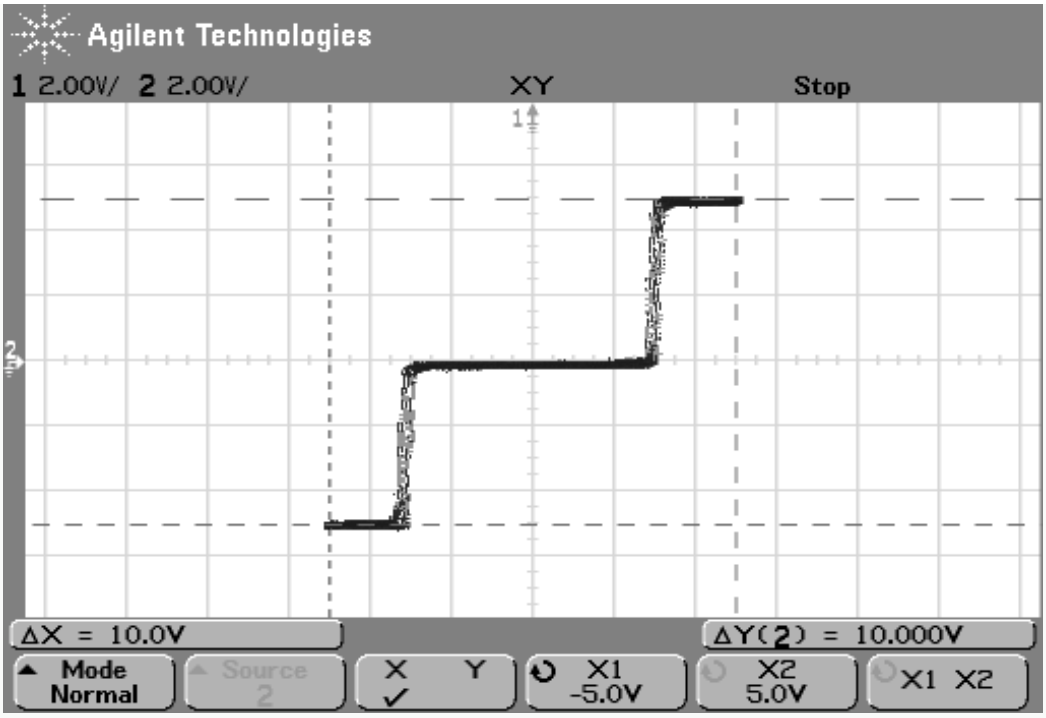


Figure D.39. Voltage Transfer Characteristic for TOR (TNOR + STI) with R = 12 kΩ, B = -5 V.

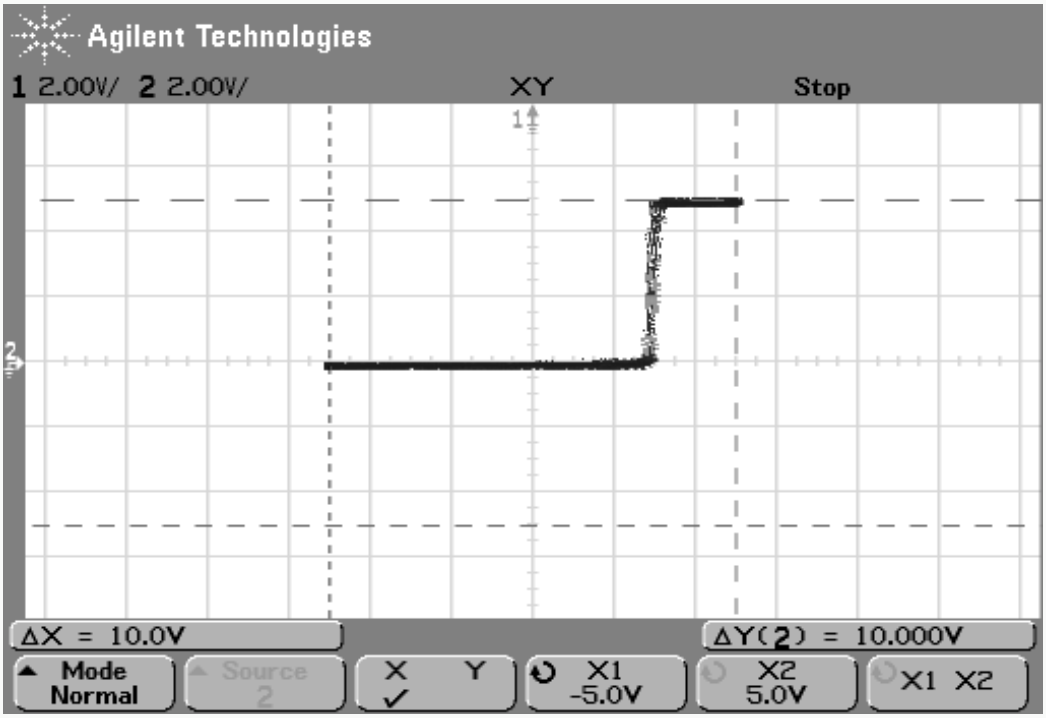


Figure D.40. Voltage Transfer Characteristic for TOR (TNOR + STI) with R = 12 kΩ, B = 0 V.

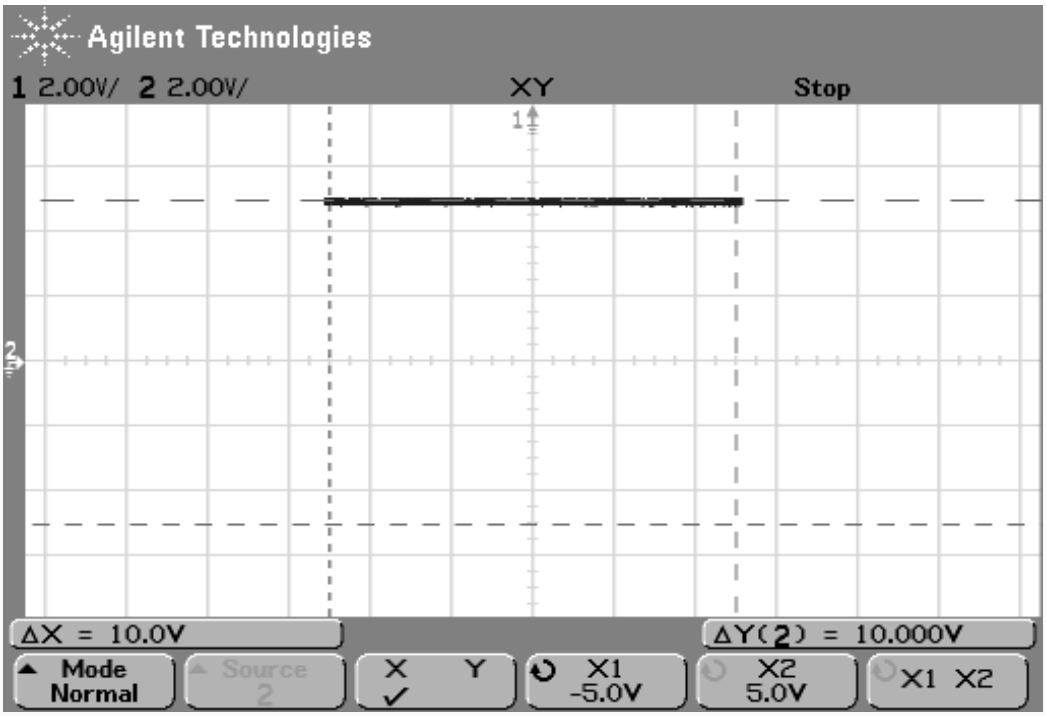


Figure D.41. Voltage Transfer Characteristic for TOR (TNOR + STI) with R = 12 kΩ, B = 5 V.

D.4.4. TAND

Analogous to TOR, the TAND gate is a TNAND followed by a Simple Ternary Inverter. Again, we only tested this in lab, but the results were as expected:

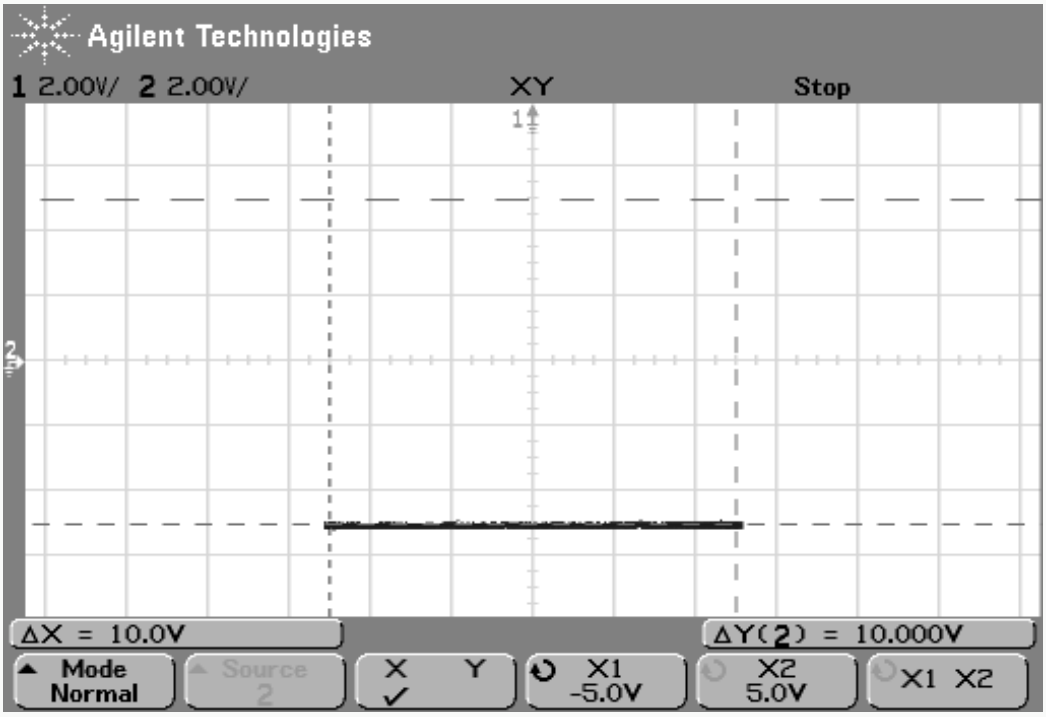


Figure D.42. Voltage Transfer Characteristic for TAND (TNAND + STI) with R = 12 kΩ, B = -5 V.

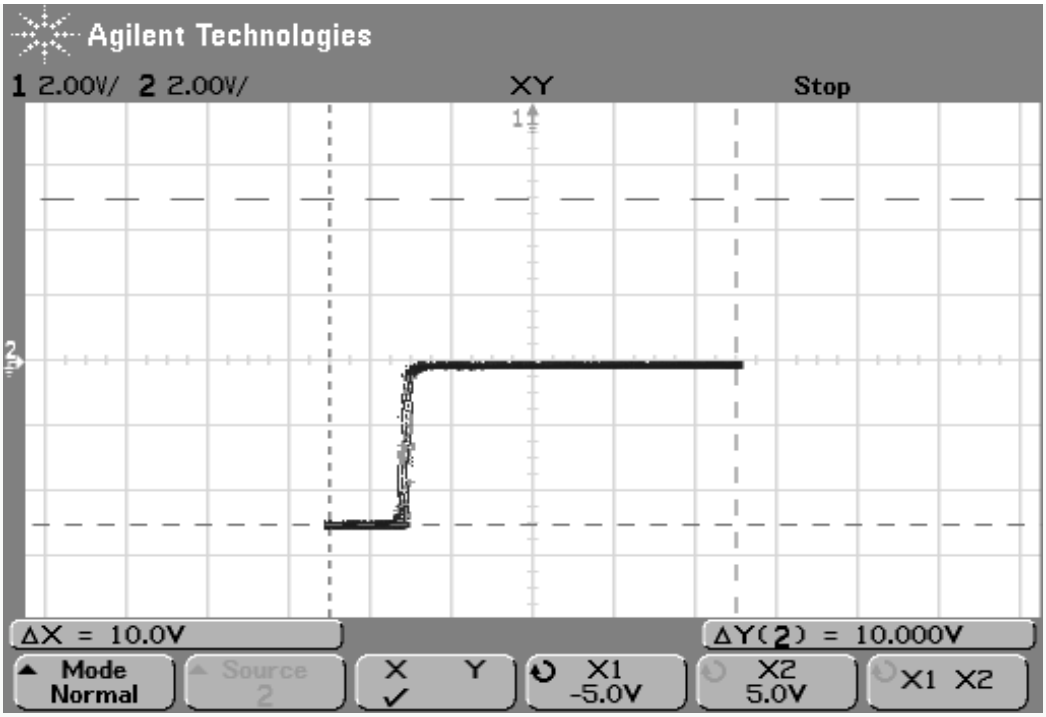


Figure D.43. Voltage Transfer Characteristic for TAND (TNAND + STI) with R = 12 kΩ, B = 0 V.

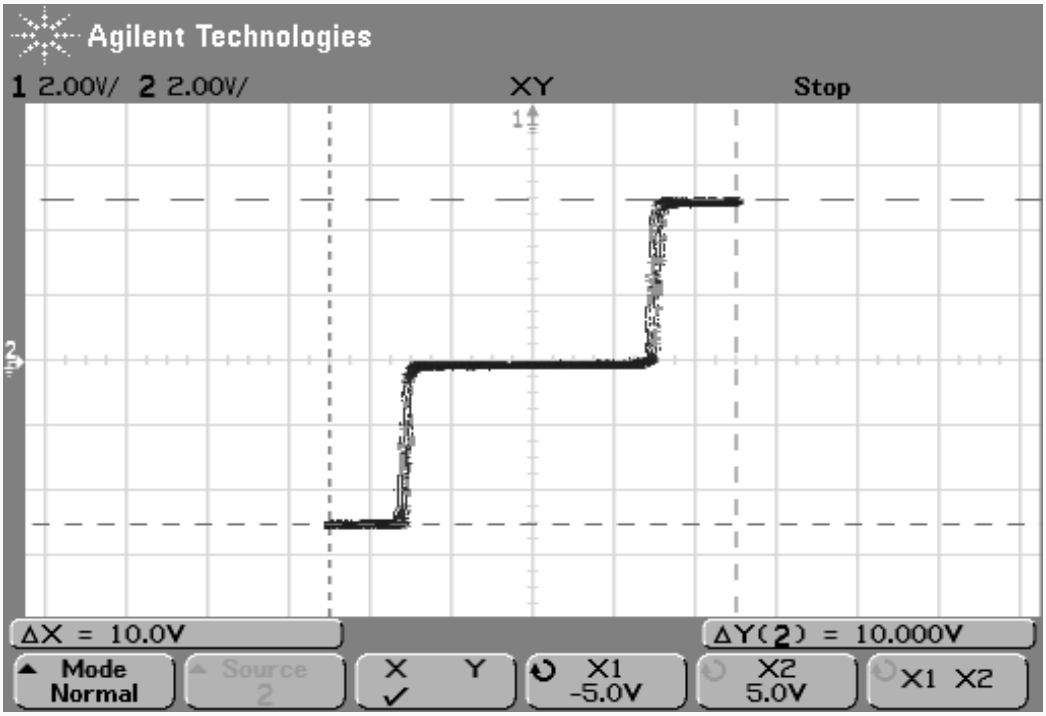


Figure D.44. Voltage Transfer Characteristic for TAND (TNAND + STI) with R = 12 kΩ, B = +5 V.

D.5. Flip-flap-flops

In binary logic, cross-coupled inverters form a static RAM cell, or a flip-flop. Knuth[134] coined the term flip-flap-flop to describe a three-state static memory device:

> Perhaps the symmetric properties and simple arithmetic of this number system will prove to be quite important some day -- when "flip-flop" is replaced by "flip-flap-flop".

Merrill[135] claims "Tristable flip-flops which use two transistors where A on B off, B on A off, and A and B

off were used to achieve the three stable states have been successfully operated by several people." There are several types of flip-flap-flops (or tri-latches) that can be built.

D.5.1. PZN Tri-Latch

A PZN tri-latch ("tri-latch" instead of "tri-flop" because it is unclocked; Mouftah uses different terminology but this is more standard in recent times) can be constructed by cross-coupled TNOR gates:

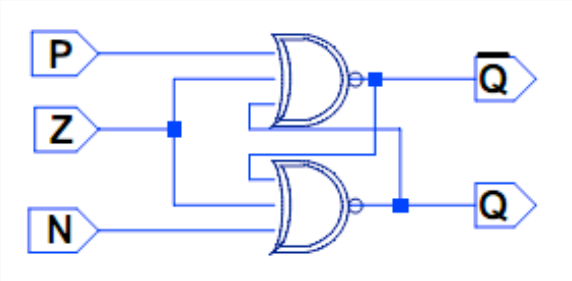


Figure D.45. PZN tri-flop, from Mouftah's Image:Mouftah-8a-PZN Tri-flop.png from Mouftah's patent[136]

With P, Z, and N negative, the tri-flop holds the output, since $\overline{x + 1} = \overline{x}$, and x inverted twice is x itself. In the hold state, the tri-flop behaves as two cross-coupled simple ternary inverters. The value stored can be modified by briefly raising the P, Z, or N inputs:

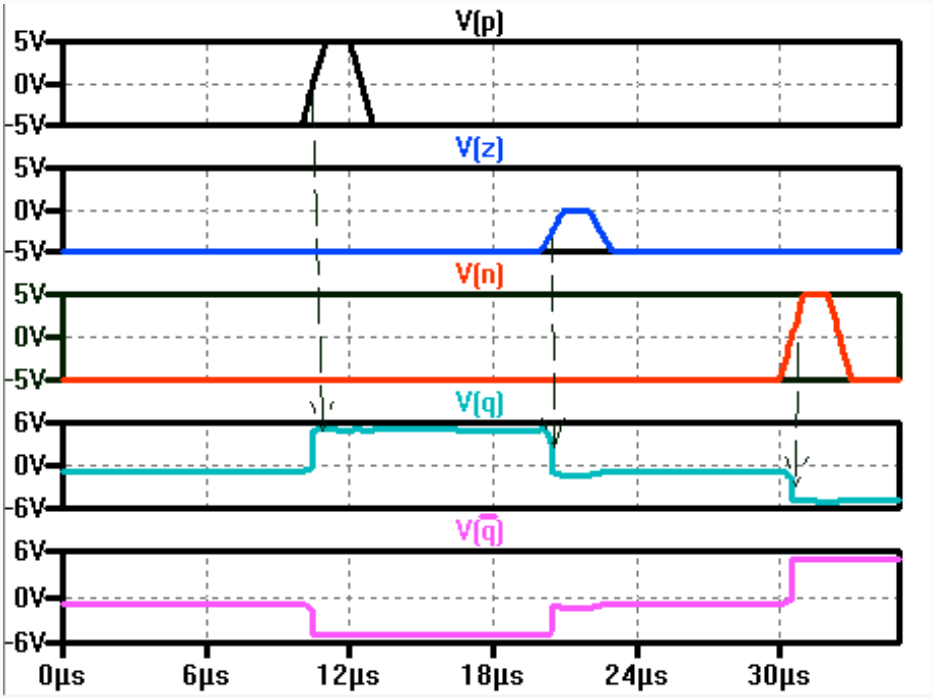


Figure D.46. PZN tri-flop timing diagram, using Mouftah's transitions. Set positive with a -5 to +5 pulse on P, zero with a -5 to 0 pulse on Z, and negative with a -5 to +5 pulse on N.

The pulses above are what Mouftah covers. Conceptually, a -5 to +5 pulse is used on the N and P inputs, to fully bring Q to its opposite value, but only a -5 to 0 voltage swing is required on the Z input since any previous value is only at most 5 volts away from zero. Logically, a positive input quickly stabilizes the stored value.

Table D.3. PZN Tri-Flop Input Signals

Signal	Pulse Operation
P	$\overline{1}$ to 1 Set 1
Z	$\overline{1}$ to 0 Set 0
N	$\overline{1}$ to 1 Set $\overline{1}$

However, a $\overline{1}$ to 0 pulse and $\overline{1}$ to 1 pulse can also be applied any of the inputs:

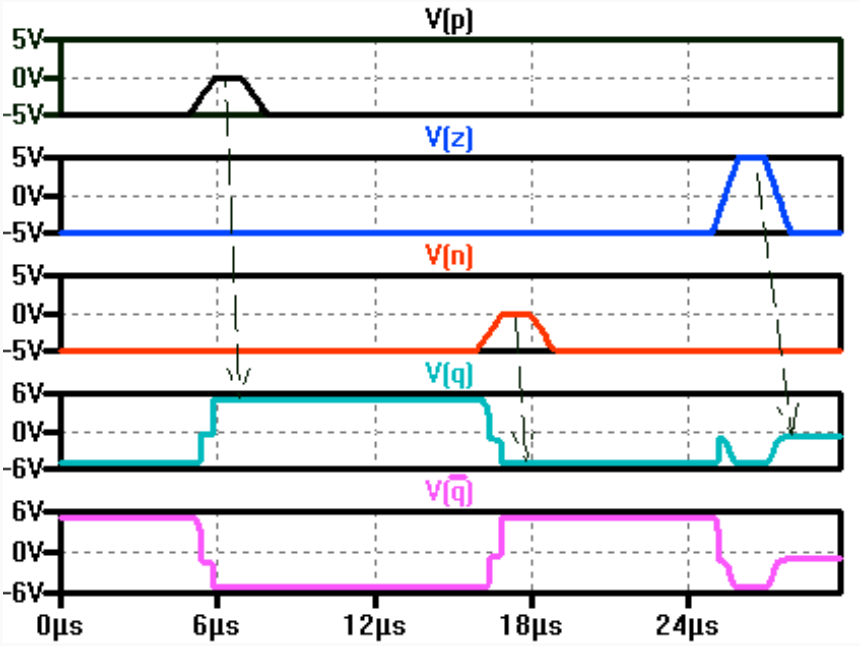


Figure D.47. PZN tri-flop timing diagram with additional pulses, not mentioned by Mouftah. -5 to 0 on P sets positive, -5 to +5 on Z sets zero (with caveat), and -5 to 0 on N sets negative.

Here, the transition is not as smooth, requiring more time for the logic values to stabilize. Particularly, if Z goes to +5, then Q swings to -5, although Q reverts to 0 once Z returns to its idle state. The truth table will not be reproduced here, although it is available in the referenced papers.

D.5.2. PZN Tri-Flop

What Mouftah calls the "clocked PZN tri-flop" I just call the "PZN tri-flop", using the term "PZN tri-latch" for the unclocked variant:

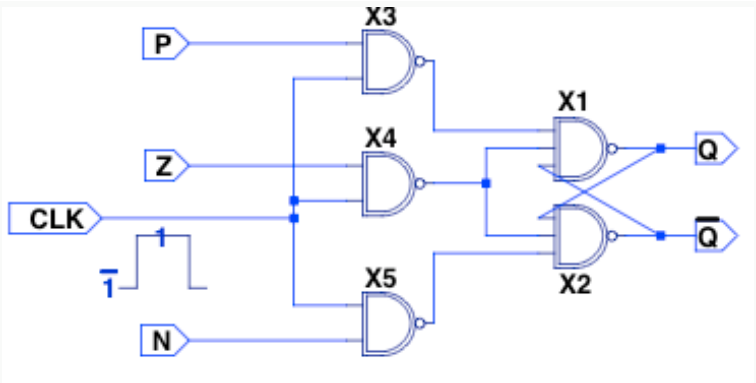


Figure D.48. Mouftah's clocked PZN tri-flop, from Image:Mouftah-9-Clocked PZN Tri-flop.png.

This circuit is available in the git repository as `pznflop`. Notice how the P, Z, and N signals only have an effect on the value stored when CLK is high:

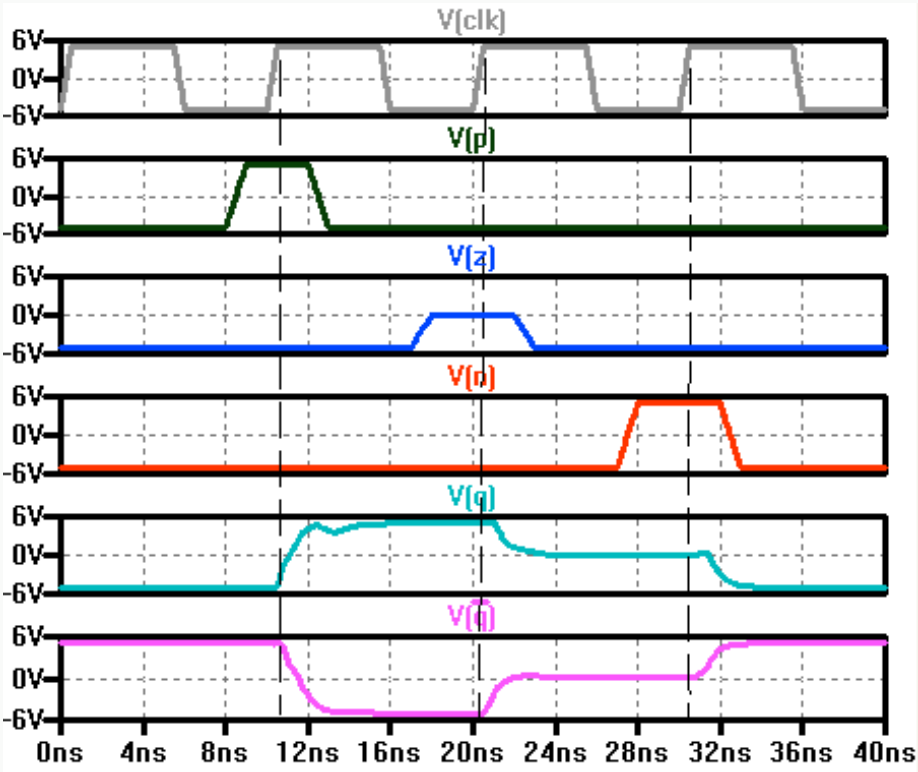


Figure D.49. Timing diagram for clocked PZN tri-flop.

D.5.3. T-Type Tri-Flop with PZN Inputs

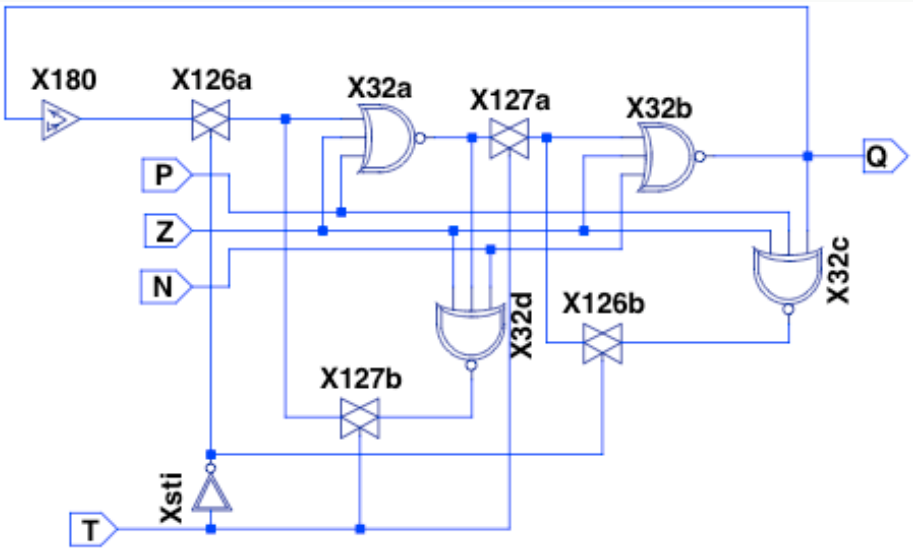


Figure D.50. T-type tri-latch with P, Z, and N inputs, from Image:Mouftah-13-T-Type Tri-Flop.png.

Table D.4. Next-states table for T-type tri-flop, based on Mouftah

T	P	Z	N	Q(t)	Q(t + 1)
1	$\overline{1}$	$\overline{1}$	$\overline{1}$	$\overline{1}$	0
1	$\overline{1}$	$\overline{1}$	$\overline{1}$	0	1
1	$\overline{1}$	$\overline{1}$	$\overline{1}$	1	$\overline{1}$
1/$\overline{1}$	1	$\overline{1}$	$\overline{1}$	X	1
1/$\overline{1}$	$\overline{1}$	0	$\overline{1}$	X	0
1/$\overline{1}$	$\overline{1}$	$\overline{1}$	1	X	$\overline{1}$
$\overline{1}$	$\overline{1}$	$\overline{1}$	$\overline{1}$	X	Q(t)
Other combinations				X	F

In this table, X = don't care ("D" in Mouftah's paper), and F is unspecified.

Pulsing the T input cycles the tri-flop up. Pulsing the P, Z, or N inputs take precedence over T, and set the tri-flop to 1, 0, or $\overline{1}$ respectively. (Mouftah notes that a T-type latch can be implemented without the property that P, Z, and N override T by using simple ternary inverters instead of TNOR gates.) Check it out:

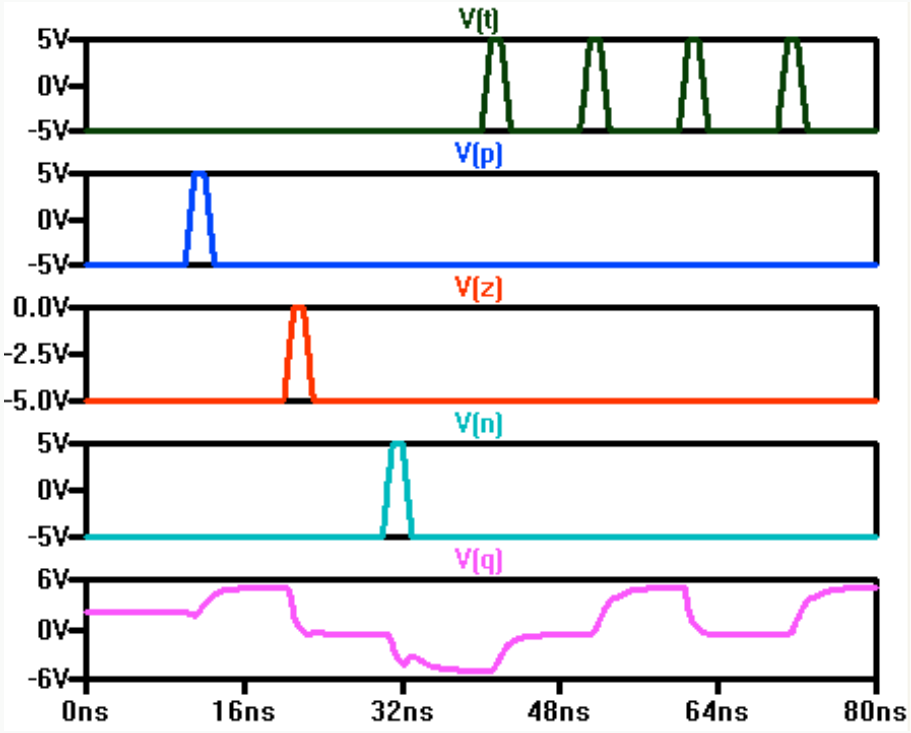


Figure D.51. Timing diagram of T-type tri-latch.

One problem is that the toggle input does not cycle through $\overline{1}$, 0, 1 completely, instead $\overline{1}$, 0, 1, 0, 1, back-and-forth between 0 and 1 alternatively. This has not yet been solved.

D.5.4. D Tri-Latch

Mouftah constructs what he calls a D-type ternary tri-flop by gating the D (data) signal to the inputs of a PZN latch. While Mouftah calls this a tri-flop, the term "latch" is now more commonly used to refer to unclocked

devices, so we'll call this a D-type tri-latch, reserving the term "tri-flop" for devices with clocks.

The D tri-latch simply stores whatever value is received on D. There is no clock, so Q always follows D. D-type flip-flops with a built-in clock are more common and generally used.

