Viability of Ternary Computing as an Alternative to Binary Computing:

An investigation into theoretical and practical tenets

To what extent does ternary computing represent a viable alternative to binary computing considering both theoretical and practical aspects?

Subject: Computer Science

Word Count: 3995

To what extent does ternary computing represent a viable alternative to binary computing considering both theoretical and practical aspects?

Introduction

All information on most modern day electronic devices are represented in binary, namely a sequence of 0s and 1s. One could also represent the same information using a base of 10 bits or digits as is done in Arabic Numerals (0-9). One of the major reasons for using base 2 number systems instead of the customary base 10 system is that it is relatively easy to represent them in terms of electrical signals: no current maps to the 0 bit, and some current maps to a 1 bit. It is also easier to detect binary signals because as the number of bases increase, any sensor would have to detect many discrete voltage states to decipher the signal into its component bits. As a result, bases other than binary have remained largely unexplored for computing despite the certain theoretical advantages that they offer (Science Studio). Ternary (or trinary) computers, for example, have only been built once until now in the Soviet Union, that too in the 1950s. In essence, ternary logic operates on three states which can be: { -1, 0, 1} or {0,1,2} among others (Brown). This essay seeks to explore whether such a computer is a viable alternative to the traditional binary computer. The next section discuss Radix Economy, a concept fundamental to understanding the advantages of a ternary computer.

Radix Economy

Among the largest theoretical allures of ternary computing is that base 3 has the lowest average radix economy, a measure of the efficiency of representing numbers in a specific

base. When considering the efficiency of representing a specific number in a base, two factors have to be taken into account: number of symbols of the base(r), and number of digits in the number(w). Thus, to consider the most efficient base for a number, we can consider the product of the two factors. For instance, to calculate this product for the number 93 in base 10, we have r = 10 (0-9 symbols) and w = 2. Thus the product is 20. Graphing certain numbers we can see the most efficient base in which to define them.

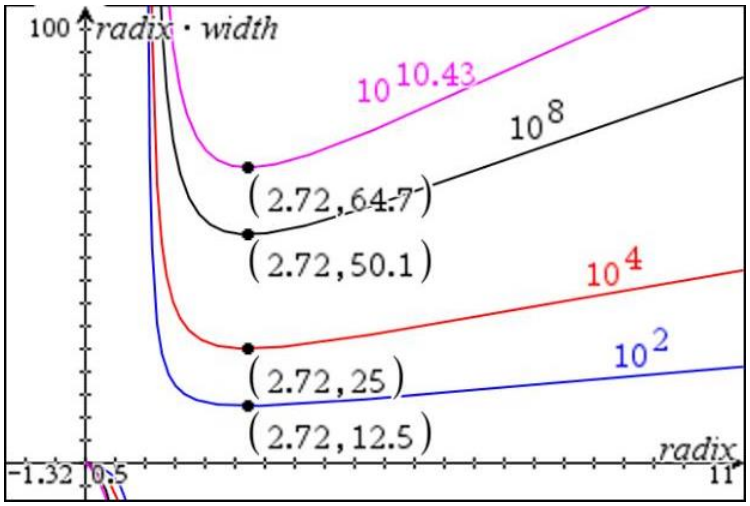


Figure 1. A graph showing the complexity of all radices in the range [1,11] for the numbers 10^2, 10^4, 10^8, $10^{10.43}$ along with the minima for each function.

In figure 1, the x-axis represents the radix, or base, in which the given number is expressed against its complexity, on the y-axis, which is defined as the product of its radix and width. The given minima of the radix value for each of these functions is 2.72, which is the rounding of the number e. The mathematical reasoning behind this is beyond the scope of the exploration. However, the result shown here is always true, e is the most efficient base. However, since designing components requires that numbers be represented in an integer base, as the graph shows, it is the value 3 that provides the most efficient integer solution(Hayes).

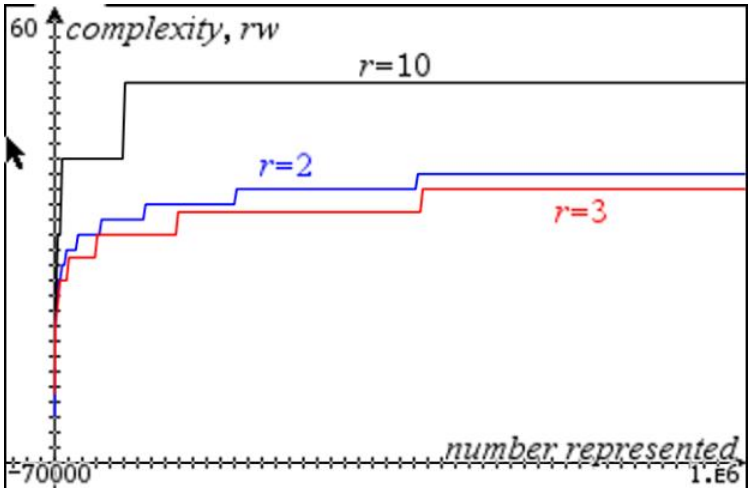


Figure 2. A graph comparing the complexity of the bases 2, 3 and 10 between the range 0 to 10^6

Figure 2 also makes abundantly clear the superiority of base 3 in terms of representing numbers as it shows a lower total cost and hence higher efficiency for a significant majority of values between 0 and 10^6(Pimentel).

While radix economy doesn't provide definite proof of the superiority of a specific base over another for computing as it doesn't factor in increasing complexity of hardware and implementation logics with higher bases, it does mathematically suggest that ternary computers can be more efficient at fundamental representations which may translate to greater speeds(Hayes). Thus, the mathematical properties of ternary merit investigation into its electronic applications.

Before delving into ternary logic in comparison to binary logic, we briefly explore the two types of ternary representation themselves: balanced ternary and unbalanced ternary. Balanced ternary is the most aesthetic representation of the three discrete states of ternary with the values {-1, 0, 1}. The benefits of balanced ternary are primarily the much easier

representation of negative numbers, subtraction, negation, and the logical abstraction to equate -1 with false, 0 with unknown and 1 with true. Another variation of ternary representation is the set {0, 1, 2} which is known as unbalanced ternary. The number representation provides with arithmetic logic being much more similar to binary with signed digits or complement subtraction(explained later). While there are other ways for ternary representations, these two are adequate to understand the basic ideas for the third state and certain differences between them(Brown). Since the simulation used to evaluate ternary computing is based on balanced ternary and it is generally the most unique and aesthetic, this exploration will seek to explain its basics only while referring to other types if necessary.

