Set Computing: A Theoretical Framework for Set-Based Transistor Operations

Temitayo Adefemi[1]
University of Edinburgh
t.m.adefemi@sms.ed.ac.uk

January 30, 2025

Abstract

Binary computing has evolved from basic calculators capable of only arithmetic in the 1940s [1] to today's quantum-classical hybrid systems [2] performing complex simulations and artificial intelligence tasks. While this evolution continues, this paper explores a novel computational paradigm titled *Set Computing*, where transistors operate in sets rather than solely through binary logic. Rather than proposing a replacement for current binary systems, this work theoretically explores this new methodology, discussing its theoretical limits and the implementation challenges that would arise in practical applications. The paper establishes the foundational framework for *Set Computing*, providing computer scientists and engineers with the theoretical basis to explore and derive their own conclusions about this computational paradigm's potential applications and limitations.

Keywords: Set, Mathematics, Transistors, Binary, Computing

1 Introduction

Since the dawn of computing, binary logic has provided us with our computational needs. With Moore's law still valid within the current computational ecosystem [3], we have no reason to stray away from binary logic as it provides us with the fundamental and foundational building blocks of modern-day computing. From transistors to logic gates to beyond, we have little to no reason to build other computing methodologies to override our current binary systems. Although we have quantum computing, a growing ecosystem, binary remains the foundational computing mode. So, this paper is not to argue a replacement for binary computing but to explore an alternative computing paradigm that would yield more computing power than our current binary systems. This paradigm is called "Set Computing," in which transistors operate in sets rather than in 0s and 1s.

2 What is Set Computing

Set Computing represents a new computational paradigm in which transistors act within a set framework rather than 0 and 1. For example, there are a million transistors in a chip in binary computing. Still, in set computing systems, we say there are two hundred thousand sets of transistors in the chip (using a set derivative of 5). The transistors within a single set operate in binary within their confined sets, but more computation is derived from union and intersection with other sets.

The computational power can be calculated as follows: for binary logic with one million transistors, each transistor has two possible states (0 or 1), resulting in a total computational power of $2^{1,000,000}$ possible states. In the set computing paradigm, each set of 5 transistors can have $2^5 = 32$ possible internal states, and with 200,000 such sets, the base computational power is $32^{200,000}$. Additionally, the sets can interact through unions and intersections, adding $C(200000, 2)$ possible pairwise interactions, which equals 19,999,900,000 interaction possibilities. Therefore, the total computational power of the set computing system is $32^{200,000} \times 19{,}999{,}900{,}000$, representing both the internal states of the set and the interaction possibilities between sets. This demonstrates how set computing leverages both traditional binary operations within sets and additional computational power from set theory operations between sets.

Set Computing demonstrates theoretical potential for greater computational power than traditional binary systems. This increased power comes from the interchangeable operations between sets of transistors, which scale exponentially as more sets are added. For example, with 200,000 sets of 5 transistors each, we achieve not only the base computational power of $32^{200,000}$ from internal binary operations but also an additional factor of approximately 20 billion from set interactions. However, it is crucial to acknowledge that implementing such a system would face substantial engineering challenges, requiring a complete redesign of the current computing architecture. Therefore, this paper presents Set Computing as a theoretical framework rather than a practical alternative to binary computing systems.