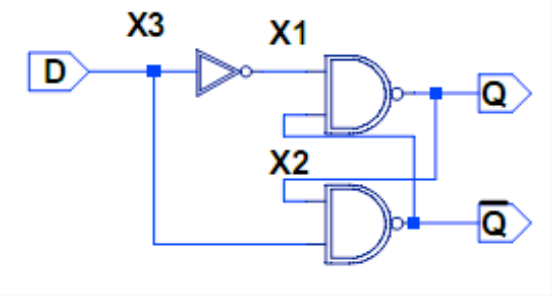


Figure D.52. D-Type Ternary Latch, from Image:Mouftah-10-D-type Tri-flop (latch).png

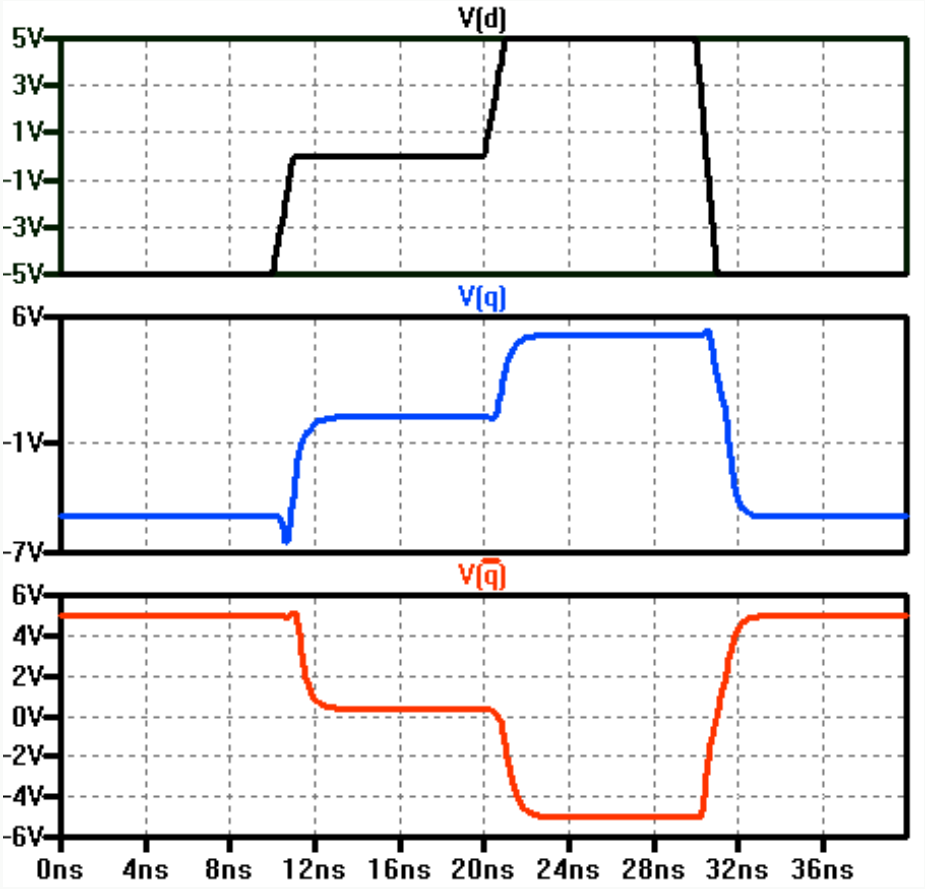


Figure D.53. Timing diagram for D latch.

D.5.5. D Tri-Flop

The D-type tri-flop is useful for storing arbitrary trits on the D input. However, it is level-triggered, so the D input must be held stable during the high duration of the clock pulse, otherwise the input will also change. The level-triggered nature of this tri-flop limits its usefulness.

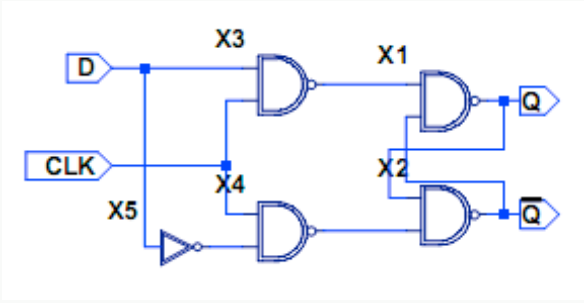


Figure D.54. D flip-flop.

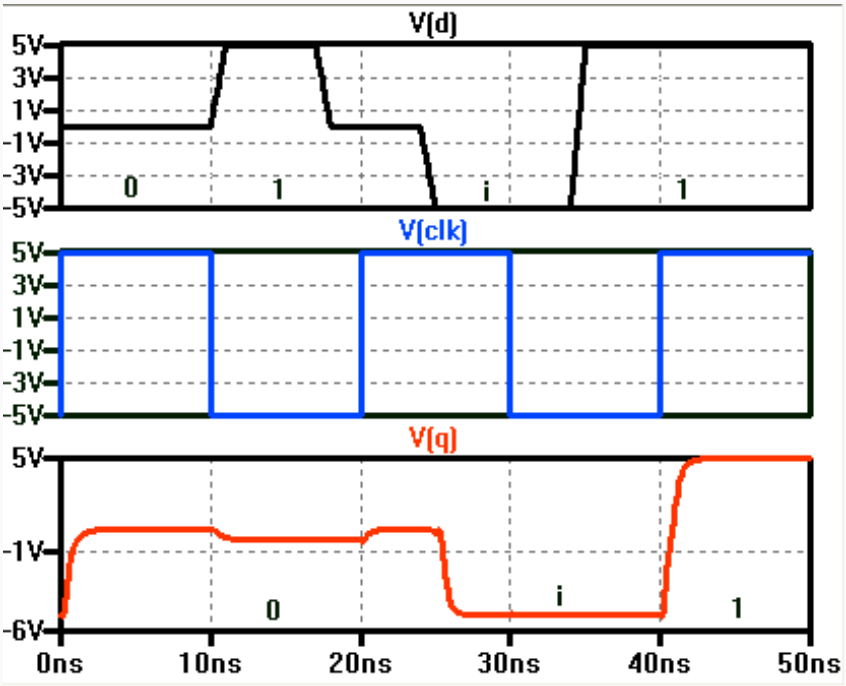


Figure D.55. D flip-flop timing diagram. The data is clocked it only when the clock is high.

In the above diagram, first a 0 is clocked in. D then goes to 1 when CLK is low, and Q does not change logic levels (however it slightly dips when clock is low). Next, a logic $\overline{1}$ is clocked-in during the middle of a period where CLK is high, and Q goes to $\overline{1}$. Lastly, D goes to and stays at 1 while CLK is low, and when CLK goes high again Q becomes 1.

D.5.6. Rising Edge-Triggered Master-Slave D Tri-Flop: Mouftah's version

Mouftah implements the master-slave D tri-flop using CMOS transmission gates and inverters (this is the `dtflop-ms` circuit in git):

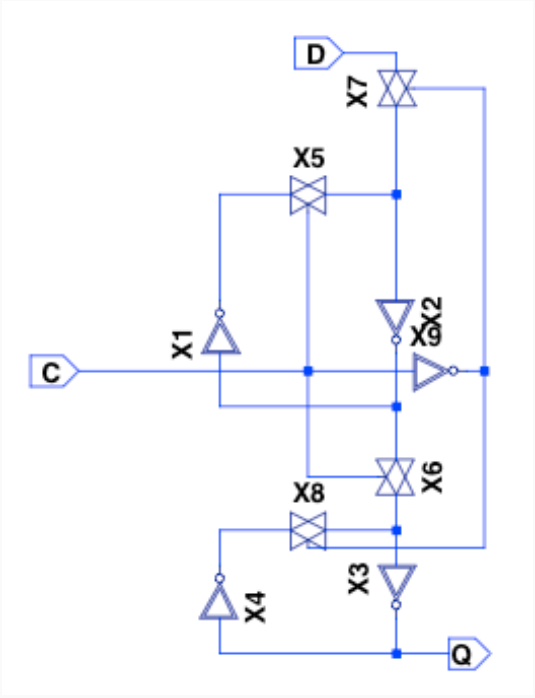


Figure D.56. Master-slave rising edge-triggered D tri-flop using transmission gates. From Image:Mouftah-11-Master Slave D-Type Tri-Flop.png.

Mouftah's master-slave tri-flop behaves very similarly to one built using two D-type tri-flops. The only noticeable difference between `dtflop-ms2` (see below) is in the transitions at the nanosecond scale, which take slightly longer most likely because the transmission gate model I have for the CD4016 includes two integrated inverters:

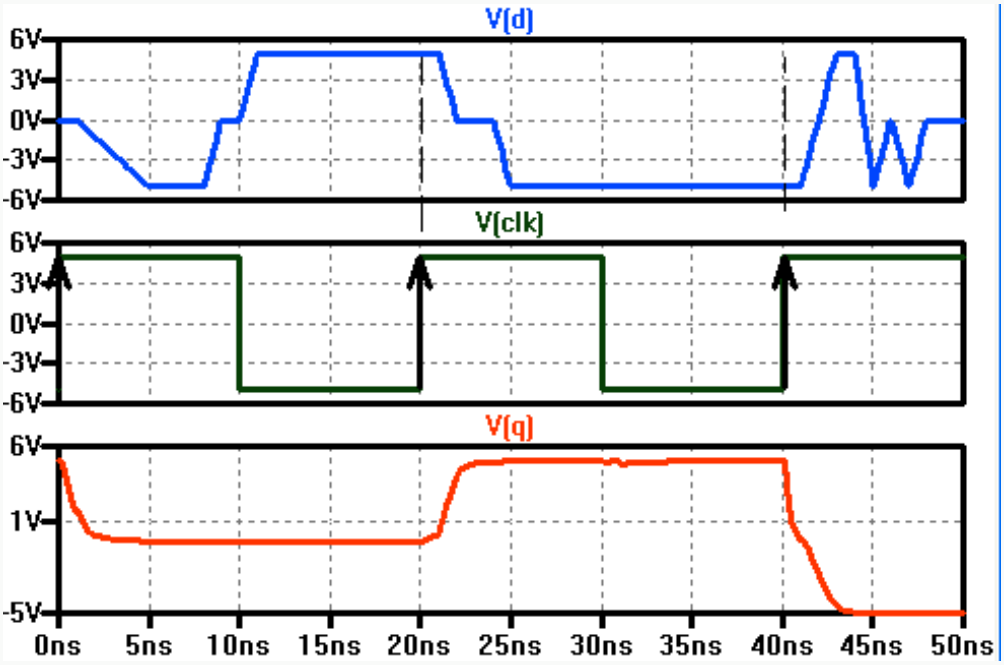


Figure D.57. Timing diagram for master-slave rising edge-triggered D tri-flop using transmission gates.

D is clocked in only on the rising edge of CLK (clock at 1 Hz):

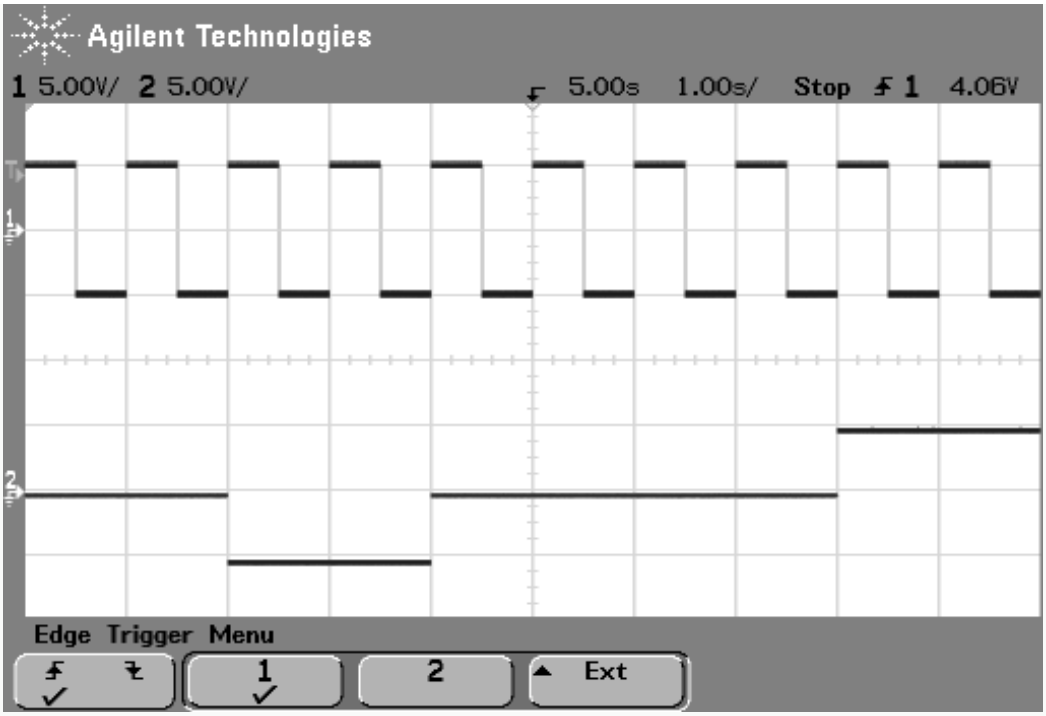


Figure D.58. Tested timing diagram for D-type master-slave tri-flop.

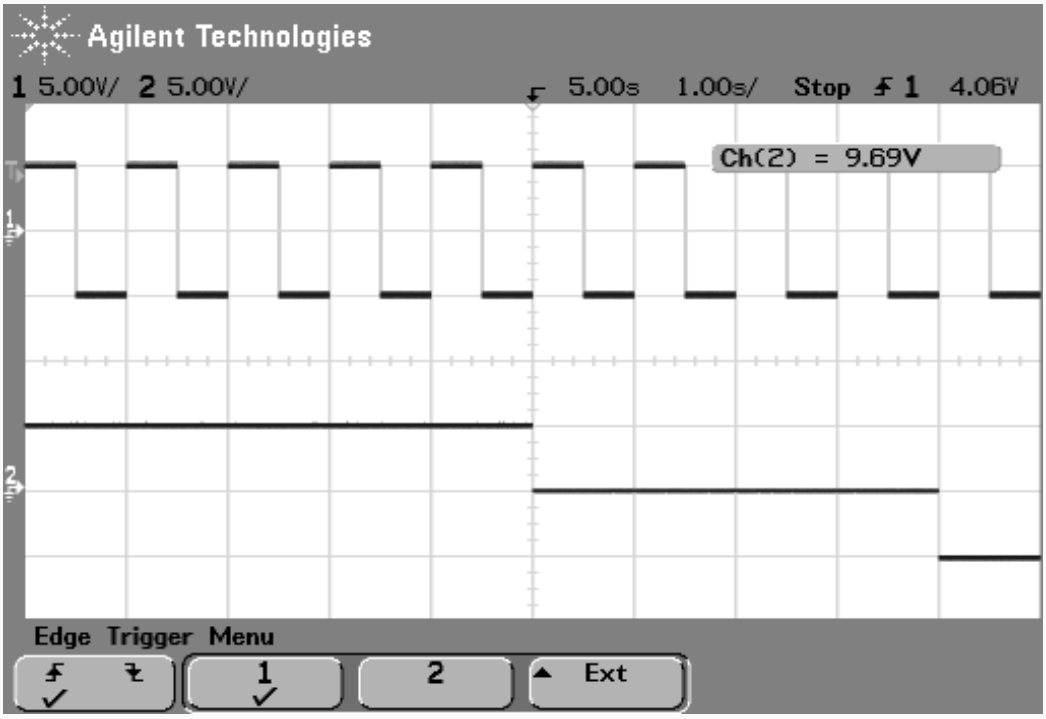


Figure D.59. Second tested timing diagram for D-type master-slave tri-flop.

We encountered an unusual behavior in lab, where when transitioning from $\overline{1}$ to 1 or 1 to $\overline{1}$, goes through the 0 transition.

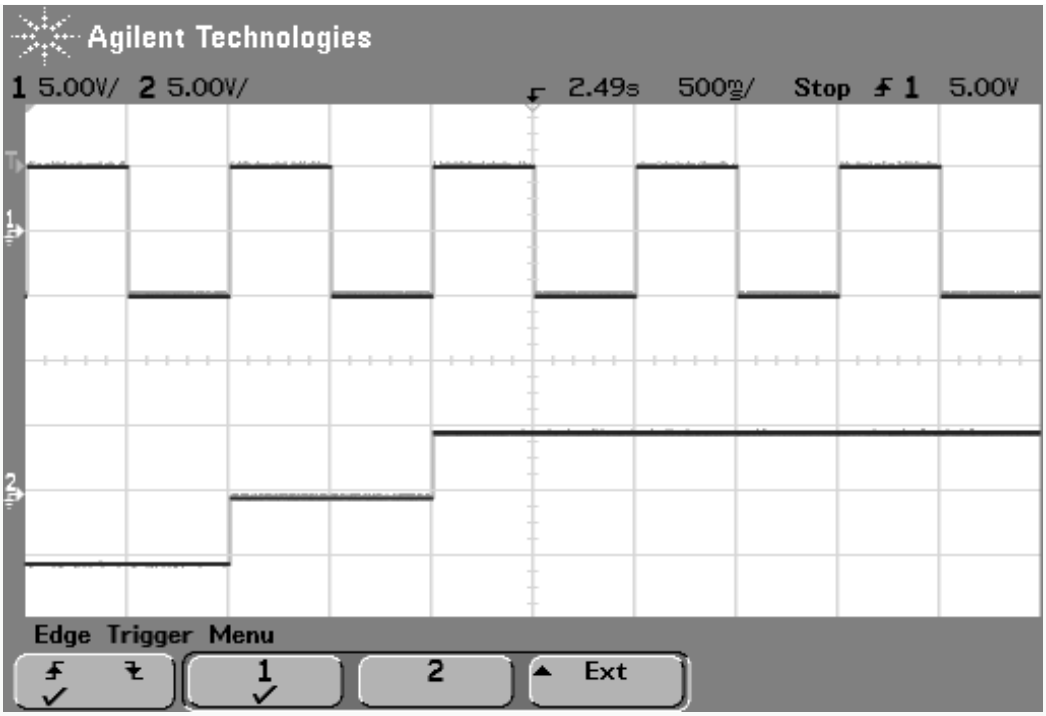


Figure D.60. Third timing diagram for D-type master-slave tri-flop, demonstrating i to 1 transistion (goes through 0 first).

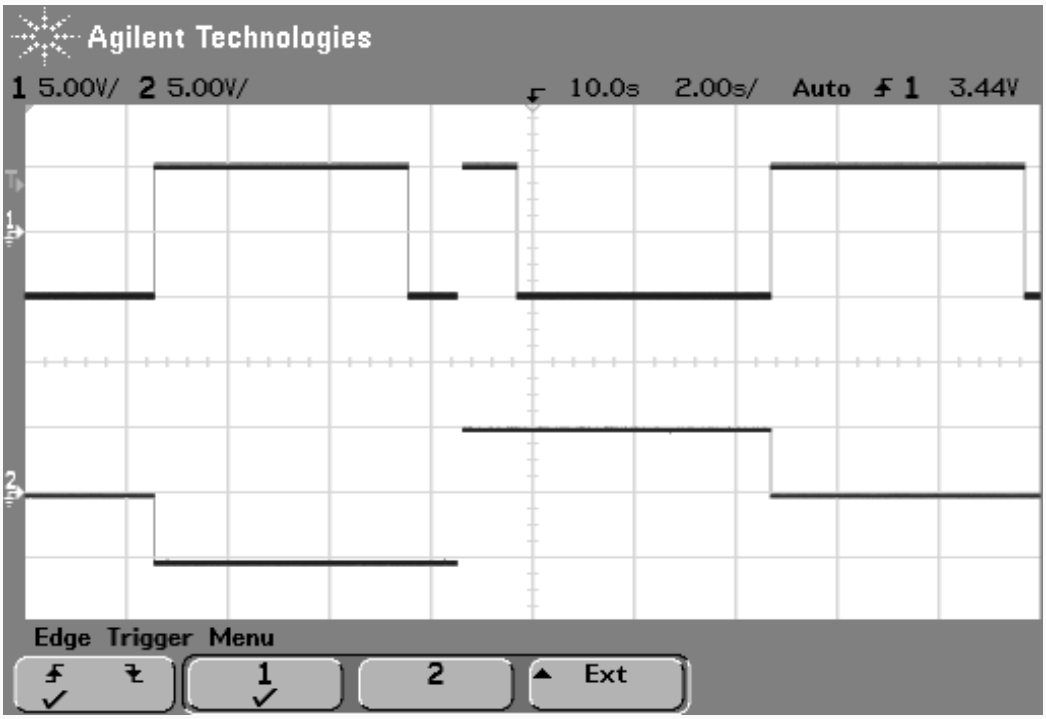


Figure D.61. Fourth timing diagram for D-type master-slave tri-flop, demonstrating 1 to i transistion (goes through 0 first).

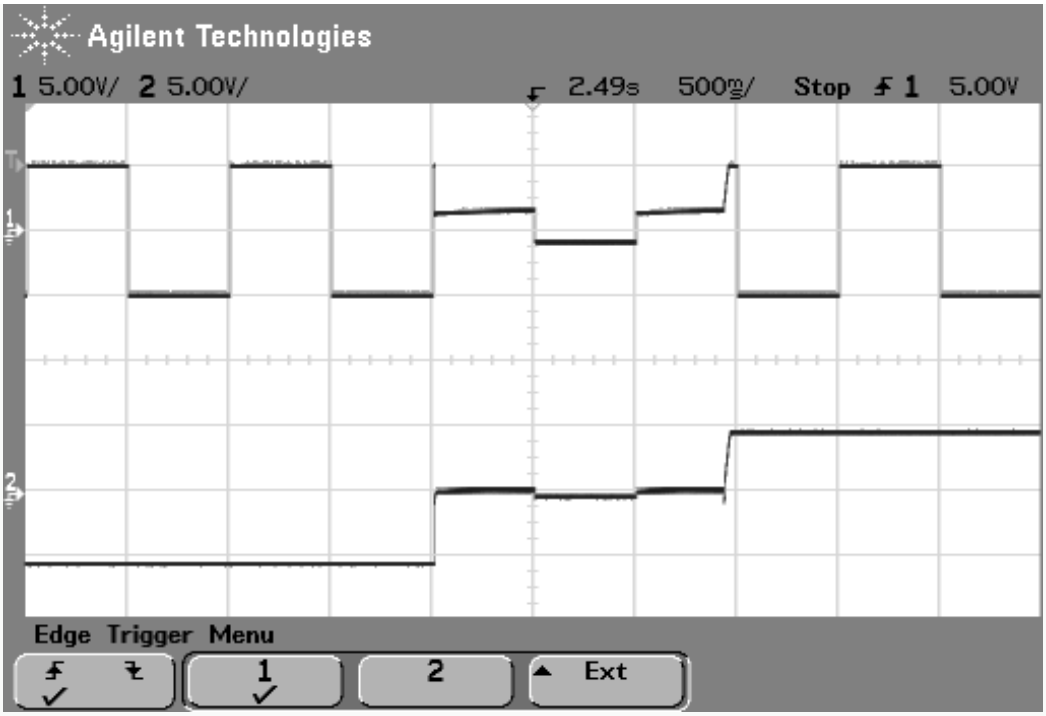


Figure D.62. Fifth timing diagram for D-type master-slave tri-flop, with a make-before-break switch.

When CLK is low or high, Q holds the previous value no matter what D is.

When CLK is zero, Q follows D if D is zero or high but not negative.

This behavior was observed in main.asc when dtflop-ms was used as the program counter:

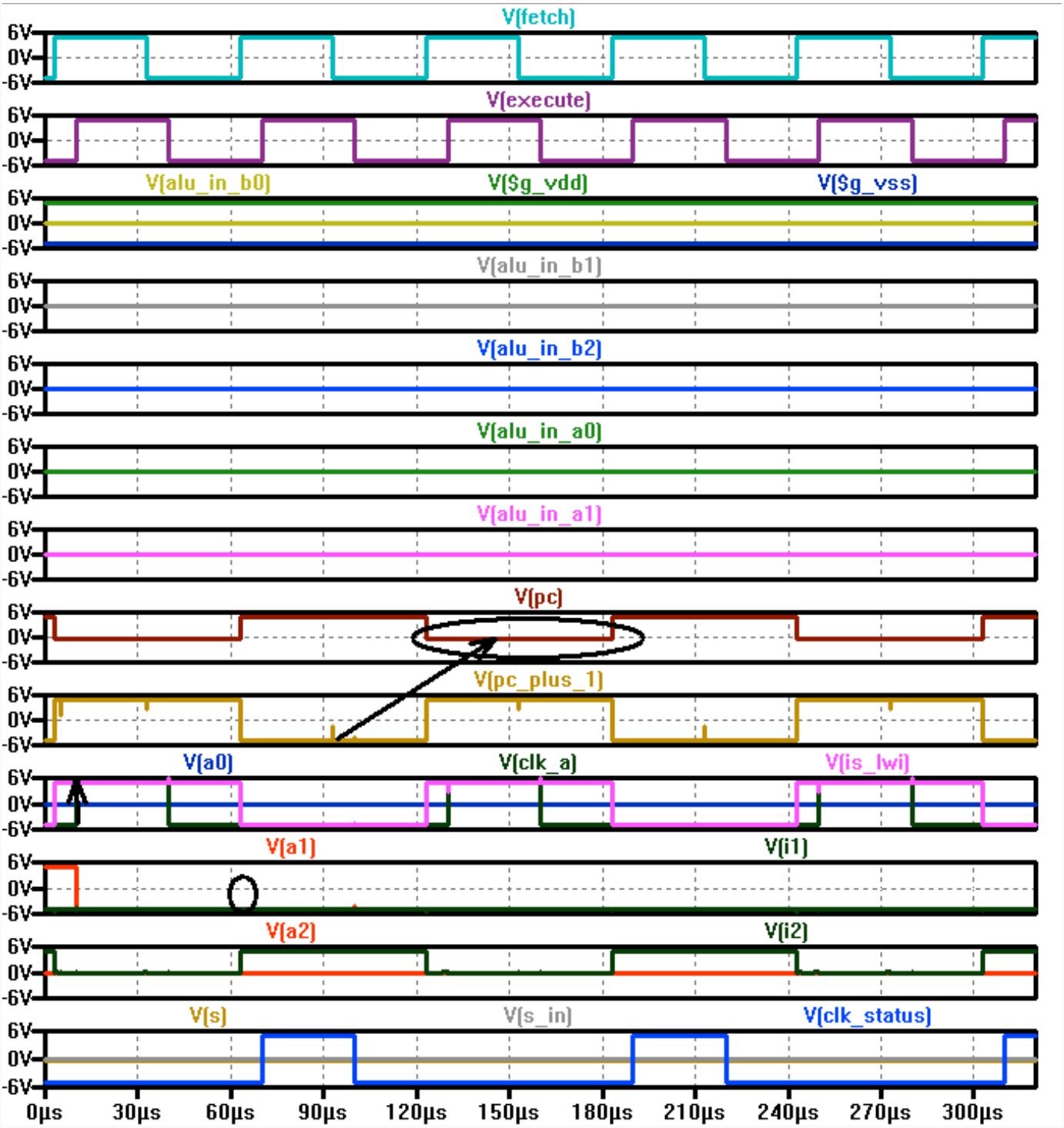


Figure D.63. As observed in lab, dtflop-ms cannot transition between 1 and $\overline{1}$ without first clocking in a 0. This makes PC cycle from 0,1,0,1,0 instead of the desired 0,1,$\overline{1}$,0,1,$\overline{1}$ sequence.

D.5.7. Rising Edge-Triggered Master-Slave D Tri-Flop: Made with 2 Level-Triggered Tri-Flops

A master-slave tri-flop can also be made by connecting two level-triggered as follows (`dtflop-ms2` in git):

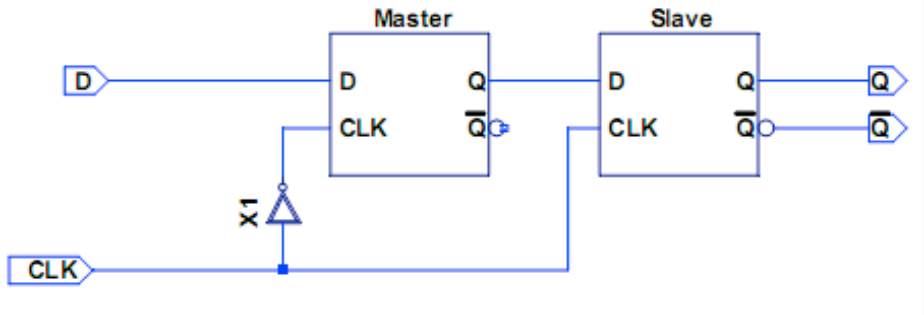


Figure D.64. Master-slave rising edge-triggered D tri-flop.

The input is sampled on the rising edge:

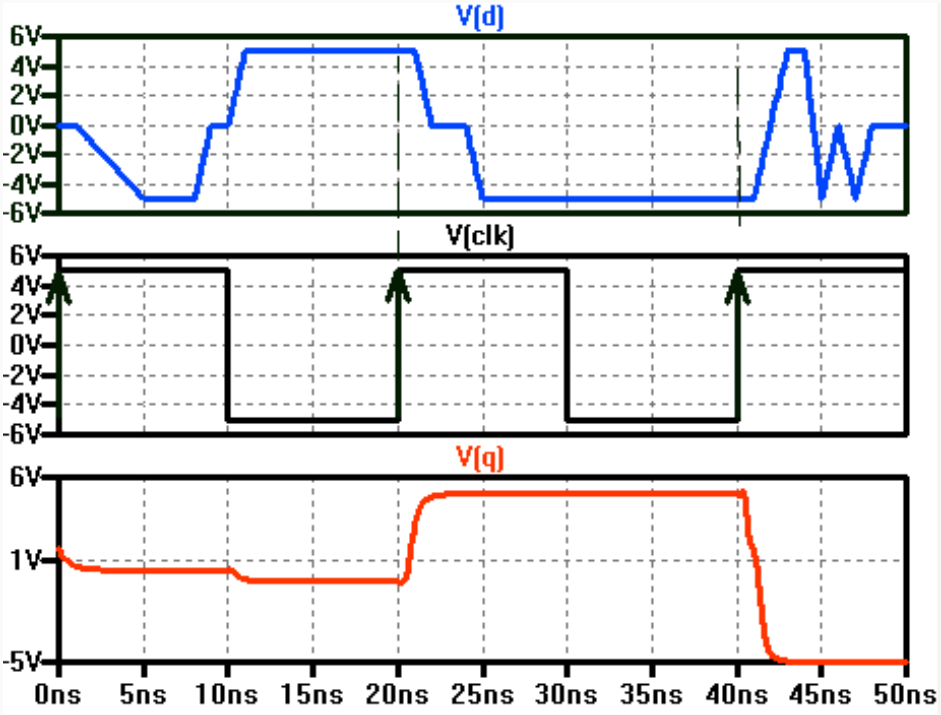


Figure D.65. Timing diagram for master-slave rising edge-triggered D tri-flop.

Registers can be formed with this type of flip-flop. See the trit_reg_3 circuit in the git repository for a 3-trit register, with true and complementary outputs.

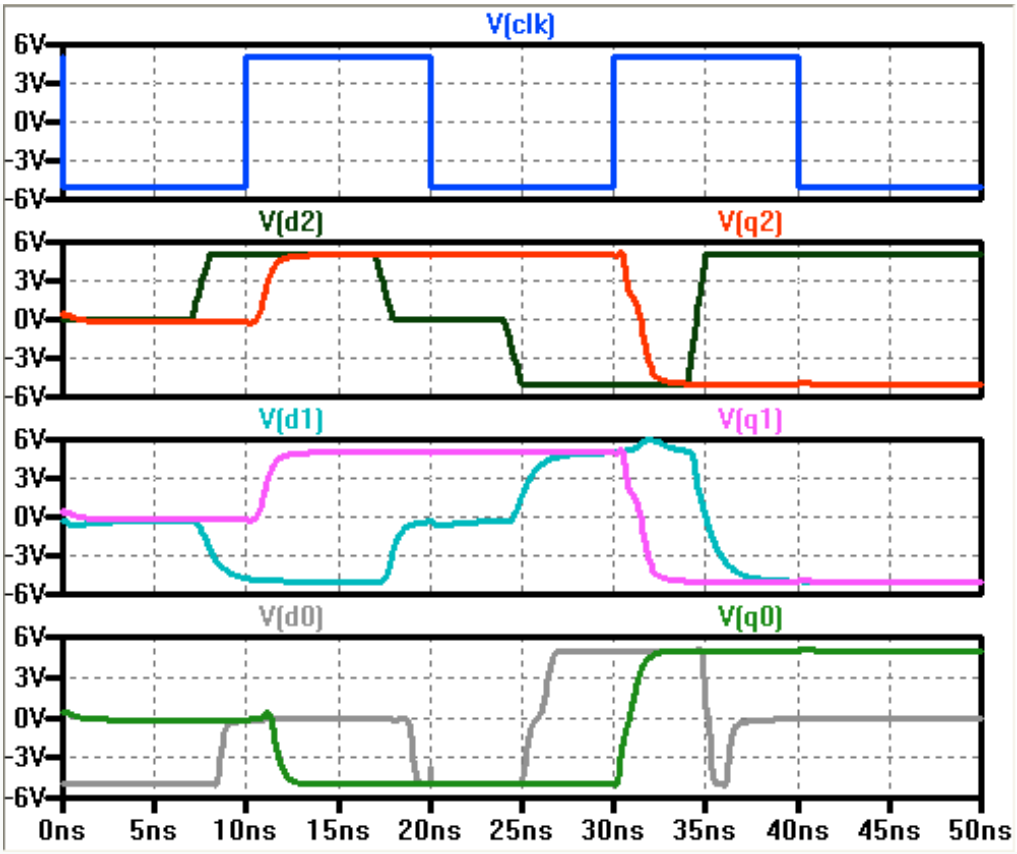


Figure D.66. 3-trit register timing diagram.

