CS247 Term Paper Index - Fall 2021

Dear CS247 Student,

Attached are the term papers for the entire class.

For the final exam, pick 5 papers (not including your own) that you choose to read and study thoroughly.

Special Note to Authors B & C: Do not pick either B or C as any of your selections.

Special Note to Authors I & L: Do not pick either I or L as any of your selections.

Special Note to Authors F & R: Do not pick either F or R as any of your selections.

Your final exam will consist only of questions from your chosen set of 5 papers.

Please email me your choice of papers by Monday, November 29th, 2021.

Simply provide the 5 letters corresponding to the papers you select in the body of the email.
(The title and author are not necessary - just the five letters).

HW3 is your (constructive) feedback (using the attached form) to the 5 authors of the papers you select.
(HW3 will not be "graded". Simply turning it in with helpful comments will earn you 100%)

Thank you,

-- Dr. Chun

IMPORTANT DATES:

November 29, 2021 - Selection of 5 papers to study for the final exam are due via email.
December 1, 2021 - Review to prepare for Final & Course Wrap-up (Synchronous Zoom at 4:00pm)
December 8, 2021 - Final Exam (Synchronous at 5:15pm to 7:30pm) and HW 3 due

A - Mohile,Shardul Sanjay – Global Semiconductor Shortage
B - Chang,Justin Hum & Chang,Chaz Hum (Team) – An Analysis of Quantum Computer Architecture
C - Goswami,Ikshaku – Quantum Computers – A Review
D - Domala,Sudhin Bhargava – Impact of High Performance Processors in Production & Animation
E - Singh,Inderpreet – Branch Prediction: A Retrospective
F - Ravi,Akshay – Design, Features and Architecture of the Leading Smart Chips in Autonomous Vehicles
G - Tong, Xiaoli - Overview of Cloud Computing Architecture
H - Shivayogi,Peeyusha – Multichip Modules and Their Performance
I - Kotla,Bhavya Reddy – Insights into Apple's Silicon Chips
J - Muthyala Sudhakar,Sriramm – Energy Efficient Mobile Cloud Computing
K - Aggarwal, Ishaan - Evolution of Virtualization
L - Yadav,Nishant – Analysis of Apple's SoC Processors
M - Dholakia,Pankti – Evolution of Tensor Processing Units and Comparative Performance Analysis
N - Balineni,Sriya – An Analysis of Trends in Obtaining Energy Efficient Data Centers
O - Shyamsundar,Pooja – Computer Architectures for Deep Neural Network-Based Computing
P - Zhong,Sida - Simple CPU Design
Q - Verma,Aneesh – In-depth Analysis of Spectre and Meltdown Vulnerabilities
R - Gupta,Srajan – Role of Computer Architecture in Autonomous Driving
S - Kale,Saurabh Satish – Nvidia's Turing Architecture – Changing the Game with Ray Tracing

A - Global Semiconductor Shortage

Shardul Sanjay Mohile
Computer Science Department
San Jose State University
San Jose, CA 95192
+1(669) 609-3591
shardul.mohile@gmail.com

ABSTRACT

The unexpected global pandemic made millions of people stay at home and brought about a transforming change in people's habits and lifestyles. This unique situation led to a boost in demand for electronics of all types, including mobile phones, laptops, speakers, personal computers, and others, which led to a staggering rise in the demand for microprocessors and microcontrollers, which in turn led to a semiconductor shortage on a global scale. In this paper, I have attempted to provide ways to alleviate the same. Since this is an ongoing problem, there is no specific single solution, however, measures can be taken to make it less severe.

1. INTRODUCTION

A semiconductor chip consists of a semiconductor wafer on which multiple transistors and other hardware components are wired together to create a device that can be used to perform specific tasks my managing and controlling the flow of electric current in gadgets and electronic devices. Modern devices are usually very complex in structure and hardware design and require an embedded system consisting of multiple semiconductor chips to function efficiently. When a high number of semiconductor chips are integrated together on a single board, usually made of silicon, an integrated circuit or an IC is formed.

That is why, by definition, an integrated circuit also known as a microchip, consists of two types of components. They are active components and passive components. Examples of active components are transistors and diodes whereas examples of passive components are capacitors and resistors. A combination of these active and passive components is fused onto a silicon wafer and the resulting circuit is called an integrated circuit or an IC.

This is also known as a monolithic chip. A semiconductor chip is usually very small and may even be microscopic in most cases whereas the size of the integrated circuit is sometimes of the order of square millimeters or in rare cases, square centimeters.

Semiconductor chips and integrated circuits are more pervasive than we think. Their usage is not limited to household gadgets and devices. Integrated circuits are the driving force behind most big and small machines. Some common examples of devices using integrated circuits are cars, toasters, air conditioners, televisions, remote controls, refrigerators, etcetera.

That is precisely why the shortage of semiconductors had a huge impact on most medium-scale and large-scale industries. During the global pandemic that started last year, the manufacturing sector took a huge hit because of the phenomenal rise in demand for gadgets, which led to a deficit in the required production in terms of keeping up with the demand for the gadgets in the market.

It is important to understand why the hit taken by the semiconductor manufacturing sector rapidly turned into a global problem that has still not been solved, however, immense efforts are being taken on a regular basis to come up with a foolproof solution to the same. A lot of conglomerates and industrial titans from most major sectors like technology, automobiles, construction, heavy-duty mechanical equipment, etcetera obtain their raw materials and small components from countries in Asia because Asian countries, owing to high population and many individuals belonging to low-income groups, offer some of the cheapest human labor in the world.

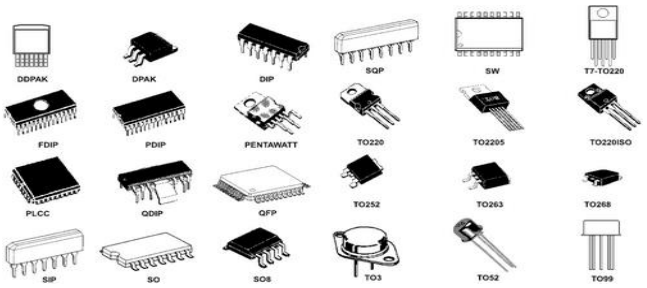


Figure 1. Integrated Circuits

As an example of this component outsourcing commonly used in the industrial sector, Apple, which is the company with the second highest market cap in the world after Microsoft, uses components manufactured in Germany, South Korea, India, China, Japan, Indonesia, Taiwan, Australia, Brazil, United Kingdom, and myriad other countries. If manufacturing of components in two or three of these places takes a hit, then the principal component cannot be assembled, and the production is massively delayed or may even fail.

This analogy can be used to understand how the semiconductor shortage turned out to be a global problem within a few months of the beginning of the pandemic and how continents like North America where a lot of large international companies are based, were struck down in terms of massive losses in revenue and in the case of mid-level companies, the manufacturing sector was even shut down in some cases.

Being a severe problem that had to be handled on a priority basis, there is an urgent need to address, assess, peruse, and resolve it on a priority basis.

2. HISTORY

The first microprocessor was invented by Intel in the year 1971 and the background to this invention is rooted in an exemplary assignment offered to Intel by Nippon in the year 1969, when the needed to equip their latest calculator with 12 custom-made semiconductor chips. After Intel were approached by Nippon Electronics, they suggested a lighter, cheaper, and faster solution which involved an integrated circuit made of only 4 chips instead of 12.

2.1 Intel® 4004

The world's first semiconductor chip is called the Intel® 4004 and is made up of four components which are together called as the MCS-4. The first component is the Central Processing Unit chip or the CPU chip, also known as the 4004. The second component is the standard Read-Only Memory or the ROM chip for the custom application programs. The third component is the Random-Access Memory or the RAM chip which is extremely useful in data processing and the fourth and final chip is the Shift-register chip which is efficiently utilized for the input/output port of the integrated circuit.

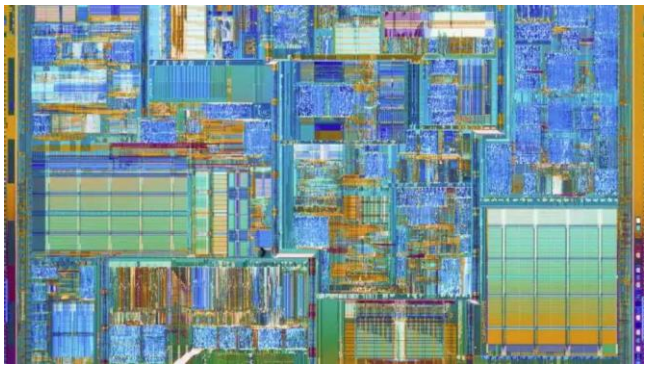


Figure 2. Intel® 4004

2.2 Intel® 8086

Another revolutionary addition to the semiconductor technology industry that brought about a massive transformation on computer architecture in general is the Intel® 8086 chip. This semiconductor chip was invented by Intel in the year 1978, and it gave rise to the x86 family of computer chips. The x86 is a group of instruction set architecture chips which is built based on the 8-bit Intel®8080 and the 16-bit Intel® 8086 chip.

The x86 family of semiconductor chips is one of Intel's is one of Intel's most successful ones and the suffix in the name comes from the suffix used for many microprocessors manufactured by Intel after it manufactured the 8086 processor.

The architecture of the 8086 consists of the Execution Unit or the EU and the Bus Interface Unit or the BIU. They are connected by an Internal Bus, which communicates with the Arithmetic Logic Unit and the Flags in the Execution Unit using a Temporary Register.

In addition to this, the Execution Unit consists of Data Registers, Pointer Registers, and Index Registers. On the other hand, the Bus Interface Unit consists of Segment Registers, a BUS Control Logic Unit and an Instruction Queue. The BUS Control Logic Unit interacts with other processes using the External Bus.

The Intel 8086 processor has an internal architecture that can be represented as follows:

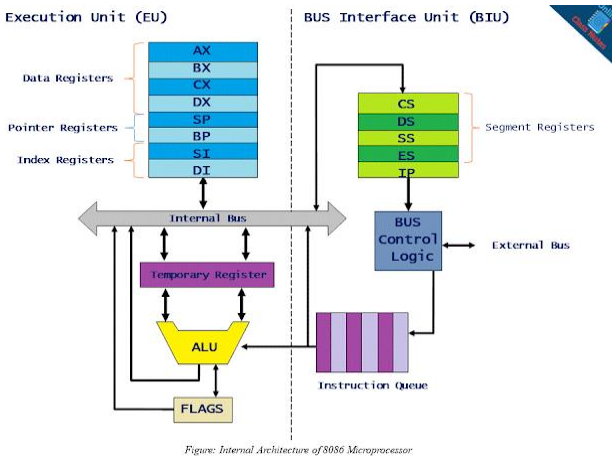


Figure 3. Internal Architecture of 8086 Microprocessor

In this way, phenomenal progress was made in a very short span of time, and this is testament to the fact that the introduction of microprocessors was a welcome change to the world of computer hardware architecture.

2.3 Intel® i9-12900K

The Intel® i9-12900K is the most powerful processor in the semiconductor market right now and is better in terms of both, performance and cost-effectiveness as compared to the best processor launched by Intel's market rival, Advanced Micro Devices or AMD, that is, the Ryzen 9 5950X.

Figure 4. Intel® i9-12900K

To conclude the historical overview for this paper, I would like to point out the stark differences in functionality, performance, and size between the first microprocessor in the world and the most powerful microprocessor in the world:

1. The Intel® 4004 was made up of 2300 transistors whereas the latest most powerful processors manufactured by Intel are made up of about 560 million transistors.
2. The circuit processing die of the Intel® 4004 processor is about 10,000 nanometers wide whereas the circuit processing die of the latest processor is just 32 nanometers wide.
3. The processing speed of the Intel® 4004 was of the order of a few kHz whereas the processing speed of modern-day processors if of the order of a few GHz.
4. Thus, modern day microprocessors are about 1 million times faster than the Intel® 4004.
5. Modern calculators are faster than the Intel® 4004 in terms of computing time.
6. The die size of the Intel® 4004 processor is 180 square nanometers whereas the die size of the latest processor by Intel is 180 square nanometers.
7. Having outlined major differences between the first processor in the world and the best processor in the world, let us proceed to understand the various methods that I have proposed to alleviate the semiconductor shortage problem that has plunged most industries of the world into inadequacy, heavy financial losses and thrust them out of the international, national, or local semiconductor markets.

3. PROBLEM-SOLVING APPROACHES

3.1 The Ostrich Algorithm

The Ostrich Algorithm is a problem-solving technique or strategy in computer science which states that one can ignore potential problems by assuming that the problem has an exceedingly low probability of occurrence.

In the current scenario of a global semiconductor shortage, the analogy of the above problem can be used by relating it to the action of 'burying one's head in the sand and pretending that there is no problem'.

This effectively means that this method suggests that the manufacturer make no change whatsoever in the way the semiconductor chips are being manufactured. This suggestion follows the concept of eventual consistency that is used in distributed systems, which implies that whenever an update is made in a distributed system, all data storing nodes will eventually be updated with the same value and there will be synchronization of data at some point of time.

The Ostrich Algorithm is usually used in the case of deadlock detection and prevention and is a feasible method to use if the cost of prevention of the problem is very high. A deadlock is defined as a situation in which a resource, say A, required by a process 1 is allocated by the system to another process 2 and another resource, say B, required by process B, is allocated to process A. This situation causes a deadlock which is like the concept of a stalemate that occurs in a game of chess when then there is no outcome among the possible list of outcomes that can take the desired process towards completion.

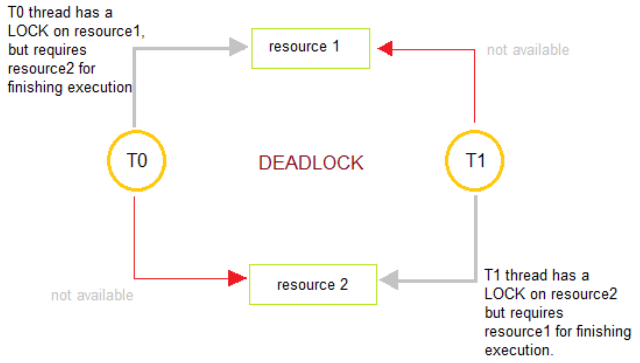


Figure 5. Deadlock

In the current situation, however, the problem is one that will aggravate, and demand will increase exponentially if no steps are taken to ramp up production of semiconductor chips, microprocessors, microcontrollers, and integrated circuits. Manufacturers will take a heavy hit in terms of revenue, customer satisfaction and overall brand value and this is precisely what makes this suggestion weak when it comes to dealing with a global problem of a pervasive type, that is the global semiconductor shortage problem.

3.2 Securing Commercially Available Parts

This approach suggests that the manufacturers try and secure the required parts from other companies. If company A purchases raw materials for manufacturing semiconductor chips from a company B and if company B is unable to provide the same, this approach suggests that company A approach other companies for obtaining said raw materials for the manufacturing process.

At first glance, this may look like a feasible and simple solution, but it comes with its own share of complications, both, architectural and legal.

Monopolization of parts in semiconductor chip manufacture is a very real concept the global semiconductor market. To cite an example for the same, let us take into consideration the Dutch company ASML or Advanced Semiconductor Material Lithography. ASML manufactures photolithographic machines which are used to emboss integrated circuits onto a semiconductor wafer. Being a relatively new company in the semiconductor market, ASML now owns 62 percent of the market share of manufacture of photolithographic machines and has surpassed the Japanese giants Canon and Nikon in the manufacture of the same.

Therefore, each company has a specific forte and area of specialization and it is not easy or convenient to merely look for another supplier in case one of them is unable to come up with a solution to the manufacturing requirements. This is the expertise-based problem attached to this solution.

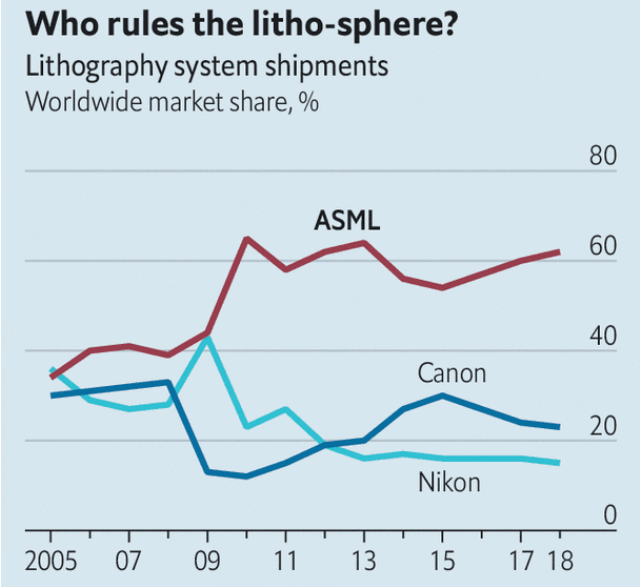


Figure 6. Monopoly in Semiconductor Manufacture

There is also the problem of compatibility when it comes to purchasing raw materials from a company that does not know the purchaser's manufacturing requirements well. Their products might be incompatible with the final finished product, that is the semiconductor chip, so this approach does not guarantee success.

Another aspect to this problem is the legal aspect. Companies with mutual agreements regarding supply of raw materials usually have binding legal agreements between them. Approaching a different company might entail heavy fines or lawsuits for suing the company breaking the legal agreement.

3.3 Redesigning the Process Control Block

The Process Control Block or the PCB is a data structure that stores information about a process. It consists of the following blocks:

1. Process ID: The unique identification number allocated to every process
2. Process State: The current state of the process, that is, whether the process is live, dead, undergoing execution, in a wait queue, aborted indefinitely or terminated
3. Pointer: It points to the address of the block where the information about the process is stored
4. Priority: It contains the priority number of the process, which enables the system to decide where the process execution stands in terms of the significance of its order of occurrence.
5. Program Counter: It is a register that contains the block address of the process that is next in queue for execution
6. Central Processing Unit Registers: They contain small but fast data storage and processing capabilities
7. Input/ Output Information: It keeps a log of the input information and the output information
8. Accounting Information: This part of the Process Control Block contains information about the processing time, latency, process turnaround time and other numeric data regarding the complete execution cycle of the processes

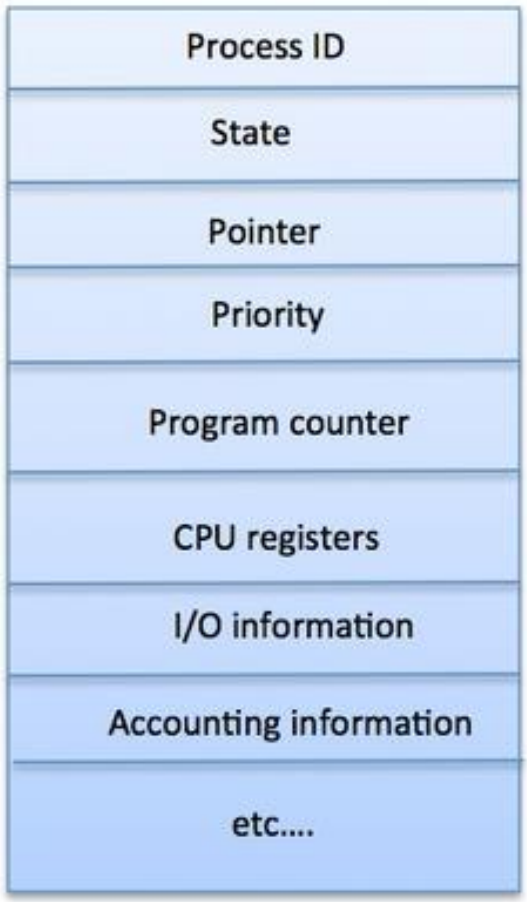


Figure 7. Process Control Block

Making modifications in the core structure of a chip would require modifications in the Process Control Block and its structure since it defines how a set of processes is executed. The Process ID, Process State, Program Counter, Pointers and Priority blocks store information about an individual process, so structural changes cannot be made in them.

However, changes can be made in the Central Processing Unit Register, which contains small and fast modules that improve the data processing capabilities of the semiconductor system. This solution is not unrealistic, unlike the previous two solutions.

Structural changes in the Central Processing Unit Registers will directly change the way the Central Processing Unit interacts with the main memory of the system and these changes will control how date is fetched from main memory and executed efficiently. This is better illustrated from the following figure which

highlights the role of a Central Processing Unit Register in a computer system.

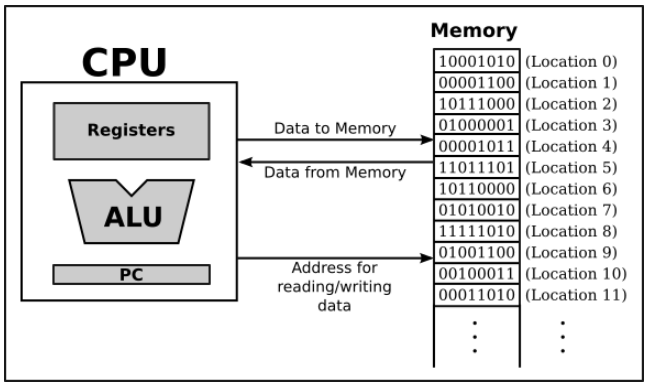


Figure 8. Significance of CPU Registers

3.4 Porting The Code

This is a proposed software solution to the global semiconductor shortage problem. Modern-day semiconductor chips are not directly manufactured. The hardware model of a chip is represented in the form of a code that is written in a hardware development language like VHDL. Hardware development languages are usually event-driven, and the code is simultaneously executed for all blocks of the code.

Code that is written for a specific purpose for a particular type or brand of hardware may not work well with hardware components of a different type. That is why, the hardware development language code needs to be modified to make it compatible with another type of hardware component. This process is known as porting of the code.

Any hardware development language code contains references to the architectural platform that it has been programmed to represent. These references need to be modified for the code to work well across a different architectural platform. Also, the code must be written in the highest possible hardware development language to ensure that it merely needs to be recompiled when it is implemented across multiple architectural platforms.

Here, it is important to note that porting refers only to the modification of code in the software in which it has been programmed and has no reference to the physical loading of the code into the hardware component in question.

4. CONCLUSION

Having cited four possible solutions to tackle the global semiconductor shortage, I would like to conclude that the most effective method is the part that discusses porting of the code. Porting the code allows the user to make a large variety of changes in the internal functioning of the semiconductor chip in question and is the most efficient solutions among the proposed ones.

Followed by this one is the part if the paper that discusses the altering of the Process Control Block. This solution is not as effective as compared to the previous one for the reason that there are not enough changes that can be made in a pre-existing data structure because its internal structure is already well-defined and established.

The monopolization of the semiconductor chip raw materials is the third most effective solution. Though it does not contain any kind of logical fallacy, it is realistically not feasible because of the multiple reasons mentioned in the paper.

The first solution pertaining to the Ostrich Algorithm is neither logical nor realistic when it comes to solving the problem even partly.

In this way, I have proposed four different types of possible solutions to the global semiconductor shortage problem and have analyzed them and concluded which ones of them are feasible enough to execute in the real world.

The porting of code is a software-based solution, altering the Process Control Block is a structural solution, searching the market for available components is a logical solution and the Ostrich Algorithm is an algorithm-based solution that revolves around a core concept in computer science, that is, decision-making.

As I have mentioned before, since this is an ongoing problem on a global scale, there is no single correct solution to solve it, however there can be multiple incorrect ones. Efforts are being made daily all over the world to alleviate the problem. The curve of the revenue loss that occurred globally because of the semiconductor shortage has started to flatten in recent months and that is precisely why there is a lot of scope for future work in this topic, since it is a relatively new one.

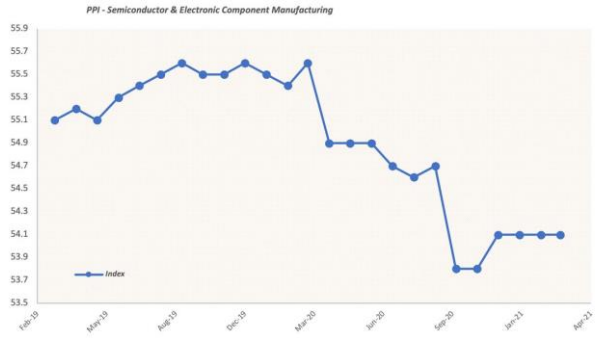


Figure 9.Revenue change in Global Semiconductor Sector

5. REFERENCES AND CITATIONS

I have mentioned the websites wherein I found some of the content, both informational and diagrammatic.

1. https://onebyzeroelectronics.blogspot.com/2015/09/types-of-ic-pakages.html (Figure 1)
2. https://www.intel.com/content/www/us/en/history/museum-story-of-intel-4004.html (Figure 2)
3. https://onlineclassnotes.com/draw-internal-architecture-of-8086/ (Figure 3)
4. https://www.walmart.com/ (Figure 4)
5. https://en.wikipedia.org/wiki/Ostrich_algorithm (Section 3.1)

6. https://www.studytonight.com/operating-system/deadlocks (Figure 5)
7. https://www.economist.com/business/2020/02/29/how-asml-became-chipmakings-biggest-monopoly (Figure 6)
8. https://www.computersciencejunction.in/2018/10/23/process-control-block-in-operating-system-html/ (Figure 7)
9. https://math.hws.edu/javanotes/c1/s1.html (Figure 8)
10. https://www.edaboard.com/threads/what-is-porting-of-code-into-microcontroller.145993/ (Section 3.4)
11. https://www.gep.com/blog/mind/why-is-there-a-global-semiconductor-chip-shortage-and-what-can-we-expect (Figure 9)

B - An Analysis of Quantum Computer Architecture

Justin Chang, Chaz Chang
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000

justin.h.chang@sjsu.edu
chaz.chang@sjsu.edu

ABSTRACT

Quantum computing is an alternative approach to the classical binary computing. This nascent technology will break many established assumptions and rules all computer scientists hold. Superposition is the principal concept that displays the power of this new computing paradigm. The power of representing 0 and 1 simultaneously shatters traditional algorithmic run time upper bounds and renders previously unbreakable security measures vulnerable. While this technology is still in development, it is important for computer scientists to understand and develop algorithms and software to utilize this technology to be prepared for the future of computing.

1. INTRODUCTION

The theory of using quantum mechanics to represent a higher order of magnitude for bit representation has been a very groundbreaking theory since its inception in the 1980s [1]. However, quantum computing has been the technology of the future since then. While the development of quantum computers has been slow, computer scientists have been continuously developing new algorithms that can utilize the power of quantum computers. The limitation of quantum computing has been on the hardware spectrum. Some of the limitations include the low number of qubits able to be captured and the accuracy of the readings of the qubits. This report is designed to give computer scientists with minimal knowledge of physics a high-level understanding about quantum mechanics and computers. This understanding will be useful for computer scientists looking to create algorithms that take advantage of the power of quantum computers.

2. THEORY OF TRADITIONAL COMPUTING

The traditional computing model is represented by the pipeline in figure 1, which shows the different building blocks at different levels. From biggest to smallest (left to right), each building block is composed of the building block to the right of it. The lowest level of this model demonstrates the most important difference between classical and quantum computers. There are many transistor designs, but they all follow the same two-state representation. Figure 2 shows an example of a field effect transistor (FET). When voltage is applied to the gate, it allows the electrons to flow from the source to the drain. The "off state" of the transistor blocks the flow of the signal and the "on state" allows the signal to propagate. This deterministic state of either being 0 or 1 is a key concept in this classical model. This deterministic two-state model is transcended by the qubit in the quantum computing model. Currently, many computer processing units are composed of transistors of minuscule size. The smaller the transistor size, the more computing power per size a chip has. When transistors reach 1-3 nm in size, an effect called quantum tunneling happens [2]. AMD currently can produce transistors of that are 7nm, which demonstrates the size limit that quantum tunneling occurs is approaching soon. When quantum tunneling occurs, electrons can teleport past the gate from the source to the drain, violating the function of a transistor. This unpredictable behavior will render the classical computers useless. However, the erratic behavior of sub-atomic particles can be captured to develop a new computing model called the quantum computing model. In this model the notion of binary bits is obsolete.

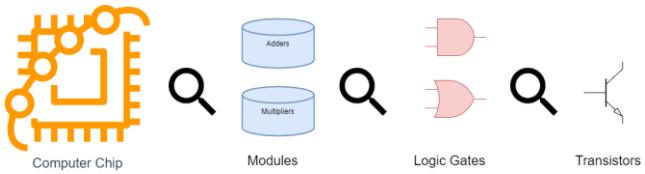


Figure 1. Traditional computing pipeline.

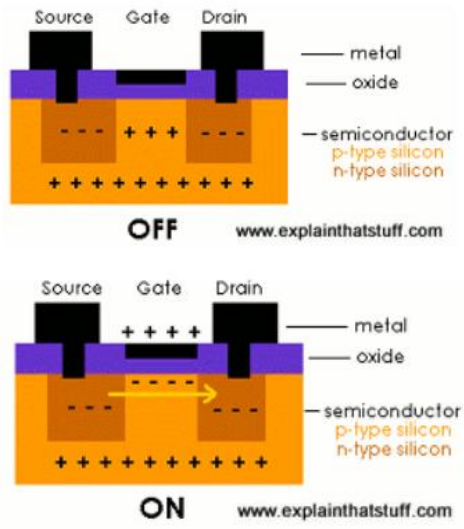


Figure 2. Diagram of a field effect transistor (FET).

3. THEORY OF QUANTUM MECHANICS

This section covers the history and theory of quantum mechanics necessary towards an understanding of quantum computers. Richard Feynman, one of the founders of quantum theories and computing, was a theoretical physicist that wanted to model

quantum events. The modeling and observation of quantum events in the real world was near impossible, thus he desired to create a simulation to model quantum events. However, the power required to simulate quantum particles was far too great for a traditional computing model to handle. He then attempted to create a computer that used hardware based off quantum physics to simulate quantum physics.

3.1 Amplitudes

To measure quantum systems, the concept of using a function of probabilities called amplitudes was conceived. A probability amplitude is a complex number representing properties of a subatomic particle such as energy and momentum. They provide insight to understanding the bridge between waveforms and a series of discrete observations of the particles. The integration of the square of these waveforms gives us the probability density. After calculating the probability density, the probability that a particle will have a certain value is found. At subatomic levels we can never be 100% confident about position, velocity, energy, etc., we can only calculate, the probability of the value being observed [3]. Many quantum physics concepts such as this one break the established laws of physics and continue to be debated about in present time.

4. QUBITS

The amount of information n qubits can store is the same as 2^n traditional bits. The different quantum subatomic particles used to represent qubits also have different representations of 0 and 1. For instance, an electron can have a spin up represent 0 and spin down represent 1. Each state has a probability of being observed in that state.

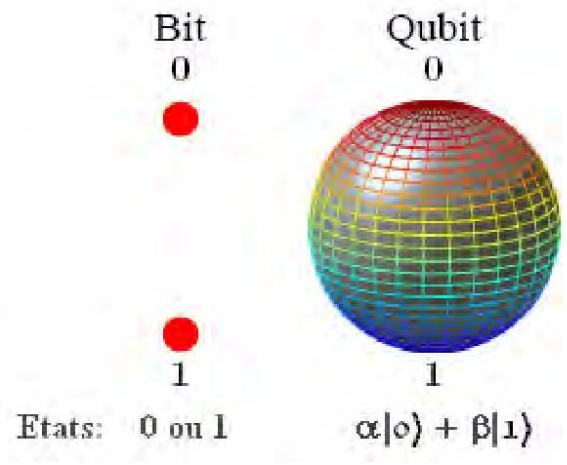


Figure 3. Qubit diagram.

Each combination of these qubit states has a coefficient associated with it that represents the probability of that event being observed. However, as soon as the qubit is observed, it loses its intrinsic property of being in multiple states at once. Figure 3 demonstrates the difference in the degrees of freedom that bits and qubits have. Qubits can have an infinitely large combination of probabilities of 0 and 1, while bits are either always 0 or always 1.

4.1 Qubit Representation

The pure qubit state can be represented by a linear combination (superposition) of its 2 states. The 2 states are $|0\rangle$ = ket 0 and $|1\rangle$ = ket 1. The state of the qubit is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where α and β are probability amplitudes and are complex numbers (a real and imaginary part). The probability of observing $|0\rangle = |\alpha|^2$. The probability of observation $|1\rangle = |\beta|^2$. Probabilities must add up to 1, thus there is a constraint $|\alpha|^2 + |\beta|^2 = 1$ [4].

4.2 Bloch Sphere Notation

There are many ways to represent the state of a qubit. In Bloch sphere notation, the state is represented by 2 variables (degrees of freedom) φ and θ. As seen in Figure 4. this notation can be pictorially represented as a unit sphere.

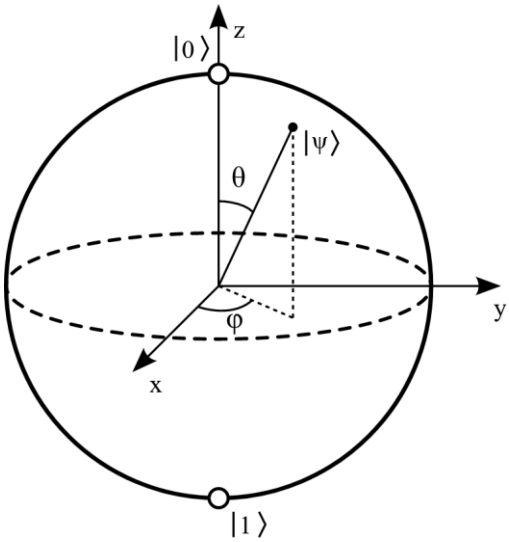


Figure 4. Bloch sphere representation.

φ and θ are similar to polar coordinates and represent angles pointing to a spot on the sphere. φ and θ are related to α and β by the following equations: $\alpha = \cos(\theta/2)$ and $\beta = e^{i\varphi}\sin(\theta/2)$ [5]. The state space (all possible states) of qubit can be visualized as the surface of the sphere.

5. QUANTUM SUPERPOSITION

Quantum Superposition states that 2 or more quantum states can be added together to form another valid quantum state. [6]

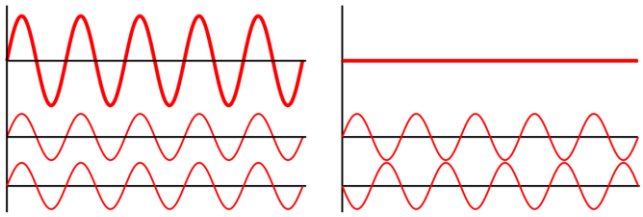


Figure 5. Superposition wave summing.

As shown in Figure 5, in the 1st pair of waves on the left, the waves combine to make a wave of higher amplitude. The peaks of the waves line up to create a bigger peak and the troughs of the waves line up to create a bigger trough. In the 2nd pair of waves on the right, the waves combine and cancel each other out. The peak of the 1st wave lines up with the trough of the 2nd wave

which results in an amplitude of 0. The trough of the 1st wave lines up with the peak of the 2nd wave which results in an amplitude of 0. Some examples of Quantum Superposition are the double slit experiment, diffraction (light bending around a corner of an object), and qubit state.

A popular analogy of super position is a thought experiment called Schrödinger's cat. A cat is placed in a box and a mechanism inside the box releases poison after it detects radioactive decay from a radioactive source. For a period of time that the box is not opened, the cat is both alive and dead. This is analogous to a qubit being in both states 0 and 1 simultaneously. However, when the box is opened, the probability of the cat's state collapses to either alive or dead. This event is similar to when a qubit is measured, the state of the qubit collapses to either 0 or 1.

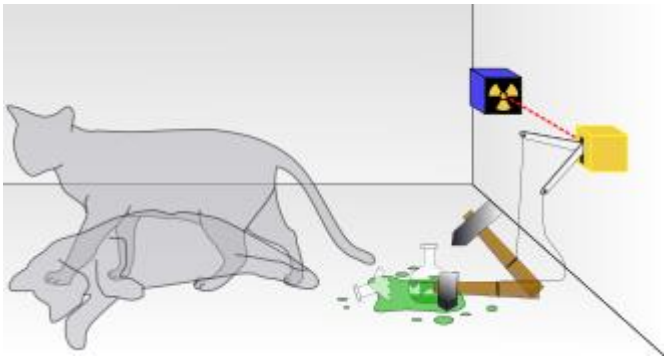


Figure 6. Schrödinger's cat.

5.1 Double Slit Experiment

The Double Slit Experiment was first performed by Thomas Young. In the experiment, a laser is pointed at a screen with 2 slits and another screen is placed behind it. The laser must be wide enough so that it can shine on both slits at the same time [3].

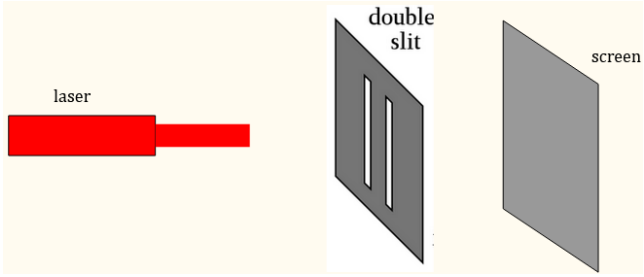


Figure 7. Double slit experiment setup

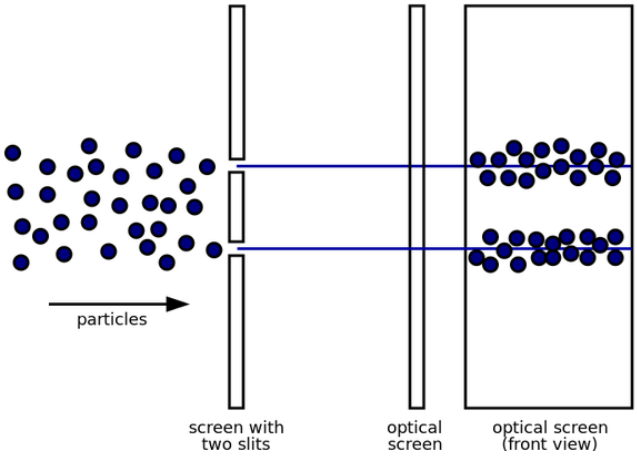


Figure 8. Double slit experiment expectation

Consider light as a series of particles, then the expectation is that the light will move in a straight line and 2 lines of light will be observed on the screen. Some of the particles will move at a slight angle if the slit is big enough and hit the screen slightly off the line. However, this is not what happens in real life.

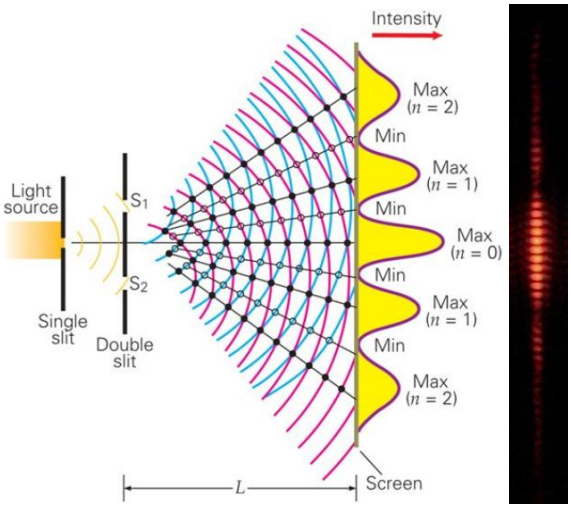


Figure 9. Double slit experiment result.

In real life, light can behave like a wave. Two waves can combine to create a wave of greater amplitude or cancel each other out. The screen will show alternating bright and dark spots. The brightest spot is in the middle of the 2 slits. The bright spots get darker when it's further from the middle. The filled points where the purple curves and blue curves intersect form a line where the bright spots appear on the screen. The hollow points where the purple curves intersect with the halfway point between 2 blue curves form a line where the dark spots appear on the screen. Light has a wave-particle duality. It acts like both a wave and particle, but it is only observed as a particle [7].

6. QUANTUM ENTANGLEMENT

Quantum entanglement is the phenomenon when a group of particles states are codependent on each other. Even across vast distances their states still do not remain independent from one another. In the EPR (Einstein-Podolsky-Rosen) paradox, suppose 2 entangled particles have their total spin equal to 0. The 1st particle is observed to have clockwise spin on an axis [8]. Then

the spin of the 2nd particle is counterclockwise on the same axis. This contradicts that a quantum particle does not have a definite spin until it is observed/measured. Measuring one qubit collapses the other in a correlated state. This is also known as "spooky action at a distance". This shows that the accepted formulation of quantum mechanics is incomplete.

China did a study on quantum entanglement where they had a pair of entangled qubits. Both qubits have the same state at the same time. 1 qubit was on a satellite in space, and 1 qubit was on earth. The clocks on both machines were synchronized and the satellite was far enough away so that if any communication occurred between the qubits it would have to happen faster than the speed of light. Somehow the qubits always show the same state at a given time. The qubits exhibit this correlation, but the events are not predetermined [9].

7. TECHNIQUES TO CAPTURE AND ANALYZE QUBITS

A common representation of qubits that is easy to compute is a vector representation. The vector representation of the super position state of a single qubit |0⟩ would be a vector of 2x1 dimensions.

$$|a\rangle = v_0|0\rangle + v_1|1\rangle \rightarrow \begin{bmatrix} v_0 \\ v_1 \end{bmatrix}$$

Figure 10. Single qubit vector representation.

A combination of two qubits requires the tensor product of two qubits. This product is the result of all the combinations of the values in each vector multiplied by the values in the other vectors.

$$|01\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Figure 11. Example of tensor product.

To transform qubits, quantum gates function also changes the output of the qubit in a way similar to traditional gates. Gates can be represented by a unitary matrix of 2^n by 2^n for n qubits. However, unlike classical logic gates, quantum logic gates are reversible, meaning that applying the gate to the input and applying the gate again to the output of the first gate results in the original input.

7.1 Hadamard Gates

The Hadamard gate acts on a single qubit and maps classical bits to qubits of equal superposition. They are used in many quantum algorithms to map qubits starting at |0⟩ to a super position of all 2^n states. The reversibility of quantum gates also allows the transformation of qubits of equal super position back into classical bit notation. This is very useful as it allows the transition out of superposition without measurement, which collapses the qubit. The figures below only use real numbers for simplification to demonstrate the transformation the Hadamard gate performs.

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Figure 12. Matrix representation of a Hadamard gate.

$$H|0\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{pmatrix} \qquad H|1\rangle = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \end{pmatrix}$$

Figure 13. Example of Hadamard gate transformation on ket 1 and ket 2.

7.2 Controlled NOT Gates

The controlled NOT gate acts on 2 or more qubits. The most significant qubit is designated the control bit and the others represent the target bit. When the control bit has the value of 1, the target bits are flipped, 0 → 1 and 1 → 0. We can represent the two bits in a logic table shown in table 1.

Table 1. Logic table for CNOT on a 2 bits

Before		After	
Control	Target	Control	Target
|0⟩	|0⟩	|0⟩	|0⟩
|0⟩	|1⟩	|0⟩	|1⟩
|1⟩	|0⟩	|1⟩	|1⟩
|1⟩	|1⟩	|1⟩	|0⟩

The CNOT gate has a matrix representation in figure 10 and is represented this way when applied to vector and matrix representations of qubits. An example of CNOT gates being applied to 2 qubits is shown in figure 15.

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 14. CNOT matrix.

$$C|11\rangle = C\left(\begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |10\rangle$$

$$C|00\rangle = C\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |00\rangle$$

Figure 15. CNOT gate applied to |11⟩ and |00⟩

8. DESIGNS OF QUANTUM COMPUTERS

Any system involving quantums (sub-atomic particles) can be used to represent a qubit. Different particles are more stable than others and are favored depending on how the manufacturer

captures the particle. However, the IBM quantum computers that are available to the public are not made from natural qubits. They use superconducting material such as niobium and aluminum to mimic the properties of a qubit. This type of qubit is called a transmon and are resistant to charge noise and conduct electricity with no electrical resistance. Below is a table of some qubits and their representations of $|0\rangle$ and $|1\rangle$. To design enclosures to isolate qubits and keep them at optimum environmental variables is extremely tedious compared to manufacturing traditional hardware. For instance, the joseph junction operates at temperatures below 100mK.

Table 2. Examples of physical representations of qubits

Physical Support	NAME	$\|0\rangle$	$\|1\rangle$
Photon	Polarizati -on Encoding	Horizontal	Vertical
Electrons	Electron Spin	Horizontal	Vertical
Electrons	Charge	No electron	One electron
Nucleus	Nuclear Spin	Up	Down
Josephson Junction	Supercon- ducting flux qubit	Clockwise current	Counterclock- wise current

9. IMPROVEMENTS OVER CLASSICAL COMPUTERS

Quantum computers are useful for computations involving large amounts of parallelism. The concept of involving quantum superposition allows for this parallelism. A single pipeline algorithm will perform similarly when run by classical computers and quantum computers. However, the cost of using the quantum computer is wasted on a process that a cheap classical computer can compute in the same time. In fact, a classical computer may even be faster with a single pipeline algorithm. The parallelism of large amounts of operations makes quantum computers powerful. Quantum computers provide no additional power over classical computers in terms of computability, they can just drastically decrease the time complexity.

Grover's algorithm can utilize the parallelism of qubits. Suppose there is a function f such that y = f(x), where x is the input and y is the output. Given f and y, the goal is to find x with high probability (some high probability that is close to 100%). Grover's algorithm has the time complexity of O(sqrt(N)), where N is the size of the function's domain. A classical approach solves this problem in time complexity of O(N). This speeds up brute force attacks on symmetric key cryptography by a quadratic factor. If function f was a hash function, then this algorithm can be used for a collision attack (find different input x with the same output y) [10].

Shor's algorithm can also utilize the parallelism of qubits. Suppose N is a large number with 2 prime factors. In RSA (public key cryptography) there is a public key composed of modulus N and public exponent e). The public key is public so an attacker can easily get these values. Factoring N allows you to find the private key (private exponent d). The private key must be kept secret because the private key allows someone to decrypt an encrypted message. This is a major security concern because RSA is commonly used today. If RSA is easily broken, then any attacker can easily decrypt encrypted messages. [11]

10. ROADBLOCKS FOR QUANTUM COMPUTING

One of the major roadblocks for quantum computing is quantum decoherence (the loss of quantum coherence). The state of quantum particles can be represented by a wave function with probabilities of being in a certain state. When the outside environment interacts with the quantum particle, the particle's wave function will change, and it will decohere. Quantum gates and electricity/heat from the computer can cause the particle to decohere [12]. It is hard to isolate the particles from the outside environment. If a particle was completely isolated, then the particle will not decohere, but it cannot be changed or measured.

Due to quantum decoherence it is hard to read qubits without changing the wave function or state of the qubit. Exposing the qubit to the outside environment is necessary to read a qubit but exposing the qubit to the outside environment can cause the qubit states to change. Quantum decoherence is not reversible. It is hard to take a measurement when even the quantum gates can cause decoherence.

Qubits are very hard to build and therefore very expensive. Today, quantum computers don't have many qubits. In 2019, IBM created IBM Q System One which has 20 qubits [13] and Google created Sycamore which has 53 qubits [14]. In order to solve problems with a quantum computer, the quantum computer must have enough qubits to store the problem in memory. If the problem cannot fit into the number of qubits, then the quantum computer cannot be used to solve that problem.

11. CONCLUSION AND FUTURE AREAS FOR RESEARCH

One of the limiting factors in quantum computers is that they don't have many qubits. If quantum computers had a lot more qubits, then RSA encryption would be easily broken and other NP hard problems could be solved a lot faster. Machine Learning has a lot of functions that could be computed in parallel, so quantum computers could be used. Machine Learning training involves computing a lot of functions between many layers. Computer Scientists already have developed algorithms suitable for quantum computing, however the bottleneck in development is with the engineers and physicists that need to create more powerful quantum computers. IBM has a cloud-based quantum computing service called Qiskit where anyone can try out using a quantum computer [15]. There is a wait time of a couple of days before IBM runs your code, however the service is free. The potential in the amount of problems quantum computing can solve is vast and if in the future, large scale quantum computers can be developed at a reasonable cost, then computing paradigm will shift dramatically.

