Micro-sequencing, Machine and Assembly Language, and Supercomputers

Computer Architecture and Operating Systems: Formative Assessment

Department of Computer Science

University of York

April 2024

Word Count: 433, Task 1

388, Task 2

414, Task 3

93, Task 4

Total (1328)

1. Instruction micro-sequencing. Write a short tutorial-style summary of the ideas and principles of instruction micro-sequencing.
   1. Explain the basic operations and sequence involved in the execution of a simple instruction by a CPU.

Instruction micro-sequencing is a fundamental concept in the operation of CPUs (Central Processing Units) that allows for the orderly execution of machine instructions. Instruction micro-sequencing breaks down the execution of a simple instruction into a series of sequential operations, each controlled by the CPU's microarchitecture. This process ensures efficient instruction processing and contributes to the overall performance of the computer system.

Basic Operations and Sequence in Instruction Execution:

1. The execution of a simple instruction by a CPU involves a series of basic operations that can be generalised into the following sequence:
2. Fetch (IF) : The CPU retrieves the instruction from memory, based on the address held in the Program Counter (PC). The fetched instruction is then placed in the Instruction Register (IR).
3. Decode (ID) : The CPU decodes the fetched instruction to determine what operation is to be performed. This involves interpreting the opcode and identifying the operand(s) if any.
4. Fetch Operand(s): If the instruction requires operands (data on which the operation is performed), the CPU fetches them from memory or internal registers.
5. Execute: The CPU performs the operation specified by the instruction, using the ALU (Arithmetic Logic Unit) for arithmetic or logic operations.
6. Write-back: The result of the operation is written back to the destination register or memory, completing the execution of the instruction.
7. Update PC: The Program Counter is updated to point to the next instruction, preparing the CPU for the next cycle.
   1. In addition, use an example of a simple ADD or SUB operation to illustrate the control sequence operations involved in the execution of a simple instruction by a CPU.

Example Case:

* Initial Values: Register1 = 8, Register2 = 7, Register3 = 33
* Operation: Add Register1 and Register2, then subtract Register3 from the result.

Program Trace:

Start with ADD operation:

* Fetch: Retrieve the ADD instruction.
* Decode: Decode the ADD instruction, identifying it requires the contents of Register1 and Register2.
* Fetch Operands: Fetch values 8 and 7 from Register1 and Register2, respectively.
* Execute: Perform addition (8+7=15) using the ALU.
* Write-back: Store the result (15) back into Register1.
* Update PC: Move to the next instruction.

Proceed with SUB operation:

* Fetch: Retrieve the SUB instruction.
* Decode: Decode the SUB instruction, identifying it requires the contents of Register1 and Register3.
* Fetch Operands: Fetch values 15 (result of the previous operation stored in Register1) and 33 from Register3.
* Execute: Perform subtraction (15-33=-18) using the ALU.
* Write-back: Store the result (-18) back into Register1 or another specified register.
* Update PC: End of operations.

1. Machine and assembly languages. Write a tutorial-style summary discussing the concepts of machine and assembly languages.
   1. Explain why machine language uses binary and hexadecimals while assembly language uses hexadecimal notation. Also, convert the following hexadecimal number to a binary number and vice versa, making sure you show all your working in the conversion:

Machine and assembly languages sit at the heart of computer programming, enabling direct communication with the computer's hardware. They are crucial for understanding how software controls hardware. Let's delve into these foundational concepts. Machine language is the most fundamental level of programming language, consisting of binary code (0s and 1s) that a computer's central processing unit (CPU) can execute directly. Each binary digit (bit) represents the most basic form of data in computing, with sequences of bits (usually grouped in bytes) representing instructions or data values. Assembly language is a low-level programming language that uses mnemonic codes and symbols to represent machine language instructions. It serves as a more readable and manageable interface to machine code, abstracting the binary sequences into human-readable text.

Why Binary and Hexadecimal?

The use of binary in machine language is dictated by the digital nature of computer hardware, specifically transistors, which have two states: on and off. These states map perfectly to 1 and 0 in binary, making binary the natural language of computers. However, binary code can become unwieldy for humans to read and write due to its verbosity. To address this, hexadecimal (base-16) notation is often used as a shorthand. Hexadecimal is more compact than binary, making it easier to read and write large values. For instance, the hexadecimal number "A" represents the binary sequence "1010".