This was successfully tested in lab on a PCB, see the section on PCB construction.

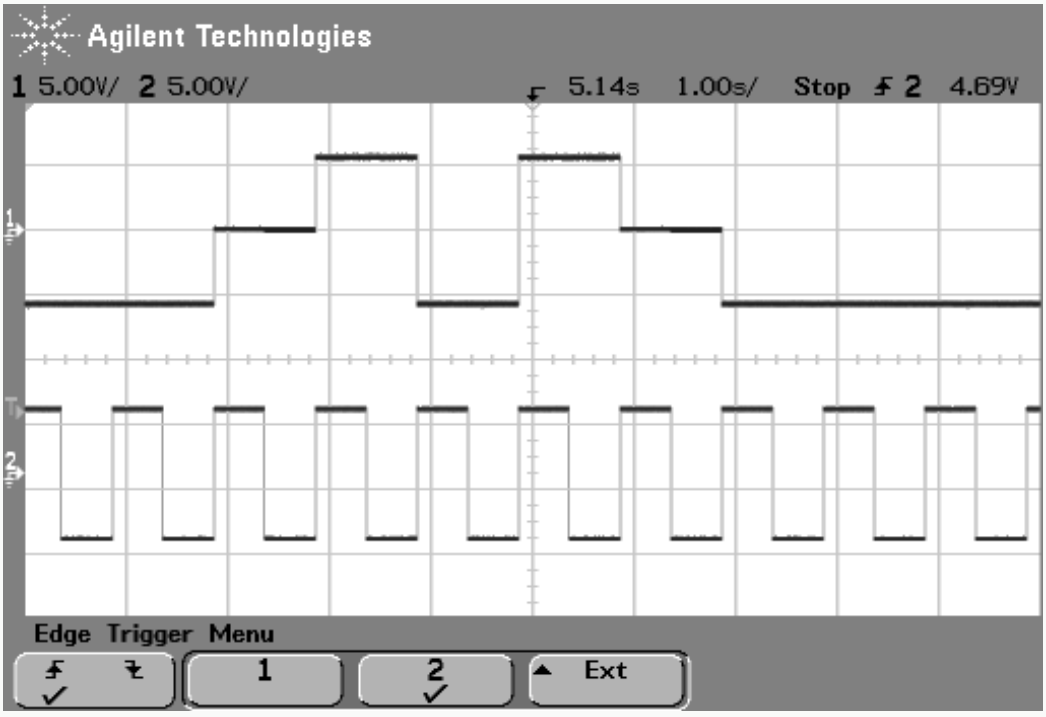


Figure D.67. dtflop-ms2 tested with $\bar{1}$,0,1,$\bar{1}$,1,0,$\bar{1}$.

Other tests showing each of the transitions are:

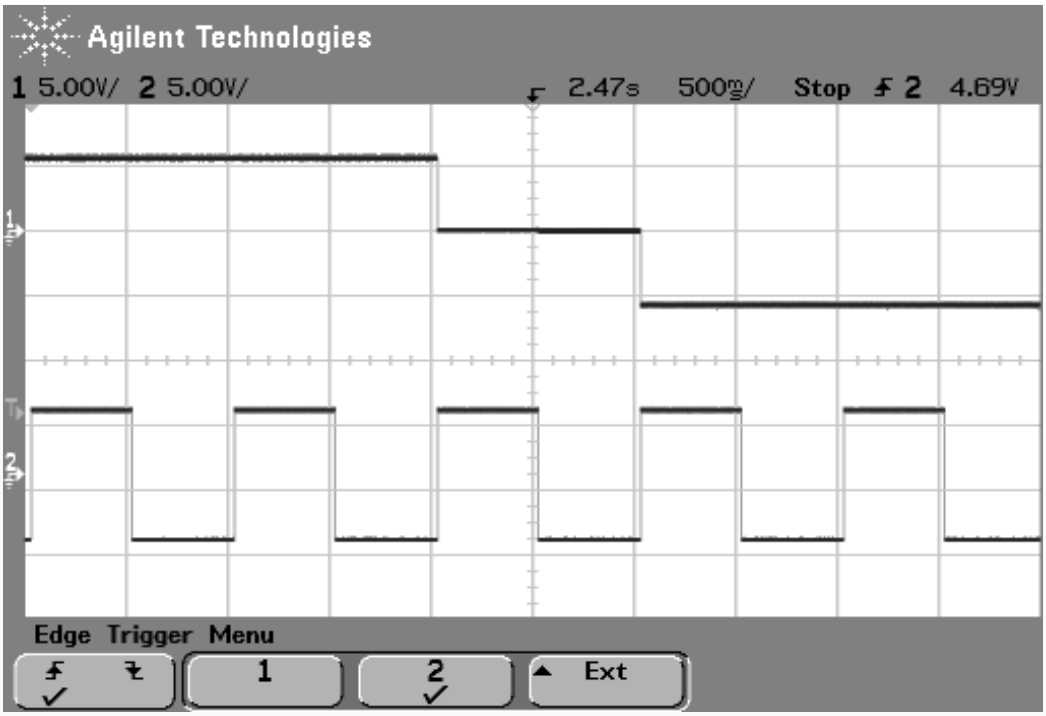


Figure D.68. dtflop-ms2 tested with 1,0,$\overline{1}$

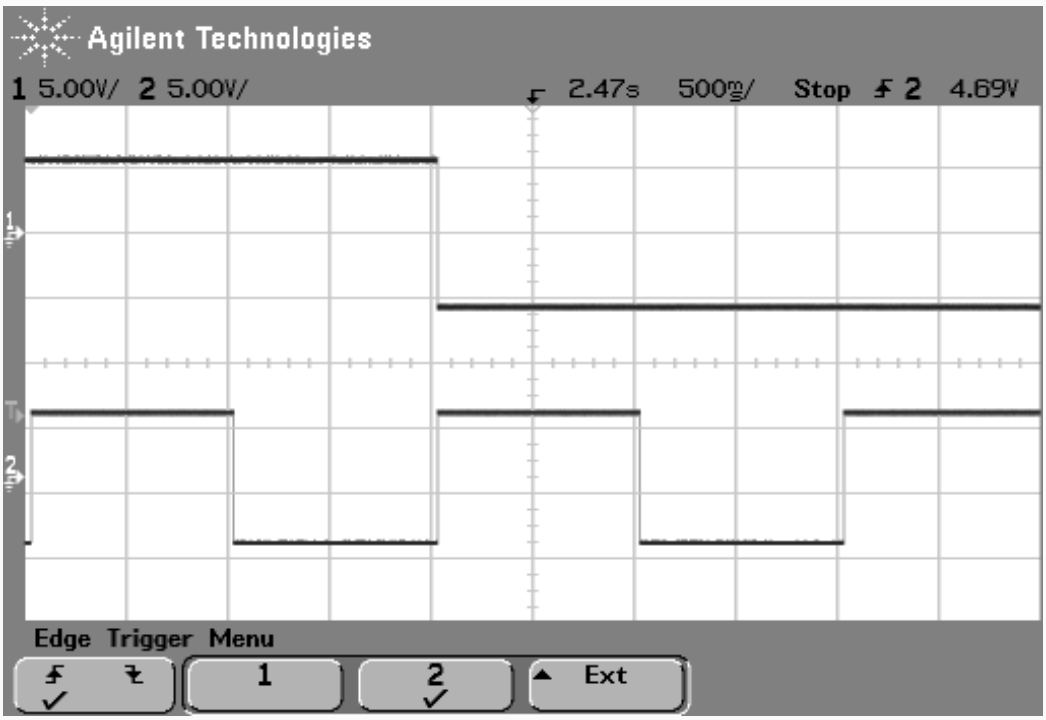


Figure D.69. dtflop-ms2 tested with 1,$\overline{1}$

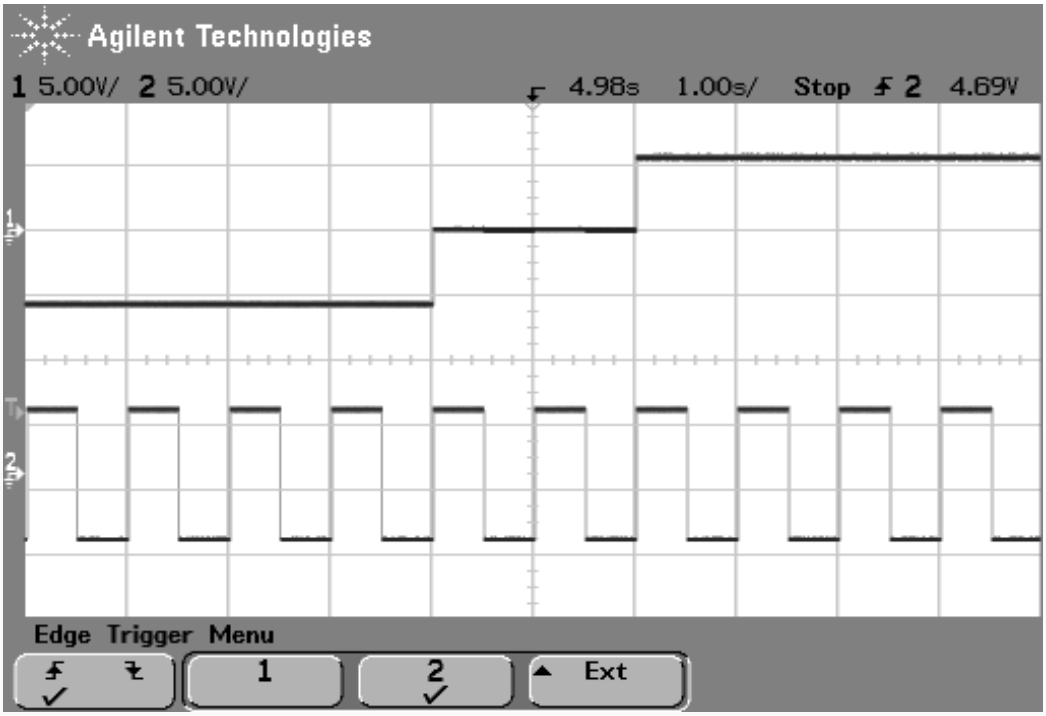


Figure D.70. dtflop-ms2 tested with $\overline{1}$,0,1

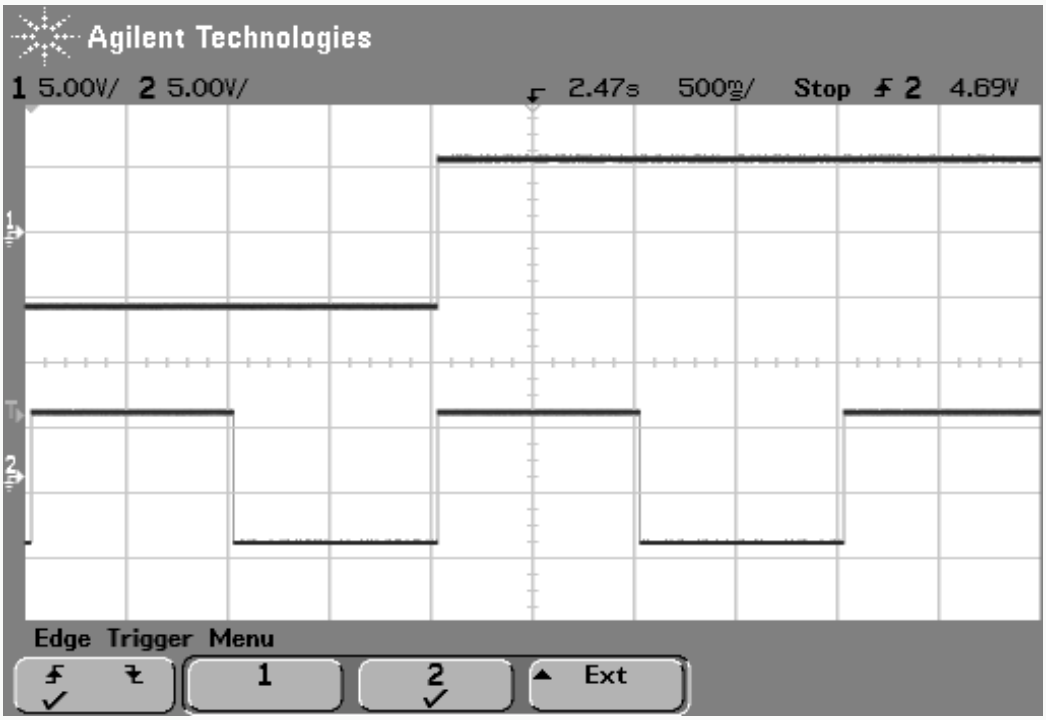


Figure D.71. dtflop-ms2 tested with $\overline{1}$,1

The rising-edge triggered master-slave D-type tri-flop is the most generally useful tri-flop for digital trinary computers, and is the type we used in building our computer system.

D.6. Ring Oscillator

A ring oscillator can be made with STI gates very similarly to as in binary. This could possibly be useful for a clock, but a 555 timer may give better results.

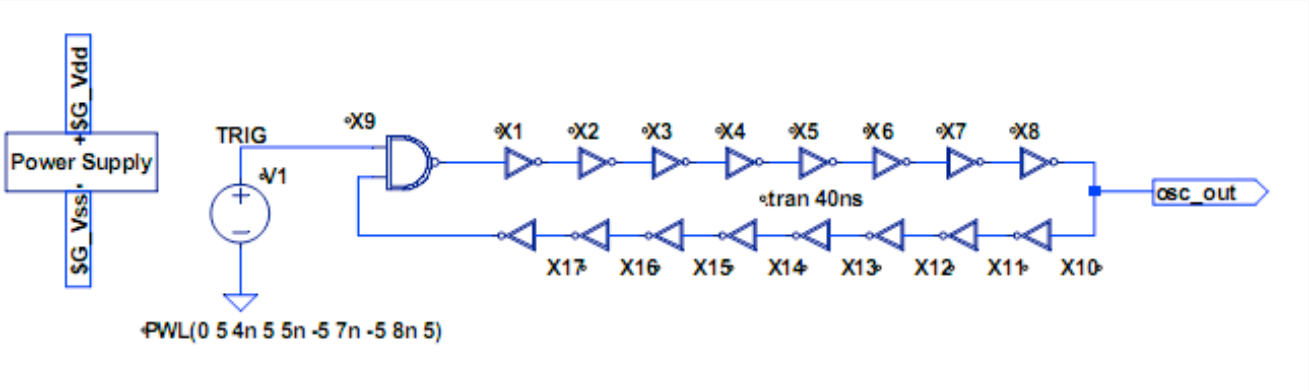


Figure D.72. 17-stage ternary ring oscillator.

The trigger input should normally be +5 V to maintain the transparent inverter action of the TNAND. When it dips to -5 V, the oscillator is initialized to +5 V, and the oscillations begin:

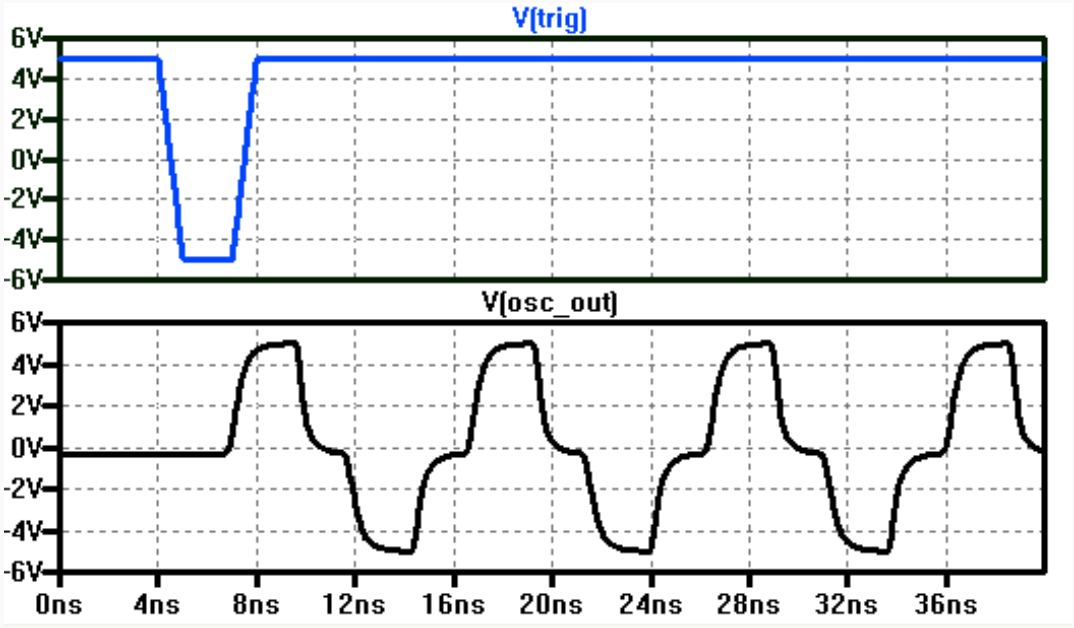


Figure D.73. Transient response of 17-stage ternary oscillator.

While the ring oscillator is useful to determine the maximum switching speed of the ternary CMOS logic family, it is not very useful for precise clocking. For this reason, we did not use a ring oscillator for our clock, instead opting to use a 555 timer circuit as described later.

D.7. 1:3 Decoder

The 1:3 decoder, or J_K circuit, decodes a trinary signal into three signals, one of which is active corresponding to the logic level of the input:

Table D.5. CMOS T-gate Control Input Truth Tables

S	Sa	Sb	Sc
$\overline{1}$	**1**	$\overline{1}$	$\overline{1}$
0	$\overline{1}$	**1**	$\overline{1}$
1	$\overline{1}$	$\overline{1}$	**1**

From this table, the logic functions can be deduced:

- Sa = $1\overline{1}\overline{1}$ = ?S = $S^{\overline{1}}$
- Sb = $\overline{1}1\overline{1}$ = $\Delta(S + \overline{S})$ = S^0
- Sc = $\overline{1}\overline{1}1$ = $\Delta\Delta S$ = S^1

Notice that these functions are the "unary functions commonly used in ternary logic minimization" (see the "many-to-one" functions under the section on trinary logic), hinting at an underlying fundamental nature of these logic functions. Mouftah calls this circuit a "J_K arithmetic circuit", and constructs it like this:

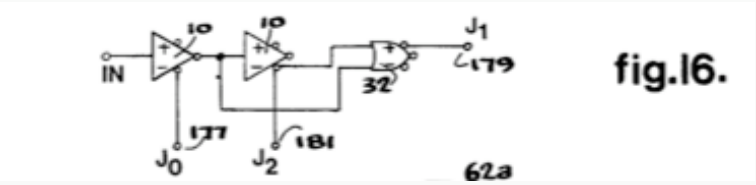


Figure D.74. J_k Arithmetic Circuit, Figure 6 from [137].

Srivastava and Venkatapathy do it like so:

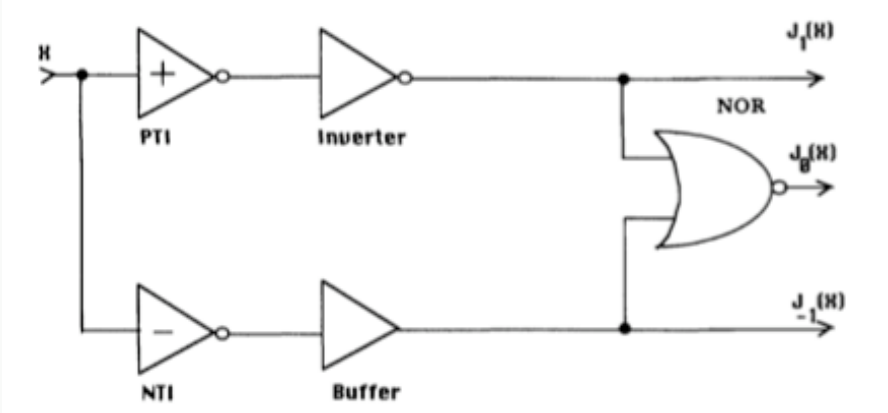


Figure D.75. J_k Arithmetic Circuit, Figure 6 from [137].

It is possible to implement this circuit with more components, to give a better signal. Prior to [3] (http://repo.or.cz/w/trinary.git?a=commit;h=abd3d31ad43fe314ed35c72c90413cf98f00440f) , this is how we did it (old timing diagram: [4] (http://jeff.tk/w/images/archive/f/f3/20080504004917%213-1_Decoder_Timing_Diagram.png) , old decoder: [5] (http://jeff.tk/w/images/archive/e/ef/20080504004020%213-1_Decoder.png)). But now `decoder3-1` is

implemented as Srivastava, except all using trinary circuits. *Note, however, that the inverter and NOR gate in Srivastava's circuit may be binary devices powered by Vdd and Vss, since the output is always 1 or $\overline{1}$.*

This circuit decodes a trit into three signals, which are 1 when the input is a certain value, and $\overline{1}$ otherwise:

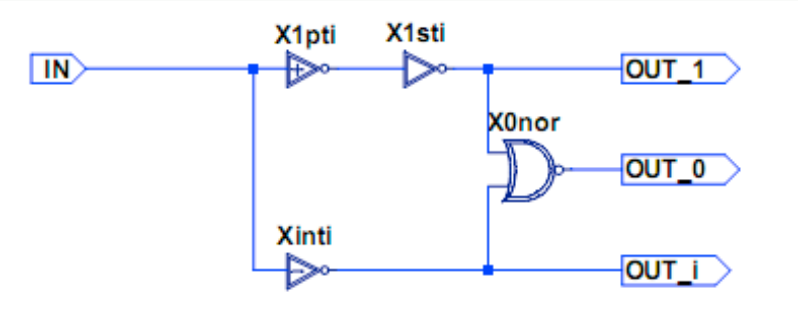


Figure D.76. 3:1 decoder.

Here's an example of how the decoder operates. Note that the labels come from the CPU instruction decoder, but 3:1 decoders can be used in other places, too:

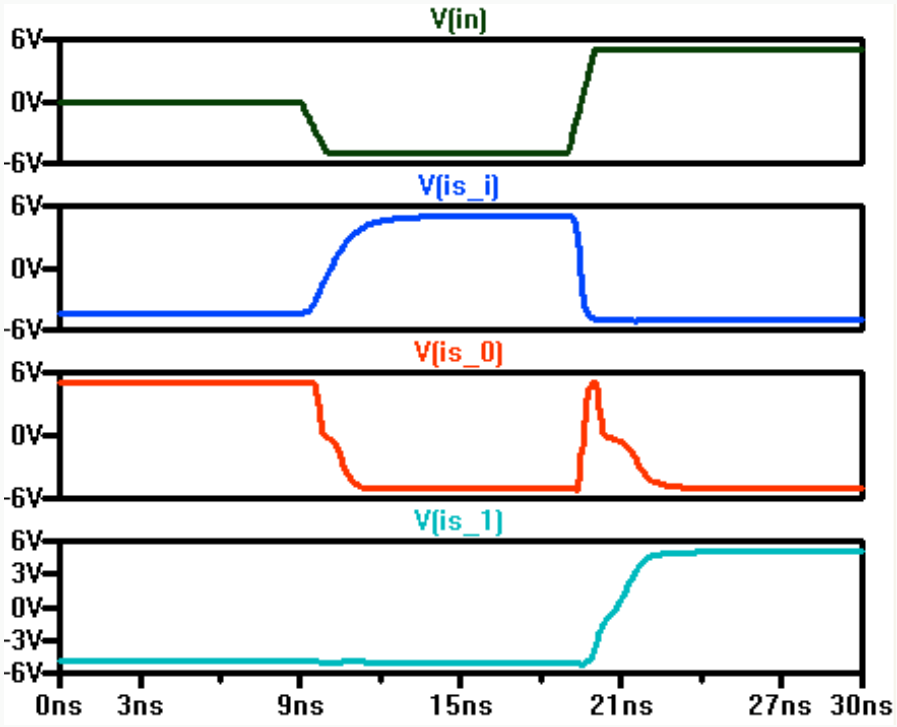


Figure D.77. 3:1 decoder timing diagram example. is_cmp is high when i0 is $\overline{1}$, is_lwi when 0, and is_be when 1

In lab, the circuit operates as expected:

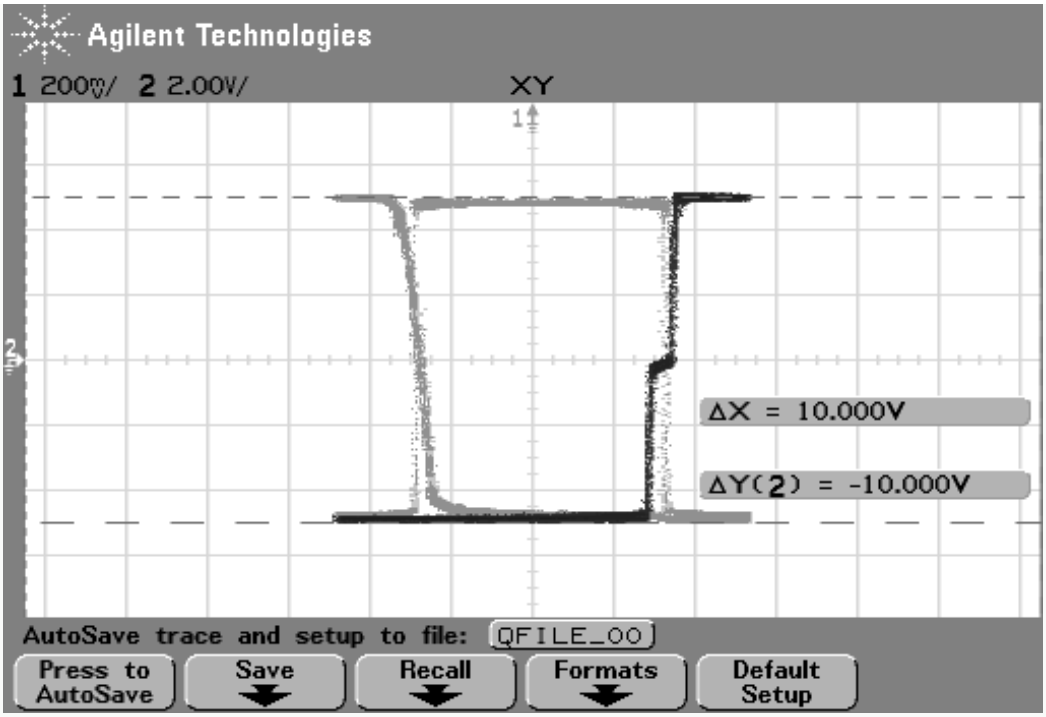


Figure D.77. 3:1 decoder VTC, all outputs. For individual traces, see: 3:1 decoder VTC, i output., 3:1 decoder VTC, 0 output., 3:1 decoder VTC, 1 output..

D.8. 3:1 Multiplexer

Compact truth table for a 3:1 decoder, with inputs A, B, and C that are routed to Q depending on the select input, S:

Table D.6. 3:1 mux

S	Q
$\overline{1}$	A
0	B
1	C

How could this be implemented? Grubb [138] has one way, but I had another idea: use a *CMOS transmission gate* (such as in the CD4016 quad transmission gate) on each of A, B, and C, with the control signals set based on an appropriate unary function of S. Power the CMOS transmission gate with V- and V+, then it can pass any trinary voltage, if the control signal is high, or it will block it (with open drain output) if the control signal is low. The open drain output of the CMOS transmission gate allows inputs to be tied together, traditionally in a "wired-AND". All the T-gate outputs will be tied together to form the final Q output of the multiplexer. The control signals of the T-gates will be set appropriately to enable or disable the appropriate output. Since they are powered by V- to V+ (in order to be able to pass all voltages), a V- control signal (logic $\overline{1}$) turns off the gate, and V+ (logic 1) turns it on. The control signals for the transmission gates that control each signal Sa, Sb, and Sc, for inputs A, B, and C, come from a 3:1 decoder:

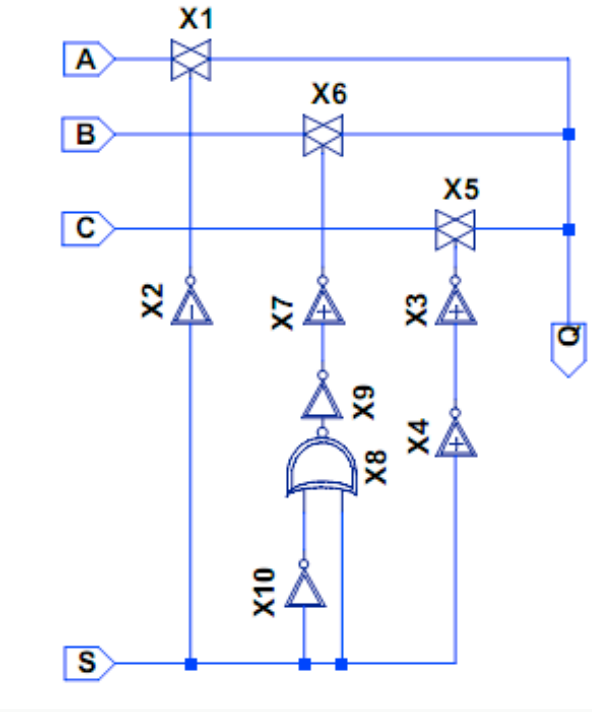


Figure D.78. 3:1 multiplexer schematic diagram. The logic gates are the 3:1 decoder, which is abstracted away as a black box in the actual schematic.

The multiplexer can pass analog signals, since it merely activates CMOS transmission gates. Here is an example of selecting between three separate signals:

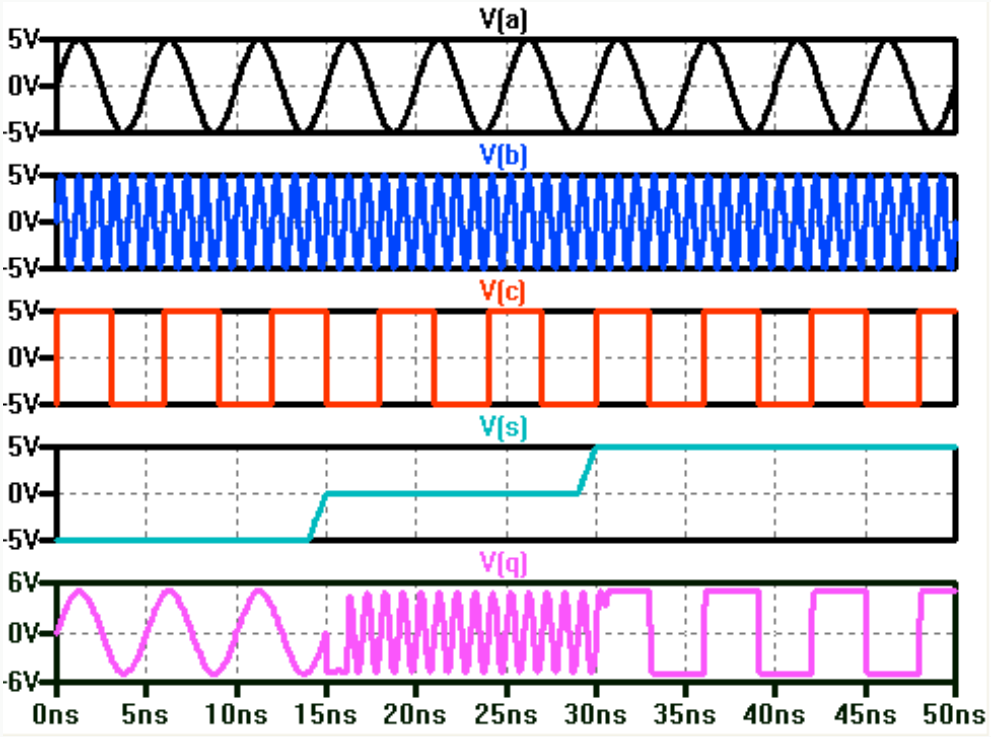


Figure D.79. 3:1 multiplexer timing diagram example.

When S is $\overline{1}$, the low-frequency sine wave, labeled A, is selected. S then goes to 0 and the high-frequency sine wave, B, is selected; lastly, S goes 1 and the square wave in C is selected. Our digital computer will more often

use the multiplexer as a digital device, but multiplexing analog signals serves as a better example.

