# CPU/GPU Data Analysis

# Contents

# 1 | Introduction

## 1.1 Description

In this report we present the results from exploratory analysis performed on a CPU and GPU dataset, posted on Kaggle by user 'michaelbryantds' [Link to Dataset]. This dataset contains fundamental information (foundry, release date, product name) as well as technical performance metrics (thermal design point, number of transistors, number of operations per second) on a variety of chips released over the years. We will explain the nature of the data in detail in the following sections.

As we present our results, we will also explain our methods and also include code snippets that will allow the reader to replicate them locally. We encourage curious readers to actually check the source code, publicly available in github [Link to source code].

## 1.2 Objective

Firstly, we will underline the similarities and distinctions, both on technical and fundamental metrics, between CPU's and GPU's. Then we will uncover vendor preferences on foundries and whether that changes based on the type of the product. Further, we will dive deeper into production metrics, examining the number of processors released by year. Finally, we will put Moore's Law to test to see if it still holds to this day.

## 1.3 Results

No results to report yet

## 1.4 Environment Setup

We will walk-through our environment setup process. Readers who are strictly interested in the results should skip this part of the report.

We use the R programming language to perform analysis. Further, a list of libraries that will be used is presented:

```
> library(tidyverse)
> library(dplyr)
> library(ggplot2)
> library(zoo)
```

Before pulling the dataset, we fetch the project directory, set in the ' /.Renvironment' file:

```
project_dir <- Sys.getenv("R_CPU_GPU_DIR")
setwd(project_dir)
```

After having imported the downloaded dataset as shown below, we are ready to get to work.

```r
> cpu_gpu_data <- read_csv('chip_dataset.csv')
```

# 2 | Understanding the Data

## 2.1 Specifics

The dataset contains $4,854$ examples and $14$ columns.

```
> str(cpu_gpu_data)
spec_tbl_df [4,854 × 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ ...1                 : num [1:4854] 0 1 2 3 4 5 6 7 8 9 ...
 $ Product              : chr [1:4854] "AMD Athlon 64 3500+" "AMD Athlon 200GE" ...
 $ Type                 : chr [1:4854] "CPU" "CPU" "CPU" "CPU" ...
 $ Release Date         : chr [1:4854] "2007-02-20" "2018-09-06" "2020-09-02" "2013-09-01"...
 $ Process Size (nm)    : num [1:4854] 65 14 10 22 45 22 65 65 10 90 ...
 $ TDP (W)              : num [1:4854] 45 35 28 80 125 95 125 130 28 89 ...
 $ Die Size (mm^2)      : num [1:4854] 77 192 NA 160 258 160 285 140 NA 156 ...
 $ Transistors (million): num [1:4854] 122 4800 NA 1400 758 1400 450 376 NA 154 ...
 $ Freq (MHz)           : num [1:4854] 2200 3200 2600 1800 3700 2400 2400 3000 2000 2200 ...
 $ Foundry              : chr [1:4854] "Unknown" "Unknown" "Intel" "Intel" ...
 $ Vendor               : chr [1:4854] "AMD" "AMD" "Intel" "Intel" ...
 $ FP16 GFLOPS          : num [1:4854] NA NA NA NA NA NA NA NA NA NA ...
 $ FP32 GFLOPS          : num [1:4854] NA NA NA NA NA NA NA NA NA NA ...
 $ FP64 GFLOPS          : num [1:4854] NA NA NA NA NA NA NA NA NA NA ...
```

Before proceeding, we will clarify what each of the columns are referring to in our dataset:

- **'...1'**: Index column, which is dropped as it is superfluous.
- **'Product'**: The product name. A few examples are: "Intel Atom E620", "AMD Ryzen 7 4800U", etc.
- **'Type'**: Type of processor: 'CPU' or 'GPU'.
- **'Release-Date'**: Release date, in format 'YYYY-MM-DD'. With range of years being 2000-2021.
- **'Process Size (nm)'**: Process size is the size of the smallest component, in nanometers.
- **'TDP (W)'**: TDP is short for Thermal Design Power, which is the theoretical max heat the GPU/CPU can withstand, in watts.
- **'Die Size (mm2)'**: The die size refers to the size of the unpackaged chip, in millimeters squared.
- **'Transistors (million)'**: The number of transistors in the processor, in millions.
- **'Freq (MHz)'**: The frequency of operations the chip can perform, in MegaHertz.
- **'Foundry'**: The factory where the chip was manufactured, such as "TSMC", "Intel", etc.

- **'Vendor'**: The vendor of the chip: "Intel", "AMD", "NVIDIA", "ATC" or "Unknown".
- **'FP16 GFLOPS'**: Number of 16-bit precision additions and multiplications the CPU/GPU can perform in a second.
- **'FP16 GFLOPS'**: Number of 32-bit precision additions and multiplications the CPU/GPU can perform in a second.
- **'FP16 GFLOPS'**: Number of 64-bit precision additions and multiplications the CPU/GPU can perform in a second.

## 2.2 Pre-Processing

Before working with the data, we need to make some adjustments.

### 2.2.1 Initial Steps

We start by dropping the index column. As mentioned before, we will not be needing this since R has automatic indexing functionality.

```
> cpu_gpu_data <- select(cpu_gpu_data, -c('...1'))
```

Further, we handle special characters such as space or paranthesis in the column name, either by removing or replacing them, which will make the data easier to work with.

```
> names(cpu_gpu_data)[3] = "Release_Date"
> names(cpu_gpu_data)[4] = "Process_Size_nm"
> names(cpu_gpu_data)[5] = "TDP_W"
> names(cpu_gpu_data)[6] = "Die_Size_mm2"
> names(cpu_gpu_data)[7] = "Transistors_million"
> names(cpu_gpu_data)[8] = "Freq_MHz"
> names(cpu_gpu_data)[11] = "FP16_GFLOPS"
> names(cpu_gpu_data)[12] = "FP32_GFLOPS"
> names(cpu_gpu_data)[13] = "FP64_GFLOPS"
```

It seems like many Foundries have less than 100 products in the dataset. As they are insignificant for our purposes, we will simply ignore these by collapsing them into the "Unkown" foundry category.

```
> foundry_table <- table(cpu_gpu_data$Foundry)
> foundry_table

     GF      IBM    Intel      NEC Renesas Samsung     Sony     TSMC      UMC Unknown
    261        3     1385        2        1       50       10     2178       79     866
```

This is a simple task if we use a look-up table as follows:

```
> lut <- c("IBM" = "Unknown", "NEC" = "Unknown", "Renesas" = "Unknown",
+          "Samsung" = "Unknown", "Sony" = "Unknown", "UMC" = "Unknown")
> cpu_gpu_data <- (cpu_gpu_data %>% mutate(Foundry = recode(Foundry, !!!lut)))
> unique(cpu_gpu_data$Foundry)
[1] "Unknown" "Intel"   "GF"      "TSMC"
```

### 2.2.2 Missing Values

We observe that the data contains many missing values:

```
> sum(is.na(cpu_gpu_data))
[1] 12833
```

Here is the number of missing values by column:

```
> sapply(cpu_gpu_data, function(x) sum(is.na(x)))
           Product               Type       Release_Date    Process_Size_nm          TDP_W
                 0                  0                  0                  9            626
       Die_Size_mm^2 Transistors_million          Freq_MHz            Foundry         Vendor
               715                711                  0                  0              0
        FP16_GFLOPS        FP32_GFLOPS        FP64_GFLOPS
              4318               2906               3548
```

