An Introduction to Reversible Computing

Avi Patel, Joshua Yue
Department of Computer Science
University of Texas at Austin
avipatel@utexas.edu
joshuajyue@utexas.edu

***Abstract*—Reversible computing is an unconventional computing paradigm that promises to overcome fundamental energy dissipation limits in computation. This paper provides an educational overview of reversible computing for readers with a computing background but little prior exposure to the concept of reversible computing itself. We first introduce the theoretical foundations, including Landauer's principle and the distinction between reversible and irreversible logic operations, establishing why reversible computing can, in principle, achieve arbitrarily low energy dissipation. We then look at practical developments and applications, particularly focusing on energy efficiency gains. Topics include adiabatic CMOS circuit techniques for near-reversible logic, recent reversible hardware prototypes (such as Vaire Computing's reversible chips), reversible programming languages and compiler support, and the special case of quantum computing as inherently reversible computation. We evaluate the practical feasibility of reversible computing in terms of energy savings and scalability, discussing current challenges such as circuit complexity, speed trade-offs, and noise. Finally, we present a balanced outlook on the future of reversible computing, assessing whether it is likely to play a role in mainstream computing in the coming decades.**

I. Introduction

In the modern era, the limits of computation have not been defined by clock speed or transistor size, but by energy. Virtually all machines today use *irreversible logic*, where logic operations irreversibly lose information (for example, a logic gate that merges two input bits into one output bit irreversibly "forgets" some an input bit), unavoidably converting some energy into heat. This energy loss per bit has a fundamental lower bound known as the Landauer limit, approximately $kT \ln 2$ joules (where k is Boltzmann's constant and T is temperature in kelvins) for each erased bit of information. This is more commonly known as **Landauer's principle**, which was first articulated by Rolf Landauer in 1961 [1].

In a reversible computation, every operation can be inverted to recover its inputs from its outputs. In principle, an ideal reversible computer could perform computations with arbitrarily small energy dissipation, reusing the same energy repetitively instead of converting it to heat [2]. Reversible computing has long been of theoretical interest for its promise of ultra-low-power computation, but it has remained mostly in the research domain since Landauer's initial realization [3]. Recently, however, there has been renewed attention on translating reversibility into practice as energy efficiency has become a paramount concern. For example, a startup company named Vaire is now attempting to commercialize reversible computing chips by 2027, aiming for drastic reductions in energy per operation [4].

This paper provides a comprehensive introduction and survey of reversible computing, intended for other students familiar with computing basics but not with this field. We begin with the theoretical foundations, explaining why and how reversible computing can save energy in Section II. We then discuss real-world implementations and applications in Section III, from adiabatic CMOS circuits [5] and prototype reversible hardware to software support via programming languages and compilers [6], as well as the role of reversibility in quantum computing [7].

In Section IV, we examine the practical feasibility of reversible computing, including the energy-saving potential observed [8], scalability issues, and engineering challenges that must be overcome. Finally, Section V concludes with a balanced perspective on the future of reversible computing, weighing its potential to transform modern computing in the next decades against the obstacles on the road to that goal.

II. Theoretical Foundations

At the heart of reversible computing is the connection between information and thermodynamics. Landauer's principle formalizes this connection by stating that any logically irreversible manipulation of information – for instance, erasing a bit to a known state "0" or "1" – must be accompanied by a corresponding increase in entropy (disorder) in the physical world, i.e. at least $kT \ln 2$ of energy dissipated as heat for each bit erased [1]. This principle is a consequence of the second law of thermodynamics and was first proposed by Landauer in 1961.

Thus, as long as an operation discards information that cannot be recovered from the output, it has a fundamental energy cost. In current CMOS circuits, every logic gate that outputs fewer bits than it inputs (such as an AND or OR gate combining two inputs into one output) effectively throws away information and thus wastes energy. In the 2020s, the gap between the smallest transistor gates and Landauer's limit was only on the order of hundred-fold. Michael P. Frank predicts that in around 10 years, progress in reducing logic-gate heat dissipation will stall due to a fundamental, unavoidable margin of thermal noise - around 20-80× Landauer's limit [8].

A computation that is logically reversible avoids this inherent dissipation by never compressing many input states into one output state. Formally, a function (or operation) is

logically reversible if it is one-to-one (injective); each output state corresponds to exactly one input state. In practical terms, a reversible logic gate has the same number of output bits as input bits, so that it can theoretically be run backward to recover inputs from outputs. A simple example is the NOT gate which maps an input bit x to its complement $\bar{x}$ – this operation is its own inverse, and no information is lost since knowing the output $\bar{x}$ tells us the input x. By contrast, the AND gate is not reversible: e.g. inputs $(1, 0)$, $(0, 1)$, and $(0, 0)$ all produce output 0, so given a 0 output, one cannot tell which input was applied – that "lost" information is dissipated as heat in an irreversible circuit.