12. REFERENCES

[1] Paul Benioff. 1980. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of

computers as represented by Turing machines. Journal of Statistical Physics. 22 (5): 563–591

[2] Rita Lerner and George Trigg. 1991. Encyclopedia of Physics (2nd ed.). New York: VCH. p. 1308

[3] Richard P. Feynman, Robert B. Leighton, Matthew Sands. 1989. Probability Amplitudes. The Feynman Lectures on Physics. Volume 3. Redwood City: Addison-Wesley. ISBN 0-201-51005-7.

[4] Colin P. Williams. 2011. Explorations in Quantum Computing. Springer. pp. 9–13. ISBN 978-1-84628-887-6.

[5] Michael A. Nielsen and Isaac Chuang. 2010. Quantum Computation and Quantum Information. Cambridge: Cambridge University Press. pp. 13–16. ISBN 978-1-10700-217-3. OCLC 43641333.

[6] Paul A.M. Dirac. 1947. The Principles of Quantum Mechanics (2nd ed.). Clarendon Press. p. 12.

[7] Clinton J. Davisson. 1928. The diffraction of electrons by a crystal of nickel. Bell System Technical Journal. 7: 90–105. doi:10.1002/j.1538-7305.1928.tb00342.x.

[8] Albert Einstein, Boris Podolsky, Nathan Rosen. 1935. Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? Phys. Rev. 47 (10): 777–780. Bibcode:1935PhRv...47..777E. doi:10.1103/PhysRev.47.777

[9] Gabriel Popkin. 2017. China's quantum satellite achieves ‘spooky action' at record distance. (June 2017). Retrieved November 11, 2021 from https://www.science.org/content/article/china-s-quantum-satellite-achieves-spooky-action-record-distance

[10] Lov K. Grover. 1996. A fast quantum mechanical algorithm for database search. Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery: 212–219. doi:10.1145/237814.237866. ISBN 978-0-89791-785-8.

[11] Peter Shor. 1994. Algorithms for quantum computation: discrete logarithms and factoring. Proceedings 35th Annual Symposium on Foundations of Computer Science. IEEE Comput. Soc. Press: 124–134. doi:10.1109/sfcs.1994.365700. ISBN 0818665807.

[12] Antti Vepsäläinen, Amir Karamlou, John Orrell, Akshunna Dogra, Ben Loer et al. 2020. Impact of ionizing radiation on superconducting qubit coherence. Nature. 584 (7822): 551–556.

[13] Rosalie Chan. 2019. IBM unveils the world's first quantum computer that businesses can actually use to solve previously impossible problems. (January 2019). Retrieved November 10, 2021 from https://www.businessinsider.com/ibm-unveils-ibm-q-system-one-the-first-commercial-quantum-computer-2019-1

[14] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando Brandao, David Buell, Brain Burkett, 2019. Quantum supremacy using a programmable superconducting processor. Nature. 574 (7779): 505–510.

[15] Robert Wille, Rodney Meter, Yehuda Naveh. 2019. IBM's Qiskit Tool Chain: Working with and Developing for Real Quantum Computers. 2019 Design, Automation, and Test in Europe:1234–1240.

C - Quantum Computers – A Review

Ikshaku Goswami
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000
ikshaku.goswami@sjsu.edu

ABSTRACT

Classical computers of today are limited in their scope of solving certain complex problems. Even supercomputers, with many cores, are not capable of finding solutions to these problems in finite time. Moreover, designing faster computers with the currently available technologies has become highly limited. Moore's law is almost close to an end as finer gate widths are becoming impossible due to deep sub-micron effects. One solution could be parallel computing but programming highly parallel applications utilizing the power of multi-cores is challenging. Moreover, it only speeds up the computations linearly. This brings the necessity of Quantum Computers, which use concepts from quantum mechanics like superposition, entanglement, annealing, etc., and information science to perform complex computations faster. This paper will review the quantum computing architecture and contrast it with the traditional architectures. Furthermore, it will discuss the advantages that Quantum Computers have over traditional computing systems and the current state of this technology.

1. INTRODUCTION

Computers that are ubiquitous today have a rich history of several years in the making. The first generation of computers that came into being in 1946 were vacuum tube-based machines. They were used for batch processing and magnetic- tapes, punch cards were used as input-output devices in these machines. Some examples of first-generation computers include IBM-650, ENIAC. Then came the transistor-based second-generation machines in 1959. This was the generation when assembly and high-level languages like COBOL, FORTRAN were developed. Examples include IBM 1620, UNIVAC 1108 machines. The third generation of computers started in 1965 and Integrated Circuits or IC's replaced the transistors. A famous computer belonging to this generation is the IBM 360 computer which was a mainframe. The tremendous growth in computer technology continued as VLSI came into prominence in 1971. This marked the beginning of the fourth generation of computers which continued till 1980. This is perhaps one of the most important stages in computer generation as modern high-level languages like C, C++ were developed and also gave rise to the generation of personal computers. The introduction of personal computers marks a revolutionary change in computer technology as it empowered every person in the world to own a powerful computer and create something meaningful. Throughout the development of computers and microprocessor technology computers and chips kept getting smaller and smaller. The fifth generation of computers is the one we are in today. This was the generation of Ultra Large-Scale Integration technology which resulted in the production of microprocessors with over ten million components. This was also the time when a famous American engineer and founder Gordon Moore presented his Moore's law which simply stated that the number of transistors in Integrated Circuits would double every two years. This amazing observation was held for a long time. However, it does face challenges since it becoming harder to increase the number of transistors in the circuits without causing electrical and heat leakage which puts the entire chip at risk.

The main computer architecture includes the components Central Processing Unit, Memory, and Input/output system. Most computers today have originated from the original Von-Neumann architecture.

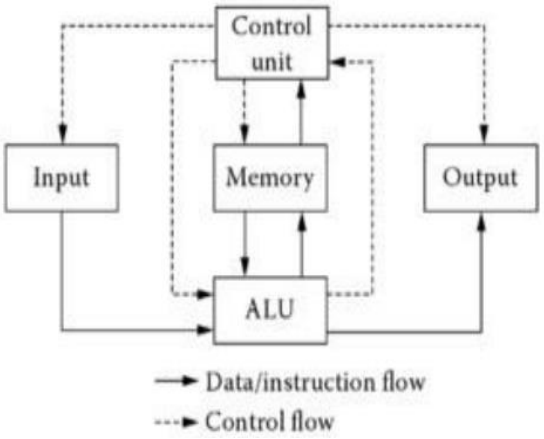


Fig 1 – Von-Neumann Architecture [9]

As we navigate the fifth generation of computers we also see the rise of new technologies like parallel or distributed computing, machine learning, and artificial intelligence. Parallel computing holds significant importance as it ensures applications can be run using the multithreading concept on multiple cores of a processor. But programming such applications is difficult and it only guarantees a linear speed-up.

Fortunately, enough we have a new generation of computers i.e., the Quantum Computers.

2. QUANTUM COMPUTERS

Quantum computers unlock the power of quantum mechanics to bring about computers incredibly faster and complex. They can help us solve many of the intractable problems of yet. Quantum computers are programmed in qubits or quantum bits, which have the properties of superposition and entanglement, which makes a quantum computer exponentially faster and also complex.

2.1 History

Quantum computing traces its origin to 1980 when Dr. Paul Benioff proposed a quantum machine identical to the Turing machine. Richard Feynman in 1981, proposed a computer based on quantum mechanics. However, the interest in the field of quantum computing lay dormant till mathematician Peter Shor came up with Shor's algorithm for breaking any cryptosystem using a quantum computer. After, Peter Shor, Lov Grover of Bell Labs developed Grover's algorithm in 1996 as a quantum database search algorithm. Later in 1997, MIT published papers on models of quantum computers using the principles of spin resonance and thermal ensembles. Perhaps, the biggest push in quantum computer architecture development came in 1999, when IBM developed a 3-qubit computer and executed Grover's search algorithm. IBM undoubtedly has invested a lot in quantum computer development as in 2001 they built a working 7-qubit computer and executed Shor's algorithm successfully.

2.2 Qubits

We know that classical computers store information in bits '0' or '1'. This means any bit at a point in time could be either '0' or '1' only. It can be considered as a switch. But, quantum bits or qubits are different since they represent the probability of an object's state. Due to a principle in quantum mechanics called superposition, a qubit could be in a mixed state of '0' and '1'. The states of a qubit are represented as |0> and |1>. This notation is called 'kit vector' in Dirac notation [7].

2.3 Superposition

The superposition principle is drawn from the discovery that light could be both a particle and a wave. Qubits can be in multiple states at the same time. Just like classical computers qubits are also represented as 0 and 1. The difference although is that due to superposition qubits can also be in a mixed state of 0 and 1. This means that if we have two bits in a classical computer we can have four possible states like 00,01,10 or 11 but only one of them at a time. So, for n bits, 2^n states in total and 2^n steps to process through all these states. The power of a quantum computer is that it can process all these 2^n states in one step. Thus, giving it an aura of extreme parallelism and exponential speed-up.

2.4 Entanglement

In quantum mechanics, two qubits are present in an entangled state if the measurement of one qubit can be correlated with the other qubit. It is also not possible to destroy the quantum state of one particle as it is entangled with another particle. The states can be reversed no matter how far these particles are present.

2.5 Recent Developments

Many well-renowned companies have heavily invested in the area of quantum computers. IBM promises to develop a 1000 qubit quantum computer by 2023. Recently in 2019 Google demonstrated quantum supremacy by solving a problem that would take the fastest supercomputer 10000 years in a few minutes. One of the major developments in the area of quantum computing was in 2010 when D-Wave unveiled its first commercial quantum system which had a 128-qubit processor. D-Wave systems have already breached the 1000-qubit barrier and have built powerful quantum systems.

3. QUANTUM ARCHITECTURES

Much like the classical computer architecture, a quantum computer also has components like a quantum Arithmetic and Logic Unit (ALU), quantum memory, and a dynamic scheduler. Rao in [1] proposes a layered quantum architecture like shown below

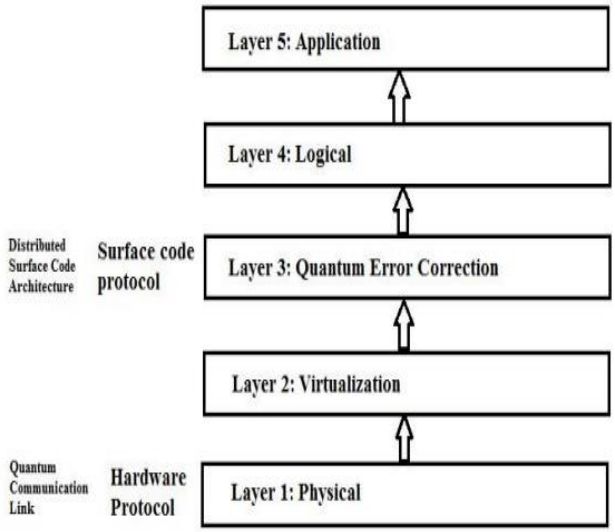


Fig 2 – Layered Quantum Architecture [9]

3.1 Layered Architecture

In a layered quantum computer, each layer plays a specific role. All layers are interfaced with one another and lower layers provide information or services to the upper layers [7]. The application layer runs quantum algorithms and interfaces to the classical user. The logical layer supports universal quantum computation. Quantum error correction corrects the bit flip and phase flip errors. The virtualization layer much like hypervisors gives the illusion of running multiple quantum OS's on a single computer. It can provide virtual qubits, CNOT gates etc. The physical layer provides the hardware apparatus including physical qubits, wires, and control operations.

Oskin et al in 2002 had given a general-purpose architecture of a quantum computer as shown below.

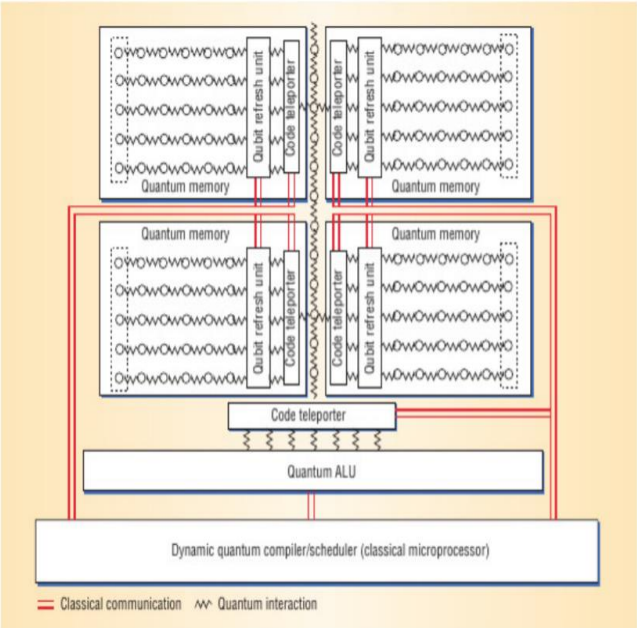


Fig 3 – General Purpose Architecture [5]

3.2 General Architecture [Oskin et al 2002]

The Quantum ALU performs both computation and error correction. The ALU applies a sequence of multiple transforms with gates like the Hadamard, Identity, bit flip (Quantum NOT), controlled NOT (CNOT) etc.

In classical computers, data suffers only bit flip errors. But, in quantum computers, qubits suffer from both bit flip and phase flip errors. Correcting both is a hard task which is why error correction plays an important role in quantum architectures.

Oskin et al [5] proposed a method of using two classical codes for error detection. One classical code for bit flip and other for a phase flip. This is similar to parity correction codes but only differ in their detection mechanism. An [n,k] code will use n qubits to encode k qubits of data [5].

In [5] Oskin et al describe the usage of multiple quantum memory banks in the quantum processors. Multiple memory banks serve as the storage for error correction qubits, quantum ALU operations etc.

The code teleporter performs code or quantum state teleportation function. For it to happen successfully two qubits must be in an entangled state as information or state change information is teleported between them without collapsing either.

The dynamic scheduler uses a classical high-performance processor as a control operator. It takes in logical quantum operations, interleaves it with classical control-flow constructs, and translates them into physical qubit operations [5].

4. APPLICATIONS

Many well-known quantum computing algorithms have been developed so far. They hold wide-scale application in different areas like cryptography, data storage etc. This section will look into a few popular algorithms and their application areas.

4.1 Shor's Algorithm and Cryptography

Peter Shor in 1994 presented the Shor's algorithm. It deals with prime factorization. Given a large number N, Shor's algorithm could factorize N into say P and Q in O (log N^3) time [6]. This is significantly faster than any classical deduction and also holds a significant value in cryptography.

In cryptography, most public-key crypto systems such as RSA uses integer factorization [6]. This means that Shor's algorithm could be used to break any public key-based system out there which uses factorization in its core. Although Shor's algorithm has not been demonstrated for large numbers yet, still experiments like that done by a group in IBM Almaden which factored a small number 15 into 3 and 5 using 7 qubits. It's only a matter of few years when large numbers will also be factored using Shor's algorithm and a greater set of qubits.

4.2 Grover's Algorithm/Unstructured Search

Most data generated in the world is unstructured. Grover's algorithm is based on searching an element in this unstructured or unordered space. Such an unordered search will take exponential time for regular classical algorithms on classical computers, however, quantum computers using Grover's algorithm reduces the time complexity of such search to O (sqrt(N)) for N elements.

This algorithm finds its application in all areas of computer science today. Especially in areas like data mining, big data or machine learning.

4.3 Other Applications

Quantum computers can find its application in multiple areas today. Some of these applications are discussed below: -

4.3.1 Machine Learning

In [6] Guizani et al discusses several application of quantum computers in machine learning. The authors talk about the use of Grover's algorithm and amplitude amplification techniques in developing faster versions of k-Nearest Neighbor algorithm, a well-known machine learning algorithm used in classification.

Similarly, quantum Support Vector Machines (SVM) have also been proposed where quantum algorithms can be used to generate the most optimum hyperplane in SVM.

Quantum neural networks and quantum deep-learning are also new areas that have emerged in recent times. Most of the current work in this area is on replacing binary bits with qubits, utilizing features such as superposition and entanglement.

4.3.2 Logistics

Quantum computing can be effectively utilized to solve transportation and logistics problems. ExxonMobil uses IBM's quantum services to efficiently transport liquid natural gas (LNG) around the world. In a rough estimate, the number of combinations of decisions in a global LNG shipping network is $2^{1,000,000.}$ To deal with such a vast amount of decisions quantum algorithms can be utilized on qubits which can be in multiple states at the same time. This inherently is an optimization problem and quantum computers can provide far higher optimization and speed-up compared to even the fastest supercomputer today.

4.3.3 Modeling

It is hard to model problems in the areas of finance or astrophysics. Because of their inherent complexity, even the fastest supercomputers of today cannot accurately predict or analyze an outcome in real-time. Quantum computing can significantly benefit in this

area. Organizations like CERN utilize IBM quantum services to model the universe. CERN is using quantum computing to make sense out of its huge spectrum of data and make decisions out of it. A famous use case of this is CERN's use of quantum computing to detect and analyze Higgs Boson particles.

5. CONCLUSION

In terms of processing quantum computers have the potential to explore their zenith. Quantum computers have their application in all areas such as cryptography, medicine, defense, physics exploration, logistics, finance, etc. Already, with some of the well-known algorithms, quantum computers have displayed in real-time how incredibly faster and efficient they can be. Still, with all the power and efficiency around, it's difficult to practically build a quantum system. Harnessing qubits in entangled states is difficult since a slight vibration could make the qubit lose its superposition. It's also highly expensive to run quantum algorithms on a quantum system. In the future, quantum computing needs will likely be fulfilled via cloud computing, hence it is imperative to start designing quantum systems around it.

6. REFERENCES

[1] Tzvetan Metodi; Arvin I. Faruque; Frederic T. Chong, Quantum Computing for Computer Architects, Second Edition, Morgan & Claypool, 2011.

[2] J. Singh and M. Singh, "Evolution in Quantum Computing," 2016 International Conference System Modeling & Advancement in Research Trends (SMART), 2016, pp. 267-270

[3] V. S. Igumnov and V. N. Lis, "Influence of Quantum Computers on Classical Cryptography," 2007 8th Siberian Russian Workshop and Tutorial on Electron Devices and Materials, 2007, pp. 220-24R. Vignesh and P. G. Poonacha, "Quantum computer architectures: An idea whose time is not far away," 2015 International Conference on Computers, Communications, and Systems (ICCCS), 2015, pp. 6-13

[4] Oskin, M., Chong, F.T., & Chuang, I.L. (2002). A Practical Architecture for Reliable Quantum Computers. Computer, 35, 79-87.

[5] V. Hassija, V. Chamola, A. Goyal, S. S. Kanhere, and N. Guizani, "Forthcoming applications of quantum computing: peeking into the future," vol. 1, no. 2, pp. 35–41

[6] Y. Kanamori, Y. Kanamori, S.-M. Yoo, W. D. Pan, and F. T. Sheldon, "A SHORT SURVEY ON QUANTUM COMPUTERS," vol. 28, no. 3. pp. 227–233, 2006.

[7] S. Jain, "Quantum computer architectures: A survey," 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015, pp. 2165-2169.

[8] K. B. Rao, "Computer systems architecture vs quantum computer," 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), 2017, pp. 1018-1023

[9] https://www.ibm.com/quantum-computing/case-studies/

[10] https://cosmosmagazine.com/science/quantum-computing-for-the-qubit-curious/

D - Impact of High Performance Processors in Production & Animation

Sudhin Domala
Computer Science Department
San Jose State University
San Jose, CA 95192
925-314-6621
sudhin.domala@sjsu.edu

ABSTRACT

VFX and animation studios revolve around high-definition image processing to bring complicated 3D assets to life onscreen. The Central Processing Unit (CPU) is the heart of such technology that augments a typical Feature Film Pipeline in any visual effects production. CPUs are High Performance Processors that deliver professional computing demands that are scalable and stable in increasingly complex visual effects. For a while, CPU performance for Media & Entertainment had been stagnant till recent leaps in core counts, clock speeds, etc. that noticeably improved studios' processor-intensive tasks. Though these tasks range from gaming to deep learning, this paper will refer to comparing AMD Hardware innovations,CPU vs GPU rendering, and Intel Software innovations to have a comprehensive understanding of how High Performance Processors validate a studio's significant long term investment with leading render engines for their 3D modeling and animation. Whether it's around an artist's workflow or studio's production pipeline, CPU vs GPU rendering yields tradeoffs in Speed vs Stability, Cost, Initial Graphic Fidelity, and Real Time Visualization. Such tradeoffs extend to various Hardware-based or Software-based overhauls to the 3D Rendering pipeline when it ultimately comes down to how one wants to address their Rendering workload in their customized workstation.

1. INTRODUCTION

When dealing with Computer Architecture, there is a Design Optimization interplay between Hardware and Software that needs to be considered when designing a successful computer architecture. Rapid changes in the HW/SW technologies necessitate the oversight of important tradeoffs between a chosen criteria of important parameters on a computer architect's end. The desire to make current products cheaper and faster comes with discrete insights in cost, marketplace integration, and accountability of relevant technology trends for demand metrics. A successful architecture must last through such a design criteria between the applicable optimization one can make on the Hardware or Software end. With hardware as tangible devices, they have a lower cost of errors, easier design, and simpler upgrading schemas in general. However software architecture avoids the cost and easier performance testing schemas than when dealing with Hardware. Thus there is no single correct HW/SW-centric solution when it comes to design optimization. For this paper, the application to optimize is Rendering in Computer Graphics: the process of producing 2D images from a 3D environment from a model by means of application programs. Rendering comes with two subcategories of pre-rendering and real-time rendering where their difference lies in the speed of which computer and finization of images take place. The global optimization to always consider is how these processors merge the semantic gap between Studio Visualization of Storyboarding to Graphical Output of an Animated Feature. The consumer demand lies in a better architecture to crunch numbers in rendering algorithms for high-fidelity images that match a developer's vision. This paper explores 4 areas of improvement in rendering capabilities that are not HLL or tool specific. These are CPU Rendering, Hardware evolutions that AMD brings to studios, GPU Rendering, and Platform Cooperative Computing that Intel brings to its customers.

2. CPU RENDERING

A CPU consists of a few cores optimized for sequential serial processing, while a GPU has a massively parallel architecture consisting of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously. GPUs are markedly faster than CPUs, but only for certain tasks. GPUs may have some limitations in rendering complex scenes due to interactivity issues when using the same graphics card for both rendering and display, or due to insufficient memory. And while CPUs are best suited to single-threaded tasks, the tasks of modern games become too heavy for CPU graphics solutions.

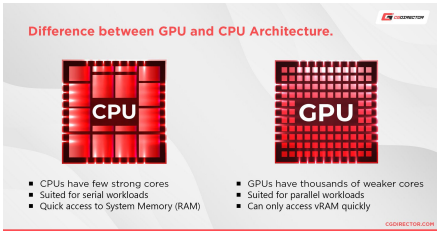


Fig 1: Difference between GPU & CPU Architecture [1]

2.1 CPU Performance along CA Tradeoffs

Suppose you are a developer at your workstation that is testing out a new CPU. You must keep in mind that Rendering uses all cores of your CPU; 100% of the time while rendering. Thus you would want as many cores in the HPP as possible even if these cores are clocked relatively low. "This is because the render engine assigns a so-called "bucket" to each core in your CPU. Each individual core will render its bucket and then get a new bucket once it's finished rendering the old one."[1]

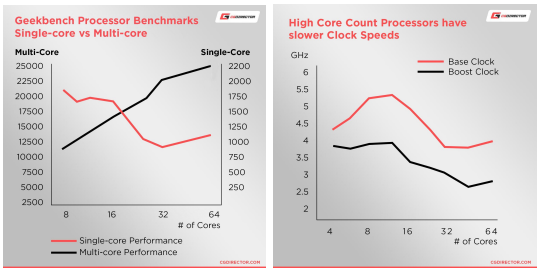


Fig 2: Processor Core & Clock SpeedBenchmarks[1]

However there comes a tradeoff in performance even if your budget warrants for more cores with higher clock speeds that needs to be accounted for. This is proportional between the number of CPU-cores and clock-speeds as seen in the second table above.Such CPUs will run and render fast but with a higher power consumption and likely reach assigned heat limits.[1] These will be hotter than lower clocked cores and can't be avoided upon application. Manufacturers like AMD built in architectural features like Turbo Core to "automatically overclock Cores until thermal and power limits are reached". Similar to Intel's Turbo Boost, there are dynamic auto-overclocking features that are governed by power draw to run the processor at the highest frequency within an extended range for better cooling.

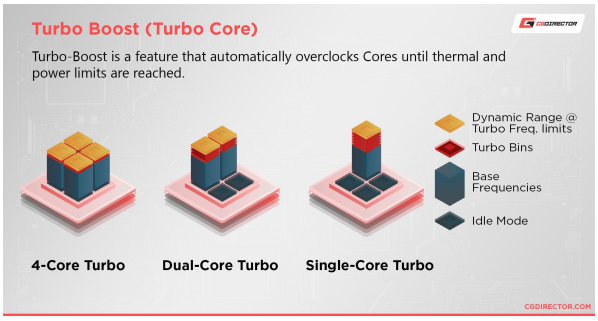


Fig 3: Turbo Boost [1]

2.2 CPU Rendering Advantages

With the mentioned differences in the CPU and GPU architecture, CPUs are adept to handle more complex tasks where they are able to synchronize several different tasks together unlike GPUs. Thus your studio's ideal CPU rendering use case would be an architectural project that requires different rooms to be designed differently. For instance, Games and Modeling Software that also use CPUs are handled differently thus there are many hardware options that are designed specifically for a certain render engine. Thus in general, CPUs are given priority in processing large and complex 3D scenes.

GPUs come to this issue due to non optimal usage of memory where their memories aren't stackable (even if you add multiple GPUs to your device). You are susceptible to crashes when confronted with over demanding tasks. CPU's can go up to 768GB RAM (unlike GPUs' 80GB cap) thus the worst outcome would be slower performance due to architectural designs like Turbo Boost. Crashes during client demos can be unprofessional and pile-up billable hours while the application recovers during a studio's working hours. Due to direct access to the device's hard drives and main system memory, this enables CPUs to hold a greater amount of data that is expandable and effective. This comes with overall general stability and tunability due to the maturity of the available tools. CPU renderers are built-in to be completely integrated with your computer atmosphere and offer a seamless user experience. Frequent Driver Updates for an external device like GPUs can prove to be a hindrance in the workstation.Where adding features for a current CPU have more familiarity amongst developers when it comes to programming them.

2.3 CPU Rendering Disadvantages

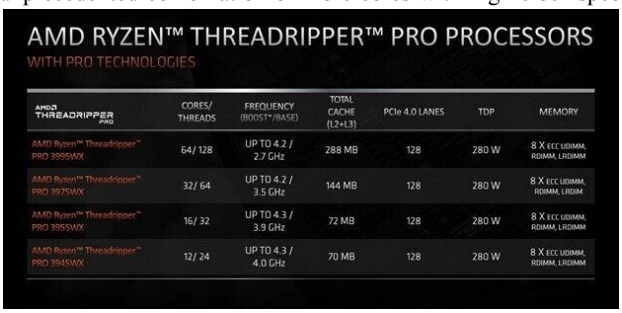
However, CPUs architectural designs come with their own drawbacks in real-life applications. When wanting to upgrade a current device's CPU, scalability of your device is restricted with the surrounding hardware's upgrade limitations like added costs to upgrade to a suitable motherboard with a new CPU.

Knowing if your CG algorithms aren't suited to be parallel is another big indicator to go with CPU Rendering. Given its design, these processors are suited with parallelism where GPUs continue to excel in such a domain. Only recently have GPU Render Engines such as Octane, Redshift, V-RAY RT, or FurryBall become mature enough to slowly but surely overtake CPU Render Engines in popularity.Especially, because GPU Render Engines are much faster in many cases and allow for extremely interactive preview Renderers. Changing to a GPU Render Engine is one of the top methods in achieving faster renders – given your GPU is strong enough and has a sufficient amount of VRAM. Those built-in CPU Renderers are beginner friendly, CPU's are still working on accelerating a 3D-Artists Workflow that GPU's excel in to attract intermediate developers to often switch to costly 3rd Party GPU Render Engines.

3. AMD CONTRIBUTIONS TO VFX

AMD takes the Hardware-centric approach to introducing new CPU architecture and applications in studios from various sectors of the Entertainment Industry. They have increased core counts, IPC, and clock speeds to branch to various content pipelines like movie production and gaming. This shift in the industry's expectations on CPUs came from AMD Ryzen™ Threadripper™ PRO Processors where they address a long-standing dilemma of which to prioritize: high CPU clock speeds or large number of CPU cores.[7] Each hardware option had benefits to lightly-threaded applications and heavily threaded applications respectively. This decision was followed by greatly increased costs and power consumption of studio machines to implement higher core counts for performance gains. The entry-level AMD Ryzen Threadripper PRO 3945WX is the first twelve-core workstation CPU with a boost clock speed up to 4.3GHz, while the AMD Ryzen Threadripper PRO 3995WX is the first workstation processor to offer 64 cores in a single CPU socket.[2] These processors are built to eliminate such difficult choices to deliver computer-intensive performance that's driven by an

unprecedented combination of more cores with high clock speed.



AMD THREADRIPPER PRO	CORES/ THREADS	FREQUENCY (BOOST*/BASE)	TOTAL CACHE (L2+L3)	PCIe 4.0 LANES	TDP	MEMORY
AMD Ryzen™ Threadripper™ PRO 3995WX	64/ 128	UP TO 4.2 / 2.7 GHz	288 MB	128	280 W	8 X ECC UDIMM, RDIMM, LRDIMM
AMD Ryzen™ Threadripper™ PRO 3975WX	32/ 64	UP TO 4.2 / 3.5 GHz	144 MB	128	280 W	8 X ECC UDIMM, RDIMM, LRDIM
AMD Ryzen™ Threadripper™ PRO 3955WX	16/ 32	UP TO 4.3 / 3.9 GHz	72 MB	128	280 W	8 X ECC UDIMM, RDIMM, LRDIMM
AMD Ryzen™ Threadripper™ PRO 3945WX	12/ 24	UP TO 4.3 / 4.0 GHz	70 MB	128	280 W	8 X ECC UDIMM, RDIMM, LRDIMM

Fig 4: AMD Ryzen Threadripper Features Table[2,7]

With the additional support of PCIe® 4.0, these processors enable individuals to benefit from the full power of the other components in their machines from multiple GPU set-ups to modern NVMe solid state drives.

At SIGGRAPH 2021, the premier conference & Exhibition in Computer Graphics & Interactive techniques, AMD showcased their processor's ability to provide performance beyond compromise to standardize on a single workstation configuration that can "efficiently address different professional application bottlenecks."[8]A production pipeline consists of Planning, Production, and Post-Production. For Planning and Production, virtualization computing is an avenue where CPUs contribute to build out innovative new solutions to keep a studio's architects, designers, artists and other visualization specialists reliably connected, productive and secure. CG on stages with LED Walls to replace physical production lights and camera personal movements lets you blend Production and Post-Production within a specific platform like Unreal Engine. This CPU-intensive method is called Light Baking Process to speed and minimize the steps between visualization to virtualization.

CPU-intensive workloads like virtualization have traditionally been left out the pipeline but Thread Ripper PRO helps virtualize workstation needs to reduce the cost of creativity and collaboration with a single virtualized machine (Intel also introduced this concept via invested software engineering). VoxelVision as a Balitimore-based firm sets up multiple VMs on individual workstations to enable numerous artists to work remotely under a single AMD Processor.[5] The high core count and clock speeds enabled them to use a range of single and multi-threaded applications for various rendering tasks.

AMD focuses on further customer support with application vendors like Red and Sony to work on high-resolution dailies like 8k resolution post-production in real time as a small footprint of one CPU. Within Post-Production, AMD focuses their CPUs to revamp methods to visualize high-resolution content in CG on the same CPU. If a studio works with background effects like lighting and simulation/water effects then there is a definite need of high-core count to get final renderings in high efficiency power packages. Modern HPPs can provide High-fidelity displays multiple times for an artist to work on where it iternates and render way faster than they used to (with energy efficiency in mind). In a studio environment, AMD Ryzen Threadripper PRO aims to deliver the most number of pixels per watt, which helps with limitations in power/cooling at a studio's data center.

4. GPU RENDERING

GPU rendering uses a graphics card for rendering in place of a CPU, which can significantly speed up the rendering process as GPUs are primarily designed for quick image rendering. GPUs were introduced as a response to graphically intense applications that burdened CPUs and hindered computing performance.GPU rendering takes a single set of instructions and runs them across multiple cores on multiple data, emphasizing parallel processing on one specific task while freeing up the CPU to focus on a variety of different sequential serial processing jobs.

4.1 GPU Rendering Advantages

GPU Rendering is best suited for single-threaded tasks in modern games that become too heavy of a workstation workload for CPU graphics solution. This accounts for the scalability options in multi-GPU setups that will consume less power than when using CPUs. Keep in mind, GPUs tend to also be priced lower per a unit's computation power when comparing a similar CPU hardware. Many modern render systems like Arnold (Autodesk), Iray (NVIDIA), and Redshift (Redshift Rendering Technologies) are suited for GPU software and hardware, which are designed for massively parallel tasks and can provide overall better performance. GPUs offer superior memory bandwidth, processing power, and speeds up to 100 times faster to tackle several tasks that require multiple parallel computations and large caches of data.

4.2 GPU Rendering Disadvantages

This type of rendering has some limitations in rendering complex scenes due to interactivity issues or insufficient memory when using the same graphics card for both rendering and display. Memory issues come from their lack of direct access to the main system memory or hard drives where they must communicate through the available CPU in the workstation. Crashes around such issues can disrupt the creative process of studio artists and their ability to stay in the "zone". For instance, a 1 hour per day of downtime due to unreliable GPU rendering can equal a loss of 12% of an artist's 8-hour day.[13] Specifically, GPUs reach high VRAM consumption due to scene complexity but adding an additional GPU tends to be the best option to resolve this concuring issue. As an animator, you can render the Foreground and Background separately and then merge both render outputs in a Compositing/Image Editing software. However, memory optimization techniques like this separation approach, optimizing a 3d scenes' textures or geometry are time-consuming and add tasks on a workload if not automated already.[6] This might also opt you to close other Memory Intensive applications to increase the amount of memory available for the rendering process. However if the scene is massive and there is no way to fit it into the GPU memory, it is still possible to render it with a GPU by using Hybrid Rendering with CPU enabled only. Hopefully, future evolutions that account for this dilemma can come from using Heterogeneous Processing where certain tasks are designated to the CPU & others via a larger platform management.

5. CPU+GPU Heterogeneous Processing

Although better CPU-GPU collaboration is inevitable to achieve high-performance computing, there haven't been many

applicable breakthroughs for rendering purposes with this specific use case of heterogeneous computing.

Heterogeneous platforms are Computer Systems that combine main processors and accelerators which in this rendering use case: CPU + GPU. In theory, this computer architecture yields these design optimizations compared to standalone processors in a workstation. A definite increase in performance where both devices work in parallel. Thus, there is an expected decrease in data communication (Which is often the bottleneck of the system) where different devices have different assigned roles within the provided task criterion.[3] Such a platform increases flexibility where the element of choice comes in the picture for what togglable *PUs for execution. Consequently, an increase in reliability with a fall-back solution when one *PU fails. Given all such design optimizations, there would be a clear increase in energy efficiency.

The efficient utilization of platform resources can be automated and maximized with future improvements to the Heterogeneous Earliest Finish Time heuristic. Heterogeneous Earliest Finish Time (or HEFT) is a heuristic to schedule a set of dependent tasks onto a network of Heterogeneous workers taking communication time into account. Load deployment decision algorithms similar to HEFT, can greatly improve the utilization rate of GPU and the speedup ratio in the calculation process.[4]

When dealing with a real-life rendering engine's workload, CPU+GPU Heterogeneous Processing would be an optimal alternative to consider when it has a load balanced algorithm with shorter scheduling length and higher efficiency. Unfortunately, there haven't been fruitful render-specific load balancing algorithms that companies like Intel and AMD can pitch to studios and their application users in the present day. Thus, companies like Intel took different software design optimization techniques like Platform Cooperative Computing.

6. HPP RENDERING COMPARISONS

Prior to delving into Platform Cooperative Computing, you can now deduce that comparing and choosing one High Performance Processor (HPP) over another isn't as straightforward as it looks. While GPU rendering has its own advantages, both CPU-based renderers and GPU-based renderers are perfect for their specific tasks. In fact, GPU only serves to enhance your current CPU system and allows you to significantly accelerate image rendering. Your GPU can take on the resource-intensive 3D visualization elements, as your CPU executes the remaining tasks.

Keep in mind, neither CPUs or GPUs are intended to completely replace the other in workstations and workflows. It ultimately depends on what the animator or studio needs. These processing units do live and operate in a synergistic harmony where they both aim to accelerate and streamline existing practices and workflows, maximize output, and offset processor-heavy computations in applications. However, there are certain performance metrics that studios prioritize to customize their budget distribution for both processors to their workloads to come.

6.1 Speed

If speed is the main priority in a studio's workflow, GPU-based rendering is the preferred solution. As a relatively newer technology, GPU render engines are heavily focused on the speed of image rendering processes. As many Label GPUs are to be the future of rendering, there are instances where a single GPU has the same processing power and features that can only be matched by an entire cluster of CPUs. With more core processors to come, GPU renderers can get the work that used to take hours for CPUs, completed in minutes.

6.2 Initial Graphic Fidelity

Though it might take hours (maybe even days) to finish rendering an image, traditional CPU-based rendering is more likely to deliver higher image quality and much clearer images that are devoid of noise.When several CPUs are interconnected and put to use in a render farm-like environment, for example, they can potentially produce a more exquisite final result than a GPU-based graphics solution could.If a studio is willing to take their time and isn't pressured by deadlines to get the very best possible image, then CPU-based rendering may be what they are looking for.

6.3 Cost

As image quality and animation continue to expand, high-end designs require resource-intensive systems that, in turn, require investment-heavy CPU rendering clusters.On the contrary, GPU render engines allow you to accomplish intricate rendering tasks (like glossy reflection and depth of field) and install multiple units to churn out studio-quality images at a significantly lower price point.

Without the need for expensive CPU render farms, individual creators can afford and rely on their own GPU workstations and have studio-quality work for a fraction of the cost. The best GPU for rendering depends on the intended use and budget.

6.4 Real Time Visualization

Real-Time Rendering is the prominent rendering technique used in interactive graphics and gaming where images must be created at a rapid pace. Because user interaction is high in such environments, real-time image creation is required. Dedicated graphics hardware improves on the long time to set up a scene and manipulate lighting visually in a software's viewport. Prior to the viewport display, the GPU usually has to wait for the CPU to finish its tasks to continue working. It means quite frankly that having lots of CPU-Cores will do nothing towards speeding up your modeling and does not usually make your Viewport faster. A GPU can drive viewport performance in your studio's software, allowing for real time viewing and manipulation of your models, lights, and framing in three dimensions. In general, a Processor is usually the bottleneck in having a snappy Viewport (esp a CPU).

It's clear the benefits of working and rendering with GPU-accelerated machines outweigh the traditional CPU-based workstations that slowed down production or limit project budgets due to potentially necessary upgrades.

7. INTEL CONTRIBUTIONS TO VFX

In terms of rendering, Intel has pushed for Platform Cooperative Computing as it's Design Optimization. This type of computing is similar to Heterogeneous Processing where they don't deal with a Separate CPU and GPU Backend systems. In

SIGGRAPH 2021, Intel presents "Intel oneAPI Rendering Toolkit" as part of its platform approach to rendering that merges all separate workflows to make it more comprehensible, cross-platform ready, and scalable.[12] Unlike AMD, Intel's software's strong suit lies in Openness and Flexibility for their product users that range from individual creators to large-scale studios. Intel gives the developer better insight in the code via open environments where development is easier and straightforward via platform-specific tasks. This circumvents the necessity to make the ambiguous hardware decisions to choose a processor that is good at certain things and not others.

Intel's oneAPI has the platform advantage to have a plethora of things a user needs at their fingertips. These include High-Performance Hardware like Intel Optane Persistent Memory that compliments DRAM with lowering the impact/occurrence of crashes, strengthens auto-save/instant crash recovery features, and augments the in-memory snapshot tools of 3d assets.[13] Other Advanced Capabilities are in the horizon with AI, Advanced Ray Tracing features, and broad ecosystem collaboration. Given the diversity of Ray-Tracing, Intel oneAPI Rendering Toolkit accounts for a variety based on their own performance & memory metrics.

In terms of AI, Intel oneAPI gives "AI acceleration support in GPUs to bring new ways to render a frame faster and better. Via addressing development flow challenges between a concept to its graphical output, AI avoids redundant development steps that 3d artist, a developer, and a gpu take in real time between concept and output.[15] To do this, oneAPI AI intends to enact "a pipeline to capture hierarchical data collection as input and ray traced output as target as seen Figure".[15]

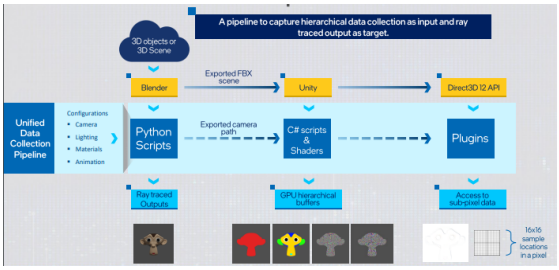


Fig. 5 Unified Data Collection Pipeline [15]

A DExperience Company, Dassault Systemes (3DS) uses oneAPI's rendering capabilities to provide Global Illumination Rendering to power Car Brand Applications in Design & Engineering and Marketing.[14] Their transition to oneAPI's Wavefront render from previous iterative CPU renders yielded performance Improvements in the time to compute images with defined quality as seen in the image below.

Fig. 6 Wavefront Render Images & Performance Gain[14]

Given the accessibility of oneAPI's platform resources, 3DS also utilized Intel Open Image Denoise, Threading Building Blocks, and Embree version 3 to then have continuous performance improvements in denoising and huge cost savings.

In essence, Intel is going beyond CPU/GPU-centric computer architecture design with its oneAPI Rendering Toolkit. This toolkit has open-source core libraries, capabilities, and components like Intel openVKL, Intel OSPRay, Intel OpenSWR and the previously mentioned ones used by 3DS.[12] These prove to be flexible and cost-efficient to create the professional-level renderings via ray tracing like global illumination. With AI capabilities and more system memory space, oneAPI is an innovative platform to support industry-wide integrations of Render Tools, Game Engines, and 3D Software. Such a plethora of platform features enables modern graphics applications to scale with their high-performance and high-fidelity outputs.

8. CONCLUSION

In conclusion, we explored Global Optimization techniques of both Hardware and Software revolving around Rendering in VFX Production. These are illustrated from both the designer and the user viewpoints of how they make their decisions to configure/use their products in the 3D rendering pipeline. Via referencing AMD ThreadRipper PRO's capabilities, CPU vs GPU rendering, and Intel's oneAPI Rendering Toolkit + Platform, this paper demonstrated a comprehensive understanding how High Performance Processors innovate and integrate themselves in the 3D Rendering Pipeline in the VFX & Animation industry. These were shown to exemplify how a computer architect designs their product to meet the user's expectations in its functional specs, pipeline objectives, and performance goals without ignoring the current Marketplace demands and technological trends. As such advancements constantly change, adaptability and usability of every processor, render engine, and software interface will come into question for a studio's productivity to bring their visions to life. Such a dynamic environment of Rendering applications requires a dynamic plethora of design avenues to choose from when it comes to computer architecture design and pipeline integration guidelines.

9. REFERENCES

[1] Alex Glawion, "Best Workstation Computer for 3D Modeling and Rendering" cgdirector.com. https://www.cgdirector.com/best-computer-3d-modeling-rendering/ (accessed Nov 10th, 2021)

[2] Jeff Mottle, "Why AMD Ryzen™ Threadripper™ PRO Processors continue to rewrite the rules of visualization" cgdirector.com. https://www.cgarchitect.com/features/articles/996cf095-why-amd-ryzen-threadripper-pro-processors-continue-to-rewrite-the-rules-of-visualization (accessed Nov 10th, 2021)

[3] Sparsh Mittal,and Jeffrey S. Vetter, "A Survey of CPU-GPU Heterogeneous Computing Techniques", ACM Computing Surveys Volume 47 Issue 4 Article No.:69 pp 1–35, July 2015. [Online] Available: https://dl.acm.org/doi/10.1145/2788396

[4] Di Yang, Jinquan Ma,Chunsheng Yue, ZHICHONG Shen,and Xiaolong Shen, “Load Deployment Decision Algorithm Based on CPU+GPU Heterogeneous Processing Platform” ICBDC 2021: 2021 6th International Conference on Big Data and Computing Pages 166–171, May 2021 [Online] Available: https://dl.acm.org/doi/abs/10.1145/3469968.3469996

[5] IDG Communications, “Visualize What Comes Next” computerworld.com. https://www.computerworld.com/article/3636412/visualize-what-comes-next.html (accessed Nov 10th, 2021)

[6] “Memory Usage Optimizations for GPU rendering” docs.chaos.com. https://docs.chaos.com/display/KB/Memory+Usage+Optimizations+for+GPU+rendering (accessed Nov 10th, 2021)

[7] “AMD Ryzen™ Threadripper™ PRO Processors For VFX Artists” amd.com. https://www.amd.com/en/processors/workstation-media-entertainment (accessed Nov 10th, 2021)

[8] James Knight,Chris Hall, David Bitton, and Eddie Camara “The Impact of AMD Processors in Media & Entertainment - Presented by Hypertec” siggraph.org. https://s2021.siggraph.org/presentation/?id=exs_128&sess=sess548 (accessed Nov 10th, 2021)

[9] Marc Potocnik, “Advantages of Powerful CPU Rendering with Intel (renderbaron) - Presented by Intel” siggraph.org. https://s2021.siggraph.org/presentation/?sess=sess576&id=exs_146 (accessed Nov 10th, 2021)

[10] VB Staff, “‘No pixel left behind’: The new era of high-fidelity graphics and visualization has begun” venturebeat.com. https://venturebeat.com/2020/09/22/no-pixel-left-behind-the-new-era-of-high-fidelity-graphics-and-visualization-has-begun/ (accessed Nov 10th, 2021)

[11] “SIGGRAPH 2021: In Case You Missed It” intel.com. https://www.intel.com/content/www/us/en/events/developer/siggraph-2021.html (accessed Nov 10th, 2021)

[12] Jim Jeffers, “High-Fidelity Ray Tracing Pushing New Boundaries” intel.com. https://www.intel.com/content/dam/develop/external/us/en/documents/high-fidelity-ray-tracing-siggraph-2021.pdf (accessed Nov 10th, 2021)

[13] Ben Looram, Andrew Degnan, Charlie Yu, and MemVergeMark Wright, “Recover in a Snap[shot] with Intel® Optane™ Persistent Memory and MemVerge*” intel.com. https://www.intel.com/content/dam/develop/external/us/en/documents/pdf/rick-hack-siggraph-2021-memverge.pdf (accessed Nov 10th, 2021)

[14] Jan Meseth, “Boost Cloud Global Illumination Rendering Capabilities with Stellar Physically Correct & Intel oneAPI Rendering Toolkit” intel.com. https://www.intel.com/content/dam/develop/external/us/en/documents/pdf/jan-meseth-siggraph-2021-stellar-physically-correct.pdf (accessed Nov 10th, 2021)

[15] Selvakumar Panneer, “Simplify Data Collection for AI on CPU and GPU Platforms Using Blender* and Unity* Graphics APIs” intel.com. https://www.intel.com/content/dam/develop/external/us/en/documents/pdf/selvakumar-panneer-data-collection-siggraph-2021.pdf (accessed Nov 10th, 2021)

E - Branch Prediction: A Retrospective

Inderpreet Singh

Abstract

Modern Branch Predictors predict the correct output with a very high accuracy. This results in achieving higher levels of efficiency, reducing energy consumption and maximizing performance. This paper surveys various forms of branch prediction techniques. This paper presents a high level overview of Branch Prediction techniques. It covers the history of Branch prediction and how branch prediction algorithms have evolved over the decades. The paper gives a brief overview of static branch prediction, dynamic branch prediction and deep learning based branch prediction algorithms. This paper describes the advantages and disadvantages associated with each of these techniques.