Most of the missing values come from GFLOPS columns. If we were to replace these missing values by the mean, this could affect our results greatly. Instead, we replace these values by a special value, $-1$, and handle this case when summarizing the data.

```
> cpu_gpu_data[["FP16_GFLOPS"]][is.na(cpu_gpu_data[["FP16_GFLOPS"]])] <- -1
> cpu_gpu_data[["FP32_GFLOPS"]][is.na(cpu_gpu_data[["FP32_GFLOPS"]])] <- -1
> cpu_gpu_data[["FP64_GFLOPS"]][is.na(cpu_gpu_data[["FP64_GFLOPS"]])] <- -1
```

The remaining NA values, we replace by the column-wise mean. Note that this probably will have a slight effect in reducing the standard deviation and perhaps changing the mode.

```
> cpu_gpu_data <- replace(cpu_gpu_data, TRUE, lapply(cpu_gpu_data, na.aggregate))
```

We also note a few invalid entries, such as "NaT" values for release date (10 examples) as well as 0 process size (4 entries). We can drop these rows as this will have negligible effect on our results.

```
> cpu_gpu_data <- cpu_gpu_data[cpu_gpu_data$Release_Date != "NaT",]
> cpu_gpu_data <- cpu_gpu_data[cpu_gpu_data$Process_Size_nm != 0,]
```

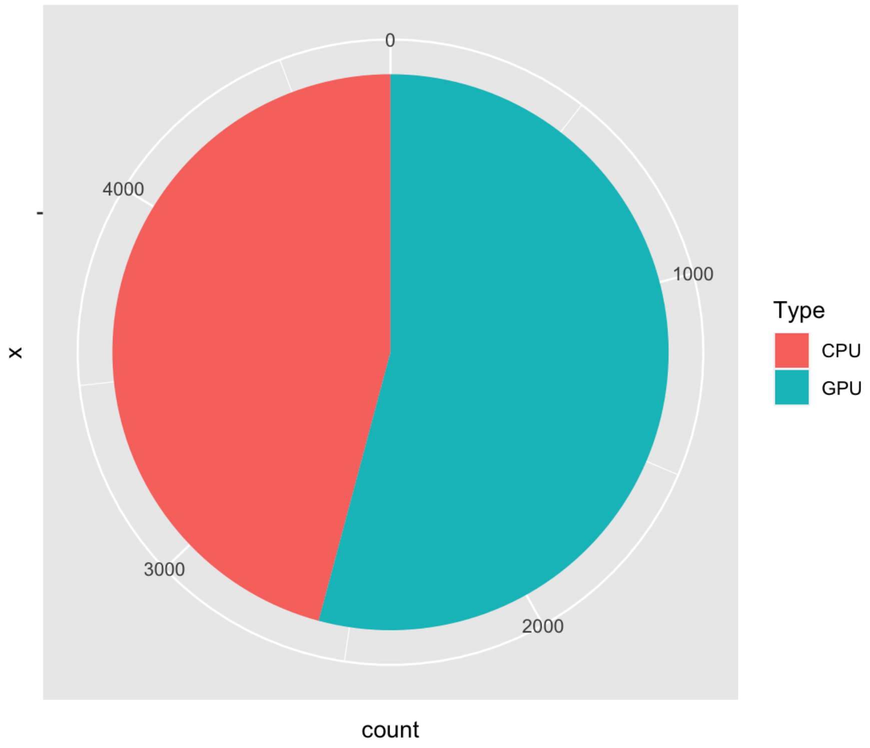

Figure 2.1: Class distribution of chip type after pre-processing.

### 2.2.3 Finishing Touches

We notice that the release date format (YYYY-MM-DD) is hard to work with, we use YYYYMMDD instead and convert the values to integers.

```
> cpu_gpu_data <- cpu_gpu_data %>%
+   mutate(Release_Date = gsub("-", "", cpu_gpu_data$Release_Date))
> cpu_gpu_data$Release_Date <- as.numeric(cpu_gpu_data$Release_Date)
```

Also, as will be discussed in section 4.3 (Vendor Preferences in Foundry), AMD has completed its acquisition of ATI on 25th of October 2006, but kept releasing GPU's under ATI's name until 2010. We will re-name GPU's that fall into this category, in order to provide a more accurate view when comparing these companies.

```
> cpu_gpu_data <- cpu_gpu_data %>%
+   mutate(Vendor = ifelse(Vendor == "ATI" & Release_Date > 20061025, "AMD", Vendor))
```

Finally, we add another column called 'Release Year', which we will use in Chapter 5 in order to examine production trends over time.

```
> cpu_gpu_data$Release_Year <- str_extract(cpu_gpu_data$Release_Date,"^[0-9]{4}")
```

# 3 | GPU versus CPU

In this section, on top of briefly describing what a CPU or GPU is, we will separately **summarize their physical and performance characteristics** as seen in the data. Further, we will compare them with each-other in order to gain a better understanding on the nature of these essential hardware components.

## 3.1 Central Processing Unit (CPU)

The **Central Processing Unit** (CPU), (also called **central processor** or **main processor**) is, in essence, the brain of the computer. It is the electronic circuitry that executes instructions described by **computer programs**. In contrast to **GPU's**, almost every modern electronic device needs a CPU.

## 3.2 Graphical Processing Unit (GPU)

The **Graphics Processing Unit** (GPU) is a hardware component that is responsible of **computer graphics** and **image processing**. As it is very efficient in manipulating large blocks of data in parallel, it has found its way into a wide variety of fields such as machine learning, oil exploration and stock options pricing.

## 3.3 Process Size

Recall that the process size refers to the **size of the smallest individual element** on a processor. Therefore the smaller the process size, the better.

We can look at some key statistics using the following command:

```
> cpu_gpu_data %>% group_by(Type) %>% summarize(mean = mean(Process_Size_nm),
+   sd = sd(Process_Size_nm), median = median(Process_Size_nm), max = max(Process_Size_nm),
+   min = min(Process_Size_nm))
```

Which outputs the following table:

| Type | Mean | Standard Deviation | Median | Max | Min |
|---|---|---|---|---|---|
| CPU | 52 | 42.2 | 32 | 180 | 7 |
| GPU | 57.5 | 46 | 40 | 250 | 7 |

Table 3.1: Table of Process Size (nm) key statistics.

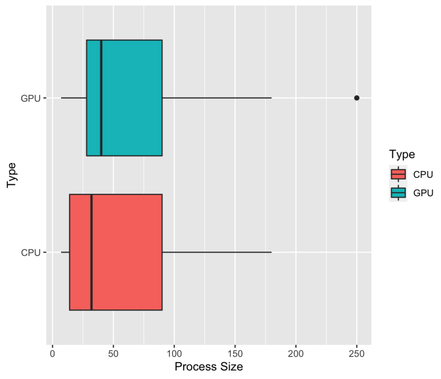

Figure 3.1: Boxplot of Process Size

We see that, **both GPU's and CPU's more or less seem to be centered around the same values**. This is verified by Figure 3.1 generated with the following code:

```
> cpu_gpu_data %>% ggplot(aes(x=Type, y=Process_Size_nm, fill=Type)) +
+   ylab("Process Size") + geom_boxplot() + coord_flip()
```

It is apparent that the center and skew are almost identical, with the CPU's centered slightly lower. We suspect this might be due to components that are shared between the two types.

## 3.4 Thermal Design Power (TDP)

Recall that the TDP of a GPU/CPU is the theoretical maximum heat its cooling system is designed to dissipate, measured in Watts. It follows that the processor which is expected to output a greater amount of computational power should have a higher TDP. Naturally **we would expect the GPU to have a greater TDP**.