However, it is possible to make an AND operation reversible by preserving one of the inputs. For instance, a Toffoli gate (also known as the controlled-controlled-NOT gate) takes three inputs (a, b, c) and outputs $(a, b, c \oplus (a \cdot b))$ [9]. If a and b (the control bits) are both 1, this gate flips the third bit c; otherwise c is left unchanged. The outputs include the original a and b so that no information about them is lost. The Toffoli gate is a universal reversible logic gate: by configuring its inputs (for example, if one input c is set to 0, the gate's third output becomes the AND of a and b, and with a different configuration it can act as a NOT), one can implement any Boolean function reversibly using a combination of Toffoli gates. Another notable reversible gate is the Fredkin gate (controlled swap), which conditionally swaps two data bits depending on a control bit [10]. These multi-bit gates illustrate that any irreversible operation can be embedded in a reversible one given enough additional bits to carry extra information.

In 1973, C. H. Bennett showed that a universal Turing machine could be made logically and thermodynamically reversible by carrying along an arbitrarily large history tape [11], and that in principle this allows computation with arbitrarily little energy dissipation if run slow enough. Later, Bennett also described methods to uncompute or clean up the unused garbage bits during a computation so that they don't accumulate entropy [2].

It is here where it is important to note the distinction between logical reversibility and physical reversibility. A logically reversible operation can still dissipate energy if it is implemented carelessly. While the minimal transistor-level energy is approaching Landauer's limit, real-world computations dissipate millions of times more energy. This discrepancy is not due to Landauer's principle itself, but due to the physical irreversibility of standard architectures — such as capacitors dissipating $\frac{1}{2}CV^2$ by dumping charge to ground. Proponents of reversible computing argue that only by adopting both logically reversible operations and physically reversible (adiabatic) implementations can we eliminate these inefficiencies and push past the energy barriers of modern computing [8].

To be physically reversible, a computing process must be carried out in such a way that it can be undone physically with minimal energy cost, which typically means changing the physical variables gradually and avoiding energy loss to heat due to resistance [3]. The process must be quasi-static and introduce negligible entropy. In practice, perfect reversibility is unattainable – there will always be some resistance, some leakage, or other non-ideal conditions causing slight energy dissipation. However, there is no known lower limit to how close we can approach ideal reversible operation if we engineer the system well enough. The potential reward is clear: by recovering and reusing signal energy instead of dissipating it as heat, a reversible computer could perform many more operations per joule of energy than an irreversible one. In the extreme limit, as Bennett's theoretical constructions suggested, one could do an arbitrary number of operations with only a tiny energy cost, as long as the computation can be reversed or "undone" when needed [2].

These ideas remained theoretical for many years. Pioneers like Fredkin and Toffoli proposed theoretical models such as the billiard ball computer, a mechanical reversible computer where colliding balls in a frictionless environment would perform logic operations without energy loss. In this computing model, computing elements conserve momentum and kinetic energy, ideally dissipating no heat [10]. Others, like Bennett, compared such schemes to thermally driven "Brownian" computing, where random thermal motion could be harnessed for computing with minimal free-energy expenditure [2]. These early thought experiments reinforced the physical plausibility of reversible computing without immediately yielding practical hardware.

Nonetheless, they set the stage for later work by establishing that no fundamental law of physics prevents computing with arbitrarily low energy. To actually exploit this in a computer, one must carefully design systems that avoid bit erasure and allow nearly all the energy invested in a logic transition to be recovered. The remainder of this paper explores how researchers are attempting to do this in real devices and algorithms.

III. Applications and Implementations

Although reversible computing was conceived decades ago, only in recent years have we seen growing efforts to apply its principles in practice. In this section, we highlight several domains where reversible computing ideas are being implemented or explored, with a focus on how they contribute to energy efficiency.

A. Adiabatic CMOS

One of the most direct ways to implement reversible computing principles with existing technology is through adiabatic CMOS circuits. The term "adiabatic" in electronics refers to processes that occur with minimal energy exchange as heat – in essence, charge transfers that are so gradual that energy dissipation is greatly reduced and ideally some of the energy can be recovered [3]. Traditional CMOS logic dissipates energy $E = \frac{1}{2}CV^2$ each time a capacitance C is charged (and the same again when discharged) because it draws charge from a fixed-voltage source and then dumps it to ground, irrecoverably losing that energy as heat. Adiabatic CMOS circuits instead use time-varying power supply waveforms

(often quasi-sinusoidal or ramped voltages) and clever circuit techniques to reclaim charge [5].