Introduction

In Computer Science terminology, prediction may be defined as the conditional execution of instructions. Historically, most computer architectures have implemented the conditional execution of instructions through branches. Speculative execution of instructions is a critical component in most modern computer architectures [1]. This speculative execution of instruction is necessary to facilitate instruction level parallelism. Correct prediction significantly simplifies compiler optimization and removes branches. Modern computer architectures implement a long pipeline consisting of fetch/ decode/ execute/ store stages. The lifecycle of an instruction can be described as follows: an instruction is fetched, decoded, executed, the state variables are stored in memory and then the instruction is retired. Older microprocessors required an instruction to complete its entire lifecycle (i.e fetch/decode/execute/retire) before the next instruction was fetched. This approach was highly inefficient because of obvious reasons. For example, if an instruction is in the execution stage, the fetch, decode and other units of the microprocessor are sitting idle. Modern processors overlap these processes. When one instruction is being retired, the next one is being executed and the next one being decoded and the next one being fetched.fetched. This overlap between instructions poses a bigger challenge. When an instruction is being executed, it may change the instruction which needs to be executed next. If the wrong set of instructions followed the current executing instruction, the microprocessor has to back up,

fetch the correct instruction and so on. Hence predicting the correct instruction becomes of paramount importance.

Branch predictions have evolved over the decades. Early branch prediction algorithms used static information for prediction. This included compile time information and did not take into account the run time (execution) dynamics of a program being executed. The information used as input for static branch prediction was generated during the compilation of the program. Static techniques are easy to implement in hardware and inexpensive addition to the microprocessor architecture.

Dynamic Branch Prediction uses dynamic information to predict the correct branch. This information is generated during the execution phase of a program. Prediction is based on the history of execution of branches. Dynamic algorithms use time varying and input dependent execution parameters of a branch. These algorithms are more complex to implement as compared to static algorithms.

The advent of Artificial intelligence and machine learning has opened new areas for improvement in branch prediction. Machine learning algorithms are well suited for the task of prediction. AMD's piledriver microarchitecture was the first one to implement a neural network based algorithm for branch prediction [2]. One of the main advantages of deep learning based prediction algorithms is its high efficiency. The order of growth of resources required for dynamic branch prediction algorithms increases exponentially as the workload increases [6]. Comparing this to Deep learning algorithms, they only require a linear increase in resources as compared to dynamic algorithms [6].

Static Branch Prediction

Static branch predictors were the first algorithms designed solely for the purpose of branch prediction. They were first released for commercial use in MIPS and SPARC micro architectures which were the very first implementation of RISC architecture [3]. Static branch predictors use statistical information such as the history of a branch, compile time statistical data as input parameters for prediction. All decisions made by the static branch prediction algorithms happen at compile time [3]. These algorithms do not consider run time information of a program for prediction. Static Branch predictors are easy to implement in hardware and require less resource utilization.

Dynamic Branch Prediction

Dynamic branch prediction uses runtime information of a program to predict correct branches. Prediction is based on the history of branch execution [4]. They are more complex to implement than static branch prediction algorithms but guarantee significant accuracy improvement over static branch predictors. There are many variations of dynamic branch prediction algorithms [4].

Static branch Prediction using neural networks

There is a large amount of literature where the power of neural networks has been harnessed to perform branch prediction [5]. Inputs to such a network usually consist of compile time statistics of a program. Statistics such as opcode information, data dependency and control flow are provided as an input to a trained neural network [5]. Most statistical branch prediction techniques have less accuracy than dynamic branch prediction techniques but are worth considering due to its low overhead.

Deep Learning based Branch Prediction

Branch prediction with perceptrons

This section will give a brief overview of one of the simplest neural networks (perceptron). Understanding of this section is necessary to fully understand how the algorithm works. This work was first described by [6].

Why Perceptron?

Perceptron is one of the simplest neural networks known. One of the main advantages of using a perceptron is that it is easy to implement in hardware. Many other deep learning algorithms can also be used for the prediction problem. However, perceptron is highly advantageous because other techniques are significantly difficult to implement in hardware.

Another advantage of perceptron is that by understanding the correlation of weights, it is easy to deduce the mechanism by which the output is produced. A major negative point about neural network architectures is that it is very difficult to deduce the inner workings of the network by looking at the weights. Perceptron usage is therefore encouraged. The decision making of a perceptron is well understood and can be deduced by looking at the mathematical formulas [6].

How Perceptron works?

A perceptron models neurons of the human brain. I will consider a single perceptron for explanation purposes. Perceptrons can be combined in many different forms to form various types of neural networks. Hence, understanding of the basic building block of any neural network (a perceptron) is of paramount importance when trying to understand deep learning based solutions for branch prediction.

A perceptron outputs a boolean value given n different inputs. It learns to output a boolean function based on the inputs. In case of branch prediction, the inputs can be specified by the computer architect and can consist of compilation statistics, run time statistics, global branch history, etc. The figure below depicts a single perceptron [6].

$$y = w_0 + \sum_{i=1}^{n} x_i w_i .$$

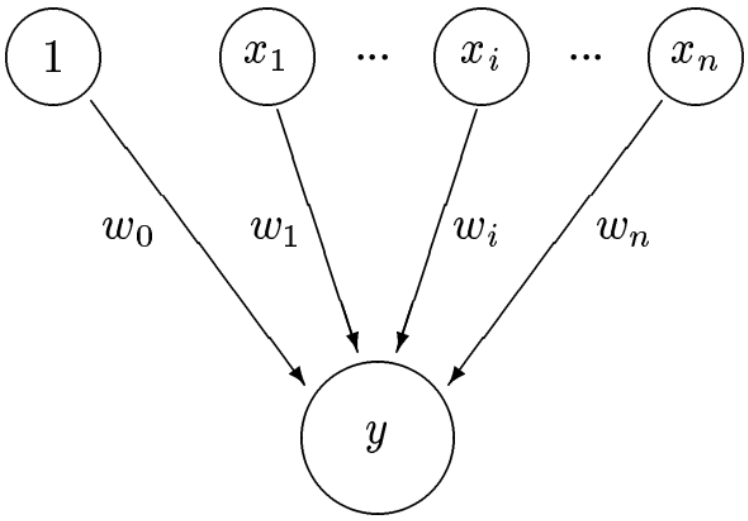


Figure 1: Perceptron Model

Working

A perceptron based solution is implemented to deduce the correlation between different input parameters. These parameters are given in the form of weights. Larger the magnitude of a weight, the larger is its importance. Figure 2 below illustrates the schematics of a single layer perceptron based branch predictor [6]. In this design the microprocessor needs to keep track of n

perceptrons in main memory. The number of perceptrons is based on the energy usage, hardware budget of the machine among other concerns. Circuitry labelled as y in Figure 2 below produces the training data required for predicting the correct branch. The working of the schematics labelled in figure 2 is described as follows. When an instruction is in its fetch stage, its branch address is hashed to form an index. The perceptron corresponding to the index is fetched into a vector register of weights. Output y is computed by taking the dot product of the weights and the global history register. Negative value of y means that the branch is not taken and positive value means that it is taken [6].

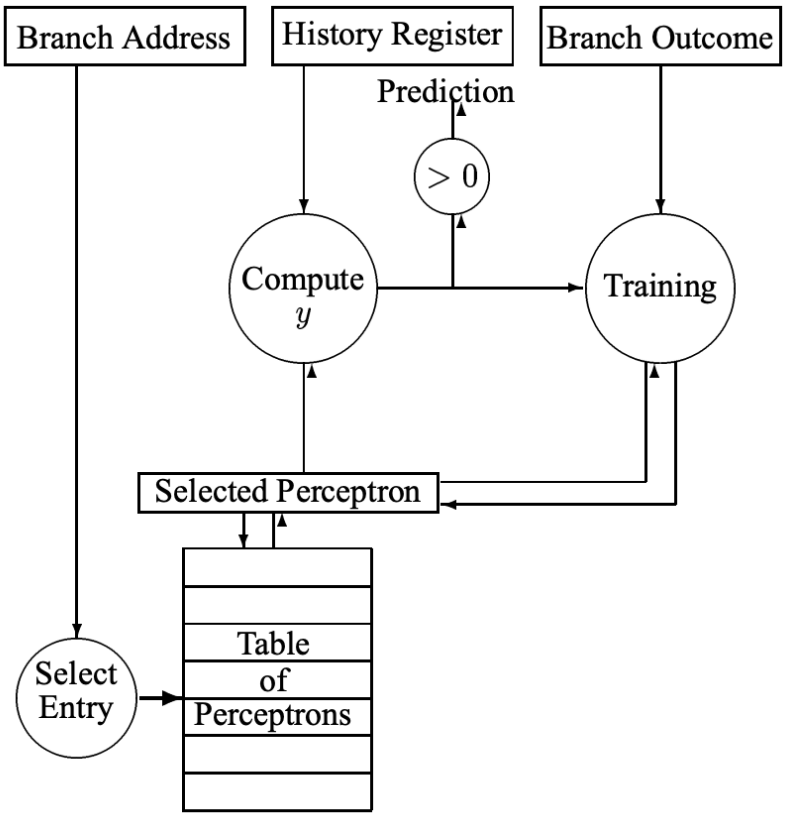


Figure 2: Perceptron Predictor Model Diagram

Conclusion

This paper has surveyed various algorithms which have historically been used for Branch Prediction. The paper gave a brief overview of the importance of a branch predictor and details how a branch predictor improves performance, reduces energy consumption and boosts instruction level parallelism.

The paper then describes various techniques such as static, dynamic and neural network based branch prediction. The paper provides a detailed overview of the advantages and disadvantages of each of these techniques. Branch prediction using a simple neural network is described in detail. One of the earliest uses of deep learning based models for branch prediction was the use of a simple perceptron. The paper describes the working of a perceptron and how it can be used for the task of branch prediction.

Most computer architectures in use today deploy a combination of dynamic branch prediction algorithms combined with a neural network based model. AMD's piledriver architecture was the first commercial use of neural networks for branch prediction.

References

[1] J. L. Hennessy and D. A. Patterson. Computer Architecture: A Quantitative Approach, Second Edition. Morgan Kaufmann Publishers, 1996.

[2] Walton, Jarred (2012-05-15). "The AMD Trinity Review (A10-4600M): A New Hope"

[3] Fog, Agner (2016-12-01). "The microarchitecture of Intel, AMD, and VIA CPUs"

[4] Cheng, Chih-Cheng. "The Schemes and Performances of Dynamic Branch predictors"

[5] B. Calder, D. Grunwald, M. Jones, D. Lindsay, J. Martin, M. Mozer, and B. Zorn. Evidence-based static branch prediction using machine learning. ACM Transactions on Programming Languages and Systems, 19(1), 1997.

[6] JIMENEZ ´ , D. A., AND LIN, C. Dynamic branch prediction with perceptrons. In High-Performance Computer Architecture, 2001. HPCA. The Seventh International Symposium on (2001), IEEE, pp. 197–206.

F - Design, features and architecture of the leading chips in Autonomous vehicles

Akshay Ravi
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000
rakshay0397@gmail.com

ABSTRACT

Autonomous vehicles are the future of road transportation. Technology has advanced so much that systems like self-driving cars are no longer just feasible, but are already being implemented in various countries . The chips used by the manufacturers for use in autonomous vehicles are some of the best chips in the world. The architecture and hardware is of great interest for various studies. All the companies constantly improve their chips for better performance, lesser power consumption and lower costs. This term paper would discuss the computer architecture of autonomous vehicles by giving attention to Nvidia's Drive hardware and software.

1. INTRODUCTION

An autonomous vehicle, or a driverless vehicle, is one that is able to operate itself and perform necessary functions without any human intervention, through the ability to sense its surroundings. The vehicle uses both hardware and software components in order to make this possible. These components make up what we call a fully automated driving system. They allow the vehicle to respond to external conditions that a human driver would manage. Autonomous Driving (AD) is a complex task that needs reliable on-board systems adaptable to any internal or external conditions.

The US Department of Transportation's National Highway Traffic Safety Administration (NHTSA) has defined five levels of autonomous driving[1]:

Level 0: No automation - It is like normal cars we have today, where the driver is the complete controller.

Level 1: Semi-autonomous cars - Here although most functions are controlled by the driver, the autonomous systems can do functionality like adaptive cruise control

Level 2: Driver-assisted automation: In this level, an autonomous system can take over steering, acceleration, and braking in specific scenarios. But, even though Level 2 driver support can control these primary driving tasks, the driver must remain alert and is required to actively supervise the technology at all times.

Level 3: Conditional driving automation - In this, the driver does not need to supervise the technology, which means they can engage in other activities. However, a human driver must be present, alert, and able to take control of the vehicle at any time, especially in the case of an emergency due to system failure.

Level 4: High driving automation - This autonomy does not require any human interaction in the vehicle's operation and is well programmed to drive under some specific conditions, vision is clear, speed is within some threshold value. The human driver is needed and has to drive if the condition changes.

Level 5: Full driving automation - This autonomy does not require any human interaction in the vehicle's operation and is well programmed to drive under all conditions, also it is programmed to stop itself in the event of system failure. A human driver is not needed.

2. HISTORY

Self-driving cars have been around for almost 100 years. In the 1920s Houdina radio controls showed off a radio-controlled car called American wonder. In the 1930s general motors sponsored an exhibition called futurama with radio-controlled cars using electromagnetic fields. In the 1950s rca labs introduced cars guided by wires and embedded circuits. In the 1980s vision guided systems from Mercedes Benz used lidar computer vision and robot control. In the 1990s, tests continued with the same technology across the US with Carnegie Mellon university's Nav lab project driving cross country.

In 2009 Google jumped into the autonomy game with their google xlab self-driving project. The project was headed by Sebastian Thrun, director of Stanford's Artificial Intelligence Laboratory. The goal was to launch a commercial vehicle by the year 2020. The team started out with seven prototypes, six Toyota Priuses and an Audi TT, which were souped up with an array of sensors, cameras, lasers, a special radar and GPS technology that allowed them to detect objects, people and numerous potential dangers up to hundreds of yards away, and circumnavigate a predetermined route. By 2015, Google cars had logged more than 1 million miles without causing an accident, though they were involved in 13 collisions. The first accident for which the car was at fault occurred in 2016. Further, the issue with autonomous cars was not just technology related, but also legality. It posed a difficult question to lawmakers because there is no surefire way to determine who is at fault for a collision between two autonomous vehicles. More importantly, all this progress has since spurred many of the biggest names in the automotive industry to pour resources into an idea whose time may very well have arrived[1].

Other companies that have started developing and testing autonomous car technology include Uber, Microsoft, Tesla as well as traditional car manufacturers Toyota, Volkswagen, BMW, Audi, General Motors, Honda and Tesla [1].

3. TECHNOLOGY STACK

The technology stack of a self-driving vehicle comprises the parts that make it autonomous. These are shown in the image below.

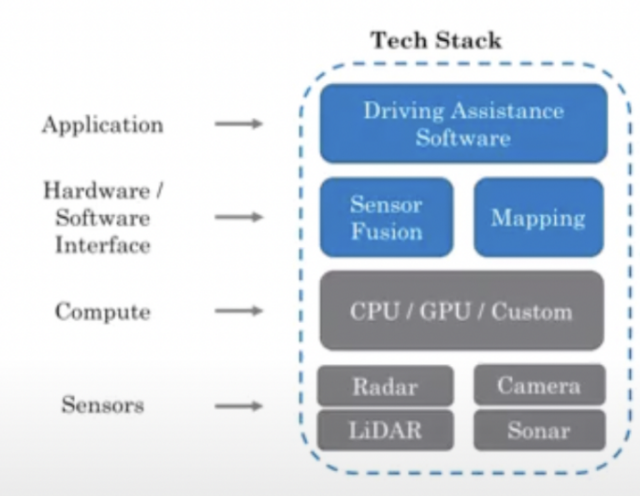


Figure 1. Technology Stack of an Autonomous Vehicle

3.1 Driving Assistance Software(DAS)

This is the control component. It routes the vehicle, dictates how the car must drive and helps with its decision making.

3.2 Sensor fusion and Mapping

This is the software component that supports the driving assistance system and takes signals from the sensor, processes them and maps them to the hardware.

3.3 CPU/GPU/Custom

These are processors that perform the computation i.e. number functions. This is where the intense number crunching happens. The chips are produced by various companies like Nvidia, Intel, Tesla, etc.

3.4 Sensors

Provide a perception of the real world to the vehicle. It detects movements around the vehicle and tells the software to take action based on that information. Sensors can be a combination of Cameras, Radars, Lidars and Sonars. Different vehicles use different numbers and combinations of sensors.

LIDAR stands for light detection and ranging. It is used for localization, mapping, and obstacle avoidance [2]. LIDAR uses infrared light and measures the time it takes for the light to bounce off an object and back to the sensor, thus creating a three-dimensional map of the surrounding area. It is used in iPhones as well. It is pretty reliable and can accurately determine the distance of the object from the desired point. It can work in the dark as well because it uses infrared light. We cannot say the same for cameras. Their drawback lies in their cost. Costing a few times more than cameras, they are not the ideal hardware sessor if a company wants to cut down costs.

Cameras are the other equally popular alternative because they can be embedded with image detection and object recognition algorithms. Unlike Lidars which can only tell us the distance, cameras can identify the object as well. The boom of computer vision helps the case for cameras as they can be used to identify humans, lanes, signals, animals etc. The drawback lies in the high computation required to process images from more than 5 cameras on a vehicle every second. This is definitely not an easy task. The fact that the images not only have to be processed in real-time, but also have to be transmitted to the automated driving software to help it make decisions immediately makes it even more difficult to implement in real-time. That being said, it is not that it cannot be used, because Google's Waymo does precisely that.

Radar and sonars use sound waves to calculate the distance of the objects in front. The distance information is used to immediately make decisions such as braking if there is an object in front. Radars can work in any weather condition and even at night. However, they cannot detect small objects and cannot identify images like cameras.

In the below image, Ford's second generation autonomous vehicle is shown with the placement of its sensors in the circled regions. The sensors above are designed well into the structure of the vehicle so that they don't stand out.

Figure 2. Ford autonomous vehicle and it's sensors

It is a challenge for companies to find the right balance of sensors. More sensors means increased cost but it also offers more failure protection. Less sensors leads to lesser costs but it also means it's harder to have backup in case of failure. Moreover, many sensors means more redundancy in the identification of objects around. This requires some amount of unnecessary computation and image detection algorithms.

3.5 Delphi and it's technology stack

All the parts in the stack are not always built in house by manufacturers. They are outsourced from different companies that specialize in specific components. In the below image, you can see precisely what companies Delphi uses to get its components.

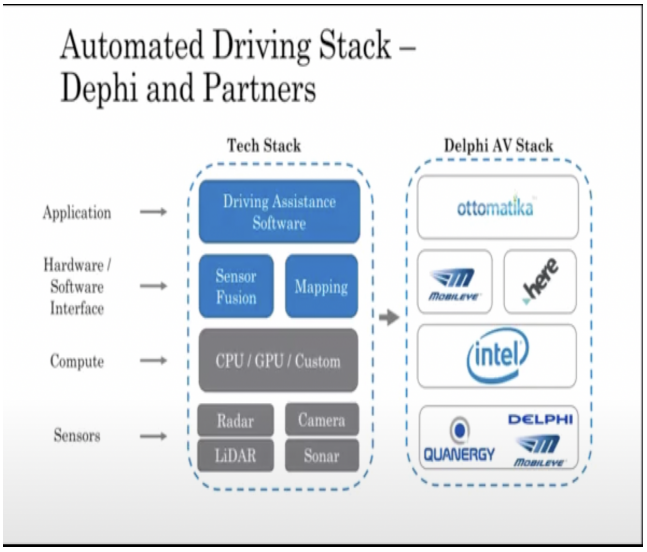


Figure 3. Delhi's technology stack

4. NVIDIA DRIVE SOFTWARE

The big question lies in how we test these cars. It's not exactly feasible to test a car out on the open road and let it learn. We need to test it on inner roads designed only for testing. However, with space constraints, it's not always feasible. Moreover, we cannot mimic a road network with traffic in the real world. Therefore, we simulate the hardware in software, very much similar to how we have learned VHDL. This is done in one example, by NVIDIA.

Nvidia creates GPUs (graphics processing units) and SoCs (systems on a chip) that are simply bought by automakers or their tier 1 suppliers and used in various cars on the road today. Nvidia also works much more closely with automakers to jointly develop software systems. Importantly, while Nvidia started out as a hardware company and is best known for that, it is now more of a software company than hardware company.

4.1 DRIVE family

The NVIDIA DRIVE family of products for autonomous vehicle development covers everything from the car to the data center. DRIVE Hyperion is the in-car solution, a vehicle architecture that includes sensors, DRIVE AGX for compute, and software tools necessary for robust self-driving and intelligent cockpit capabilities. In the data center, NVIDIA provides the hardware and software you need for AV development, including NVIDIA DGX™ to train DNN for perception and DRIVE Sim for generating data sets and validating the entire AV stack [3].

4.1.1 Drive Hyperion

NVIDIA DRIVE Hyperion is an end-to-end development platform which includes multiple DRIVE AGX Orin™ Developer Kits, each powered by an Orin SoC delivering up to 254 Trillion Operations per second(TOPS), 360 degree sensor support, and extra compute power to offload data recording and replay[3].

4.1.2 Drive SDK

NVIDIA DRIVE SDK comprises the foundational DRIVE OS and DriveWorks SDK, as well as advanced applications such as highly automated supervised driving (DRIVE AV) and AI cockpit (DRIVE IX). It includes perception, localization and mapping, planning and control, driver monitoring, and natural language processing[3].

4.1.2.1 Drive OS

NVIDIA DRIVE OS is a software stack consisting of an embedded real-time operating system that provides access to the hardware engines. It has multiple guest operating systems, a 64-bit user space,runtime libraries, NvMedia APIs for hardware-accelerated multimedia, camera input processing and Deep learning libraries[3].

4.1.2.2 Drive AV

NVIDIA DRIVE AV provides perception, mapping, and planning modules using the DriveWorks SDK.

DRIVE Perception: Detects, tracks, and estimates distances using DNNs and sensor data for obstacle, path, and wait perception.

DRIVE Mapping: Creates and updates HD maps and localizes the vehicle to a map.

DRIVE Planning: Plan and control the vehicle's motion, including path, lane, and behavioral planning[3].

4.2 Companies using NVIDIA's chips

When Nvidia released its third generation of the Drive PX automotive line, code-named Pegasus, the company noted that the new line represented the "first computer chips for developing fully autonomous vehicles." It also noted that it had over 25 customers working on the development of robotaxis, self-driving cars, and/or autonomous long-haul semi trucks. One of those partners was Volvo Cars through auto parts supplier Autoliv. Toyota and Volvo Group (trucks) uses Nvidia's tech from end to end, "from data collection, training, simulation, and deploying [4]. In February 2018, Germany-based auto parts supplier Continental chose NVIDIA for it's AI platform for the development of it's self-driving vehicle systems. In 2019, Xpeng P7 was the first production vehicle in the world to use the NVIDIA DRIVE AGX Xavier system-on-a-chip (SoC) autonomous driving hardware platform. Xavier delivers 30 TOPS (30,000,000,000,000 operations per second), which would have required a full-blown data center, a "full room of servers," not long ago. That's just one chip delivering 30 TOPS now. Xpeng is fully building the software stack on top of that, not Nvidia. "The company's Smart Electric Platform Architecture (SEPA) runs on 2 chips – Nvidia for the XPILOT and Qualcomm's Snapdragon™ 820A for intelligent services and infotainment, including cameras inside and outside, radars, HD-map, and ultrasonic sensors[5]. Nvidia is also working closely with Mercedes-Benz to develop a full smart EV platform which future vehicles will be built on top of and around. This includes a supercomputer AI system as well as the software to go on top of that. In this case, the Mercedes fleet will be using Nvidia's next-gen chip beyond Xaviar, Orin, which will offer *200 TOPS*, at under 70W. (Naturally, other automakers will be eager to use Orin as well.) Going further, Nvidia has a single board, integrating GPUs as well, that is able to process 2,000 TOPs. It's not clear who will be using that, but it would be used for robotaxis. Nvidia has been testing self-driving tech for years, including in the US (East Coast, Midwest, and West Coast), China, Japan, and Germany[6]. In 2016, Nvidia supplied Tesla with the hardware 2.0 computer that started getting installed in all Teslas produced from October 2016. This was

based on the Drive PX 2 platform. Tesla since 2020 however, decided to develop its own SoC and Full Self-Driving computer and hence would stop using Nvidia's hardware for these purposes. The reason, in part, was that Nvidia's offering was broadly based for multiple segments of the market and not tailored enough to an automobile, specifically a Tesla. Developing its own SoC, Tesla was able to optimize the specific performance it needed at much greater efficiency because it was tailored to one application — use in a car. The other main benefit of making their own chip is the ability to lower the cost for Tesla by around 20%. The FSD computer was designed by Pete Bannon and his team — Pete is renowned for previously having designed Apple's A5 through A9 CPUs, amongst other achievements[7]. Nvidia admitted that Tesla, by building its own chip, raised the bar for all the companies, including itself.

5. BENEFITS OF AUTONOMOUS VEHICLES

Despite the amount of work that has to go into producing a self-driving car, and the legal issues that surround them, if they are the talk of the town, there must be pretty convincing benefits that they provide. Let's look at some of the main ones.

Automated vehicles' potential to save lives and reduce injuries is rooted in one critical and tragic fact: 94% of serious crashes are due to human error. They can remove human error, which will protect drivers and passengers, as well as bicyclists and pedestrians.

Another factor to note is the cost of accidents. Vehicles cost a lot to repair and the availability of spare parts is also highly dependent on the market. Moreover, damages to sidewalks, fences and public property causes disruptions to daily life. By reducing accidents, these additional economic costs and repair times will be reduced.

Roads filled with automated vehicles could also cooperate to reduce traffic congestion. The "delay time" that humans take to start their car can be removed by communication between smart vehicles and the signals. "Delay time" refers to the time that is taken to start a vehicle after the signal changes from red to green. This is a result of the human time to analyse the change, start the vehicle, change the gear and accelerate. Once one car moves, only then will the car behind perform the same set of actions. All the vehicles don't start at the same time. If the vehicles were autonomous, they would start as soon as the signal change is detected.

In many places across the country employment rests on the ability to drive. Automated vehicles will provide new mobility options to millions of disabled people. Automated vehicles would provide this freedom to millions. These types of societal benefits are not quantifiable, but cannot be neglected.

6. SUMMARY/CONCLUSION

It is well known how powerful these smart chips can get and there is no stopping manufacturers from producing even more powerful machines. But the shift comes when there are electric vehicles in the market. Now, the integration of autonomy in vehicles is not limited to the current gas based cars, but also to the electricity-operated ones. There is no future for gas-based vehicles if the resources are depleted at the pace they are being used at now. Thus, numerous companies are shifting to electric vehicles. Tesla, and by no surprise, is leading the market in this sphere. They dominate the market with both electric and autonomous vehicles, all manufactured in house. They have a huge advantage at making their vehicles very well integrated with their software. The only hurdle remains the safety of the lives of many people. Reaching level 5 automation for all road conditions is not easy. There is a long way to go, but it is not impossible. We could see a future with integrated smart cities wherein the vehicles are routed by the signals and the road network is designed that way. The movement of autonomous vehicles in smart predictable environments has already been overcome. Thus, the playing field is wide open. Whether we would see smart cities built to incorporate autonomous vehicles or even smarter autonomous vehicles for the current cities, is an interesting area for research.

7. REFERENCES

[1] https://www.thoughtco.com/history-of-self-driving-cars-4117191

[2] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "Computer Architectures for Autonomous Driving," *Computer* , vol. 50, no. 8, pp. 18–25, 2017, doi: 10.1109/mc.2017.3001256.

[3] https://developer.nvidia.com/drive

[4] https://www.nvidia.com/en-us/self-driving-cars/partners/volvo-group/

[5] https://cleantechnica.com/2019/12/21/xpeng-in-house-autonomous-vehicle-technology-keeps-prices-down/

[6] https://cleantechnica.com/2020/06/23/mercedes-benz-nvidia-partner-on-autonomous-driving-numerous-questions-thoughts/

[7] https://cleantechnica.com/2019/04/23/tesla-has-raised-the-bar-for-self-driving-computers-admits-nvidia/

G - Overview of Cloud Computing Architecture

Xiaoli Tong
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000
Xiaoli.tong@sjsu.edu

ABSTRACT
The Cloud computing emerges as a new computing paradigm in recent years. It has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed. In this paper, the definition of cloud computing defined by different authorities has been described. In addition, deployment models and service models are illustrated, which is essential to understand the cloud computing technology. Four more common cloud architectures are introduced in the paper, each exemplifying a common usage and characteristic of cloud computing environment.

1. INTRODUCTION
Cloud computing technology grows rapidly recent years. It has become more and more popular through applications, services, storage, and computing over the internet. As more and more companies shift towards online storage and services, cloud computing technology becomes an important part of the business. Cloud consumers and cloud provider can benefit a lot from adopting cloud computing. First, it can reduce investments and proportional costs. Cloud providers can act as the wholesalers of IT resources and offer attractive priced service by utilizing resource pooling. For enterprises, they can reduce the up-front costs of purchasing and maintain of hardware, software, even upgrade the current system. They can utilize cloud services based on their demands, and increase their IT resources gradually. For consumers, they can pay the services based on pay-as-you-go. They are able to remove or add IT resource at fine grained level. Second, it can increase scalability. Cloud consumer can automatically allocate their IT resources based on their demands by utilizing the built-in features of clouds. Third, it can increase availability and reliability. The cloud service is available for a longer time period to cloud consumer, which can eliminate or limit the outrage. Cloud architectures provide extensive failover support that can increase the reliability.

2. DEFINITION
The definition of cloud computing is an essential part to understand the cloud computing. A Gartner report listing cloud computing at the top of its strategic technology areas by announcing its formal definition [1], "... a style of computing in which scalable and elastic IT-enabled capabilities are delivered as a service to external customers using Internet technologies." Forrester Research provides its own definition of cloud computing as: "…a standard IT capability (services, software, or infrastructure) delivered via Internet technology in a pay-per-use, self-service way." [1] NIST(National Institute of Standards and Technology) also provide its definition of cloud computing. This definition has received industry-wide acceptance. The definition is "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network across to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models." [2]

3. ESSENTIAL CHARACTERISTICS
The following five essential characteristics are defined by NIST for cloud environment. Cloud provider and cloud consumers can access these characteristics individually and collectively to the measure the value offering of a given cloud platform.

3.1 On Demand Self Service
A cloud consumer can unilaterally access cloud based IT resources, such as server time and network resources. Consumer can self-provision the IT resources automatically without any interaction from the cloud providers.

3.2 Broad Network Access
A cloud service can be widely accessible. Establishing ubiquitous access can require support from a range of devices, transport protocols, interfaces, and security technologies [1]. The cloud service architecture must be tailored to the particular needs of different cloud service consumers in order to enable the access.

3.3 Resource Pooling
The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demands [1]. Generally, cloud consumers have no control or knowledge over the precise location of cloud resources. But cloud consumer cloud can specify the abstract location, such as country, state, etc. The resources include network resource, storage, CPU, memory, etc. Figure 1 shows a single instance of IT resources, such as a cloud storage device, serves multiple consumers in a multitenant environment [1]. Cloud consumers A and B can access cloud service on the shared cloud storage device without knowledge of others existing.

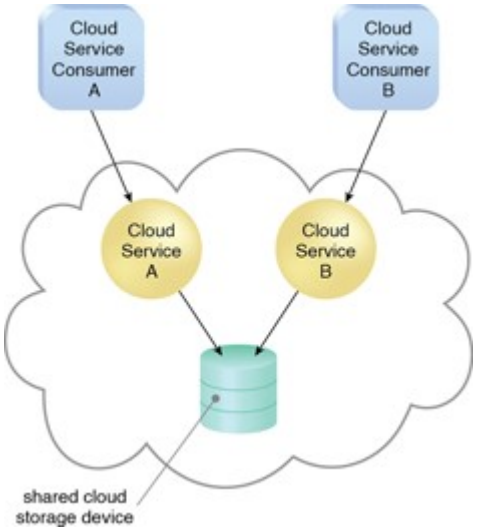


Figure 1. Multitenant resource pooling

3.4 Rapid Elasticity

Rapid elasticity allows cloud consumers to automatically request additional space in the cloud or other types of services [3]. On the other hand, cloud consumer can also rapidly decrease the space if the capabilities are over their usage. Although cloud providers still need to allocate and de-allocate resources, it is irrelevant to cloud consumer. From cloud consumer perspective, the capabilities available for provisioning seem to be unlimited and can be appropriated any time.

3.5 Measured Service

Cloud consumer can keep track of the usage of their IT resources on cloud platforms and pay the service based on the mount of usage. Cloud providers can charge cloud consumers based on what is measured. Cloud systems automatically control and optimize resource use by leveraging the type of service, such as processing, active user accounts, storage, etc. Cloud provider and cloud consumer can monitor, control, and obtain report of cloud services.

4. ACTORS

Five major actors have been described in the cloud computing according to NIST. These are cloud consumer, cloud provider, cloud auditor, cloud broker, and cloud carrier. Figure 2 shows the interactions between the actors in cloud computing.

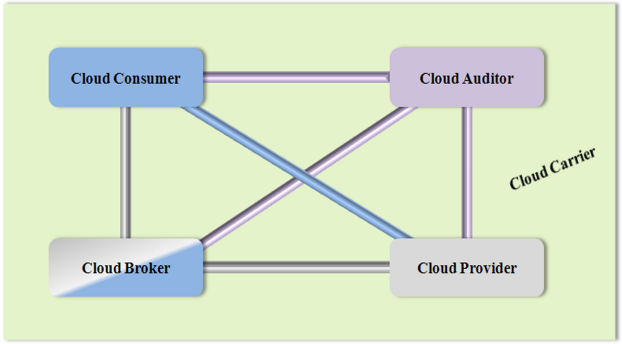


Figure 2. Actors in cloud computing [6]

Cloud consumer: A person or organization that starts and keeps a business association with and requires services from suppliers of cloud services [5].

Cloud provider: A person, organization engaged in supplying cloud computing services to interested persons or organizations.

Cloud auditor: An organization in charge of conducting independent evaluation of cloud computing, and determining the systems effectiveness and security.

Cloud broker: A third-party organization or individual that serves as an intermediary between cloud consumers and cloud providers. He/she is useful for negotiating terms and conditions of the contract for the purchase of cloud services.

Cloud carrier: An intermediary person, organization or entity that provides connectivity and transport of cloud services from cloud provider to cloud consumers.

5. DEPLOYMENT MODELS

Four deployment models are defined according to NIST [2]. These are private cloud, public cloud, community cloud, and hybrid cloud.

5.1 Private cloud

A private cloud is exclusively setup and run for a particular enterprise, but third party organizations can give access to manage them on the behalf of the cloud owner [7]. The private cloud can be operated on-premise or off-premise. Private cloud has privacy, security and control. Private cloud enable an organization to use cloud computing as a means of centralizing access to IT resources by different parts, locations, or departments of the organization. How the organizational and trust boundary are defined and applied may change the use of private cloud. The organization is both cloud consumer and cloud provider with a private cloud. Figure 3 below shows the private cloud owned by a company. Only the company A can access the cloud.

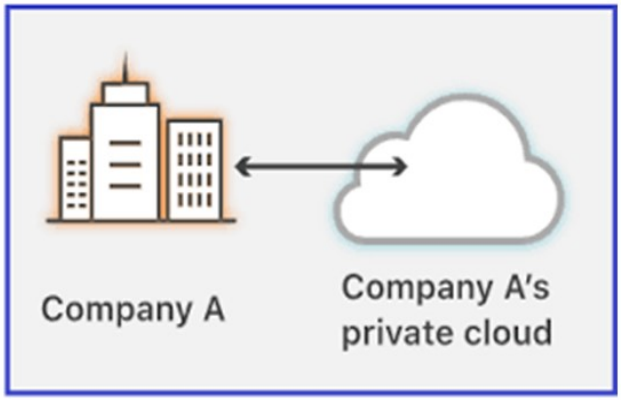


Figure 3. Private cloud

5.2 Public cloud

A public cloud is a publicly accessible cloud environment owned by a third-party cloud provider [1]. The IT resources on public records are provisioned. Consumers pay the services by the measure of their usage. It is cloud provider's responsibilities to create and maintain the public clouds. Services are made available to an organizations and users over a public network through a browser [8]. Public clouds are location independent, reliable and highly scalable, but less secure and not customizable [7]. Figure 4

shows consumers can access public cloud services. The public cloud services can be made by different cloud providers.

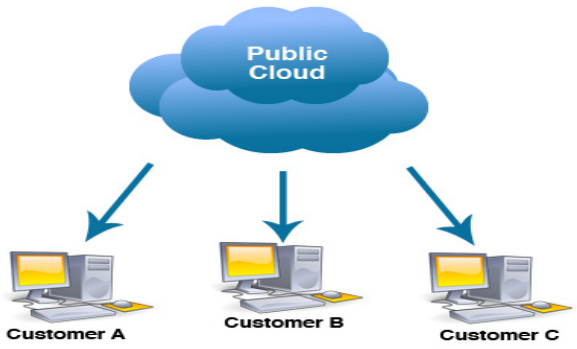


Figure 4. Public cloud

5.3 Community cloud

Community cloud is for exclusive use by a specific community of organizations with shared concerns. It can be owned, managed, or operated either by organizations in the community or by third parties. The community cloud can be operated on or off promise.

5.4 Hybrid cloud

Cloud consumer can choose two or more different cloud deployment models. For example, a company can choose private cloud and public cloud together. Private cloud can be used to deploy cloud services processing sensitive data. For less sensitive data, public cloud can be utilized to provide the service. By combining private cloud and public cloud, flexible and secure resources can be obtained. In addition, cost can be reduced by adapting to the demands that each company needs for space, memory, and system. Figure 5 shows a hybrid cloud deployment models [9].

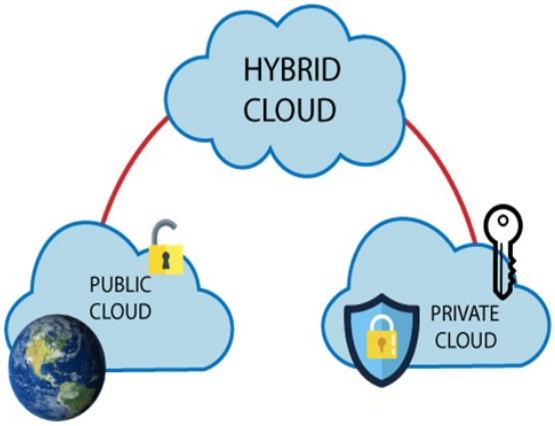


Figure 5. Hybrid cloud

6. SERVICE MODELS

A cloud delivery model represents a specific, pre-packaged combination of IT resources offered by a cloud provider [1]. There are three common cloud delivery models, Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). Figure 6 shows the three service models and responsibilities from vendor and customers.

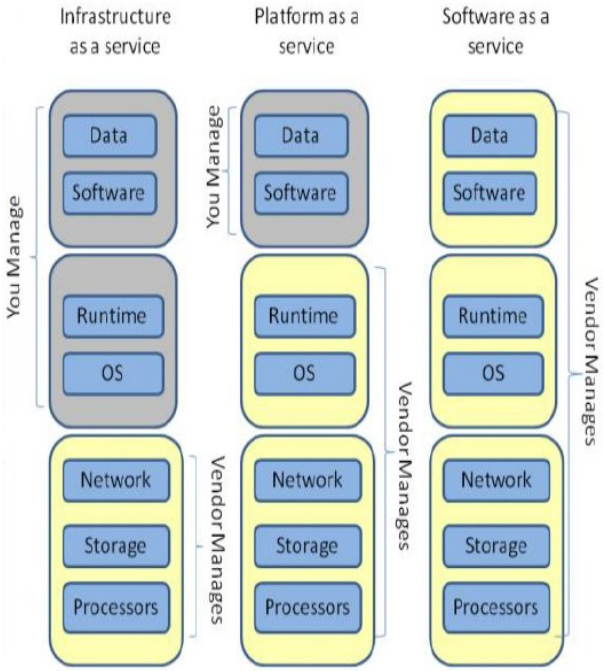


Figure 6. Three service models

6.1 Infrastructure-as-a-Service (IaaS)

The IaaS delivery model represents a self-contained IT environment comprised of infrastructure-centric IT resources that can be accessed and managed via cloud service-based interfaces and tools [1]. Network, storage, and processor are included in this service model. The vendor is responsible to manage the platform, while the cloud consumer is responsible for data, OS, software, etc. The purpose of an IaaS is to provide a high level of control and responsibility over its configuration and utilization to cloud consumers.

Popular examples of IaaS include:

- Microsoft Azure
- Google Compute Engine (GCE)
- Amazon Web Service (AWS)

6.2 Platform-as-a-Service (PaaS)

The PaaS delivery model represents a pre-defined “ready-to-use” environment typically compromised of already deployed and configured IT resources [1]. In IaaS, vendors have more responsibilities than those in PaaS. Runtime, OS, network, storage, and processor are managed by vendors. Cloud consumers are responsible for Data and Software. There are some reasons that cloud consumers use PaaS cloud service. First, cloud consumer could extend on-premise environments into cloud for scalability and economic purpose. Second, cloud consumers could use the ready-made cloud environment to substitute an on-promise environment. Third, cloud consumers can be a cloud provider by provide services based on its own cloud service to other cloud consumers.

Popular examples of PaaS include:

- Google App Engine
- Microsoft Azure
- Amazon AWS Elastic Beanstalk
- Heroku

6.3 Software-as-a-Service (SaaS)

A software program positioned as a shared cloud service and made available as a "product" or generic utility represents the typical profile of a SaaS offering [1]. The provider provides all of the applications running on a cloud infrastructure to the consumers. Cloud consumers could access the cloud service from a various client devices, such as web browser, a program interface. For the IaaS service model, cloud providers are responsible for Data, software, runtime, OS, network, storage, and processor. Cloud consumer has only limited administrative control over the platform. Cloud consumers can reduce cost for companies by eliminating the upfront cost of purchase and installation, and maintenance costs. SaaS also provides pay-as-you-go model, which gives consumer more flexibility.

Popular examples of SaaS include:

- Google Apps
- Netsuit
- Salesforce
- Microsoft office
- Slack

7. FUNDAMENTAL CLOUD ARCHITECTURES

7.1 Workload Distribution Architecture

IT resource can be horizontally scaled via the addition of one or more identical IT resources, and a load balancer that provides runtime logic capable of evenly distributing the workload among the available IT resources [1]. The load balancing algorithms and runtime logic can reduce both IT resource over-utilization and under-utilization. This model can be applied to any IT resource, virtual machine, cloud storage, cloud services, etc. There are some variations of workload distribution architecture, such as service load balancing architecture, load balanced virtual server architecture, and load balanced virtual switch architecture. Figure7 shows a load balance intercepts cloud service consumer requests and directs them to Virtual Service A and Virtual Service B. By doing this, it can guarantee the even workload distribution among two virtual servers. This cloud architecture may include some mechanisms [1]: audit monitor, cloud usage monitor, hypervisor, logical network perimeter, resource cluster, resource replication.

Examples:

- VMware Distributed Resource Scheduler (DRS)
- Amazon AWS Elastic Load Balancing (ELB)
- Google Load Balancer

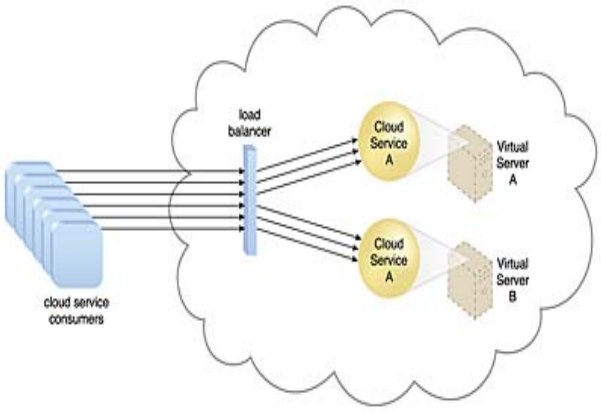


Figure 7. Workload distribution architecture

7.2 Resource pooling Architecture

A resource pooling architecture is based on the use of one or more resource pools, in which identical IT resources are grouped and maintained by a system that automatically ensures that they remain synchronized [1]. By pooling together the resources, the IT resources act as one resource and become more powerful than the individual resources. There are some reasons that using resource pooling. First, the IT resource can be reused. Second, it can increase the percentage of resource utilized. Third, it can reduce costs. The primary audience of this architecture is enterprise IT people and any cloud provider.

There are some types of resource pooling architecture [1]:

- Physical server pool
- Virtual server pool
- Storage pool
- Network pool
- CPU pool
- Memory pool

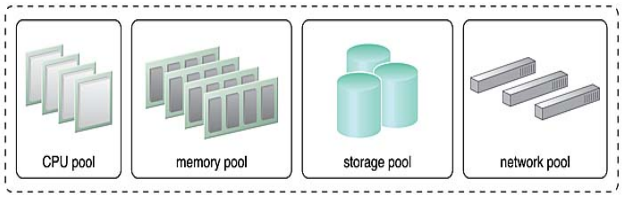


Figure 8. A type of resource pool

Multiple resource pools, each of the resource pool have the same type, can also been group together to form an even larger resource pool. Figure 8[1] shows a resource pool that have 4 different sub-pools, which are CPU pool, memory pool, storage pool, and network pool. The structure of resource pool can be complex, it can be organized based on business requirements, such as separate pool per application, or shared pools for applications. It can also be organized by a hierarchical structure, such as sibling pools, nested pools.

The common mechanisms used in resource pooling architecture are audit monitor, cloud usage monitor, hypervisor, logical network perimeter, pay-per-use monitor, remote administration system, resource management system, and resource replication.

7.3 Dynamic Scalability Architecture

The dynamic scalability architecture is an architectural model based on a system of predefined scaling conditions that trigger the dynamic allocation of IT resources from resource pools [1]. The unnecessary IT resources can be reclaimed automatically under dynamic scalability architecture.