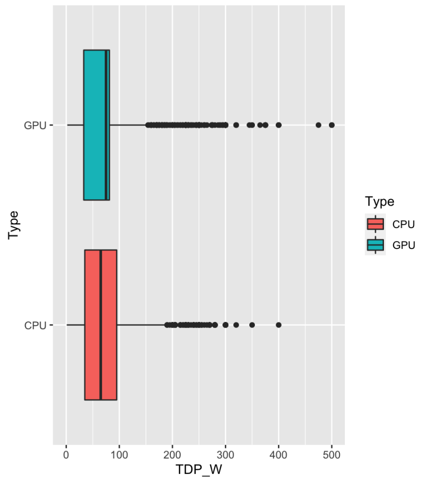

Figure 3.2: Box plot of Thermal Design Power (Red: CPU, Blue: GPU)

| Type | Mean | Standard Deviation | Median | Max | Min |
|---|---|---|---|---|---|
| CPU | 75.3 | 54 | 65 | 400 | 1 |
| GPU | 85.3 | 81.9 | 75 | 900 | 2 |

Table 3.2: Table of TDP (W) key statistics.

Examining Figure and Table 3.2, we see that, **in general GPU TDP's tend to be higher**, with a lot of skewness reflected in the figure in the form of extreme outliers, as well as in the table in the form of high variance. In comparison, the difference in centers seems less drastic: Although the $25\%$ and $75\%$ quartiles are almost the same, the center of GPU TPD's are slightly higher, as seen from the mean.

## 3.5 Die Size

As mentioned before, the die size of a processor is the size of the chip without any packaging. We summarize the data in Table and Figure 3.3

| Type | Mean | Standard Deviation | Median | Max | Min |
|---|---|---|---|---|---|
| CPU | 172 | 70.1 | 188 | 684 | 1 |
| GPU | 203 | 141 | 154 | 826 | 6 |

Table 3.3: Table of Die (mm2) Size key statistics.

The data for **CPU die size seems much more compact**, with little skew. It has a mean of 173, which, as can be seen in Figure 3.3, is surprisingly close to its $75\%$ quartile. On the other hand the data for **GPU die size varies greatly**, with a standard deviation of $141$, two times that of the CPU, which is $70.1$.

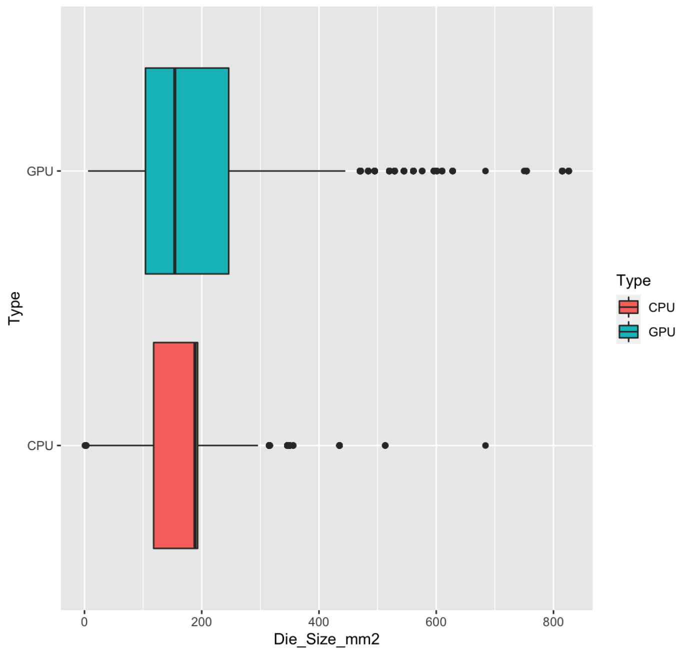

Figure 3.3: Box plot of Die Size (mm2)

## 3.6 TDP / Die Size Correlation

Let us examine the **association between thermal design power and die size**, separately for GPU's and CPU's. We will first look at scatter plots to gain a high level understanding whether such a relationship exists, and then dive into the underlying math to prove or disprove our hunch.

### 3.6.1 Measure of Correlation: Pearson Coefficient

We will use the **Pearson correlation coefficient** in order to determine numerically whether there is an association or not. Given a pair of random variables $X, Y$, the Pearson coefficient $\rho_{X,Y}$ is calculated as follows:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

Where $\sigma_X$ is the **standard deviation** of random variable $X$, calculated as $\sigma_X = \sqrt{\mathbb{E}[X^2] - \mathbb{E}[X]^2}$, and $cov(X,Y)$ is the co-variance of two random variables $X, Y$, given by: $cov(X,Y) = \mathbb{E}[(X - \mu_X)(Y - \mu_Y)]$.

A Pearson coefficient of 1 or higher indicates strong positive correlation, similarly a coefficient of -1 or lower indicates strong negative correlation. Values in-between signal varying degrees of correlation.

### 3.6.2 Correlation in CPU's

First we generate Figure 3.4 using the following block of code:

```
> cpu_gpu_data[cpu_gpu_data$Type == 'CPU',] %>%
+   ggplot(aes(x=TDP_W, y=Die_Size_mm2)) +
+   geom_point(col = "dodgerblue", alpha = .5) +
+   ggtitle("TDP & Die Size (CPU)")
```

Although there doesn't seem to be a strong association between these two characteristics for CPU's, Figure 3.4 suggests some positive correlation.

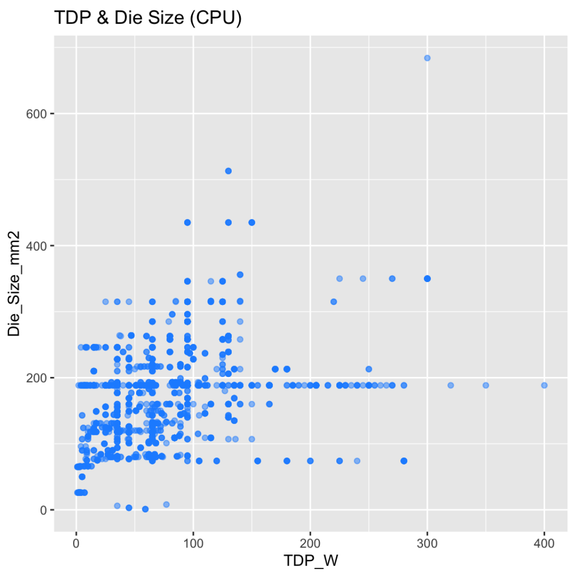

Figure 3.4: TDP vs Die Size for CPU's

Now we can see if the math supports our hypothesis. We use the built-in 'cor' function of R in order to compute the Pearson correlation coefficient:

```
> tmp_cpu <- cpu_gpu_data[cpu_gpu_data$Type == 'CPU',]
> cor(tmp_cpu$TDP_W, tmp_cpu$Die_Size_mm2, method = "pearson")
[1] 0.3298087
```

Indeed, a correlation coefficient of $0.33$ signals **weak positive correlation**.