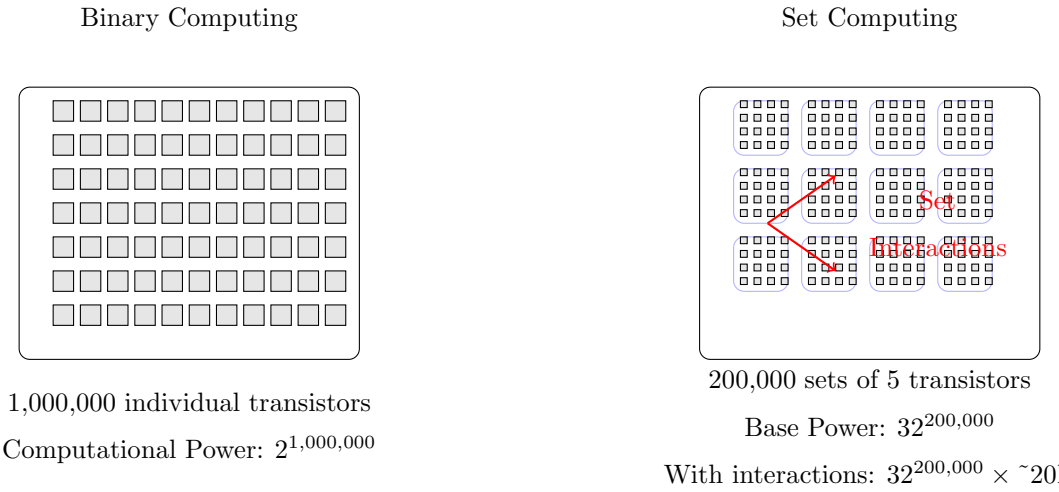

Figure 1: Comparison of computational paradigms showing traditional binary computing (left) versus set computing (right). In binary computing, each transistor operates independently with two possible states. In set computing, transistors are grouped into sets of 5, enabling both internal binary operations and set theory operations between groups. The total computational power is significantly enhanced through the combination of internal states ($32^{200,000}$) and the approximately 20 billion possible set interactions ($C(200000, 2)$).

3 Methodology

3.1 Theoretical Framework for Set Interactions

In this section, we outline the theoretical basis for how sets of transistors interact, develop a mathematical model to describe these set operations and discuss the underlying assumptions and constraints of the computational model. While the principles described here are largely conceptual, they serve as a foundation for understanding the exponential growth in computational power when transistors operate within an interconnected set framework.

3.2 Interaction of Sets of Transistors

In traditional binary computing, each transistor is viewed as an independent element that can take on one of two states (0 or 1). By contrast, *Set Computing* groups transistors into confined clusters, each retaining binary functionality internally but interacting externally with other sets. We denote each set as S_i for $i \in \{1, 2, \ldots, m\}$, where m is the total number of sets. Within each set, the transistors coordinate to produce a local state. Still, the key innovation lies in the interplay between different sets, creating exponential computational power that cannot be extracted while utilizing binary logic.

Concretely, let each set S_i consist of k transistors, enabling 2^k possible local states. These sets can then perform higher-level operations such as union(S_i, S_j), intersection(S_i, S_j), and other set-theoretic functions that yield additional modes of computation beyond simple binary states. This framework provides a multi-layered computational space: the internal binary processes within each set form the first layer, and the cross-set interactions form subsequent layers that can drastically expand the overall potential state space. We limit the discussion of these higher-level operations to unions and intersections; there are significantly more set operations that could be processed within this computational paradigm,

which include inverses, composites, and functions that would allow more specialized computation within the transistor level.