For example, rather than abruptly connecting a capacitor to V_{dd} (which would instantly dissipate $\frac{1}{2}CV_{dd}^2$ in the channel resistance), an adiabatic driver might slowly raise the voltage on the capacitor in sync with the logic operation, then later lower it, returning the charge to a reservoir (like an inductor or another capacitor) instead of ground. In theory, if this charging and discharging is done infinitely slowly, the energy dissipation can be made arbitrarily small – the process is nearly reversible, and most of the energy goes into storing charge and is then recovered, rather than being lost as heat [8].

In practice, "infinitely slow" operations are not feasible if we want a useful computing speed, and there are always some resistive elements causing loss. Nevertheless, numerous adiabatic logic families have been developed that demonstrate substantially lower energy per operation than conventional CMOS by trading off speed and complexity. Examples include ECRL (Efficient Charge Recovery Logic), SCAL (Split-Level Charge Recovery Logic) developed in the 1990s, and more recent multi-phase adiabatic clocking schemes [5]. These often require specialized clock generators that produce multi-phase or oscillating power signals. A simple conceptual implementation is to use an LC resonator (inductor-capacitor tank) as the power supply: the inductor can store the energy and exchange it with the capacitive load, acting like a pendulum that swings energy back and forth between kinetic (current in the inductor) and potential (charge on the capacitor) forms. Only a small fraction is lost on each swing, which related to the resistance in the circuit, analogous to friction. In adiabatic CMOS, logic gates are designed so that they perform their boolean function while the power clock slowly transitions, ensuring that current flows through transistors mostly when the voltage difference is small, thereby reducing $I \cdot V$ (current times voltage) dissipation [3].

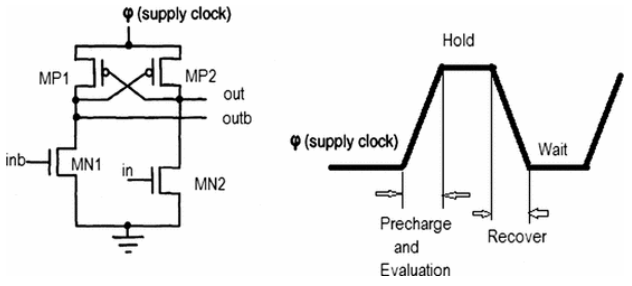

Fig. 1. A sample circuit and voltage curve of an ECRL NOT gate [12]. Voltage is not constant, but a cycled process, with calculations on the rising edge and decomputations on the falling.

A key point is that adiabatic circuits approximate reversible computation – they don't strictly produce the exact input back from outputs, but they attempt to minimize energy loss per logic operation. Many adiabatic logic implementations still output a valid logic result like conventional circuits, often with some additional "return-to-zero" phases or requiring one to supply the input in a particular way so it can later be recovered. In theory, fully reversible logic would also need to preserve all bits (including intermediate ones), whereas practical adiabatic CMOS might still discard bits but does so in a more energy-efficient manner. Even so, adiabatic techniques can be combined with logically reversible gate design. For instance, one could implement reversible gates like the Fredkin or Toffoli gate in an adiabatic fashion, so that not only is the logic reversible, but the physical process of switching is also energy-recovering [9], [10].

Researchers demonstrated early adiabatic reversible gate arrays in the late 1990s and 2000s that achieved orders-of-magnitude less energy dissipation than conventional CMOS when operated at low speeds [3]. As a proof of concept, Younis and Knight (MIT) showed an "adiabatic" 4-bit processor that could in principle run without the large energy losses of a normal CMOS CPU by recycling charge [5]. More recently, adiabatic CMOS ideas have been refined in the context of near-threshold computing and ultra-low-power design, although they have not yet been adopted in mainstream processor products. The main challenges are that adiabatic circuits typically require more components (e.g. complementary outputs, multi-phase clocks, large inductors or capacitors for energy storage) and they impose timing constraints that complicate design. Also, at higher clock rates, the benefit diminishes because any finite resistance causes some I^2R loss that grows with frequency – thus many adiabatic prototypes show great efficiency only at relatively low speeds (KHz to low MHz). Nevertheless, adiabatic CMOS research is a crucial stepping stone in the journey toward practical reversible computing, as it demonstrates how energy can be saved and mostly recovered in logic operations using conventional devices [8].