Ternary Logic Implementations

Adders

Half Adder:

Among the most basic functionality of computers based on which their overall efficiency is determined is an adder. Thus, we consider the half adder, full adder and the problem of a carry-look ahead adder in balanced ternary. Whenever two numbers are added, the two resulting digits are called the sum and carry. The sum represents the addition of the numbers and the carry is the remainder that gets carried over to the next position. Here we show the truth table for half adder that adds two trits(analogous to two bits in binary):

INPUT		**OUTPUT**	
A	B	C_{out}	S
-	-	-	+
-	0	0	-
-	+	0	0
0	-	0	-
0	0	0	0
0	+	0	+
+	-	0	0
+	0	0	+
+	+	+	-

Table 1. A table showing the truth table for a ternary half adder.

Let us consider the first row of addition along with it's decimal equivalents for familiarization:

$(-)_3 + (-)_3 = (-+)_3$ where + is the carry and - is the sum

In decimal: $-1_3 = -1 \times 3^0 = -1_{10}$ *and* $(-+)_3 = -1 \times 3^1 + 1 \times 3^0 = -3_{10} + 1_{10} = -2_{10}$

Therefore, $-1_{10} + -1_{10} = -2_{10}$ (Mansaf 3).

Similar to binary then we must consider the outputs required by carry and sum to determine which dyadic(two input) operators can be implied. We define two new different functions: consensus and sum that allow us to implement the logic for half adders. The consensus

operator defines the carry output and the sum operator defines the sum output. The truth tables for both are shown below:

C_{out}(Consensus)		B		
		-	**0**	+
A	-	-	0	0
	0	0	0	0
	+	0	0	+

Table 2. A truth table for the “Consensus” function

S(Sum)		B		
		-	**0**	+
A	-	+	-	0
	0	-	0	+
	+	0	+	-

Table 3. A truth table for the “Sum” function

Full adder:

Using multiple half adders, a ternary full adder can be built that takes 3 inputs to calculate the sum. An example row of the truth table is shown below:

A	B	C_{in}	C_{out}	S
+	+	-	+	0

Table 4: Sample row of truth table for ternary full adder

There are 27(3 cubed) of these combinations, and a ternary full adder that takes three inputs and returns the sum can be built using three half adders. If we combine multiple full adders, addition can be performed for any two numbers depending on the number of full adders available. An example for the addition of two 4 digit numbers is shown below which helps clarify the process through which multiple outputs from full adders are concatenated to give the sum.

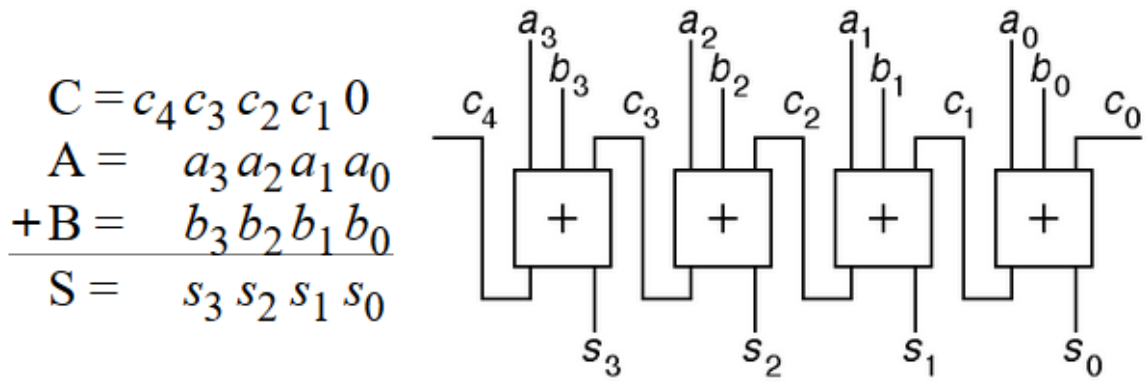


Figure 3. A diagram representing a full ripple carry ahead adder(Jones)

(Note: Each one of the "+" boxes represents a full adder circuit.)

This full adder, however, has a severe limitation. To compute s_1 from the figure above, c_1 is needed, which means that all the digits in the same positions have to be added sequentially. This means that the second adder has to wait for the first, and so on which slows down the entire addition process. This way the time taken is O(n) which means that the total time taken will be proportional to the number of inputs to the function. Therefore, in order to tackle the delay involved in such adders, all modern CPUs contain what are known as carry-lookahead adders which predict the carry for any specific adder without relying on the previous inputs. This way the adders work parallely and the sum is calculated in O(log n) time which is significantly faster. The problem of building a carry lookahead adder however remains

unsolved for balanced ternary which directly suggests that currently it is not a good choice to use balanced ternary to build an arithmetic logic unit. It is though fully possible right now to build a full adder using unbalanced ternary. According to Douglas Jones, however, “it appears that they will be more complex than unsigned ternary adders but just as fast” referring to the problem of building carry-lookahead adders for balanced ternary(Jones). Therefore, according to the analysis of basic functions ternary is as efficient theoretically as binary.

Benchmarking

To evaluate the current potential of ternary, a ternary emulator called Tunguska and a modern day computer will be compared. The reason for using a modern day computer is straightforward: we must be able to see how any ternary machine matches up with current consumer grade options. The decision to employ Tunguska is that among the few ternary emulators available, Tunguska is one that employs ternary logic at the most fundamental level possible with an emulator. While most softwares only simulate a ternary environment through the use of high level languages such as Python, Tunguska has its memory cell states also in three levels. Additionally, Tunguska is based off of balanced ternary which is the focus of this investigation, and therefore provides the closest available alternative to a ternary computer at the moment(Lofgren).

Hypothesis:

Considering the very limited research and software development in the field of ternary computing, the modern day consumer grade system will perform better than the ternary

computer by at least a factor of 3 in terms of the benchmark variable due to heavily optimized software and greater functionality in terms of employing computer resources.