D.8.1. 3:1 Multiplexer Tested on Breadboard

We first built the 3:1 multiplexer on a breadboard to test it:

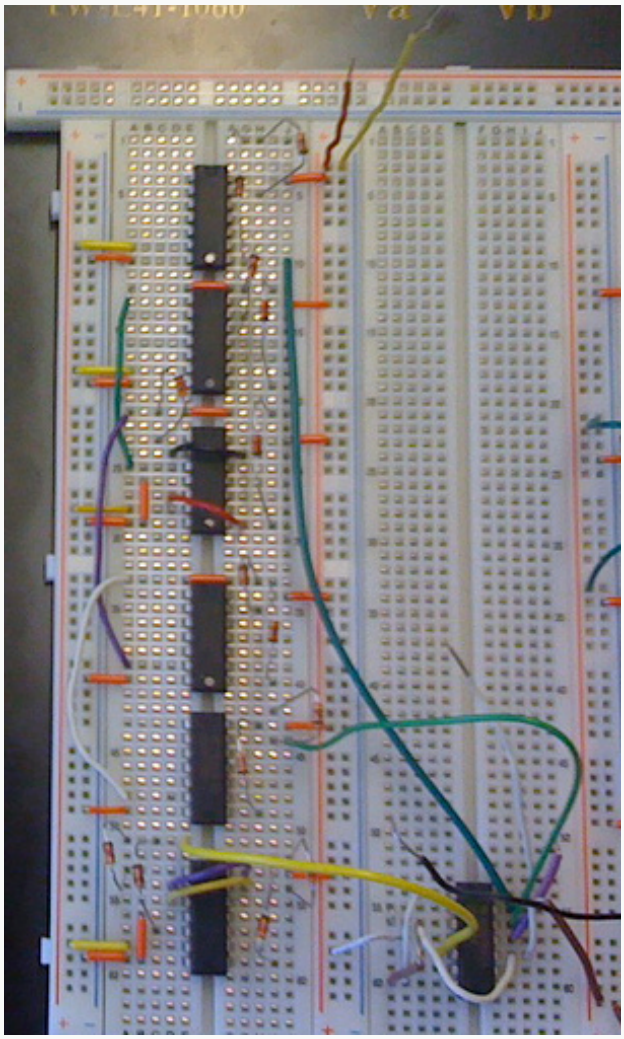


Figure D.80. 3:1 multiplexer on a breadboard.

The 3:1 multiplexer built with CD4016 transmission gates and a 3:1 decoder (above), works well with the select signal sweeping from -5 V to +5 V at 0-100 Hz. The $\overline{1}$, 0, and 1 inputs are connected to logic levels $\overline{1}$, 0, and 1 respectively.

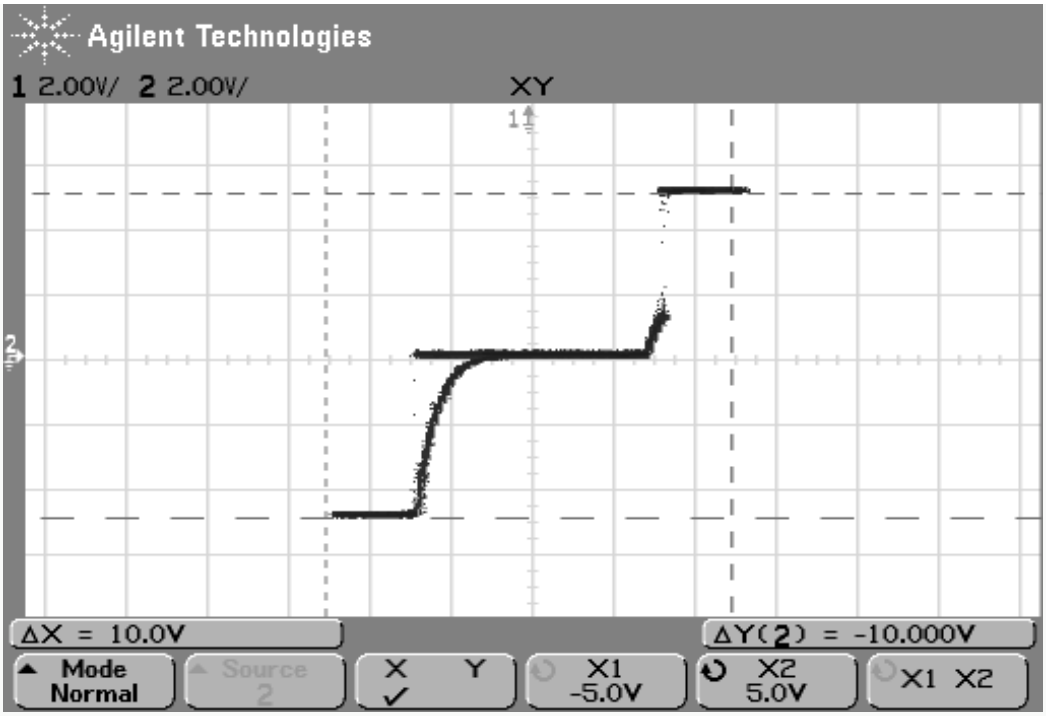


Figure D.81. 3:1 multiplexer VTC, with select input from -5 to 5 V, triangle wave, changing at 75 Hz.

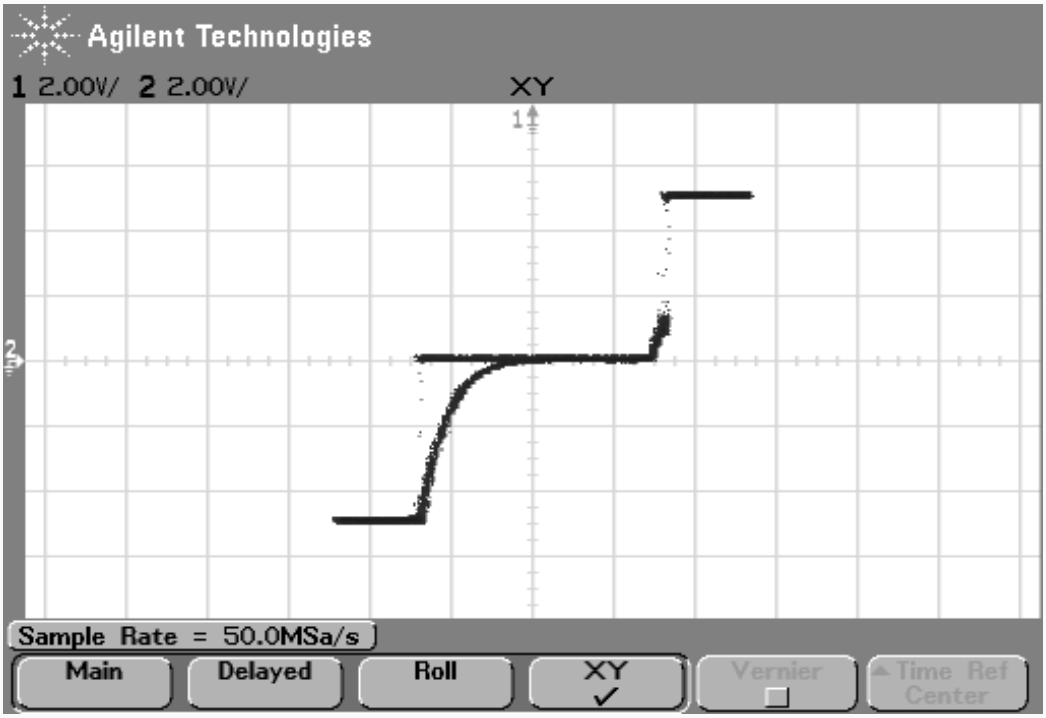


Figure D.82. 3:1 multiplexer VTC, with select input from -5 to 5 V, triangle wave, changing at 100 Hz.

At 250 Hz, the hysteresis is fairly extreme:

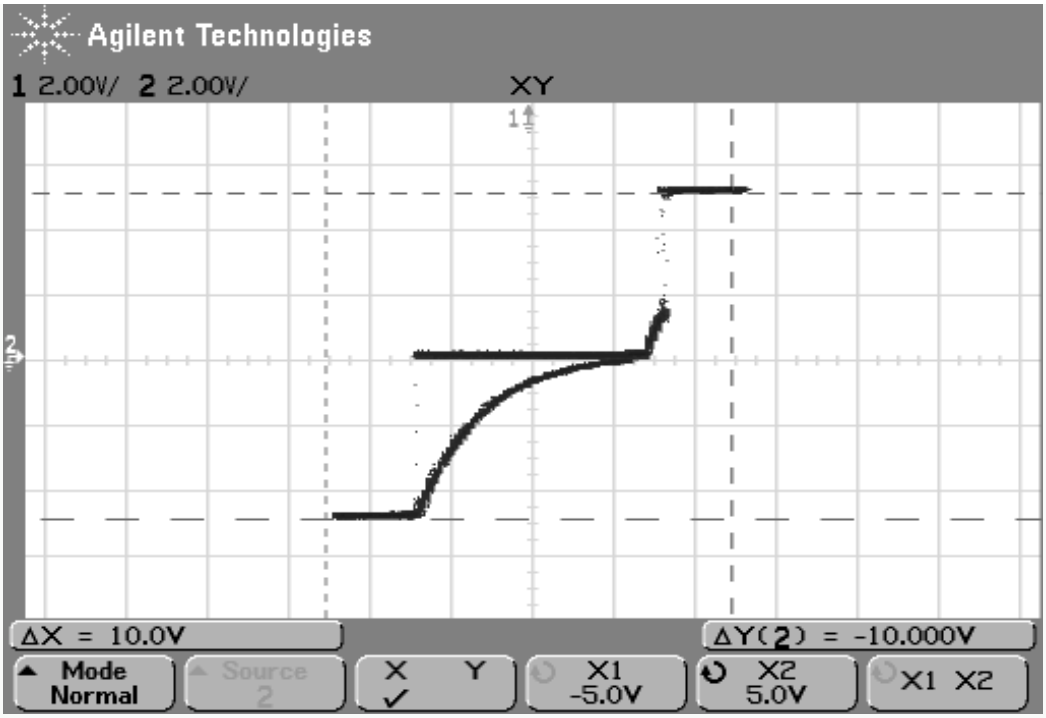


Figure D.83. 3:1 multiplexer VTC, with select input from -5 to 5 V, triangle wave, changing at 250 Hz.

The multiplexer switches sharply on the falling edge, but when changes slowly when rising, as seen at 100 Hz and even worse at 1 kHz:

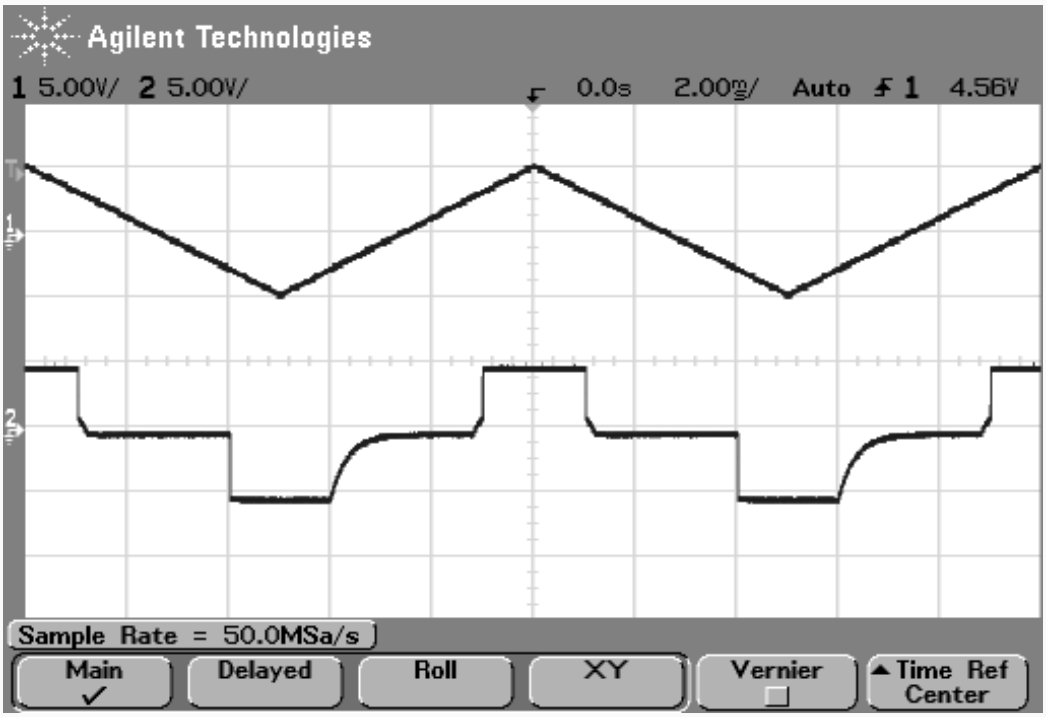


Figure D.84. 3:1 multiplexer time domain, with select input from -5 to 5 V, triangle wave, changing at 100 Hz.

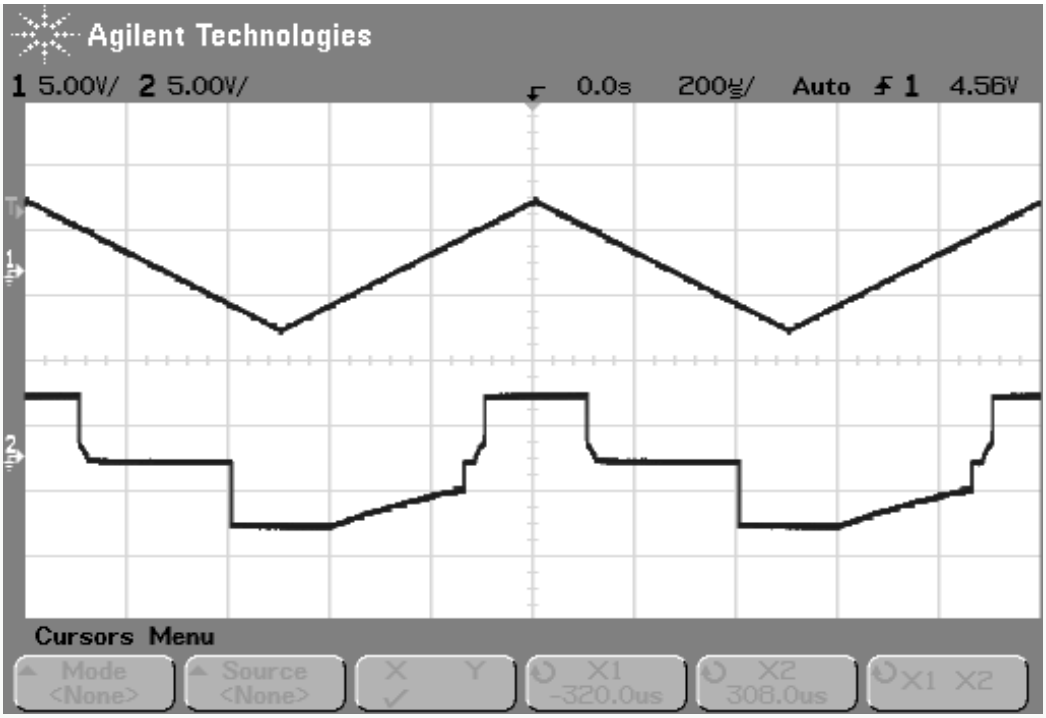


Figure D.85. 3:1 multiplexer in the time domain, with select input from -5 to 5 V, triangle wave, changing at 1 kHz.

Back in the voltage-transfer-characteristic domain, the rising edge does not fully reach the 0 logic level at 1 kHz:

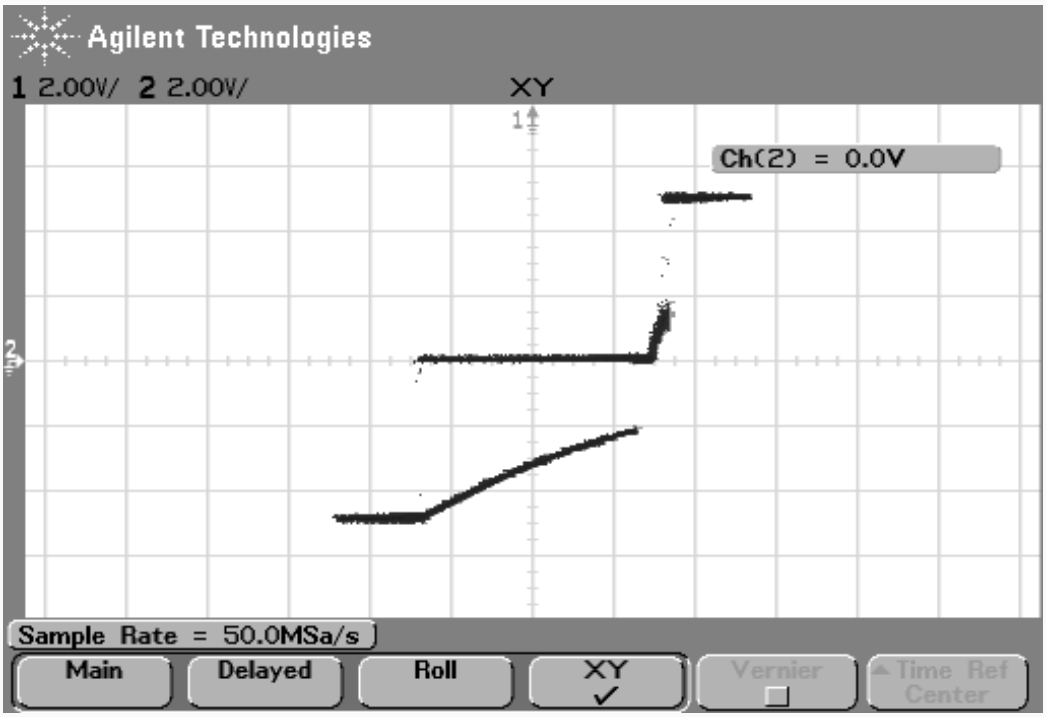


Figure D.86. 3:1 multiplexer VTC, with select input from -5 to 5 V, triangle wave, changing at 1 kHz.

At 100 kHz, the rising edge skips the 0 logic level completely:

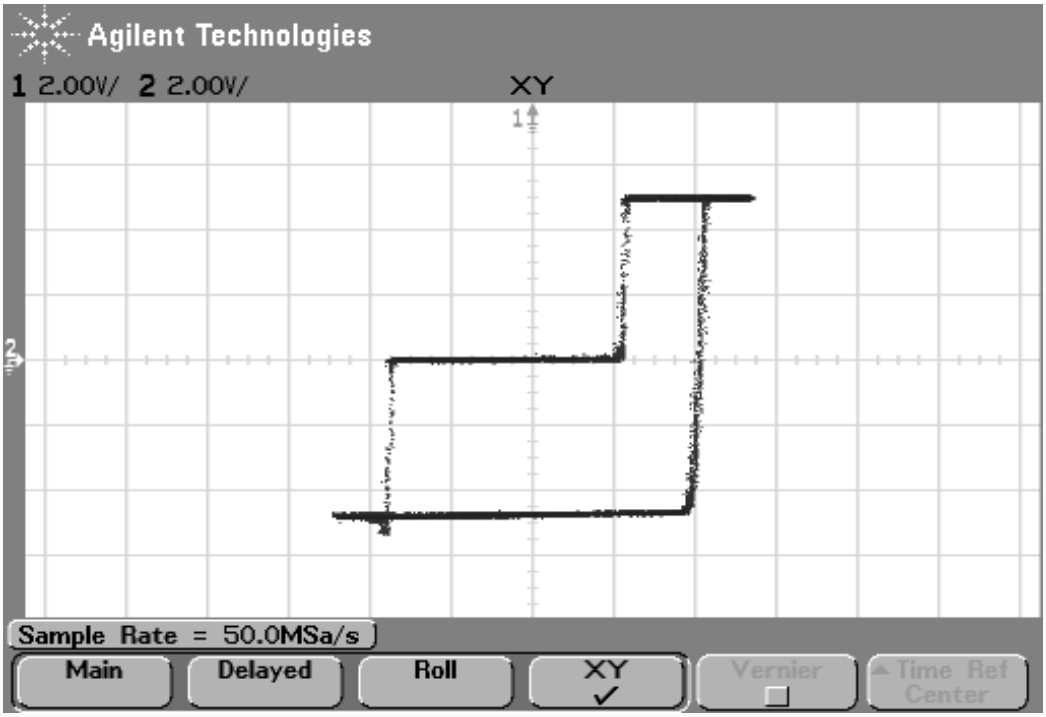


Figure D.87. 3:1 multiplexer VTC, with select input from -5 to 5 V, triangle wave, changing at 100 kHz.

The multiplexer is non-functional. At 1 MHz, the VTC degrades further:

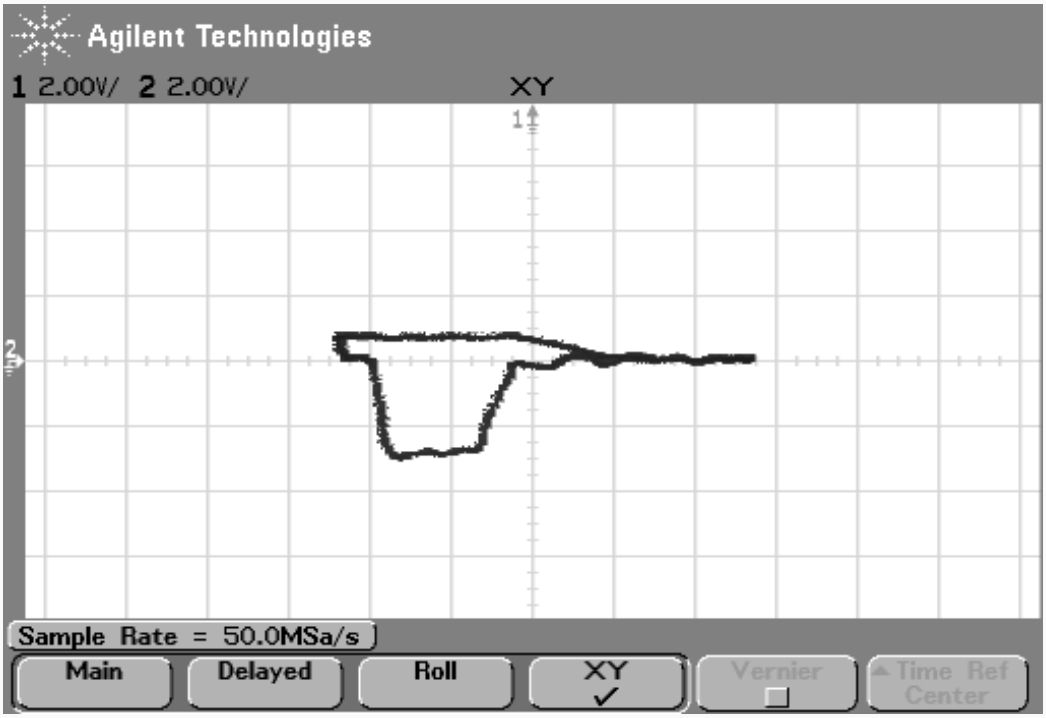


Figure D.88. 3:1 multiplexer VTC, with select input from -5 to 5 V, triangle wave, changing at 1 MHz.

This multiplexer design may be improved by replacing the CD4016 transmission gates with faster chips, but at frequencies below about 100 Hz, it works fine. Advanced techniques would need to be investigated for higher-speed multiplexers, but as the trinary computer system we are building is merely a proof of concept, the performance of this circuit is adequate for our purposes.

D.8.2. 3:1 Multiplexer Tested on PCB

We also tested the mux on a PCB, which will be discussed further in later sections. The results match that of the breadboard:

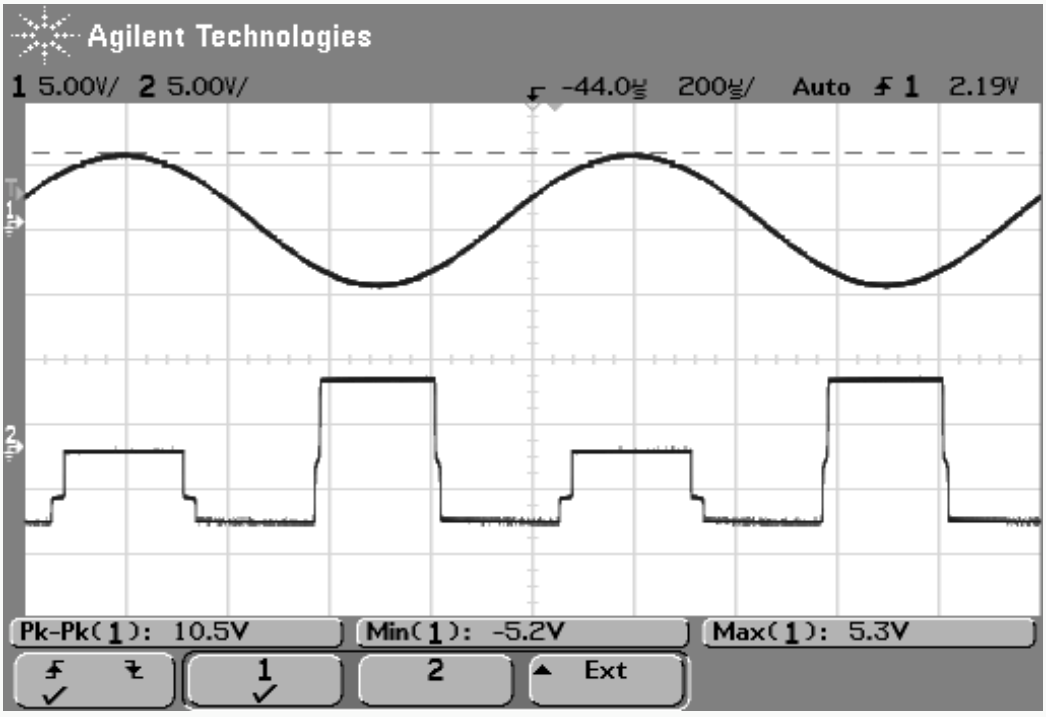


Figure D.89. 3:1 multiplexer tested on PCB, with inputs 1, $\overline{1}$, 0. Select signal on top, output on bottom.

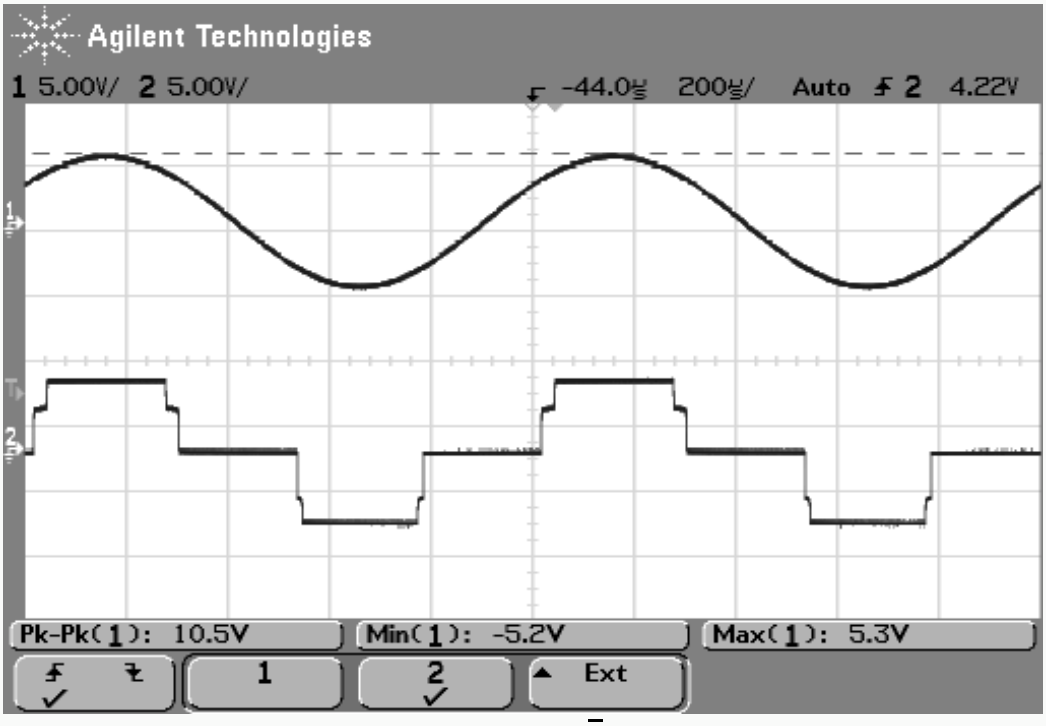


Figure D.90. 3:1 multiplexer tested on PCB, with inputs $\overline{1}$, 0, 1. Select signal on top, output on bottom.

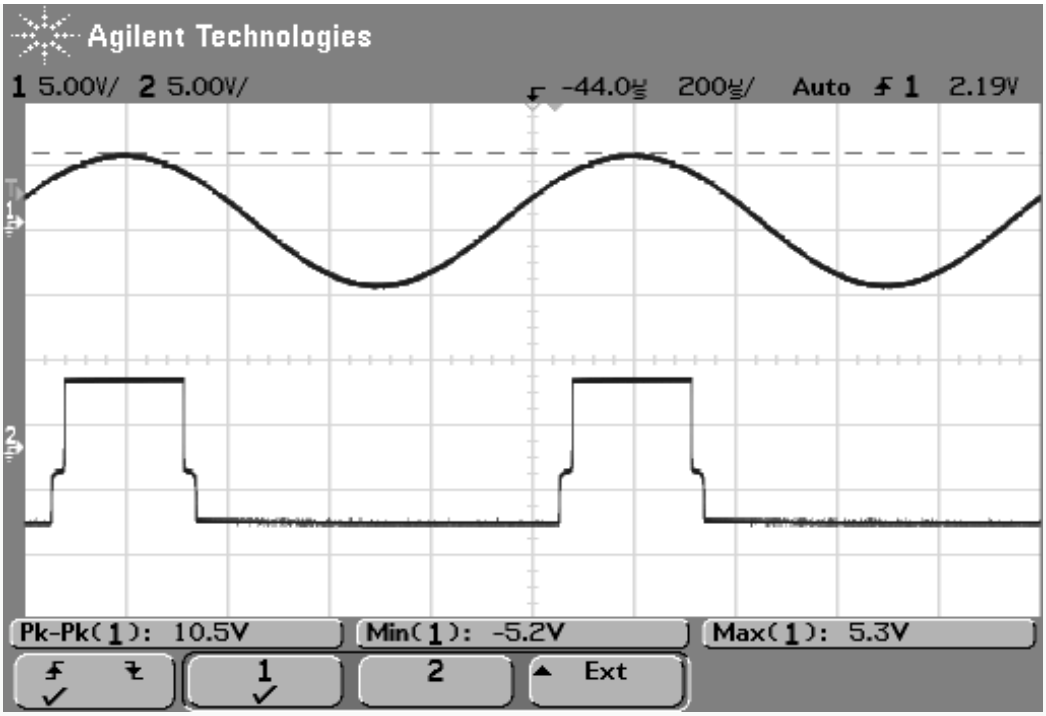


Figure D.91. 3:1 multiplexer tested on PCB, with inputs $\overline{1}$, $\overline{1}$, 1. Select signal on top, output on bottom.

D.9. Untested Circuits

These circuits have not yet been simulated, but are included here for completeness.

D.9.1. Trinary to Binary Converter

US Patent #5498980 *Ternary/binary converter circuit*[139] describes a trinary to binary conversion circuit.

D.9.2. AND-OR-INVERT

This type of gate is often used to implement complex functions. Mouftah defines two designs:

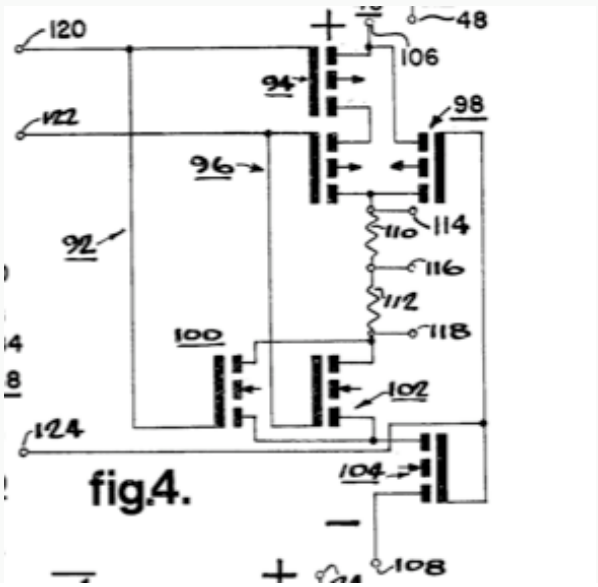


Figure D.92. Mouftah Figure 4: Ternary AND-OR_INVERT, design #1, from Image:Mouftah - Ternary Gate Figures.pdf

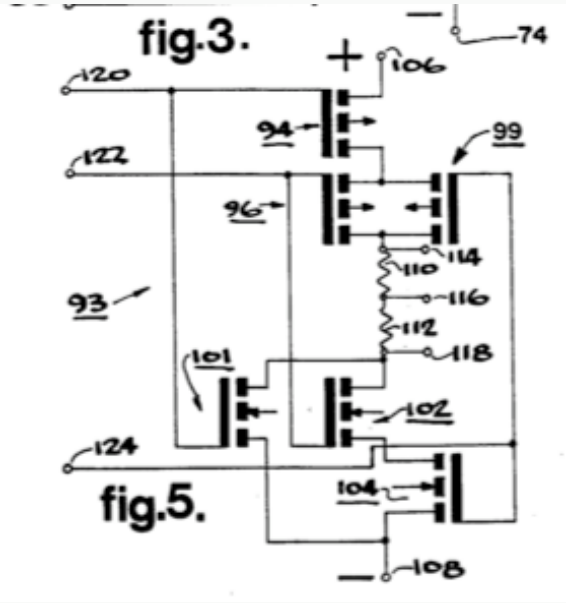


Figure D.93. Mouftah Figure 5: Ternary AND-OR_INVERT, design #2, from Image:Mouftah - Ternary Gate Figures.pdf

There is also an Electronic Letters article on this circuit[140]. Such a circuit would be useful for efficiently implementing advanced logic circuitry, but we did not use it for our computer.