B. Reversible Hardware: Vaire Computing

Until recently, reversible computing hardware was largely confined to laboratory experiments, but the landscape is starting to change. One notable effort is by Vaire Computing, a London-based startup (founded in collaboration with academics in the field) that is bringing reversible computing concepts into actual chip prototypes [4]. Vaire's approach is firmly rooted in adiabatic CMOS, as described above, but pushes the concept closer to a fully reversible computing architecture. Their strategy is to use standard CMOS transistors and logic designs, but to power and clock them in a resonant, reversible manner.

According to a recent IEEE Spectrum report, Vaire's first prototype – expected to be fabricated in early 2025 – is a simple reversible arithmetic circuit (specifically, a reversible binary adder) embedded in an on-chip LC resonator [4]. The LC resonator provides an oscillating power supply that enables the circuit to recover energy each cycle. Essentially, as the adder computes, energy is exchanged back and forth with the resonator rather than being dumped as heat. This prototype is intended to demonstrate, for the first time in silicon, that a computation (adding two numbers) can be performed while significantly recycling the signal energy. Following that, Vaire

plans a more complex chip: by 2027 they aim to produce an energy-saving processor specialized for AI inference tasks. AI inference yields massive numbers of arithmetic operations (matrix multiplications, etc.), so even small savings per operation could yield large overall benefits in power consumption.

Vaire projects that in the long term (perhaps 10–15 years out), reversible computing techniques could improve energy efficiency by as much as 4,000× compared to today's conventional CMOS logic [4]. In the nearer term, the improvements will be more modest, but still substantial if realized – their 2027 inference accelerator chip is expected to demonstrate a clear advantage in operations-per-joule over standard designs. It's important to note that Vaire's team is not changing the underlying transistor technology, opting for conventional CMOS fabrication. As their CTO Hannah Earley states, "Reversible computing is disruptive enough as it is – we don't want to disrupt everything else at the same time." In other words, they focus on innovating in circuit design and clocking methodology, while leveraging the reliability of existing CMOS processes.

One of the main challenges in their design is how to implement the high-Q resonant clocking on chip. The simplest approach is using an LC tank built from an on-chip inductor and the inherent capacitances of the circuit. This essentially turns the whole circuit into a pseudo-"pendulum" where energy oscillates between electric and magnetic forms. However, on-chip inductors typically have low quality factors (due to resistance and substrate losses), meaning there is still some "friction" that causes energy to be lost each cycle. Vaire's initial prototype uses an LC resonance and will certainly dissipate some energy, but far less than a non-resonant circuit doing the same computation. They are concurrently working on integrating microelectromechanical systems (MEMS) resonators as an alternative clocking mechanisms. MEMS resonators (tiny mechanical oscillators) can have extremely high Q factors, oscillating with very little energy loss. If such a resonator can be coupled to the logic, it could significantly reduce the "friction" in the reversible computing system, allowing more of the computational energy to be recovered each cycle. The drawback is that MEMS devices are harder to integrate with standard CMOS and might oscillate at limited frequencies.

Vaire's effort marks a turning point by moving reversible computing from theoretical analyses and one-off academic experiments into a planned commercial product [4]. It's worth noting that previous academic prototypes laid important groundwork. For example, researchers in the 2000s built small-scale reversible logic circuits (like 1-bit ALUs or gate chains) using adiabatic charging and showed factors of 10–100× reduction in energy for slow operation, validating the principles [3], [5]. Superconducting electronics groups have also experimented with adiabatic switching in Josephson junction circuits to achieve reversible logic (taking advantage of superconductors' zero DC resistance) [13]. However, none of these earlier projects reached the level of a general-purpose computing chip. The hope is that if Vaire's chips succeed, they will demonstrate to the broader industry that reversible computing is not just a laboratory curiosity but a viable strategy for ultra-low-power computing. We may then see more investment and development in reversible processor architectures, perhaps initially for specific niches like low-power cryptography, sensor processors, or AI accelerators where energy efficiency is paramount.

C. Reversible Programming

Designing and using reversible hardware also requires rethinking software. Traditional software is written for von Neumann architectures where operations are irreversible (e.g. a function that produces an output and discards intermediate computations). If one naively runs ordinary programs on a reversible processor, they might undo the benefits by forcing the hardware to do extra work to save and restore information. Ideally, algorithms themselves should be structured in a reversible manner – only erasing information when absolutely necessary (and consciously managing the cost of that erasure).