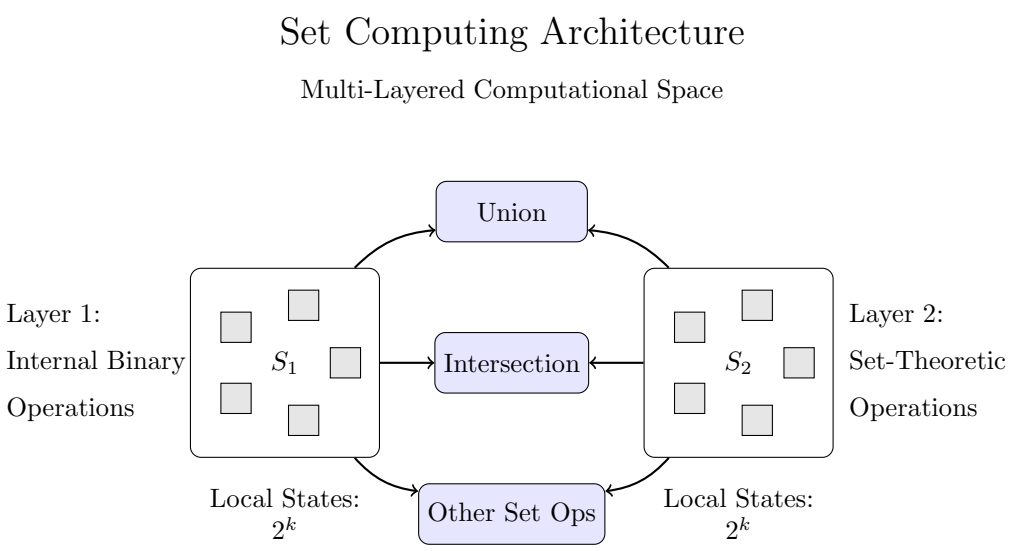

Figure 2: Architectural overview of Set Computing showing the two primary computational layers: internal binary operations within sets and set-theoretic operations between sets. Each set S_i contains k transistors that can produce 2^k local states, while inter-set operations provide additional computational capabilities through unions, intersections, and other set-theoretic functions.

3.3 Mathematical Model for Set Operations

To illustrate the mathematical underpinning, consider a system with N total transistors grouped into $m = \frac{N}{k}$ sets of size k. Each set S_i can be in any of the 2^k internal configurations. The operation union(S_i, S_j) can be conceptualized as combining the state information of two sets into a new, derived state, while intersection(S_i, S_j) can be viewed as extracting only the states shared by both sets. In principle, any standard set-theoretic operation—difference, symmetric difference, and so on—could be implemented, although the practicality of doing so in hardware remains an open question.

From a purely combinatorial perspective, the total number of internal configurations across all sets is $\left(2^k\right)^m = 2^{k \cdot m} = 2^N$. The theoretical enhancement emerges when considering pairwise (or multi-way) set operations. In the simplest form, we can account for the number of possible pairwise interactions as $\binom{m}{2}$. If each pairwise interaction adds another dimension of possible states, the overall computational space grows by a factor proportional to $\binom{m}{2}$. Consequently, we can approximate the potential computational power as:

$$\left(2^k\right)^m \times \binom{m}{2}.$$

While this expression is a simplified model—it treats set operations as if each yields entirely distinct outcomes—it underscores how interactions among sets can compound the total system capability beyond what is typically captured by a straightforward binary model.

3.4 Assumptions and Constraints in the Computational Model

Several assumptions underlie this framework:

- **Idealized Interactions:** We assume that union, intersection, and other set operations occur without delay or energy loss. In real hardware, these interactions would require complex routing

and control mechanisms that might introduce significant overhead, typically reflecting the derived computational power gained from this new paradigm [4].

- **Perfect Independence of Sets:** We treat each set's internal state as entirely independent from others before set operations. Physical effects such as interference, wiring constraints, and synchronization overhead could couple the sets in practice. It would require significant engineering effort to isolate sets from others while ensuring they operate cohesively within the framework.
- **Uniform Set Size:** The model presumes each set consists of the same number of transistors k. This uniformity simplifies the mathematics but may be restrictive for practical hardware designs, where yield variations, or functional requirements might dictate unequal partitioning.
- **Scalability:** We extrapolate from a theoretical standpoint, implying that the computational power continues to grow as the number of sets m increases. However, real-world systems will face limits in power consumption, thermal dissipation, and fabrication feasibility, all of which cap the practical scalability [5].

These assumptions and constraints make clear that while the *Set Computing* paradigm offers an intriguing exponential expansion of computational possibilities, its feasibility is contingent upon solving numerous engineering and design challenges. This is why this paper remains a theoretical exploration rather than claiming or arguing for a replacement of our traditional computing systems. Current engineering can't solve the challenges this new computing paradigm brings. While the computational power it would yield is promising and intriguing, we have to be clear that this theoretical computational power is based on assumptions that could be potentially misleading. The following sections discuss the broader implications of these theoretical results and explore how they align with or diverge from traditional approaches to computing.