### 3.6.3 Correlation in GPU's

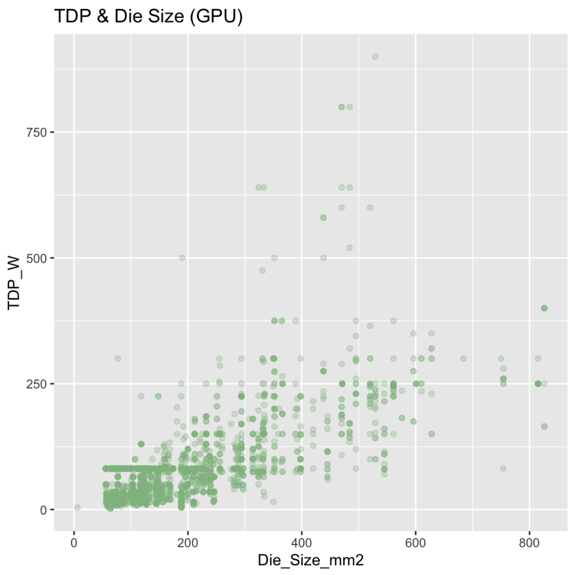

Figure 3.5: TDP vs Die Size for GPU's

Compared to Figure 3.4, Figure 3.5 is closer to a line rather than a shapeless cluster. Although it is hard to assert a strong positive association here, it is certain just by looking at the plot that **die size and thermal design power of GPU's are more strongly associated** compared to CPU's.

```
> tmp_gpu <- cpu_gpu_data[cpu_gpu_data$Type == 'GPU',]
> cor(tmp_gpu$TDP_W, tmp_gpu$Die_Size_mm2, method = "pearson")
[1] 0.6782556
```

A correlation coefficient of $0.67$ **does not indicate strong association**, however it is still **greater than that of the CPU**.

## 3.7 Transistors

Transistors are devices which can amplify or switch electric signals. They are arguably the most fundamental building blocks of modern electronic devices. In fact, there are thousands of millions of transistors in modern processors. Here, we examine the difference in number of transistors between CPU's and GPU's.

| Type | Mean | Standard Deviation | Median | Max | Min |
|---|---|---|---|---|---|
| CPU | 1,338 | 1,812 | 904 | 19,200 | 37 |
| GPU | 2,379 | 4,688 | 930 | 54,200 | 8 |

Table 3.4: Table of Transistors key statistics in millions.

Note that Figure 3.6 (where the values above the $97\%$ quartile has been cut off), shows that the **centers, quartiles and medians are more or less the same** for the two types of processors. However, we see a **drastic difference in mean and standard deviation**, upon examining Table 3.4. This suggests that GPU's have a number of extreme outliers, which pull these statistics higher.

Upon closer examination, we notice that the 19 processors which have more than $25,000$ million transistors have all been released post $2020$:

```
> cpu_gpu_data[cpu_gpu_data$Transistors_million > 25000,]$Release_Date
 [1] 20201028 20210412 20210412 20210412 20201116 20201005 20200514
 [8] 20201028 20200901 20200514 20200514 20210101 20201116 20201028
[15] 20210412 20200622 20200901 20210412 20201005
```

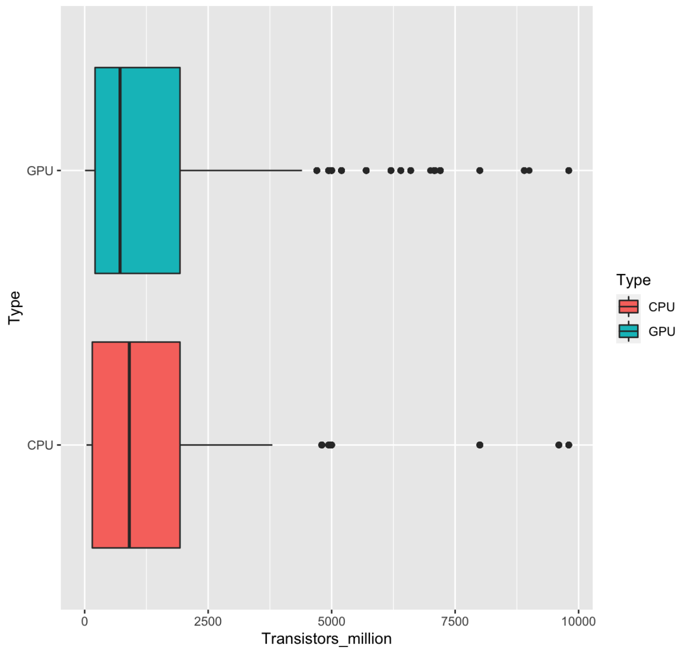

Figure 3.6: Boxplot of Transistor data

Further, we see that these outlier values are all **AMD** and **NVIDIA GPU's**, manufactured in **TSMC and Samsung** foundries, post $2020$.

```
> unique(cpu_gpu_data[cpu_gpu_data$Transistors_million > 25000,]$Type)
[1] "GPU"
> unique(cpu_gpu_data[cpu_gpu_data$Transistors_million > 25000,]$Vendor)
[1] "AMD"    "NVIDIA"
> unique(cpu_gpu_data[cpu_gpu_data$Transistors_million > 25000,]$Foundry)
[1] "TSMC"    "Samsung"
```

This hints at a **technological innovation** dated around $2020$, after which the **number of transistors have exploded**. It is indeed **tempting to define a causality relation** here, however, with the information available in the scope of this report, **asserting whether this change in number of transistors is a cause or an effect of a technological leap would be unfounded**.

## 3.8 Clock Rate (Frequency)

The frequency of a CPU/GPU refers to the the clock rate or clock speed, which is the **frequency at which it can generate pulses**. These pulses are used to **synchronize the operations of its components**. Naturally, this metric is **more important for CPU's** because it needs to 'control' and 'synchronize' many components of the computer system at once.

Indeed, according to the data CPU's on average go through roughly **4 times as many clock cycles compared to a GPU**.

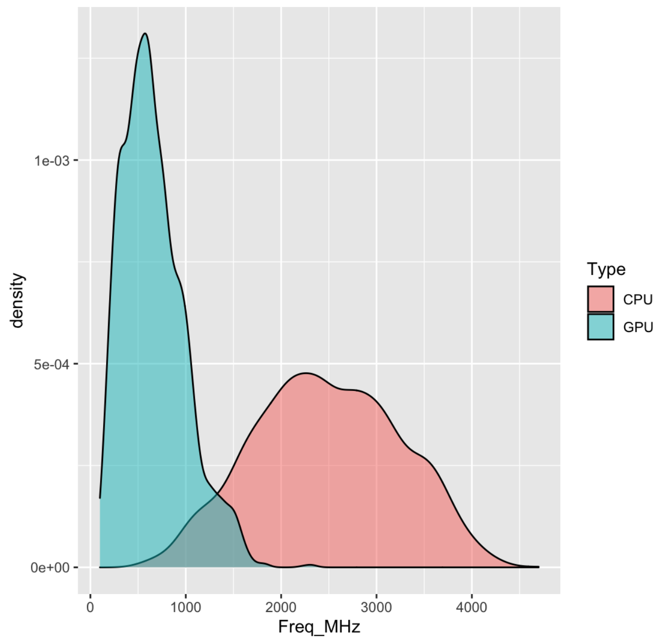

Figure 3.7: Density plot of Clock Rate in MegaHertz

| Type | Mean | Standard Deviation | Median | Max | Min |
|---|---|---|---|---|---|
| CPU | 2,482 | 755 | 2400 | 4,700 | 600 |
| GPU | 662 | 328 | 600 | 2,321 | 100 |

Table 3.5: Table of Transistors key statistics in millions.

## 3.9 Floating Point Operations per Second (GFLOPS)

GFLOPS or gigaFLOPS refers to the **number of floating point operations** a processor can perform in a second, times $10^9$. Measurements are split into different categories depending on floating point precision (16-bit, 32-bit, 64-bit) since they can vary greatly.