There are usually three types of dynamic scaling:

- Dynamic horizontal scaling
- Dynamic vertical scaling
- Dynamic relocation

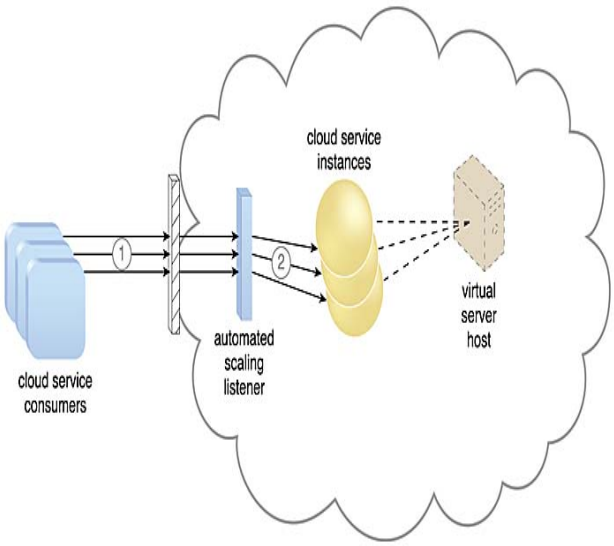


Figure 9. Dynamic scalability architecture

Figure 9 [1] is the first step in the dynamic scalability architecture. The cloud service consumers send requests to a cloud provider. The automated scaling listener will monitor the cloud service. The automated scaling listener will determine if predefined capacity thresholds are being exceeded.

The mechanisms can be used in dynamic scalability architecture can include cloud usage monitor, hypervisor, and pay-per-use monitor.

7.4 Elastic Resource Capacity Architecture

The elastic resource capacity architecture is primarily related to the dynamic provisioning of virtual servers, using a system that allocates and reclaims CPU and RAM in immediate response to the fluctuating processing requirements of hosted IT resources [1]. The VM resources are allocated from resource pool. The virtual machine capacity is specified by range. The virtual machine statistics is captured by "Intelligent Automation Engine" in the elastic resource capacity architecture. The IT resources can be dynamic allocate from resource pools before the predefined capacity thresholds are met.

One example of elastic resource capacity architecture is shown in figure 10 [1]. Cloud consumers send requests to a cloud service. The automated scaling listener is monitoring the cloud service. An intelligent automation engine script is deployed with workflow logic that is capable of notifying the resource pool using allocation requests [1].

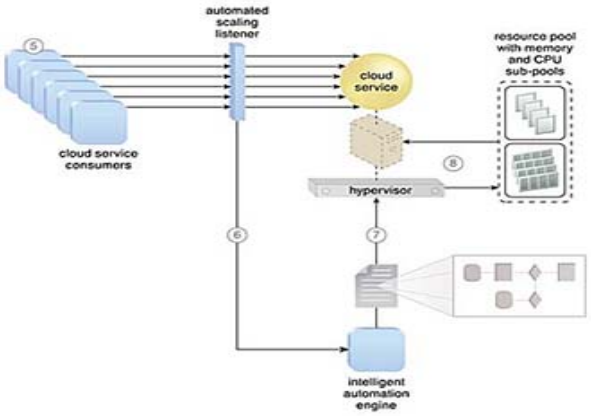


Figure 10. Elastic resource capacity architecture

The mechanism can be used in elastic resource capacity architecture include: cloud usage monitor, pay-per-use monitor, resource replication.

Examples:

- VMware virtual machine
- Microsoft hyper-V virtual machine

8. CONCLUSION

Cloud computing had and will continue to have a big influence on a large number of individuals, communities, and enterprises. It provides scalable, on demand resources anywhere and at any time to cloud consumers. Cloud consumers could choose the cloud services based on their needs and budgets. With cloud computing, organizations could save money on storage, servers, maintenance of the services by moving their services to the cloud. Although there are some risks and challenges existing to cloud computing, cloud computing has the power to continue growing and bring more benefits to consumers to the future.

9. REFERENCES

[1] Thomas Erl, Ricardo Puttini, and Zaigham Mahmood. 2013. Cloud Computing: Concepts, Technology & Architecture (1st. ed.). Prentice Hall Press, USA.

[2] Mell, P. and Grance, T. (2011), The NIST Definition of Cloud Computing, Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, [online],https://doi.org/10.6028/NIST.SP.800-145 (Accessed October 29, 2021)

[3] Tech Opedia Rapid Elasticity https://www.techopedia.com/definition/29526/rapid-elasticity

[4] F. Lui *et al.*, "NIST Cloud Computing Reference Architecture: Recommendations of the National Institute of Standards and Technology," *NIST Spec. Publ. 500-292*, p. 35, 2011.

[5] R. B. Bohn, J. Messina, F. Liu, J. Tong, and J. Mao, "NIST cloud computing reference architecture," in *Proceedings - 2011 IEEE World Congress on Services, SERVICES 2011*, 2011, no. July 2011, pp. 594–596.

[6] Cloud computing actors. NIST. https://www.nist.gov/image/cloudcomputingactorsjpg

[7] Verma and S. Kaushal, "Cloud Computing Security Issues and Challenges: A Survey," no. July, 2011, pp. 445–454.

[8] M. Ali, S. U. Khan, and A. V Vasilakos, "Security in cloud computing: Opportunities and challenges," *Inf. Sci. (Ny).*, vol. 305, pp. 357–383, 2015.

[9] Hybrid cloud. https://www.javatpoint.com/hybrid-cloud

H - Multichip Modules and their performance

Peeyusha Shivayogi
Computer Science Department
San Jose State University
San Jose, CA 95192
408-604-5249
peeyushakavitha.shivayogi@sjsu.edu

ABSTRACT

Multichip Module is an assembly or a package of multiple Integrated chips stacked together which can perform like a single entity when mounted on Printed Circuit Boards (PCBs) just like any other component. However, they possess certain properties that are useful in designing circuits that are useful in today's applications. Therefore, in this paper, we try to provide an overview of MCMs, their classification, a word on their production, performance, their drawbacks and a discussion for improvements.

1. INTRODUCTION

Multichip Module (MCM) also known as 'Hybrid Integrated Circuit' is one such assembly where Semiconductor dies, Integrated Circuits, pins and other discrete components are fused together to work as a larger Integrated Circuit (IC). The need to use MCMs is the failure of Moore's law. That is, after a point, the increase in number of transistors will not increase the performance of the device as the performance reaches a saturation. According to experimental results, the development and usage of MCMs in Graphics processing units (GPUs), high power communication devices, power amplifiers etc., can scale their performance beyond Moore's law. Due to this, there are various kinds of MCMs used based on their applications in Deep learning, high performance computers, Space avionics and many more.

Along with increasing the performance they offer other benefits as well. MCMs reduce the size of the device considerably while making them energy efficient. It offers a packing efficiency of around 30% which is useful in compact designing and in turn provides various benefits such as Flexibility, Reduced Complexity, Better Performance and Compact nature and Energy efficiency.

i. Flexibility: As multiple chips are integrated together it offers flexibility in bringing those components together to mount as one entity.

ii. Reduced Complexity: It is in terms of connecting and packaging the components. In case of MCMs multiple chips are stacked together, we need not be concerned with drawing wires between its constituent components unlike the regular chip designs on PCB. As multiple ICs are fused together, they can perform an entire operation or function as a whole rather than dividing the tasks among many chips and to combine their result.

iii. Compact nature: The spacial capacity required to incorporate multiple chips, to draw a circuit among them to develop a working unit as a whole is more than the spacial capacity required to mount an entity of multiple chips that are fused as one.

iv. Energy efficient: MCMs are energy efficient in the way they are packed densely compared to the regular approach of building them on PCB where they use smaller system footprint.

Understanding the benefits MCMs offer has to do with the understanding of the types they are classified. Because, it is based on their type that they are used at the right place for the right application [1][4].

2. CLASSIFICATION

Three are various ways MCMs are classified. However, one of the common approaches in classifying MCMs is the way their substrate is created. They are:

i. MCM – L
ii. MCM – D
iii. MCM – C

MCM- L are MCMs whose substrates are laminated. In MCM-L high density Printed wiring boards that make up for the laminated substrate are used, as it says the laminates are plastic.

MCM-D are where the substrates are deposited. In MCM-D, the conductor width is less that is around 25picometer and also the spacings between the lines are as less as 75micrometer which helps in high-speed performance. It is a multilayer substrate where metals are deposited as thin films with dielectrics in between the layers.

MCM-Cs have a ceramic substrate. In MCM-C, the substrate is thick, cofired and made of ceramic and is further classified as High temperature cofired ceramic (HTCC) and Low temperature cofired ceramic (LTCC) being its two sub-types.

MCM-Ds are most suitable for aviation, military, space, sonar, signal intelligence systems because of its structure. However, they are expensive compared to the latter. Therefore, there is also an availability of the combinations made from the types mentioned earlier where either MCM-C or MCM-L are used at the top layer in place of MCM-D thin film which could reduce the expense.

3. PACKING

Packing is a way of protecting MCM against damage while a form of design abstraction. The most common packing technique is molding where plastic encapsulation is used. Molding is widely used for most of the components as it is cost effective and easy where the components are encapsulated in plastic films for the purpose of protection and packing. However, they are suitable for single chip components while packing multichip modules there are risks of molding pressures, stress, cracking or distortion of wire bonds in the circuit.

Therefore, a technique called selective encapsulation is used where only some of the selected parts are encapsulated without disturbing those parts at risk of damage while packing, thereby addressing the risks mentioned earlier. A technique called Glob-topping is used where a liquid encapsulant is used. Epoxy encapsulation is a common kind of Glob Topping [2].

4. TESTING

As the use and application of MCMs increase, there is also an increase in burden on testing of these modules. This is because, MCMs are a combination of densely packed components and very large die, where testing and fault detection is hard, time consuming and expensive. But complete testing of the modules is crucial because a failure that could occur will lead to repair or rework which is harder to carry out as we see in the challenges section. Therefore, it is recommended to carry out a full testing at high speed to avoid losses.

Some of the techniques employed for full at-speed testing are:

A. Hierarchical test methodology
B. Design for Test
    a) Boundary scan
    b) Built-in Self-test

4.1 Hierarchical testing

Hierarchical testing is a methodology for testing entire system which can help detect faults at different levels. For example, a component failure is detected in assembly or integration stage while a connection failure is detected at interconnect testing stage, a communication failure between components is tested at Interaction testing stage and so on. As a whole, there are generally four stages involved:

1) Ingredients testing stage
2) Interconnects testing stage
3) Interactions testing stage
4) Interface testing stage

4.2 Design for Test (DFT)

On the other hand, In Design for Test (DFT) the MCMs are divided into partitions physically and considering the partitions to be smaller systems is easier as they can be tested thoroughly. DFT offer Low-cost fault simulation which is useful in testing a lot of hardware and provide scan paths that can be traced back and forth to detect faults. Two of the common DFT methods are:

1) Boundary scan
2) Built-in Self-Test (BIST)

The Boundary scan as defined by IEEE 1149.I, is a DFT method for complete testing of modules where the requirement for the Integrated Circuits is to have latches for boundary scans. In this method, the circuits are connected to form a shift register. Then the data is shifted from one end of the chip to the other followed by shift to the next chip and so on. This will help in shifting data around the chips and work as a scan from one end boundary to another.

And the Built-in Self-Test can be employed at many levels where Integrated chips can test themselves with on-board testers. These testers are used in every chip and a failure at any chip will be reported by their respective testers. However, carrying out tests to identify failure at higher levels than just the components is hard in BIST [3].

5. CHALLENGES

Designing and deploying MCMs has few challenges to address and the most commonly faced challenges are their cost of production, the thermal management and the overhead of repair and rework.

5.1 Cost

Even though we say MCMs are cost effective, there are few challenges if unaddressed can increase the cost instead of reducing. With increase in complexity and density, the yield of production can be low. And adding to this is the use of expensive components in MCMs. Therefore, better designing of the MCMs is important where the right kind of semiconductors are used along with processing smaller substrates from larger panels which altogether increases the yield.

5.2 Thermal management

As the density of the components has increased, the amount of heat dissipated per unit area or for a module has seen an exponential growth. To overcome this issue, certain materials such as Aluminum nitride, CVD deposited diamond, metal composites are used in which

the heat conductance is relatively high in place of plastic or other encapsulates. Using plastic substrate in MCM-L is also a concern here as it is poor conductor of heat.

5.3 Repair and Rework

As we already know fault detection is hard in MCMs, even if you succeed in locating one, the repair or rework is difficult because of the way they are assembled. As there are close spacings it is not possible to insert any instruments or tools to modify a part of the component without damaging the working parts in the surrounding. Therefore, certain techniques of removing the chips with various twisting and tensile forces are used to disassemble the modules first then paying careful attention in resolving the issue in the damaged parts of the circuit. Also, certain thermoplastic adhesives are used because of their property of melting and solidifying at particular known temperatures. This will enable us to modify the required components at required temperatures without disturbing others who have different temperatures. However, care should be taken when the difference in temperature among the components is not huge.

5.4 Known Good Die (KGD)

A Known Good Die is a debated topic and is defined in quite a few ways. The one by MCC (Microelectronics and Computer Technology Corp.) is "Unpackaged ICs that are known at a high confidence level to perform to packaged device specifications and will continue to perform to specification throughout their life." And the one according to Intel, is "equivalent to or better than the corresponding package parts in terms of electrical and reliability performance"

In KGD processing and testing we employ some of the techniques in the four categories being Additive contact, Temporary contact, Pressure contacts and Probe Scrub contact [2][3].

However, these definitions do not talk about the extensive testing a KGD is put through. As we expect a good die, the testing parameters increase drastically and so the test vectors they generate. So, these are hard to handle by tester and the cost of producing a KGD is also significantly higher than a packaged die. Therefore, in most of the cases a failure in die is fixed by replacing a die than relying on KGD. Adding to this, the definition by Intel does not mention about the Burn-in phase failure of dies that is noticed in the Bathtub failure rate curve which we will see in the next section.

5.5 Yield

When we talk about the yield of MCMs we need to consider two sub yields being Assembly Yield and Die yield. The defect rate DR, for a module of N_{bond} wire bonds give us:

$$Y_{assembly} = (1 - DR)^{N}{}_{bond}$$

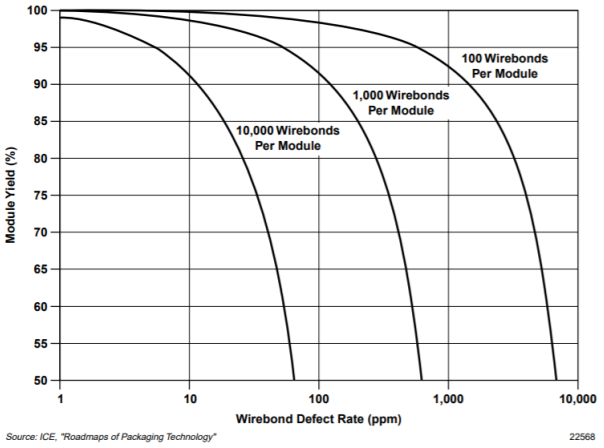


Figure 1. Module Yield in percentage v/s Defect Rate in ppm

As we see for a defect rate of 100ppm the yield is around 89% and for an improved defect rate of 10ppm the yield is around 99% which is achievable with careful assembly and attention to detail during design and production.

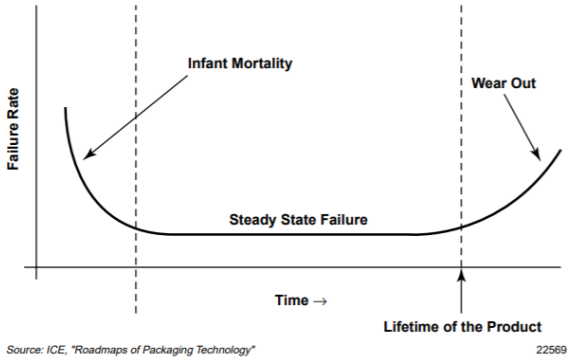


Figure 2. Bathtub Curve showing the lifetime of a product as a function of failure rate

We have to consider the failure rate for chips and it follows the bath tub shaped curve where the initial period in lifetime is Burn-in phase during which those erroneous components that were undetected in testing fail in real world application which is represented by Infant Mortality. This is a real-world testing phase that will cover the possible missed test cases. Followed by the Steady state failure where there is a less margin of failure for a longer time which is the most useful period of the product or termed as the lifetime of the product. After that, we have the Wear Out phase where the products age and the failure rate again rises drastically. Therefore, it is necessary to keep an eye on the produce at the beginning to account for the incoming complaints on infant mortality followed by replacement after the lifetime of components [4].

6. APPLICATIONS

Due to their better performance and compact nature, MCMs are used in devices that need good performance with mobility or compactness such as Wireless devices, Power amplifiers, High power devices, portables, wearables and space applications. Some of the examples of MCMs are IBM Bubble memory MCMs, Intel Pentium Pro, Pentium D Presler, Xeon Dempsey and Clovertown, Sony memory sticks and more. Some of the noteworthy applications include, the Thermal Conduction Modules (TCM) by IBM, switching circuits by AT&T telecommunications, Radars by Hughes Aircraft, Simba MCM for deep learning and more.

When we talk about the applications of MCMs, we need to look at the MCM inspired MCM-GPU. With time Moore's law does not hold good as the performance reaches a saturation. To address this issue, MCMs inspired larger GPUs are designed.

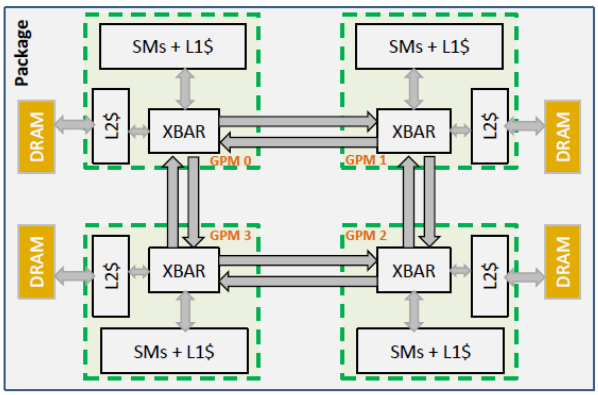


Figure 3. Basic MCM-GPU architecture comprising four GPU modules (GPMs).

The idea is to partition the GPU into smaller GPU modules also called GPMs whose area is expected to shrink by 40-60% compared to a single GPU with an assumption in process node coming down to around 10nanometer and GPMs have multiple SMs and their dedicated L1 caches. This design also enables the L2 cache to cache data only from its local DRAM partition so that there is only one location for each cache line thereby removing the need for cache coherency as shown in the figure 3.
Then integrating these GPMs with the use of high bandwidth and power efficient signaling technologies instead of using a single monolithic die like the conventional design. It is the abstraction of multiple GPUs working as a single GPU. This design has potential benefits such as resource sharing of structures that were underutilized and removing the need for hardware replication among GPUs.

Table 1. Baseline MCU-GPU Configuration

Number of GPMs	4
Total number of SMs.	256
GPU frequency	1GHz
Max number of warps	64 per SM
Warp scheduler	Greedy then Round Robin
L1 data cache	128 KB per SM, 128B lines, 4 ways
Total L2 cache	16MB, 128B lines, 16 ways
Inter-GPM interconnect	768GB/s per link, Ring, 32 cycles/hop
Total DRAM bandwidth	3 TB/s
DRAM latency	100ns

Along with the architecture proposed, the authors build three mechanisms to minimize inter-GPM bandwidth that is a GPM-side hardware cache, distributed CTA scheduling and data partitioning and locality aware page placement. These mechanisms improved the performance drastically which is shown by the s-curve of the speedup v/s workloads of around 50 workloads and it depicts, around 30 workloads perform better with improvised MCM-GPU except for the few in the initial stage.

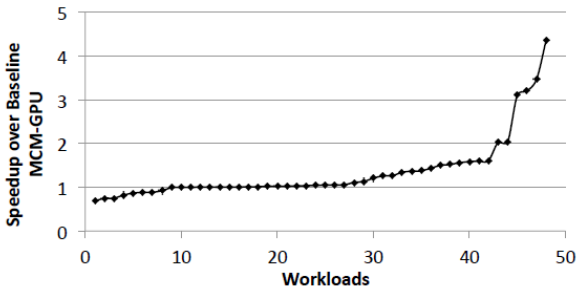


Figure 4. S-curve summarizing the optimized MCM-GPU performance speedups for all workloads.

Another highly used approach to improve the performance of GPU is to build a multi-GPU system. But, the concerns like workload distribution, load imbalance, interconnection bandwidth and data placement as a result of lower inter-GPU bandwidth play a major role in bringing their feasibility down. However, an improvised design of multi-GPU system outperforms the regular multi-GPU design as well as the Monolithic GPU by a factor of 25% while the proposed improved design of MCM-GPU outperforms the rest by a factor of 50% [5].

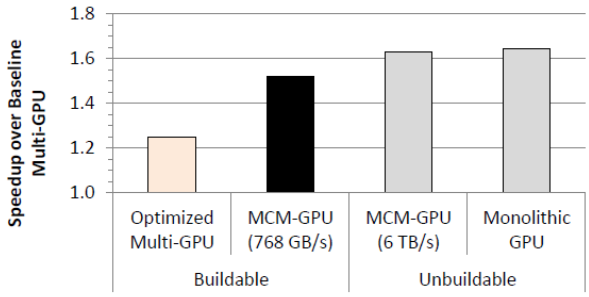


Figure 5. Performance comparison of Multi-GPU and MCM-GPU

MCMs are used in Deep learning applications as they require high computation and relying on monolithic dies is hard and not scalable with increase in performance requirement. In [6] the authors architected and tested SIMBA which is a 36-chiplet MCM prototype for deep learning interface.
In [7] the authors propose the idea of first-generation field programmable Multichip Module (FPMCM) and the idea of using them as a test vehicle for a MCM technology.

7. CONCLUSION

Multichip modules (MCMs) are an efficient way of designing circuits for better performance with energy efficiency and compactness when designed and deployed carefully which can offer wide range of applications in high performance computing, space applications, military, deep learning and so on. But, in practicality they have certain drawbacks that come into light in designing and deployment. However, these concerns can be addressed with improvement in better testing mechanisms and optimizing module designs.

8. REFERENCES

[1] MultichipModule,https://www.techopedia.com/definition/11836/multi-chip-module-mcm

[2] James J. Licari, Leonard R. Enlow, Hybrid Microcircuit Technology Handbook (Second Edition), William Andrew Publishing, 1998, Pages 526-564, ISBN 9780815514237, https://doi.org/10.1016/B978-081551423-7.50015-8.

[3] James J. Licari, Dale W. Swanson, In Materials and Processes for Electronic Applications,
Adhesives Technology for Electronic Applications (Second Edition),
William Andrew Publishing, 2011, Pages 1-34, ISBN 9781437778892,
https://doi.org/10.1016/B978-1-4377-7889-2.10001-4.

[4] Integrated Circuit Engineering Corporation, Multi module chips(mcms), Smithsonian, The chip collection, http://smithsonianchips.si.edu/ice/cd/PKG_BK/CHAPT_12.PDF

[5] Akhil Arunkumar, Evgeny Bolotin, Benjamin Cho, Ugljesa Milic, Eiman Ebrahimi, Oreste Villa, Aamer Jaleel, Carole-Jean Wu, and David Nellans. 2017. MCM-GPU: Multi-Chip-Module GPUs for Continued Performance Scalability. In Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17). Association for Computing Machinery, New York, NY, USA, 320–332. DOI: https://doi.org/10.1145/3079856.3080231

[6] Yakun Sophia Shao, Jason Clemons, Rangharajan Venkatesan, Brian Zimmer, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, Nathaniel Pinckney, Priyanka Raina, Stephen G. Tell, Yanqing Zhang, William J. Dally, Joel Emer, C. Thomas Gray, Brucek Khailany, and Stephen W. Keckler. 2019. Simba: Scaling Deep-Learning Inference with Multi-Chip-Module-Based Architecture. In Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO '52). Association for Computing Machinery, New York, NY, USA, 14–27. DOI: https://doi.org/10.1145/3352460.3358302

[7] J. Darnauer, P. Garay, T. Isshiki, J. Ramirez and W. Wei-Ming Dai, "A field programmable multi-chip module (FPMCM)," *Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines*, 1994, pp. 1-10, doi: 10.1109/FPGA.1994.315592.

I - INSIGHTHS INTO APPLE'S SILICON CHIPS

Bhavya Reddy Kotla
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000
bhavyareddy.kotla@sjsu.edu

ABSTRACT

Apple CEO, Tim Cook has kept reinstating the fact of having a long-term strategy of owning and controlling the primary technologies behind the products they make. Proving his visions, Apple began designing its A-series chips for phones, watches, and iPads back in 2010. And to drive the point home, in 2020, they introduced M1 chips that replaced Intel processors for their laptops and desktops. In this term paper, we look at how both these chips, M1and A14 Bionic vary, their architectures along with their features and pitfalls. We also try to look at how Apple is trying to make the transition smoother, through Rosetta, an emulator that is built into macOS that will enable ARM Macs to run old Intel apps. We also try to view Apple's advancements with respect to other leading companies in the industry, specifically, Intel processors and Qualcomm's Snapdragon 888 chips.

1. INTRODUCTION

Back in June, Apple announced to move away from Intel and use processors built by itself using the same ARM architecture it had long been using on phones and tablets. The longest-lived CPU partnership in Mac history came to an end and Apple had confirmed, not only the development of their ARM-based chips in the house but also their intention to complete this migration over the next two years. After this announcement, the Macs and even the PCs can potentially never be the same. So, how did Apple get here? What triggered this transition?

Apple had long been rumored to be working on its processors to gain the same top-down control it enjoys with its mobile devices, iPhones, and iPads. The beginning of this was with the acquisition of PA Semi in 2008 for $278million [1]. This company then laid the foundations for the A series of chips, the power grids for every iPhone and iPad since 2010. Since the development of the A4 SoC (System-on-a-chip) in 2008, Apple has been consistently designing their silicon in-house, owing to advanced RISC machines (ARMs) generous licensing model. This allowed third parties to implement their instruction set in standard ways but without limitations in terms of core layouts [18]. Owing to this flexibility, Apple got the ability to control the entire design of the iPhone and since the iPhone 4s, they've consistently defended their fastest smartphone crown [18]. Building on this, Apple further went to acquire more semiconductor talent with Intrinity in 2010 ($121million), part of Dialog semiconductor in 2018($600million), and the final destination, sort of with a part of Intel's modem business in 2019($1billion).

All these acquisitions over the year pointed to Apple's growing in-house silicon market. But, the transition or extension to Macs was prompted with growing criticism of Intel's inability to perform a die shrink (i.e., reducing the size of elements on the chip without changing the fundamental circuit design) after being stuck on 14-nanometer nodes for years before moving to slower 10nm chips. Things started going downhill when Tim Cook, Apple's CEO blamed the Intel chip shortages for declining Mac sales in 2019. Thus, with 12 years of experience in designing CPUs and having achieved a claimed 100x improvements since they started, Apple thought, why they had to buy CPUs that turn out to be subpar from Intel or AMD? They went with the fact that if they want it done right, then they better to do it themselves. Thus, a year later, in 2020, Apple revealed the MacBook Air and MacBook Pro with M1 chips or Apple's custom ARM-based silicon. The debut of Apple's first computer chips marked the beginning of the end of a 14-year collaboration between Apple and Intel [2].

Before looking at M1 chips that triggered this conversation, let's first look at A14 Bionic, the latest one in the A series which started to get the best of all worlds: performance, energy, efficiency, and compatibility for small devices.

2. A14 BIONIC

The A14 Bionic is a mobile microprocessor designed by Apple, the latest descendant of a long series of A-series processors by Apple, that was first introduced as the A4, and debuted in the first iPad [3]. Currently, the A14 Bionic is featured in the iPhone 12 lineup and the iPad Air 4. The key features of the A14 bionic as seen in Figure 1 are as follows:

a) The chip is designed on a 5nm process with 6 CPU cores in two core configurations: two high-performance Firestorm cores, and four high-efficiency Icestorm cores.
b) Constructed with 11.8 billion transistors, an upgrade from A13's 8.5 billion.
c) Four-core GPU that is 50% faster than any nearest competitor. The main focus of the GPU is on efficiency i.e., delivering most of the tasks, most of the time at the lowest voltage and frequency but still being ready to ramp up or spike when and if needed [4]. This would lead to preserving as much battery life as possible whilst providing sustained peak performance.
d) Separate Neural Engine core with 16 neural processors. The neural engine was introduced in 2017, as part of A11 with the Face ID being the showcase feature [4]. The current Neural Engine is twice as fast as the ones

found both on A12 and A13 performing up to 11 trillion operations per second.

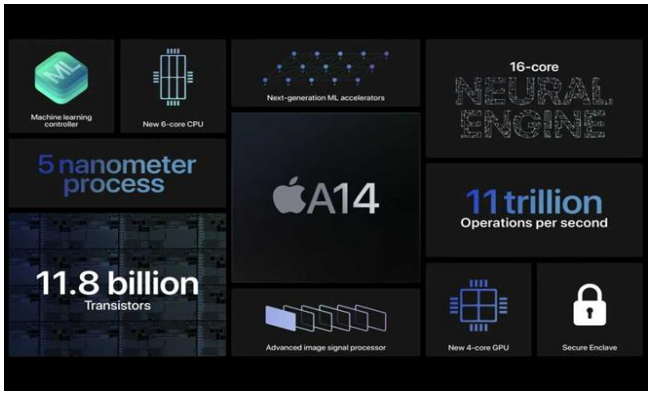


Figure 1: A14 Bionic Features

In parallel with these key features, A14 has also been optimizing the entire System-on-chip for Machine Learning including CPU and GPU with custom accelerators called AMX blocks for accelerated matrix multiplication, something that is frequently used in Machine Learning [4]. Along with this, it also has included a new signal processor to record videos in HDR and for capturing photos in ProRAW format and night mode; a new I/O core for USB3 connection and 5G connection via Qualcomm X55 4G/5G modem.

2.1 Architecture

The exact architecture of the A14 chip is not revealed by Apple. But, a few of the nuances that make the chip perform the way it does in terms of efficiency, performance as seen in Figure 2 are as follows. The chip is designed to handle videos in such a way that it redirects it away from the Intel chips in current Macs and towards the T2 chips- variants of the A10 [3]. Also things like the performance controller, the secret component which helps to figure out what to direct to the CPU, GPU, Neural Engine and other components, to get the best performance at the highest efficiency for any task; a new ML controller for doing the same between the Neural engine, the AMX and the main cores; the custom storage controller for making sure the solid-state chips aren't just performing as fast as possible, but are making sure that every photo, every frame of the video gets right and properly saved [3].

Figure 2: Architecture or IP blocks of the A14 Bionic

All these key features and various blocks of the chip may not seem as flashy as many of the other things that we have come to expect of Apple. But, it is the kind of thing that helps real people avoid real problems and in having a better experience every day [3]. This is why, when we get numbers like 20% more performance and 40% greater efficiency, it is not the result of a process shrink or a massive global alteration. It would then be the result of touching all these corners, working on all these features, shaving off half a point- making every tiny advance to push the overall performance and efficiency forward [3].
Next, let us look at the M1 chip, the component that triggered the current discussion.

3. APPLE M1

Culminating more than a decade of Apple's work on chips, M1 is the first Apple-designed SoC developed for the design in Mac, marking the transition from Intel chips that had been in use since 2006. As an SoC, like the name indicates M1 integrates several different components including the CPU, GPU, RAM, Neural engine, secure Enclave, SSD Controller, image signal processor, encode decode engines, Thunderbolt controller with USB4 support, and more, all of which power different features in Mac [5]. There were multiple chips for CPU, I/O, security but Apple's effort to integrate these chips is why M1 is much faster and more efficient than prior Intel chips [5]. The key features of the M1 chip as seen in Figure 3 are as follows:

a) Eight core CPU featuring four high-performance cores, Firestorm and four high-efficiency cores, Icestorm offering up to 3.5x increase in CPU performance.
b) Eight or seven core GPU amounting to 6x faster graphic performance and delivering up to 2.6 TFLOPS. Comparing this with a desktop graphics card that draws 75W of power with 3.3 billion transistors delivering 2.1 TFLOPS, an integrated graphics card on a passively cooled MacBook Air beats it.
c) Sixteen core Neural Engine capable of 11 trillion operations per second and up to 15x faster CPU performance compared to the previous generation of models.
d) Integration of more components than an Intel CPU.
e) Integration of RAM in the same package i.e., unified memory architecture allowing the CPU, GPU, and other process components to access the same data without having to swap between multiple pools of memory.
f) Rosetta 2 dynamic binary translation allowing the running of x86 software.
g) Five-nanometer process with 16 billion transistors.

Figure 3: Apple M1's key features

In addition to these, Apple M1 also features a new image signal processor, Secure Enclave, Rosetta hardware, as well as a dedicated encode and decode engine for audio and video content.

3.1 Architecture

Before looking at the Apple M1's architecture, let us first understand what ARM-based architecture means. Advanced RISC machines (ARM) are designed to perform a smaller number of computer instructions to operate at higher speeds and thereby, perform millions of instructions per second. Load/store architecture, orthogonal instruction set, and single-cycle execution are ARM-based architecture key features [7]. ARM uses hundreds of smaller, less sophisticated, low-power processors that share processing tasks among that large number instead of just a few higher-capacity processors [8]. This approach is referred to as scaling out, in contrast with scaling up which is done in the case of x86-based servers [8].

Keeping the above in mind, the Apple M1 chip features four big Firestorm CPU cores for high-load scenarios, backed by four smaller Icestorm CPU cores for efficiency. This architecture is similar to Android phones with ARM CPU layout. ARM calls this layout ARM big. LITTLE as seen in Figure 4 and it has been around since 2014 [8]. The M1 CPU uses the ARM64 extension set of the ARM architecture.

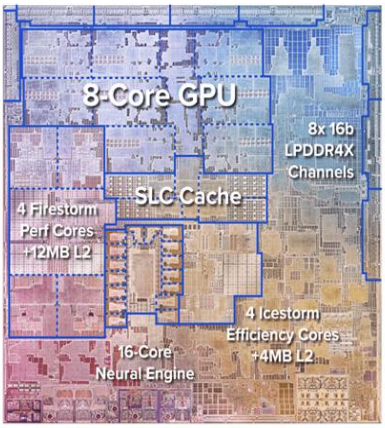


Figure 4: big. Little architecture of Apple M1

The cores of the CPU offer work alone when significant power isn't needed but for demanding tasks, all eight cores can be engaged at once [9]. For less intensive tasks like web browsing that doesn't require the same power, the four high-efficiency cores use 1/10th of the power to preserve battery life [9]. The GPU on the other hand is capable of running 25,000 threads simultaneously and according to Apple, has the fastest integrated graphics in a PC [9]. Let us now look at what all these additional features brought to the table with regards to improvements and pitfalls.

3.2 Advantages

a) Improved Battery Life: Apple claims that the M1-powered Mac is more efficient than any other Mac released; it lasts up to 2x longer than prior generation Macs. And due to improved efficiency, the MacBook could run a few hours longer with no changes to battery capacity [8].

b) Efficiency and Speed: The M1 promises a lot of performance without much noise or heat [8]. It claims to bring up to 3.5x faster CPU performance, 6x faster GPU performance, 15x faster ML capabilities compared to Intel.

c) Vertical Integration: For the first time in almost two decades, Mac users will be using processors superior to x86 chips powering Windows PCs [10]. Currently, Apple has its silicon to back its operating system. This isn't just a matter of pride or boasting- this is vertical integration at a level previously unseen in the industry. Apple now controls its OS as well as its CPU design [10]. It relies on third parties only for manufacturing and commoditized components such as storage, displays, touchpads, and so on [10].

3.3 Pitfalls:

a) Lack of user-upgradeable RAM: Since the M1 integrates RAM in the SoC package for unified memory architecture, it has the downside of being available only with 8GB and 16GB of RAM. And since we are used to PCs with soldered RAM, with RAM integrated into the SoC in M1, swapping out the RAM chips with higher capacity ones, would require doubling the memory capacity in the chip package i.e., a revised M1 chip [8].

b) Transition period performance issues: ARM-based Apple M1 is supposed to run legacy x86 software using Rosetta, which could adverse effects on performance when running x86 applications [10]. Also, initially, some teething problems are to be expected, as exotic tools and applications might not run out of the box, or they may incur a performance penalty [10]. These issues are to be expected on new hardware, as developers need time to catch up to ensure compatibility and port their software for the new hardware.

c) Optimization and Compatibility issues: Some applications need to be optimized for the new processor to ensure support for M1 processors and enable them to utilize their full performance potential. It will be essential to distinguish between software that is not optimized for Apple M1 and software that currently cannot run on Rosetta 2 [8]. Lack of optimization will result in degraded performance, while lack of compatibility will result in unworkable projects [8].

d) Prevalence of x86 servers: Although the ARM processors made inroads in certain niches of the server market, most servers still use x86 chips [8]. For years, Macs were the go-to platform for software developers because they allowed them to work on a UNIX-based operating system running on x86 hardware [8]. They would produce code designed to run on servers using the same instruction set and another UNIX-based operating system. With the M1, this will change as

Apple developers will be developing software on ARM hardware and then rolling it out on x86 servers [8].

Supporting the journey to Apple Silicon, Apple's decision to migrate is history in action. The company now controls the future of all of its platforms and its processors are already impressing users with their performance and stability. That said, applications are built to run on specific processors, and not every developer has done all the work necessary to make their apps run natively on Apple Silicon Macs [11]. So, what about these applications?

To solve this issue, until the transition is complete, Apple has employed the Rosetta 2 translation technology. The next section discusses Rosetta 2 and its operation in detail.

4. ROSETTA 2

Rosetta 2 is a behind-the-scenes feature in macOS as part of Apple's shift away from Intel-based Mac designs. It's used to translate Intel-based Mac apps so they can run on Apple Silicon Macs without having to modify the source code [12]. This is an improvement on Rosetta, the translation technology that allowed PowerPC apps to run on Intel in 2006. More specifically, Rosetta 2 translates x86 processor instructions (aka 64-bit Intel) for ARM-based Apple Silicon, which debuted on Macs with the M1 chip. It's not meant to substitute native apps but to give developers time to create a "universal binary" for apps already offered on Mac [12]. The key features of Rosetta 2 are as follows:

i) Fast performance
ii) Translation at install time
iii) Dynamic Translation for JITs (Just in time)
iv) User Transparency

Rosetta 2 works with Intel-based apps distributed and installed from external sources. The big difference with Rosetta 2 compared to the original version is that it automatically translates non-native apps when they're installed, not during runtime [12]. This increases the app's overall performance because there is no additional processing overhead. However, it does translate code on the fly as needed, like just-in-time (JIT) JavaScript compilers for web browsers [12].

According to Apple's documentation, when Rosetta encountered a call to a routine that it had not yet translated, it translated the needed routine and continued the execution. This process ensured "smooth and continual transitioning between translation and execution" [12]. It also optimized the translated code to provide a near-native experience on non-native hardware.

While rolling out any new hardware or software, as we learned in class, there is a set of Design optimization factors to be considered. The next section will discuss the same with regards to the factors discussed in class and how Apple's choice of launching a new chip, M1 is following these factors.

5. DESIGN OPTIMIZATION

Performance or success is always considered in terms of speed or efficiency. But there are other factors that determine how well the invention or improvisation worked. Let us now look at the various factors that have to be considered when new hardware or software is rolled out and how well Apple performed with all the following criteria:

5.1 Machine/Cost performance optimization

With Apple's move to M1 chips, savings of $2.5Billion is estimated this year, with far larger savings impending as Apple transitions its entire Mac lineup to its silicon [13]. With this switch to Apple silicon across the Mac range, the financial savings will be in billions- whilst achieving best of all worlds- vastly improved performance, greatly increased battery life, and multi-billion-dollar savings [13].

5.2 Global optimization of total cost of both Hardware and Software

While introducing a new M1 chip into the existing Intel chip, the issue of running already published apps on a Mac with a completely different CPU design arises i.e., global environment for all applications is paramount. For instance, an app designed for Macs with Intel processors can't natively run on macOS modified for an ARM-based chip. Something behind the scenes must "translate" the app.

To resolve this, Apple used the concept of cross-platform virtualization, through a technology that "translated" binaries designed for one CPU architecture to run on a different operating system or processor [10]. It remaps all operating-system calls given the code differences between one processor design and another. This is Transitive Corporation's Quick Transit technology, the one behind Rosetta 2 [10].

5.3 Backward Compatibility

The marketplace strongly desires backwards compatibility. The requirement of older applications to still work on new OS is crucial. Changing the architecture often outdates the existing application software. This shouldn't be the case. Whenever we change the architecture, the cost to rewrite application software for different architectures is enormous. Thus, from a market standpoint, compatibility with the existing software is imperative. This is why Apple introduced Rosetta 2, to achieve compatibility for Intel-based applications on M1 during the transition period.

5.4 Design Complexity

The key to Apple's success lies in the M1's incredibly "wide" design. This refers to how many instructions the chip can process each cycle. While the M1 is single-threaded, meaning each core can only process one stream of instructions at once (unlike Intel and AMD's multi-threaded design), it can process as many as 8 instructions per cycle [14]. That's nearly twice as many as most modern designs. This lets it still do a ton of work, even when running at a lower clock speed, which can help save power. This is just one of a series of smart, simple design choices Apple made that added up to an impressive processor [14].

Consideration of all these factors while releasing the new M1 chip, made Apple mint profits right from the first fiscal year after the release. Mac sales for the quarter ended Dec. 26, 2020, came to $8.68 billion, up from $7.16 billion in the same period a year ago [15]. The iPhone sales revenue for the quarter came to $65.60 billion, up from $55.96 billion in the year-ago period, driven by "strong demand" for Apple's new 5G iPhones, Cook

said. Apple in October 2020 launched its first slate of 5G-capable iPhones — the iPhone 12, iPhone 12 mini, iPhone 12 Pro, and iPhone 12 Pro Max — powered by a new, faster A14 Bionic chip [15]. Turning to financials, the company's total net sales for the quarter ended Dec. 26, 2020, jumped 21% year over year to $111.4 billion, up from $91.82 billion a year earlier [15]. Figure 5 depicts the growth in Apple's Financials during Q4 Fiscal of 2020.

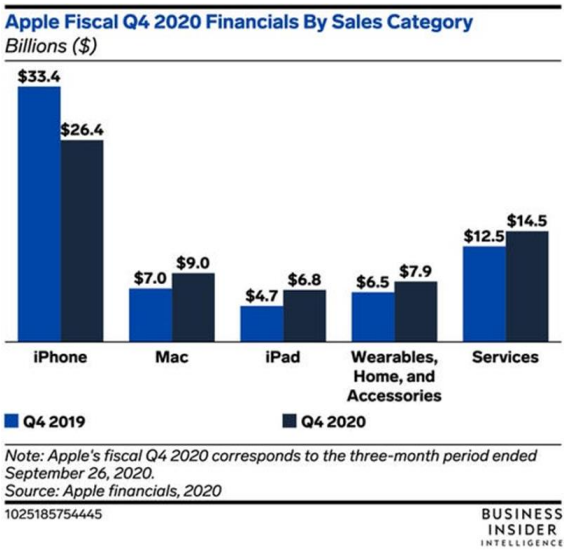


Figure 5: Apple's Q4 Fiscal 2020 Financials

After looking at the history, architecture, the design optimization factors, let us now understand where the other competitors of the market, in our instance, Intel and Qualcomm stand in comparison with Apple's M1. We primarily concentrate on Intel, the one for which M1 was released to replace.

6. APPLE vs INTEL

In addition to the fundamental differences that Intel and Apple have with Intel employing a x86 architecture to Apple using an Arm based architecture, we now consider, how these differences in architecture and hardware impact the overall features and working of a processor in which they are inbuilt.

6.1 Performance

The most important element of a processor for many customers is performance. And for the longest time (the better part of a decade), Intel was unrivaled when it came to mobile-chip performance with its "Core" processors running circles around the competition [16]. But, currently the Apple Silicon powering the latest MacBook Pro and MacBook Air put up jaw-dropping results in our synthetic benchmark tests — even with software running through a Rosetta 2 translation layer [16]. The M1 chip has only gotten faster as developers optimize their software to run natively on ARM.

This doesn't mean the MacBook Pro and Air will be faster than the quickest Windows 10 PCs at every application or tool. Before reading too much into these tests, understanding the fact that these tests were conducted internally with third-party benchmarks is critical [16]. But in general, the M1 chip seemed to match or outperform the rival Intel chips.

6.2 Battery Life

Based on battery tests, Apple's latest laptops are among the longest-lasting on the market with the MacBook Pro surviving for an outstanding 16 hours and 32 minutes [16]. That being said, there are a handful of Intel-powered laptops capable of similar or even lengthier runtimes [16].Thus, in regards to battery life, both Apple and Intel seemed to have the same capacity.

6.3 Hardware Selection

With Apple, the Macs are premium, attractive systems but, they lack some of the features found on PCs. Most notably, we don't find MacBooks with 2-in-1 form factors because Apple has publicly voiced its disdain for convertible laptops and touchscreens [16]. Whereas, notebooks like the Spectre, Yoga 9i, can convert from a laptop into a tablet and the ThinkPad to be detachable if and when required [16].

Additionally, not only do Intel systems span various categories, but the customer can pick and choose which laptop to buy based on the features and specs needed. As much as the MacBook Air and MacBook Pro try to be the best premium laptops for the widest group of users, they have a few potentially deal-breaking limitations [16]. Those include an unfortunate lack of ports with both models relying on Thunderbolt ports and a headphone jack. Windows 10 systems, some of which are even thinner and lighter than the Apple laptops, come with USB Type-A inputs, HDMI ports, microSD card slots [16].

6.4 Security

The built-in T2 chip handles security in Intel while in Macs with M1, this functionality is built right in, without the need of a secondary chip [9]. The built-in secure Enclave for managing Touch ID and a storage controller with AES encryption hardware makes the performance faster and more secure compared to that of Intel [9].

According to the claims made by Apple, the graphical representation for CPU and GPU power in comparison with Intel are seen in Figure 6 and Figure 7.

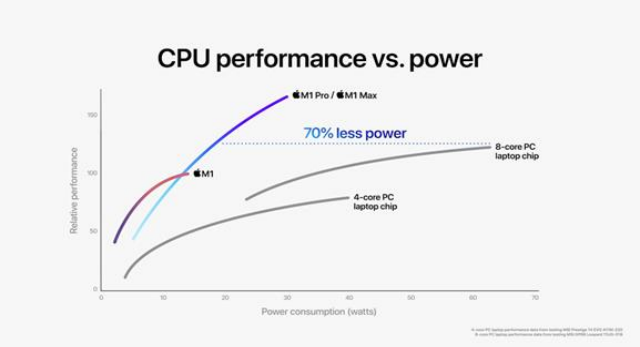


Figure 6: CPU performance vs power for M1 and Intel chips

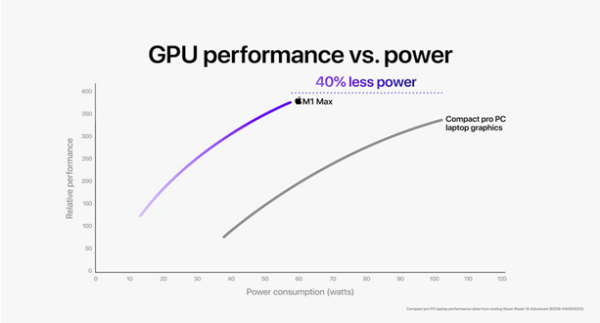


Figure 7: GPU performance vs power for M1 Max and PC laptop graphics

7. APPLE vs QUALCOMM

Qualcomm's Snapdragon 888, like the Apple M1, is made on the 5-nanometer process. It has 8 cores in three different configurations: the single Prime core featuring very high performance, three Gold cores for high performance, and four Silver cores for high efficiency [17]. The Apple A14 was able to beat out the Qualcomm SnapDragon 888 in most tests despite having less compute and graphic cores, less memory in most of the phones tested [17]. Even though the SnapDragon 888 had an edge over the A14 in raw performance benchmark, however, it did better in some syntactic benchmarks like GeekBench 5 and gaming benchmark.