4 Implementation Challenges

While the theoretical underpinning of Set Computing suggests a vast increase in computational power, its realization in physical hardware poses enormous challenges, while theoretically possible, implementing it would require a significant engineering challenge. This section explores some of the most pressing problems and hurdles that would likely arise as we implement this new computational system, including physical constraints, heat dissipation, signal integrity, and clock synchronization [6]. Addressing these issues would be critical to future attempts to build a set-based computing system.

4.1 Physical Limitations and Constraints

Today, transistor fabrication processes are optimized for binary logic architectures. Introducing sets of transistors that interact at multiple levels (internal state plus set operations) could require novel manufacturing processes or additional layers of circuitry. For instance, each set would need dedicated routing for union and intersection operations, and this routing can become very substantial as every transistor within every individual set would also need to be routed to all transistors within every set, which would be challenging and very time-consuming. The resulting increase in interconnect density might not be practical given existing lithography resolutions and might require entirely new materials or fabrication techniques.

4.2 Heat Dissipation Concerns

Exponential expansions in computational states often come with proportional increases in power consumption. A set of transistors interact more frequently, the total number of operations performed per clock cycle can skyrocket. This, in turn, generates substantial heat, which must be mitigated to avoid thermal damage and performance degradation. Traditional cooling approaches might be insufficient for chips designed around large-scale set interactions, necessitating advanced packaging solutions or alternative computational models that inherently limit thermal output [4].

4.3 Signal Integrity Between Sets

Signal integrity issues such as cross-talk, voltage droop, and electromagnetic interference can degrade performance in complex integrated circuits. When transistors are grouped into sets that interact, the

volume of signaling activity between these groups increases. Ensuring clean signal transitions, especially at higher clock speeds or within densely packed interconnect networks, becomes more challenging. This problem is exacerbated as we scale up the number of sets and the complexity of their interconnections, potentially requiring specialized shielding or routing techniques that further inflate manufacturing and design costs.

4.4 Clock Synchronization Across Set Operations

Synchronizing traditional binary logic pipelines is already a well-studied challenge, but it becomes even more complicated when large numbers of sets are coordinated via union and intersection operations. The resulting logical errors could negate any theoretical gains if different sets operate out of phase or the clock skew across the chip is too significant. Designers must consider using a global clock, adopting a distributed clocking scheme, or even exploring asynchronous design methodologies. Each approach introduces trade-offs in terms of performance, complexity, and reliability [6].

4.5 Latency

Computational latency presents a significant challenge in this paradigm due to the physical distribution of transistor sets across the chip. While computations within individual sets and between nearby sets may be relatively quick, operations involving the intersection and union of widely separated sets can introduce notable delays. When sets are spread inches or meters apart, the time required to complete these cross-set computations increases substantially, resulting in observable latency impacts on overall system performance.