To facilitate this, researchers have created reversible programming languages – languages in which every computation can in principle be run backwards as well as forwards. One of the earliest such languages is Janus, created by Lutz and Derby [14]. Janus is a programming language with the unique property that its semantics are reversible: given the final state of the program and the program text, one can run the program backward to obtain the initial state. To achieve this, Janus imposes certain constraints: for example, it does not allow uncontrolled creation or deletion of data (no unrestricted heap allocation), and it introduces reversible constructs like the swap operation (exchanging two variables' values without losing information) in place of conventional destructive assignment. A program in Janus essentially performs a deterministic permutation on the state space, so it can be inverted.

Syntax Domains

$p \in \text{Progs[Janus]}$
$x \in \text{Vars[Janus]}$
$c \in \text{Cons[Janus]}$
$id \in \text{Idens[Janus]}$
$s \in \text{Stmts[Janus]}$
$e \in \text{Exps[Janus]}$
$\oplus \in \text{ModOps[Janus]}$
$\odot \in \text{Ops[Janus]}$

Grammar

$$
\begin{aligned}
p &::= d^* \ (\texttt{procedure}\ id\ s)^+ \\
d &::= x \mid x\texttt{[}c\texttt{]} \\
s &::= x \oplus\texttt{=}\ e \mid x\texttt{[}e\texttt{]} \oplus\texttt{=}\ e \mid \\
&\quad \texttt{if}\ e\ \texttt{then}\ s\ \texttt{else}\ s\ \texttt{fi}\ e \mid \\
&\quad \texttt{from}\ e\ \texttt{do}\ s\ \texttt{loop}\ s\ \texttt{until}\ e \mid \\
&\quad \texttt{call}\ id \mid \texttt{uncall}\ id \mid \texttt{skip} \mid s\ s \\
e &::= c \mid x \mid x\texttt{[}e\texttt{]} \mid e \odot e \\
c &::= \texttt{0} \mid \texttt{1} \mid \cdots \mid \texttt{4294967295} \\
\oplus &::= \texttt{+} \mid \texttt{-} \mid \texttt{\^{}} \\
\odot &::= \oplus \mid \texttt{*} \mid \texttt{/} \mid \texttt{\%} \mid \texttt{*/} \mid \texttt{\&} \mid \texttt{|} \mid \texttt{\&\&} \mid \texttt{||} \mid \\
&\quad \texttt{<} \mid \texttt{>} \mid \texttt{=} \mid \texttt{!=} \mid \texttt{<=} \mid \texttt{>=}
\end{aligned}
$$

Fig. 2. Syntax and grammar of Janus [14]. Only reversible mutations are allowed (e.g. x += y, instead of x = x + y)

In the 2000s, Yokoyama and Glück formalized Janus's semantics and built a reversible interpreter and compiler for it [6]. They showed how to systematically generate a dual of any Janus program that undoes its computation. Janus and languages like it demonstrate that writing reversible code is feasible, though it requires a change in mindset. For instance, a function in a reversible language cannot simply return a result and discard local variables – it must somehow pass back any

extra information or use a reversible mechanism to clean it up (such as reversing the function's own computations when they are no longer needed).

Beyond language design, the theoretical underpinnings of reversibility also introduce a shift in how we understand computation itself. Conventional languages are Turing-complete, capable of expressing any computable function. Reversible languages, however, are only r-Turing-complete [15], meaning they can express injective computable functions – a subset of the total computable functions. This might seem like a fundamental constraint, but in practice, any non-injective function can be embedded in an injective one by augmenting it with enough ancillary data (e.g., input history) to make it reversible. Glück and Yokoyama formalize this connection, showing how a conventional program can be transformed into a self-invertible reversible program by either injectivizing the function or tracking computational history explicitly. Rather than limiting what can be computed, reversibility shifts the burden toward preserving and uncomputing information.

Compiler support is crucial for reversible computing to become practical, because manually managing all the information to avoid unintended erasures is tedious and error-prone. A compiler can automatically introduce ancilla bits (extra storage) when needed and later insert operations to uncompute those ancillas (reversibly reset them to 0) once their information has served its purpose [2].

D. Quantum Computing

Quantum computing is often cited as a prominent example of inherently reversible computing. Quantum logic operations are unitary transformations on quantum state vectors – mathematically, these transformations are reversible since unitary operators have inverses (their adjoints) [7]. A quantum gate, such as a Pauli-X (which is effectively a NOT), a controlled-NOT, or a Hadamard transform, never discards information; instead, it rotates or entangles the quantum state in a reversible fashion. The only non-reversible aspects of quantum computing are measurement (when a qubit's state collapses probabilistically and information is lost to the environment) and interactions with a non-ideal environment (decoherence). But the computation itself, as long as it remains quantum, adheres to reversible dynamics. This is one reason quantum computation does not directly hit Landauer's limit in the way classical irreversible logic does – a perfect quantum computer could, in theory, perform computations with very little thermodynamic cost per operation (though in practice maintaining quantum coherence and correcting errors incurs other energy costs).