Controls:

The testbench used to benchmark the ternary emulator will be kept constant with the following specifications:

Computer Component	**Specific Model**
CPU	Intel i5-6500(at 3.2 GHz)
Graphics	Intel HD Graphics 530(Built into CPU)
Motherboard	ASRock Z170 Pro4 Motherboard
RAM	8 GB of DDR4 Ram(at 2100 MHz)
Operating Software	Ubuntu 18.04 Desktop

Table 5: A table showing the specifications of the Testbench used during data collection

As one can probably see, considering the very low complexity of the programs benchmarked in terms of resource requirements, the testbench will provide excessive access to resources needed by the emulator and the native programs running the benchmarks, ensuring that there are no hardware bottlenecks affecting the results of the experiment. Furthermore, to ensure that the comparison between the ternary emulator and modern day computer is restricted to the differences in logic to the greatest extent, the program “ulimit” will be employed which would permit both the ternary emulator and native software to use only 20% of the CPU at

maximum and 1 GB of RAM. The reason for this is that the native binary software on the computer may use significantly more resources of the computer(such as using all 4 cores parallelly as opposed to the 2 cores that the ternary emulator may use), which would skew the results in favour of the binary software.

Rationale behind choice of Benchmarks:

For the comparison of the two systems, three different benchmarks will be carried out on both systems, while attempting to keep the programs as similar as possible. Firstly, the Mandelbrot set will be graphed with different iteration limits. While the mathematics behind the Mandelbrot set is beyond the scope of the question, essentially the Mandelbrot set uses multiple iterations in its calculations to graph a very aesthetic image known as a fractal. Fractals are extensively used in the graphics for modern games(Young). Therefore, the graphing of the Mandelbrot set provides the perfect opportunity to evaluate both the graphics processing and arithmetic processing. Secondly, an independent benchmark file has been created that performs a set of calculations involving basic arithmetic coupled with logarithms, square root functions, and a randomizer. This benchmark will help better determine the relative speeds of the two systems. For both of these benchmarks, the run time will be the independent variable will be compared. Lastly, a comparison of the operations/second between the ternary emulator and the defined test bench will be done to evaluate the systems on a much more general and fundamental scale.

Processed Data and Analysis:

1. Mandelbrot Set

Platform	Trial 1(s)	Trial 2(s)	Trial 3(s)	Trial 4(s)	Trial 5(s)	Average(s)
Tunguska(50 iterations)	51.597	50.723	49.871	50.236	52.015	50.8884
Testbench(50 iterations)	3.481	3.732	3.316	3.774	3.803	3.6212
Tunguska(100 iterations)	68.044	68.034	68.291	70.612	68.429	68.682
Testbench(100 iterations)	4.193	4.322	3.987	4.016	4.256	4.1548
Tunguska(150 iterations)	87.353	82.932	84.531	86.492	84.355	85.1326
Testbench(150 iterations)	4.78	4.691	4.513	4.842	4.573	4.6798

Table 6: A table showing the time taken in seconds for Tunguska and the defined testbench to graph the Mandelbrot set for the iterations 50, 100 and 150, along with their respective averages

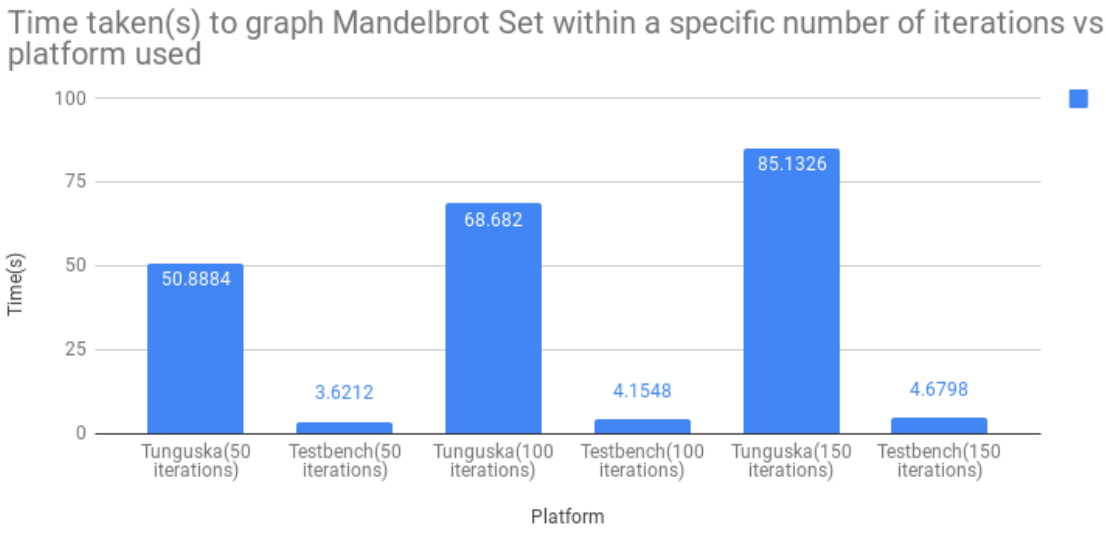


Figure 4. A bar graph depicting the time for Tunguska and the testbench to graph the Mandelbrot set for the iterations 50, 100 and 150.

From the bar graph above, the superiority of the testbench and consequently present day binary systems is apparent. In each case, the testbench performed faster by a factor of at least 14 which is a significant margin and indicates the current lack of feasibility of the system. Another interesting aspect of the result is the percentage increase in the time taken to graph the set when compared to graphing the set on the same platform. For instance, graphing 100 iterations vs 50 iterations on Tunguska led to approximately 26% greater time taken as opposed to a 13% increase in time taken by the testbench when graphing the same two iterations. This result may suggest that the relative performance of the two systems as the computations become increasingly complex or time consuming, would worsen with the ternary emulator. Therefore, according to the Mandelbrot Test Suite, it can be established that the current ternary emulator is not a viable alternative to the testbench for any graphical or computational purposes.

2. Indigenous Math based Test

Program	Platform	Trial 1(s)	Trial 2(s)	Trial 3(s)	Trial 4(s)	Trial 5(s)	Average(s)
Indigenous Math based Test	Tunguska	13.163	13.641	13.405	13.58	13.573	13.4724
Indigenous Math based Test	Testbench	0.00373	0.004465	0.005133	0.004488	0.003847	0.0043326

Table 7: A table displaying the values for the runtime in seconds for the independently developed Math test along with their averages on Tunguska and the Testbench in seconds