D.9.3. Counters

Mouftah defines divide-by-M counters:

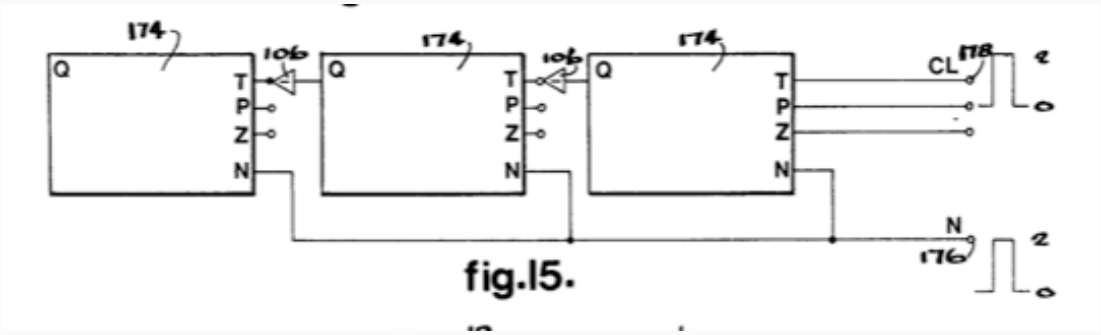


Figure D.94. Mouftah Figure 15: Ternary Counter, counts -13 to +13 or 0 to 27, from Image:Mouftah - Ternary Gate Figures.pdf

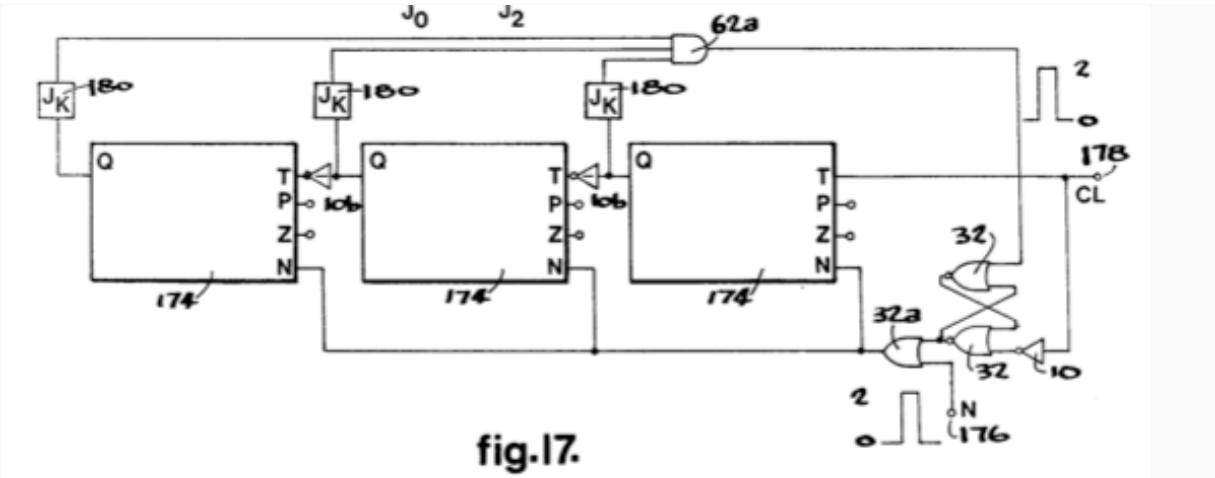


fig.17.

Figure D.95. Mouftah Figure 17: Divide-by-M Counter, from Image:Mouftah - Ternary Gate Figures.pdf

These counters would be useful for trinary computer architectures with higher trits per word, but for our trinary computer, a 1-trit cycle up gate was sufficient.

D.9.4. Memory

Mouftah defines several other memory elements, including a ternary shift register, storage cell, and ROM:

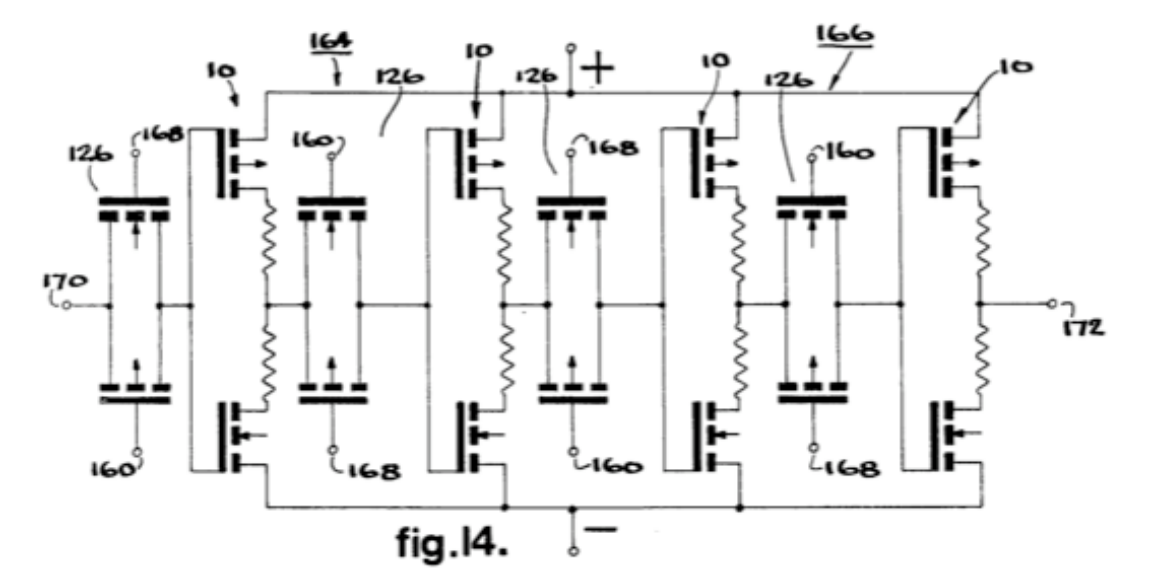


fig.14.

Figure D.96. Mouftah Figure 14: Dynamic Ternary Shift Register, from Image:Mouftah - Ternary Gate Figures.pdf

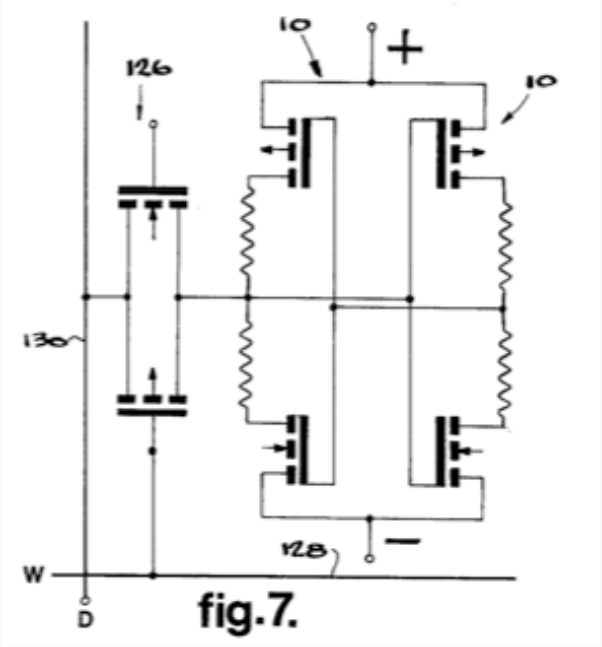


fig.7.

Figure D.97. Mouftah Figure 7: Ternary Storage Cell from, from Image:Mouftah - Ternary Gate Figures.pdf

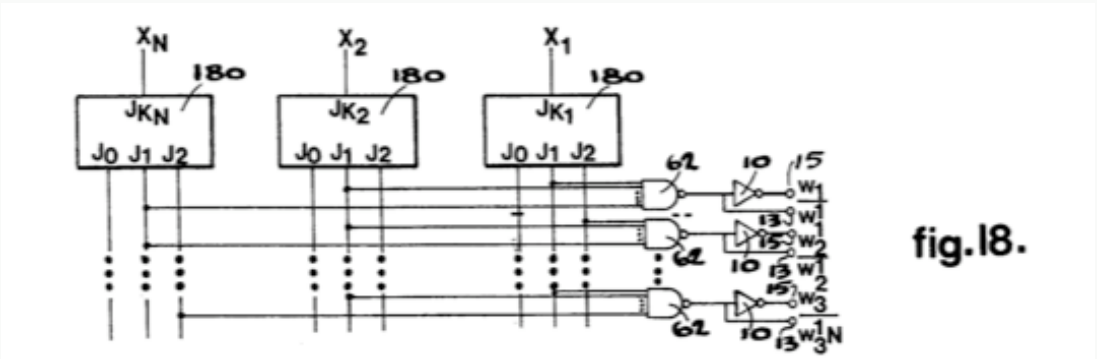


fig.18.

Figure D.98. Mouftah Figure 18: Ternary Decoder, N to 3^N, from Image:Mouftah - Ternary Gate Figures.pdf

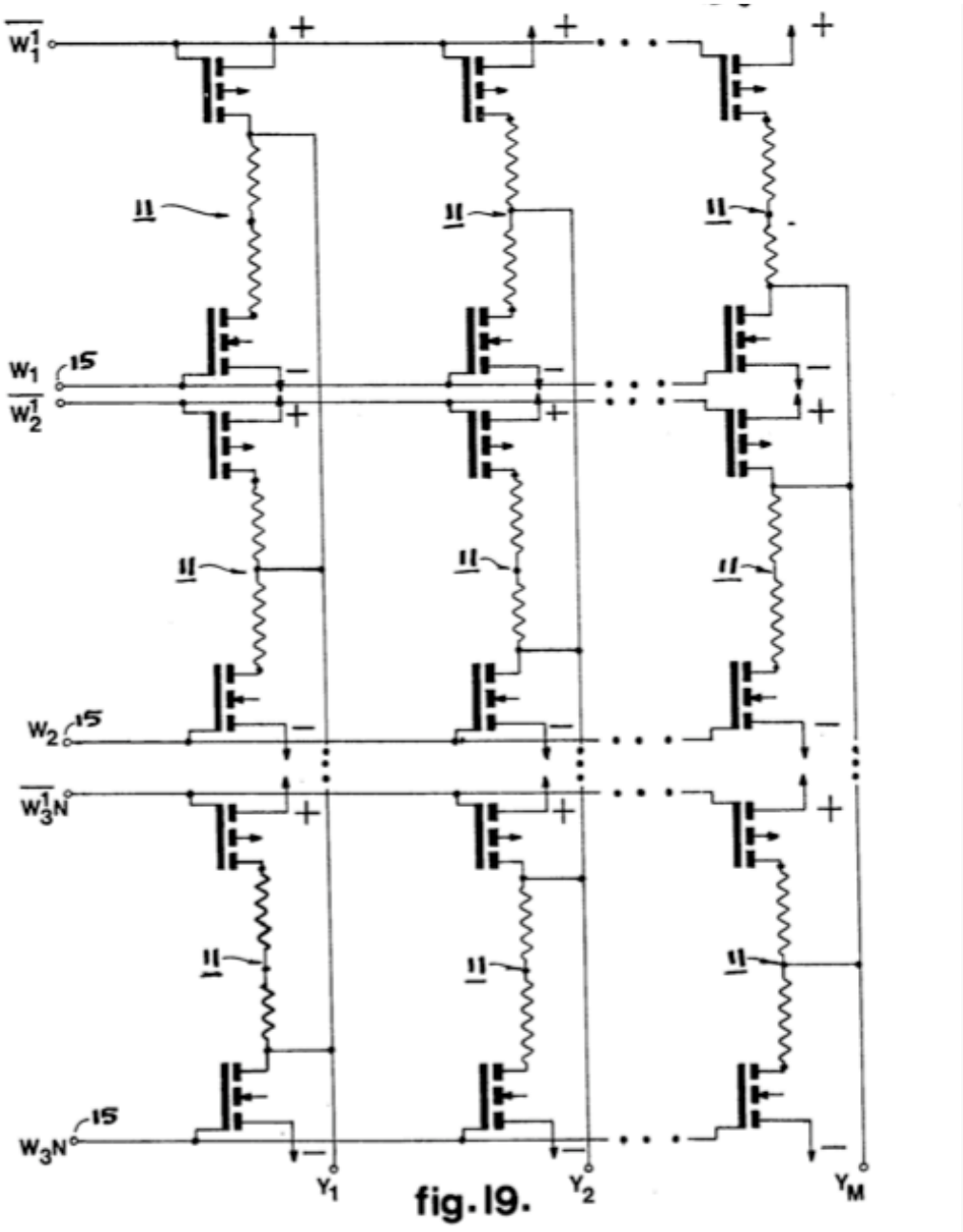


Figure D.99. Mouftah Figure 19: Ternary ROM Encoder, 3^N to M, from Image:Mouftah - Ternary Gate Figures.pdf

We did use any of these circuits in our system, preferring instead to build RAM out of tri-flops and ROM out of manual switches with an address decoder.

D.9.5. Other

Mouftah defined a handful of additional circuits:

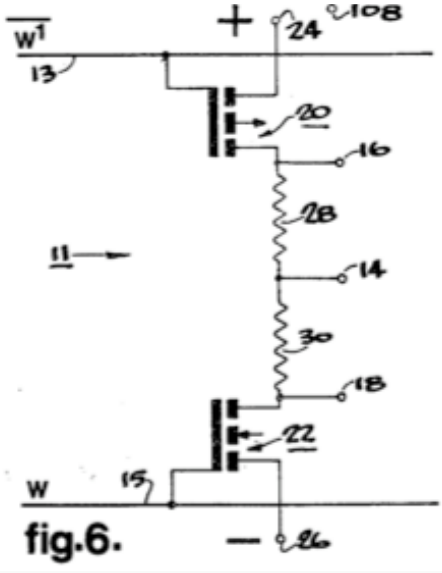


Figure D.100. Mouftah Figure 6: Modified Ternary Inverter (MTI) for use in memory, from Image:Mouftah - Ternary Gate Figures.pdf

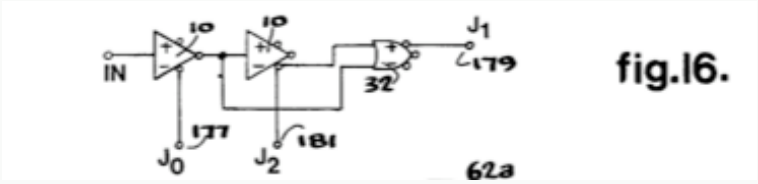


Figure D.101. Mouftah Figure 16: J-K Arithmetic Circuit, from Image:Mouftah - Ternary Gate Figures.pdf

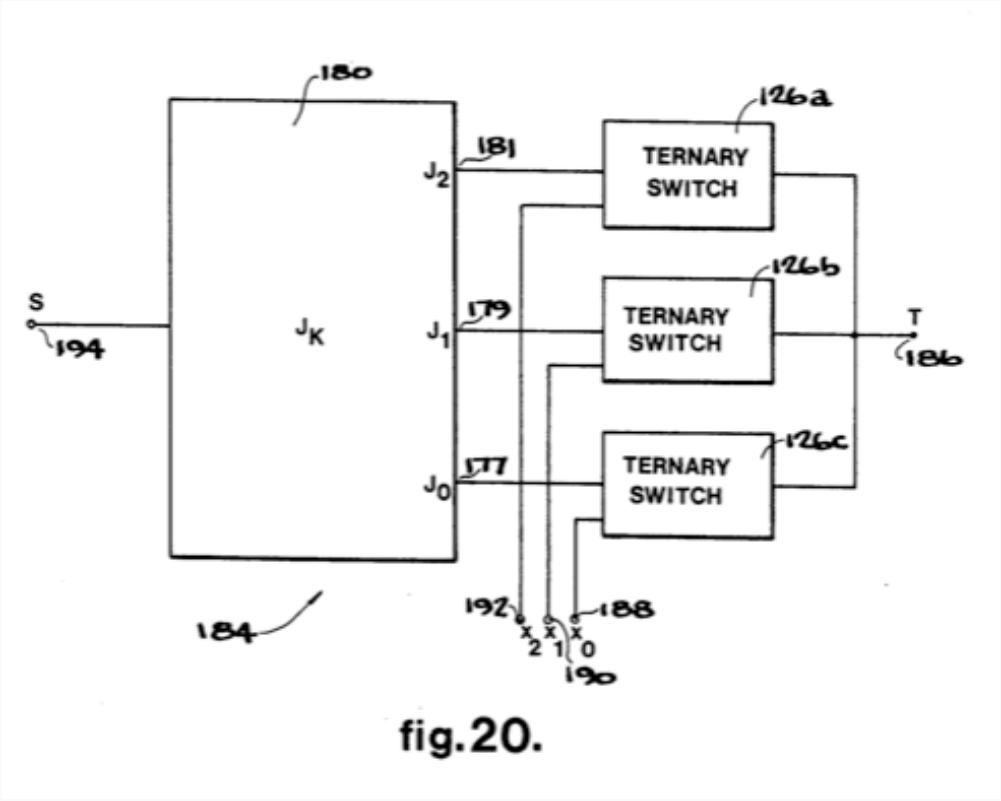


Figure D.102. Mouftah Figure 20: T-gate, from Image:Mouftah - Ternary Gate Figures.pdf

The J_K arithmetic circuit is what is referred to as the 1:3 decoder, and the T-gate is essentially embedded in the 3:1 multiplexer and not used directly.

D.10. Works Cited

1. D.E. Knuth, The Art of Computer Programming - Volume 2: Seminumerical Algorithms, pp. 207-208. Addison-Wesley, 3rd ed., 1998. ISBN 0-201-89684-2. Available: http://jeff.tk/wiki/Image:Knuth-TaoCPVol2-pg207%2C8.pdf
2. The Elements of Computing Systems: Building a Modern Computer from First Principles by Noam Nisan and Shimon Schocken (MIT Press, 2005).
3. Connelly, Jeff. Jeff.tk - Trinary/Meetings. Available: http://jeff.tk/wiki/Trinary/Meetings
4. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary/Status. Available: http://jeff.tk/wiki/Trinary/Status
5. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary. Available: http://jeff.tk/wiki/Trinary
6. Chavez, Antonio and Connelly, Jeff. Trinary/Tools - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Tools
7. Chavez, Antonio and Connelly, Jeff. Trinary/CPU Simulation - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/CPU_Simulation
8. Chavez, Antonio. Trinary/Compiler - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Compiler
9. Swanson, William. *Introduction to Binary Numbers*. 2002. Available: http://www.swansontec.com/sbinary.htm
10. davido, Perl Monks. *Ternary operator (there's no Trinary operator)*. Available: http://www.perlmonks.org/?node_id=562203
11. Wall, Larry. perl.com: Apocalypse 3. Available: http://www.perl.com/pub/a/2001/10/02/apocalypse3.html?page=6
12. Wall, Larry. et. al. *Programming Perl*, 3rd. Edition. ISBN: 978-0596000271 Section 3.16: Conditional Operator.
13. The Antikythera Mechanism Research Project. Available: http://www.antikythera-mechanism.gr/project/overview
14. Lexikon. *Analog Computers*. Available: http://www.computermuseum.li/Testpage/AnalogComputers.htm
15. National Semiconductor, Application Note 31. September 2002. *Op Amp Circuit Collection*. Available: http://www.national.com/an/AN/AN-31.pdf
16. Goldstrasz, Thomas et. al. *Computers During World War Two*. Available: http://waste.informatik.hu-berlin.de/Diplom/WW2/default_e.html
17. Bains, Sunny. *Analog computer trumps Turing model*. EE Times. 11/03/1998. Available: http://www.eetimes.com/story/OEG19981103S0017
18. Principia Cybernetica Web: *Digital Computer*. Available: http://pespmc1.vub.ac.be/ASC/DIGITA_COMPU.html
19. Maney, Kevin. USA Today, September 1997. *Debate Stirs Over Origins of Computers*. Available: http://www.scl.ameslab.gov/ABC/Articles/Debate9-97.html
20. Bebop's BYTES Back. *Claude Shannon's master's Thesis*. Available: http://www.maxmon.com/1938ad.htm
21. Hannah, Eric. *United States Patent 7309866: Cosmic ray detectors for integrated circuit chips*. Available: http://tinyurl.com/3ysdmk
22. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://dx.doi.org/10.1511/2001.40.3268 and mirrored at http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
23. A. Srivastava and K. Venkatapathy, “Design and Implementation of a Low Power Ternary Full Adder,” VLSI Design, vol. 4, no. 1, pp. 75-81, 1996. doi:10.1155/1996/94696. Available: http://jeff.tk/wiki/Image:Design_and_Implementation_of_a_Low_Power_Ternary_Full_Adder.pdf
24. J.T. Butler, Multiple-Valued Logic in VLSI, IEEE Computer Society Press Technology Series, Los Alamitos, California, 1991.
25. A.K. Jain, M.H. Abd-E1-Barr and R.J. Bolton, "A new structure for CMOS realization of MVL functions," International Journal of Electronics, vol. 74, no. 2, pp. 251-263, 1993.
26. S.L. Hurst, "Two decades of multiple valued logic--an invited tutorial," in Proceedings of IEEE

International Symposium on Multiple-Valued Logic, p. 164, May 1988.
27. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
28. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
29. A. P. Dhande and V. T. Ingole. Design And Implementation Of 2 Bit Ternary ALU Slice. SETIT 2005, 3rd International Conference: Science of Electronic, Technologies of Information and Telecommunications. March 17-21, 2005, Tunisia. Available: http://jeff.tk/wiki/Image:Dhande%2C_Ingole_-_Design_and_Implementation_of_a_2_Bit_Ternary_ALU_Slice.pdf
30. P.C.Balla & A.Antoniou "low power dissipation MOS ternary logic family" IEEE journal on solid state circuits Vol. Sc-19 no-5, P.739-749, October 1984.
31. D.I.porat "Three valued digital system" Proc.IEE Vol.116, No6, P.947-955, June 1969.
32. K.C.Smith "The prospects of multivalued logic technology & application view " IEEE transaction on computer, Vol.-C -30, P-619-627 September 1981.
33. Chung-Yu-Wu"Design & application of pipelined dynamic CMOS ternary logic & simple ternary differential logic" IEEE journal on solid state circuits Vol.28, No-8, August 1993.
34. CS150. Berkeley EECS. *Bits, Bytes, Nibbles, and Words: Some definitions*. Available: http://inst.eecs.berkeley.edu/~cs150/sp98/lectures/week6_2/tsld002.htm
35. Slashdot. *Ternary Computing Revisited*. Available: http://slashdot.org/comments.pl?sid=23934
36. Sloppy. Slashdot | Ternary Computing Revisited. Monday November 19 2001. *Trits?* Available: http://slashdot.org/comments.pl?sid=23934&cid=2585807
37. Merrill, Roy D. *Ternary Logic in Digital Computers*. January 1965. Avaialble: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
38. Bowles, Gary-alexander. US Patent #5498980 Ternary/binary converter circuit (Publication Date: 03/12/1996). Available: http://www.freepatentsonline.com/5498980.html
39. Setun' W. H. Ware, S. N. Alexander, N. M. Astrahan, H. H. Goode, M. Rubinoff, P. Armer, L. Bers, H.d. Huskey, "Soviet computer technology - 1959," Communications of the ACM, pp. 149-150, 1960.. Available: http://jeff.tk/wiki/Image:Communications_of_the_ACM_-_Soviet_Computer_Technology_-_1959.pdf
40. Faden, David. Reverse Fad Productions: Flip. Available: http://www.revfad.com/flip.html
41. Crispin, M. Panda Programing. 1 April 2005. Network Working Group, Request for Comments: 4042. *UTF-9 and UTF-18 Efficient Transformation Formats of Unicode*. Available: http://www.ietf.org/rfc/rfc4042.txt RFC 4042
42. Aspinwall, Jim. eCoustics. *Hacking CPU Voltage to Speed Up Your PC*. Available: http://forum.ecoustics.com/bbs/messages/34579/147079.html
43. Engelhardt, Mike. *LTspice/SwitcherCAD III User's Manual*. Available: http://ltspice.linear.com/software/scad3.pdf
44. Howell, Louis and Raymond, Eric S. Available: http://jeff.tk/wiki/Trinary/Logic#TriINTERCAL_Manual:_5.5.2.1_UNARY_LOGICAL_OPERATORS
45. Connelly, Jeff. Trinary/Parts - Jeff.tk - First Purchase. Available: http://jeff.tk/wiki/Trinary/Parts#Shopping_List:_First_Purchase
46. All About Circuits. Producing negative supply rails - Urgent - All About Circuits. Available: http://forum.allaboutcircuits.com/showthread.php?t=10415
47. All About Circuits. negative supply - All About Circuits Available: http://forum.allaboutcircuits.com/showthread.php?t=876
48. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
49. Connelly, Jeff. Trinary Computer Architecture - Older Diagrams. Available: Image:Proposed Architecture 2.png, Image:Proposed architecture 1.png
50. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
51. Lite-On Electronics Inc. Part No. LTL-30EHJ. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-30EHJ.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1057-ND

52. Lumex. T-5mm LED, 6 leaded, multi-colored, 636 nm AlInGoP Red/574 nm, AlInGoP Green BiColor, 470 nm Ultra Super Blue, Water Color Lens Datasheet. Part #SSL-LX5099SIUBSUGB. Available: http://rocky.digikey.com/weblib/Lumex/Web%20Data/SSL-LX5099SIUBSUGB1.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=67-1829-ND
53. Lite-On Electronics Inc. Part No. LTL-293SJW. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-293SJW.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1038-ND
54. Quicktar. Current limiting resistor calculator for LEDs. Available: http://www.quickar.com/noqbestledcalc.htm
55. Eigenratios of Self-Interpreters: The Mark II OISC Self-Interpreter. Available: http://eigenratios.blogspot.com/2006/09/mark-ii-oisc-self-interpreter.html
56. Connelly, Jeff. Public Git Hosting - trinary.git/tree - circuits/. Available: [http://repo.or.cz/w/trinary.git?a=tree;f=circuits
57. Carothers D. Christopher. Evolution of Intel Microprocessors: 1971 to 2007. Available: http://www.cs.rpi.edu/~chrisc/COURSES/CSCI-4250/SPRING-2004/slides/cpu.pdf
58. Connelly, Jeff. Jeff.tk - Trinary/Symbols/Tips. Available: http://jeff.tk/wiki/Trinary/Symbols/Tips
59. Grubb, Steve. Trinary.cc. Available: http://www.trinary.cc/
60. H. T. Mouftah May 1976 Proceedings of the sixth international symposium on Multiple-valued logic. *A study on the implementation of three-valued logic*. Available: http://jeff.tk/wiki/Image:P123-mouftah_Study_on_the_Implementation_of_Three-valued_Logic.pdf
61. Mouftah, H. T. A study on the implementation of three-valued logic H. T. Mouftah May 1976 Proceedings of the sixth international symposium on Multiple-valued logic. Available: http://jeff.tk/wiki/Image:P123-mouftah_Study_on_the_Implementation_of_Three-valued_Logic.pdf
62. D.I. Porat, "Three-valued digital systems", Proc, lEE, Vol. 116, No. 6, June 1969, pp. 947-954.
63. M. Yoeli and G. Rosenfeld, "Logical design of ternary switching circuits", IEEE Trans. Elect. Comp., Vol. EC-14, February 1965, pp. 19-29.
64. M. Bitran and M.J.O. Strutt, "Minimization of ternary logic and complete set of integrable circuits", Electron. and Commun., AE0, Band 25,No. 8, 1971, pp. 387-392.
65. R.S. Nutter and R.E. Swartwout, "A ternary logic minimization technique", Conference Record of the 1971 Symposium on the Theory and Applications of Multiple-valued Logic Design, May 1971, pp. i12~125.
66. Connelly, Jeff. Trinary/Unary Quick Reference - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Unary_Quick_Reference
67. Hyde, Randall. *Art of Assembly Language*. Available: http://webster.cs.ucr.edu/AoA/DOS/ch01/CH01-2.html
68. Howell, Louis and Raymond, Eric S. *The C-Intercal Supplemental Reference Manual*. 1992-01-18. Available: http://webster.cs.ucr.edu/AoA/DOS/ch01/CH01-2.html
69. Connelly, Jeff. *Unary Gates Inside Binary Gates*. 2001-11-17. Available: http://jeff.tk/bingates/
70. Grubb, Steve. Trinary.cc. Binary Operations. 2001. Available: http://www.trinary.cc/Tutorial/Algebra/Binary.htm
71. I. Halpern and M. Yoeli, "Ternary arithmetic unit", Proc. lEE, Vol. 115, No. i0, October 1968, pp. 1585-1588. Table II : Multiple input ternary operators
72. H.T. Mouftah and I.B. Jordan, "Integrated circuits for ternary logic", Proceedings of the 1974 International Symposium on Multiple-valued Logic, May 1974, pp. 285-302.
73. E.L. Post, "Introduction to a general theory of elementary propositions", Amer. J. Math., Vol. 43, 1921, pp. I~3-185.
74. J.B. Rosser and A.R. Turquette, "Many-valued logics", North-Holland Publishing Co., Amsterdam, 1952.
75. M. Yoeli and G. Rosenfeld, "Logical design of ternary switching circuits", IEEE Trans. Elect. Comp., Vol. EC-14, February 1965, pp. 19-29.
76. R. Vacca, "A three-valued system of logic and its applications to base three digital circuits", Proc. Intern. Conf. Inform. Processing, (UNESCO), June 1959, pp. 407-414.