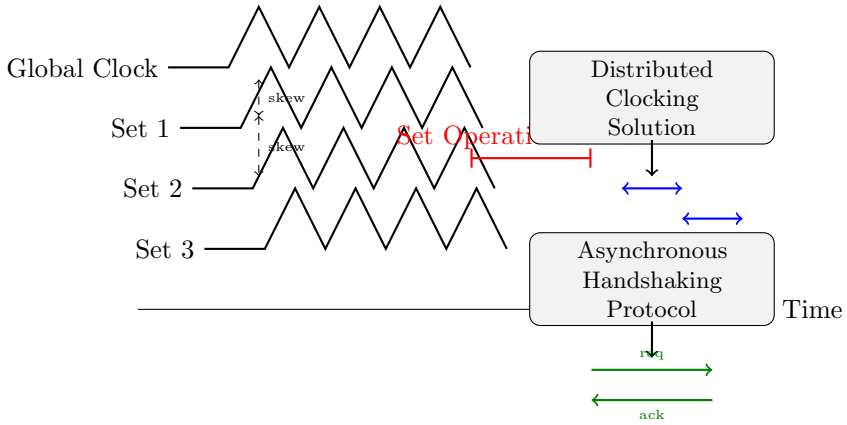

Figure 3: Clock synchronization challenges in Set Computing. The diagram illustrates clock skew between different sets (top), highlighting the timing window required for set operations. Two potential solutions are shown: distributed clocking with local synchronization signals (middle) and asynchronous handshaking protocols (bottom). Clock skew between sets can lead to timing violations during set operations, necessitating careful design of the synchronization mechanism.

These implementation challenges highlight the gap between the theoretical promise of Set Computing and the realities of modern chip design. There are significantly more hurdles which are not discussed in this paper. Overcoming these obstacles would require substantial research and investment in circuit-level innovations, fabrication processes, thermal management, and systems architecture. Nonetheless, the potential computational gains remain an intriguing incentive to continue exploring this new computational paradigm.

5 Comparative Analysis

Set Computing, rooted in binary operations at the transistor level, aspires to surpass traditional binary computing through its meta-level interactions across sets. We can compare it with current binary systems, quantum computing, and other alternative paradigms to better understand its potential and limitations.

Comparison with Current Binary Systems. Traditional binary systems form the foundation of modern computing and have enjoyed decades of optimization in fabrication, design tools, and software ecosystems. Moore's law is still being observed, implying we have yet to reach the peak of what binary systems could offer. By contrast, Set Computing adds a layer of complexity through set-theoretic operations, which promise an expanded state space by allowing transistors to be grouped and to interact across sets. In principle, this approach could yield a super-exponential scale of computational possibilities, but the overhead of routing, clock synchronization, heat dissipation, and other challenges may offset theoretical benefits. While binary systems scale effectively to advanced process nodes (like sub-5nm), Set Computing might require entirely new design paradigms to handle the increased interconnect and operational complexity. Binary systems has also flourished with an advanced ecosystem, switching from binary to set computing would require a complete and drastic overhaul of a computing ecosystem that has been built across multiple decades and has all the infrastructure put in place to allow it to continue to blossom grow which makes adaptation of set computing very unlikely but it is still exciting to explore theoretically.

Comparison with Quantum Computing Approaches. Quantum computing leverages qubits, which exploit superposition and entanglement for exponential speedups in specific problem domains, such as prime factorization or quantum simulations [2]. On the other hand, Set Computing remains classically deterministic but uses set operations to expand the combinatorial space. Both paradigms challenge the limits of classical binary logic, yet quantum computing faces its obstacles, including decoherence and error correction. In contrast, Set Computing grapples with hardware-level interconnect complexity rather than quantum decoherence, offering a different route to high-performance computing if the engineering issues can be addressed.

Other Alternative Computing Paradigms. Beyond quantum and binary, alternative paradigms such as analog computing, neuromorphic computing [7], and multi-valued logic [8] offer distinctive ways to handle computation. Analog computing uses continuous signals instead of discrete states and is well-suited to specific differential equation problems. Neuromorphic systems emulate brain-like structures, emphasizing parallel, event-driven operations. Multi-valued logic increases the logic levels beyond 0 and 1 to reduce circuit depth. Set Computing differs by maintaining binary logic within each set but introducing higher-level interactions among multiple sets, potentially enabling exponential growth in state space for combinatorial operations without abandoning classical (binary) transistor fundamentals.

6 Future Research Directions

Despite being at a largely theoretical stage, Set Computing opens several promising avenues of investigation. Future research could validate the concept through limited-scale implementations or propose new computational architectures leveraging set-based logic.