Assembly language commonly uses hexadecimal notation for the same reasons it is used in machine language: compactness and readability. Hexadecimal addresses and values are easier for programmers to interpret than long strings of binary digits. Additionally, assembly language allows for the use of labels and comments, further enhancing readability and maintainability.

i. Hexadecimal to Binary Conversion:

* + 7 = 0111
  + F = 1111
  + F = 1111
  + F = 1111
  + F = 1111
  + F = 1111
  + F = 1111
  + A = 1010

Combine the binary sequences:

7FFF FFFAhex = 0111 1111 1111 1111 1111 1111 1111 1010two

ii. Binary to Hexadecimal Conversion: 1100 1010 1111 1110 1111 1010 1100 1110

* + 1100 = C
  + 1010 = A
  + 1111 = F
  + 1110 = E
  + 1111 = F
  + 1010 = A
  + 1100 = C
  + 1110 = E

Combine the hexadecimal digits:

= CAFE FACEh

1. Supercomputers: Write a tutorial-style summary discussing the technologies behind supercomputing.
   1. Your discussion should address the performance of supercomputers affected by Moore’s Law, as well as Amdahl’s Law and Gustafson’s Law.

Also discuss the following:

* 1. The limitations of supercomputers in terms of technology.
  2. Where supercomputer development is expected to go in the future decade.

Supercomputing represents the zenith of computational power. These powerful machines leverage cutting-edge technologies and principles, including those outlined by Moore’s Law, Amdahl’s Law, and Gustafson’s Law, to perform quadrillions of calculations per second.

Impact of Moore’s Law on Supercomputers

Moore’s Law, observing that the number of transistors on microchips doubles approximately every two years, has significantly influenced the development of supercomputers [1]. This exponential growth has enabled supercomputers to become more powerful and energy-efficient over time. The reduction in transistor size from microns to nanometres has allowed for the creation of supercomputers that can perform at exaflop levels, executing a billion calculations per second [2].

Amdahl’s Law and Supercomputer Performance

Amdahl’s Law addresses the limitations of, as being a useful tool for estimating performance improvements in, parallel processing, emphasising, that the speedup of a program using multiple processors is limited by the program’s serial portion, but its applicability depends on the fraction of sequential code in an application and the type of processor used [3] [4]. In the context of supercomputing, this law highlights the challenges in achieving linear performance scaling with increased processor count due to the inevitable serial components of computational tasks.

Gustafson’s Law in Expanding Supercomputer Capabilities

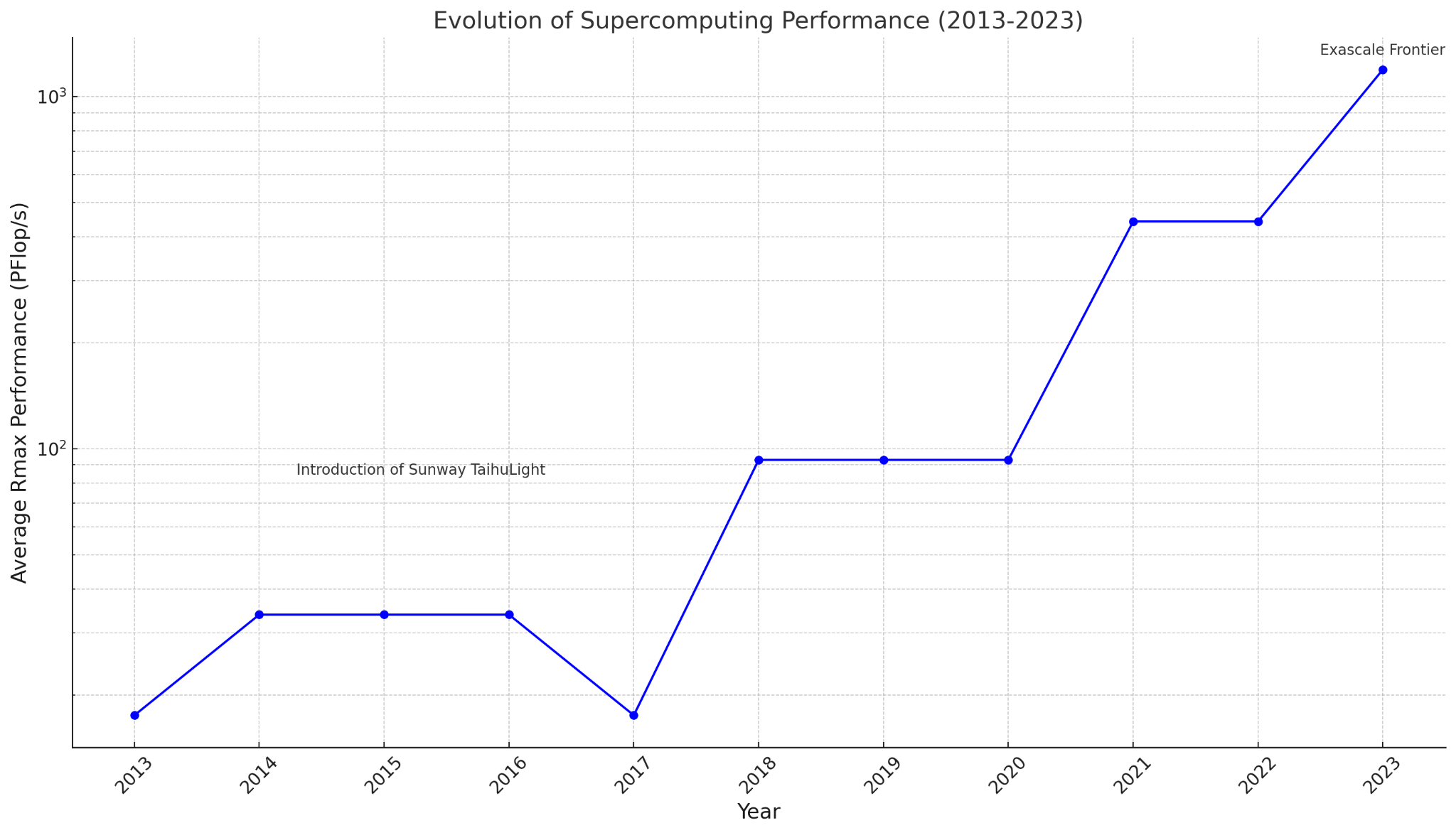
Gustafson’s Law counters Amdahl’s pessimistic view by suggesting that the size of the problem can grow to efficiently utilise the increased computing resources, thus providing a more optimistic outlook on scaling supercomputing performance. This principle has encouraged the design of supercomputers that can tackle larger, more complex problems, pushing the boundaries of what is computationally possible [5].