Many quantum algorithms include a step where a classical function $f(x)$ is computed on a superposition of inputs. To do this without measurement, one implements a reversible version of f as a quantum circuit: typically an operation mapping $|x, 0\rangle$ to $|x, f(x)\rangle$ [7]. This must be reversible, so f is embedded in a larger reversible circuit (using techniques like Toffoli gates to handle any non-invertible behavior). After the computation, sometimes an uncomputation step is used to erase the garbage (ancilla qubits) by running the circuit backward, thus disentangling and zeroing those qubits [2]. These practices in quantum algorithm design directly leverage classical reversible computing principles. Aside from any speedup advantages, the fact that quantum computing must be reversible (until measurement) provides a strong proof-by-existence that large-scale reversible computations are physically possible.

However, one should note that quantum computers are not pursued primarily for energy efficiency – currently they actually consume a lot of power in dilution refrigerators and control electronics. The value of quantum computing lies in its ability to solve certain problems faster than classical computers. That said, if we imagine a future where quantum computers are commonplace, their logic operations (qubit rotations, entangling gates) could be done with extremely low energy dissipation per operation, far below $kT \ln 2$, because they are essentially reversible transformations of a physical state. Some researchers have pointed out that a quantum computer, if error correction overhead is minimized, could execute operations in a thermodynamically reversible manner, giving it an energy advantage for long computations [7].

From the perspective of this paper, quantum computing is a special case that reinforces the importance of reversible logic. The cross-pollination between the fields means advances in reversible computing (e.g., better reversible logic synthesis) directly benefit quantum algorithm implementation, and conversely, quantum computing's success provides a strong motivation to better understand and develop reversible classical computing. In summary, quantum computing demonstrates reversibility on a grand stage – it is a working example of non-trivial computations being done with reversible operations. Any future reversible classical computing devices will likely borrow techniques from quantum computing (without the quantum part) to manage information without losing it.

IV. Feasibility and Challenges

In theory, the potential energy benefits of reversible computing are extraordinary. Practically however, several interlocking challenges determine whether reversible computing will actually become a reality.

A. Energy Measurements

How much energy can reversible computing save today and in the future?

So far, experimental demonstrations of adiabatic and reversible logic have shown energy per operation well below conventional logic when run at low speeds. For example, seen below, 2LAL (a variant of CRL) has achieved a 10× to 100× reduction in energy dissipation compared to its static CMOS counterparts by operating quasi-statically at just a few MHz [5]. 2LAL performs even better at lower Hz, reaching a 100× to 1000× reduction. Reversible superconducting logic prototypes have also indicated potential for vastly lower energy per bit-flip when executed slowly and at cryogenic temperatures [13]. These examples confirm that reversible computing

can substantially reduce CMOS energy. However, the energy levels in these experiments are still far above the Landauer limit – typically 10^3 to $10^5\,kT$ per bit, whereas conventional logic is in the range of 10^6–$10^8\,kT$ per bit [8].

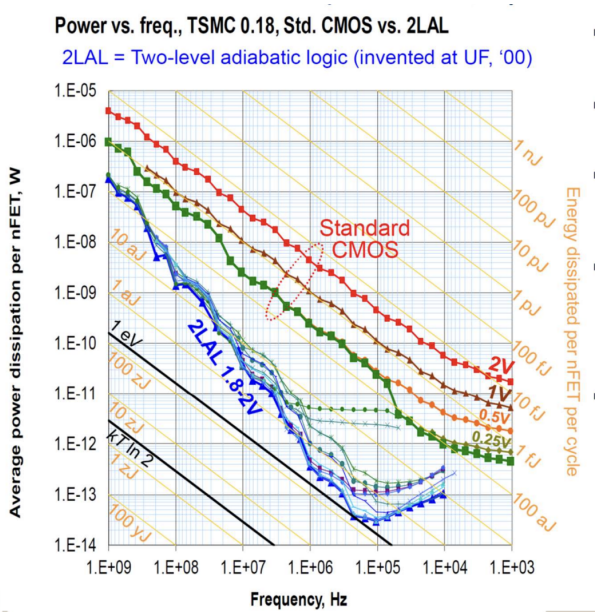

Fig. 3. A comparison between conventional CMOS logic and 2LAL at 180nm [8]

Reversible techniques have chipped away a few orders of magnitude, but there is still a long way to go to approach the theoretical bound. The energy of a bit operation in conventional circuits has been dropping as technology scales (with smaller transistors and lower voltages), narrowing the gap to Landauer's limit [1]. It is not unrealistic to surmise that within a couple of semiconductor generations, a high-performance logic operation might dissipate on the order of $10^4\,kT$ (at room temperature), bringing us closer to a system where Landauer's limit ($kT \ln 2$) is no longer astronomically small compared to the real signal energies.

