**The End of Moore’s Law Report**

Contents

[1. What is Moore’s Law 2](#_Toc532526738)

[2. The decline of Moore’s Law 2](#_Toc532526739)

[3. Beneficial aspects of multithreading 3](#_Toc532526740)

[4. Detrimental aspects of multithreading 3](#_Toc532526741)

[5. Conclusion 4](#_Toc532526742)

[6. References: 6](#_Toc532526743)

# 1. What is Moore’s Law

By definition, Moore’s Law states that the number of transistors per square inch on integrated circuits had doubled every year since they have been invented. Not only did Moore observe this discovery but also predicted that this would be the case for the foreseeable future. Voller et al (2002) explains that Gordon Moore figured out that computer power which is measured by the number of transistors that could fit onto a chip doubled every once 1.5 years. Mack (2011) also supports this statement by implying that just after six years since the introduction of the first commercial planar transistor in 1959, Moore was able to identify the pattern that the number of components per chip roughly doubles every year. This specific trend has been established ever since and to this day it has been more than 50 years in which Moore’s Law has been functioning. This proves that the statement that he made was indeed correct and can easily be identified as a substantial prediction done at the early stages of the development of new technology for that time period. Mollick (2006) explains that Moore’s Law has been credited with being the engine of the electronics revolution and is an example of a self-fulfilling prophecy in both academic and popular press. Regardless, with recent software and hardware advancements this trend might not persist for much longer. An increasing demand for multithreading applications and use of multi-core processors can be observed. Due to this, Moore’s Law is seemingly becoming increasingly unrealistic when based on future technological advancements. This is why a substantial amount of studies show that Moore’s Law might actually be at its end after its long-lasting existence of over 50 years.

# 2. The decline of Moore’s Law

The continuous decline of Moore’s Law is currently a fact. Provided that it’s lifespan greatly outmatched any expectations, the point at which its validity is nearing its end is quite near. Schaller (1997) explains that at a certain point the technological aspect behind Moore’s Law will inevitably stop its exponential growth and will then execute with diminishing marginal returns. Waldrop (2016) says that as of recent years the chip-making process has been getting so much more expensive and complex with the addition of a large amount of stages and extra materials needed, that its cost would exceed the value it can potentially provide given that Moore’s Law is still followed. There is no denying that Moore’s Law can be regarded as a foundation for the evolution of transistors on processors, but it is without a doubt that the time at which the benefits from multithreading and multi-core use are overshadowing its effectiveness. The continuous need for cost efficiency and the creation of products that exponentially increase their complexity and usefulness has pushed the boundaries to the point where the statement made by George Moore can no longer be valid when put into perspective of modern day technology.

# 3. Beneficial aspects of multithreading

When looking into the positive functionality of multithreading programming there are certain areas that clearly benefit from parallelism. Eggers et al (1997) says that by utilizing multithreaded programming for a mix of independent programs the overall throughput of the machine is improved and similarly, programs that are parallelizable reap the same throughput benefits which results in a program speed up. This claim supports the statement that multithreading can indeed improve the quality and complexity of certain programs that currently exist and will be created in the future with attention primarily focused on multi-core processing. There are other benefits that can be observed, one of which is program structure simplification. This specific example allows for the threads to be used to simplify the structure of complex applications and by doing so make them more adaptive to a wider variation of user demands. Improved server responsiveness is also achievable through the use of multithreading as it allows for an overall greater throughput of the server. In his paper Tullsen et al (1996) states that multithreading can achieve significant throughput gains when running multiple threads while also having minimal performance impact on a single thread executing alone. Another noteworthy beneficial aspect is the minimized system resource usage, meaning that threads require less overhead to create and are easier to maintain and manage then when using a traditional process. Also, multithreading allows for the simultaneous and fully symmetric use of multiple processors for computation. By taking into account all the benefits that can be derived from multithreading, it can be said that multithreaded applications will continue to improve as they can provide better performance and meet the demand of the market, in order to fill the growing gap that is created by the absence of Moore’s Law.

# 4. Detrimental aspects of multithreading

Even though there is an abundance of positive functional usages for multithreading, there are some negative ones as well. The main issue is that writing code for multithreaded programs is more difficult than doing so for single-threaded applications. The code required generally needs to be more confined and is normally only produced by highly qualified programmers with expert knowledge of the topic. Berger et al (2009) explains that there has been a recent shift from single to multiple architectures and due to this programmers are now required to produce concurrent multithreaded programs to satisfy the increasing application performance requirements. Berger continues by underlining a specific downside to this, which is the fact that these applications are prone to numerous errors such as: deadlocks, order violations and others that increase the difficulty of writing the required code. Debugging and testing are also influenced by this as the code produced is much more complex and difficult to work with, making it impractical to identify and verify root causes whenever an error occurs. Managing concurrency among threads is also an issue as the potential of introducing new problems into the application is increased. It should also be noted that parallelizing is not required for every individual part of an application. Because of this it is crucial to identify the aspects of the program that can be benefited from it and only try to code them with multithreading in mind. This will ensure that no needless complication of the application will be made and allow it to be more stable and less error susceptive.

# 5. Conclusion

In conclusion, modern day technological advancements have reached the point to where high priority is put on producing applications that benefit from multithreading programming and usage of multi-core processors. This is a trend that is continuously being built upon and therefore, future hardware and software will most certainly be revolved around it. Moore’s Law has provided a stable guideline for developers and manufacturers to follow for decades, allowing them to produce applications and products that will satisfy the consumer needs and also be cost effective. However, as some of the papers above show, the reliability of Moore’s Law is diminishing at a steady rate to the point where it will no longer be valid at a certain point of time. This means that manufacturers will need to come up with new ideas to improve the quality of the chips as consumer interest and expectations will continue to increase, following the trend that has been present for such a long time. This is exactly why utilizing multithreaded programing needs to be a priority for all developers. Even though, there are risks that will come with this, specifically focused on the difficulty of coding and finding the required qualified programmers, it is more than clear that ensuring a more secure and long-lasting development cycle is the highest priority.

# 6. References:

Voller, V. and Porté-Agel, F. (2002). Moore's Law and Numerical Modeling. 179(2), pp.698-703.

Mack, C. A., (2011). IEEE Transactions on Semiconductor Manufacturing. 24, 202-207.

Schaller, R. R., (1997). IEEE Spectrum. 34, 53.

Waldrop, M. (2016). The chips are down for Moore’s law. *Nature*, 530(7589), pp.144-147.

Eggers, S., Emer, J., Levy, H., Lo, J., Stamm, R. and Tullsen, D. (1997). Simultaneous Multithreading: A Platform for Next-Generation Processors. In IEEEMicro, pages 12-19.

Berger, E. D., Yang, T., Liu, T., and Novark., G., (2009) Grace: Safe Multithreaded Programming for C/C++. OOPSLA.

Tullsen, D., Eggers, S., Emer, J., Levy, H., Lo, J. and Stamm, R. (1996). Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor, in ISCA.

Mollick E.,(2006). IEEE Annals of the History of Computing 28(3), 62.

Oracle - https://docs.oracle.com/cd/E19455-01/806-3461/6jck06gqj/index.html?fbclid=IwAR1EzEtXqQteuZdDSAl6os2KdrxkgiaV0ZG8aap-rsVf2GWMdEmYnyIP4Uc