The respective scores as compared syntactically for both are seen in Figure 8.

Snapdragon 888 +16%		720062
A14 Bionic		618736
CPU	193435	173864
GPU	295493	208037
Memory	111279	106696
UX	97756	93575
Total score	720062	618736

Figure 8: Comparison of Qualcomm's SnapDragon 888 with Apple's A14 Bionic

8. FUTURE ENHANCEMENTS

Currently, the transition period can be painful. Going from PowerPC to Intel might have been much more complex compared to the transition from Intel to M1 owing to technological advancements. But, this doesn't mean that Apple wouldn't be fraught with compatibility and performance implications early on. Along with this, there are still a lot of things to be addressed and in predicting how this thing is going to play out [18]. What if Intel gets better? Will PCs completely overtake them like what happened during PowerPC? Or, what if the opposite happens? [18]

Considering these impending questions, according to me, the following future enhancements should be considered to make the M1 chip a success for desktops:

a) Either Apple's ARM cores need to get competitive with x86 cores in single-threaded tasks, or they need to augment them [18]. Desktops will improve, and based on the direction Apple is heading in, it might be taking the latter path. By giving themselves time, for ensuring developers take advantage of many-core CPU designs and of heterogenous to compute, which is, using a combo of CPU and GPU or even neural processors to do conventional CPU work, faster would be critical [18].

b) ARM given that it's a RISC architecture that can do more work in fewer clock cycles and with less power than a CISC architecture, as x86 can, this was an advantage that Apple used to enjoy with early PowerPC Macs outperforming their intel counterparts at much lower clock speeds [18]. But, there are some drawbacks to this approach. While RISC is more streamlined, with no microcode, it lacks flexibility meaning that if applications use an x86 instruction for which there is no ARM equivalent, developers would be looking at either having to work around it manually or simulate it with many more cycles than needed [18]. Thus, Apple ensuring that third-party apps like Adobe, Docker accustom and adjust to these changes will be crucial.

c) Apple should be clear about who it is putting first. Promising custom hardware solutions tailor-made to what the customers need might be worth some monopolistic behavior to some people [18]. But there's one small problem, does Apple even know what their customers want to do with their computers? The last few years have been spotty at best when it comes to listening to both customer and developer feedback [18]. Thus, keeping both the hardware and software as flexible as possible, with priority being given to customer requirements will be of paramount importance.

9. CONCLUSION

Apple, in conclusion, has an advantage since it doesn't have a definitive profit or loss on selling chips, they don't have to balance the competing demands of multiple buyers. With this edge, they can work multiple years out with other teams inside Apple, from Hardware to Software engineering to Human Interface Design. Thus, their goal of creating a silicon architecture, a vertical integration, that scales from mobile devices to personal computers is something that has not happened before but, it is in the making and even has potential for being a success. Hence, while we are watching history being made, whether Apple succeeds or fails, one thing is evident, that the computer industry will never be the same with regards to the transition that Apple has bought it in the silicon industry.

10. REFERENCES

[1] Dan Frommer, Apple Buys Chipmaker P.A. Semi For $278 Million. https://www.businessinsider.com/2008/4/apple-buys-chipmaker-p-a-semi-for-278-million

[2] CNBC, Apple is breaking a 15-year partnership with Intel on its Macs — here's why. https://www.cnbc.com/2020/11/10/why-apple-is-breaking-a-15-year-partnership-with-intel-on-its-macs-.html

[3] iMore, Apple A14 Bionic Explained — From iPad Air to iPhone 12. https://www.imore.com/apple-a14-bionic-explained-ipad-air-iphone-12

[4] MacWorld, A14 Bionic FAQ: What you need to know about Apple's 5nm processor. https://www.macworld.com/article/234595/a14-bionic-faq-performance-features-cpu-gpu-neural-engine.html

[5] SVA NYC Campus Store, What is the big deal about Apple's new M1 chip? https://svacampusstore.com/blogs/news/whats-the-big-deal-about-apples-new-m1-chip

[6] Apple Newsroom, Introducing M1 Pro and M1 Max: the most powerful chips Apple has ever bui, https://www.apple.com/newsroom/2021/10/introducing-m1-pro-and-m1-max-the-most-powerful-chips-apple-has-ever-built/

[7] Wikipedia, ARM Architecture, https://en.wikipedia.org/wiki/ARM_architecture

[8] Developers, Apple M1 processor overview and compatibility https://www.toptal.com/apple/apple-m1-processor-compatibility-overview

[9] Juli Clover, Apple M1 chip: Everything you need to know https://www.macrumors.com/guide/m1/

[10] What makes Apple M1 processor different? http://ggregi.com/articles/Apple-M1.docx

[11] Jonny Evans, Everything you need to know about Rosetta 2 on Apple Silicon Macs https://www.computerworld.com/article/3597949/everything-you-need-to-know-about-rosetta-2-on-apple-silicon-macs.html

[12] Apple Insider, Rosetta 2 https://appleinsider.com/inside/rosetta-2

[13] 9to5 Mac, Apple's move to M1 chips will save $2.5B this year, estimates IBM exec

[14] Engadget, Apple's M1 isn't witchcraft, it's good chip design https://www.engadget.com/apple-m1-architecture-upscaled-141515465.html

[15] S&P Global Market Intelligence, Apple's inaugural M1 chip puts company on 'new growth trajectory' https://www.spglobal.com/marketintelligence/en/news-insights/latest-news-headlines/apple-s-inaugural-m1-chip-puts-company-on-new-growth-trajectory-8211-ceo-62333834

[16] Laptop, Apple M1 vs. Intel CPU: This is the best processor for your laptop https://www.laptopmag.com/news/apple-m1-vs-intel-cpu-this-is-the-best-processor-for-your-laptop

[17] Tech Journeyman, Snapdragon 888 vs A14 Bionic Comparison: Architecture, Benchmark and Ecosystem https://techjourneyman.com/blog/snapdragon-888-vs-a14-bionic/

[18] Linus Tech, Is Apple's betrayal the end of Intel? https://www.facebook.com/LinusTech/videos/is-apples-betrayal-the-end-of-intel/1373510769705207/

J - Energy efficient Mobile Cloud Computing

Sriramm Muthyala Sudhakar
Computer Science
San Jose State University
San Jose, CA 95192
669-260-4406
sriramm.muthyalasudhakar@sjsu.edu

ABSTRACT
Achieving long battery lives has been a long-standing challenge for the design of mobile devices. Though the capacity of the battery might have increased over the past decade, the battery consumption effectively depends upon the amount of computation that is performed on the device. To overcome this situation, a novel solution is presented which couples mobile cloud computing and microwave power transfer (MPT) to enable computation in low complexity devices such as sensors. The base station plays a major role here by transferring the computation from the mobile device to the cloud or vice versa. The framework for the same contains a set of criteria that controls the CPU cycles for local computing, time division between MPT, and offloading for the other mode and mode selection. Moreover, the policies aimed at maximizing the probability of successfully computing the given data called the computing probability under the constraints of energy harvesting and deadline. We use these policies to minimize energy consumption on mobile devices and maximize the energy savings for offloading using convex optimization theory.

Keywords— Wireless power transfer, Cloud computing, Mobile computing, Microwaves, Offloading

1. INTRODUCTION

Mobile devices are becoming more and more smart every day due to the presence of sensors, the Internet, and the Internet of Things. The presence of these applications and devices inside a smartphone or any smart device requires constant power to be supplied to these devices, at least when they are enabled. Moreover, all these devices present inside a smartphone transmit and receive signals through wireless communications. Wired systems are slowly replaced by wireless systems which are cost-efficient and give the user freedom to move around. Though the devices are helpful in this way, the devices need to be powered by a battery. With the presence of sensors and IoT devices, the battery consumption is much more in smart devices. A good example of this would be a GPS tracker in the mobile. When it is enabled, we would notice that the power would drain much faster than usual. A smartphone that can be unlocked by face recognition would keep that module active all the time so that it would unlock the mobile without the user pressing any buttons. That is the device is “on a watch” for the user to give some input [1]. These abilities, though extremely helpful, take up a lot of computational power on the devices. Hence, it is a challenge to guarantee the quality of the device to the user and in turn, provide a good quality of experience as well. To ensure this, we present multiple options that can help us provide a good degree of service. These problems can be tackled [1] by many technologies: 1) microwave power transfer, which would transmit and receive the information using microwaves 2) mobile computation offloading where we offload intensive tasks to the cloud, and 3) doing efficient computing on the mobile CPU.

1.1 Prior Work

Energy-efficient mobile computing has been one of the prime research topics [3] ever since wireless devices existed. Though more and more capacity can be added to mobile devices and as a matter of fact we can never end up at a dead-end, the technology that we might be looking at might take decades to happen. Currently, research has been focusing on designing mobile-cloud computing [4], [5] where we offload the tasks from the mobile device to a cloud server to

let the server take care of jobs that are out of the mobile device's computational power, by using virtual machines [6] or by code-partitioning techniques in the mobile devices [4].

Energy-efficient local computing on mobile devices has also been a prime research topic as multiple techniques have been proposed to reduce the load on the mobile CPU and the battery consumption [7] - [12]. We can reduce the energy consumption by reducing the CPU cycle frequencies [9] – [11].

2. MICROWAVE POWER TRANSFER

Microwave power transfer is one of the commonly used techniques in Wireless power transfer. Research on MPT started a long time ago, in the early 20th century [13] when Nikola Tesla was conducting experiments on MPT. The system uses a microwave generator to generate microwaves and uses them to transmit and receive the messages. The DC current from the transmitter is converted into a microwave and then transmitted to the receiver. At the receiver end, the recipient gets the microwave and converts it back to DC current. The diagram depicts the method in which the MPT happens.

At first, the microwaves are generated using the Microwave power source. The microwaves are passed to the coaxial cables which are connected to a waveguide via an adaptor. All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified. The main purpose of a waveguide is to guide the waves and they are sometimes referred to as the transmission line.

From the coax-waveguide adaptors, the waves are sent to the waveguide circulators. The waveguide circulators are devices that have multiple ports, which creates isolation between transmitted and received signals. These circulators are commonly used for microwave signals in RF equipment and systems [20]. Another purpose of the waveguide circulator is that it reduced the radiation of the microwave [19].

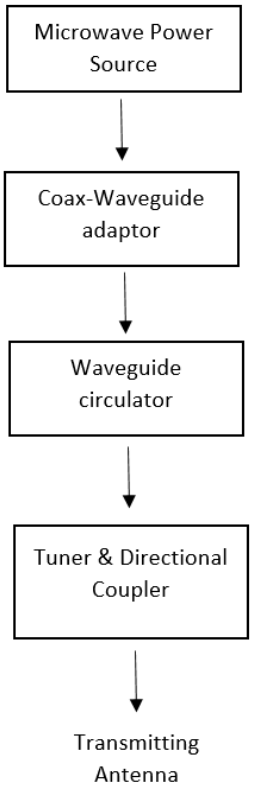


Fig 1. Architecture of transmitter. Adapted from [13]

Finally, the radiation passes to the Tuner and Directional Coupler, which assigns the directions to the signal. Finally, the signal is sent out via antenna. There are many devices that are used to transmit microwaves to space [19] such as magnetron, klystron, and traveling wave tube. The magnetron generates microwaves by exciting the electrons using a magnetic field, after which they are transmitted. These signals are received by the receiving antenna. Below is a diagram comprising of the components of the receiver.

The receiving antenna receives the signals from the transmitter and passes them through a low pass filter. The low pass filter allows the signals of the frequency of 0Hz to a particular cut-off frequency to pass through it.

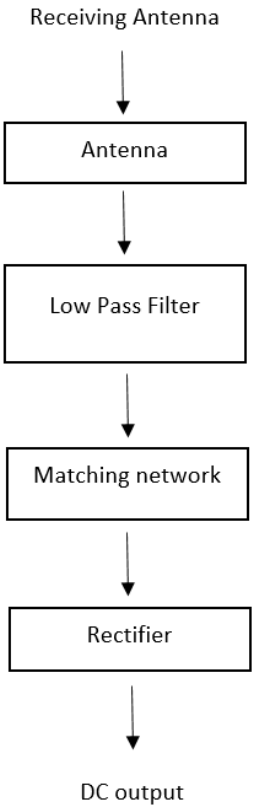


Fig 2. Architecture of receiver. Adapted from [13]

It would ignore all the frequencies greater than the cut-off frequencies. From the low pass filter, the signals are sent to a Matching network that helps to match the impedances in the transmission lines [21]. Finally, a rectifier converts the waves back into DC current.

2.1 Other applications of MPT

MPT has been the prime source of research for powering electric cars. The primary cause of global warming has been the emissions from automobiles. With the petroleum rapidly depleting, people have started to look at other options such as electrically powered vehicles. Current options are the vehicles need to be connected to a power supply through huge power cords. MPT provides options to charge the vehicles wirelessly.

A similar idea as above is implemented for charging smart devices such as mobiles and watches.

2.2 Disadvantages of WPT

Practical implementation of WPT is very high compared to the current wireless power transfer methods. WPT has interference issues with the current system as well.

3. MOBILE COMPUTATION OFFLOADING

Mobile computation offloading is a service that allows the user to use the ubiquitous wireless services, where the computationally intensive tasks and tasks that require high storage [3] are offloaded to them. Traditionally, these services were used to hardware accelerators which would provide higher computational power compared to a mobile CPU. Cloud computing is becoming more and more famous, and it provides higher computational power and storage than most of the hardware accelerators as clouds give us the option to process the data with a CPU, GPU, and a TPU. The advantage that these offload computing provide us is that we do not need to buy expensive hardware and software to perform some computation, rather we can rent it out from the cloud service providers. These services are extremely common these days and are becoming cheaper and cheaper every day. Mobile cloud computing enables developers to build applications that are designed for mobile users without the barrier of mobile OS, storage, or memory. Mobile cloud computing is a combination of cloud and mobile computing. Some of the common examples for mobile cloud computing are

- Games that require high memory, graphics
- Downloading and installing big applications (by size)
- Image processing

Hence we can see from the above examples that it is really common to offload the tasks from the mobile device to the cloud. The above examples are some of the most common examples of offload computing, due to hardware limitations on the mobile device. There are two types of mobile cloud offloading. They can be arranged into two or three-level hierarchies.

3.1 Two-level hierarchy

In the two-level hierarchy, there are only two modules present- the mobile device and the cloud service. The mobile user offloads the resource-intensive tasks to the cloud through a communication network. The cloud server will perform computations on the task using the required amount of power and will send the results back to the mobile user via a communication network. This hierarchy is simple and effective. But, this system depends upon the availability of the cloud resources at the moment the user requires them. Offloading the tasks to the cloud via any communication network requires a high amount of network bandwidth and network access latency.

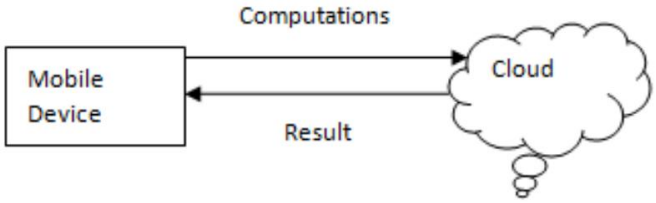


Fig 3. Two-level hierarchy. Adapted from [16]

3.2 Three-level hierarchy

In the three-level offloading hierarchy, there exists one extra module called cloudlet. As the name depicts, a cloudlet is a small cloud-like server, which can be a single data center or a cluster of computations that offers quick service to the users who are in close geographical proximity to this module. This module is particularly helpful when the devices requesting cloud services do not require too much computational power. They can be satisfied using a cloudlet using a Wi-Fi connection.

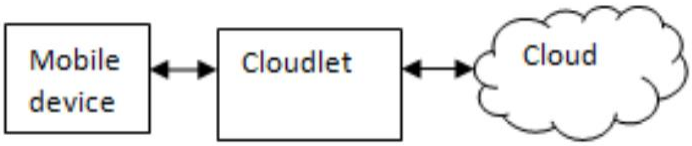


Fig 4. Three-level hierarchy. Adapted from [16]

3.3 Architecture

Before offloading, the information regarding the device and the network are collected. This information is analyzed to provide an estimate on which of the modules require computation [17] on the cloud and which of them requires computation on the mobile CPU itself. Based on these the task is partitioned for local and remote computing. The tasks assigned for local computation are run on the mobile while the remote tasks are offloaded to a cloud server.

3.4 Advantages

From the architecture, we understand that there is a need to offload some heavy tasks to a cloud as we cannot entirely depend on the mobile device. This helps us to save power on mobile devices. Mobile devices might be multi-tasking, due to which a lot of memory and battery will be consumed. Moreover, offloading allows us to push the limitations of the mobile device. They allow us to improve the overall performance of mobile devices.

3.5 Limitations

Offloading reduces the limitations that exist on mobile devices. But there do exist limitations. The most obvious limitation is that the offloading is done on WLAN, which means that we will undergo bandwidth issues [18]. Offloading a heavy task will require more bandwidth, more amount of data to be transferred. Different users might have different wireless services with varying bandwidths, which highly affects the offloading process.

Hence, the decision-making module is important to ideally decide which of the tasks should be offloaded to the cloud and which tasks should be computed locally. There exist latency issues due to the distance between the mobile user and the cloud, which also need to be addressed. Usually, the cloud services are present all around the world and the users are allowed to choose their location. Choosing the nearby locations will help the users to reduce the latency issues.

4. Energy Efficient Local Computing

All of the mobile devices have embedded CPU cores, which are the most basic approach to host some computation. Upon receiving a task, the CPU assigns it to a scheduler, which takes decisions to split the tasks into multiple parts and assigns them to the processor. If there are multiple processors, the

parts are assigned to these processors. Moreover, there exists a decision-making module that decides if any part of the task should be computed on a cloud server. For these tasks to execute, they require some operands and data from the Data Manager, [19] for which the requests are raised. For the tasks that can be run on the mobile CPU, the frequency of the CPU is reduced to improve the efficiency of the mobile.

Mobile Computing is efficient and provides a lot of advantages to the users as they can be carried around and the users have the freedom of movement. For the Internet, people need not go to any specific location to access it. Each year, the cost of mobiles is becoming cheaper and cheaper, and in turn, they offer more capabilities. They are so cheap that even people who are struggling could afford at least a basic smartphone. Moreover, mobile devices provide us with a wide range of entertainment, such as playing games on them, streaming movies, etc.

The issues with the mobile devices occur due to the size limitations. Due to the portability requirement, mobile devices need to be smaller in size. This could be a problem when designing a high-end device. There will be a limit after which we would essentially arrive at a dead-end for reducing the size of the components.

4. Conclusion

A mobile device is a device that is capable of performing the same kind of computations as a plugged-in device. The major issue with a mobile device is the amount of battery that is being consumed for each computation. Also, there exists a limit beyond which a mobile device cannot push its computational speed. This is essentially a triple problem- one where our device cannot perform the required action, the other is that we need more power, and the device cannot possibly offer more. Hence, we need to look beyond our mobile devices. Hence, we looked at different methods by which we could possibly reduce the computation and memory load of the mobile device such as Microwave Power Transfer, Mobile Computing Offloading, and Energy-efficient local computing on Mobile CPU.

5. References

[1] Soliman, Hassan & Saleh, Ahmed & Fathy, Eman. “Face Recognition in Mobile Devices” International Journal of Computer Applications. 73. 13-20, 2013 10.5120/12712-9525. [Online]. Available:https://www.researchgate.net/publication/271069265_Face_Recognition_in_Mobile_Devic

[2] C. You, K. Huang, and H. Chae, "Energy-Efficient Mobile Cloud Computing Powered by Wireless Energy Transfer," in *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1757-1771, May 2016, doi: 10.1109/JSAC.2016.2545382. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7442079

[3] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, “A survey of computation offloading for mobile systems,” Mobile Netw. Appl., vol. 18, no. 1, pp. 129–140, 2013. [Online]. Available: https://www.cs.purdue.edu/homes/bb/mobile-cloud-survey.pdf

[4] E. Cuervo et al., “MAUI: Making smartphones last longer with code offload,” in Proc. ACM MobiSys, Jun. 2010, pp. 49–62. [Online]. Available: https://dl.acm.org/doi/10.1145/1814433.1814441

[5] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, “ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading,” in Proc. IEEE INFOCOM, 2012, pp. 945–953. [Online]. Available: https://ieeexplore.ieee.org/document/6195845

[6] L. Gkatzikis and I. Koutsopoulos, “Migrate or not? Exploiting dynamic task migration in mobile cloud computing systems,” IEEE Trans. Wireless Commun., vol. 20, no. 3, pp. 24–32, Jun. 2013. [Online]. Available: https://ieeexplore.ieee.org/document/6549280

[7] F. Yao, A. Demers, and S. Shenker, “A scheduling model for reduced CPU energy,” in Proc. IEEE Annu. Symp. Found. Comput. Sci., 1995, pp. 374–382. [Online]. Available: https://ieeexplore.ieee.org/document/492493

[8] L. Benini, A. Bogliolo, G. Paleologo, and G. De Micheli, “Policy optimization for dynamic power management,” IEEE Trans. Comput. Aided Design Integr. Circuits Syst., vol. 18, no. 6, pp. 813–833, Jun. 1999. [Online]. Available: www.cecs.uci.edu/~papers/compendium94-03/papers/1998/dac98/pdffiles/11_2.pdf

[9] P. Pillai and K. G. Shin, “Real-time dynamic voltage scaling for low power embedded operating systems,” Proc. ACM SIGOPS Oper. Syst. Rev., vol. 35, pp. 89–102, 2001. [Online]. Available: sosp.org/2001/papers/pillai.pdf

[10] J. R. Lorch and A. J. Smith, “Improving dynamic voltage scaling algorithms with pace,” in Proc. ACM SIGMETRICS, Jun. 2001, vol. 29, pp. 50–61. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/pace.pdf

[11] W. Yuan and K. Nahrstedt, “Energy-efficient soft real-time CPU scheduling for mobile multimedia systems,” ACM Trans. Comput. Syst., vol. 37, no. 5, pp. 149–163, 2003. [Online]. Available: https://dl.acm.org/doi/10.1145/1165389.945460

[12] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, “Energyoptimal mobile cloud computing under stochastic wireless channel,” IEEE Trans. Wireless Commun., vol. 12, no. 9, pp. 4569–4581, Sep. 2013. [Online]. Available: https://ieeexplore.ieee.org/document/6195685

[13] Rajeev Mehrotra, “Cut the Cord: Wireless Power Transfer, its Applications, and its limits” cse.wustl.edu. https://www.cse.wustl.edu/~jain/cse574-14/ftp/power/ (accessed Nov 10th, 2021)

[14] “What is a Waveguide Circulator” chetumenu.com. https://chetumenu.com/what-is-a-waveguide-circulator/ (accessed Nov 10th, 2021)

[15] “Understanding Matching Networks” allabourcircuits.com. https://www.allaboutcircuits.com/textbook/radio-frequency-analysis-design/selected-topics/understanding-matching-networks/ (accessed Nov 10th, 2021)

[16] P. Garg, S. Sharma, S. Kaur, “Offloading approach in Mobile Cloud Computing”, International Journal of Engineering Sciences and Management Research, Mar 2017. [Online]. Available: https://www.academia.edu/31876707/OFFLOADING_APPROACH_IN_MOBILE_CLOUD_COMPUTING

[17] M. Dudeja, K. Soni, “Offloading Schemes in Mobile Cloud”, International Journal of Computer Applications, June 2014. [Online]. Available: https://research.ijcaonline.org/volume96/number8/pxc3896563.pdf

[18] Junaid Rehman, “Advantages and disadvatnges of mobile computers” itrelease.com. https://www.itrelease.com/2020/10/advantages-and-disadvantages-of-mobile-computers/ (accessed 10th November 2021)

[19] Rosa M. Badia, “Programming models for mobile environments” upcommon.upc.edu. https://upcommons.upc.edu/bitstream/handle/2117/121038/TFJLG1de1.pdf;sequence=1 (accessed 10th November 2021)

K - Evolution of Virtualization

Ishaan Aggarwal
Computer Science Department
San Jose State University
San Jose, CA 95192
ishaan.aggarwal@sjsu.edu

ABSTRACT

It is obvious in our minds that a 'computer', having its own operating system, will have its own set of resources which are fixed and belong to that system only. For a conventional industry computer, this might seem true but gone are the days where large-scale computers had their individual hardware resources. In the era of cloud computing and virtual machines, computers have nothing but slices of 'virtual resources', which the OS 'thinks' belong to it specifically, from a large sized physical hardware. This increases elasticity, scalability, and efficiency while decreasing the cost. All of this is made possible by virtualization. How did virtualization become such an important aspect of modern-day computing? This paper will attempt to shed light on this topic.

1. INTRODUCTION

As described by Red Hat [1], "Virtualization is technology that lets you create useful IT services using resources that are traditionally bound to hardware." It is a technology that allows one to use a physical hardware resource to its full efficiency by provisioning and distributing it among multiple computer systems. Using virtualization, an abstraction layer is created on top of a physical hardware machine. In turn, we can create multiple and independent virtual machines with their own operating systems. This allows us to run more than one virtual machine on one physical hardware machine.

2. HISTORY

2.1 Advent - 1950s to 1990s

The term virtual machine finds its origins in mid-20th century. IBM came up with one of the first virtual machine systems in 1960s [2]. Around late '60s, they developed System/360 model 67, which was IBM's first proper computing system with virtualization done for memory. They built a self-virtualizing instruction set for the processor of model 67 and made improvements as new models were launched. The operating system that was used by model 67 evolved into the virtual machine (VM) operating systems. This virtual machine allowed users to run multiple guest operating systems on a single processor computing machine. Since the virtual machine and the mainframe host hardware cooperated, multiple instances of any guest operating system, each having its own protected access to the instruction set, could exist simultaneously and concurrently.

The concept of virtualization in hardware also started to show up around this time. This technology allowed the virtual machine monitors to run VMs in an isolated provisioned environment. Transparency of the virtual machine monitor to the software running in the VM gives the illusion to the software that it has exclusive and independent control of the hardware. The concept was improved steadily and eventually it reached to such efficiency that the virtual machine monitors could function with only small performance and resource overhead. These virtual machine monitors would later be known as Hypervisors.

In 1970's, IBM started developing solutions for time sharing resources. Time sharing meant sharing of the same computer resources by different users. This not only increased the productivity and the efficiency of the computer resources but also reduced the cost of customers. They can use and pay for the resources as per their usage instead of owning the expensive computer resources. In practical life, there is no single process which can utilize the complete server. So instead of using independent resources for different tasks, we can create a pool of services for networking, databases, CPU, etc. resources which can be utilized

by different users at the same time. This led to the evolution of virtualization and its supporting technologies.

Even though the origin of virtualization technology can be traced back to the 1950s, even though the virtualization enabling technologies like hypervisors were developed around that time and was visibly able to give simultaneous access of batch processing computers to multiple users, virtualization didn't come into adoption until the early 21st century.

2.2 Revolution

As we entered the 90s, many companies had physical hardware as servers and single-vendor information technology stacks, which didn't support interchangeability of hardware from different vendors. As the enterprises started to save money by replacing their information technology environments with cheaper commodity servers, operating systems, and applications from a variety of vendors, encountered the issue that they were heavily 'under-using' the physical hardware that they had because each server could only run a single vendor-coupled task.

This became the reason for the onset of research and adoption of virtualization really took off. It was the natural solution to 2 problems: companies could partition their servers and run legacy apps on multiple operating system types and versions. Servers started being used more efficiently (or not at all), thereby reducing the costs associated with purchase, set up, cooling, and maintenance. Virtualization's widespread applicability helped reduce vendor lock-in and made it the foundation of cloud computing. It's so prevalent across enterprises today that specialized virtualization management software is often needed to help keep track of it all.

3. IMPORTANT CONCEPTS

Following sub-sections will introduce some of the important concepts which need to be understood to fully grasp the concept of virtualization.

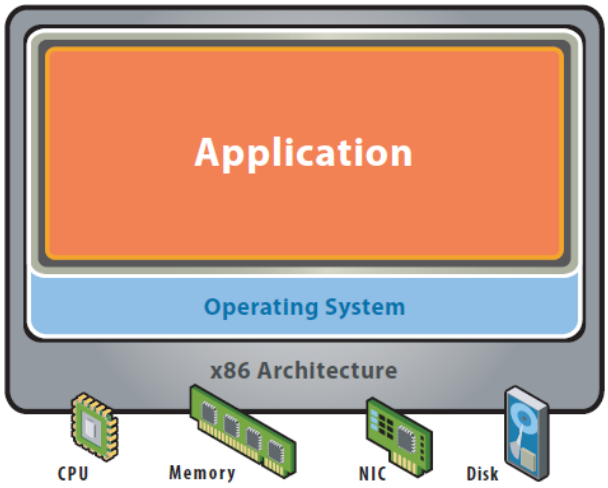


Fig. A computer system before virtualization

3.1 Hypervisor

According to VMWare [3], Hypervisor, is a virtual machine manager/monitor. It is a software that enables the creation, managing and operation of virtual machines (VMs). The objective of a hypervisor is to control one host computer to support more than one guest operating systems by sharing its resources, like memory and CPU, virtually. Hypervisors enable efficient and full use of a machine's available resources and provide independent provisioning of resources for the virtual machines so that they can function independent of the host hardware. This allows easy portability of the virtual machines between different servers. Since more than one VMs can run on a single physical server, a hypervisor increases efficient utilization of space, energy, and maintenance facilities. There are two categories of hypervisors based on how they work and where they work [3].

3.1.1 Type 1 – Bare Metal Hypervisor

This hypervisor is also called native or bare metal hypervisor. It runs directly on the host hardware and can manage multiple guest operating systems. It behaves like a host operating system which can control the hardware directly to let the programs running in it execute. VMs are the programs and their 'virtual' resources are scheduled for the actual hardware by the hypervisor. This type of hypervisor is commonly used in enterprise data centers and other server-based environments where there is a lot of requirement to allocate and provision resources for new machines and the hardware is present in bulk.

3.1.2 Type 2 – Hosted Hypervisor

A type 2 hypervisor, also know as hosted hypervisor runs on a typical operating system installed on a machine, as a software layer application. It is just like any other software installed on that system and behaves like it. It creates abstractions from the host operating system to pretend as if it's just a bunch of resources to the guest operating system. For the host operating system, resources are scheduled just like it happens in any other software, which are then executed by the actual hardware. This kind of hypervisor is the go to option for users who want to run multiple os on a personal computer.

3.2 Virtual Machine

It is a representation of the physical system with its own operating system and set of resources. But since these are virtual and they don't have hardware of their own, they need to use the underlying hardware of the actual server. They cannot directly interact with the hardware so they use a software called hypervisor to interact with the hardware. Commonly called VM, a virtual machine is exactly like any other physical computer system like a desktop, laptop, smart phone, or a data center server. It has its own resources like CPU, memory, storage, network cards, and can connect to the internet if needed, that too with a separate MAC address. The only difference between a VM and a common computer is that the hardware of a common computer is physical and unique to that system, VMs should be understood as software-defined constructs which behave as computers and exist only due to code in a physical hardware environment [4].

According to Microsoft [4], "Virtualization is the process of creating a software-based, or "virtual" version of a computer, with dedicated amounts of CPU, memory, and storage that are "borrowed" from a physical host computer—such as your personal computer— and/or a remote server—such as a server in a cloud provider's datacenter." (We will ignore the new keyword - cloud provider – for now and just think of it as a service) A VM is a computer program which is stored as a file in its dormant form. It is usually called an image. This image, when run correctly, behaves like an actual physical computer system. It can run as an independent computing system which usually runs a different os called guest operating system. Use of virtual machines is very common in present day work scenarios. Both the operating systems, that of the host operating system (primary operating system), and the second operating system (guest operating system) have distinct boundaries from each other and do not interfere with functioning of each other. Just that the host operating system provides of the resources for the guest operating system without actually knowing.

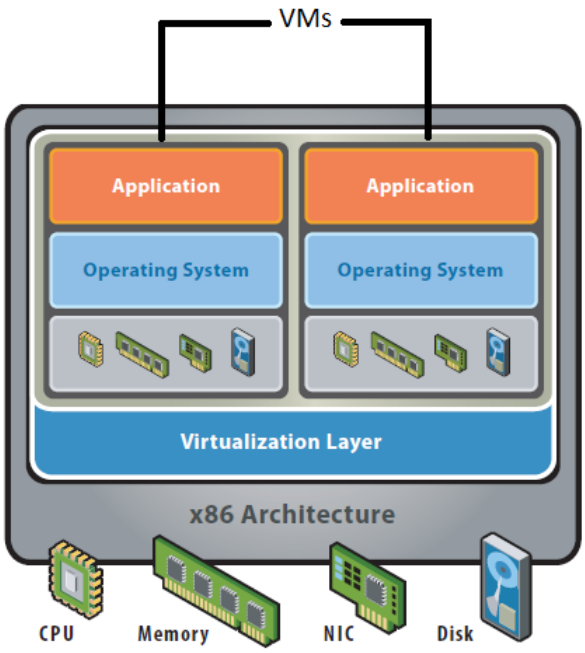


Fig. Multiple virtual machines after virtualization

4. VIRTUALIZATION

Virtualization has changed the IT industry completely. It not only led to changes in decisions related to infrastructure but also in the architecture of the software. People have more money, resources, power, memory and network capacity. It provides high availability, security and easy access. Some of the major advantages are explained below.

4.1 Advantages of Virtualization

4.1.1 Reduced Cost

As we discussed in the last section, that it is very rare or impossible to have a single process using all the resources of the single server completely, dedicated servers for processes leads to higher costs for IT companies. With virtualization, you convert the same machine into multiple virtual machines therefore you

don’t use the whole machine but a part of it which leads to reduction in cost significantly.

4.1.2 High Availability and Fault Tolerance

In case of any disaster in dedicated servers, recovery time is very high. In case of virtualization, we just need to clone the existing VM and run it again on the same underlying machine. There is greater resiliency of the environment which leads to higher availability of the system. This directly enhances the company’s products and the profits.

4.1.3 High Portability

Since there is no direct interaction between the virtual machine and the underlying hardware, it is easy to clone an existing machine and migrate it to a different location. So, virtualization leads to a higher degree of portability. This is used by many data center companies. They can move their machine to a location which is closer to the client's location. For example, one of the IT companies had data centers outside their country. But their government passed a law that user’s data belonging to their country can now reside anywhere in the world except in their own country. In this case, virtualization helped a lot because it was easy to clone and port to the server in their own country.

4.1.4 Increased efficiency and Productivity

IT teams have to spend less time on maintaining the server because in case of any updates, you can install the updates on the server. Otherwise in case of independent servers, you will have to install updates on every server one by one. So, this directly increases the efficiency of the team and saves the cost.

4.1.5 Better Control over DevOps

Since environment consists of VMs, developers can easily and quickly create new machines without any deteriorating impact on the production environment. This is an important use case for development, operational and testing environments. For e.g., to test a new software release version, developers and testers can simply clone the running virtual machine, apply whatever patch they want to test and pull that VM into production after testing. This increases the speed and agility of an application.

4.1.6 Improved Monitoring and Troubleshooting

Since virtual machines are controlled by a lightweight software called hypervisor, we get a centralized place to monitor all the resources and tasks for the virtual machines. Since user is abstracted from actual hardware, their workload can be shifted to any other place until repairing is done. Since user won’t know about this, their services won’t be impacted. This happens because we have control over virtual machines at a single point. Troubleshooting also gets easier because of centralized architecture.

4.2 Types of Virtualization

There are multiple types of virtualizations based on what is being virtualized and for what purpose it is being virtualized.

4.2.1 Desktop Virtualization

This kind of virtualization aims at enabling the use of multiple operating systems on a single desktop without the need to install them on the device. Two sub-types of desktop virtualization are further explained.

4.2.1.1 Virtual Desktop Virtualization

This runs multiple virtual machines on the same server. Each virtual machine can have its own and different operating systems. Based on the need, these virtual machines can be used by the users from any device without installing the actual operating systems.

4.2.1.2 Local Desktop Virtualization

This helps to run and switch between multiple operating system by installing the hypervisor on the local computer.

4.2.2 Network Virtualization

Controlling network hardware devices, like switches and router, directly can be very difficult. Network virtualization creates the abstraction of network devices into the software running on the hypervisor. This way the network administrator does not have to control the hardware devices directly. The network administrator can just control the traffic, nodes and routing through the software.

4.2.3 Storage Virtualization

If we have multiple storage devices installed on the single server on different servers, storage virtualization helps us to manage all storage devices as if there is

only one storage device. Specifically, all blocks of storage are viewed as shared pool of storage for the VMs. Based on the requirement, storage will be allocated to different virtual machines. We can increase and decrease the storage based on the requirement from a single location.

4.2.4 Data Virtualization

These days different format of data is created from multiple different applications in different locations. Data virtualization helps us to view the data at same place irrespective of source and the format. It creates a layer between the software accessing the data and the software storing the data. This layer converts the source data into one common format. This leads to more integration capabilities.

4.2.5 CPU Virtualization

CPU virtualization is the basic building block technology that enables the functioning of hypervisors, virtual machines, and operating systems. It involves dividing a single CPU into multiple virtual CPUs to be used by more than one virtual machine. Initially, CPU virtualization was defined using software, but today, many processors come equipped with extended instruction sets to give support for CPU virtualization. This in turn improves VM performance even more.

5. CONCLUSION

Virtualization technology has come very far from its specific beginnings and is becoming a significant part of building servers and desktop machines. It has played a major role in the advent of cloud computing and distributed computing. There are still some challenges which need to be overcome, like the unification of terminology related to different platforms, developing more robust software, and providing greater device virtualization support independent of platforms. Even with these challenges, virtualization keep making significant improvements and impact on the way we compute.

6. REFERENCES

[1] https://www.redhat.com/en/topics/virtualization/what-is-virtualization.

[2] Campbell, S., & Jeronimo, M. (2006). An introduction to virtualization. Published in "Applied Virtualization", Intel, 1-15.

[3] https://www.vmware.com/topics/glossary/content/what-is-a-hypervisor

[4] https://azure.microsoft.com/en-us/overview/what-is-a-virtual-machine/.

[5] Xing, Y., & Zhan, Y. (2012). Virtualization and cloud computing. In Future Wireless Networks and Information Systems (pp. 305-312). Springer, Berlin, Heidelberg.

[6] Nakao, A. (2010). Network virtualization as foundation for enabling new network architectures and applications. IEICE transactions on communications, 93(3), 454-457.

L - Analysis of Apple's SoC Processors

Nishant Yadav
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000
nishantyadav025@gmail.com

ABSTRACT
With the increased use of personal computers, there is a continuous requirement for improved processors with more computing power now more than ever. The increase in computing power can be increased in multiple ways, by increasing the number of cores in a processor or improving the thermal performance of the processor. In this paper, we discuss the architecture of Apple's M series SoC's, the huge performance improvements and the energy efficiency. We further compare the performance of these chips with other industry leading processors.

1. INTRODUCTION
Whenever we talk about personal computers, the first thing that comes to mind is, which is the microprocessor and how fast it is. As the use of personal computers is increasing, microprocessor manufactures such as Intel and AMD are continuously trying to make them faster by implementing various new technologies and techniques. Up until November 2020, this competition was merely between Intel and AMD. Then Apple released its custom developed M series silicon chips (M1 being the first one and released in 2020) for their mac computers destroying the top-of-the line Intel chips in performance as well as energy efficiency. Not only these new Apple silicon chips were fast but also cost effective when compared to the top-of-the line Intel and AMD chips. The next logical question which comes to one's mind is, why are these chips so fast that they are beating the industry leading chips in performance and energy consumption. There are multiple reasons behind this, the first one being that M1 is not a CPU, but it is a SoC, meaning that it contains all the component on a single chip which are typically separate on Intel chips. These components include the central processing unit (CPU), graphical processing unit (GPU), memory, input and output controllers, and many other specialized chips which perform only certain specialized tasks. Other reasons include the Instruction Set Architecture (ISA) being used, Pipelining etc. These will be explained in further sections of the paper.

Further, Apple have released two new SoC chips in October 2021 namely the M1 Max and M1 Pro. These two chips have further improved the Apple's silicon chips performance by many folds compared to industry leading chips of Intel and AMD. We will also discuss about these chips in some detail and compare them with M1.

2. MICROPROCESSOR (CPU) vs SoC
Microprocessor or the Central Processing Unit (CPU) is commonly known as the "brain of a computer". In laymen terms, a CPU is a very fast calculator. It usually consists of an Arithmetic Logic Unit (ALU) which performs the basic arithmetic operations such as addition, subtraction, multiplication, division, and logical operations. Apart from the ALU, a CPU consists of multiple other chips which performs various tasks. These chips include but are not limited to on board memory for data storage, a graphic processing unit (GPU), input and output controllers, wireless radios, encoders, and decoders etc. All these chips and components together make a personal computer.

On the other hand, a SoC or System on a Chip as the name suggests integrates all the above-mentioned chips and components and many more specialized chips into a single silicon die. We can build a complete computer using a single SoC whereas CPU is just one of the chips among various other chips which are required along with it. Apple's M series chips are SoC, and this is one of the biggest advantages which gives industry leading performance and power efficiency.

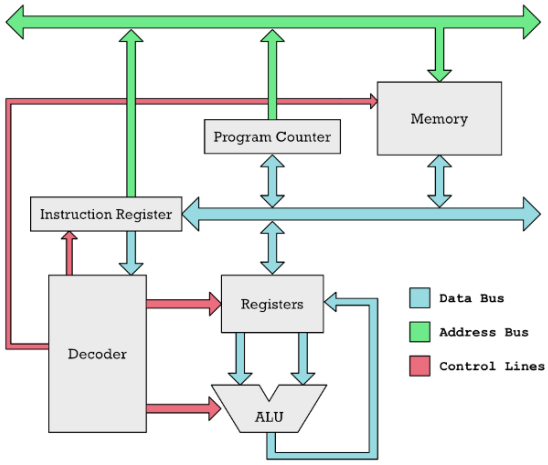


Figure 1. A very basic CPU architecture

2.1 Advantages of SoC over CPU
Since a single SoC houses every chip and component which is required for a computer, the size of the chip is quite small. In fact, it is slightly bigger than a normal CPU, hence it becomes easy to put these chips in smartphones and tablets. This small size also helps in increasing the battery size of the devices as small chips provides more space for the battery.

The second advantage is less power consumption. Since, all the components are present on a single chip with less wiring and very high level of integration, the energy consumption becomes very less compared to a normal CPU with all other required chips and controllers.

The third advantage is high performance. As mentioned in the above point, since all components are present on a single chip accessing memory and processing instructions becomes easy and very fast compared to a normal CPU.

2.2 Disadvantages of SoC over CPU

The only visible disadvantage of an SoC is lack of upgradability and flexibility compared to normal CPU processors. Since every component including CPU, GPU, memory, RAM, input/output controllers etc. are present on a single integrated chip, it is impossible to upgrade any component in near future. But this is the reason for the improved reliability and performance behind the Apple's M series chips.

2.3 Future of Computing Chips

Seeing the current trends, we can say that SoC are the future of computing chips, and that these will take over the CPU market. This can be already seen with the latest CPUs developed by AMD and Intel, namely the Llano and Ivy Bridge respectively which integrates the PCI Express, graphic processing unit (GPU), and the memory controller onto the same chip. Future of computers include handheld and wearable devices and hence SoC are the future chips. CPUs will always have a general market, where power consumption and size are not that much of a concern mainly in supercomputers and servers.

2.4 Moore's Law

In 1965 Gordon Moore observed that the number of transistors in a dense integrated circuit (IC) doubles about every two years. This observation is commonly known as the Moore's Law. It also stated that the speed and capability of a computer can be expected to increase every couple of years and eventually they will cost less. Experts have predicted that the development in integrated circuits will eventually reach its physical limit somewhere in 2020s. This is because of the physical limitation of transistors to operate at increasingly higher temperatures because of smaller size circuits. Hence, to improve performance further manufacturers have started to put multiple CPU cores onto a single chip. Adding more general-purpose CPU cores has been a long-time trend to increase the performance of chips.

2.5 Instruction Set Architecture (ISA)

A CPU can understand only a limited number of instructions and knows about specific registers which it used to complete an operation. This combination of instructions and the registers which are known to a CPU is called the Instruction Set Architecture (ISA) of a CPU.

Two famous ISA are:

1. x86 – commonly used by Intel and AMD chips
2. ARM – being used in new Apple M1

2.6 RISC and CISC Instruction Sets

RISC stands for Reduced Instruction Set Computer and CISC stands for Complex Instruction Set Computer.

In 1970s when CISC processor were being developed, memory was expensive. Hence, there was a need to develop ways to save memory. For this purpose, powerful CPU instructions were created which could do a lot of stuff. Therefore, named as Complex Instruction Set. This helped programmers to write assembly code easily and helped in saving memory. Later it became really complicated to deal with all these powerful instructions, as designing decoders for all these instructions was quite complex task.

Later memory got cheaper and less programmers were using assembly programming, and this led to the development of RISC. For a CISC instruction set, 80% of the time only 20% of the instructions were used from the instruction set. This idea was used to develop RISC in which complex instructions were removed and only a few simpler instructions were used. In RISC, reduced has been interpreted as the reduced number of instructions and the reduced instruction complexity.

3. APPLE M1 CHIP

In November 2020, Apple announced the first system on chip (SoC) developed by Apple for the Mac computers. The Apple's M1 chip is not a CPU, but it is a system on chip (SoC) which contains multiple chips in a single silicon die and CPU is one of these chips.

M1 can be considered as a whole computer on a chip. It contains all important components of a computer including the central processing unit (CPU), graphical processing unit (GPU), input and output controllers, integrated memory, cache, and many other components. Hence, it is called a system on a chip (SoC). As mentioned above in 2.4, the general approach to increase performance is to add more computing power by adding more general-purpose CPU cores. But Apple has taken a different strategy to increase the performance of their chips. Specifically, Apple have added multiple specialized chips instead of general-purpose chips which perform only certain specialized tasks. This addition of specialized chips has led to significantly faster performance compared to general-purpose CPU cores with much less power consumption. This approach has been used previously in graphical processing units (GPU) which performs only graphic related tasks.

Following specialized chips are added to the Apple M1 chips:

1. Central processing unit (CPU) – runs the operating system and user applications
2. Graphics processing unit (GPU) – 2D/3D graphic related tasks including OS user interface
3. Image processing unit (ISP) – speeds up image processing related tasks and applications
4. Digital signal processor (DSP) – speeds up intensive mathematical calculations
5. Neural processing unit (NPU) – used to accelerate Machine Learning and Artificial Intelligence related tasks, such as voice recognition for digital assistants
6. Video encoder/decoder – conversion of video files and formats efficiently
7. Secure Enclave – Apples own designed chip to handle encryption of data, authentication, and security
8. Unified memory – a shared memory pool which allows CPU, GPU, and other cores to quickly access memory

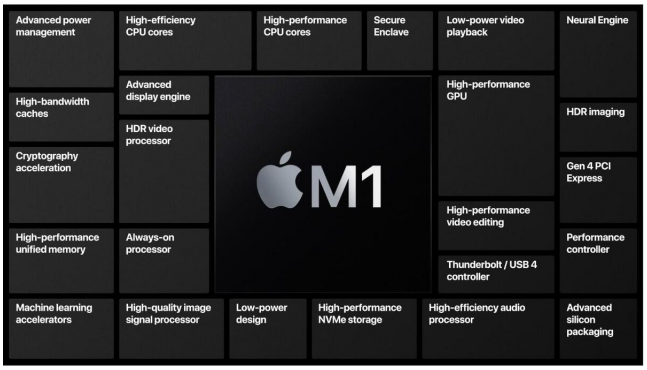


Figure 2. Various specialized chips in Apple's M1

3.1 Unified Memory Architecture (UMA)

Initially the computer systems have had the CPU and GPU integrated into the same silicon die. These GPU were often called as "integrated graphics". But this architecture was slow because of various reasons, one being that CPU and GPU were not able to access the data directly from each other's reserved memory. Instead, the data was explicitly copied from one memory area to another.