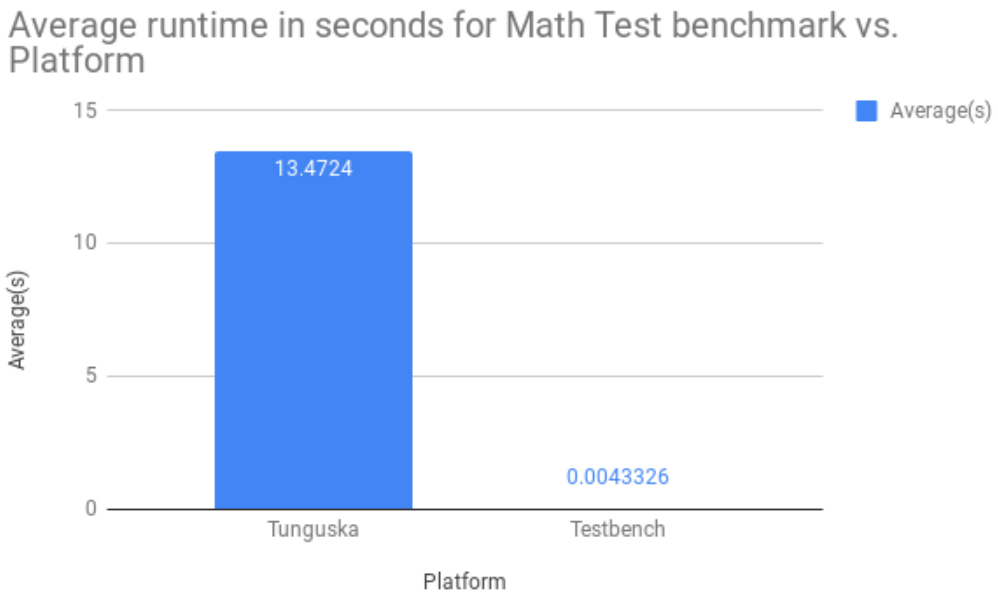


Figure 5. A bar graph illustrating the average runtime for the indigenous benchmark on Tunguska and the Testbench platform

The findings of the Math Test benchmark reinforce the inferences from the Mandelbrot test on a much larger scale, directly attesting to the much greater efficiency of the testbench. Running the code more than 3100 times faster after compilation on the testbench provides

strong evidence that for basic computations and certain math functions, the implementations in binary logic are unparalleled by a far margin. However, the difference in magnitude between the runtimes in this testbench may also be reflective of other unoptimized libraries for functions in Tunguska as opposed to C. For instance, for the cosine function implementation in Tunguska involves searching through an array of size 729 which is likely to make it take much longer to compute the cosine of functions as opposed to the heavily optimized libraries in programming languages such as C which have had years of dedicated software development and improvements designed to utilize the CPU resources much more effectively.

3. Floating Point Operations Per Second

Benchmark	**Platform**	**Trial 1**	**Trial 2**	**Trial 3**	**Trial 4**	**Trial 5**	**Average(GFLOPS)**
Native Benchmark	Tunguska	1.608504	1.602823	1.618952	1.613881	1.634409	1.6157138
Intel Linpack	Testbench (4 cores)	126.6893	125.6516	127.3717	127.447	119.9661	125.42514
Intel Linpack	Testbench (1 core)	48.4416	39.9253	48.0616	48.4783	48.496	46.68056

Table 8. A table giving the raw data for the number of floating point operations carried out by the CPU in GFLOPS for the Tunguska platform, the testbench with all its cores enabled and the testbench with a single core enabled along with their respective averages

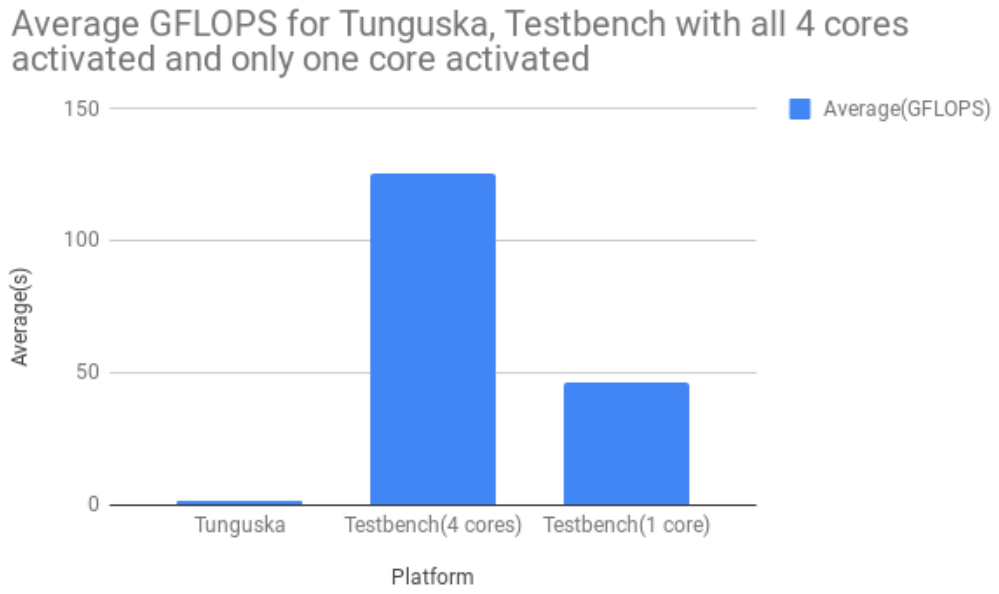


Figure 6. A bar graph depicting the average GFLOPS for each platform

Before delving into the data, we must understand the metric: Floating point operations per second. FLOPS are standard benchmarks for a computer processor to measure computing power. A floating point number is a number written in scientific notation with only one significant figure before the decimal point and the magnitude of the number with an associated power of the base. (“Gigaflops”) Essentially, an operation in this context tends to refer to basic arithmetic calculations such as addition and multiplication. As such the operations are not rigorously defined and even the magnitude of the numbers as determined by the number of bits or trits may not be constant(Nordhaus 5). Therefore, the comparisons between systems is not exactly similar and doesn’t offer an ideal benchmark but it can provide a rough idea for the relative performance of the systems.

The data derived from the benchmark reaffirms the vast advantage that the testbench would give you over Tunguska at present, especially with all the cores activated. Tunguska however only uses one core of the CPU in its emulates and hence to compare performance one could

also compare it with the testbench with a single core activated. We see the value drop accordingly with the reduction in cores, but the single core i5-6500 still far outperforms Tunguska, in the number of GFLOPS.

However, the GFLOPS metric lends itself to other very interesting analysis. Over the years, various systems have been benchmarked for their GFLOPS and two graphs showing the development from the late 1950s supercomputers onwards can allow us to place the rough development of Tunguska in comparison to the systems it would be equivalent to in terms of performance. This analyses would better place in context how far a single ternary emulator built by one software engineer has reached, albeit with a lot of the learnings and optimizations that have come to computer systems in general over the past 30-40 years.