The relative advantage of reversible computing could become very significant when that day comes: if irreversible logic cannot go any lower without hitting the $kT \ln 2$ barrier, then reversible logic becomes the only way to continue to improve energy efficiency. The work by Frank and others argues that even before hitting that exact limit, architectural and control overheads in conventional designs amplify the effective cost of bit erasures [3], so reversible techniques will be needed to sustain the historical improvement trend in operations per joule.

B. Speed and Throughput

A major trade-off with reversible computing is that making an operation adiabatic (i.e., nearly dissipationless) typically requires slowing it down or involving additional control steps. In general, the energy-delay trade-off is a fundamental consideration [3]. Reversible computing trends toward the extreme of that trade-off: minimal energy change per step, but many steps and long time per operation. If a computation must be done quickly, reversible approaches might have to be run in a more dissipation-heavy mode, losing some of the benefit [8].

For reversible computing to be useful, designs that still offer adequate performance (in operations per second) while significantly reducing energy per operation will be needed. This could either mean operating resonant systems at moderately high frequencies with high Q, or using some form of pipelining and parallelism to compensate for the lower clock rates [3]. The good news is that many energy-constrained computing applications – like IoT sensors, implantable devices, or massive data centers seeking efficiency – are willing to trade some raw speed for energy savings. In such domains, a factor of 10 or 100 in energy efficiency is more valuable than a 2× increase in speed.

Still, if reversible computing techniques limit clock frequency to just a few hundred MHz or a few GHz, it might be challenging to use them in applications that demand very high single-thread performance. A future scenario might be hybrid: critical sections of a computation run irreversibly at high speed, while large bulk computations (especially ones that can be parallelized) run on specialized reversible cores that maximize energy efficiency [8].

C. Overheads

Reversible computations often require extra bits (ancilla or garbage bits) to store intermediate information that would otherwise be thrown away [2]. Likewise, reversible logic gates typically have more wiring and sometimes more transistors than their irreversible counterparts [3]. This raises concerns about area overhead and signal routing complexity. A straightforward reversible implementation of an n-bit adder might output n sum bits and need to carry along the n input bits, effectively doubling the width of the datapath unless one employs techniques to uncompute carries. Similarly, memory operations pose a dilemma: a standard RAM write destroys the previous bit stored at that address – a reversible computer would need to somehow remember that overwritten bit (perhaps by copying it out somewhere rather than truly destroying it) [8].

Some proposals for reversible architectures include dual-rail memory (store both the data and its inverse or a history log) or reversible update schemes, but this inherently has an $O(ST)$ space requirement, where T is the process time and S is process space [3]. Bennett has also defined a recursive checkpointing algorithm, which operations before recursively selected checkpoints are uncomputed at intervals [8], reducing space to something more like $O(T \log S)$ space. However, the impact of such overheads on a full system (CPU + memory + interconnect) is still an open question. If the overhead is too large, the purported energy savings might be partially or fully eaten up by the energy cost of managing all the extra bits or by a reduction in computational density (operations per second per unit area).

Research in reversible computing attempts to minimize these overheads – for instance, using optimizing compilers that reuse garbage space and reversible algorithms that minimize space-time overhead can reduce the bloat. In recent years, there have been encouraging results on reducing the ancilla count for common computations and on design patterns for reversible computations that are space-efficient [8]. Nonetheless, when comparing a reversible architecture to an irreversible one, one must account for the fact that the reversible one might need 2–3× the number of logic gates to do the same job (even though each gate uses far less energy). Whether this is a good trade-off may depend on the application and on how energy-constrained it is.

D. Error Accumulation

By avoiding the "refresh" of signals at each logic gate (which is what irreversible logic does), reversible circuits risk accumulating errors or noise over time. In an classic circuit, even if a bit arrives with a slight error (0.1V off in analog value), the next gate will restore it to a full 0 or 1.