Since the dataset does not contain any CPU GFLOPS measurements, we will **not be presenting a comparison**. In fact, even if it did, GPU performance would beat it by a landslide as they are optimized to perform especially well in this metric. Therefore we present a **table of GPU GFLOPS with varying precision**.

| Precision | Mean | Standard Deviation | Median | Max | Min |
|---|---|---|---|---|---|
| 16-bit | 8,249 | 13,871 | 2913 | 184,600 | 10 |
| 32-bit | 2,074 | 3,810 | 691 | 40,000 | 12.8 |
| 64-bit | 365 | 1,161 | 88.6 | 11,540 | 4.8 |

Table 3.6: Table of GPU Giga Floating Point Operations per Second with varying precision

# 4 | Manufacturer and Producer Preferences

In this chapter we will examine manufacturer and producer preferences in the semiconductor industry, specifically **who the players tend to choose to work with and whether that changes based on the type of the product**.

## 4.1 The Processor Market and Its Giants

The Desktop CPU/GPU Market is a perfect example of a **technological oligarchy**, where: a handful of players who have some sort of competitive edge, mostly in the form of **proprietary technology**, utterly dominate. Today, the three big semiconductor giants are: **AMD, NVIDIA** and **Intel**.

## 4.2 Vendors & Foundries

The **vendor** of a product usually exclusively refers to the company which sells it. This exclusivity does not hold in most technology markets. In fact, this term massively under-emphasizes the **role in production of processor vendors**. Each and every one of these companies have their own factories and production line, however most of the time it is simply **not feasible to setup a production cycle for each and every component** of the product. That's where **foundries** come in.

**Foundries**, or **manufacturers** are companies which produce the **intermediary products**, that are used by vendors to put together the final product. The foundry sector follows a similar structure in that a few players dominate, namely **TSMC, Intel and GF**.

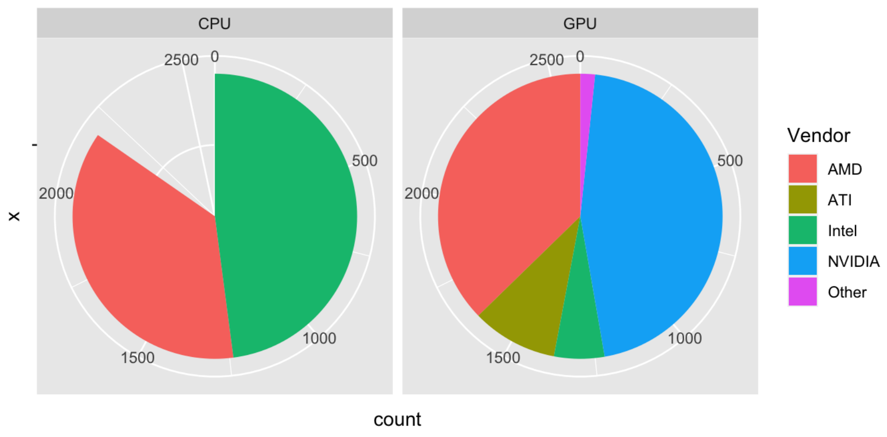

Figure 4.1: CPU/GPU Vendor Class Distribution

## 4.3 Vendor Preferences in Foundry

Let us first visualize the foundry preferences of the semiconductor giants by using the following code. Note that "ATI" and "Other" are excluded from the graphic to provide better clarity.

```
> cpu_gpu_data %>% filter(Vendor != "ATI" & Vendor != "Other") %>%
+   ggplot(aes(x=Foundry, fill=Type)) +
+   geom_bar() + facet_wrap(~Vendor) +
+   theme(axis.text.x = element_text(angle = 90))
```

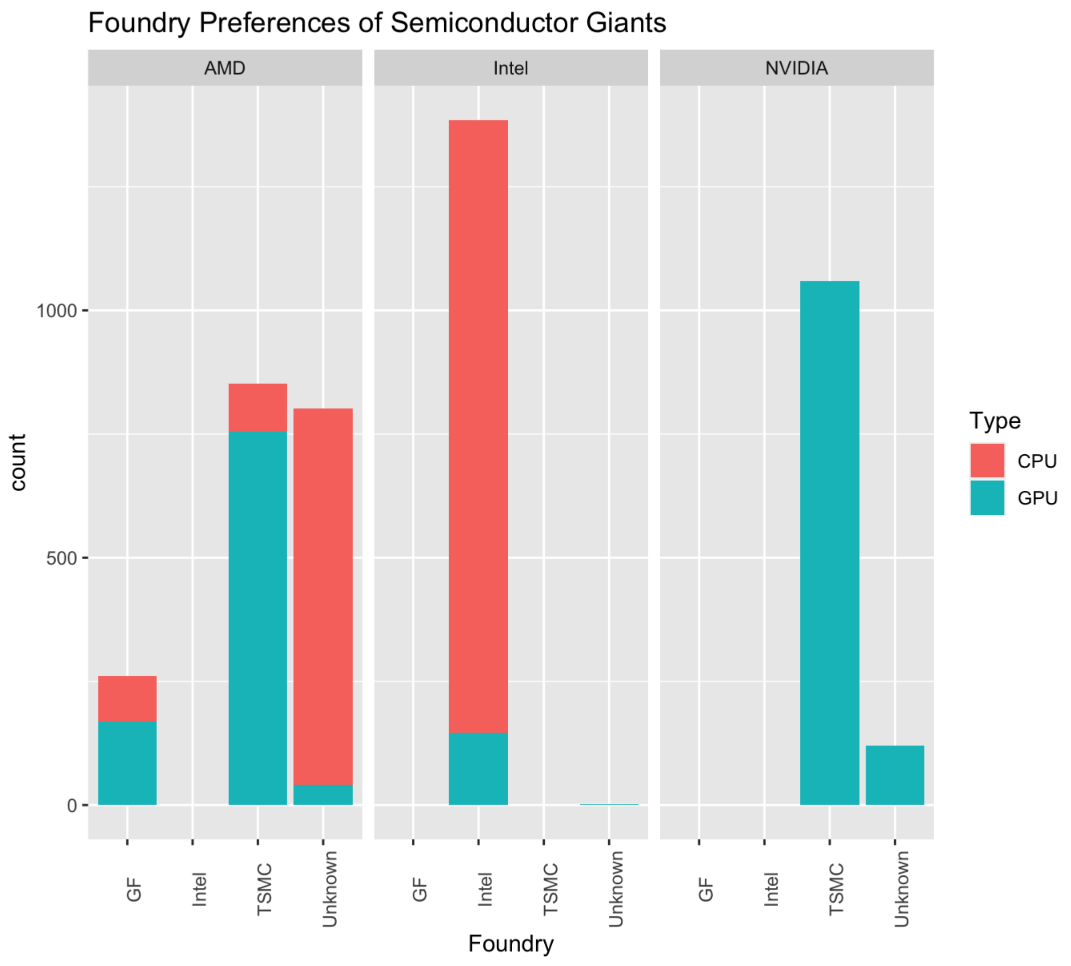

Figure 4.2: Foundry preferences of the largest CPU/GPU Companies

### 4.3.1 NVIDIA

Although it has plans to release state-of-the-art CPU's for data centers, **NVIDIA has almost exclusively focused on producing GPU's** ever since its inception in 1993. Looking at Figure 4.2, it is of course obvious that NVIDIA has preferred the Taiwan Semiconductor Manufacturing Company Ltd (TSMC) as foundry.

| GPU or CPU | Foundry | Number of Examples |
|---|---|---|
| GPU | TSMC | 1,059 |
| GPU | Unknown | 120 |