77. H. Mine, T. Hasegawa, M. Ikeda and T. Shintani, "A construction of ternary logic circuits", Electron. Commun. in Japan, Vol. 51-C, No. 12, pp. 133-140.
78. Sobie, Rick. Troolean operators, Available: http://sci.tech-archive.net/Archive/sci.physics/2006-03/msg00869.html
79. Nynaeve. Blog Archive - The troolean strikes back. Available: http://www.nynaeve.net/?p=87
80. CSE 460 - Spring 2006, Boolean Algebra Definitions, Theorems, and Postulates, Available: http://www.arl.wustl.edu/~lockwood/class/cse460/ba.pdf]
81. Connelly, Jeff. Extensions:Trinary MediaWiki Extension. Available: http://jeff.tk/wiki/Extensions:Trinary
82. Allright, James. Balanced Ternary Web Page. Available: http://web.archive.org/web/20050211091401/http://perun.hscs.wmin.ac.uk/~jra/ternary/
83. W. Ahrens, *Mathematische Unterhaltungen und Spiele* **1** (Leipzig: Teubner, 1910), Section 3.4; H. Hermelink, *Janus* **65** (1978), 105-117
84. *Philos. Trans.* **34** (1726) 161-173
85. *The Philosophy of Arithmetic* (Edinburgh: 1817); see pages 33-34, 54, 64-65, 117, 150
86. *Computes Rendus Acad. Sci. Paris* **11** (1840), 789-798
87. J. Bharati, *Vedic Mathematics* (Delhi: Motilal Banarsidass, 1965)
88. *Mathematical Education* **5**, 3 (1989), 129-133
89. *Computes Rendus Acad. Sci. Paris* **11** (1840), 903-905
90. *American Mathematical Monthly* **57** (1950), 90-93
91. *High-speed Computing Devices*, Engineering Research Associates (McGraw-Hill, 1950), 287-289.
92. *Communications of the Association for Computing Machinery* **3** (1960), 149-150
93. Bhattacharjee, Abhijit. *A polar place value number system for computers and life in general*. Available: http://abhijit.info/tristate/tristate.html
94. H.T. Mouftah, K.C. Smith and Z.G. Vranesic Department of Electrical Engineering University of Toronto Toronto, Ontario, Canada. *Ternary Logic In a Positional Control System*. Available: http://jeff.tk/wiki/Image:P135-mouftah_Ternary_Logic_in_a_positional_control_system.pdf
95. Walker, John. Forumilab, August 19, 1996. Minus Zero. Available: http://www.fourmilab.ch/documents/univac/minuszero.html
96. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
97. Ternary computers: part I: motivation for ternary computers, International Symposium on Microarchitecture archive, Conference record of the 5th annual workshop on Microprogramming table of contents, Urbana, Illinois, 1972
98. Merrill, Roy D. *Ternary Logic in Digital Computers*. January 1965. Available: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
99. Connelly, Jeff. Full Adder Timing Diagram - Internal Signals I. Available: http://jeff.tk/wiki/Image:Full_Adder_Timing_Diagram_-_Internal_Signals_I.png
100. Connelly, Jeff. Full Adder Timing Diagram - Internal Signals II. Available: http://jeff.tk/wiki/Image:Full_Adder_Timing_Diagram_-_Internal_Signals_II.png
101. Connelly, Jeff. Full Adder Y Decoder Signals. Available: Full Adder Y Decoder Signals
102. Halleck, John (via email) and Eide, Leroy. Fast BT division-by-2 using "Just-in-Time Subtraction". Available: http://www.dyalog.dk/dfnsdws/n_JitSub.htm
103. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
104. Frieder, Gideon. Part 1 - Motivation for Ternary Computers. Available: http://jeff.tk/wiki/Image:P83-frieder_Ternary_Computers_-_Part_1_-_Motivation_for_Ternary_Computers.pdf
105. Frieder, Gideon. Part 2 - Emulation of Ternary Computers. Available: http://jeff.tk/wiki/Image:P86-frieder_-_Ternary_Computers_-_Part_2_-_Emulation_of_Ternary_Computers.pdf
106. Setun' W. H. Ware, S. N. Alexander, N. M. Astrahan, H. H. Goode, M. Rubinoff, P. Armer, L. Bers, H.d. Huskey, "Soviet computer technology - 1959," Communications of the ACM, pp. 149-150, 1960.. Available: http://jeff.tk/wiki/Image:Communications_of_the_ACM_-_Soviet_Computer_Technology_-_1959.pdf

107. Chillet, Daniel et. al. Team: r2d2. Available: http://web.archive.org/web/20060514100747/http://www.inria.fr/rapportsactivite/RA2004/r2d22004/uid51.html
108. Hyperphysics: Magnetic Properties of Solids. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/magpr.html
109. Hyperphysics: Ferromagnetism. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/ferro.html
110. Porter, Harry. Harry Poter's Relay Computer. Last updated 2007-11-15. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/ferro.html
111. Woodgears.ca: Binary marble adding machine. Available: http://woodgears.ca/marbleadd/
112. Phun — 2D Physics Sandbox — Home. Available: http://www.phunland.com/wiki/Home
113. mlittman. Youtube: Logic gates using toys. Available: http://www.youtube.com/watch?v=H-53TVR9EOw
114. Grubb Steve. Slashdot: Ternary Computing Revisited: Re:impractical circuits. 2001-11-19. Available: http://slashdot.org/comments.pl?sid=23934&cid=2586271
115. H. Chan, T. van Duzer, D. Erne. A tri-stable state Josephson device memory cell. Available: http://slashdot.org/comments.pl?sid=23934&cid=2586271
116. Merrill, Roy D. Ternary Logic in Digital Computers. January 1965. Available: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
117. A. E. Slade, "A Cryotron Memory Cell," Proc. IRE, Vol. 50, Jan. 62, p. 81.
118. C. F. Kooi, R. D. Merrill and H. H. Nakano, "Superconductive Ternary Information Storage Device," Lockheed Missiles & Space Co., *Research on Automatic* Computer Electronics*, Vol. I, RTD-TDR-63-4173, Oct. 31, 1963, pp. A-36-A-48.*
119. Yi, Jin. Institute of Physics. Ternary Optical Computing Architecture. Available for purchase: http://www.iop.org/EJ/abstract/1402-4896/2005/T118/025/
120. NewScientistTech. Ten weirdest computers, New Scientist, 18:30 11 April 2008 by Duncan Graham-Rowe. Available: http://technology.newscientist.com/article/dn13656-ten-weirdest-computers.html?DCMP=ILC-hmts&nsref=news1_head
121. Grubb, Steve. Trinary.cc: Transistor Models. Available: http://www.trinary.cc/Tutorial/Transistors/Transistors.htm
122. Connelly, Jeff. Jeff.tk - Trinary/LTspice tips. Available: http://jeff.tk/wiki/Trinary/LTspice
123. Linear Technology — Design Simulation and Device Models. Available: http://www.linear.com/designtools/software/
124. Patel, Chirag. Jeff.tk - Trinary/VLSI. Available: http://jeff.tk/wiki/Trinary/VLSI
125. Connelly, Jeff. Additional files for LTspice (SwitcherCAD III). Available: http://jeff.tk/wiki/Image:SwCADIII-jc.zip
126. Connelly, Jeff. repo.or.cz / trinary.git. Git repository. Available: http://repo.or.cz/w/trinary.git
127. Maxim. Application Note 638: Selecting the Right CMOS Analog Switch. Available: http://www.maxim-ic.com/appnotes.cfm/appnote_number/638
128. Fairchild Semiconductor. CD4066BC Quad Bilateral Switch. November 1983, Revised October 2005. Available http://www.fairchildsemi.com/ds/CD/CD4066BC.pdf
129. Maxim. Low-Voltage, Quad, SPST CMOS Analog Switches. 19-4793; Rev 5; 6/07. Available http://www.fairchildsemi.com/ds/CD/CD4066BC.pdf
130. Hussein T. Mouftah, Ternary Logic Circuits with CMOS Integrated Circuits, United States Patent 4,107,549. August 15th, 1978. Figure 1: Ternary Inverters. Available: http://jeff.tk/wiki/Image:Mouftah-1-Ternary_Inverters.png
131. Patel, Chirag and Connelly, Jeff. Gates with pull-middle resistors other than 12 kΩ.zip. Available: http://jeff.tk/wiki/Image:Gates_with_pull-middle_resistors_other_than_12k.zip
132. Hussein T. Mouftah, Ternary Logic Circuits with CMOS Integrated Circuits, United States Patent 4,107,549. August 15th, 1978. Figure 12: Cycling Gates. Available: http://jeff.tk/wiki/Image:Mouftah-12-Cycling_Gates.png
133. Hussein T. Mouftah, Ternary Logic Circuits with CMOS Integrated Circuits, United States Patent 4,107,549. August 15th, 1978. Figure 3: Ternary NAND. Available: http://jeff.tk/wiki/Image:Mouftah-3-Ternary_NAND.png
134. D.E. Knuth, The Art of Computer Programming - Volume 2: Seminumerical Algorithms, pp. 207-208.

Addison-Wesley, 3rd ed., 1998. ISBN 0-201-89684-2. Available: http://jeff.tk/wiki/Image:Knuth-TaoCPVol2-pg207%2C8.pdf
135. Merrill, Roy D. Ternary Logic in Digital Computers. January 1965. Available: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
136. Hussein T. Mouftah, Ternary Logic Circuits with CMOS Integrated Circuits, United States Patent 4,107,549. August 15th, 1978. Available: http://jeff.tk/wiki/Image:Mouftah_-_Ternary_Gate_Figures.pdf
137. A. Srivastava and K. Venkatapathy, "Design and Implementation of a Low Power Ternary Full Adder," VLSI Design, vol. 4, no. 1, pp. 75-81, 1996. doi:10.1155/1996/94696. Available: http://jeff.tk/wiki/Image:Design_and_Implementation_of_a_Low_Power_Ternary_Full_Adder.pdf
138. Grubb, Steve. Trinary.cc - Multiplexers. Available: http://www.trinary.cc/Tutorial/Mux/Mux.htm
139. Bowles, Gary-alexander. US Patent #5498980 Ternary/binary converter circuit (Publication Date: 03/12/1996). Available: http://www.freepatentsonline.com/5498980.html
140. Electronic Letters. 17th of October, 1974. Volume 10, Number 21. Implementation of 3-Valued Logic with COS/MOS Integrated Circuits. Available: http://jeff.tk/wiki/Image:ELECTRONICS_LETTERS_17th_October_1974_Vol._10_No._21_-_IMPLEMENTATION_OF_3-VALUED_LOGIC_WITH_C.O.S._M.O.S._INTEGRATED_CIRCUITS.pdf

Appendix E: Circuit Construction

The circuits described in the previous section were physically realized using the following integrated circuits:

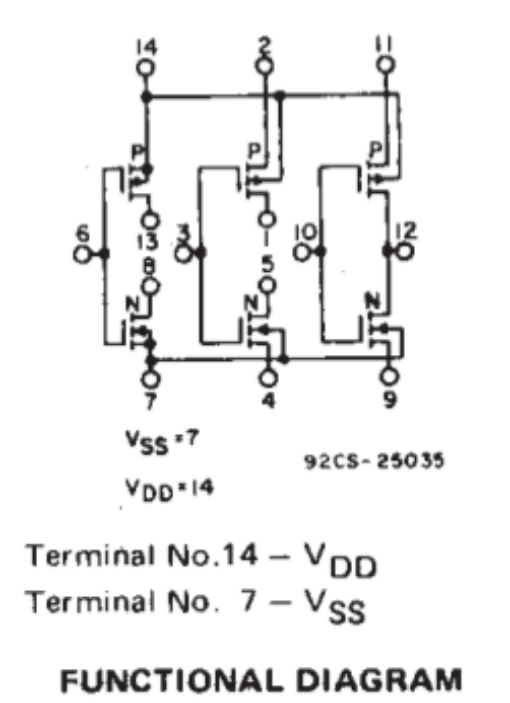


Figure E.1. TI CD4007UB Dual Complementary Pair + Binary Inverter, functional diagram. With 100 ICs, we have 100 pairs like on the far left, and 100 in the middle (more usable).

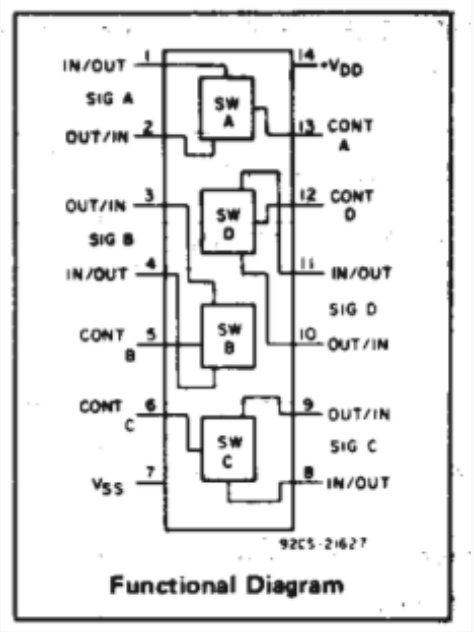


Figure E.2. TI CD4016 Quad Bidirectional Analog Switch functional diagram.

There are alternative MOSFET complementary pair ICs[141], but the above chips are time-tested, widely available, and are the ones we used.

E.1. Layout

We procured several breadboards to build the circuits on. While they were initially wired in an *ad hoc* manner, we found it helpful to use a computer-aided design tool to produce a layout. For this purpose, we used FreePCB[142] to design Printed Circuit Board (PCB) layouts.

A significant problem is that the LTspice schematics were built using MOSFETs, not CD4007s, and using a transmission gate model, not a CD4016. The CD4016 integrated circuit contains four transmission gates, and the CD4007 two complementary MOSFET pairs, one with the sources hardwired to $G_Vdd and $G_Vss, the other not. To map the individual MOSFETs and transmission gates in LTspice to pins on ICs, I wrote a suite of Python scripts totaling approximately 1,300 lines to accomplish this task. The main program is `bb/bb.py`, found inside our git repository[143], although it can be easily driven using the convenience wrapper script `bb/pcb.py` which interactively asks questions and performs the appropriate operations.

`bb.py` performs the following tasks:

- Takes a hierarchical SPICE netlist (*.net, in LTspice, exported using View -> SPICE Netlist) and flattens it so only transistor-level subcircuits are instantiated.
- From the flattened netlist, allocates parts within ICs for each MOSFET or t-gate.
- Writes out a new circuit with what pins of what ICs and resistors to connect where, in SPICE netlist format to a .net2 file

Next, `pads.py` does the following, or `bb.py` also does this if given the `-p` flag:

- Convert the .net2 file to a PADS-PCB netlist, named ending in .pads, assigning footprints as needed

The `.pads` is imported into FreePCB (or other programs that support the PADS-PCB netlist format) using "Import Netlist...". From there, we designed a PCB, auto-routed it, and sent it off for manufacturing.

Additionally, I also developed a tool, `mergepads.py`, to merge PADS-PCB netlists before importing into FreePCB. This tool can be used to combine multiple circuits, as designed in LTspice, into one board in FreePCB.

E.2. Footprints

A *footprint* is the physical area that the a given part occupies. The following board footprints were used for the following component models:

Table E.1. Footprints for Trinary Circuit Boards

Models Translated	Footprint	Source	Purpose
CD4007/CD4016	14DIP300	FreePCB	14-pin DIP (CD4007 or CD4016)
R	RC07	FreePCB	1/4 W resistor (12 kΩ)
V	1X2HDR-100	FreePCB	Voltage source (attach it here)
?	SS14MDP2	Custom design	SP3T switch

We designed a footprint for the SS14MDP2 switch to use for input:

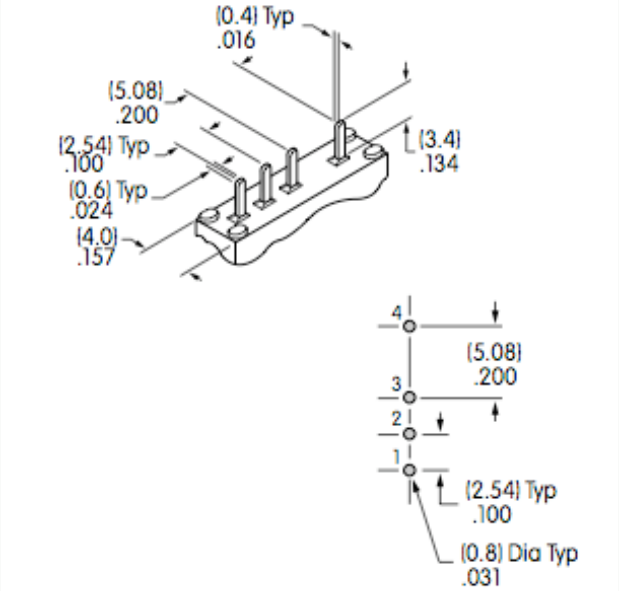


Figure E.3. Physical dimensions of the NKK SS14MDP2 switch, from the *Ultra-Miniature Slides SS Series Datasheet*[144].

We ran into an issue with the footprints for the headers. The `th_header` library included with FreePCB included footprints named *n*XHDR-100, that use holes of 28 mil diameter. The 28 mil holes are too small[145] for the most common type of header, which uses 40 mil headers. We were able to resolve this issue by acquiring 28 mil headers from Samtec, Inc.[146].

E.3. Chip Maps

bb/ (http://repo.or.cz/w/trinary.git?a=tree;f=bb) contains several Python modules that contain data structure that map low-level subcircuits (such as tnand) to pins on an integrated circuit (such as the CD4007). In some cases, several subcircuits may map to one chip. Current chip maps are:

Table E.2. Transmission Gate (tg) Chip Map A

Port	Chip	Pin
IN_OUT	CD4007	1
OUT_IN	CD4007	2
CONTROL	CD4007	13

Table E.3. Transmission Gate (tg) Chip Map B

Port	Chip	Pin
IN_OUT	CD4007	4
OUT_IN	CD4007	3
CONTROL	CD4007	5

Table E.4. Transmission Gate (tg) Chip Map C

Port	Chip	Pin
IN_OUT	CD4007	8
OUT_IN	CD4007	9
CONTROL	CD4007	6

Table E.5. Transmission Gate (tg) Chip Map D

Port	Chip	Pin
IN_OUT	CD4007	11
OUT_IN	CD4007	10
CONTROL	CD4007	12

Table E.6. Basic Ternary Inverter (tinv) Chip Map A

Port	Chip	Pin
Vin	CD4007	6
PTI_Out	CD4007	13
NTI_Out	CD4007	8
STI_Out	N/A	STI_Out(**)

Table E.7. Basic Ternary Inverter (tinv) Chip Map B

Port	Chip	Pin
Vin	CD4007	6
PTI_Out	CD4007	13
NTI_Out	CD4007	8
STI_Out	N/A	STI_Out(**)
$G_Vdd	CD4007	2
$G_Vss	CD4007	4

Table E.8. Basic Ternary Inverter (tinv) Global Pins

Port	Chip	Pin
$G_Vdd	CD4007	14
$G_Vss	CD4007	7

(**) STI_Out connects to the other ends of two 12 kΩ resistors, which themselves connect to PTI_Out and NTI_Out. See tinv.asc.

Table E.9. TNAND Chip Map

Port	Chip	Pin
A	CD4007	3
B	CD4007	6
$G_Vdd	CD4007	2, 14
TNAND_Out	N/A	TNAND_Out(**)
$G_Vss	CD4007	7
Internal Nodes		
NI	CD4007	4, 8
NP	CD4007	1, 13
NN	CD4007	5

Table E.10. TNOR Chip Map

Port	Chip	Pin
A	CD4007	6
B	CD4007	3
TNOR_Out	N/A	TNOR_Out
$G_Vss	CD4007	4, 7
$G_Vdd	CD4007	14
Internal Nodes		
NI	CD4007	13, 2
NP	CD4007	1
NN	CD4007	8, 5

(**) Connects to two resistors; see tnor.asc or tnand.asc for details. NI connects two pins together. NP and NN connect to the positive and negative pull-middle resistors, respectively.

E.4. Interconnects

Before showing the actual boards we designed and constructed, let us show how the boards are to be connected together to build a computer system. Two separate architectures are described here, TCA2 and TCA0, which will be discussed in great detail in the coming sections.

E.4.1. TCA2

The Trinary Computer Architecture v2, executes the full 3-instruction architecture with cmp, be, and lwi. This

is the architecture that runs the guessing game.

The following boards needed (the identifiers of the board are in parenthesizes):

- 2 x Multiplexer Boards (control, mux-alu)
- 1 x Logic Board (logic)
- 2 x Memory Boards (register-1, register-2)
- 3 x Adder Boards (adder-1, adder-2, adder-3)
- 1 x Sign Board (sign)

Total: 9 boards

Table E.11. Components and Boards for TCA2

Part Name	Instruction	Board Type	Board Name	Subcomponent Name/Number
JUMP_MUX	be	#Multiplexer Board	control	1/9
MUX_PC	be	#Multiplexer Board	control	2/9
CYCLE_PC	all	#Logic Board	logic	cycle up
PROGRAM_COUNTER	all	#Memory Board	register-2	1/4
SWROM	all	#Multiplexer Board	control	4/9, 5/9, 6/9 (note: 3/9 currently unused)
INSTR_DEC	all	#Logic Board	logic	1:3 decoder
REGISTER_A	lwi	#Memory Board	register-1	1/4, 2/4, 3/4
DO_LWI	lwi	#Logic Board	logic	tand 1/3
MUX_ALU_A	cmp	#Multiplexer Board	mux-alu	1/9, 2/9, 3/9
MUX_ALU_B	cmp	#Multiplexer Board	mux-alu	4/9, 5/9, 6/9
BUF_A0	cmp	#Logic Board	logic	buf1
BUF_A1	cmp	#Logic Board	logic	buf2
BUF_A2	cmp	#Logic Board	logic	buf3
DO_CMP	cmp	#Logic Board	logic	tand 2/3
STATUS_REG	cmp	#Memory Board	register-2	2/4
ALU:fa0	cmp	#Adder Board	adder-1	the one adder on the board
ALU:fa1	cmp	#Adder Board	adder-2	the one adder on the board
ALU:fa2	cmp	#Adder Board	adder-3	the one adder on the board
ALU:negB0	cmp	#Logic Board	logic	sti 1
ALU:negB1	cmp	#Logic Board	logic	sti 2
ALU:negB2	cmp	#Logic Board	logic	sti 3
ALU:sign	cmp	#Sign Board	sign	the one sign detector on the board
CLK_NEG	all	#Logic Board	logic	sti 4

We did not build TCA2, although it is fully simulated at the transistor-level, as will be detailed in the coming sections.

E.4.2. TCA0

The Trinary Computer Architecture v0 is a simplified architecture that only supports the lwi instruction. The purpose of this architecture is to run a "Christmas lights game", demonstrating that the computer is programmable, while simplifying construction.

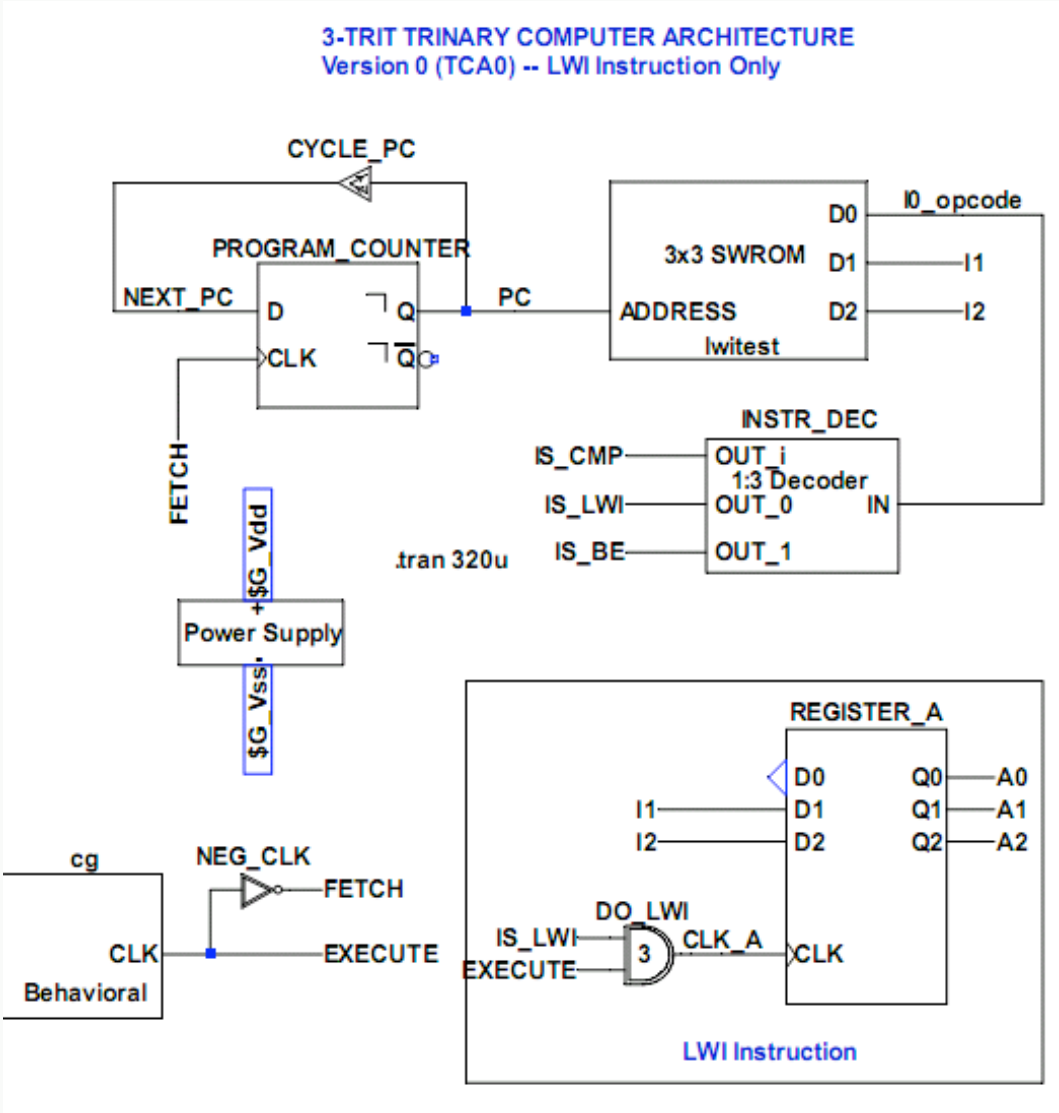


Figure E.4. High-level schematic of Trinary Computer Architecture v0, also known as the LWI instruction example.

The following boars are needed:

- 1 x Multiplexer Board (control)
- 1 x Logic Board (logic)
- 1 x Memory Board (registers)

Total: 3 boards

Table E.12. Components and Boards for TCA0

Part Name	Instruction	Board Type	Board Name	Subcomponent Name/Number
CYCLE_PC	all	#Logic Board	logic	cycle up
PROGRAM_COUNTER	all	#Memory Board	registers	4/4
SWROM	all	#Multiplexer Board	control	1/9, 2/9, 3/9 (note: 4/9 to 9/9 are unused)
INSTR_DEC(*)	all	#Logic Board	logic	1:3 decoder
REGISTER_A	lwi	#Memory Board	registers	1/4, 2/4, 3/4
DO_LWI(*)	lwi	#Logic Board	logic	tand 1/3
CLK_NEG	all	#Logic Board	logic	sti 4
cg	all	Clock generator(**)	clock	clock

(*) During testing and construction, the INSTR_DEC 1:3 decoder and DO_LWI TAND components can be eliminated and the hardware can be hardwired to always execute LWI. To do this, leave I0_opcode disconnected (or connect it to the D0 input of REGISTER_A) and connect CLK_A to CLK. This means that initially, only the CYCLE_UP cycle up gate and CLK_NEG inverter need to be built on breadboard if the logic board is unavailable.

(**) The clock generator will be built on breadboard. For testing, a function generator can be used.

FreePCB designs for all these boards are available in the git repository.

E.5. Boards

We designed the following boards:

Table E.13. Printed Circuit Boards

Board Name	#	Rev.	Computer	Filename	Size (Inches)	Contents	Status
Multiplexer Board	2	1	TCA0/2	muxes_routed.fpc	8 x 5.6	9 x 3:1 trinary multiplexers	Constructed
Memory Board	2	1	TCA0/2	dtflop-ms2_test_routed.fpc	8 x 6.3	4 x D-type Trinary Master-Slave Tri-Flops	Constructed
Logic Board	1	3	TCA0/2	logic_board-routed.fpc	5 x 6	1 x 1:3 decoder, 1 x cycle-up, 3 x TAND, 4 x basic ternary inverters, 4 buffers	Constructed
Adder Board	3	1	TCA2	full_adder_routed.fpc	4.5 x 6	Full adder; 3 of these + 4 inverters + sign detector make an ALU.	Constructed
Sign Board	1	1	TCA2	sign_detector_routed.fpc	4.55 x 3.7	4-trit sign detector, part of ALU	Constructed
I/O Board	1	-	TCA0/2	-	-	3 x 3 SP3T switches for instructions, 2 x 3 LEDs for OUT and A registers, 3 x break-before-make SP3T	(Breadboard)

					switches for IN	
ROM Board	0 2	TCA0/2	swrom-fixed_routed.fpc	6 x 6	3 x 3 SP3T switches, 1 x 9:3 multiplexer	Designed

Each board will be briefly described and the layout will be shown in this section, although the details of contents of the board will be described in the upcoming sections.

E.5.1. ROM Board

The ROM board provides read-only memory. This was our first board, but it was nonfunctional because pins 3 and 4 of the SP3T switch footprint were swapped:

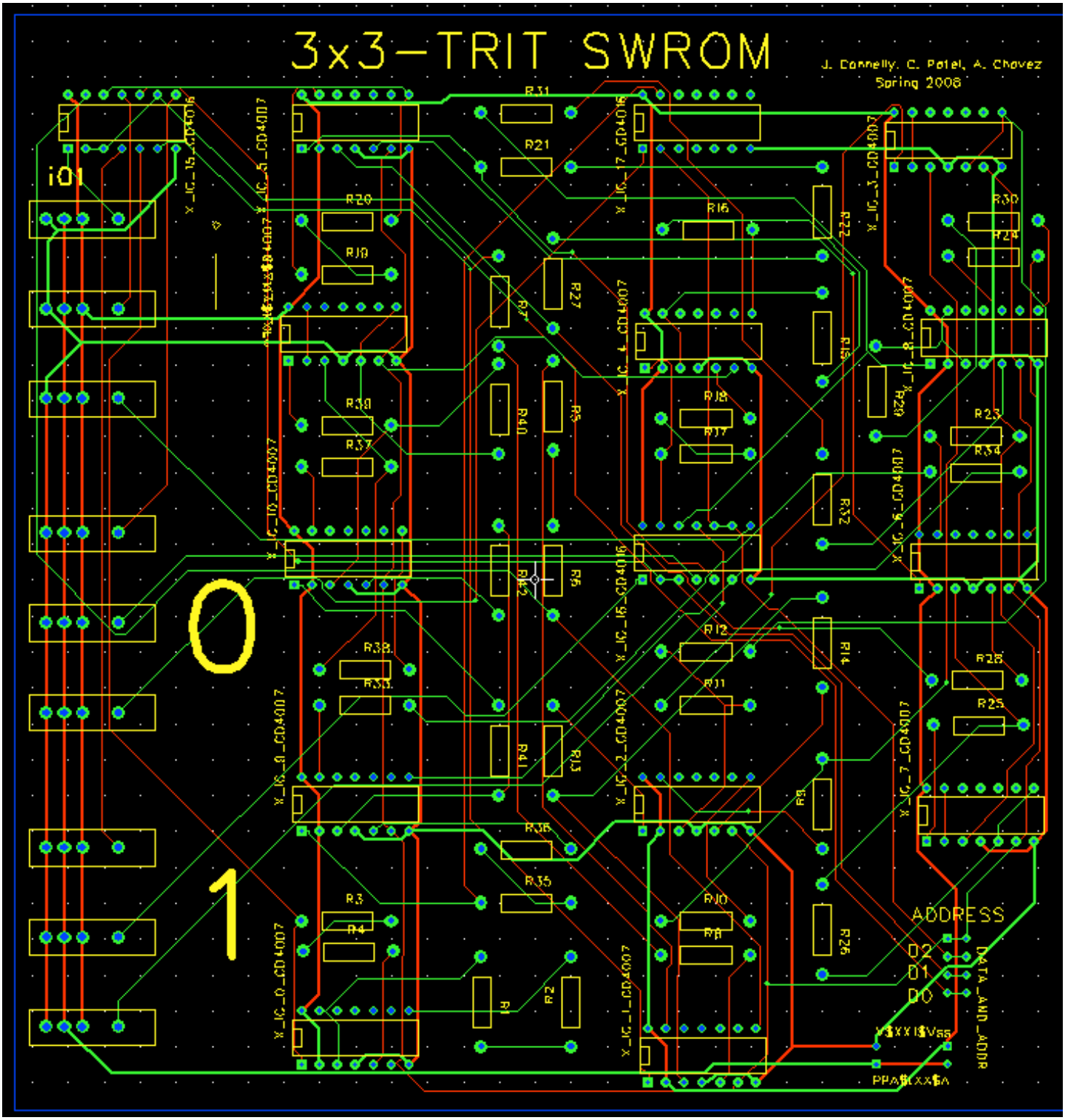


Figure E.5. SWROM PCB layout, revision 1. Incorrect SP3T switch pinout.

We have two copies of this board. We also fixed the SP3T pinout, although the board remained nonfunctional.

E.5.2. Multiplexer Board

The multiplexer board provides several multiplexers to use for control circuitry. 9 x 3:1 multiplexers are on this board. The top row has incompletely-routed traces, but the multiplexers on the second and third row are functional.