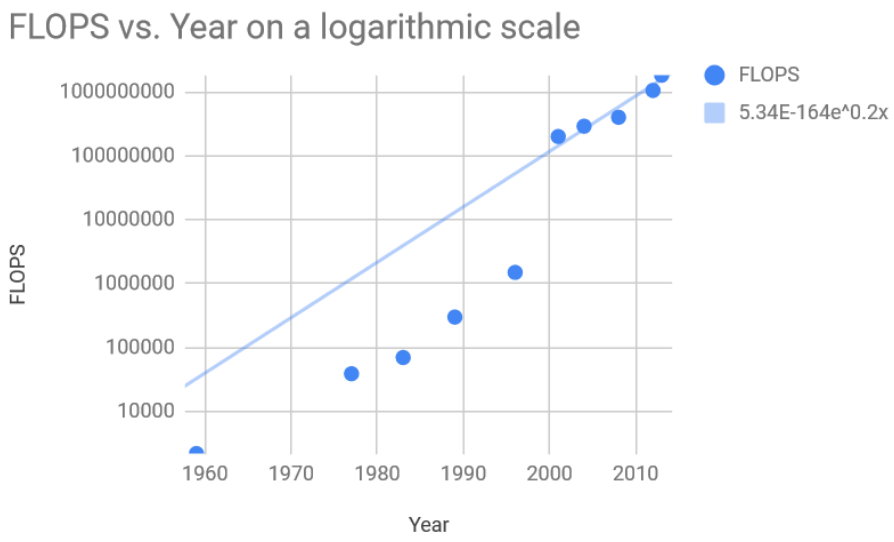


Figure 7. A scatter plot showing the increase in FLOPS over time on a logarithmic scale for specific devices according to their launch dates with an exponential trendline.(Note: All data can be found in the appendix and for some of the video game consoles benchmarked, the GPU was benchmarked in place of the CPU for FLOPS) (Kinghorn)(McCarthy)

According to the trendline given, the Tunguska GFLOPS benchmark would place it equivalent to a computer from 1986 which is reflective of the Tunguska specifications that claim the performance is around that of a computer from the 1980s(Lofgren). Thus, we can see that even though the modern consumer grade testbench trumps the ternary emulator by very wide margins in most cases, it is not as if Tunguska is primitive by any means. This could even be interpreted as the potential for Tunguska as the software itself was single handedly developed and has not had any industry funding for its optimizations.

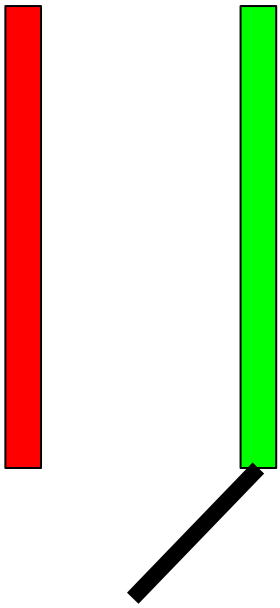
Ternary definitely does not provide an alternative at the moment and is significantly lagging in terms of any development. Therefore, a broader comment on the superiority of either of the logic systems(binary or ternary) can not be made from this gathering of data, and should be considered a survey of the present technologies.

Hardware Considerations

A vital aspect in determining the usability of any ternary computer system would be the benefits and drawbacks of its hardware relative to other systems. Hence, we consider possible ternary hardware implementations at the most basic levels such as the transistors themselves. A transistor is the basic element in an integrated circuit and modern day computers have over billions of transistors. Essentially, transistors act as switches or gates for electronic signals. The bases, {0 and 1} or {-1,0,1} are essentially just abstractions for voltages. The most straightforward abstraction is in binary where closed/off state on a specific transistor (no electrons flowing through) represents 0 and open/on represents 1. As the number of states increases thus so does the required precision with which every transistor must determine the voltage passing through it(“What is Transistor?”). This was the primary reason for the early

architects of computers to choose binary over earlier analog signals as with only two states the voltage thresholds required to be passed are relatively much larger in comparison to more than one states. For instance, if a transistor has to operate in base 10, it must be able to detect if only 1/10th of the current is passing through to represent 1, 2/10th to represent 2 and so forth. A direct consequence of the accuracy of the transistors to determine the voltage thresholds is the reliability of the information stored. If the transistors are inaccurate in their storage and transmission then this would lead to numerous errors leaving the device unreliable(Science Studio). This problem of reliability is further exacerbated with use of the electronic devices as the heat produced when transmitting signals through transistors leads to degradation of the semiconductors that are the primary materials for building the transistors. Therefore, heat dissipation is a very important concern before the design of any complicated electronic circuitry and binary provides a relatively low heat dissipation. (“Managing Heat for Electronics.”)

Binary transistor diagram



Ternary transistor diagram

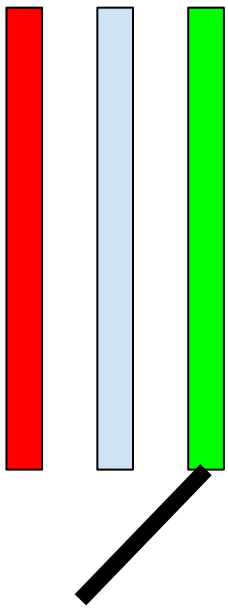


Figure 8. A comparison of the abstractions of binary and ternary transistor diagrams that help visualize the reasons that higher bases are more likely to be unreliable since the difference in voltage thresholds is significantly reduced which reduces reliability. In the diagram, green represent on, blue an arbitrary midway threshold and red represents off(Science Studio).

Even though there are considerable benefits to present day binary systems solely within the theoretical space and binary lends itself well to simpler implementations in terms of hardware, it is not without its bottlenecks. In 1956, co-founder of Intel, Gordon Moore, made an observation that the number of transistors on a chip would double every year, and Moore later modified his observation to a doubling every two years. This became known as Moore's law and the chip industry has roughly met this prediction for the past few decades. The increasing transistor density has played a vital role in the exponential improvement of computer performance and "has been the feedstock of exuberant innovation"(Simonite).