Table 4.1: NVIDIA Preferences in Foundry

### 4.3.2 Intel

Founded in 1968, Intel (with a lead of one year against AMD) is the **oldest out of the semiconductor giants**, and also the one with the largest market share. We can see from Figure 4.2 that the company has **focused more on CPU production**. One interesting thing to note is that Intel is the only giant that has successfully and consistently **relied on its own foundries for manufacturing parts.**

| GPU or CPU | Foundry | Number of Examples |
|---|---|---|
| CPU | Intel | 1,239 |
| GPU | Intel | 146 |
| GPU | Unknown | 2 |

Table 4.2: Intel Preferences in Foundry

### 4.3.3 AMD

Even though they are more focused on GPU production nowadays, AMD seems to have **produced almost an equal number of GPU's and CPU's** over the years. We can see that they **rely mostly on Unknown foundries for CPU production**, whereas **for GPU production their go-to seems to be TSMC**. It seems like AMD is the **most versatile** out of these three companies.

| GPU or CPU | Foundry | Number of Examples |
|---|---|---|
| CPU | GF | 93 |
| CPU | TSMC | 97 |
| CPU | Unknown | 760 |
| GPU | GF | 168 |
| GPU | TSMC | 755 |
| GPU | Unknown | 41 |

Table 4.3: AMD Preferences in Foundry

### 4.3.4 ATI & Other

Although encompassing most of the market, the three-company list is not exhaustive. Most importantly, **we've left out ATI**, which used to be a semiconductor technology firm focused on producing GPU's. As mentioned before, **the GPU's they've released after their acquisition of AMD in 2006 have been renamed to AMD chips** during the pre-processing stage (Section 2.2.3).

We take a look at the production preferences of the rest of the vendors as well, which **shows a strong preference towards GPU production** and **working with TSMC**.

| Vendor | GPU or CPU | Foundry | Number of Examples |
|---|---|---|---|
| ATI | GPU | TSMC | 204 |
| ATI | GPU | Unknown | 48 |
| Other | GPU | TSMC | 4 |
| Other | GPU | Unknown | 39 |

Table 4.4: Other Companies Preferences in Foundry

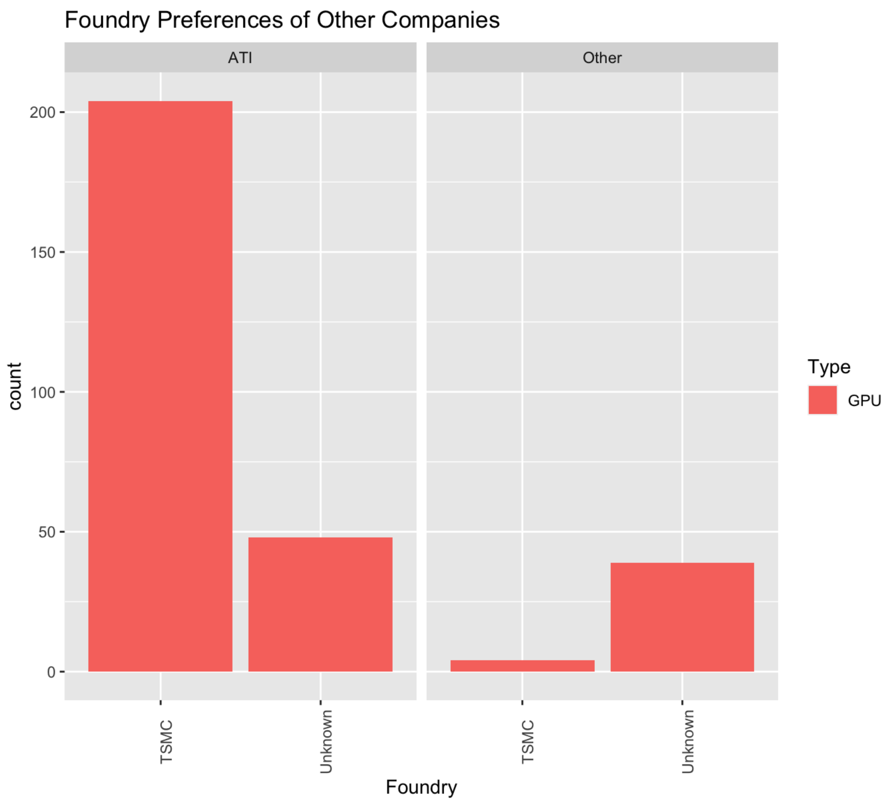

Figure 4.3: Foundry preferences of other CPU/GPU Companies

### 4.3.5 Conclusion

In general, we see that **most Vendors who produce GPU's prefer TSMC**, whereas companies which produce CPU's mostly collaborate with Unknown foundries; with the exception of **Intel which prefers using its own foundries for manufacturing**.

Further, we see **NVIDIA focusing strictly on GPU production**, similar to **Intel which focuses on CPU's**. The notable outlier here is **AMD, which competes with both companies on both markets**.

# 5 | CPU/GPU Production Trends

## 5.1 Introduction

In this chapter we will look at production trends, specifically the total number of processors produced, of Foundries and Vendors.

First we examine the number of processors in total over time:

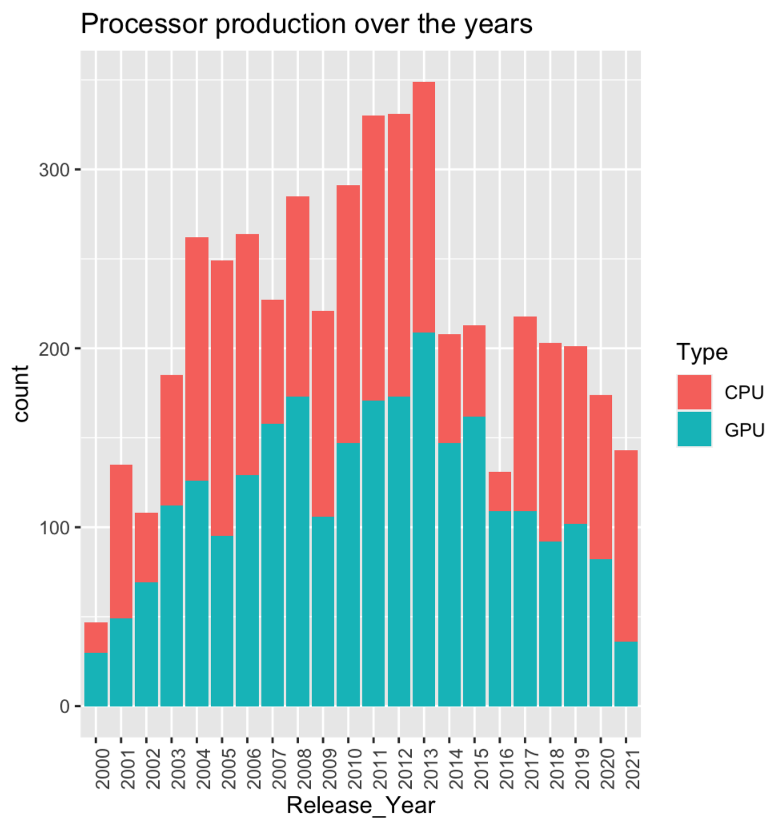

Figure 5.1: Total Processor Production over Time

We see a clear upwards trend until early 2010's at which production peaked. After that point, we see a slight drop followed by an incremental decrease each year.