The second problem was the heat generated by large GPUs; this doesn't allow to integrate CPU with the GPU.

Apple developed its Unified Memory Architecture to solve these problems:

1. Memory is allocated to both processors and is shared by both processors, hence copying from one area to another is not required
2. Apple uses memory which provides low latency and high throughput
3. Apple reduced the power usage of GPU, so that it can be integrated easily within the SoC. Also, ARM based CPU produce less heats allowing GPU to handle more heat

Using the Unified Memory Architecture has its own tradeoffs. Full integration of memory is required to get such high bandwidth, which means that memory can no longer be upgraded by user.

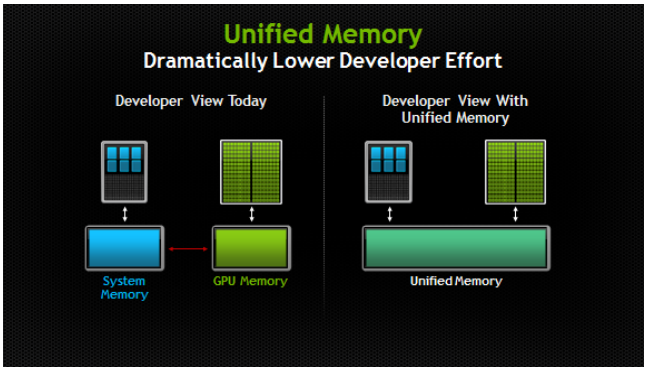


Figure 3. Unified Memory architecture of Apple's M1

3.2 5-Nanometer Process

Apple's M1 is manufactured by using the 5-nanometer process which allows it to pack 16 billion transistors. This led to the production of world's fastest CPU core in low-power silicon, great per watt CPU performance, world's fastest integrated laptop graphics, and industry leading machine learning performance.

3.3 CPU Performance per Watt

Personal computers use multiple chips for the CPU, input/output, security, graphics, and more. With Apple's M1, all these chips are combined into a single SoC, leading to high level of integration which helps in delivering great performance and power efficiency. M1 chip features an 8-core CPU, which consists of four high-performance cores and four high-efficiency cores. The four high-performance cores provide industry leading performance and a huge boost in multithreaded performance. While the four high-efficiency cores provide outstanding performance. Together this SoC architecture delivers great performance at much lower power.

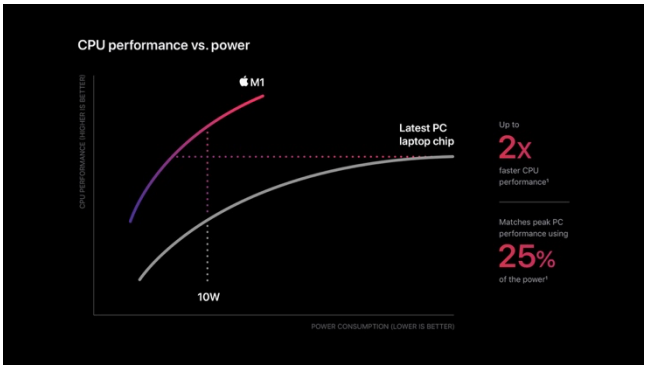


Figure 4. Apple's M1 CPU Performance vs Power

3.4 Apple Rosetta 2

When Apple announced the M1 chip, the biggest challenge was to provide backwards compatibility for the applications which were developed for x86 instruction set architecture. Since Apple's M1 was built on ARM architecture, apps which were developed for Intel's architecture won't run natively on Apple silicon.

To overcome these challenges, Apple developed Rosetta 2 an emulator built into macOS Big Sur which allowed ARM Macs to run old applications developed for Intel CPUs. Rosetta 2 simply translates the instructions written for Intel processors into instructions which can run on Apple's silicon chips. This was part of Apple's 2 yearlong transition program from Intel based processors to Apple's own silicon based SoC. Rosetta 2 simply converts an Intel x86 based application right at the time of installation, creating an ARM based optimized version of the app before the user have even opened it.

Rosetta 2 has its own limitations, for example it's not compatible with some programs, including apps used for virtualization. Hence, you might not be able to run Windows or any other operating system on the M1 Macs using virtual machines.

4. APPLE'S NEW M1 MAX AND M1 PRO

After the release of Apple's M1 chip, in October 2021 Apple released two of its newest M series chips, M1 Max and M1 Pro. Although M1 was fast, provided fantastic performance and best in class power efficiency, it was a somewhat smaller SoC still losing to large and power-hungry chips from Apple's competition.

Hence Apple went all-out for performance, introducing two new chips M1 Max and M1 Pro, adding more CPU cores, more GPU cores, and more specialized chips. These two chips were built on the same architecture as that of M1 but adds more computing power to the SoC compared to M1. These chips mainly improved on the graphics computing power offered by M1 by many folds.

4.1 M1 Pro

Among the two newly launched chips, the smaller sibling M1 Pro seems to be a completely new implementation of the first generation M1 chip. These were built from ground up with scalability and performance in mind. This chip consists of 10-Core CPU, 16-Core GPU, and a total of 33.7bn Transistors, which is almost double when compared to 16bn Transistors present in the M1 chip.

At the heart of the SoC there is a new 10-Core CPU setup, with 8 cores being high-performance cores and 2 being high-efficiency cores. The 8 high-performance cores in the SoC consists of two 4-core clusters, each having its own 12MB L2 caches. This allows each of these clusters to clock their CPUs independently from the other. Compared to M1, there are only 2 high-efficiency cores, and they have 4MB of L2 cache. One large improvement over M1 is the addition of 256-bit LPDDR5 memory, which provides 204GB/s bandwidth. The M1 Pro comes in configuration of up to 10-Core CPU, 16-Core GPU and 16GB of Unified Memory.

4.2 M1 Max

M1 Max is essentially the bigger version of M1 Pro, its architecture is same as that of M1 Pro including many functional blocks – the only difference being that Apple has added much larger GPU and media encoder/decoder into the M1 Max. The number of GPU cores and media blocks have been doubled in M1 Max compared to M1 Pro, which technically provides twice the GPU and media performance.

M1 Max comprises of a 10-Core CPU setup, 32-Core GPU, and a total of 57bn Transistors, which is almost 1.7 times that of M1 Pro and 3.5 times that of M1. The memory interface has also been doubled in the M1 Max to 512-bit LPDDR5 memory, which is very rare in a GPU design and unheard of in a SoC. When the silicon die of M1 Max is compared to that of M1 Pro, it can be observed that the top part of the chip is essentially the same as that of M1 Pro and that the M1 Max grows downwards in the block layout accommodating more of those GPU cores.

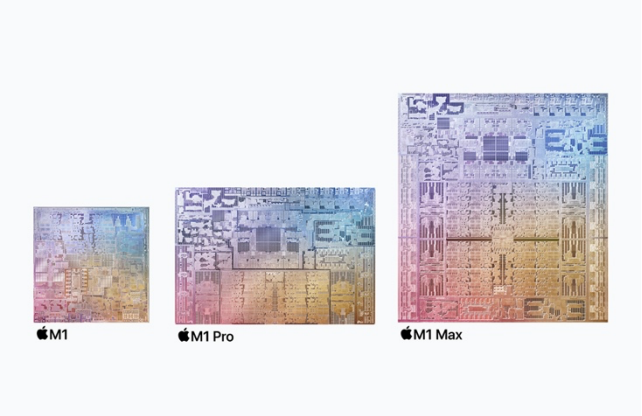


Figure 5. Apple M1, M1 Pro and M1 Max family of chips

5. COMPARISON OF M1, M1 PRO AND M1 MAX

When comparing the benchmarks of the three SoC, it is found that the performance of single cores is almost the same. Hence, the performance of individual SoC varies and depends on number of CPU cores, number of GPU cores, memory bandwidth, and many other things.

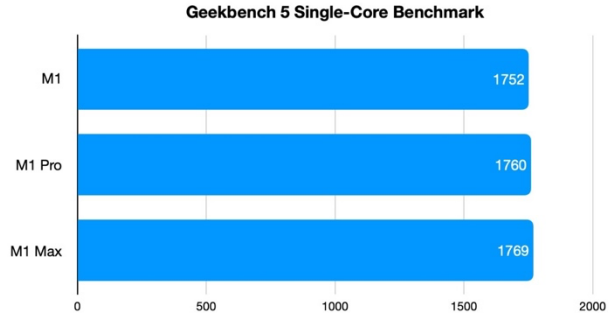


Figure 6. Geekbench 5 Single-Core Benchmark

From the above benchmark scores, it can be noticed that every SoC scores almost the same single-core scores. This makes sense as all the chips use the same high-performance CPU cores.

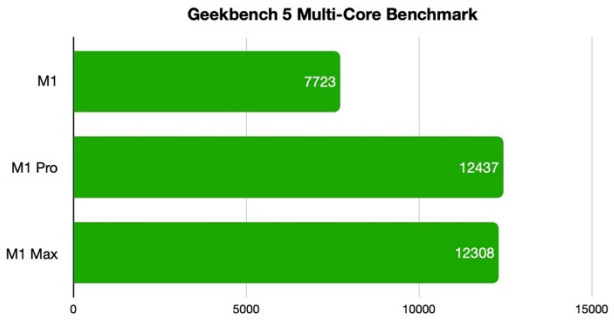


Figure 7. Geekbench 5 Multi-Core Benchmark

From the above benchmark scores, it can be noticed that M1 Pro and M1 Max have almost similar multi-core performance and have much better when compared to M1. This also makes sense, since M1 Pro and M1 Max have same number of CPU cores.

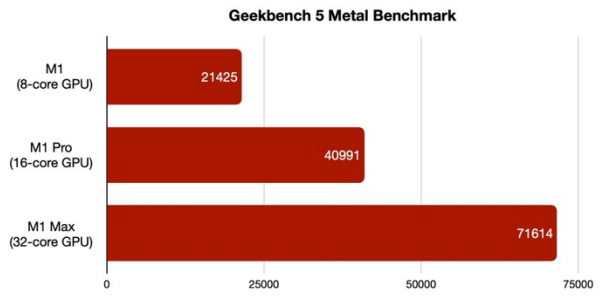


Figure 8. Geekbench 5 Metal Benchmark

The above figure 8 shows the graphics performance of all three chips. This is where we see huge difference in performance of all three chips. This is because M1 have an 8-Core GPU, M1 Pro have a 16-Core GPU whereas M1 Max have a 32-Core GPU. Hence, the benchmark scores make sense.

6. REFERENCES

[1] What does RISC and CISC mean in 2020? https://medium.com/swlh/what-does-risc-and-cisc-mean-in-2020-7b4d42c9a9de

[2] Why is Apple's M1 chip so fast? https://debugger.medium.com/why-is-apples-m1-chip-so-fast-3262b158cba2

[3] Apple Announces M1 Pro & M1 Max: Giant New ARM SoCs with All-Out Performance https://www.anandtech.com/show/17019/apple-announced-m1-pro-m1-max-giant-new-socs-with-allout-performance

[4] Introducing M1 Pro and M1 Max: the most powerful chips Apple has ever built https://www.apple.com/newsroom/2021/10/introducing-m1-pro-and-m1-max-the-most-powerful-chips-apple-has-ever-built/

[5] Apple Unleashes M1 - https://www.apple.com/newsroom/2020/11/apple-unleashes-m1/

[6] Apple M1 Chip: Everything you need to know https://www.macrumors.com/guide/m1/

[7] SoC vs CPU - The battle for the future of computing https://www.extremetech.com/computing/126235-soc-vs-cpu-the-battle-for-the-future-of-computing

M - Evolution of Tensor Processing Units and Comparative Performance Analysis

Pankti Dholakia
Computer Science Department
San Jose State University
San Jose, CA 95192
650-960-6497
pankti.dholakia@sjsu.edu

ABSTRACT

Machine learning has increasingly become a widely used approach in the any industry that deals with large amount of data extraction and processing. This gave rise to the requirement of specialized hardware with an ability to meet the computational and performance requirement. Tensor Processing Unit (TPU) was developed by Google as an application specific integrated circuit (ASIC) that accelerates the execution of machine learning models. In this work we aim to discern the difference in hardware introduced over time with TPU generations v1, v2, v3, Edge and the recently introduced t4. Furthermore, the objective is to provide a performance analysis and review current benchmarks as they were tested for performance.

1. INTRODUCTION

With advancement in computing, requirements for processing power are consistently growing especially with the extensive increase in the usage of machine learning and deep neural networks. Unfortunately, for our general-purpose processors, there is a bottleneck in improving processing power. Modern general-purpose processors are already at their peak processing power feasible according to the limitations of physics [3]. Since the components like transistors will not be getting much better. There is now a limitation to power budget and architectures have already been using more than 1 efficient core and chips for processing. This is why the need for domain-specific architecture has risen in order to address the requirements of the industry [3]. The core idea of this development was general purpose versus specialization. In 2013, Google realized that they would have to double the number of data centers unless they could build a chip that could handle machine learning inferencing. Google's TPU was the first application-**specific integrated circuit (ASIC)** which was designed to accelerate these machine learning workloads [1].

2. HISTORY AD HARDWARE

2.1 TPU v1

The TPU (Tensor Processing Unit) project was started in late 2013 at Google as a DSA (Domain-specific Architecture) hardware. Google initially came out with TPU v1 which had 28nm process with a clock speed of 700MHz and 40W power consumption [1].

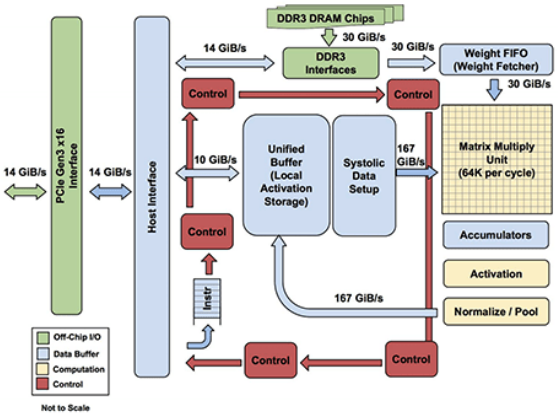


Figure 1. Architecture Block Diagram of TPU v1

It was made as a PCI Express expansion card i.e., peripheral component interconnect express that could be connected to existing architecture by plugging it in, similar to Wi-Fi cards or SSD. TPU v1 were built to plug into the Google's existing servers. The first TPUs have been in production since 2015. It powers things like Google search, translate and photos [1]. The standout feature of this ASIC was that it outperformed existing CPUs as well as GPUs at the time but it had the highest performance per watt energy. The chip featured architecture that was especially specific for deep learning applications. This chip utilised reduced precision, matrix processor and a minimal design in order to reduce the overhead. Tensor in TPU stands for multidimensional array which lies at the core of TPU processing in form of a systolic array that we say in this paper.

The key features of the TPU v1 architecture [3] are:

1. 4 MiB of on-chip Accumulator memory
2. The Matrix Unit: It is a 256x256 (65,536) 8-bit multiplier with accumulate units
3. The clock rate is 700 MHz

4. Peak Execution: 92Triliion operations/second i.e., 65,536 * 2 * 700M
5. 24 MiB of on-chip Unified Buffer which used as activation unit memory
6. It has 3.5x more on-chip memory as compared to a GPU
7. Two channels - DDR3 DRAM of 2133MHz
8. DRAM memory of 8GiB that hangs off -chip

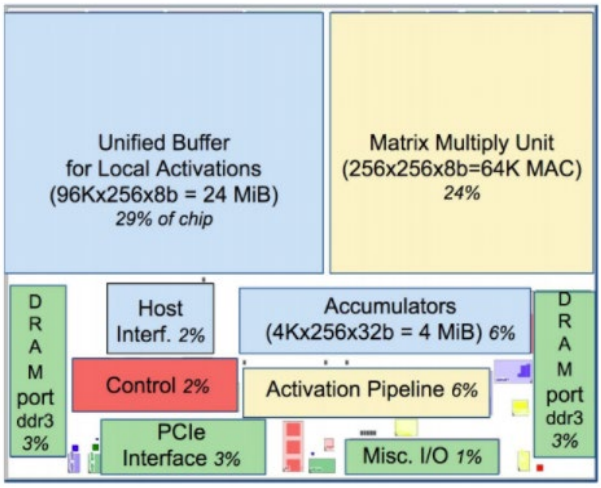


Figure 2. Floor Plan of TPU Die

The layout defines the sections in the chip as yellow = compute, blue = data, green = I/O, red = control

The control logic parts are larger and more complicated in CPUs and GPUs. In the TPU, the control logic or the red parts take up minimal space i.e., less than 2% of the die.

Reducing Precision: The precision of the chip was reduced using just 8-bit integers in comparison to the general practice of 32-bit floating point numbers. The technique used to achieve this reduction is called **quantization –** it maps 32-bit floating point numbers to 8-bit integers. This allowed the designers to fit a significantly larger amount of integer multiplier units on the chip in the limited space.

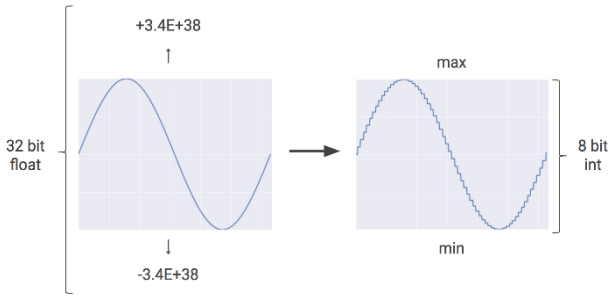


Figure 3. Quantization for reduced precision

Matrix Processor: Conventional processors read and write to memory very often, in certain cases after every single operation. In order to support varied kinds of operations it is required that the hardware design should be generic – but this leads to memory bottleneck due to frequent memory access for every operation. The TPU consists of a special purpose called matrix processor that is not generic in nature, i.e., it only performs a subset of operations but much more quickly. It uses a systolic array which allows the chip to perform large hard-wired matrix calculations without memory access.

A systolic array is a network of processors that perform computations and pass the results across the system.

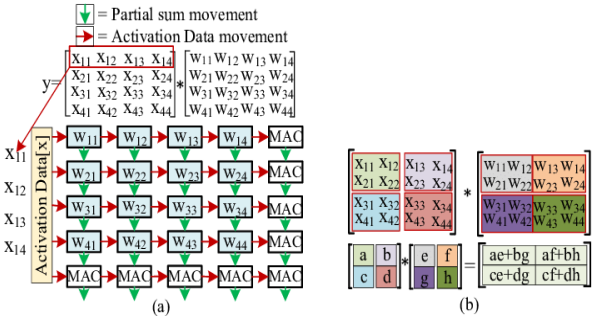


Figure 4. Systolic Array Representation

It consists of a set of interconnected cells known as "processing elements" (PEs). Each PE is capable of performing simple operation.

Data flows directly between cells in a pipelined fashion. This addresses the problem of storing and loading intermediate data, reducing frequent memory access.

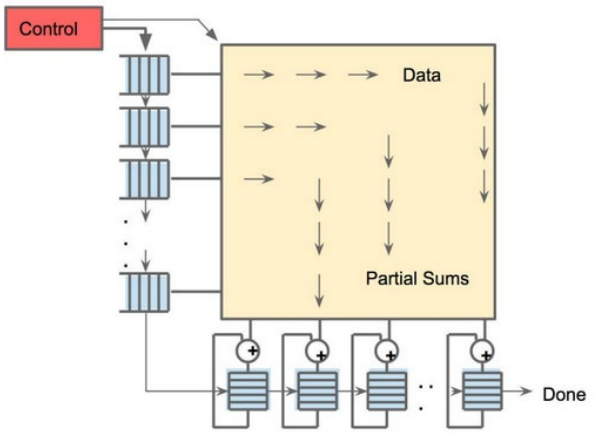


Figure 5. Matrix Multiplier Unit (MUX) of TPU

The TPU instruction set surprisingly follows the CISC tradition, including a repeat field. The average clock cycles per instruction (CPI) of these CISC instructions is

typically 10 to 20 [1]. It has about a dozen instructions overall, but these five are the key ones:

Read_Host_Memory: Reads the data from memory. It reads from the CPU of the host memory in the Unified Buffer (UB).

Read_Weights: It reads weights from memory. The weights from Memory into Weight FIFO as input to the MXU.

MatrixMultiply/Convolve: With this the MXU performs matrix multiplication or convolution from the Unified Buffer into the Accumulators. The matrix operation has an input of size B*256, multiplied by a 256x256 constant weights, and produces the B*256 output. This is computed over B pipeline cycles.

Activate: Apply activation functions.

Write_Host_Memory: writes data from the Unified Buffer into the CPU host memory.

The other instructions are alternate host memory read/write, set configuration, two versions of synchronization, interrupt host, debug-tag, nop, and halt.

The chip can execute up to 265 operations per cycle. Most of the chip is covered with this matrix multiplier followed by an adder. After every multiplication operation, the results are passed to the next layer of multipliers while simultaneously taking the dot product. The output is then given as the summation of all multiplication results between data and parameters.

Comparing the number of operation cycles of CPU, GPU and TPU we can see the computational benefit of TPUs [5]

Table 1. Comparison of Operations Per Cycle

	Operations per cycle
CPU	**A few**
CPU (Vector Extension)	**Tens**
GPU	**Tens of thousands**
TPU	**Up to 128 thousand**

In order to achieve this design, the TPU v1 only did **predictions.** This TPU is not directly available for use to end clients but it is used when accessing Google Photos, Translate and Search.

3. EVOLUTION OF TPUs

After TPU v1 was designed and integrated into the internal architecture at Google, they came up with Cloud TPUs that were made available to the users in order to train their machine learning models and access the computing power of TPUs. Cloud TPUs accelerate the performance of linear algebra computations which are used in machine learning applications. They minimalize the training time to achieve optimal accuracy with training of large datasets using complex neural networks. Models that used to take weeks to train on other hardware platforms can now be trained within hours using the Cloud TPUs.

3.1 TPU v2

It is considerably larger than the v1 of the TPU with 4 chips in place of just one. It consists of 180 tera flops of computing, i.e., it can execute 180 trillion floating point operations in one second. This chip can do both training and predictions.

The layout of the TPU v2 is as follows – each board has four chips and each chip has two cores. Each core consists of a matrix unit, scalar unit and a vector unit connected to 8GB of high bandwidth memory (HBM). In total each board is made of 8 cores and 64GBs of memory. The matrix unit is 128 by 128 systolic array.

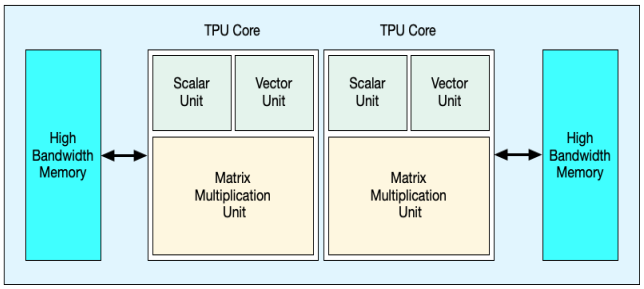
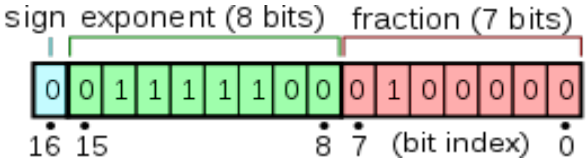


Figure 6. Block Diagram for TPU v2

The salient feature in the v2 TPU is the use of a new **data type – bfloat16.** This data type maps the range of 32-bit floating number to the storage space of 16-bit floating point number. Precision is lost in this translation, but since neural networks can work well with reduced precision, it is a viable trade off to fit in more computational power .

The b in bfloat16 stands for Brain, for the Google Brain team that came up with this custom datatype in order to be used in the Cloud TPU's to reduce precision and increase computational capacity.



Comparison between normal floating-point numbers and bfloat16 [6]-
float32: 1 bit sign, 8 bits exponent, 23 bits significand
float16: 1 bit sign, 5 bits exponent, 10 bits significand
bfloat16: 1 bit sign, 8 bits exponent, 7 bits significand

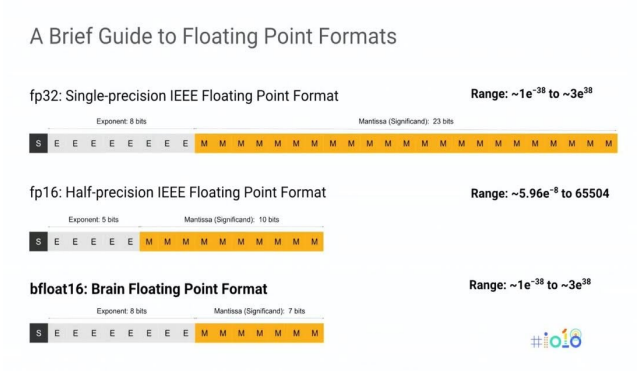


Figure 7. Floating Point Data Type Comparison

Some of the key differences between the TPU v1 and the cloud TPU v2 are -

The MXU in TPUv1 is a 256x256 8-bit integer array, lesser precise than the 16-bit bfloat16 128x128 MXU in the TPUv2.

The activation pipeline is swapped with full vector and scalar units in the TPUv2 to increase the range of executable activations.

Unified buffer is changed with a HBM (high bandwidth memory) which frees up space on the chip for improvement.

These TPU's at Google are arranged in Pods. One pod consists of 64 TPU units. An entire pod can be used as a single machine. That totals up to 11.6 PFLOPS (Peta Floating Point Operations) of processing power. While TPU v1 was built for inferencing, the new version of TPUs known as Cloud TPUs were designed to perform both inference and training.

3.2 TPU v3

The TPU v3 architecture uses water coolant, due to which it takes up lesser vertical space. This allows for the pods to fit a larger number of TPU's in it. The TPU v3 pod is 8 times faster than v2 and has 100 PFLOPS of processing power.

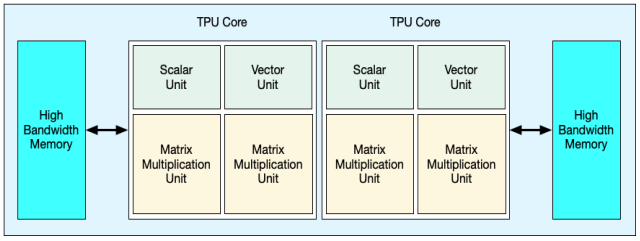


Figure 8. Block Diagram of TPU v3

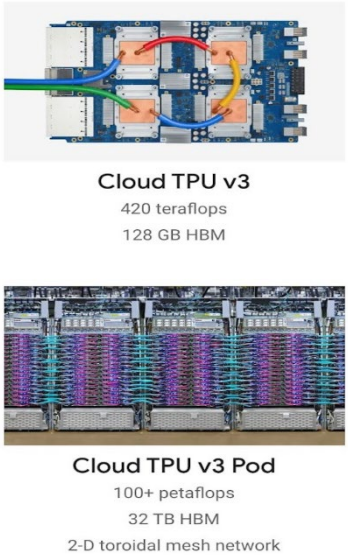


Figure 9. TPU v3 chip and Pod

TPU v3 has a significant increase in FLOPS per core and the memory capacity from v2 [4]. The models run on v3 tend to have the following benefits over v2 [4] –

The configurations of v3 deliver substantial performance improvements per core for computation heavy models. However, there is not much of difference in performance of memory-bound models on v2 and v3.

TPU v2 has a significantly smaller memory capacity, for the cases where data does not fit into TPU v2, TPU v3 improves the performance and offers reduced recomputation of the intermediate values.

TPU v3 architecture configurations support new ML models with larger batch sizes that were not supported by TPU v2. For example, TPU v3 can support much deeper ResNets and larger image sizes with RetinaNet.

TPUs v2 and v3 are available for users to use as a minimal price as a single TPU board, full pod or slices of a pod depending on the usage requirements.

3.3 Edge TPU

Internet of Things (IoT) applications heavily rely on machine learning models to perform inferencing. These applications also use various devices knows as edge devices such as different kinds of sensors and smart devices that constantly collect real-time data and perform processing on it to make intelligent decisions. These edge devices operate in restricted environments with limitations of power including battery, Edge TPU was designed to accelerate ML processes, especially on these low-power devices.

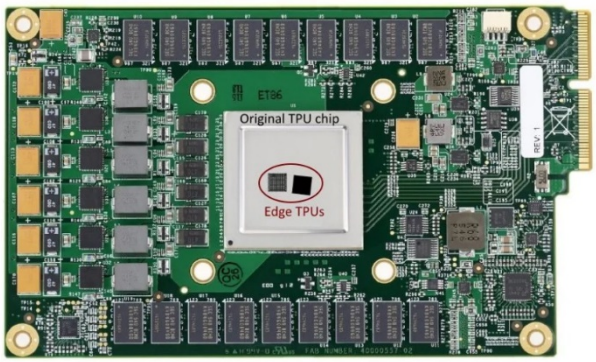


Figure 10. Edge TPU versus TPU v1

Edge TPU offer 2 TOPS (trillion operations) per watt and 4 TOPS per second while using only 2 watts of power. It enables state-of-the-art models like MobileNet v2 for mobile vision at an unbelievable 400 frames per second with power consumption. This accelerator for low-power device combines Cloud TPU and Cloud IoT to deliver an end-to-end architecture to support ML and AI based solutions.

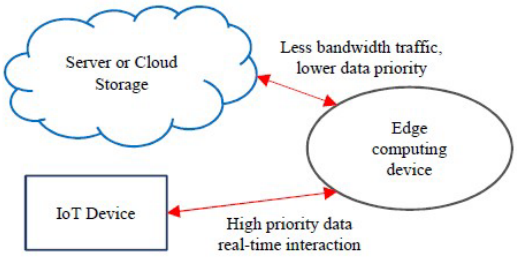


Figure 11. IoT Applications with Edge TPU

Edge TPUs are available for use as a single-board computer, as PCIe/M.2 card, and a surface-mounted module or a system-on-module.

3.4 TPU v4

In May, 2021, Google introduced its 4th generation of TPU which is over 2x faster than its TPU v3.

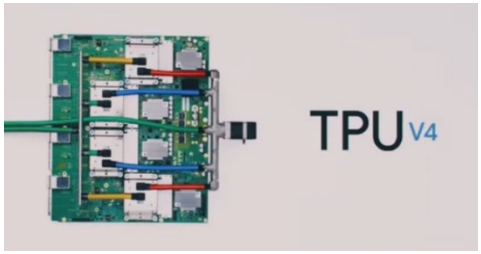


Figure 12. TPU v4 Chip

This latest generation TPU ASIC v4, delivers a significant improvement in raw processing power over its predecessor TPU v3. A single v4 pod contains a total of 4,096 chips, and each pod has 10x the interconnect bandwidth per chip at scale, with a capacity of 1.1 exaflop/s of peak performance. That means that a single v4 pod has greater computing power than a total of 10 million general purpose processors.

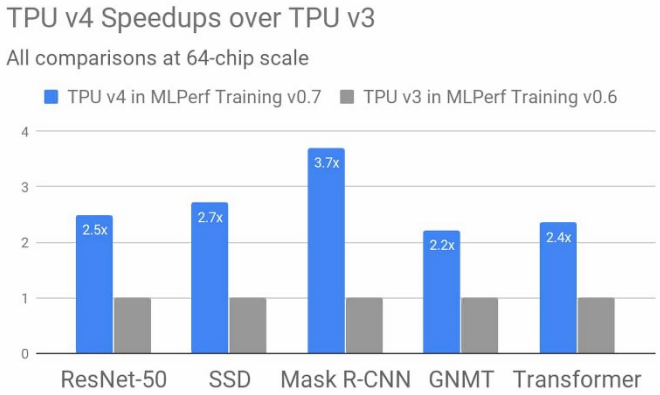


Figure 13. TPU v4 Speedup Over TPU v3

Google is yet to release the details of their state-of-the-art fourth generation chip architecture.

4. PERFORMANCE ANALYSIS

Here's a comparison of current Google TPUs from [2]

Features	TPUv1	TPUv2	TPUv3
Peak TeraFLOPS/ Chip	92 (8b int)	46 (16b) 3 (32b)	123(16b) 4 (32b)
Network links x Gbits/Chip	-	4x496	4x656
Max chips/ super-computer	-	256	1024
Peak PetaFLOPS/ super-computer	-	11.8	126
Clock Rate (MHz)	700	700	940

TDP (Watts) / Chip	**75**	**280**	**450**
Die Size (mm^2)	**<331**	**<611**	**<648**
Chip Technology	**28nm**	**>12nm**	**>12nm**
Memory Size	**28MiB/ 8GiB**	**32MiB/ 16GiB**	**32MiB/ 32GiB**
MXUs/Core	**1 256x256**	**1 128x128**	**2 128x128**
Cores/Chip	**1**	**2**	**2**
Chips / CPU Host	**4**	**4**	**8**

TPU v4 has even more powerful configurations while demonstrating improvement of more than 2.7 times over TPU v3

As mentioned in [3], MLPerf is a newly launched suite of different benchmarks to test the performance of various ML frameworks, hardware accelerators and different cloud platforms.

It has been developed and supported by researchers from universities like Harvard, Stanford, University of California, Berkeley, and University of Toronto. Including companies like AMD, Arm, Baidu, Cadence, Cerebras, Cisco, Facebook, Google, Huawei, Intel, Microsoft, NVIDIA, One Convergence, Qualcomm, Rpa2ai, Samsung S.LSI, Synopsys, Tensyr etc.

According to reports Google's supercomputer with the latest TPU v4 chip has set performance records in six of the eight MLPerf benchmarks.

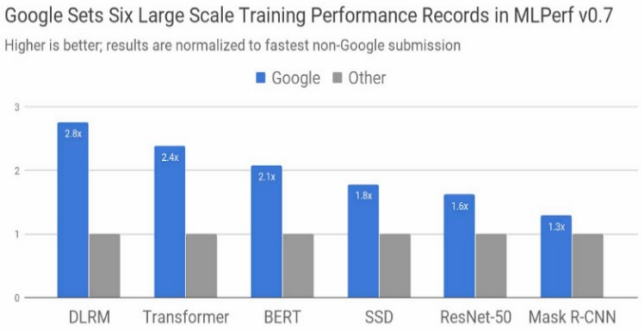


Figure 14. Google MLPerf Benchmarks

5. CONCLUSION

The DSA for ML processing is still at its beginning even with Google's major leaps with the TPU. However, The TPU is a move in the right direction for such specific hardware, and Google has been able to achieve massive improvement in the domain in terms of both architecture and performance. Specific hardware has immense potential to breakdown the costs of training and executing ML models; This creates the opportunity for us to innovate more.

6. REFERENCES

[1] N. P. Jouppi, et al. "In-Datacenter Performance Analysis of a Tensor Processing Unit."

[2] N. P. Jouppi, D. Patterson, et al. "A Domain-Specific Supercomputer for Training Deep Neural Networks." – in *Communications of the ACM*, 2020.

[3] Domain-Specific Architectures for Deep Neural Networks slides by David Patterson, Google AI and UC Berkeley, 2019

[4] HotChips 2020 Conference "Google's Training Chips Revealed: TPUv2 and TPUv3"

[5] An in-depth look at Google's first Tensor Processing Unit (TPU) -*Goole Cloud Blog* - https://cloud.google.com/blog/products/ai-machine-learning/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

[6] A. Paquin blog – "What's inside a TPU"- https://medium.com/@antonpaquin/whats-inside-a-tpu-c013eb51973e

N - An analysis of trends in obtaining energy efficient data centres

Sriya Balineni
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000
sriya.balineni@sjsu.edu

ABSTRACT

Increasing demand for efficiency in performance has led to evolution of computing systems used in data centres which is directly proportional to the total energy consumed. Many practices and theories have been proposed considerably over the last few years to make data centers energy efficient as the power consumption by data centres has been increasing exponentially which is resulting in higher maintenance costs, carbon emissions, higher electricity bills. Many strategies have been proposed by researchers to optimize energy efficiency in data centers. In this study we discuss and analyze various trends and strategies that have been proposed or in action to make data centres energy efficient.

Keywords: Data centers, virtualization, energy efficiency

1. INTRODUCTION

Cloud computing services offered by companies deliver virtual resources charged on a pay as-you go basis. This ability to scale up or down resources according to the demand of customer has resulted in companies towards data centres which is resulting in very high consumption of electricity. As the global climate crisis worsens and very high carbon emissions of data, cloud providers are under pressure to make data centres energy efficient to reduce the harm it causes to the environment.

Improvement of the system performance has been the goal of every company which resulted in constant rise of performance per watt ratio leading to high power consumption by computing systems in data centres. According to current study results electricity consumption by data centres has increased by 65% from 2010-2015. Energy use is expected to continue increasing in the future. Based on current estimates, U.S. data centres have consumed 73 billion kWh in 2021.

Increasing consumption of energy creates an additional requirement to have an efficient and consistent cooling system to provide an optimal environment for the physical equipment present in the data centers along with massive electricity bills. Having an effective cooling system has become very crucial as the consumption of energy in data centers is improving every day. Therefore, heat dissipation techniques must be improved for the proper operation of data centers and energy efficiency. The main goal of this paper is to present an overview of the latest studies and works towards achieving energy-efficient cloud data centers.

An overview and analysis of the studies and research taken place in designing energy efficient data centres is presented in section 2. Section 3 consists of a conclusion and future works.

2. ENERGY-EFFICIENT APPROACHES FOR DATA CENTRES

An approach for efficient management of power resources in data centers has been proposed by Nathuji et.al [1]. A novice technique to manage power called soft resource scaling has been proposed by the authors. This technique uses VMM (Virtual Machine Manager) scheduling capacity to provide less period to virtual machines for utilization of resources. Simulations of data centres have helped in finding out that when hard scaling and soft scaling are used together it increases the energy consumption making it inefficient.

An approach for automating energy efficient virtual machine consolidation in cloud data centers using reinforcement learning has been proposed by Rachael Shaw [2]. They implemented a RL agent called ARLCA, an intelligent solution to optimize the distribution of virtual machines across data centres. This approach is more efficient in highly dynamic environments compared to the heuristic algorithms. This research helps in understanding the advantages of using intelligent and adaptive RL modeling to gain make data centers more energy efficient.

An approach to achieve energy efficiency in data centres based on resource allocation has been proposed by Dang Minh Quan et al. [3]. The interesting aspect of their research is the strategy of assigning priority to all the servers based on the number of cores the serves consist of. The server with higher number of cores will have higher probability of having full load, simultaneously server with lesser number of cores will have less probability of having full load thus resulting in having higher chances of being switched off to save energy. Their algorithm assigns the applications with heavy load to new servers with higher number of cores servers while moving applications with light load to old server having lesser number of cores servers. It then switches off as many old servers as possible. Constraints like human effort is also considered when developing the algorithm.

An approach to achieve energy efficient data centres drive by CSL (Customer Satisfaction Level) constraint has been proposed by Hongjian Li and Yuyan Zhao [4]. They implemented a customer satisfaction level driven energy-efficient scheduling framework to make cloud data center energy efficient. Three customer satisfaction level states have been identified based on violation rate of service level agreement, metric based on workload and metric based. on response time. Three states are imperceptible, unusable, and tolerable. When CSL state enters the imperceptible state, energy consumption should be minimized; when cloud system enters an unusable state, customer satisfaction level driven strategy must be utilized to obtain maximum customer satisfaction level as violation of SLA has in prioritizing customer satisfaction level over saving the energy. When CSL state enters a tolerable state, a trade-of strategy is utilized to increase the customer satisfaction level per energy.

The optimized mapping of tasks to virtual machines and virtual machines to physical machines which is known as virtual machine placement problem are important for minimizing energy. Verma et al. [5] have proposed a heuristic bin packing algorithm to deal with VM placement problem while minimizing the consumption of energy in data centres. A pMapper placement framework has been implemented in this paper which minimizes the total power consumed while maintaining the SLA (Service Level Agreement). This framework has an arbitrator, power manager, migration manager and performance manager. The problem is formalized as a continuous improvement where placement of virtual machine must improve at every time frame to minimize the consumption of energy while maximizing the performance. They have implemented a bin packing algorithm based on variable bin sizes and variable costs to address the virtual machine placement problem. pMapper is unique as it addresses the problem of placement of power and migration cost aware application in a heterogeneous server cluster which supports virtualization with live virtual machine migration.

A new design approach called disaggregated server data center for minimizing total energy consumed by datacentres has been proposed by Howraa Mehdi Mohammad Ali et.al [6]. A disaggregated server breaks up components and resources into subsystems. tearing down a disaggregated server when its role is complete due to modularization could be reused for other purposes. The disaggregated server design arranges resources available in data centers in many physical pools, such as processing, memory, IO pools instead of packing resources into a single server. They presented this new design for the photonic DS-based data center architecture. They have also implemented a mixed integer linear programming model for optimizing the virtual machine allocation in the DS-based data center. Finally, for real-time implementation of DS approach they also proposed an energy efficient resource provisioning heuristic for disaggregated server based on the mixed integer linear programming model insights. Experimental analysis proved that in disaggregated server data centers, optimized allocation of pooled resources and their communication power results in 42% average savings of power consumed when compared to the conventional server approach.

An approach to the energy efficient virtual machine allocation in data centres has been proposed by Cardosa et al. [7]. They utilized parameters like shares, min ad max in their research. Minimum and maximum permissible level of CPU resources allocated to a virtual machine is represented by Min and max parameters. Shares parameter are set as percentage, where CPU resources will be assigned to many virtual machines which share resources. This methodology is apt for environments, where a complete support for Service level agreements is not crucial. The implemented algorithm improves chances to decrease the total resources required by a virtual machine to the minimum. One of the drawbacks of this strategy is that the migration of virtual machine is not leveraged for adjusting the allocation at run-time. Another limitation is that CPU is only considered during the optimization. Additionally, it requires the list of priorities to be set initially and does not support them changing dynamically which makes it unsuitable for real world application where priorities keep changing constantly over the period.

An approach to obtain energy efficient datat centres using renewable sources of energy like wind, sun is proposed by Xiaopu Peng et.al [8]. As the study involves usage of renewable sources of energy to power the entire data center, it is important to handle the intermittent availability of renewable energy by deploying a distributed UPS. They have implemented a renewable energy manager called REDUX which manages energy supplied to data centers by grid and renewable sources of energy. REDUX also

achieves a balance between the total expenses of energy and performance of systems. To achieve the goal of making data centres efficient REDUX leverages UPS devices for allocation of energy when the grid price is low or high or when renewable energy generated is not sufficient. The proposed methodology not only provides stable day to day operation at centres but also cut back on the total money spent on electricity.

An approach to server virtualization to make data centers efficient has been proposed by Mohd Shukri Bin Md Desa et al. [9]. To achieve sustainable data center for a cloud, A five-layer model has been proposed by the authors to achieve sustainability. In data centers. It starts from identifying the resources, categorizing the application, categorizing of servers and resource allocation. Selection of vendors is also made to meet the energy requirement of data centres. In the virtualization management layer virtual machines are placed over each other which results in a stack like virtual machine with so many layers. The paper also mentions about the experimental analysis which showed optimal energy efficiency which led to sustainability in cloud data centres. They have also mentioned the possible future works that could be done in resource allocation algorithms using machine learning in data centres.

Han et al. [10] have proposed a virtual machine placement algorithm and another algorithm to identify physical machines with less load to switch them to sleep mode for saving the energy thus making the data centers energy efficient and sustainable. The authors have implemented a combination of the above mentioned two algorithms in cloud data centres to finish the virtual machine consolidation. Results have shown that there is a trade-off between service level agreement violations and the consumption of energy as chances are high that when virtual machines are switched off to save energy consumption, other active virtual machines might be overloaded thus resulting in violation of service level agreements. Moreover, virtual machine placement algorithm would be able to handle the variable load to prevent physical machines from being overloaded to decrease the service level agreement violations dramatically.

Most of the studies to decrease the consumption of energy in data centers is based on minimizing the energy consumed by servers or the energy consumed by network elements. There is no known research on minimizing energy consumed by both servers and network elements. An approach to this problem has been implemented by A Sudarshan Chakravarthy et.al [11] where they implemented a solution for joint optimization of the energy consumed by both servers and network elements using optimal virtual machine scheduling and routing. They have implemented a two-ant colony based meta-heuristic algorithm. The approach has been tested on three basic data center networks. They are 3-tier, B-Cube and Hyper-tree of various sizes the results are compared against the very basic and most frequently used algorithms called first-fit and round-robin algorithms. Results proved that the algorithm developed by authors improved the total saving of energy by 20% compared to the above-mentioned standard algorithms.

The quality-of-service guarantee is important for regulating services between companies which provide cloud services and their customers in the cloud environment. An approach to reduce the total power consumed while maintaining quality of service guarantee is proposed by Sun-YuanHsieh.at.el [12]. They have implemented a virtual machine consolidation approach based on present and future utilization of resources utilizing the “host overload detection” methods to detect the overloading of host and “host underload detection” method to detect if host is underloaded. Utilization of the resources in future is predicted using Gray-Markov-based model. This approach effectively reduces unwanted migrations of virtual machines and the count of active hosts to reduce the power costs.

Multimedia form of data is the most massive data content that is being created now a days as there are many contents streaming, photo sharing applications. Therefore, it is important for multimedia cloud providers to reduce the total consumption of energy by providing a guarantee for quality of service as much as possible. Han, G et.al [13] have proposed an algorithm called “Remaining Utilization-Aware” algorithm for the placement of virtual machines. They have also implemented an algorithm to detect hosts to shutdown to save energy consumption. Both algorithms are combined and implemented in cloud data centers for finishing virtual machine consolidation. It is a well-known fact that there is a tradeoff between the energy consumed by data center and service level agreement violations. However, the Remaining Utilization-Aware algorithm implemented in this study can handle inconsistent workload which prevents the overloading of virtual machines after virtual placement and also reduces the service level agreement violations drastically.

Cooling system of data centers also contribute highest to the total energy consumed by a data centre. Cooling technologies in data centres is used to maintain suitable environment for the proper operation of physical equipment present in data centres. This can be achieved by eliminating the total heat generated by the physical components by transferring the heat to a heat sink. It is important for the that the cooling system operate consistently and continuously. Y. Berezovskaya et.al [14] have implemented a strategy to handle a liquid-based air-cooled cooling method which uses liquid to cool the air. They implemented an

approach to segregate the space at inlet into racks and space of the racks to three zones called top, middle, and bottom. The temperature of air in the bottom zones is lower than the temperature in the top zone, which results in higher utilization of CPUs present in the lower racks than utilization of CPUs in top racks. This approach of distributing the utilization of CPUs will result in less heating of the processors even though the ITE load remains same. This results in reduction of utilization of the cooling systems thus saving energy and reducing the total power costs in a data center. The proposed strategy was tested idea using the simulations of data model and the results showed that the total energy consumption was decreased by 15 - 20%.

Song et al. [15] has proposed an approach for effective allocation of resources in multi-application data centres. The goal is to decrease the total energy consumed by improving the utilization of resources. Resources are allocated to those applications based on priorities of applications, which use many virtual machines instantiated on various physical machines. Utilization of CPU and random-access memory are only considered in making the decisions. A reduction in the performance happened to applications with low priority in case of limited or less resources since prioritized applications utilize the resources. Virtual machines have been preinstantiated on few physical machines and have been assigned by parts of the complete resources. The drawback of the implemented strategy is that the migration of virtual machine is not utilized for adjusting allocation during run time.