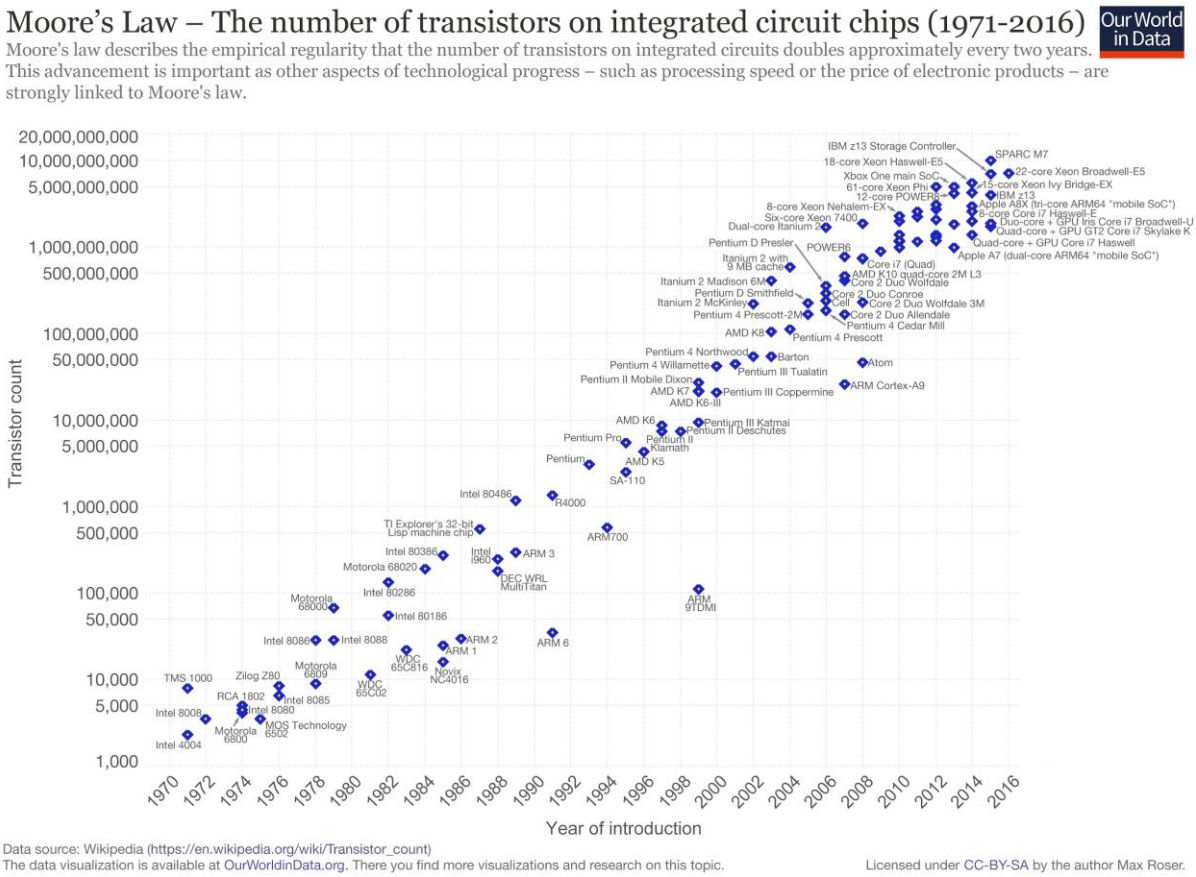


Figure 9. A scatter plot showing the number of transistors on integrated circuit chips from 1971 through 2016(“Technological Progress”)

As is clearly evidenced by the graph with a logarithmic scale on the vertical axis, shows that there has been a consistent exponential increase in the number of transistors on a circuit. However, there are some indicators that Moore’s law may be coming to an end in the near future as seen in the graph below. As evidence consider the scatter plot below.

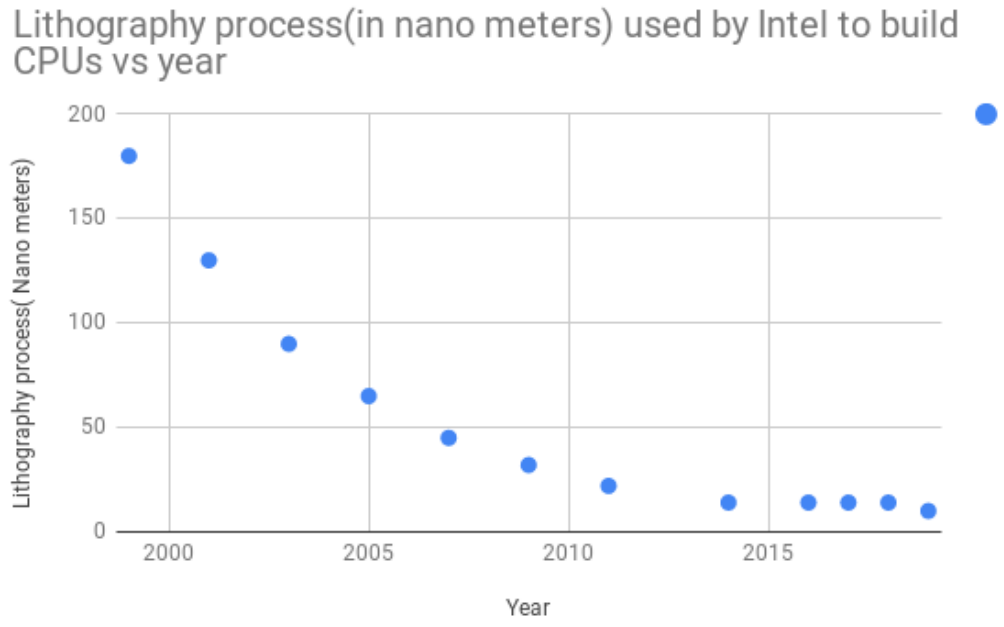


Figure 10. A scatter plot showing the change in the nanometer scale used in building Intel CPUs from 1999 onwards(Shilov).

The near end of Moore’s law here is substantiated through the graph above which displays a slowdown in the further shrinking of the lithography process suggesting that we are fairly close to the threshold values beyond which Moore’s law is likely to be inapplicable. For instance, transistors have since 2014 been built using a 14 nm lithography process for Intel CPUs. Furthermore, only recently Intel has delayed mass production for its CPUs that have a 10 nm based process for consumer chips citing economic and technical viability. Essentially,

at very tiny scales, that we are approaching electrons no longer behave as predictably as they do on slightly larger scales due to quantum effects and other physical phenomena. This would render the computer chips unreliable and therefore close down the main avenue through which CPU and other electronic component speeds have been increasing for more than the past half century(Computerphile).