Technological Limitations and Future Directions

Despite these advancements, supercomputers face significant challenges, including energy consumption, heat dissipation, and the physical limitations of silicon-based processors [6]. As we approach the limits of Moore's Law, researchers are exploring alternative computing paradigms and materials, such as quantum computing and graphene-based transistors [7]. In the future, supercomputer development is expected to focus on overcoming these physical and energy-related constraints. Innovations in 3D chip stacking, optical computing, and the integration of artificial intelligence for system management are anticipated to drive the next decade of supercomputing [8]. The goal is to achieve zettascale computing, enabling even more sophisticated simulations and analyses in fields ranging from climate science to quantum physics.

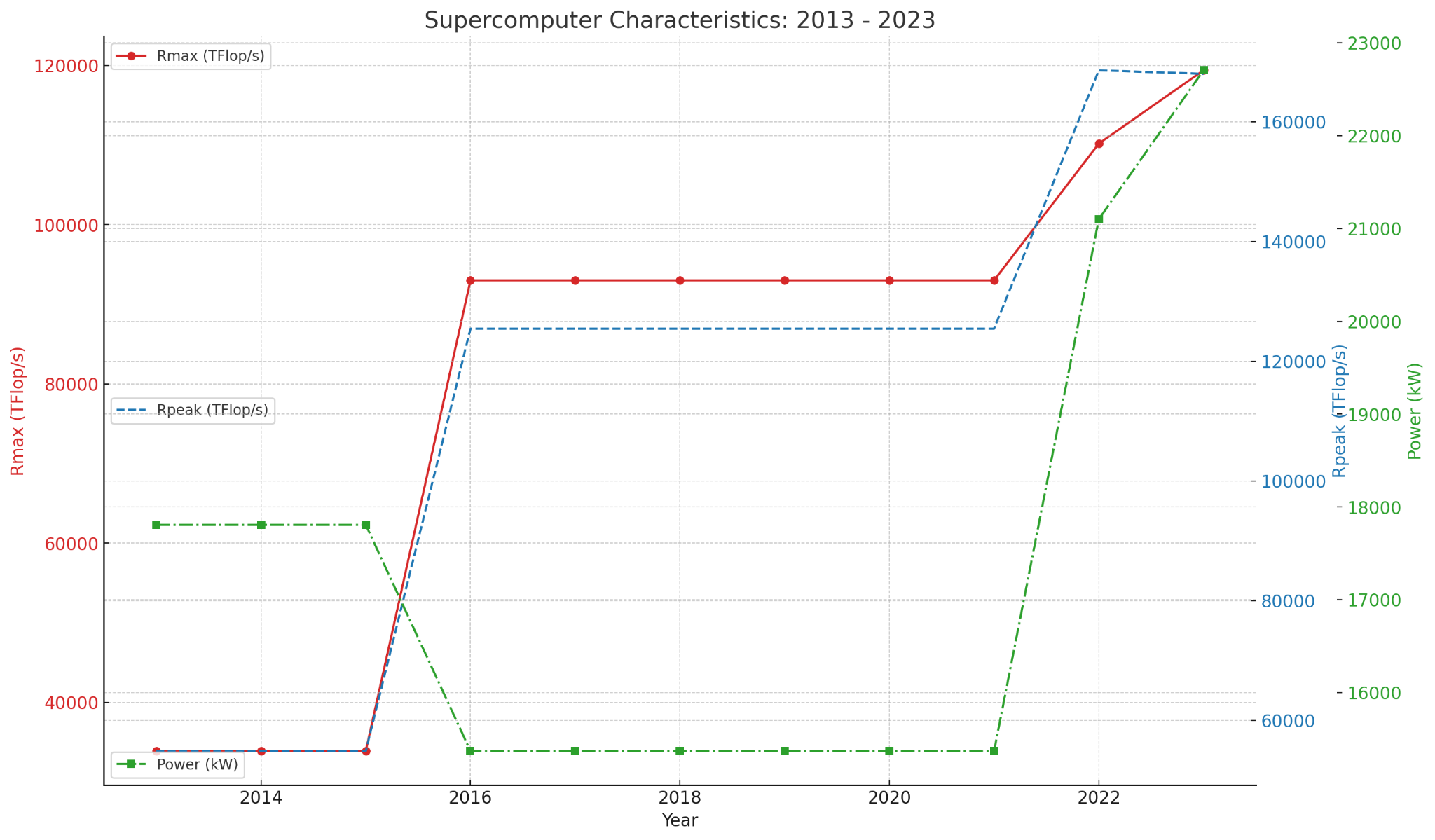
In conclusion, the journey of supercomputing is marked by incredible technological achievements and ongoing challenges. As we move forward, the convergence of traditional computing with emerging technologies promises to unlock new possibilities, solidifying the role of supercomputers in advancing human knowledge and capability.

1. On a separate page, plot your own graph showing a selection of supercomputer characteristics over the past decade. As a minimum, you should plot computing performance, using Flops/Megaflops/Gigaflops, etc, or suitable units. Otherwise, you are free to choose your style of plot (e.g. use of supercomputers in various sectors) and add additional content.