Dinkar Sitaram et.al [16] propsed an algorithm to minimize energy consumption considering the availability. The proposed algorithm lets the users of data center to mention a redundancy factor for the applications. Algorithm places applications in failure zones in a very energy efficient way along with the maintenance of redundancy factor for all the applications. They have also implemented an energy efficient placement in the occurrence of a disaster. Experimental analysis show that the selection policy proposed in this paper utilizes lesser amounts of energy with fewer migrations of virtual machines and minimum violations of service level agreements.

3. CONCLUSION

Obtaining energy efficiency is a very important requirement when designing a computing system as they consume huge amounts of electricity. This leads to higher carbon dioxide emissions thus polluting the environment and contributing to the climate crisis. Therefore, an analysis of various approaches to achieve energy efficiency in data centers has been made in this paper.

For future work, I would suggest the improvement of areas which deal with making cooling systems used in data centers efficient as cooling systems contribute over 40% of total energy consumed by data centres. Another research aspect is the cloud federations where cloud computing services of many providers are interconnected to balance the load and demand spikes

4. References and Citations

[1] R. Nathuji and K. Schwan, “VirtualPower: Coordinated power management in virtualized enterprise systems,” ACM SIGOPS Operating Systems Review, vol. 41(6), pp. 265–278, 2007.

[2] Rachael Shaw, Enda Howley, Enda Barrett,Applying Reinforcement Learning towards automating energy efficient virtual machine consolidation in cloud data centers,Information Systems, 2021,101722, ISSN 0306-4379.

[3] Quan. (2012). T-Alloc: A practical energy efficient resource allocation algorithm for traditional data centers. *Future Generation ComputerSystems*, *28*(5),791–800.

[4] Li, H., Zhao, Y. & Fang, S. CSL-driven and energy-efficient resource scheduling in cloud data center. *J Supercomput* **76,** 481–498 (2020).

[5] A. Verma, P. Ahuja, and A. Neogi, “pMapper: power and migration cost aware application placement in virtualized systems,” in Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, pp. 243–264, 2008.

[6] Mohammad Ali. (2017). Future Energy Efficient Data Centers With Disaggregated Servers. *Journal of Lightwave Technology : a Joint IEEE/OSA Publication.*, *35*(24), 5361–5380.

[7] M. Cardosa, M. Korupolu, and A. Singh, “Shares and utilities based power consolidation in virtualized server environments,” in Proceedings of the 11th IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 327–334, 2009.

[8] X. Peng and X. Qin, "Energy Efficient Data Centers Powered by On-site Renewable Energy and UPS Devices," 2020 11th International Green and Sustainable Computing Workshops (IGSC), 2020, pp. 1-3, doi: 10.1109/IGSC51522.2020.9291205.

[9] Desa, Mohd & Samuel, Joshua & Elango, Sangeetha & Johari, Zainudin & Stephen, M. (2018). Energy Efficient Approach using Server Virtualization in Cloud Data Center. 1-4. 10.1109/ROMA46407.2018.8986732.

[10] G. Han, W. Que, G. Jia, L. Shu, “An Efficient Virtual Machine Consolidation Scheme for Multimedia Cloud Computing,”Sensors, vol. 16, no. 2, pp. 246, 2016.

[11] Sudarshan Chakravarthy A, Sudhakar Ch, Ramesh T,Energy efficient VM scheduling and routing in multi-tenant cloud datacenter,Sustainable Computing: Informatics and Systems,Volume 22,2019, Pages 139-151, ISSN 2210-5379,

[12] Sun-Yuan Hsieh, Cheng-Sheng Liu, Rajkumar Buyya, Albert Y. Zomaya,Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers,Journal of Parallel and Distributed Computing,Volume 139,2020,Pages 99-109,ISSN 0743-7315,

[13] Han, G.; Que, W.; Jia, G.; Shu, L. An Efficient Virtual Machine Consolidation Scheme for Multimedia Cloud Computing. *Sensors* 2016, *16*, 246.

[14] Y. Berezovskaya, A. Mousavi, V. Vyatkin and X. Zhang, "Smart Distribution of IT Load in Energy Efficient Data Centers with Focus on Cooling Systems," IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, 2018, pp. 4907-4912, doi: 10.1109/IECON.2018.8591122.

[15] Y. Song, H. Wang, Y. Li, B. Feng, and Y. Sun, “Multi-tiered on-demand resource scheduling for VM-based data center,” in Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009), Shanghai, China, pp. 148–155, 2009.

[16] D. Sitaram, H. L. Phalachandra, G. S, S. H V and S. TP, "Energy efficient data center management under availability constraints," 2015 Annual IEEE Systems Conference (SysCon) Proceedings, 2015, pp. 377-381, doi: 10.1109/SYSCON.2015.7116780.

[17] L. Alsbatin, G. Öz and A. H. Ulusoy, "An Overview of Energy-Efficient Cloud Data Centres," 2017 International Conference on Computer and Applications (ICCA), 2017, pp. 211-214, doi: 10.1109/COMAPP.2017.8079789.

O - Computer Architectures for Deep Neural Network Based Computing

Pooja Shyamsundar
Computer Science Department
San Jose State University
San Jose, CA 95192
pooja.shyamsundar@sjsu.edu

ABSTRACT

Machine learning has always been around as a technique to solve problems computationally, but a lot of computer architectures have evolved around optimization for programming structures commonly used in procedural and object oriented languages. Some machine learning techniques like deep neural networks run differently and there are ways to optimize computer architectures for the kind of data reuse and memory access that happens while running such computationally intensive programs. This paper explores how the computation involved in deep neural networks is different and also how computer architectures are built to optimize for deep neural network based training and data analysis.

1. INTRODUCTION

A typical, general purpose computing system is not designed for the execution of deep neural networks. Processing deep neural networks requires a lot of computational power and the execution is in a lot of ways, different from the execution of say, object oriented or procedural programming constructs.

The capability of machines has been judged early on. When the Turing test was first conducted, a human judge analyzed natural language conversations between a human and a machine that was programmed to generate natural language responses to distinguish the human from the machine. Ever since then, there has been massive research and development in the field of artificial intelligence.

General purpose computing systems are popular but they are designed such that the memory capabilities and the computing components are separated with relatively lower on-chip memory and much data movement between the processing components and storage. This suits most of the general purpose applications that are commonly used but for deep neural network-based applications, this could be a hindrance. This is because there is a large amount of data being dealt with and this causes a significant delay in processing.

When dealing with applications that need a large amount of data processing to be done and are computationally intensive, there are specific architectural designs that are used. AI accelerators are tailored to have parallel processing capabilities and are organized in a way to aid computationally intensive tasks like matrix multiplications.

This paper analyses the contexts in which deep neural network-based architectures have a requirement and gives a short overview of some of the existing architectures in industry.

2. DEEP NEURAL NETWORKS - TRAINING AND INFERENCE

Artificial Neural Networks have layers that are used as memory. They store and analyze the influence of each of the inputs to the output. The layers also store inferences of significance between combinations of inputs. Deep Neural Networks have multiple hidden layers. We can define a Deep Neural Network (DNN) as a function with weights that determine the influence of each input on the outputs. The inference is the result of the DNN. The weights are evaluated using methods like Stochastic Gradient Descent. This method works by optimizing a loss function.

The dataflow in a DNN is chained. Meaning, there is a feed-forward structure. The Deep Neural Network can have many aspects such as an ReLU, a sigmoid core, batched matrix multiplications and so on. In the computation involved in Deep Neural Networks, the matrix multiplications and convolutions are a majority of operations. Convolutions can also be transformed and

computed as matrix multiplications. This allows for the same accelerators to be used for both the computations. As is evident from this, the processing of deep neural network based algorithms are very different from common computing paradigms in software engineering that general purpose processors usually deal with.

3. GENERAL PURPOSE VERSUS CUSTOM COMPUTATIONAL UNITS

3.1 General Purpose Processors

A lot of DNN computations involve using generalized computational units. As an example, parallelized GPUs or CPUs. The parallel processing is implemented as Single-Instruction, Multiple-Data (SIMD) or Single-Instruction, Multiple Thread (SIMT). General purpose CPUs do provide a medley of tools and functionality for fast development and deployment of DNN based applications. CPUs generally have a monolithic compute block with a serial connection to memory and caches. A Deep Neural Network based architecture involving traditional CPUs can be scaled intra-core or multi-core. A lot of these architectures do not provide the maximum resource utilization or energy efficiency. Single-Instruction, Multiple Data or Single-Instruction, Multiple-Thread architectures don't work very well when we deal with say, sparse matrices or any kind of sparse structures. They tend to be slower and consume more power than their customized counterparts.

.

3.2 DNN specific Processors

To Improve the performance of training and scoring using DNNs, there is also an increased use of customized hardware using Field-Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs) to speed up execution. There are a lot of FGPA as well as ASIC based designs that are being used in industry. For example, the Intel Nervana and Google's Tensor Processing Unit (TPU). The disadvantage in dealing with custom hardware is that the fabrication process is long and resource-intensive. The Deep Neural Network based algorithms are constantly evolving, given the amount of research going on in the field and the wide range of applications they care being explored for. The fabrication process is often too slow for this fast paced field. On the other hand, the logic, structural and storage customizability make them significantly faster and energy efficient.

4. TYPES OF HARDWARE ARCHITECTURES

The Multiply and Accumulate operations are the most commonly performed operations in neural networks. These instructions are easy to parallelize. There are two general classes of architectures that are used for the execution of these operations. They are - spatial architectures and temporal architectures.

4.1 Temporal Architectures

The temporal architectures are used in parallel execution such as when CPUs or GPUs are used to execute Deep Learning - based applications. Single Instructions, Multiple Data uses vectors and Single Instruction, Multiple Threads uses parallel threads.

In all these examples, there is a single memory access point. All the computational units access the single storage location.

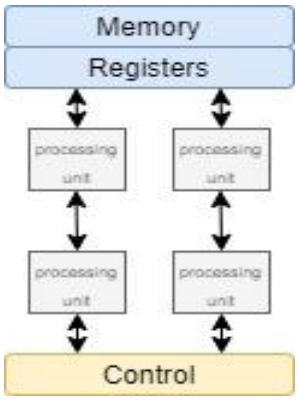


Figure 1. Temporal Architecture

The temporal architectures are used in general purpose computation units such as CPUs and GPUs.

A lot of times, the convolution computation is linked to matrix multiplications. These might lead to data access complexities and latency. So some form of optimization is used to optimize the multiplication operation. It can be something like Fast Fourier Transforms.

4.2 Spatial Architectures

In spatial architectures, the storage locations are provided locally to each computational unit. This allows data transfer between different computational units directly. This model is commonly used in ASIC and FGPA customized Deep neural network architectures.

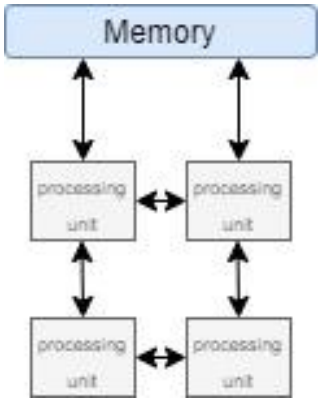


Figure 2. Spatial Architecture

These architectures are used in deep neural network based accelerators. They optimize data flow and energy efficiency. Spatial architectures remove the bottleneck for data access by sort of distributing the memory where each computational unit has it's own local cache. This also improves energy efficiency as the access for a centralized memory costs way more than local access combined with communication with other computational units.

5. OVERVIEW OF SPECIFIC HARDWARE ARCHITECTURES

5.1 Google TPU

The Tensor Processing Unit (TPU) is an AI accelerator Application Specific Integrated Circuit (ASIC). Google's TensorFlow software runs on TPUs.

The TPU is programmable like a CPU or a GPU. The TPU uses a matrix as a primitive and hence gives an optimal way to run deep neural network based computations.

The memory is used to access weights and use them in the matrix multiplication operation. The memory is operated in parallel. There is an external DRAM that is used for weights.

The instructions are used to access data, move the data, compute matrix multiplications and convolution operations and for activation functions.

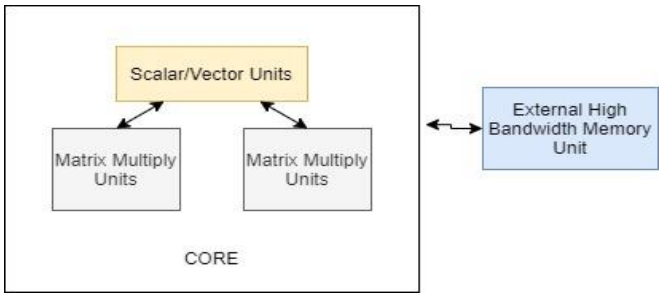


Figure 1. Google TPU core architecture.

5.2 Intel Nervana

The Intel Nervana neural network processor is used for accelerating the training and inference phases for deep neural network based architectures. There are two families of the Nervana processors. The NNP-T and the NNP-I for training and inference respectively. The processors contain Tensor Processing Clusters (TPCs) that can process floating point formats and allows multiple deep learning primitives. It also allows programming using a tensor based instruction set. It allows integration with PyTorch and Tensorflow.

5.3 IBM TrueNorth

TrueNorth by IBM is a deep neural network architecture built using a mesh of processing units. Each core contains axons, synapses and neurons. The mesh operations occur in time steps and the axons are 'activated' if there is a non-zero synapse value. It mimics the functioning of the human brain. The Neuro-synaptic system provides high computational efficiency along with energy efficient functioning. This follows a spiking neuron model. Here there are certain values that are produced as output as digital spikes.

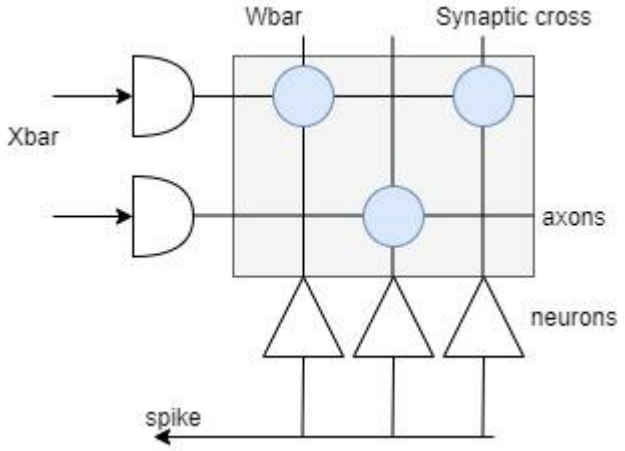


Figure 2. IBM TrueNorth Architecture

5.4 Microsoft Brainwave

The Brainwave architecture is built to deal with the data and memory intensive deep neural network based applications. It makes use of the inherent parallelism in the applications to speed up computation. Scaling is possible through the integration of more FPGAs using a hardware micro-service when one computational unit's memory is exhausted.

The processor is coupled with a Neural Functional Unit (NFU). The Matrix-Vector Unit is used to ensure high utilization of parallel multiply accumulators (MACs). It uses local storage to transfer data to the MACs to ensure low access time.

6. CONCLUSION

Deep Neural Networking based models are constantly evolving and are often not suited for running on general purpose CPUs and GPUs. There are challenges in picking the right hardware and the architecture for different applications. The costs involved in fabrication, design and manufacture of domain specific computing platforms are to be compared with the speed, optimization and energy efficiency offered by custom hardware architectures. While there are multiple computation-intensive operations involved in deep neural network based algorithms, a majority of those are convolutions and matrix multiplications and custom hardware accelerator architectures are built around optimizing for these operations. Generalizing an architecture suited for fast-paced, evolutionary fields like deep learning could be challenging but are nonetheless being attempted widely today.

7. REFERENCES

[1] T. V. Huynh, "Deep neural network accelerator based on FPGA," *2017 4th NAFOSTED Conference on Information and Computer Science*, Nov. 2017, doi: 10.1109/nafosted.2017.8108073.

[2] E. Culurciello, "Hardware for Deep Learning," *Medium*, Dec. 24, 2018. https://towardsdatascience.com/hardware-for-deep-learning-8d9b03df41a (accessed Nov. 10, 2021).

[3] J. Le, "The 2 Types of Hardware Architectures for Efficient Training and Inference of Deep Neural Networks," *Medium*, Oct. 04, 2021. https://heartbeat.comet.ml/the-2-types-of-hardware-architectures-for-efficient-training-and-inference-of-deep-neural-networks-a034850e26dd (accessed Nov. 10, 2021).

[4] "A Survey of Accelerator Architectures for Deep Neural Networks," *Engineering*, vol. 6, no. 3, pp. 264–274, Mar. 2020, doi: 10.1016/j.eng.2020.01.007.

[5] D. Shin and H.-J. Yoo, "The Heterogeneous Deep Neural Network Processor With a Non-von Neumann Architecture," *Proceedings of the IEEE*, vol. 108, no. 8, pp. 1245–1260, Aug. 2020, doi: 10.1109/jproc.2019.2897076.

[6] C.-B. Wu, C.-S. Wang, and Y.-K. Hsiao, "Reconfigurable Hardware Architecture Design and Implementation for AI Deep Learning Accelerator," *2020 IEEE 9th Global Conference on Consumer Electronics (GCCE)*, Oct. 2020, doi: 10.1109/gcce50665.2020.9291854.

P - Simple CPU design

Sida zhong
Computer science dept. Class 247
San Jose State University
San jose, United States

sida.zhong@sjsu.edu

***Abstract*—This project will design a simple CPU. The CPU memory is loaded with a loop program. This program will calculate the sum of 0 to a number N. For example, if the input number is N, the output is 0+1+2+...(N-2)+(N-1). The program is expressed in high-level language as for (i=0;i<N;i++;){result+=i;}. The purpose of this project is to understand how the high-level language is interpreted and run in the machine. The CPU is designed according to the von Neumann structure[1] and contains two main modules, the control unit and the processing unit. The processing unit contains Arithmetic Logic Unit (ALU[2]), a 2-bit address Register File (RF[3]), and a "magic" JUMP instruction module. The control unit contains a ROM[4] memory, the ROM contains the compiled binary code, which is the loop program, an Instruction Register[5] (IR) with a 4-bit address, and a Program Counter[6] (PC). The simulator uses Xilinx Vivado, the operating system is Linux Ubuntu, and the hardware description language is verilog. Initially the project planned to use VHDL, but later found that verilog is obviously more friendly than VHDL and closer to the high-level language syntax, so verilog was used instead. The simulated hardware chip model is basys3 XC7A35T-1CPG236C, which belongs to the Artix family, 28nm technology, and the market price is US$149.**

***Keywords*—Arithmetic Logic Unit*;* Register File*;* ROM*;* Instruction Register*;* Program Counter*;* Vivado*;* verilog; CPU cycle.**

I. Compiler and Instructions

In the early design of the chip, the program and the data were completely separated, and the program was based on the hardware. This makes the computer very inflexible. A computer can only do one thing at the same time. If the circuit is designed as a printer, it cannot be a calculator. The von Neumann structure completely changed this structure. The program was directly imported into the memory as data instead of being designed in the circuit, which directly led to the birth of software and programmers. Therefore, the first thing to do is to parse the high-level programming language into OPCODE. Because the program is very simple, it does not need to use any specific high-level programming language and parser, pseudo code is good enough for this project.

Fig. 1. High-level programming language to OPCODE

```
R3=10;
R0=0;
R1=0;
FOR (R0=0;R0<R3;R0++)
   R1+=R0;
```

```
ASSIGN INPUT TO R3
INIT RO=0
INIT R1=0
R1+=R0
R0++
IF RO<R3 THEN FALSE
IF FALSE THEN GO ADDR 03
OUTPUT R1
OVER
```

Obviously, this program needs 3 registers as variables for ALU operation, R0, R1, R3, which indicates that the address bit of RF is 2. It also needs the operation of ASSIGN, ADDITION, INCREMENT, JUMP, IF. After having

[1] also known as the von Neumann model or Princeton architecture — is a computer architecture based on a 1945 description by John von Neumann and others in the First Draft of a Report on the EDVAC.

[2] In computing, an arithmetic logic unit (ALU) is a combinational digital circuit that performs arithmetic and bitwise operations on integer binary numbers.

[3] A register file is an array of processor registers in a central processing unit. Register banking is the method of using a single name to access multiple different physical registers depending on the operating mode.

[4] Read-only memory (ROM) is a type of non-volatile memory used in computers and other electronic devices. Data stored in ROM cannot be electronically modified after the manufacture of the memory device.

[5] In computing, the instruction register or current instruction register is the part of a CPU's control unit that holds the instruction currently being executed or decoded.

[6] The program counter, commonly called the instruction pointer in Intel x86 and Itanium microprocessors, and sometimes called the instruction address register, the instruction counter, or just part of the instruction sequencer, is a processor register that indicates where a computer is in its program sequence.

OPCODE, an instruction table is needed to summarize these OPCODE in binary code.

Fig. 2. instruction table

INSTRUCTION	ENCODING
Raa=INPUT	00001000aa
ASSIGN Raa=nnnnnnnn	1aannnnnnn
ADD Rcc=Raa+Rbb	0100ccbbaa
INC Raa++	0010aa0001
IF Raa<Rbb THEN FALSE	000111aabb
IF FALSE JUMP nnnn	000101nnnn
OUTPUT=Raa	00001001aa
OVER	0000000000

Finally, these OPCODEs are merged into a compiled binary file according to the instruction table, ready to be loaded in the ROM later.

Fig. 3. instruction OPCODE

ADDRESS	INSTRUCTION	OPCODE
0000	0000100011	ASSIGN INPUT TO R3
0001	1000000000	INIT RO=0
0010	1010000000	INIT R1=0
0011	0100010100	R1+=R0
0100	0010000001	R0++
0101	0001110011	IF RO<R3 THEN FALSE
0110	0001010011	IF FALSE THEN GO ADDR 03
0111	0000100101	OUTPUT R1
1000	0000000000	OVER

The suffix of the binary compiled file recognized by Vivado is .coe, and the format is as follows.

Fig. 4. Vivado instruction.coe

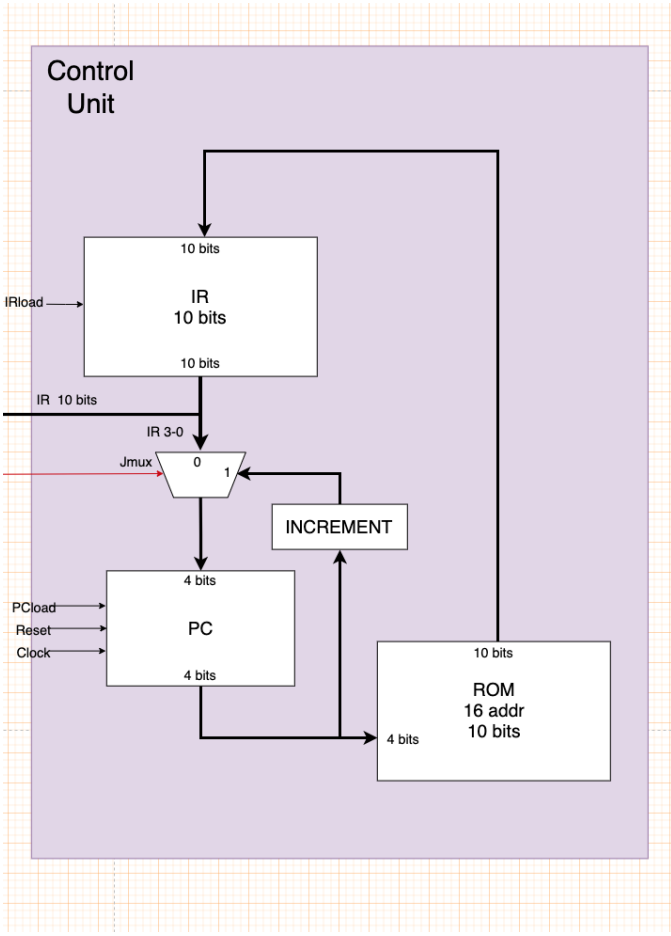
```
memory_initialization_radix=2;
memory_initialization_vector=
0000100011
1000000000
1010000000
0100010100
0010000001
0001110011
0001010011
0000100101
0000000000;

```

II. CONTROL UNIT

The 3 main components of the control unit are IR, PC and ROM. The output is a 10-bit instruction, and the input is Jmux. Jmux is a signal that can control the instruction jump.

Fig. 5. Control Unit



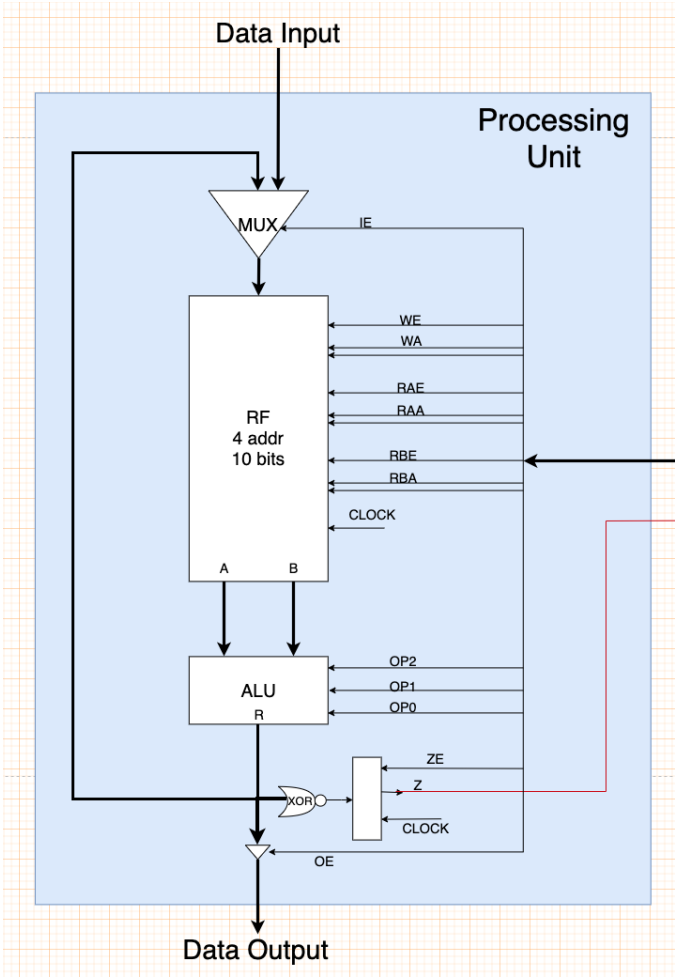
OPCODE will first be directly imported into the 4-bit address ROM. The address of the instruction will be recorded in the PC and driven by an increment module. The instruction from ROM will be recorded in IR to hold the instruction currently being executed. Because the program contains a JUMP instruction, and the 2-bit address of the JUMP instruction is determined by the last 4 bits of the IR instruction. For example, the 7th line of the instruction table is the JUMP command "IF FALSE JUMP nnnn: 00_0101_nnnn", which corresponds to "00_0101_0011 : IF FALSE THEN GO ADDR 03" on the 7th line of the OPCODE. The last 4 bits are 0011 (ADDR 03), which means that the instruction address in the PC needs to jump to 0011. The 0011 instruction address in OPCODE is "01_0001_0100: R1+=R0", R1+=R0 which is the body of the loop. This is how

loop operation is executed in the machine. Therefore, the Jmux signal determines whether the instruction address in the PC is incremented or jumps to the last 4 bits of the instruction address. Jmux is generated from the ALU module according to the instructions output by the Memory, which will be introduced in the next chapter.

III. Processing unit

The two main components of the processing unit are RF and ALU. RF is a register storage that can be fast read and written. The control signals passed by the Control unit like WE(write enable), WA(write address), RAE(read A enable), RAA(read A address).... The output is two 10-bit data, A and B, as the input of ALU.

Fig. 6. Processing Unit



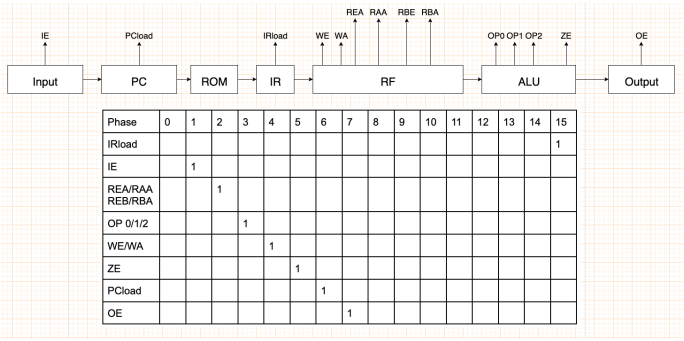
ALU is a combinational logic unit without a clock. The output is a pure function of the present input. When input A, B data comes in, it will directly generate output data R according to the input signals OP1, OP2, OP3. These 3 signals are output by the control unit. For example, the third row of the instruction table is the addition command "ADD Rcc=Raa+Rbb: 0100ccbbaa", which corresponds to the fourth row of OPCODE "0100010100: R1+=R0". The last 4 digits of 0100 are the signals OP2, OP1, OP0. Because there is only addition in ALU, 3 bits are enough. The first 6 bits 01_01_00 represent addresses R1, R1, and R0 respectively, which correspond exactly to R1,R1,R0. Actually the ALU only does the addition of R1+R0, the input result R is returned to RF and assigned to R1. These are the operations given by the instruction returned by the Control unit, and the details will be described again in the CPU chapter.

There is also a special operation in ALU. When the input A>B, the output data R will be 0. This logic corresponds to the XOR gate at the bottom of the circuit diagram, and a signal Z (Jmux) will be output as the PC address jump signal of the Control Unit. In the truth table of the XOR gate, all input signals are 0 and the output signal will be 1. When Jmux is 1, the PC of the Control Unit will carry out the increment command, which means that the JMP command will not be executed, and the OPCODE command will continue to advance, thus jumping out of the loop. In other words, the IF condition that jumps out of the loop in OPCODE is implemented with XOR gates. In a high-level language, this is like (IF A): return A, A can be any type of value. (ELSE A==0) return False, then only False can be returned.

IV. CPU Pipeline

After the control unit and the processing unit are completed, the two modules can be connected together with the CPU, and a pipeline process is required. The pipeline of the CPU mainly has 3 steps. First, get the instruction from the IR, then decode the instruction corresponding to the control signal of each module, and finally connect the signal to the module to perform the operation.

Fig. 7. Pipeline



Phase	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IRload																1
IE		1														
REA/RAA REB/RBA			1													
OP 0/1/2				1												
WE/WA					1											
ZE						1										
PCload							1									
OE								1								

As shown in the figure above, the first step is to get the input data, and then start the PC, so that the instructions in the

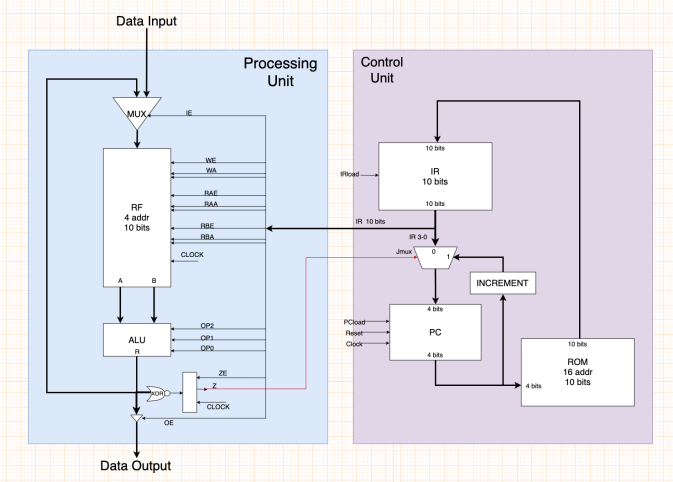
ROM can be input into the IR, and then the RF will get the instructions from the IR. The instruction will tell the ALU to perform the operation, and finally the ALU will output the result. In the specific implementation process, a 4-bit CPU cycle needs to be designed first, with a total of 16 phases. 1 phase has 1 clock time, and it will control the signal switching of a certain module. When the 16 phases are over, one cycle of the CPU ends, which represents the completion of one line of instruction in the IR. For example, as shown in the table in Figure 7, The first signal to be turned on is actually the IRload executed in phase 15. IRload controls the instruction to RF, which means that if IRload is turned off, the CPU will not execute any instructions. The intention of the design is to make the first CPU cycle do nothing, and execute IR after the first cycle is stable, which can improve the fault tolerance rate.

Fig. 8. Incorrect IE signal due to IR delay

IE means input enable and is executed in phase 1 instead of phase 0. The reason why IE skips in phase 0 is that there may be a delay between the instruction load between IR and RF. The earliest design was in phase 0, but Simulation found that IE never had the correct signal.

After phase 1, the RF address will be read (REA/RAA...) in phase 2 first, and perform ALU operation according to the instruction (OP 0/1/2) in phase 3. Then rewrite (WE/WA) the ALU result into RF in phase 4. These 3 phases are a logical operation process, which is expressed as R1+=R0 in OPCODE. The logic operation gives 3 phases of time. Phase 5 is ZE which means enable jump instruction, this signal is Jmux in Memory unit. Phase 6 is PCLoad, which means to execute the next instruction. Finally, phase 7 represents the output result. Only 8 of the 16 phases are used, and the remaining half are in the idle state, but 16 is a magic number. The complete circuit diagram is shown below in fig 9.

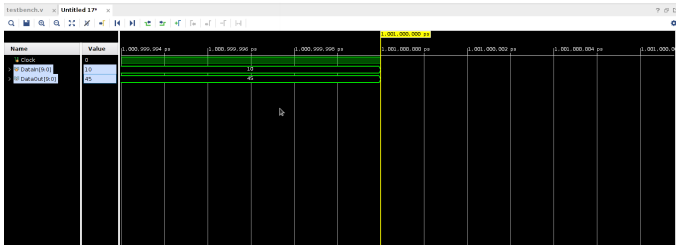
Fig. 9. CPU circuit



V. SIMULATION

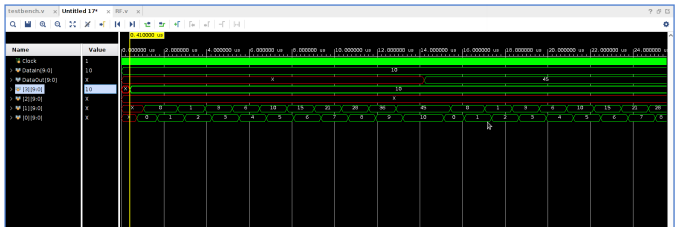
The testbench of the CPU is very simple. The clock is set to switch the signal every 10ns and the input signal is set to 10. Finally, connect the clock, input signal, and output signal to the CPU.

Fig. 10. Input and output waveform



Set the CPU execution time to 1ms, check the input and output signals, The waveform is displayed as expected in fig 10. If the input is 10, then according to the program, the output result should be 0+1+2+3+..+9=45.

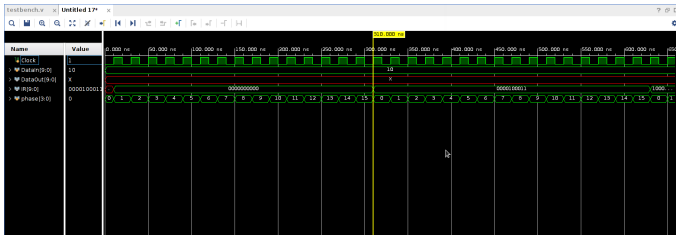
Fig. 11. Input and output waveform



Check the variables R0, R1, and R3 in the program. These three variables are in the register RF. The waveform is displayed as expected in fig 11. As the assigned variable of

input, R3 is always kept at 10. R2 is not used in the program, so the signal is X. R1 outputs the result variable, which is increasing every loop, 0,1,3,6,10,15...45. R0 is used as the IF condition counter of the loop, so in each loop it is incremented as 0,1,2,3,4...10.

Fig. 11. Input and output waveform



Check the status of the phase and instructions in the CPU. As shown in fig 12, in the first cycle, the IR instruction is 00_0000_0000. As mentioned in the previous chapter, PCload is executed from the last phase 15 instead of phase 0, so that the CPU does not perform any actions during the first cycle of startup. Instead, the first instruction is prepared in phase 15 at the end, so the CPU fetches the first instruction of IR from the second cycle. According to the first line of OPCODE, the first IR instruction is 00_0010_0011, and the second OPCODE instruction after the 15th phase.

VI. Summary

A real CPU is definitely a lot more complicated, but this project shows a process of a program from a high-level language, to OPCODE, to binary code, to a control unit, to a processing unit, and combined into a CPU. Although the program is very simple, it involves many commonly used OPCODE instructions, including IF conditions, loops, variable assignment, logic operations, input and output. In simulation, check the status of each signal, including phase, register file, instruction register, input, output, and observe how these signals change in each instruction. In the future, the designed circuit diagram can also be burned in the chip for hardware testing. Instruction table can also be extended with more functions.

References

[1] J. D. Carpinelli, "**The Very Simple CPU Simulator,**" 32nd Annual Frontiers in Education, 2002, pp. T2F-T2F, doi: 10.1109/FIE.2002.1157946.

[2] M. H. H. Ichsan and W. Kurniawan, "**Design and implementation 8 bit CPU architecture on Logisim for undergraduate learning support**," 2017 International Conference on Sustainable Information Engineering and Technology (SIET), 2017, pp. 132-137, doi: 10.1109/SIET.2017.8304123.

[3] Y. Chen and P. Cao, "**Toy CPU: An innovative curriculum design**," 2012 7th International Conference on Computer Science & Education (ICCSE), 2012, pp. 1690-1693, doi: 10.1109/ICCSE.2012.6295390.

[4] https://blog.csdn.net/linzhiheng123/article/details/78960163

[5] https://www.bilibili.com/video/BV1pK4y1C7es/

[6] https://www.youtube.com/watch?v=eVRdfl4zxfI&t=164s

[7] https://www.bilibili.com/video/BV1S7411f7sZ/

[8] https://danielmangum.com/posts/vivado-2020-x-ubuntu-20-04/

[9] https://zhuanlan.zhihu.com/p/109574885

Q - In-depth analysis of Spectre and Meltdown vulnerabilities

Aneesh Verma
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000

aneesh.verma09@gmail.com

ABSTRACT

The Spectre and Meltdown are severe hardware vulnerabilities that affect Intel microprocessors, ARM microprocessors, and IBM processors. These vulnerabilities were considered a disaster that had an impact on all devices including iOS, macOS, Linux, and Windows. In this paper, we discuss the history of Spectre and Meltdown, the mechanism on which they worked on and what kinds of attacks they employed. We also talk about the mitigation strategies and debate whether they worked or not.

1. INTRODUCTION

Meltdown and Spectre attacks can be boiled down to reading the data that is not meant to be read. It can be as simple as reading the memory past the array's end index to as complex as reading the operating system's memory where passwords are stored.

The code that the system generally executes can be of two types. The user-mode code and the kernel code. User-mode code is basically any code that we write, and kernel code is executed only when we execute a system call. With this system call, the control jumps from the user code to the OS. Attacks that allow user-mode code to access RAM that they shouldn't be able to, also called privilege RAM. Both of these attacks rely on features of the CPU that are not generally architecture-dependent. The MELTDOWN allows us (user mode code) to access any RAM inside the computer directly. Virtual memory is a memory management technique that creates an illusion to users of a larger physical memory. Physical RAM is the actual RAM and a form of computer data storage that stores currently executing programs.

Although other processors are also vulnerable, this vulnerability relies on intel-specific architectural features. In Linux, it means it can read physical RAM i.e., read every byte of RAM available. On windows, things get a little complicated. The SPECTRE allows us to access the memory that our process or another process has access to. All modern CPU types are affected by SPECTRE.

Meltdown employs a side-channel attack, which is made possible by out-of-order execution. An example would be because of out-of-order execution, the unprivileged processes can load data from physical memory addresses to temporary registers in the CPU.

Spectre also employs a side-channel attack, it is kind of a software attack to use hardware vulnerabilities to gain exploitation. Speculative execution is heavily used here. To increase the performance, the CPU will guess the future instructions and go that route. If after some time CPU finds out that the decision was wrong, then it will backtrack to that decision. This is more efficient than just sitting idle and doing nothing. Spectre attacks will basically trick the CPU into executing instructions speculatively, which shouldn't be done in normal circumstances.

The reason Meltdown is called Meltdown is because this vulnerability will melt security boundaries which are normally enforced by the hardware. All intel processors which implement out-of-order execution is affected by Meltdown. Some AMD processors are also affected by it.

The reason Spectre is called Spectre is that the name is based on the root cause. Speculative Execution. Almost all systems are affected by Spectre. These include Desktops, Laptops, Cloud Servers, and even Smartphones.

2. HISTORY

There are various ways in which we can protect our system from attacks such as Buffer overflow. We can

employ a non-executable stack, use a canary, use safer languages and functions, or/and use an ASLR. ASLR is Address Space Layout Randomization, which randomizes the place where code is loaded in memory. As an example, Windows Vista uses 256 random layouts, so there is about 1 out of 256 chances of buffer overflow. In July of 2012, apple started using KASLR (kernel address space layout randomization). In March of 2014, Linux started using KASLR. During the years 2016 and 2017 there were several research papers written on breaking KASLR, especially on cache-based architectures. In June of 2017 KASLR was found to have a huge number of new vulnerabilities. Meltdown was discovered by Jann Horn, who was working in Google Project Zero, Werner Haas, Thomas Prescher from Cyberus Technology, and Moritz Lipp, Daniel Gruss, Michael Schwarz, and Stefan Mangard from the Graz University of Technology. Google communicated with Intel and ARM and started to mitigate it. These mitigations were arguably unusual, as with these mitigations, researchers could see other vulnerabilities.

The same team who discovered MELTDOWN also discovered SPECTRE. It was Jann Horn from Google's Project Zero, Paul Kocher, Daniel Genkin, Moritz Lipp, Mike Hamburg, and Yuval Yarom. Microsoft even extended it to the browser's JS JIT engines.

Both of these vulnerabilities were made public in a pair on January 3rd, 2018.

Before the team from Graz University notified intel about this huge security flaw, two separate research teams already told Intel about it. There was another research team that informed Intel about Spectre's vulnerability. All this happened in months. These vulnerabilities that could have been identified even in the '90s, were coincidently identified together much later. It's still not clear whether this was a coincidence that 20-year-old vulnerabilities were discovered at the same time, and no one could pin down this during 20 years.

3. BACKGROUND

3.1 Cache

Caches are a small fast memory physically located on the processor core. Accessing the main memory is very slow from the processor. Just getting from CPU to RAM chip is not possible in a single clock cycle. Therefore, caches are necessary to make the CPU faster. Caches are arranged in 64-byte lines, so if you access a byte of memory not in the Cache, then the 64 bytes around that byte are brought into the cache and from then onwards, the memory is faster to access. The cache is limited so, this is temporary. Usually, there are three layers of cache. Level one is extremely fast, only a few cycles to access, and is very limited (10's of KB). Level two cache is 100's of KB and Level three is MB's of memory. We cannot tell if something is coming from the memory or the cache directly.

3.2 Out of Order Execution

A CPU has many functional units. They have units that can read or write to memory, units that could generate addresses, units that multiply/divide, etc. When instructions come to the front end of the CPU, these units are broken down into mini instructions called microoperations. An example would be, load from memory, add and store back to memory. One CPU instruction is divided into many microoperations. The order in which these instructions are written is not necessarily the order in which they will run on the CPU. To make our code run faster, the CPU rearranges the code, however, we should not be able to figure it out. Example of an out-of-order execution: Suppose we want to load multiple values and then perform some arithmetic calculations on them. If we go by the routine process, and we encounter say a cache miss which consumes hundreds of cycles, the CPU will be sitting idle and wasting its resources. However, with out-of-order execution, till this cache miss is going on for a value, we can load other values and calculate other computations and then use that cache miss value once it loads. This makes the CPU highly efficient.

3.3 Branch Prediction

We could perform out-of-order instructions in the previous example because there was no dependence between those instructions. If there is a dependency, then we cannot do anything but wait. Branches are a hindrance to out-of-order execution. Branches are gotos. They let the processor know where the next instruction is. Branches are of two types: conditional or unconditional. If a certain condition holds, the

conditional branch is used. If the branch is not used, the control goes to the next instruction. The target location can either be fixed or computed. Fixed means that they go to a specific code that is fixed at compile time. For computed targets, the target is computed at runtime. The branch target knows where the branch will go even before it goes there. Branch prediction means making an intelligent guess on which way the branch will go, before completing the instruction on which branch depends on. However, we need to be able to backtrack in case our guess goes wrong. Such instructions are speculatively executed, which is what we talked about before. These instructions are also called transient instructions. There is a big queue of instructions in the order they were written, when we know for certain that the branch has been resolved and the instruction has been executed, these instructions retire. We get rid of speculative instructions that were wrongly determined. An example of branch prediction would be that sorted arrays are much faster in processing than unsorted arrays. This is because the CPU can predict sorted arrays better.

4. MECHANISM

4.1 Meltdown

Meltdown relies on the fact in Linux that the kernel memory is mapped into every process. Even when we run our user-mode programs, the kernel memory is there in our address space. If we could guess where the kernel was, we could maybe access it. However, only kernel-mode code can access kernel memory. The reason kernel memory is mapped in processes is to make OS calls faster. User mode and kernel mode memory are mapped together and are protected by privileged bits as a performance optimization. In addition, ASLR's make it harder to find kernel memory, and even if we do we won't be able to access it. Now we discuss the steps to cause a meltdown attack:

Step 1:

Try to read a memory that we are not supposed to, which causes a page fault, mostly segmentation fault.

The fault only happens when an instruction retires. If branch predictor would have gotten rid of the microoperation then that fault would have never taken place. Therefore, we need the instruction to retire first.

Step 2:

After reading the protected RAM and before the instruction retires, certain microoperations will take place.

Knowing that reading from a protected RAM will cause a fault in some cycles, we will use the read value and do something with it which will affect the cache.

Step 3:

We will handle the fault. As it was a fault, it would backtrack. The series of events that happened are reading from RAM, affecting the cache using that value, causing a fault. All this will be undone and the value that came from RAM would be lost.

Step 4:

We can recover that value because we affected the cache. The cache will not be affected by this backtracking. Undoing the cache is almost impossible. We will use that trace in the cache to recover the original value.

All modern Intel CPUs are affected. So are ARM Cortex A15, A57, and A72. AMD claims that it has not been affected but there have been reports that it is.

4.2 Spectre

Spectre is a vulnerability that tricks the program into accessing locations in the program's memory. We can gather sensitive data by reading accessed memory's content. Spectre outlines a whole class of potential vulnerabilities rather than a single one. The base of these vulnerabilities is similar to Meltdown i.e., it exploits speculative execution, which is used to make CPU faster and more efficient. Spectre's main focus is Branch Prediction which is a subset of speculative execution.

A timing attack is a side-channel attack, in which an attacker compromises a system by analyzing the time taken to do a task. Spectre employs a side-channel timing attack on branch prediction in out-of-order executions. Similar to meltdown, the cache can be affected before mispredictions are discarded and a trace might be left.

Spectre is a suite of attacks that don't rely on specific Intel's faults. JIT is a way of executing code in which compilation is done during execution of a program meaning done at run time, rather than before execution. These attacks can allow us to bypass the

JIT's safety checks in a browser or inside the kernel or reveal a process or kernel RAM.

Now we discuss the steps to cause a Spectre attack:

Step 1: We will train the branch predictor that we are always accessing things inbound. It now predicts that skipping the read never happens. We do so by reading an inbounds value repeatedly so it thinks, this piece of code is always in bounds.

Step 2: Flush some variables from the cache so that a condition such as an IF condition takes forever to load. The code becomes super expensive to run. This causes our exploit code to speculatively runway ahead of normal code.