Experimental Validation of the Theory. A key step towards realization is creating small-scale experiments on simulated platforms or test chips that capture basic set operations (union, intersection) in hardware. Such prototypes would reveal whether the hypothesized exponential growth in computational power is achievable in practice or overshadowed by interconnect and timing overheads. Experimental validation could also highlight unforeseen physical constraints, guiding further model refinement.

Possible Architectural Implementations. Researchers could develop specialized architectures optimized for set operations if preliminary testing suggests feasibility. For instance, chips might incorporate configurable networks designed to support efficient union and intersection processes among transistors. These architectural designs could integrate 3D stacked layers or advanced materials to mitigate the wiring density challenge. Exploring these ideas might involve collaborating with semiconductor manufacturers on prototypes, laying the groundwork for a new category of computing systems.

Optimization Strategies for Set Operations. Even in a hypothetical hardware design that supports set interactions, efficient mapping of high-level software tasks onto physical sets is critical. Compiler techniques could translate complex set-based instructions into lower-level hardware commands with minimal overhead. Similarly, data layout optimizations might aim to keep commonly intersecting sets

physically close, reducing interconnect lengths. Such strategies would be central to tapping into the performance gains offered by Set Computing while managing power and timing constraints.

Applications Where Set Computing Might Be Particularly Advantageous. Finally, certain problem domains could benefit more significantly from set-based interactions. For instance, combinatorial search algorithms that rely on union and intersection to traverse large solution spaces could see notable speedups if appropriately managed in hardware. Database management, graph algorithms, and specific AI techniques dealing with set-like structures might all find unique advantages in a computing architecture that natively supports high-speed set operations. Although Set Computing may remain a conceptual alternative to traditional binary systems, it highlights new possibilities for accelerating tasks where set relationships play a pivotal role.

7 Conclusion

Set Computing offers a theoretical model in which traditional binary logic is embedded within larger, set-based structures, allowing transistors to interact through operations such as union and intersection. This approach theoretically promises exponential gains in possible system states and operations, far surpassing the capabilities of conventional binary designs when evaluated purely from a combinatorial standpoint. However, significant engineering challenges stand in the way of practical deployment. Issues such as heat dissipation, signal integrity, interconnect density, clock synchronization, and the latency required for these computations to inherently take place typically complicate any attempt to scale the paradigm beyond small experimental prototypes. Moreover, entirely new software stacks and architectural frameworks would be required to efficiently harness and manage these set interactions at runtime, requiring a typical overhaul of our current systems. Still, it's important to note that this form of computing would typically be favorable for set-related problems and computational combinatorial problems, so it is worth a methodology worth exploring.

Despite numerous challenges in implementing this new paradigm, Set Computing offers an alternative computing method to binary systems. It poses a thought-provoking question: could a new logic layer built on top of binary transistors unlock powerful and novel forms of computation? As with quantum computing, neuromorphic architectures, and multi-valued logic, Set Computing could implement future innovations that may emerge from rethinking the foundational assumptions that have guided computing for decades. Even if the paradigm remains predominantly theoretical, continued exploration may yield valuable insights, shaping how we imagine the next generation of high-performance and specialized computing systems.

References

[1] J. Bardeen and W. H. Brattain. The transistor, a semiconductor triode. *Physical Review*, 74:230, 1948.

[2] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[3] G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8):114–117, 1965.

[4] R. Mahajan, R. Nair, V. Wakharkar, J. Swan, J. Tang, and G. Vandentop. Emerging directions for packaging technologies. *Intel Technology Journal*, 6(2):62–75, 2002.

[5] D. A. B. Miller. Rationale and challenges for optical interconnects to electronic chips. *Proceedings of the IEEE*, 88(6):728–749, 2000.

[6] I. E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, 1989.

[7] C. Mead. *Analog VLSI and Neural Systems*. Addison-Wesley, 1989.

[8] D. J. D. Robinson, R. K. K. Y. Salama, and M. F. Chang. A novel ternary logic circuits in standard CMOS technology. In *Proceedings of the 2001 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 606–609, 2001.