This graph highlights the evolution of supercomputing performance, noting the average Rmax performance in PFlop/s of the top 10 supercomputers each year from <https://www.top500.org/statistics/sublist/>. As shown, there's a notable exponential growth in performance, especially marked by the introduction of the Sunway TaihuLight in 2016 and the significant leap to exascale computing with the Frontier in 2023.

Furthermore, here is a graph including Rmax (TFlop/s), Rpeak (TFlop/s), and power consumption which highlights how computing performance has developed as well as power requirements over the past decade:



# References

|  |  |
| --- | --- |
| [1] | C. A. Mack, “Fifty Years of Moore's Law,” *IEEE Transactions on Semiconductor Manufacturing,,* vol. 24, no. 2, pp. 202-207, 2011. |
| [2] | M. &. S. R. Devoret, “Amplifying quantum signals with the single-electron transistor,” *Nature,* vol. 406, pp. 1039-1046, 2000. |
| [3] | S. Krishnaprasad, “Uses and abuses of Amdahl's law.,” *Journal of Computing Sciences in Colleges,* vol. 17, pp. 288-293, 2001. |
| [4] | M. Annaratone, “MPPs, Amdahl's law, and comparing computers,” in *The Fourth Symposium on the Frontiers of Massively Parallel Computation,*, 1992. |
| [5] | B. &. M. C. Juurlink, “Amdahl's law for predicting the future of multicores considered harmful.,” *SIGARCH Comput. Archit. News,* vol. 40, pp. 1-9, 2012. |
| [6] | W. &. C. K. Feng, “The Green500 List: Encouraging Sustainable Supercomputing,” *Computer,* vol. 40, no. 12, pp. 50-55, 2007. |
| [7] | S. L. G. H. C. S. A. O. M. D. J. L. G. M. W. &. L. M. Vaziri, “A graphene-based hot electron transistor,” *NanoLetters,* vol. 13, no. 4, pp. 1435-1439, 2012. |
| [8] | G. X. Y. &. B. B. Loh, “Processor Design in 3D Die-Stacking Technologies,” *Micro,* vol. 27, pp. 31-48, 2007. |