Step 3: We now call read with an out of bounds index, we know it will cause a fault. It allows us to read anywhere depending on the out-of-bounds index.

Step 4. We get our transient result, which will never be completed and will get discarded. Therefore, we use the result from the read to affect the cache similar to meltdown attack. After the read returns -1, we look at the cache and see the value that was discarded.

These codes can also be written in JavaScript, so to make JavaScript run in a webpage to read every byte of memory. However, chrome and Firefox processes are already divided into sub-processes and this attack would be avoided.

There is another attack that is possible. In this attack, the branch target buffer, once we have figured out how it works on our CPU, we can mistrain a branch for something that is in something else's code. This can happen if we share the branch target buffer with another process. We now mistrain the branch target buffer so that a jump inside another process which may also be a kernel, is mispredicted to go somewhere else. With this, we can make the code affect cache which is dependent on secret information before the indirect jump gets resolved. This would be undone but the traces in the cache would be left. To summarize when the kernel makes an indirect jump, with us mistraining the BTB buffer, it goes to an address, and we have figured out the code at that address in the kernel's context, which does something with some registers to leave a trace in the cache. If these registers can be manipulated or one of the registers is dependent on the secret kernel's value, we can see some unauthorized information. The piece of code at that address is known as Gadget. Gadgets are also used in insecure deserialization vulnerabilities. This attack can be replicated for other processes than the kernel too.

5. MITIGATION STRATEGIES

In reality, we need to modify CPU firmware, need to modify how virtual memory systems work in our OS, need to modify all the compilers, web browsers, hypervisors, etc. Even this will mitigate most of the vulnerabilities and not all. The performance hit will definitely be there. The only actual solution is to replace the CPU with a new design. Having said that, there are some patches and techniques which can alleviate the attacks:

5.1 Meltdown Mitigation

We can make changes to the OS kernel, such as increasing the isolation of kernel memory from user-mode processes. We don't map kernel memory into every process. This is what KAISER patches do. It is also referred to as KPTI (Kernel Page Table Isolation). As a result, the system calls get more expensive. The kernel has to now map and remap the page tables every time we call a system call. If we are okay with making system calls slower, meltdown vulnerability (not 100%) can be mitigated.

Another solution can be of keeping a check when someone tries illegal access and killing the speculative execution. The process would be slightly slower but it would definitely be secure.

5.2 Spectre Mitigation

The fact that Spectre has a lot of attacks, a single patch would not suffice its mitigation. A few ways are described below:

Using a Load fence: Every load up to the load fence (including it) must complete before any instructions after it is allowed to start. This disables out-of-order execution in a way and in turns result in more cost expenditure and slowing of processes. However, it works, so this technique should be used.

Indirect Jump Stoppage: We can stop Branch Target Buffer from doing indirect jumps so it becomes difficult for attackers to exploit code.

Intel has a microcode update that adds new Model Specific Registers that control Branch Target Buffers.

They employ a sort of check on the Branch Target Buffer.

6. VARIANTS

In research, we see that there are 14 meltdown variants (6 non-exploitable) and 13 Spectre variants.

6.1 Meltdown

Some Meltdown variants are as follows:

Meltdown-US was the first variant of Meltdown disclosed. It read the memory of the kernel from userspace.

Meltdown-P is also called Foreshadow. In this vulnerability, a page fault will occur when someone tries to access page-table memory in an unauthorized way.

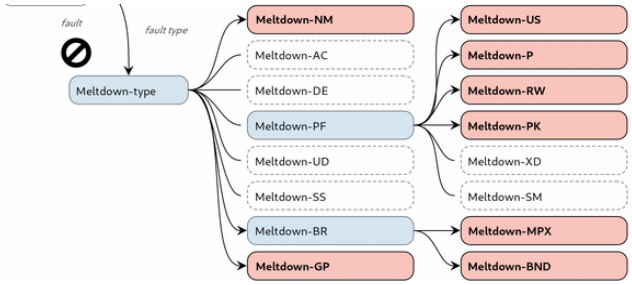
Meltdown-GP is also called Variant 3a. Attackers can read privileged system registers here.

Meltdown-NM is also called LazyFP, can be used to fetch AES-NI keys.

Meltdown-RW can overwrite read-only data having a limited privilege level.

Meltdown-PK exploits the PKU/PKEYs (Memory Protection Keys for Userspace).

Meltdown-BR exploits the exception of a bound range which resides in x86 processors.



6.2 Spectre

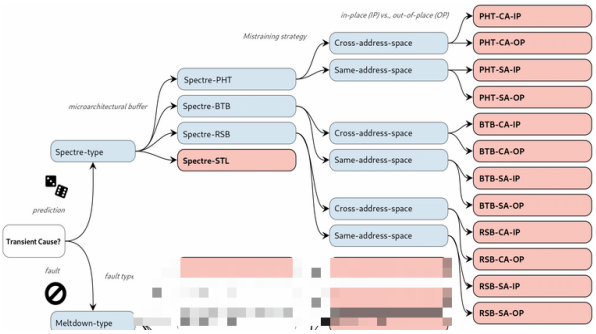
Some Spectre variants are as follows:

Spectre PHT contains Variant 1, Variant 1.1, and NetSpectre.

Spectre-BTB is also called Variant 2.

Spectre-RSB uses the Return Stack Buffer

Spectre-STL also known as Variant 4 exploits store-load-forwarding. This excludes any history-based prediction



7. CONCLUSION

Meltdown and Spectre are dangerous hardware vulnerabilities. Even after having various patches and providing so many solutions, in 2021 hackers can still exploit the flaws e.g., making web browsers leak information, etc. The bottom line of this attack is that we have traded security for efficiency. While trying to make the CPU faster and faster, the security of the CPU has been ignored. The attacks work on various techniques such as caching, out-of-order prediction, and Branch Prediction. We induce this attack by touching the cache while the wrong prediction is not discarded yet. We then get the original value through cache trace.

8. REFERENCES

[1] https://meltdownattack.com/.

[2] https://spectreattack.com/spectre.pdf

[3] https://en.wikipedia.org/wiki/Meltdown_(security_vulnerability).

[4] https://en.wikipedia.org/wiki/Spectre_(security_vulnerability).

[5] https://www.wired.com/story/meltdown-spectre-bug-collision-intel-chip-flaw-discovery/.

[6] https://medium.com/swlh/branch-prediction-everything-you-need-to-know-da13ce05787e

[7] https://www.techrepublic.com/article/spectre-and-meltdown-explained-a-comprehensive-guide-for-professionals/

[8] https://stackoverflow.com/questions/52799661/what-code-is-user-mode-code-and-what-code-is-kernel-mode-code.

R - Role of Computer Architecture in Autonomous Driving

Srajan Gupta
Computer Science Department
San Jose State University
San Jose, CA 95192
669-288-1594
srajan.gupta@sjsu.edu

ABSTRACT

The world has advanced so much in technology that systems like autonomous-driving cars no longer sound impossible. All major automobile companies are in the race to develop the best self-driving vehicle. One of the essential components in any such machine is computer architecture. To support effective functioning in autonomous vehicles the underlying computing framework should be able to provide high performance, low thermal dissipation, and low power consumption all at an affordable cost. This term paper would discuss the computer architecture of autonomous vehicles by giving attention to the above-mentioned points.

1. INTRODUCTION

In these autonomous driving cars, every functionality is achieved by using multiple sensors like - laser imaging detection and ranging (LIDAR), global position system (GPS), an inertia measurement unit (IMU), various cameras. To process the information gathered from all these units and make decisions using it, the system requires CPUs and GPUs with high processing power. The US Department of Transportation's National Highway Traffic Safety Administration (NHTSA) has defined five levels of autonomous driving [1,2]:

Level 0: No automation - It is like normal cars we have today, where the driver is the complete controller.

Level 1: Semi-autonomous cars - Here although most functions are controlled by the driver, the autonomous systems can do functionality like adaptive cruise control

Level 2: Driver-assisted automation- In this level, an autonomous system can take over steering, acceleration, and braking in specific scenarios. But, even though Level 2 driver support can control these primary driving tasks, the driver must remain alert and is required to actively supervise the technology at all times.

Level 3: Conditional driving automation - In this, the driver does not need to supervise the technology, which means they can engage in other activities. However, a human driver must be present, alert, and able to take control of the vehicle at any time, especially in the case of an emergency due to system failure.

Level 4: High driving automation - This autonomy does not require any human interaction in the vehicle's operation and is well programmed to drive under some specific conditions.

Level 5: Full driving automation - This autonomy does not require any human interaction in the vehicle's operation and is well programmed to drive under all conditions, also it is programmed to stop itself in the event of system failure. A human driver is not needed.

This paper is divided into four sections, section II will discuss the different components in an autonomous vehicle along with their functionality, section III will explore the simulation systems for testing such systems, section IV will explore Vision-based autonomous driving systems (ADS) with a focus on Tesla's autonomous vehicles and conclusions will be drawn in section V.

2. COMPONENTS OF AN AUTONOMOUS DRIVING SYSTEM

GPS/IMU: GPS is a global positioning system. The GPS sensor communicates with the GPS satellite in latitude, longitude, and altitude. An IMU is a sensor that can measure rotation speed and translational acceleration using gyroscopes and accelerometers. GPS provides like accuracy but the problem with GPS is the update rate is slow at about 10Hz, also weather conditions like the cloudy sky and physical conditions that tunnel can hinder the reception of GPS, and hence it cannot be trusted for real-time updates. On the other hand, IMU being on a vehicle sensor can provide real-time updates but the accuracy of IMU suffers from aging and cannot be trusted to make life-critical decisions. Vehicles use an algorithm that combines knowledge from both GPS and IMU to determine vehicles' accurate location and movement to solve the positioning problem.

LIDAR: LIDAR stands for light detection and ranging. It is used for localization, mapping, and obstacle avoidance [3]. LIDAR sensors send out infrared light and measure the time it takes for the light to bounce off an object and back to the sensor, thus creating a three-dimensional map of the surrounding. LIDAR in other words is the eyes of the car which enables it to see even in darkness. Today we have LIDARS with a 360-degree range and the ability to detect up to a distance of 400m, but the only drawback is that LIDAR systems are too costly making the vehicles much expensive for the market.

Cameras: Cameras are used for object recognition and detection tasks, with the recent advancements in computer vision and artificial intelligence cameras, are now able to perform complex functions like lane detection, traffic light detection, environment, and obstacle detection. Most designs have eight high-resolution cameras mounted on top of the car to provide a 360-degree view and maximum accuracy, these cameras generate around 2GB of raw data per second requiring immense computing power to process it in real-time.

Radar and Sonar: These are used in the final line of defense in the autonomous system. The functionality of these is to detect any object to show the distance of the nearest object in front of the vehicle using sound waves. If the sensor finds any object in the close vicinity of the vehicle that could lead to a collision, it directly contacts with the braking system to start breaking action and with the airbags and seatbelts system to tighten the belts and prepare in case of collision. Once these sensors generate a collision alert instantly systems start response action without any processing of these alerts. One of the advantages is that Radar can work in extreme weather conditions and even at the night, the disadvantage is that it cannot detect small objects and also fail to provide a precise image of the object.

3. TESLA'S FULL SELF DRIVING CHIP (FSD)

While designing their self-driving cars, Tesla Identified many key challenges that general-purpose chips available were unable to overcome [5]. These challenges are:

1. The Chip had to be power efficient and should consume less than 100w.
2. Could be able to fit inside its existing AP1 (AutoPilot 1) and AP2 (AutoPilot 2) vehicles available in the market.
3. Chip should have a fault tolerance mechanism in case of failures, and that too should be provided keeping cost low.
4. Minimum input batch job.
5. GPU for post-processing.
6. Increased security.
7. A neural network performance of at least 50 TOPs.

As none of the chips available in the market were able to overcome these challenges, Tesla felt the need to design a chip of its own from ground zero keeping in mind all the requirements and challenges that an autonomous car system would face. In 2019 Tesla came up with its chip called Tesla FSD Computer [4]. It is a 14-nanometer FinFET Cmos Processor that measures 260 millimeters squared and has over 6 billion transistors. Tesla decided to use a 14nm chip instead of the latest 10nm chip to keep the cost of the chip low. It consists of twelve 64-bit ARM cores organized as three quad-core clusters. The RAM on-chip is LPDDR4 RAM that boasts a peak bandwidth of 68 GB/s. Tesla's chips although they were much more powerful than previously used chips but consume only 25 percent of more power, keeping its power usage within 80 w. The chip can process images at 2300 frames per second versus the old hardware that could process at only 110 frames per second. It runs at an amazing speed of 75 TOPS making it seven times faster than the NVIDIA Xavier system which could run at 21 TOPS. For increasing the security, it uses software that is cryptographically signed by Tesla, another software verifies once in an interval if that Autonomous driving system software signature on software matches with Tesla's signature to ensure that software is not hacked. For fault-tolerance, the FSD computer has two computers running and can immediately shift over to another if one fails. The instruction set architecture consists of eight instructions - two direct memory access (DMA) read and write, three-dot product operations, one scaling operation, one element-wise addition, and one-stop to halt processing instruction. The instruction is divided into four slots and its size range varies from 32B to 256B. The processing of images using a Neural processing unit (NPU) happens in three steps. In step one, a data buffer is used which can hold up to 256 bytes of data and 128 bytes of weight, then in the second step, convolution.

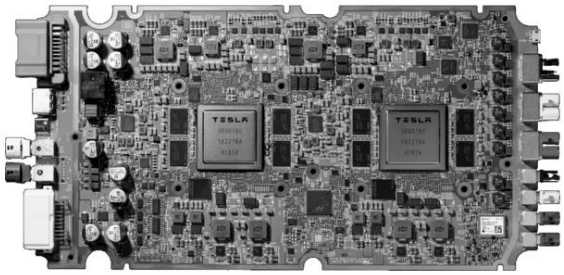


Figure 1. FSD Computer with two Tesla FSD chips in dual configurations

For complete redundancy, the board has two totally independent FSD chips, each with their own power subsystem, DRAM, and flash memory. Each chip runs its own operating system and starts up from its own storage memory. The eight camera connectors are located on the right side of the board (as seen below). The power supply and controls are located on the board's left side. The board is powered by two separate power supply, one for each of the two FSD chips. Furthermore, half of the cameras are powered by one power source, while the other half is powered by the second power supply (note that the video input itself is received by both chips). The redundancy is intended to ensure that in the event of a failure, the system will continue to function.

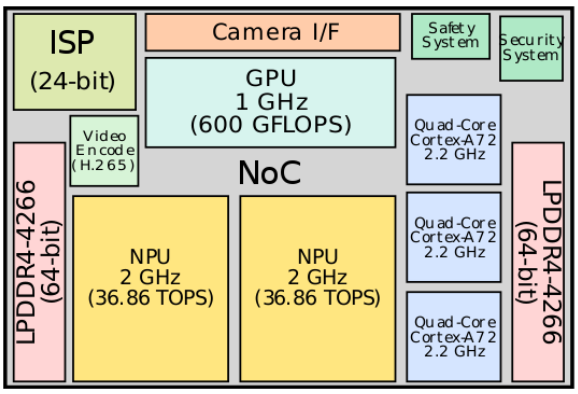


Figure 2. Block Diagram of a FSD chip.

4. SAFETY ISSUES IN AUTONOMOUS VEHICLES

According to the different types of errors in autonomous systems which could lead to an accident, the errors can be divided into three types[6]:

4.1 Perception Error

The perception layer is responsible for collecting data from multiple sensing devices to process it and make sense of it for understanding the environment which will help the next layer in real-time decision making. Perception error can be further classified into three types - hardware, software, and communication. Hardware technology includes many sensors. Since the perception highly relies on different sensors, any failure of these sensors may cause perception error, confuse the decision system, and might lead to dangerous driving decisions further leading to accidents. Therefore, it is very important to make the sensing technology reliable and fault tolerant. Second type of issues are software issues, any malfunctioning of software will lead to incorrect data processing leading to dangerous driving decisions. The the third type of perception error is communication, every ADS receives communication from its sensors, the cloud server, other autonomous vehicles, road signals, if this communication is disrupted, the system won't receive the necessary information for making decisions.

4.2 Decision Error

The decision layer interprets all data received from the perception layer, runs artificial intelligence and machine learning data processing algorithms, and make decisions. It then sends this decision to the action layer. If the layer is unable to make suitable decisions, it should be smart enough to warn the human driver in advance to take control of the system. The design of this layer should be to maximize the true positive warnings and minimize the false negatives because a false alarm for humans to take control won't do any harm, but no alarm can certainly lead to accidents.

4.3 Action Error

After receiving the command from the decision layer, it is now the responsibility of the action layer to control the steering wheel, throttle, or brake for a traditional engine to change the direction and accelerate or decelerate. Also, it monitors the result of its actions to feedback them to the decision layer to impact further decisions. Similar to traditional driving systems, action errors due to the failure of the actuators or the malfunction of the powertrain, controlling system, heat management system, or exhaust system may rise to safety problems. However, a human driver would be able to identify this type of safety issue during the driving and pull over within a short response time. How the vehicle learns in these scenarios and responds to these low-frequency but fatal malfunctions of major vehicular components would be challenging to the full automation driving system.

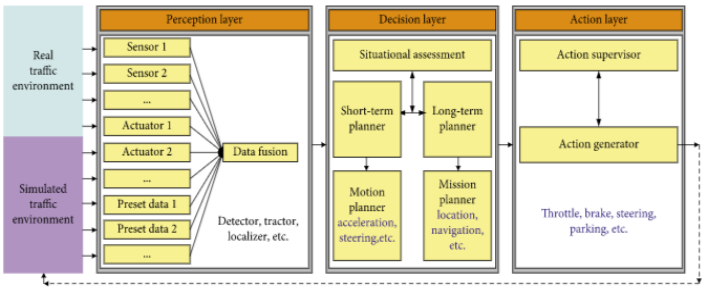


Figure 3. A typical autonomous vehicle system architecture

5. FAIL-OPERATIONAL DESIGN

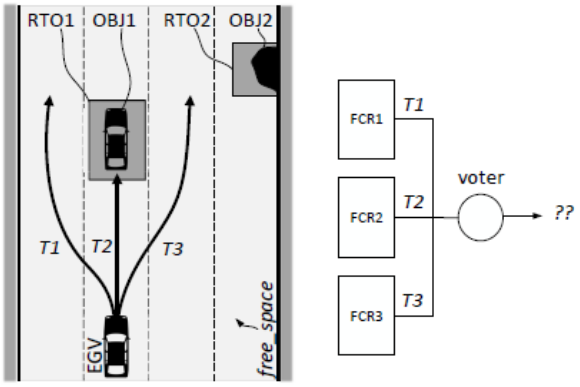
As the Autonomous driving systems have responsibility for making decisions that has a direct impact on human lives, it becomes much important to have a solid fault tolerance mechanism inside these systems, two majorly used fault tolerance mechanism are triple modular redundancy TMR and commander/monitor design changes also called Doer/Checker approach[7].

5.1 Triple Modular Redundancy (TMR) ADS

One of the ways to provide fault tolerance to the system to replicate the ADS three times such that each ADS will serve as an independent fault containment region (FCR). If a single failure occurs in the system, only one ADS would become faulty, the rest two will still function correctly and give the same output and could be used for making driving decisions. The two main challenges in designing an ADS are: (1) maintaining the independence of different FCRs, (2) maintaining consistency of correct outputs. FCRs must coordinate their activity in some way to establish consistency in their outputs. The FCRs may, for example, need to agree on their sensor inputs. Furthermore, because the ADS will be sophisticated

hardware/software entities, additional coordination points, such as to coordinate the input and output of their respective perception, sensor fusion, and/or trajectory planning phases, are likely to be required. However, the more cooperation that is put in place, the less independence that can be claimed.

On the other side, the less the coordination between the FCRs is implemented, the greater the gap between the FCRs' outputs, and, as a result, consistency becomes increasingly erratic. Worse, because a vehicle can safely go along many alternative trajectories, it's possible that several correct FCRs will yield correct but very different trajectories in the absence of adequate coordination. Thus, implementation of TMR in ADS will have many issues and some alternative needs to be thought about.



(a) T1 and T3 safe but different (b) TMR block diagram

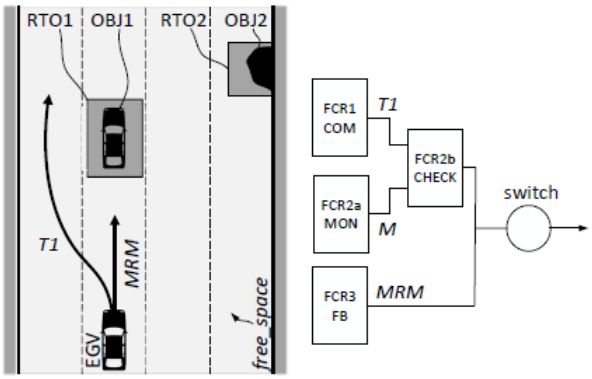
Figure 4. Dependability design with TMR

5.2 Commander/Monitor-based ADS with fallback.

The Commander/Monitor (Com/Mon) approach for fault tolerance is an asymmetric approach. In this one ADS FCR is implemented as the commander and is in charge of creating vehicle trajectories. In addition, a monitor checks if the commander's trajectories are safe. If the monitor determines that the commander's output is hazardous, the vehicle switches to a fallback system's output. The vehicle may be instructed to perform a low-risk move as a fallback (MRM). It may be advantageous to build the monitor using more than one FCR.

The Com/Mon strategy, like the TMR discussion, has a similar problem: competing needs that force a trade-off between i) commander independence from the monitor and (ii) monitor false positive and false negative detections.

The monitor technique, on the other hand, avoids the issue of merging numerous safe but considerably differing trajectories (as the scenario described above in the TMR setting). In addition, as a baseline for monitor development, a safety standard for trajectories must be created. A safety specification like this would be best developed through open collaboration, such as through a standardization committee.



(a) Com Trajectory and MRM (b) Com/Mon block diagram

Figure 5. Dependability design with Com/Mon Approach

6. MOVING TO VISION BASED SYSTEMS FROM LIDAR BASED SYSTEMS

The latest LIDAR available today are one the best way to capture the environment and detect objects, with features like 360-degree vision and 200 m range, but these systems can cost up to tens of thousands of dollars making on road price of the car unaffordable for most of the buyers. Thus, industry giant Tesla have moved into systems that uses vision-based autonomous driving [8]. Lidar captures the environment at a much higher detail than a vision-based camera system alone can. The higher detail captured from the environment is then fused with other sensors (radar, vision, etc) to create an HD map and to train a machine learning model that drives the car [9]. Whereas, the machine learning model requires a lot of training data. This means that you either deploy a very large fleet of vehicles equipped with lidars to drive through all the different cities, under different driving conditions (which is too expensive to realistically do), or use a lot of simulated driving to training machine learning models. On the good side, Vision-based systems use CMOS-based cameras you find in your smartphones. They are cheap because they are much lower costs, companies can deploy at a massive scale, even in the middle range budget vehicles. The advantage of this Massive deployment will be that even though the surrounding data generated is of lower resolution than that of lidar, a vision-based system can overcome the resolution limitation by training its neural network with a much larger data set. Also, because the system cost is much lower, it can be deployed in shadow mode in production cars today. In shadow mode, the system records the driving data, lets the human drive, and compares with what the machine learning model would do. This transforms the problem into a supervised

learning problem, where the user pays for the privilege of acquiring the car and helps train the system to be better. Tesla has millions of cars globally in training its machine learning model. With the current dataset, they have collected Tesla's CEO Elon Musk has shown confidence that a Vision-based system is a future and Tesla will achieve fifth-level autonomy using the same. Also, Tesla has removed LIDAR and RADAR from all of its systems since May 2021 production. However, most of the other automobile companies have raised major safety concerns on removing LIDAR and depending just on machine learning models to process images and take decisions.

7. CONCLUSION

Automation by use of an intelligent machine is not a new concept in this generation. But when machines become responsible for the safety of lives of many people and operate in such a dynamic and unpredictable real-time environment, it brings many challenges with it. To reach the level of automation where the human can relax and machine drives the car, several challenges like - reliability, safety, cost, power, and heat consumption have to be taken into consideration. It is certainly a long way to go, but with the pace, companies are moving towards this direction, the long way will be covered in a short time. Different automobile companies are making and claiming the excellence of their product but unless its vehicle hits the roads and trust is established, the race of designing the best autonomous vehicle will continue.

.

8. REFERENCES

[1] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "Computer Architectures for Autonomous Driving," *Computer* , vol. 50, no. 8, pp. 18–25, 2017, doi: 10.1109/mc.2017.3001256.

[2] Policy on Automated Vehicles, NHTSA, http://www.nhtsa.gov/staticfiles/rulemaking/ pdf/Automated_Vehicles_Policy.pdf

[3] M.-G. Cho, "A Study on the Obstacle Recognition for Autonomous Driving RC Car Using LiDAR and Thermal Infrared Camera," in *2019 Eleventh International Conference on Ubiquitous and Future Networks (ICUFN)*, Jul. 2019, pp. 544–546. doi: 10.1109/ICUFN.2019.8806152.

[4] E. Talpes *et al.*, "Compute Solution for Tesla's Full Self-Driving Computer," *IEEE Micro*, vol. 40, no. 2, pp. 25–35, Mar. 2020, doi: 10.1109/MM.2020.2975764.

[5] D. Schor, "Inside Tesla's Neural Processor In The FSD Chip," Sep. 22, 2019. https://fuse.wikichip.org/news/2707/inside-teslas-neural-processor-in-the-fsd-chip/ (accessed Nov. 06, 2021).

[6] J. Wang, L. Zhang, Y. Huang, and J. Zhao, "Safety of Autonomous Vehicles," *Journal of Advanced Transportation*, vol. 2020, Oct. 2020, doi: 10.1155/2020/8867757.

[7] A. Mehmed, M. Antlanger, and W. Steiner, "The Monitor as Key Architecture Element for Safe Self-Driving Cars," in *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, Jun. 2020, pp. 9–12. doi: 10.1109/DSN-S50200.2020.00015.

[8] É. Zablocki, H. Ben-Younes, P. Pérez, and M. Cord, "Explainability of vision-based autonomous driving systems: Review and challenges," *arXiv [cs.CV]*, Jan. 13, 2021. [Online]. Available: http://arxiv.org/abs/2101.05307

[9] "Self-Driving technology: Lidar vs camera - Vested Finance," Jul. 26, 2021. https://vested.co.in/blog/self-driving-technology-lidar-vs-camera/ (accessed Nov. 06, 2021)

S - Nvidia's Turing Architecture – Changing the Game with Ray Tracing

Saurabh Kale
Computer Science Department
San Jose State University
San Jose, CA 95192
408-924-1000
saurabh.kale@sjsu.edu

ABSTRACT
The Turing GPU, NVIDIA's most recent chip family, was created to realize a vision for next-generation graphics that combined rasterization, ray tracing, and deep learning. It has significant improvements in numerous critical areas, including streaming multiprocessor efficiency, a Tensor Core for rapid AI inference, and an RT Core for accelerated ray tracing. Turing enables real-time ray-tracing performance and deep-learning inference in consumer, professional, and data-centre solutions with these advances.

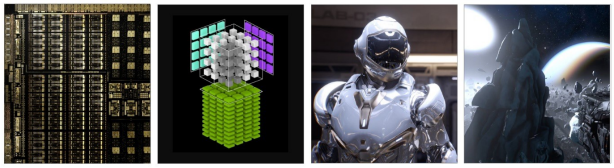
1. INTRODUCTION
Turing is the most significant architectural advancement in over a decade, with a new core GPU architecture that offers significant improvements in efficiency and performance for PC gaming, professional graphics applications, and deep learning inference. Turing fuses rasterization, real-time ray tracing, AI, and simulation to enable incredible realism in PC games, amazing new effects powered by neural networks, cinematic-quality interactive experiences, and fluid interactivity when creating or navigating complex 3D models using new hardware-based accelerators and a Hybrid Rendering approach.

A new GPU processor (streaming multiprocessor—SM) architecture with better shader execution efficiency, as well as a new memory system architecture with support for the newest GDDR6 memory technology, are essential enablers for Turing's considerable boost in graphics performance.

Image processing applications like the ImageNet Challenge were among the earliest deep learning success stories, so it's no surprise that AI has the ability to handle a wide range of graphics-related challenges. Turing's Tensor Cores are at the heart of a new set of deep learning-based Neural Services that provide outstanding graphical effects for games and professional graphics, as well as fast AI inferencing for cloud-based systems.

Microsoft introduced the DirectML for AI and DirectX Raytracing (DXR) APIs in early 2018 in tandem with Turing's development. Game creators can quickly implement real-time AI and ray tracing in their games using the Turing GPU architecture and Microsoft's new AI and ray tracing APIs.
Turing incorporates several new advanced shading techniques that boost efficiency, improve image quality, and deliver new levels of geometric complexity, in addition to its ground-breaking AI and ray tracing features.
Turing GPUs also inherit all of the Volta architecture's advancements to the NVIDIA CUDATM platform, which boost compute applications' capability, flexibility, productivity, and portability. The Turing GPU architecture includes features like autonomous thread scheduling, hardware-accelerated Multi Process Service (MPS) with address space segregation for many applications, and Cooperative Groups.

Turing GPUs will be used in a number of new NVIDIA GeForce® and NVIDIA QuadroTM GPU models. The architecture and capabilities of NVIDIA's flagship Turing GPU, codenamed TU102, will be featured in the GeForce RTX 2080 Ti and Quadro RTX 6000. The appendices provide technical information, including product specs for the TU104 and TU106 Turing GPUs.



Turing reinvents graphics with a totally new architecture that incorporates improved Tensor Cores, new RT Cores, and a slew of new advanced shading features (see figure). Turing blends programmable shading, real-time ray tracing, and AI algorithms to create graphics that are extraordinarily realistic and physically correct for games and commercial applications.

2. Turing GPU Features
NVIDIA Turing is the most advanced GPU architecture in the world. The 18.6 billion transistors in the high-end TU102 GPU were built on TSMC's 12 nm FFN (FinFET NVIDIA) high-performance manufacturing process.

The GeForce RTX 2080 Ti Founders Edition GPU has the following outstanding processing capabilities:

1. 14.2 TFLOPS1 of peak single precision (FP32) performance
2. 28.5 TFLOPS1 of peak half precision (FP16) performance

3. 14.2 TIPS1 concurrent with FP, through independent integer execution units
4. 113.8 Tensor TFLOPS1,2
5. 10 Giga Rays/sec
6. 78 Tera RTX-OPS3

The Quadro RTX 6000 is built for professional operations and has greater computational performance:

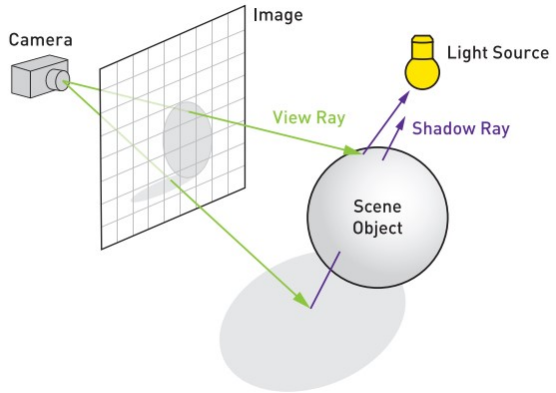
1. 16.3 TFLOPS1 of peak single precision (FP32) performance
2. 32.6 TFLOPS1 of peak half precision (FP16) performance
3. 16.3 TIPS1 concurrent with FP, through independent integer execution units
4. 130.5 Tensor TFLOPS1,2
5. 10 Giga Rays/sec
6. 84 Tera RTX-OPS3

3. Ray Tracing

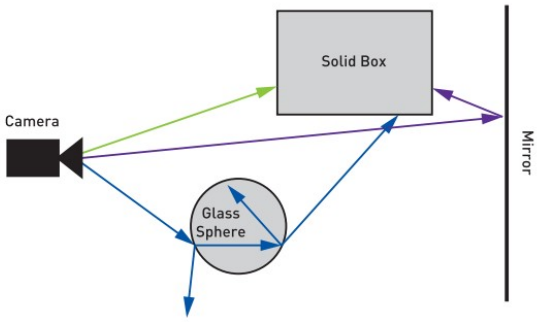
The practice of identifying the closest, or occasionally just any, object along a ray is known as ray casting. A ray exits the camera via a pixel and travels until it collides with the nearest object. A new ray could be cast toward a light source as part of shading this hit point to determine if the item is shaded.

Ray tracing has a long history in fields that study the flow of light in a given environment, often known as radiative transfer. Ideas from domains like neutron transport, heat transfer, and illumination engineering have been incorporated into the graphics industry. Because these concepts have been examined by so many disciplines, nomenclature has evolved and occasionally diverged within and between disciplines. Classic papers may then appear to use terms incorrectly, which can be confusing.

The SI unit spectral radiance is the fundamental quantity of light travelling along a ray, which remains constant along a ray (in a vacuum) and frequently behaves intuitively like the perceptual term brightness. Prior to its standardization, spectral radiance was commonly referred to as "intensity" or "brightness." Non-spectral radiance, a bulk quantity over all wavelengths, is never utilized in computer graphics, thus we normally delete "spectral."



Ray tracing recursively gathers light contributions from reflecting and refractive objects using the ray casting technique. When a mirror is encountered, for example, a ray is cast in the reflection direction from a hit point on the mirror. The final shading of the mirror is affected by wherever this reflection ray crosses. Transparent or glass objects can also produce both reflection and refraction rays. This cycle repeats itself, with each new ray potentially producing other reflection and refraction rays. A cutoff restriction, such as a maximum number of bounces, is generally applied to recursion. As before, each intersection point can be queried whether it is shadowed by casting a ray toward each light source.



Three rays pass from the camera into the scene, as shown in the diagram. The top green ray slams into a box. The purple ray in the middle collides with a mirror and reflects, picking up the back of the box. When the bottom blue ray collides with a glass sphere, reflection and refraction rays are created.

4. Design

4.1 Tensor Core for Accelerated Deep Learning

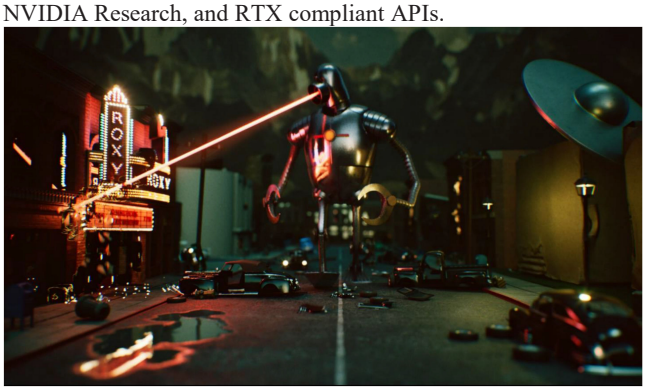
Tensor Cores are specialized execution units that are meant to conduct tensor / matrix operations, which are the primary compute function in Deep Learning. Turing Tensor Cores, like Volta Tensor Cores, provide massive speedups for matrix calculations, which are at the basis of deep learning neural network training and inference. A new version of the Tensor Core design has been added to Turing GPUs, which has been improved for inferencing.

New INT8 and INT4 precision modes for inferencing workloads that can tolerate quantization but don't require FP16 precision have been added to Turing Tensor Cores. Turing Machine For the first time, Tensor Cores offer deep learning-based AI capabilities to GeForce gaming PCs and Quadro-based workstations. Tensor Cores are used to power a novel approach called Deep Learning Super Sampling (DLSS). DLSS uses a deep neural network to extract multidimensional features from a rendered scene and intelligently merge elements from numerous frames to create a high-quality final image. DLSS uses fewer input samples than classic techniques like TAA, while avoiding the algorithmic challenges that traditional techniques like TAA confront when dealing with transparency and other complicated scene aspects.

4.2 RT Core for Accelerated Ray Tracing

The Turing RTX is the first GPU to ship with hardware-accelerated ray tracing thanks to Nvidia's new ray tracing accelerator. The traversal and intersection throughput of the RT Core is up to 10 G Rays/s. Turing RTX has seven times the ray-tracing performance of Pascal, implying that real-time ray tracing

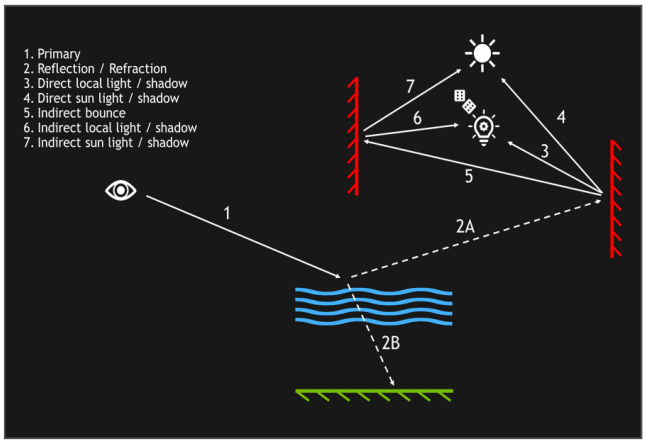
is now achievable. To provide real-time ray tracing on a single Turing GPU, RT Cores work in tandem with improved denoising filtering, a very efficient BVH acceleration structure developed by NVIDIA Research, and RTX compliant APIs.



The image above is from the interactive demo Attack from Outer Space, which was created with Epic Games' Unreal Engine 4 and won the 2019 DXR Spotlight award. Soft shadows beneath the cars on the street, as well as glossy reflections in the robot's breast plate and the puddles on the ground, are among the complex rendering techniques used in the image. Accelerated ray tracing allows them to be calculated dynamically in real time in a destructible, animated environment.

Beyond using rays for shadows and reflections, more complex approaches such as path-traced global illumination, in which rays represent virtual photons in a physical simulation, can be used to create increasingly realistic visuals. The path-tracing procedure is depicted in simplified form in figure below.

A ray, which consists of an origin and a direction, is used to determine what geometry is visible from a certain location in the scene.



4.3 New Streaming Multiprocessor (SM)

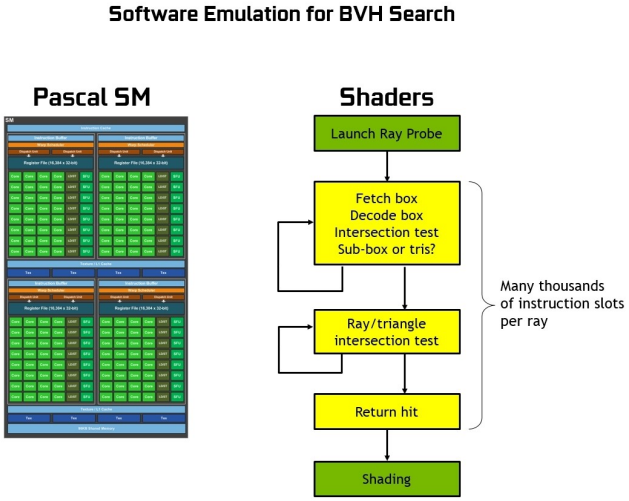
Turing offers the Turing SM, a novel processor architecture that significantly improves shading efficiency, yielding a 50% increase in delivered performance per CUDA Core over the Pascal generation. Two fundamental architecture changes enable these enhancements. The Turing SM, for starters, introduces a new independent integer data-path that can run instructions alongside the floating-point math data-path. Executing these instructions in past generations would have prevented floating-point instructions from being issued. The SM memory path has also been modified to combine shared memory, texture caching, and memory load caching into a single unit. This translates to 2x more bandwidth and more than 2x more capacity available for L1 cache for common workloads.

4.4 Deep Learning Features for Graphics

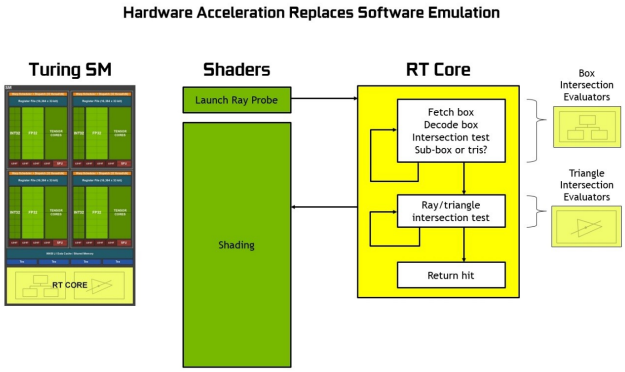
NVIDIA NGXTM is NVIDIA RTX Technology's new deep learning-based neural graphics framework. Deep neural networks (DNNs) and a collection of "Neural Services" are used by NVIDIA NGX to conduct AI-based activities that speed up and improve graphics, rendering, and other client-side applications. For deep learning-based processes, NGX uses Turing Tensor Cores, which speeds up the delivery of NVIDIA deep learning research to the end-user. Ultra-high-quality NGX DLSS (Deep Learning Super-Sampling), content-aware image replacement with AI InPainting, AI Slow-Mo very high-quality and smooth slow motion, and AI Super Rez smart resolution resizing are among the features.

5. Turing Ray Tracing Technology

Ray tracing is a computationally costly rendering technique for simulating the lighting of a scene and its objects in a realistic manner. Real-time rendering of physically realistic reflections, refractions, shadows, and indirect illumination is possible using Turing GPU-based ray tracing technology.



The image above depicts Ray Tracing Pre-Turning, while the image below depicts Turing Ray Tracing with RT Cores.

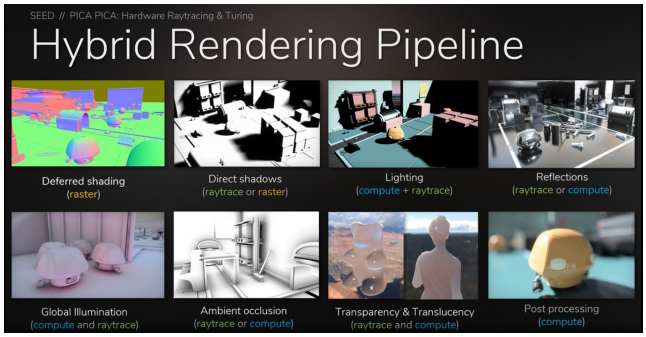


Implementing real-time ray tracing on GPUs was a huge technical challenge that took NVIDIA's research, GPU hardware design, and software engineering teams nearly ten years to complete. Multiple new hardware-based ray tracing acceleration engines called RT Cores in Turing TU102, TU104, and TU106 GPUs, together with NVIDIA RTX software technology, enable real-time ray tracing in games and other applications.
Figure below shows SOL MAN from the NVIDIA SOL ray tracing demo operating in real-time on a Turing TU102 GPU using NVIDIA RTX technology.

As previously stated, rasterization techniques have been the standard in real-time rendering for years, particularly in computer games, and while many rasterized scenes can look fantastic, rasterization-based rendering has severe drawbacks. Rendering reflections and shadows with only rasterization, for example, necessitates simplifying assumptions that can result in a variety of artifacts. Static lightmaps may appear right until something moves, rasterized shadows are prone to aliasing and light leaks, and screen-space reflections can only reflect off visible things on the screen. These artifacts detract from the realism of the gaming experience and are costly for developers and artists to try to fix with additional effects.

Ray tracing is more computationally intensive than rasterization, but it can produce far more realistic visuals. Hybrid rendering, which combines ray tracing and rasterization, has shown to be the most effective method. When rendering reflections, refractions, and shadows, rasterization is utilized where it is most effective, and ray tracing is used when it delivers the largest visual benefit vs rasterization. The hybrid rendering pipeline is depicted in the diagram below.

In the rendering pipeline, Hybrid Rendering blends ray tracing and rasterization techniques to take use of what each does well to render a scene. SEED's PICA PICA real-time ray tracing project, which contains self-learning agents in a procedurally-assembled world, uses a hybrid rendering model. PICA PICA uses Microsoft DXR and NVIDIA GPUs to implement real-time ray tracing. It was built using SEED's R&D engine Halcyon.



While Turing GPUs allow for real-time ray tracing, the amount of main or secondary rays cast per pixel or surface location is dependent on a variety of parameters, including image complexity, resolution, additional visual effects produced in the scene, and, of course, GPU horsepower. In real-time, don't expect hundreds of rays to be thrown each pixel. When Turing RT Core acceleration is used with modern denoising filtering algorithms, substantially fewer rays are required per pixel. The NVIDIA Real-Time Ray Tracing Denoiser modules can minimize the number of rays required per pixel while still delivering great results.

Many scenarios in games and applications can look as realistic as high-end movie special effects, or as excellent as ray-traced images made with professional software-based non-real-time rendering programs, thanks to real-time ray tracing of specified objects. Ray-traced reflections, ray-traced area light shadows, and ray-traced ambient occlusion can all be rendered on a single Quadro RTX 6000 or GeForce RTX 2080 Ti GPU, with rendering quality that is nearly identical to that of movies.

NVIDIA's RTX ray tracing technology, NVIDIA Real-Time Ray Tracing Libraries, NVIDIA OptiX, the Microsoft DXR API, and the soon-to-be-released Vulkan ray tracing API all function with Turing ray tracing hardware. Users will be able to see real-time, cinematic-quality ray-traced objects and characters in games at playable frame rates, as well as visual realism in professional graphics applications that was previously impossible to achieve in real time with previous GPU architectures.

Ray tracing techniques are utilized in many of the following rendering and non-rendering operations, and Turing GPUs can speed them:

1. Reflections and Refractions
2. Shadows and Ambient Occlusion
3. Global Illumination
4. Instant and off-line lightmap baking
5. Beauty shots and high-quality previews
6. Primary rays for foveated VR rendering
7. Occlusion Culling
8. Physics, Collision Detection, Particle simulations
9. Audio simulation (ex., NVIDIA VRWorks Audio built on top of the OptiX API)
10. AI visibility queries
11. In-engine Path Tracing (non-real-time) to generate reference screenshots for tuning real-time rendering techniques and denoisers, material composition, and scene lighting.

6. Turing

Turing is significantly more than the sum of its parts because these three fundamental architectural investments were skilfully integrated. RTX-enabled professional film and design renderers may now perform interactive route tracing with high-quality materials and AI denoising by combining the SM, RT Core, and Tensor Core.

A path-traced image on split screen is shown in the image above. A few randomized pathways per pixel have been ray traced and

materials evaluated on the left. Because of the path tracing algorithm's random sampling, the image appears noisy. Even with fast ray tracing, noise-free images can require hundreds of samples per pixel to converge.

Using the Tensor Cores, an AI-based denoiser6 can predict the completed image with as few as 1% of the typical samples, thereby reducing sampling noise in a postprocess step. To render the final image, the SM, RT Core, and Tensor Core work together effectively. This level of quality and interactive performance in a professional rendering would not be possible with any of these three elements missing.

7. CONCLUSION

We see the future of graphics at NVIDIA, but it requires more efficiency and new breakthrough acceleration. The Turing GPU is designed to realize that vision for next-generation graphics, with a new more efficient SM that is more than 1.5 times faster than the previous generation, new Tensor Cores that unlock real-time deep-learning inference, and the new RT Core that yields more than 7 times faster ray-tracing performance.

8. REFERENCES

[1] NVIDIA, “NVIDIA Turing GPU architecture: Graphics reinvented,” 2018. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/designvisualization/technologies/turing-architecture/NVIDIATuring-Architecture-Whitepaper.pdf

[2] Conger., NVIDIA, “NVIDIA Tesla V100 GPU,” 2017. [Online]. Available: https://images.nvidia.com/content/voltaarchitecture/pdf/volta-architecture-whitepaper.pdf

[3] NVIDIA, “RTX Technology”, 2019. [Online]. Available: https://developer.nvidia.com/rtx/raytracing

[4] NVIDIA Turing Architecture In-Depth. [Online]. Available: https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/