Therefore, ternary logic offers a potential alternative to binary as it more efficient in terms of its radix economy and hence would theoretically, if optimized to the same level, perform faster than its binary counterpart due to its greater data density. Although replacing binary computers in their entirety seems unfeasible as binary systems obviously communicate significantly faster with other binary systems, ternary systems can exploit their rarity to their benefit as well. A recent study in cryptography has built a system that relies on balanced ternary to encrypt information. As the language of encoding and hardware structures of ternary represent a fundamental break from the “legacy structures” that cyber attackers exploit to their advantage. This radical shift in approach to cryptography would likely thus lead to a safer digital ecosystem(Cambou).

Conclusion

The primary objective of this essay was to elucidate the ternary logic based ecosystem and evaluate its practicality. Consequently, the research question, “To what extent does ternary computing represent a viable alternative to binary computing considering both theoretical and practical aspects?” was developed. As evidenced by the data collected, current ternary emulators do not represent a feasible alternative to the considerably more optimized binary systems with much more processing power. However, the radix economy based argument in favor of ternary still holds as the differences in development of ternary vs binary systems leave any practical comparisons heavily distorted and unreflective of the differences in logic. Furthermore, the theoretical aspects of the most basic components of electronic processors prove that their theoretical big O runtime as a function of inputs is equivalent to that of binary. Additionally, ternary serves as an attractive solution to the current bottlenecking and slug in growth of processing speeds for binary computers with the oncoming end of Moore’s law. The additional benefit of ternary over quantum computing and other such technologies in the short term is not only the huge amount of directly transferable knowledge between current binary and ternary systems but also the unviable nature of these upcoming technologies to be mass produced at reasonable costs for the common user(Science Studio). In conclusion, keeping in mind the challenges and opportunities with respect to the state of present day consumer grade computational systems, ternary systems do represent methods of improving efficiency and warrant significantly more research and development.

Works Cited

Alam, Mansaf. “Balanced-Ternary Logic for Improved and Advanced Computing”, International Journal of Computer Science and Information Technologies, 2014, Impact Factor 3.32, ISSN: 0975-9646. 5. 5157-5160.

Brown, David Scott. “Why Not Ternary Computers?” *Techopedia.com*, 31 May 2017, www.techopedia.com/why-not-ternary-computers/2/32427. Accessed 25 Jul. 2018

Cambou, Bertrand, et al. “Can Ternary Computing Improve Information Assurance?” *MDPI*, Multidisciplinary Digital Publishing Institute, 2 Mar. 2018, www.mdpi.com/2410-387X/2/1/6/htm. Accessed 20 May 2018.

Computerphile. “Is It the End for Moore's Law? - Computerphile.” *YouTube*, YouTube, 16 Apr. 2014, www.youtube.com/watch?v=X8v1BB0UaDs. Accessed Jun. 30 2018.

“Gigaflops.” *P2P (Peer To Peer) Definition*, techterms.com/definition/gigaflops. Accessed 13 Jul. 2018.

Hayes, Brian. “Third Base.” *Williams Web*, American Scientist, 2001, web.williams.edu/Mathematics/sjmiller/public_html/105Sp10/addcomments/Hayes_ThirdBase.htm. Accessed 15 May. 2018.

Intel.”Intel Linpack Benchmark Download”. *Intel Software Developer Zone,* 6 May 2012, https://software.intel.com/en-us/articles/intel-linpack-benchmark-download-license-agreement/. Accessed 10 Sept. 2018.

Jones, Douglas W. “The Ternary Manifesto.” *Douglas W. Jones on Ternary Computing*, University of Iowa, 1 Apr. 2012, homepage.divms.uiowa.edu/~jones/ternary/. Accessed 31 May. 2018.

Kinghorn, Donald. “Linpack Performance Haswell E (Core i7 5960X and 5930K).” *Puget Systems*,www.pugetsystems.com/labs/hpc/Linpack-performance-Haswell-E-Core-i7-5960X-and-5930K-594/. Accessed 17 Jul. 2018.

“Managing Heat for Electronics.” *NeuroImage*, Academic Press, 20 May 2005, www.sciencedirect.com/science/article/pii/S1369702105709354. Accessed 21 Jul. 2018.

Nordhaus, William D. “The Progress of Computing.” *Semantic Scholar*, Yale University, 30 Aug. 2001, pdfs.semanticscholar.org/ff50/594ded01032a73bafca66dde177e1127efda.pdf . Accessed 20 Jul. 2018.

Science Studio. “Why Do PCs Still Use Binary?” *YouTube*, YouTube, 8 July 2017, www.youtube.com/watch?v=XLIvVg8Vf_I. Accessed 29 Jun. 2018

Shilov, Anton. “Intel Delays Mass Production of 10 Nm CPUs to 2019.” *RSS*, AnandTech, 27 Apr. 2018, www.anandtech.com/show/12693/intel-delays-mass-production-of-10-nm-cpus-to-2019. Accessed 16 Jul. 2018.

Simonite, Tom. “The Foundation of the Computing Industry's Innovation Is Faltering. What Can Replace It?” *MIT Technology Review*, MIT Technology Review, 6 Feb. 2017, www.technologyreview.com/s/601441/moores-law-is-dead-now-what/. Accessed 23 Jul. 2018.

“Technological Progress.” *Our World in Data*, ourworldindata.org/technological-progress. Accessed 15 Jul. 2018.

McCarthy, Patrick. “Infographic: The Growth of Computer Processing Power.” *RECOIL OFFGRID*, 2 May 2017, www.offgridweb.com/preparation/infographic-the-growth-of-computer-processing-power/#. Accessed 13 Jul. 2018.

Pimentel, Fernando. “Zero Displacement Ternary Number System: the most economical way of representing numbers”, 2008, Revista de ciências da computação, 3. 50-58.

“What Is Transistor? - Definition from WhatIs.com.” *WhatIs.com*, whatis.techtarget.com/definition/transistor. Accessed 30 July 2018.

Young, Shamus. “What The Heck Is a Fractal and How Does It Apply to Games?” *The Escapist*, 21 Apr. 2015, www.escapistmagazine.com/articles/view/video-

games/columns/experienced-points/13809-Here-is-How-Fractals-Apply-to-Procedurally-Generated-Games. Accessed 31 July 2018.