## 5.2 Foundries

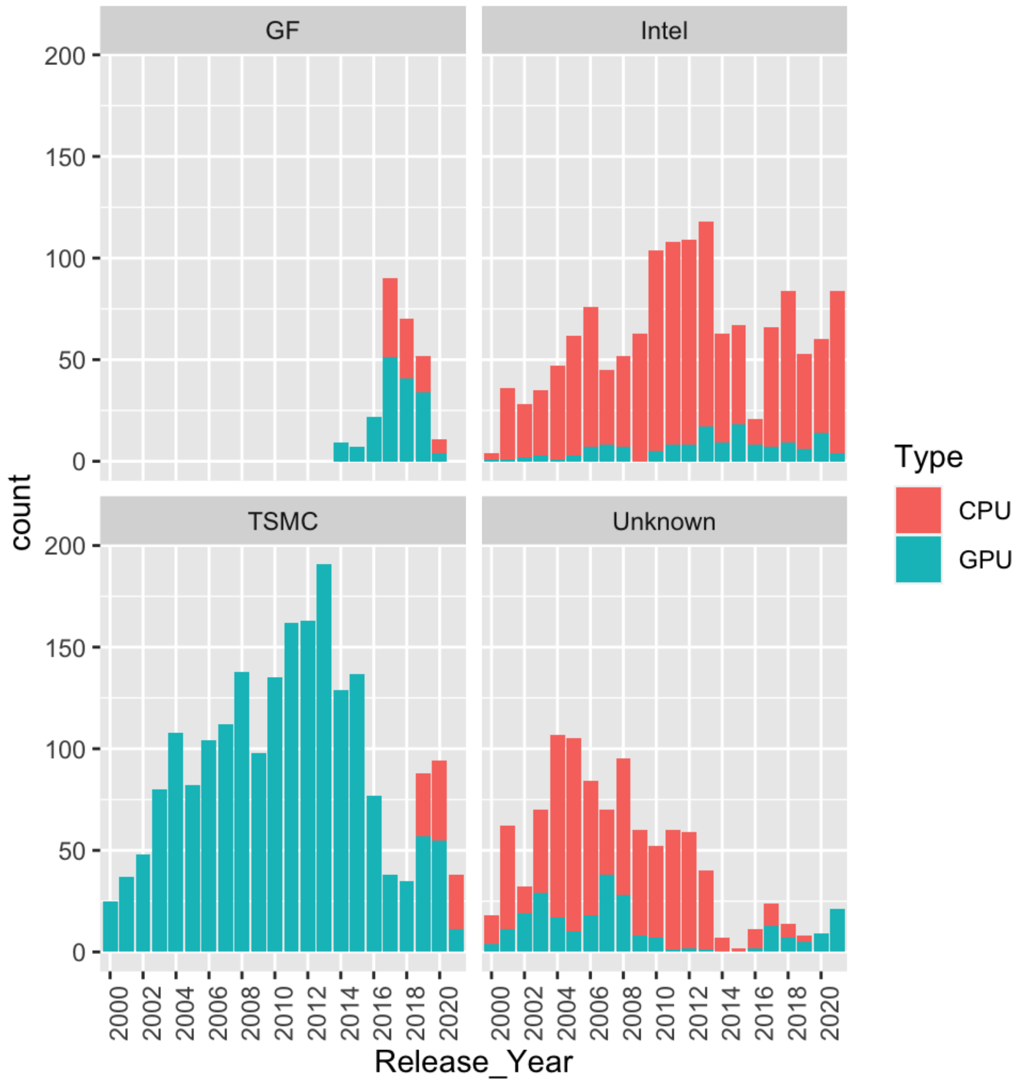

Figure 5.2: Processor Production of Different Foundries over Time

In general, we see **Intel foundries producing more or less 100 processors consistently over the years**, with a focus on CPU's.

| Foundry | count | mean |
|---|---|---|
| GF | 261 | 18.6 |
| Inteal | 1,385 | 98.9 |
| TSMC | 2,119 | 151 |
| Unknown | 1,010 | 72.1 |

Table 5.1: Numerical Summary of Processor Production of Foundries

**TSMC, mirrors the graph of total production** in that we see a spike in early 2010's, followed by a slight decrease. It is somewhat important to note that TSMC is responsible of a large bulk of processor production. Moreover we see **TSMC slowly starting CPU manufacturing as well in the late 2020's.**

Looking at Unkown foundries, we notice that a considerable amount of total production is attributed to them (around 1000 examples), however there seems to be **more and more of a struggle in competition against TSMC and Intel**, signalled through the vastly reduced production in later years. Lastly, we see a **newly emerging foundry (GF), putting a dent into Intel and TSMC production** in late 2010's.

## 5.3 Vendors

Contrary to the Foundries, where almost a third of the production comes from Unknown manufacturers, **the number of processors made by unknown vendors is incredibly small**. In fact, there's a negligible amount of examples post-2002.

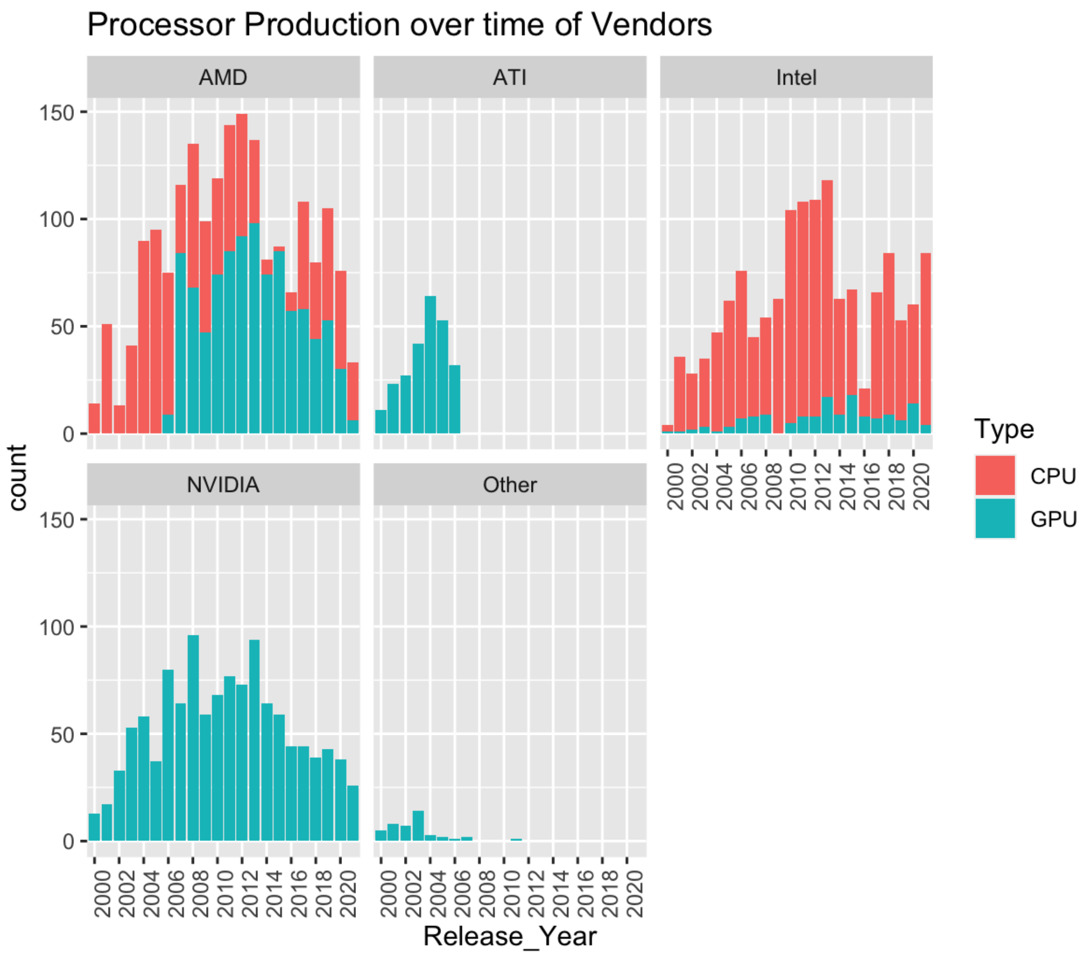

Figure 5.3: Processor Production of Different Vendors over Time

All of the 'Big-Three' processor companies seem to **more or less follow the bell shape exhibited in the total number of processors graph** presented in 5.1 as Figure 5.1.