We have four of these boards:

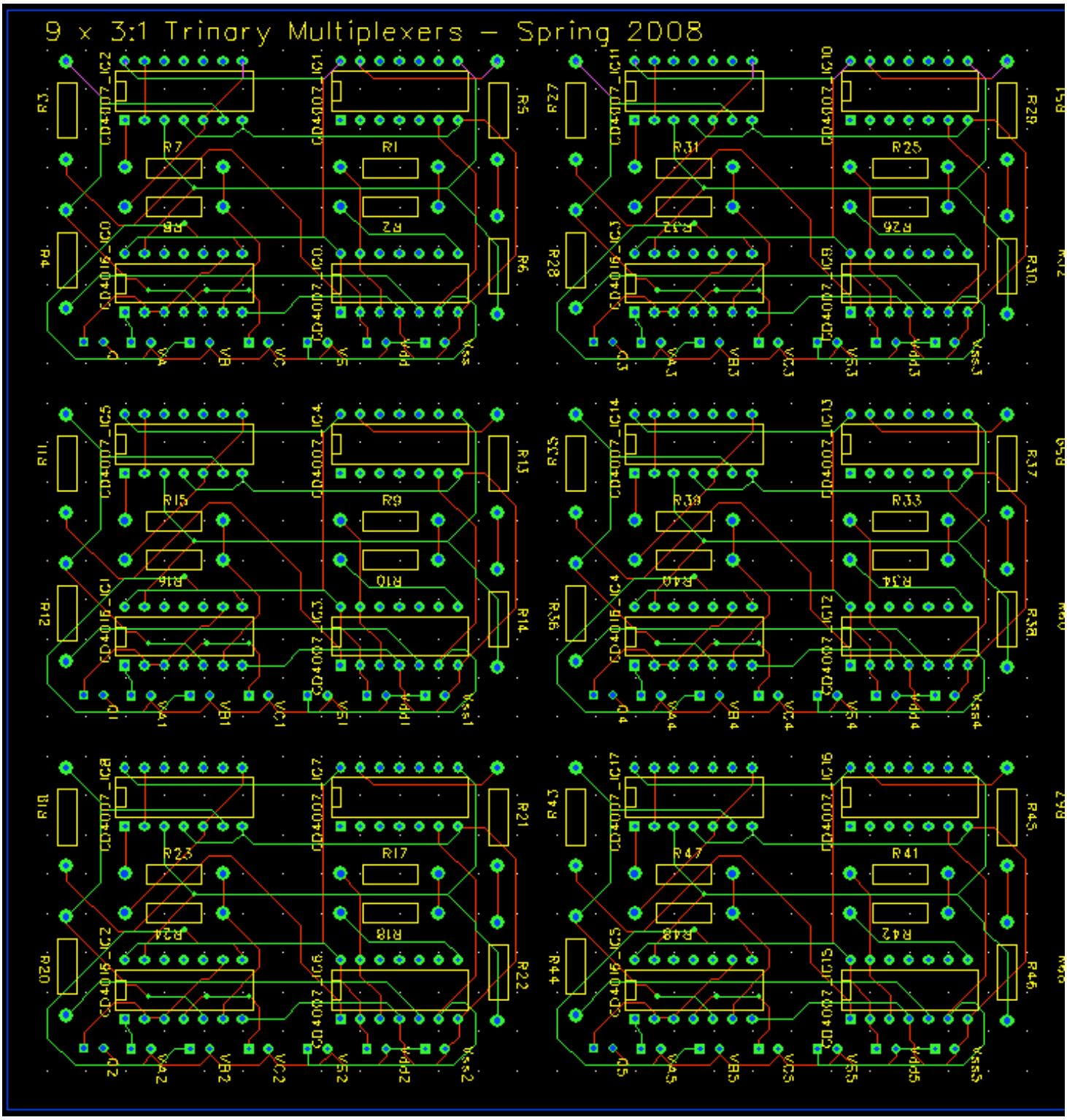
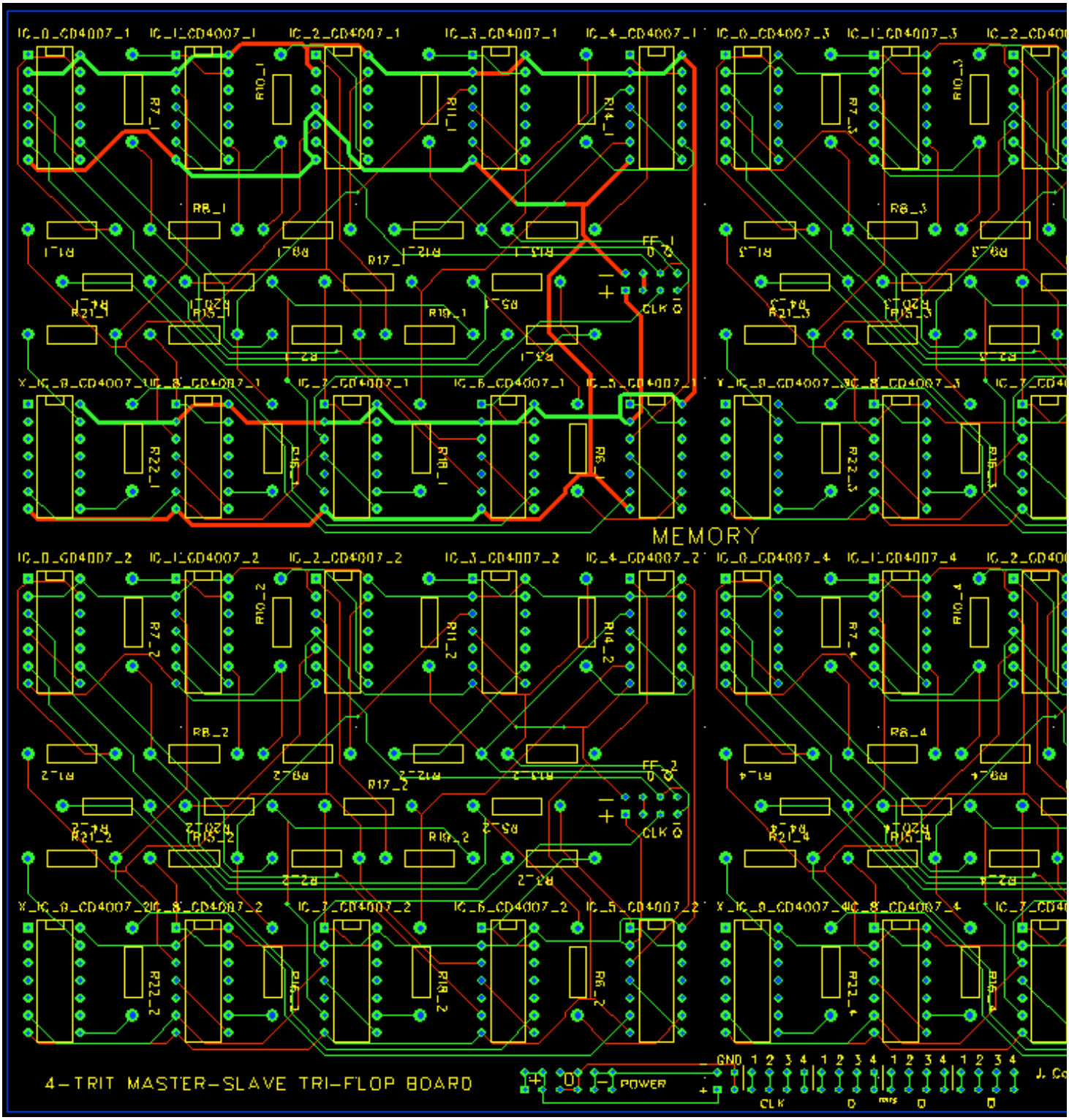


Figure E.6. Screenshot of the 9 x multiplexer board. FreePCB design and CAM files available in [1] (http://repo.or.cz/w/trinary.git?a=tree;f=bb) .

E.5.3. Memory Board

The memory board provide registers for read/write storage.

We have four of these boards:



Screenshot of the dtflop-ms2_test_routed 4-trit master-slave tri-flop memory board. FreePCB design and CAM files available in [2] (http://repo.or.cz/w/trinary.git?a=tree;f=bb) .

.

Note that in this board, pin 9 and pin 5 should be shorted, as this fixes an error.

E.5.4. Adder Board

The adder board provides a single trinary full adder, that adds three trits together into a carry and sum output. Multiple adder boards can be chained together to make multiple-trit adders. This was the first board designed

to use 7 x 12 kΩ resistor arrays (MDP1403-12K, instead of discrete resistors, for easier construction at the (acceptable) cost of more complex routing.

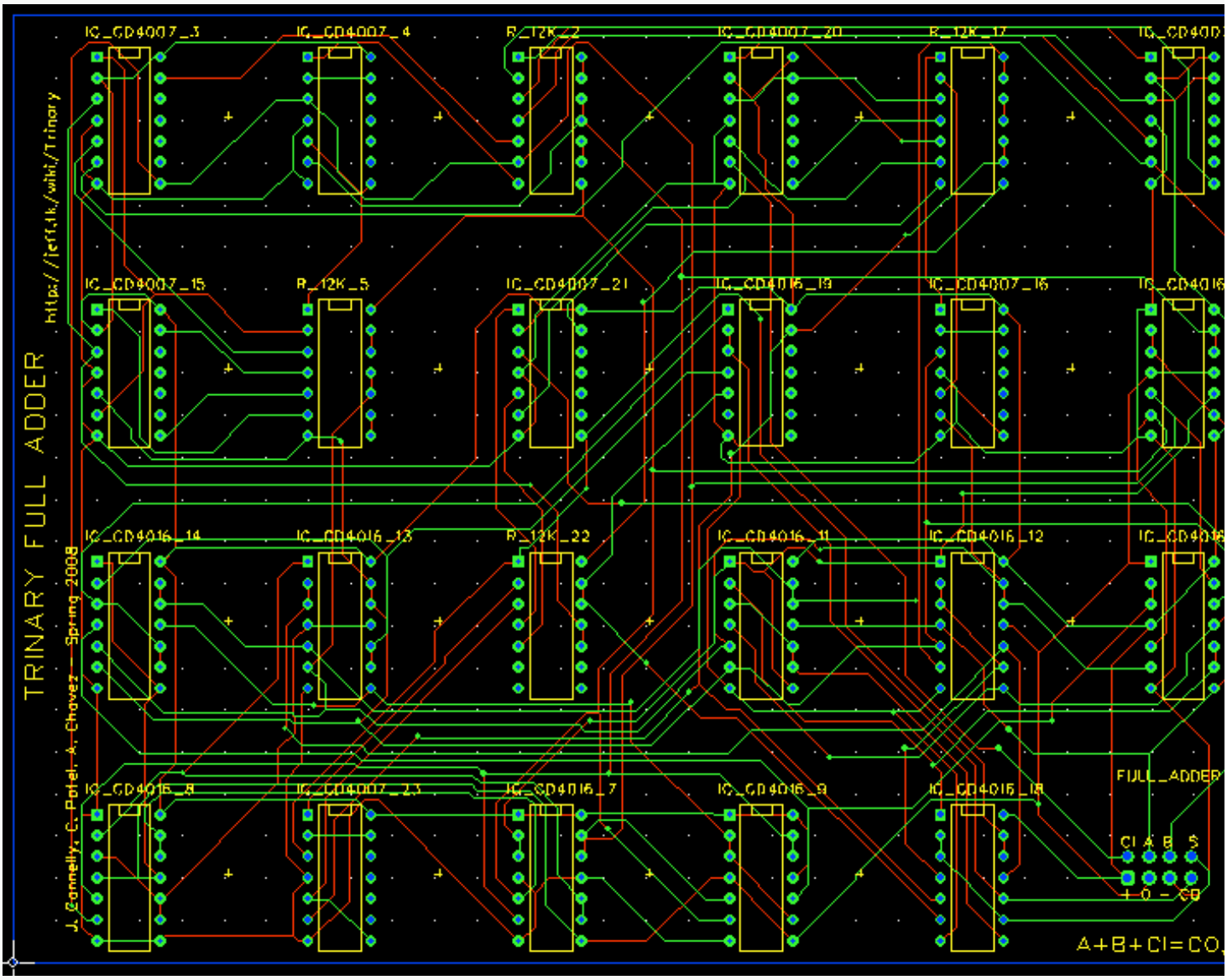


Figure E.7. Adder PCB layout.

E.5.5. Sign Board

The sign board takes a 4-trit balanced trinary number as the input, and outputs a single trit that indicates whether the number is negative, zero, or positive:

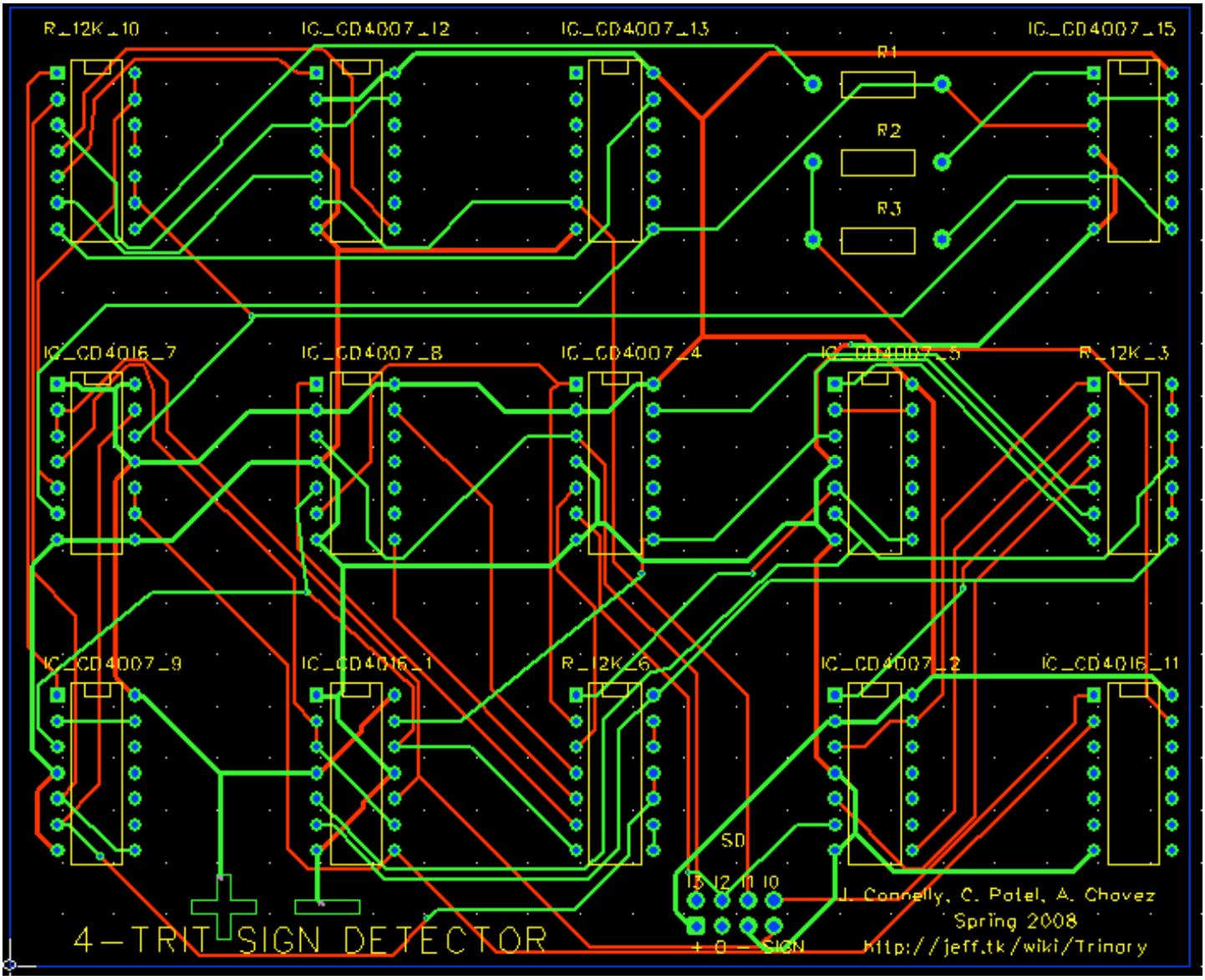


Figure E.8. Sign detector PCB layout.

E.5.6. Logic Board

This board contains various glue logic. The circuit is available in the git repository as `logic_board`, and contains:

- 4 buffers
- 4 basic ternary inverters
- 1 x 1:3 decoder
- 1 cycle up gate
- 3 TAND gates

Revision 1 used discrete resistors and multiple headers, and was not manufactured. Revision 2 uses resistor networks and one big header:

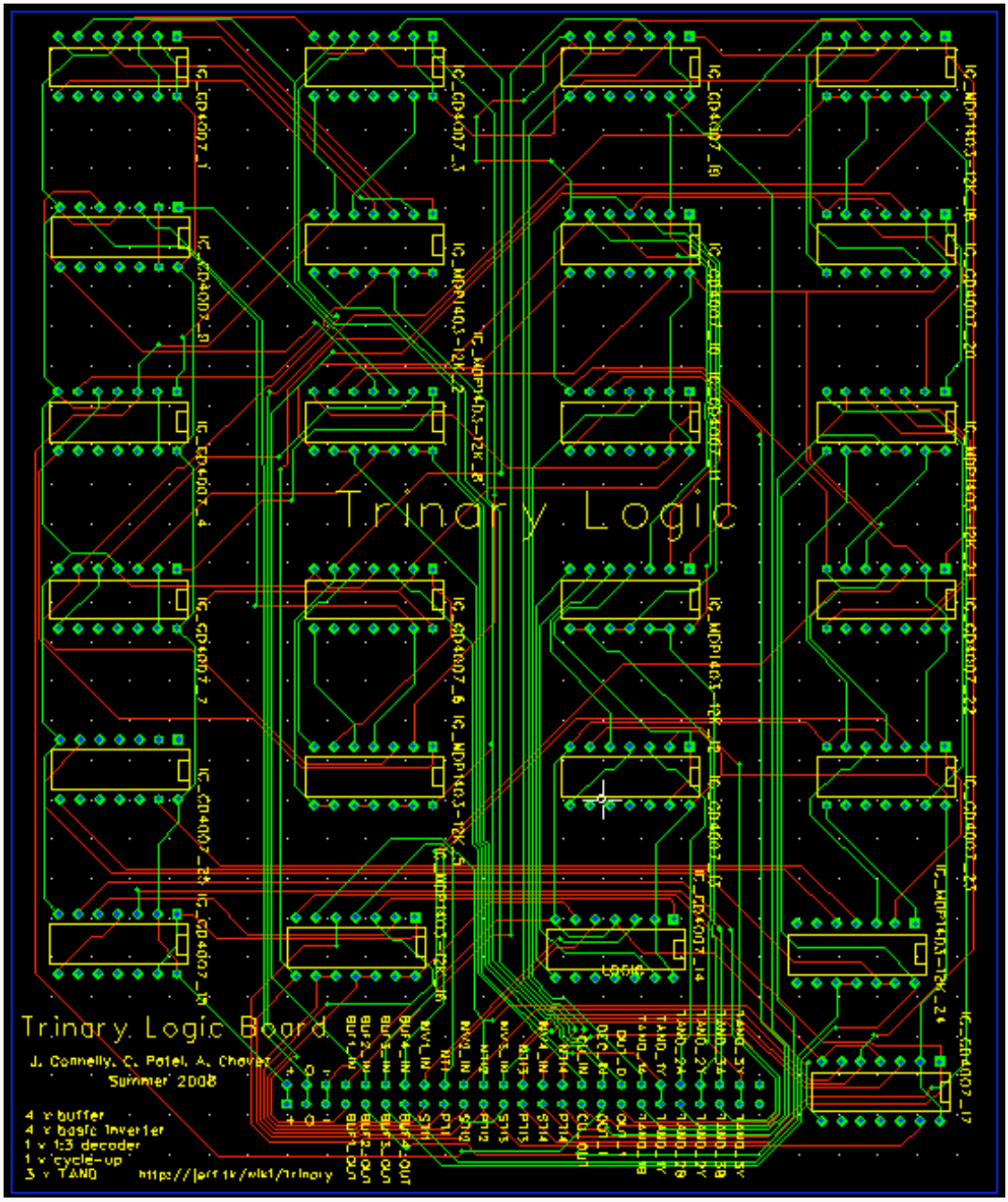


Figure E.9. Logic board revision 2.

E.6. Works Cited

1. D.E. Knuth, The Art of Computer Programming - Volume 2: Seminumerical Algorithms, pp. 207-208. Addison-Wesley, 3rd ed., 1998. ISBN 0-201-89684-2. Available: http://jeff.tk/wiki/Image:Knuth-TaoCPVol2-pg207%2C8.pdf
2. The Elements of Computing Systems: Building a Modern Computer from First Principles by Noam Nisan and Shimon Schocken (MIT Press, 2005).
3. Connelly, Jeff. Jeff.tk - Trinary/Meetings. Available: http://jeff.tk/wiki/Trinary/Meetings
4. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary/Status. Available: http://jeff.tk/wiki/Trinary/Status
5. Connelly, Jeff; Patel, Chirag; Chavez, Antonio. Jeff.tk - Trinary. Available: http://jeff.tk/wiki/Trinary
6. Chavez, Antonio and Connelly, Jeff. Trinary/Tools - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Tools
7. Chavez, Antonio and Connelly, Jeff. Trinary/CPU Simulation - Jeff.tk. Available:

http://jeff.tk/wiki/Trinary/CPU_Simulation
8. Chavez, Antonio. Trinary/Compiler - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Compiler
9. Swanson, William. *Introduction to Binary Numbers*. 2002. Available: http://www.swansontec.com/sbinary.htm
10. davido, Perl Monks. *Ternary operator (there's no Trinary operator)*. Available: http://www.perlmonks.org/?node_id=562203
11. Wall, Larry. perl.com: Apocalypse 3. Available: http://www.perl.com/pub/a/2001/10/02/apocalypse3.html?page=6
12. Wall, Larry. et. al. *Programming Perl*, 3rd. Edition. ISBN: 978-0596000271 Section 3.16: Conditional Operator.
13. The Antikythera Mechanism Research Project. Available: http://www.antikythera-mechanism.gr/project/overview
14. Lexikon. *Analog Computers*. Available: http://www.computermuseum.li/Testpage/AnalogComputers.htm
15. National Semiconductor, Application Note 31. September 2002. *Op Amp Circuit Collection*. Available: http://www.national.com/an/AN/AN-31.pdf
16. Goldstrasz, Thomas et. al. *Computers During World War Two*. Available: http://waste.informatik.hu-berlin.de/Diplom/WW2/default_e.html
17. Bains, Sunny. *Analog computer trumps Turing model*. EE Times. 11/03/1998. Available: http://www.eetimes.com/story/OEG19981103S0017
18. Principia Cybernetica Web: *Digital Computer*. Available: http://pespmc1.vub.ac.be/ASC/DIGITA_COMPU.html
19. Maney, Kevin. USA Today, September 1997. *Debate Stirs Over Origins of Computers*. Available: http://www.scl.ameslab.gov/ABC/Articles/Debate9-97.html
20. Bebop's BYTES Back. *Claude Shannon's master's Thesis*. Available: http://www.maxmon.com/1938ad.htm
21. Hannah, Eric. *United States Patent 7309866: Cosmic ray detectors for integrated circuit chips*. Available: http://tinyurl.com/3ysdmk
22. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://dx.doi.org/10.1511/2001.40.3268 and mirrored at http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
23. A. Srivastava and K. Venkatapathy, “Design and Implementation of a Low Power Ternary Full Adder,” VLSI Design, vol. 4, no. 1, pp. 75-81, 1996. doi:10.1155/1996/94696. Available: http://jeff.tk/wiki/Image:Design_and_Implementation_of_a_Low_Power_Ternary_Full_Adder.pdf
24. J.T. Butler, Multiple-Valued Logic in VLSI, IEEE Computer Society Press Technology Series, Los Alamitos, California, 1991.
25. A.K. Jain, M.H. Abd-E1-Barr and R.J. Bolton, "A new structure for CMOS realization of MVL functions," International Journal of Electronics, vol. 74, no. 2, pp. 251-263, 1993.
26. S.L. Hurst, "Two decades of multiple valued logic--an invited tutorial," in Proceedings of IEEE International Symposium on Multiple-Valued Logic, p. 164, May 1988.
27. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
28. S.L. Hurst, "Multiple-valued logic--its status and its future," IEEE Transactions on Computers, vol. C-33, no. 12, pp. 1160-1179, December 1984.
29. A. P. Dhande and V. T. Ingole. Design And Implementation Of 2 Bit Ternary ALU Slice. SETIT 2005, 3rd International Conference: Science of Electronic, Technologies of Information and Telecommunications. March 17-21, 2005, Tunisia. Available: http://jeff.tk/wiki/Image:Dhande%2C_Ingole_-_Design_and_Implementation_of_a_2_Bit_Ternary_ALU_Slice.pdf
30. P.C.Balla & A.Antoniou "low power dissipation MOS ternary logic family" IEEE journal on solid state circuits Vol. Sc-19 no-5, P.739-749, October 1984.
31. D.I.porat "Three valued digital system" Proc.IEE Vol.116, No6, P.947-955, June 1969.
32. K.C.Smith "The prospects of multivalued logic technology & application view " IEEE transaction on

computer, Vol.-C -30, P-619-627 September 1981.
33. Chung-Yu-Wu"Design & application of pipelined dynamic CMOS ternary logic & simple ternary differential logic" IEEE journal on solid state circuits Vol.28, No-8, August 1993.
34. CS150. Berkeley EECS. *Bits, Bytes, Nibbles, and Words: Some definitions*. Available: http://inst.eecs.berkeley.edu/~cs150/sp98/lectures/week6_2/tsld002.htm
35. Slashdot. *Ternary Computing Revisited*. Available: http://slashdot.org/comments.pl?sid=23934
36. Sloppy. Slashdot | Ternary Computing Revisited. Monday November 19 2001. *Trits?* Available: http://slashdot.org/comments.pl?sid=23934&cid=2585807
37. Merrill, Roy D. *Ternary Logic in Digital Computers*. January 1965. Avaialble: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
38. Bowles, Gary-alexander. US Patent #5498980 Ternary/binary converter circuit (Publication Date: 03/12/1996). Available: http://www.freepatentsonline.com/5498980.html
39. Setun' W. H. Ware, S. N. Alexander, N. M. Astrahan, H. H. Goode, M. Rubinoff, P. Armer, L. Bers, H.d. Huskey, "Soviet computer technology - 1959," Communications of the ACM, pp. 149-150, 1960.. Available: http://jeff.tk/wiki/Image:Communications_of_the_ACM_-_Soviet_Computer_Technology_-_1959.pdf
40. Faden, David. Reverse Fad Productions: Flip. Available: http://www.revfad.com/flip.html
41. Crispin, M. Panda Programing. 1 April 2005. Network Working Group, Request for Comments: 4042. *UTF-9 and UTF-18 Efficient Transformation Formats of Unicode*. Available: http://www.ietf.org/rfc/rfc4042.txt RFC 4042
42. Aspinwall, Jim. eCoustics. *Hacking CPU Voltage to Speed Up Your PC*. Available: http://forum.ecoustics.com/bbs/messages/34579/147079.html
43. Engelhardt, Mike. *LTspice/SwitcherCAD III User's Manual*. Available: http://ltspice.linear.com/software/scad3.pdf
44. Howell, Louis and Raymond, Eric S. Available: http://jeff.tk/wiki/Trinary/Logic#TriINTERCAL_Manual:_5.5.2.1_UNARY_LOGICAL_OPERATORS
45. Connelly, Jeff. Trinary/Parts - Jeff.tk - First Purchase. Available: http://jeff.tk/wiki/Trinary/Parts#Shopping_List:_First_Purchase
46. All About Circuits. Producing negative supply rails - Urgent - All About Circuits. Available: http://forum.allaboutcircuits.com/showthread.php?t=10415
47. All About Circuits. negative supply - All About Circuits Available: http://forum.allaboutcircuits.com/showthread.php?t=876
48. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
49. Connelly, Jeff. Trinary Computer Architecture - Older Diagrams. Available: Image:Proposed Architecture 2.png, Image:Proposed architecture 1.png
50. Connelly, Jeff. Trinary/IO - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/IO
51. Lite-On Electronics Inc. Part No. LTL-30EHJ. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-30EHJ.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1057-ND
52. Lumex. T-5mm LED, 6 leaded, multi-colored, 636 nm AlInGoP Red/574 nm, AlInGoP Green BiColor, 470 nm Ultra Super Blue, Water Color Lens Datasheet. Part #SSL-LX5099SIUBSUGB. Available: http://rocky.digikey.com/weblib/Lumex/Web%20Data/SSL-LX5099SIUBSUGB1.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=67-1829-ND
53. Lite-On Electronics Inc. Part No. LTL-293SJW. Datasheet available: http://media.digikey.com/pdf/Data%20Sheets/Lite-On%20PDFs/LTL-293SJW.pdf . Digi-Key Product Page Available: http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail?name=160-1038-ND
54. Quicktar. Current limiting resistor calculator for LEDs. Available: http://www.quickar.com/noqbestledcalc.htm
55. Eigenratios of Self-Interpreters: The Mark II OISC Self-Interpreter. Available: http://eigenratios.blogspot.com/2006/09/mark-ii-oisc-self-interpreter.html
56. Connelly, Jeff. Public Git Hosting - trinary.git/tree - circuits/. Available: [http://repo.or.cz/w/trinary.git?a=tree;f=circuits

57. Carothers D. Christopher. Evolution of Intel Microprocessors: 1971 to 2007. Available: http://www.cs.rpi.edu/~chrisc/COURSES/CSCI-4250/SPRING-2004/slides/cpu.pdf
58. Connelly, Jeff. Jeff.tk - Trinary/Symbols/Tips. Available: http://jeff.tk/wiki/Trinary/Symbols/Tips
59. Grubb, Steve. Trinary.cc. Available: http://www.trinary.cc/
60. H. T. Mouftah May 1976 Proceedings of the sixth international symposium on Multiple-valued logic. *A study on the implementation of three-valued logic*. Available: http://jeff.tk/wiki/Image:P123-mouftah_Study_on_the_Implementation_of_Three-valued_Logic.pdf
61. Mouftah, H. T. A study on the implementation of three-valued logic H. T. Mouftah May 1976 Proceedings of the sixth international symposium on Multiple-valued logic. Available: http://jeff.tk/wiki/Image:P123-mouftah_Study_on_the_Implementation_of_Three-valued_Logic.pdf
62. D.I. Porat, "Three-valued digital systems", Proc, lEE, Vol. 116, No. 6, June 1969, pp. 947-954.
63. M. Yoeli and G. Rosenfeld, "Logical design of ternary switching circuits", IEEE Trans. Elect. Comp., Vol. EC-14, February 1965, pp. 19-29.
64. M. Bitran and M.J.O. Strutt, "Minimization of ternary logic and complete set of integrable circuits", Electron. and Commun., AE0, Band 25,No. 8, 1971, pp. 387-392.
65. R.S. Nutter and R.E. Swartwout, "A ternary logic minimization technique", Conference Record of the 1971 Symposium on the Theory and Applications of Multiple-valued Logic Design, May 1971, pp. i12~125.
66. Connelly, Jeff. Trinary/Unary Quick Reference - Jeff.tk. Available: http://jeff.tk/wiki/Trinary/Unary_Quick_Reference
67. Hyde, Randall. *Art of Assembly Language*. Available: http://webster.cs.ucr.edu/AoA/DOS/ch01/CH01-2.html
68. Howell, Louis and Raymond, Eric S. *The C-Intercal Supplemental Reference Manual*. 1992-01-18. Available: http://webster.cs.ucr.edu/AoA/DOS/ch01/CH01-2.html
69. Connelly, Jeff. *Unary Gates Inside Binary Gates*. 2001-11-17. Available: http://jeff.tk/bingates/
70. Grubb, Steve. Trinary.cc. Binary Operations. 2001. Available: http://www.trinary.cc/Tutorial/Algebra/Binary.htm
71. I. Halpern and M. Yoeli, "Ternary arithmetic unit", Proc. lEE, Vol. 115, No. i0, October 1968, pp. 1585-1588. Table II : Multiple input ternary operators
72. H.T. Mouftah and I.B. Jordan, "Integrated circuits for ternary logic", Proceedings of the 1974 International Symposium on Multiple-valued Logic, May 1974, pp. 285-302.
73. E.L. Post, "Introduction to a general theory of elementary propositions", Amer. J. Math., Vol. 43, 1921, pp. I~3-185.
74. J.B. Rosser and A.R. Turquette, "Many-valued logics", North-Holland Publishing Co., Amsterdam, 1952.
75. M. Yoeli and G. Rosenfeld, "Logical design of ternary switching circuits", IEEE Trans. Elect. Comp., Vol. EC-14, February 1965, pp. 19-29.
76. R. Vacca, "A three-valued system of logic and its applications to base three digital circuits", Proc. Intern. Conf. Inform. Processing, (UNESCO), June 1959, pp. 407-414.
77. H. Mine, T. Hasegawa, M. Ikeda and T. Shintani, "A construction of ternary logic circuits", Electron. Commun. in Japan, Vol. 51-C, No. 12, pp. 133-140.
78. Sobie, Rick. Troolean operators, Available: http://sci.tech-archive.net/Archive/sci.physics/2006-03/msg00869.html
79. Nynaeve. Blog Archive - The troolean strikes back. Available: http://www.nynaeve.net/?p=87
80. CSE 460 - Spring 2006, Boolean Algebra Definitions, Theorems, and Postulates, Available: http://www.arl.wustl.edu/~lockwood/class/cse460/ba.pdf]
81. Connelly, Jeff. Extensions:Trinary MediaWiki Extension. Available: http://jeff.tk/wiki/Extensions:Trinary
82. Allright, James. Balanced Ternary Web Page. Available: http://web.archive.org/web/20050211091401/http://perun.hscs.wmin.ac.uk/~jra/ternary/
83. W. Ahrens, *Mathematische Unterhaltungen und Spiele* **1** (Leipzig: Teubner, 1910), Section 3.4; H. Hermelink, *Janus* **65** (1978), 105-117
84. *Philos. Trans.* **34** (1726) 161-173