Appendix

I. Program for Independently created Math Test

INDEPENDENT MATH TEST built on Tunguska(Lofgren):

```
void tester(){
    int x;
    int y;
    int z;
    int i;
```

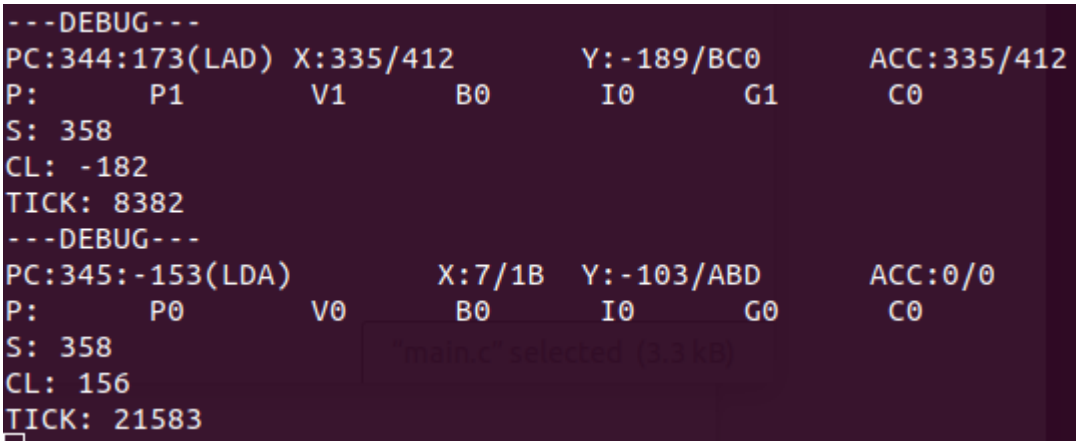
```
    struct machine_status* ms = get_ms();
    asm("DEBUG");
    for(i=0;i<500;i++){
            x = random(); y = random(); z = random();
            z =(int)(x*cos(y/2 + z*182)/364);
            z =(int)(y*log3i(x));
            z =(int)(z*log3i(sqrti(x*y+z)));
    }
    asm("DEBUG");
}
```

Very similar program adapted in C and tested

II. Sample Data Collection Process

1. Collection for Independent Math Test
    a. Tunguska

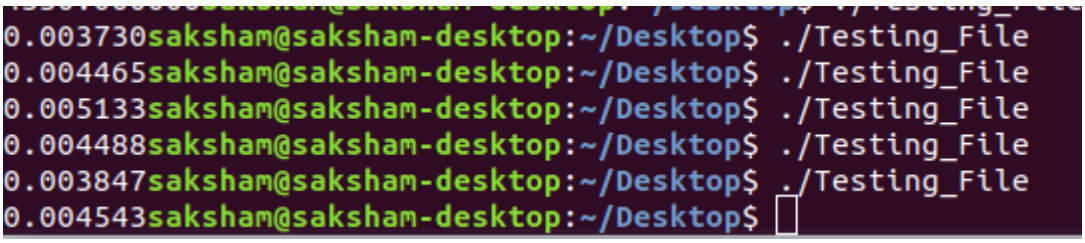
Time was calculated by subtracting the “TICK” value of the second “---DEBUG---” from the first. The “TICK” value is given in milliseconds and was accordingly adjusted in the raw data tables.



Appendix Figure 1. A screenshot of the “debug” functionality in Tunguska as shown on the Linux Terminal.

b. Testbench

The value shown in the left-most part of the section is the time in seconds it took to run the previous “Testing_File”



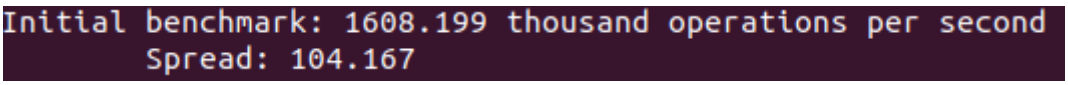
Appendix Figure 2. A screenshot of the time taken(in ms) to run the “Testing_File”

A similar process was adopted in data collection for the Mandelbrot Set.

2. Collection for Operations Per Second Benchmark

a. Tunguska

The value from the initial benchmark value when running Tunguska was taken.



Appendix Figure 3. A screenshot of the benchmark shown on the Linux Terminal immediately after opening Tunguska.

b. Testbench

The value for GFLOPS from one of the trials can be seen towards the bottom of the screenshot

```
Current date/time: Sun Sep 16 17:16:51 2018

CPU frequency:    3.599 GHz
Number of CPUs: 1
Number of cores: 1
Number of threads: 1

Parameters are set to:

Number of tests: 1
Number of equations to solve (problem size) : 7000
Leading dimension of array                  : 7000
Number of trials to run                     : 5
Data alignment value (in Kbytes)            : 4

Maximum memory requested that can be used=392144096, at the size=7000

================== Timing linear equation system solver ==================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
7000   7000   4      4.722      48.4416  4.957895e-11 3.557605e-02   pass
7000   7000   4      5.730      39.9253  4.957895e-11 3.557605e-02   pass
7000   7000   4      4.760      48.0616  4.957895e-11 3.557605e-02   pass
7000   7000   4      4.719      48.4783  4.957895e-11 3.557605e-02   pass
7000   7000   4      4.717      48.4960  4.957895e-11 3.557605e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
7000   7000   4       46.6806  48.4960

Residual checks PASSED

End of tests
```

Appendix Figure 4. A screenshot showing one of the results from Intel Linpack software running on the Testbench.

III. Compiled GFLOPS Data Table

Electronic Device	**Year**	**FLOPS**
IBM 7090	1959	220000

Atari 2600	1977	3900000
Nintendo Entertainment System	1983	7000000
Sega Genesis	1989	30000000
Nintendo 64	1996	150000000
Xbox	2001	20000000000
Intel(R) Pentium(R) 4 CPU	2004	29000000000
Core i7 920	2008	40000000000
Core i5 3570 (Ivy Bridge)	2012	105000000000
Core i7 4770K (Haswell)	2013	182000000000

Appendix Table 1. A table depicting the number of FLOPS for the CPU or GPU for multiple devices between 1959 and 2013(Kinghorn)(McCarthy).

Note: Table was compiled from both sources referenced.

IV. Data accessed for lithography process development at Intel

Year	**Nanometers**
1999	180
2001	130
2003	90
2005	65

2007	45
2009	32
2011	22
2014	14
2016	14
2017	14
2018	14
2019	10

Appendix Table 2. A table showing the size of the lithographic process used for the most recent consumer grade CPUs released at Intel. The tables include predictions for 2019(Shilov).

Note: The data has been taken as is from the referenced source.