Some company specific notes are:

- Despite under-performing in recent years, **AMD has produced the most processors in total**.
- As they're mutually exclusive, **Intel production mean and count mirrors those of its foundry**.
- Of the three, **NVIDIA is a close last in the number of processors produced**, despite having produced the most GPU's.

| Vendor | count | mean |
|---|---|---|
| AMD | 1914 | 137 |
| ATI | 252 | 18 |
| NVIDIA | 1,179 | 84.2 |
| Intel | 1,387 | 99.1 |
| Other | 43 | 72.1 |

Table 5.2: Numerical Summary of Processor Production of Foundries

# 6 | Moore's Law

## 6.1 Definition & Background

Moore's Law states that **the number of transistors in a dense integrated circuit will double every two years**. It is an observation made by and named after **Gordon Moore**, the co-founder of **Fairchild Semiconductors** and **Intel**.

Gordon Moore first asserted in 1965 that the number of transistors would **double every year** for the next decade. With more time and observation, in 1975, he **weakened his assertion to doubling every two years**, for the next decade. His assertion **was not based on any rigorous empirical research**, however it seemed to be robust and stand the test of time, which is why it has later been named as a law.

This prediction has been used by many semiconductor companies as a **target for research and development**, which might have, to some extent, acted as a **self-fulfilling prophecy**, helping maintain its validity.

## 6.2 Test

Now, we put Moore's Law to test and see if it still holds. Initially, we visualize the increase in number of transistors over the years.

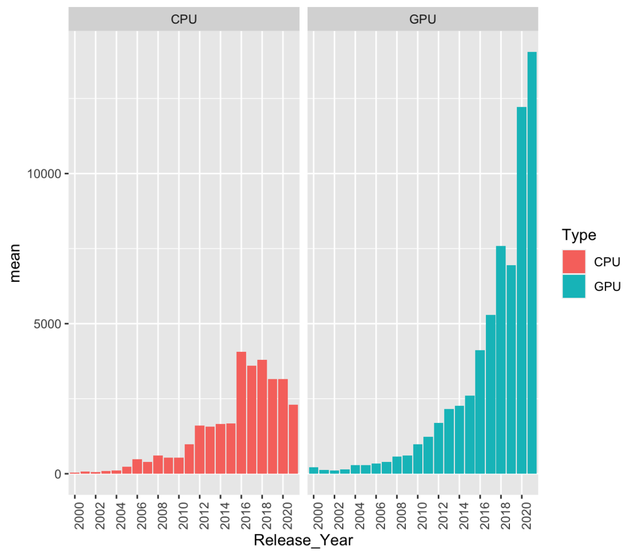

Figure 6.1: Observed Change in Number of Transistors (millions) over the Years

Note: we use the per year average number of transistors (in millions) to generate Figure 6.1

Below is the code to generate Figure 6.1:

```
cpu_gpu_data %>% group_by(Release_Year, Type) %>%
  mutate(Moore = moore(as.numeric(Release_Year))) %>%
  summarise(mean = mean(Transistors_million)) %>%
  ggplot(., aes(x = Release_Year, y = mean, fill = Type)) +
  theme(axis.text.x = element_text(angle = 90)) +
  facet_wrap(~Type) +
  scale_x_discrete(breaks=seq(2000, 2021, 2)) +
  geom_col()
```

In Figure 6.1, we see that the observed distribution for GPU's ressembles the exponential distribution, whereas the distribution for CPU's have ceased to increase. Our intuition tells us that it is more likely for Moore's Law to hold for GPU's.

Now we look at what Moore's Law would look like, given that it meant the average number of transistors would increase $2^{1/2}$ every year, which means it doubles every two years (see Figure 6.2 ).

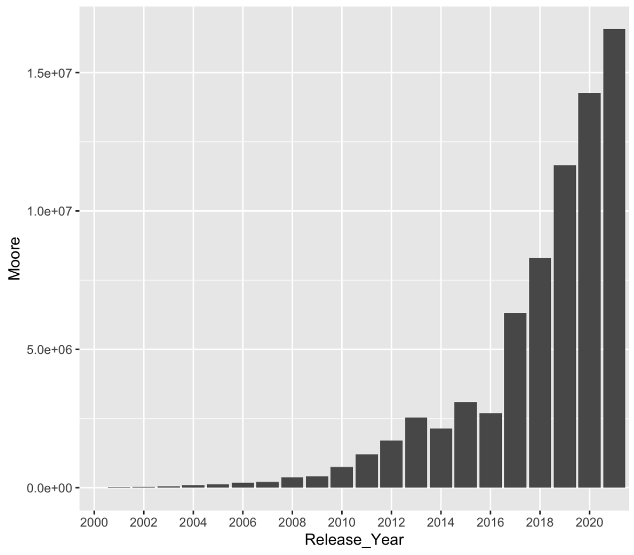

Figure 6.2: Number of Transistors given perfect Moore's, with start 80 at 2000.

Code to generate Figure 6.2 (predicted Moore):

```
moore <- function(release_year, start = 80) {
  years_passed <- release_year - 2000
  return(start * 2^(years_passed/2))
}

cpu_gpu_data %>% mutate(Moore = moore(as.numeric(Release_Year))) %>%
  ggplot(aes(x = Release_Year, y = Moore)) +
  scale_x_discrete(breaks=seq(2000, 2021, 2)) +
  geom_col()
```

It seems like GPU's are closer to perfect Moore's than CPU's, especially in recent years. However, we must calculate numerically how correct our observations are before jumping to a conclusion.

We can use the "squared loss error", which is essentially the Euclidean distance between the predicted and the observed values.

```
squared_dist <- function(release_year, transistors, start_moore) {
  return(mean((moore(as.numeric(release_year), start = start_moore) - mean(transistors))^2))
}
```

By using this, we measure if Moore's Law holds for all processors and it turns out that **Moore's Law is not as accurate as we might think it is**:

```
> squared_dist(as.numeric(cpu_gpu_data$Release_Year),
cpu_gpu_data$Transistors_million, start)
[1] 3221619514
```

Needless to say: a squared error of three billion (3,221,619,514 to be exact), is quite a lot. We find the average squared loss of Moore's for different types and ranges to generate the following table, which suggests that Moore's Law was more 'correct' for CPU's, as well as in 2000-2010 compared to 2000-2020.

| Data | Squared Dist |
|---|---|
| Only GPU | 4,710,113,066 |
| Only CPU | 258,180,400 |
| Pre 2010 | 67,002.95 |

Table 6.1: Caption

In Conclusion: Moore's Law can act as sort of a helping tool for estimation, however **it is by no means an accurate depiction of reality** and we shouldn't always trust Wikipedia.