85. *The Philosophy of Arithmetic* (Edinburgh: 1817); see pages 33-34, 54, 64-65, 117, 150
86. *Computes Rendus Acad. Sci. Paris* **11** (1840), 789-798
87. J. Bharati, *Vedic Mathematics* (Delhi: Motilal Banarsidass, 1965)
88. *Mathematical Education* **5**, 3 (1989), 129-133
89. *Computes Rendus Acad. Sci. Paris* **11** (1840), 903-905
90. *American Mathematical Monthly* **57** (1950), 90-93
91. *High-speed Computing Devices*, Engineering Research Associates (McGraw-Hill, 1950), 287-289.
92. *Communications of the Association for Computing Machinery* **3** (1960), 149-150
93. Bhattacharjee, Abhijit. *A polar place value number system for computers and life in general*. Available: http://abhijit.info/tristate/tristate.html
94. H.T. Mouftah, K.C. Smith and Z.G. Vranesic Department of Electrical Engineering University of Toronto Toronto, Ontario, Canada. *Ternary Logic In a Positional Control System*. Available: http://jeff.tk/wiki/Image:P135-mouftah_Ternary_Logic_in_a_positional_control_system.pdf
95. Walker, John. Forumilab, August 19, 1996. Minus Zero. Available: http://www.fourmilab.ch/documents/univac/minuszero.html
96. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
97. Ternary computers: part I: motivation for ternary computers, International Symposium on Microarchitecture archive, Conference record of the 5th annual workshop on Microprogramming table of contents, Urbana, Illinois, 1972
98. Merrill, Roy D. *Ternary Logic in Digital Computers*. January 1965. Available: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
99. Connelly, Jeff. Full Adder Timing Diagram - Internal Signals I. Available: http://jeff.tk/wiki/Image:Full_Adder_Timing_Diagram_-_Internal_Signals_I.png
100. Connelly, Jeff. Full Adder Timing Diagram - Internal Signals II. Available: http://jeff.tk/wiki/Image:Full_Adder_Timing_Diagram_-_Internal_Signals_II.png
101. Connelly, Jeff. Full Adder Y Decoder Signals. Available: Full Adder Y Decoder Signals
102. Halleck, John (via email) and Eide, Leroy. Fast BT division-by-2 using "Just-in-Time Subtraction". Available: http://www.dyalog.dk/dfnsdws/n_JitSub.htm
103. Hayes, Brian. American Scientist: Computing Science: Third Base, 2001. Available: http://jeff.tk/w/index.php?title=Image:American_Scientist_Online_-_Third_Base.pdf
104. Frieder, Gideon. Part 1 - Motivation for Ternary Computers. Available: http://jeff.tk/wiki/Image:P83-frieder_Ternary_Computers_-_Part_1_-_Motivation_for_Ternary_Computers.pdf
105. Frieder, Gideon. Part 2 - Emulation of Ternary Computers. Available: http://jeff.tk/wiki/Image:P86-frieder_-_Ternary_Computers_-_Part_2_-_Emulation_of_Ternary_Computers.pdf
106. Setun' W. H. Ware, S. N. Alexander, N. M. Astrahan, H. H. Goode, M. Rubinoff, P. Armer, L. Bers, H.d. Huskey, "Soviet computer technology - 1959," Communications of the ACM, pp. 149-150, 1960.. Available: http://jeff.tk/wiki/Image:Communications_of_the_ACM_-_Soviet_Computer_Technology_-_1959.pdf
107. Chillet, Daniel et. al. Team: r2d2. Available: http://web.archive.org/web/20060514100747/http://www.inria.fr/rapportsactivite/RA2004/r2d22004/uid51.html
108. Hyperphysics: Magnetic Properties of Solids. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/magpr.html
109. Hyperphysics: Ferromagnetism. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/ferro.html
110. Porter, Harry. Harry Poter's Relay Computer. Last updated 2007-11-15. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/solids/ferro.html
111. Woodgears.ca: Binary marble adding machine. Available: http://woodgears.ca/marbleadd/
112. Phun — 2D Physics Sandbox — Home. Available: http://www.phunland.com/wiki/Home
113. mlittman. Youtube: Logic gates using toys. Available: http://www.youtube.com/watch?v=H-53TVR9EOw
114. Grubb Steve. Slashdot: Ternary Computing Revisited: Re:impractical circuits. 2001-11-19. Available: http://slashdot.org/comments.pl?sid=23934&cid=2586271
115. H. Chan, T. van Duzer, D. Erne. A tri-stable state Josephson device memory cell. Available:

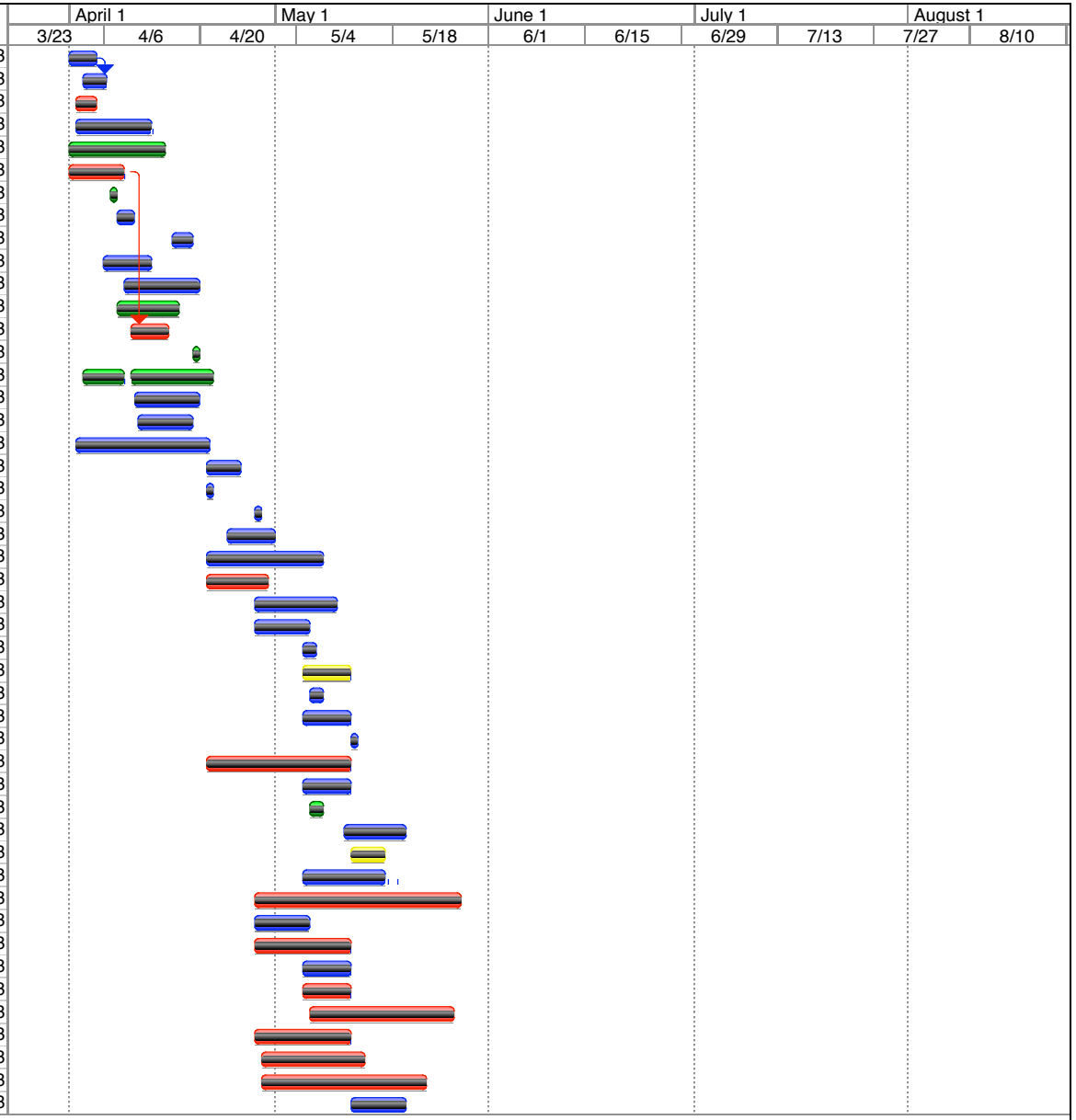
http://slashdot.org/comments.pl?sid=23934&cid=2586271
116. Merrill, Roy D. Ternary Logic in Digital Computers. January 1965. Available: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
117. A. E. Slade, "A Cryotron Memory Cell," Proc. IRE, Vol. 50, Jan. 62, p. 81.
118. C. F. Kooi, R. D. Merrill and H. H. Nakano, "Superconductive Ternary Information Storage Device," Lockheed Missiles & Space Co., *Research on Automatic* Computer Electronics, *Vol. I, RTD-TDR-63-4173, Oct. 31, 1963, pp. A-36-A-48.*
119. Yi, Jin. Institute of Physics. Ternary Optical Computing Architecture. Available for purchase: http://www.iop.org/EJ/abstract/1402-4896/2005/T118/025/
120. NewScientistTech. Ten weirdest computers, New Scientist, 18:30 11 April 2008 by Duncan Graham-Rowe. Available: http://technology.newscientist.com/article/dn13656-ten-weirdest-computers.html?DCMP=ILC-hmts&nsref=news1_head
121. Grubb, Steve. Trinary.cc: Transistor Models. Available: http://www.trinary.cc/Tutorial/Transistors/Transistors.htm
122. Connelly, Jeff. Jeff.tk - Trinary/LTspice tips. Available: http://jeff.tk/wiki/Trinary/LTspice
123. Linear Technology — Design Simulation and Device Models. Available: http://www.linear.com/designtools/software/
124. Patel, Chirag. Jeff.tk - Trinary/VLSI. Available: http://jeff.tk/wiki/Trinary/VLSI
125. Connelly, Jeff. Additional files for LTspice (SwitcherCAD III). Available: http://jeff.tk/wiki/Image:SwCADIII-jc.zip
126. Connelly, Jeff. repo.or.cz / trinary.git. Git repository. Available: http://repo.or.cz/w/trinary.git
127. Maxim. Application Note 638: Selecting the Right CMOS Analog Switch. Available: http://www.maxim-ic.com/appnotes.cfm/appnote_number/638
128. Fairchild Semiconductor. CD4066BC Quad Bilateral Switch. November 1983, Revised October 2005. Available http://www.fairchildsemi.com/ds/CD/CD4066BC.pdf
129. Maxim. Low-Voltage, Quad, SPST CMOS Analog Switches. 19-4793; Rev 5; 6/07. Available http://www.fairchildsemi.com/ds/CD/CD4066BC.pdf
130. Hussein T. Mouftah, Ternary Logic Circuits with CMOS Integrated Circuits, United States Patent 4,107,549. August 15th, 1978. Figure 1: Ternary Inverters. Available: http://jeff.tk/wiki/Image:Mouftah-1-Ternary_Inverters.png
131. Patel, Chirag and Connelly, Jeff. Gates with pull-middle resistors other than 12 kΩ.zip. Available: http://jeff.tk/wiki/Image:Gates_with_pull-middle_resistors_other_than_12k.zip
132. Hussein T. Mouftah, Ternary Logic Circuits with CMOS Integrated Circuits, United States Patent 4,107,549. August 15th, 1978. Figure 12: Cycling Gates. Available: http://jeff.tk/wiki/Image:Mouftah-12-Cycling_Gates.png
133. Hussein T. Mouftah, Ternary Logic Circuits with CMOS Integrated Circuits, United States Patent 4,107,549. August 15th, 1978. Figure 3: Ternary NAND. Available: http://jeff.tk/wiki/Image:Mouftah-3-Ternary_NAND.png
134. D.E. Knuth, The Art of Computer Programming - Volume 2: Seminumerical Algorithms, pp. 207-208. Addison-Wesley, 3rd ed., 1998. ISBN 0-201-89684-2. Available: http://jeff.tk/wiki/Image:Knuth-TaoCPVol2-pg207%2C8.pdf
135. Merrill, Roy D. Ternary Logic in Digital Computers. January 1965. Available: http://jeff.tk/wiki/Image:A6-merrill_Ternary_Logic_in_Digital_Computers.pdf
136. Hussein T. Mouftah, Ternary Logic Circuits with CMOS Integrated Circuits, United States Patent 4,107,549. August 15th, 1978. Available: http://jeff.tk/wiki/Image:Mouftah_-_Ternary_Gate_Figures.pdf
137. A. Srivastava and K. Venkatapathy, “Design and Implementation of a Low Power Ternary Full Adder,” VLSI Design, vol. 4, no. 1, pp. 75-81, 1996. doi:10.1155/1996/94696. Available: http://jeff.tk/wiki/Image:Design_and_Implementation_of_a_Low_Power_Ternary_Full_Adder.pdf
138. Grubb, Steve. Trinary.cc - Multiplexers. Available: http://www.trinary.cc/Tutorial/Mux/Mux.htm
139. Bowles, Gary-alexander. US Patent #5498980 Ternary/binary converter circuit (Publication Date: 03/12/1996). Available: http://www.freepatentsonline.com/5498980.html
140. Electronic Letters. 17th of October, 1974. Volume 10, Number 21. Implementation of 3-Valued Logic with COS/MOS Integrated Circuits. Available:

http://jeff.tk/wiki/Image:ELECTRONICS_LETTERS_17th_October_1974_Vol._10_No._21_-_IMPLEMENTATION_OF_3-VALUED_LOGIC_WITH_C.O.S._M.O.S._INTEGRATED_CIRCUITS.pdf
141. Connelly, Jeff. Trinary/Parts - Jeff.tk Available: http://jeff.tk/wiki/Trinary/Parts
142. FreePCB: freeware PCB layout software. Available: http://www.freepcb.com/. Tips: http://jeff.tk/wiki/Trinary/FreePCB
143. Connelly, Jeff. Public Git Hosting - trinary.git/summary. Available: http://repo.or.cz/w/trinary.git
144. NKK. Ultra-Miniature Slides SS Series Datasheet. Available: http://beta.octopart.com/NKK_Switches__SS14MDP2__0.pdf
145. Meaty. Gossamer Forum: FreePCB Footprints: Header holes too small (th_header.fpl). Available: http://www.freepcb.com/cgi-bin/gforum.cgi?post=3287;search_string=header%2040%20mil;#3287
146. Samtec | high-speed, rugged / power, micro interconnects. Available: http://www.samtec.com/

Appendix F: Schedule

Please see the following pages for a Gantt chart of our schedule for Spring through Summer, 2008, during which we worked on this senior project.

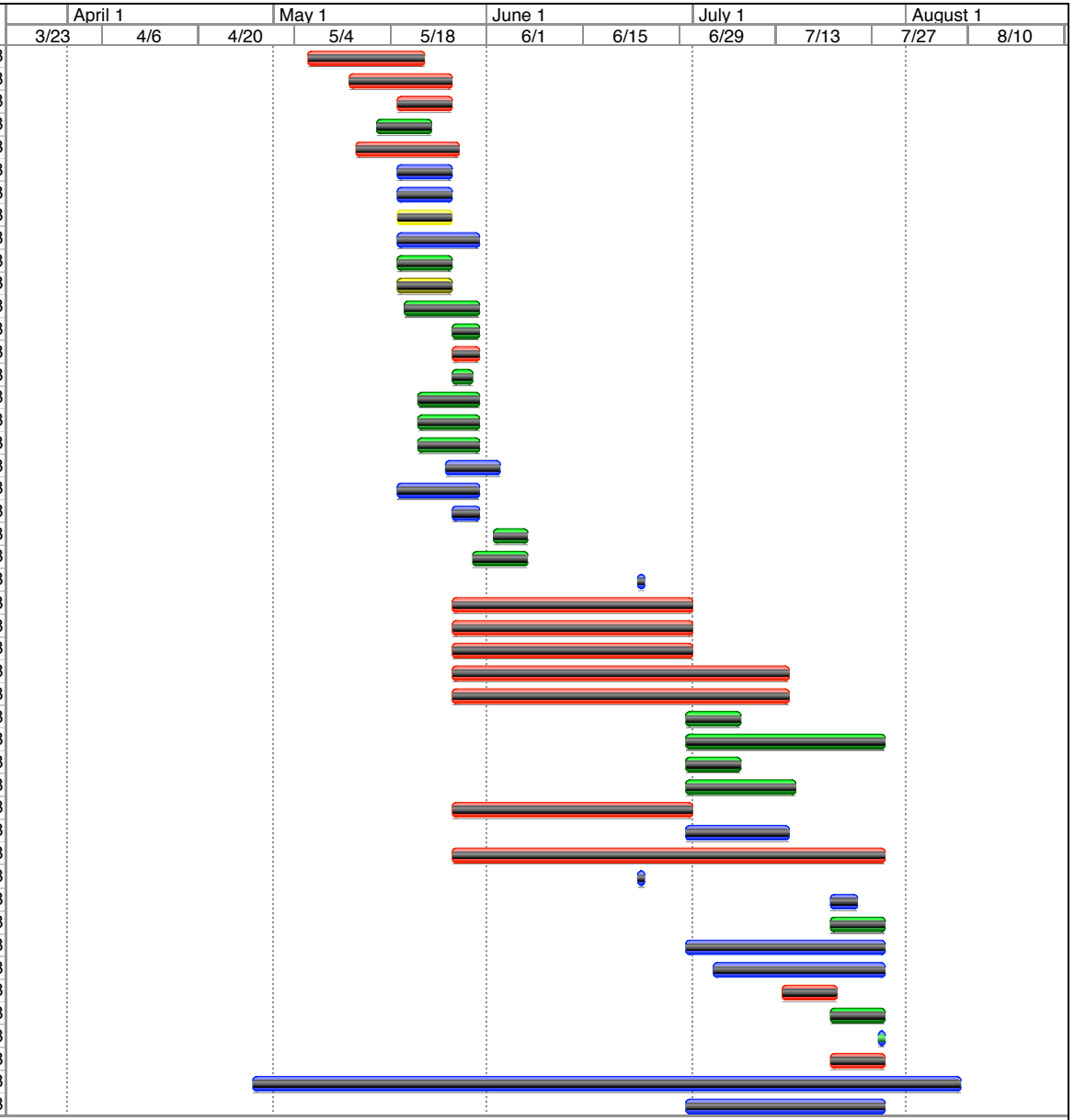
ID	Task Name	Duration	Start	Finish
1	Build all unary gates in lab	0.8 wks?	Tue 4/1/08	Fri 4/4/08
2	Build a 3:1 multiplexer	2.5 days?	Thu 4/3/08	Mon 4/7/08
3	Antonio learn Ltspice	3 days?	Wed 4/2/08	Fri 4/4/08
4	Chirag learn and setup git	10 days?	Wed 4/2/08	Sun 4/13/08
5	Himanshu administrative tasks	13 days?	Tue 4/1/08	Mon 4/14/08
6	Expression evaluator working	1.4 wks?	Tue 4/1/08	Wed 4/9/08
7	Senior project proposal due	1 day?	Mon 4/7/08	Mon 4/7/08
8	Build dyadic gates in lab	0.5 wks?	Tue 4/8/08	Thu 4/10/08
9	Build ternary cycling gate	3 days?	Wed 4/16/08	Fri 4/18/08
10	Wiki server RAID 0+1->1 disk upgrades	7 days?	Sun 4/6/08	Sat 4/12/08
11	Honors Open House - decide on timeslot	11 days?	Wed 4/9/08	Sat 4/19/08
12	Honors Open House Poster Due	9 days?	Tue 4/8/08	Wed 4/16/08
13	Instruction-level simulator working	5.5 days?	Thu 4/10/08	Tue 4/15/08
14	Honors Open House Presentation 1:30-2	1 day?	Sat 4/19/08	Sat 4/19/08
15	Simulate all tri-flop circuits	16 days?	Thu 4/3/08	Mon 4/21/08
16	Build dtflop-ms tri-flop circuit	9.5 days?	Thu 4/10/08	Sat 4/19/08
17	Circuit layout program functional	8 days?	Fri 4/11/08	Fri 4/18/08
18	Order and receive purchase #3	17.5 days?	Wed 4/2/08	Mon 4/21/08
19	Design SP3T footprint for FreePCB	5 days?	Mon 4/21/08	Fri 4/25/08
20	Label all nodes in LTspice properly	1 day?	Mon 4/21/08	Mon 4/21/08
21	Decide about reading textbook	1 day?	Mon 4/28/08	Mon 4/28/08
22	SWROM PCB layout and routing	5 days?	Thu 4/24/08	Wed 4/30/08
23	Design and simulate ALU	13 days?	Mon 4/21/08	Wed 5/7/08
24	Expression creator functional	7 days	Mon 4/21/08	Tue 4/29/08
25	Order PCB for 9-bit SWROM	10 days?	Mon 4/28/08	Fri 5/9/08
26	Retest dtflop-ms in lab	6 days?	Mon 4/28/08	Mon 5/5/08
27	Layout memory board	2 days?	Mon 5/5/08	Tue 5/6/08
28	CMP instruction working	6 days?	Mon 5/5/08	Mon 5/12/08
29	Order PCB of memory board	2 days?	Tue 5/6/08	Wed 5/7/08
30	Layout mux board	6 days?	Mon 5/5/08	Mon 5/12/08
31	Order PCB of control board	1 day?	Mon 5/12/08	Mon 5/12/08
32	Base 3 converter functional	16 days	Mon 4/21/08	Mon 5/12/08
33	Chirag install git on MacBook	6 days?	Mon 5/5/08	Mon 5/12/08
34	Honors Progress Report 2	2 days	Tue 5/6/08	Wed 5/7/08
35	Layout PCB of logic board	7 days?	Sun 5/11/08	Mon 5/19/08
36	BE instruction working	5 days?	Mon 5/12/08	Fri 5/16/08
37	Layout ALU board	11 days?	Mon 5/5/08	Mon 5/19/08
38	3-trit high-level language specification	23 days?	Mon 4/28/08	Tue 5/27/08
39	Jeff - Ch4: Machine Language	6 days?	Mon 4/28/08	Mon 5/5/08
40	Antonio - Ch7: VM I: Stack Arithmetic	11 days?	Mon 4/28/08	Mon 5/12/08
41	Jeff - Ch5: Computer Architecture	6 days?	Mon 5/5/08	Mon 5/12/08
42	Antonio - Ch8: VM II: Program Control	6 days?	Mon 5/5/08	Mon 5/12/08
43	Antonio - Ch9: High-level Language	16 days?	Tue 5/6/08	Mon 5/26/08
44	eval.py return 0, i, 1	11 days?	Mon 4/28/08	Mon 5/12/08
45	eval.py, allow literals 0, i, 1	12 days?	Tue 4/29/08	Tue 5/13/08
46	eval.py, allow trit vectors as inputs	19 days?	Tue 4/29/08	Thu 5/22/08
47	Jeff - Ch6: Assembler	6 days?	Mon 5/12/08	Mon 5/19/08



Project: Spring 2008 Schedule~671
Date: Tue 8/26/08

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

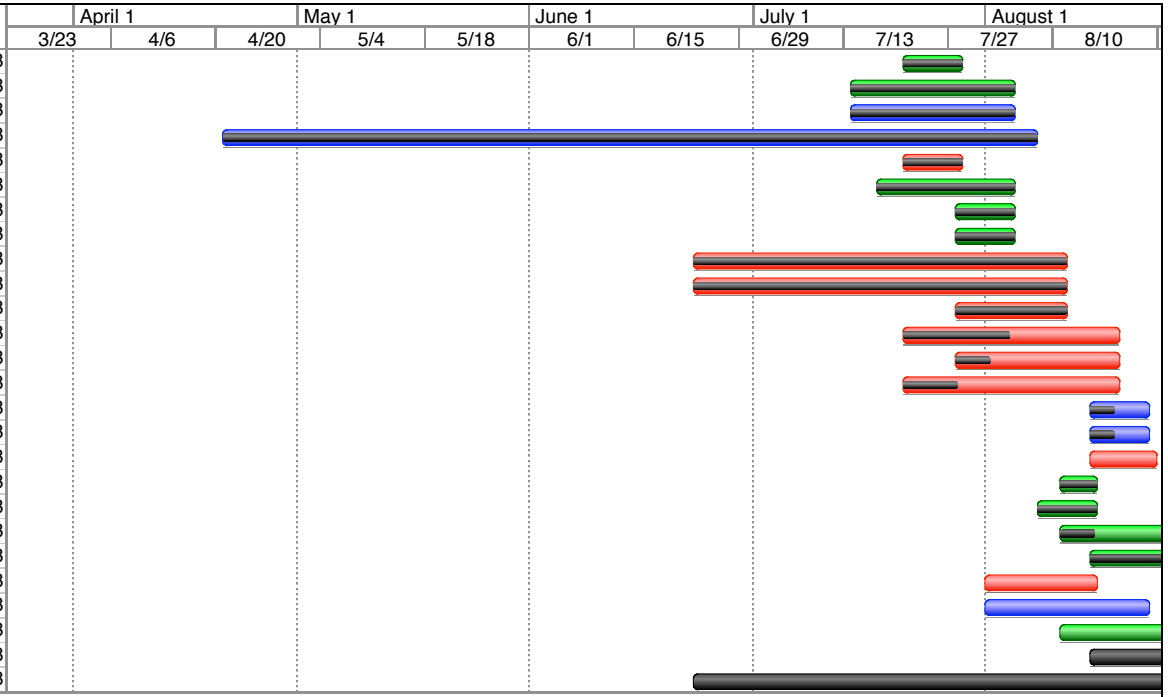
ID	Task Name	Duration	Start	Finish
48	CPU simulation, allow graceful termination	14 days?	Tue 5/6/08	Thu 5/22/08
49	Antonio - Ch10: Compiler I: Syntax Analysis	11 days?	Mon 5/12/08	Mon 5/26/08
50	Antonio - Enroll in senior project summer	6 days	Mon 5/19/08	Mon 5/26/08
51	Senior project spec & schedule draft due	6 days?	Fri 5/16/08	Fri 5/23/08
52	Antonio - Ch11: Compiler II: Code Generati	11 days?	Tue 5/13/08	Tue 5/27/08
53	Find header to add to Trinary/Parts	6 days?	Mon 5/19/08	Mon 5/26/08
54	Find jumper wires to add to Trinary/Parts	6 days	Mon 5/19/08	Mon 5/26/08
55	Demonstrate simulation of guess.t	6 days?	Mon 5/19/08	Mon 5/26/08
56	Jeff - Enroll in senior project summer	10 days	Mon 5/19/08	Fri 5/30/08
57	Design adder board	6 days?	Mon 5/19/08	Mon 5/26/08
58	Design sign board	6 days?	Mon 5/19/08	Mon 5/26/08
59	Prepare for CSC IAB Presentation	9 days?	Tue 5/20/08	Fri 5/30/08
60	Jeff - Make schedule for summer	4 days?	Tue 5/27/08	Fri 5/30/08
61	Antonio - Make schedule for summer	4 days?	Tue 5/27/08	Fri 5/30/08
62	Honors Progress Report Due	3 days?	Tue 5/27/08	Thu 5/29/08
63	Jeff - Ch1: Boolean Logic	7 days	Thu 5/22/08	Fri 5/30/08
64	Jeff - Ch2: Boolean Arithmetic	7 days	Thu 5/22/08	Fri 5/30/08
65	Jeff - Ch3: Sequential Logic	7 days	Thu 5/22/08	Fri 5/30/08
66	Antonio - Ch12: Operating System	6 days?	Mon 5/26/08	Mon 6/2/08
67	Chirag - Construct & text mux PCB	10 days	Mon 5/19/08	Fri 5/30/08
68	Chirag - Make schedule for summer	4 days?	Tue 5/27/08	Fri 5/30/08
69	Honors Final Report Due (9 pm)	5 days?	Mon 6/2/08	Fri 6/6/08
70	Senior project spec & schedule final due	6 days?	Fri 5/30/08	Fri 6/6/08
71	**Green=Jeff, Red=Antonio (below)**	1 day?	Mon 6/23/08	Mon 6/23/08
72	Define ASCII symbols for trinary	25 days?	Tue 5/27/08	Mon 6/30/08
73	Interface for base converter	25 days?	Tue 5/27/08	Mon 6/30/08
74	Base converter: int_cnvrt("111",3,10)	25 days?	Tue 5/27/08	Mon 6/30/08
75	Base converter: int_cnvrt("10i",-3,10)	35 days?	Tue 5/27/08	Mon 7/14/08
76	Base converter: to balanced trinary	35 days?	Tue 5/27/08	Mon 7/14/08
77	Clock board designed	6 days	Mon 6/30/08	Mon 7/7/08
78	Logic board designed	21 days?	Mon 6/30/08	Mon 7/28/08
79	Design clock generator circuit	6 days?	Mon 6/30/08	Mon 7/7/08
80	Interconnect diagram finished	12 days	Mon 6/30/08	Tue 7/15/08
81	Expr creator user interface	25 days?	Tue 5/27/08	Mon 6/30/08
82	Verify I/O board footprints	11 days?	Mon 6/30/08	Mon 7/14/08
83	Expr creator: replace names with expressio	45 days?	Tue 5/27/08	Mon 7/28/08
84	**Green=Jeff, Red=Antonio**	1 day?	Mon 6/23/08	Mon 6/23/08
85	Find pinout of DP3T chips	4 days?	Mon 7/21/08	Thu 7/24/08
86	TCA0 interconnects documented	6 days?	Mon 7/21/08	Mon 7/28/08
87	Construct and test dtflop-ms2 boards	21 days?	Mon 6/30/08	Mon 7/28/08
88	Build & test swrom-fixed	17 days?	Fri 7/4/08	Mon 7/28/08
89	Parser and tokenizer functional	6 days?	Mon 7/14/08	Mon 7/21/08
90	Code review eval.py	6 days	Mon 7/21/08	Mon 7/28/08
91	Define new game for TCA0	1 day?	Mon 7/28/08	Mon 7/28/08
92	Expr creator document how to use	6 days?	Mon 7/21/08	Mon 7/28/08
93	Chirag - Photoshop VTC curve	76 days?	Mon 4/28/08	Fri 8/8/08
94	Construct and test ALU boards or not	21 days?	Mon 6/30/08	Mon 7/28/08



Project: Spring 2008 Schedule~671
Date: Tue 8/26/08

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline

ID	Task Name	Duration	Start	Finish
95	CPE Analysis of Senior Project Design	6 days	Mon 7/21/08	Mon 7/28/08
96	Clean up website	16 days	Mon 7/14/08	Mon 8/4/08
97	Construct and test logic board	16 days?	Mon 7/14/08	Mon 8/4/08
98	All boards physically constructed	80 days?	Mon 4/21/08	Thu 8/7/08
99	Compiler structure planned	6 days?	Mon 7/21/08	Mon 7/28/08
100	CPE Report Rough Draft	12.5 days	Thu 7/17/08	Mon 8/4/08
101	Code review base_converter.py	6 days	Mon 7/28/08	Mon 8/4/08
102	Design output LED circuit	6 days?	Mon 7/28/08	Mon 8/4/08
103	Extended CPU assembler functional	36 days?	Mon 6/23/08	Mon 8/11/08
104	Extended CPU simulator functional	36 days?	Mon 6/23/08	Mon 8/11/08
105	Fix test case failure in eval.py	11 days?	Mon 7/28/08	Mon 8/11/08
106	ILOC generated	21 days?	Mon 7/21/08	Mon 8/18/08
107	ILOC -> trinary assembly	16 days?	Mon 7/28/08	Mon 8/18/08
108	Control flow graph generated	21 days?	Mon 7/21/08	Mon 8/18/08
109	Finish interconnection of all boards	6 days?	Fri 8/15/08	Fri 8/22/08
110	Test trinary output LED circuit	6 days?	Fri 8/15/08	Fri 8/22/08
111	Compiler register allocation complete	6 days?	Fri 8/15/08	Sat 8/23/08
112	CPE Report Final Draft	5 days?	Mon 8/11/08	Fri 8/15/08
113	Code review CPU simulator	6 days	Fri 8/8/08	Fri 8/15/08
114	CPE Senior Project Requirements	15 days	Mon 8/11/08	Fri 8/29/08
115	CPE Report Finished	11 days?	Fri 8/15/08	Fri 8/29/08
116	Compiler functional	11 days?	Fri 8/1/08	Fri 8/15/08
117	Entire system functional and working	16 days?	Fri 8/1/08	Fri 8/22/08
118	Submit CPE Report to Undergraduate Rese	14 days?	Mon 8/11/08	Thu 8/28/08
119	Last day of Cal Poly summer classes	11 days?	Fri 8/15/08	Fri 8/29/08
120	Honors funds expended by	61 days?	Mon 6/23/08	Mon 9/15/08



Project: Spring 2008 Schedule~671
Date: Tue 8/26/08

Task
Split
Progress
Milestone
Summary
Project Summary
External Tasks
External Milestone
Deadline