In a fully reversible circuit, however, if a bit's signal degrades, there is no automatic restoration since no new energy is used to regenerate it; the signal just gets carried forward through reversible transformations [3], possibly even mixing with other signals. This is akin to analog computing or chained computations without normalization, drift or noise will build up.

To manage this, reversible computing might still need occasional error correction, which can be done in a reversible manner with some overhead [2]. Environmental interactions are another issue: a truly reversible process in a mathematical sense can still dissipate energy if the system is not perfectly isolated. Any interaction that leaks information to the environment, like a photon emitted or a phonon (thermal vibration) generated, will produce entropy [1], [8]. Reversible computing systems need to be carefully shielded, and designed such that these interactions are minimized.

This is somewhat analogous to quantum computing, where isolation and error correction are key concerns, except that in reversible classical computing the states are classical, so error correction is potentially easier. The bottom line is that keeping a real physical system nearly reversible over many operations might require fighting against naturally occurring friction, resistance, leakage currents, radiation, and so on. Each of those adds a little entropy, and engineering them all away is a formidable task [3].

E. Outlook

The jury is still out, but trends suggest that reversible computing will gradually grow in importance. In the near term (next 5–10 years), we are likely to see reversible computing remain somewhat niche: specialized chips from startups or research labs demonstrating impressive energy efficiency on certain tasks, possibly only used in environments where power is a top priority. For example, a reversible computing module might find use in a spacecraft or a remote sensor where energy harvesting only provides a limited power budget but computing needs are significant.

If Vaire Computing's plan succeeds, their 2027 reversible AI inference chip could be a catalyst, especially for the AI industry which is hungry for more efficient inference accelerators to reduce data center power consumption. A 1000× improvement in energy per operation would be game-changing for large-scale deployments (imagine running a server farm on 1/1000th the energy – the economic and environmental implications are huge).

In the longer term (10–20+ years), whether reversible computing becomes mainstream will depend on several factors. One is how quickly conventional technology approaches the energy wall. If CMOS and related technologies keep improving and we can squeeze out more efficiency without needing reversibility, the outlook would be excellent. However many experts believe that we are already seeing diminishing returns in energy efficiency from conventional methods of scaling.

Another factor is the development of design automation for reversible computing. The wider engineering community will only adopt reversible logic if there are robust tools to design, verify, and program such systems. The current work on reversible compilers, languages, and logic synthesis needs to mature into something as user-friendly as today's digital design flows. There is also a cultural shift: computer architects and developers would need to embrace a new paradigm, which can take time.

However, recall that quantum computing was once viewed as a very distant, exotic idea, but due to clear advantages, it has seen enormous investment and progress in the last decade. Reversible computing might see a similar trajectory if the need for energy-efficient computing becomes even more critical. Already, major semiconductor roadmaps (like the IEEE's IRDS – International Roadmap for Devices and Systems) have included reversible computing as a potential emerging technology to watch [16], in acknowledgment that new logic paradigms might be required to continue performance scaling under power constraints. Some government research programs are funding projects on reversible and energy-efficient computing, sometimes in tandem with cryogenic or superconducting computing research.

A balanced view might say that reversible computing is likely to play a significant role in the future of computing, but not in isolation. It may be combined with other technologies – for instance, one might see a hybrid classical/quantum processor where the classical part is made reversible to minimize heat generation alongside qubits, or a memory hierarchy where data movement (a big contributor to energy use) is managed by reversible circuits to cut down interconnect power.

Over the next decades, we might see incremental adoption: first in special accelerators and research processors, then in wider applications if the former prove their worth. If insurmountable obstacles were to arise (we can't scale reversible logic to high complexity or the cost savings never offset the overhead), reversible computing could remain a specialized solution. But if even a subset of computing tasks – cryptographic

hashing, AI matrix operations, or big data processing – can be drastically improved by reversible techniques, that will ensure the approach stays relevant and increasingly utilized.

V. Conclusion

Reversible computing, once a mere theoretical curiosity, is steadily moving towards a practical reality. By eliminating the fundamental energy cost of information loss, reversible computing holds the promise of breaking through the thermodynamic limits that govern today's computing technologies.

Reversible computing has the potential to transform mainstream computing in the realm of energy efficiency, which is arguably as important as raw performance in our current era (considering that power and thermal limits are what constrain many modern CPU and GPU designs). If current trends hold, reversible computing could enable continued improvements in computational capability per watt, extending the trajectory that has driven computing progress for decades.

In an optimistic scenario, future processors might incorporate reversible logic units that perform massive computations with only negligible heat generation, making possible computation that today would melt any silicon chip. Data centers could carry out intensive tasks with minimal energy draw, and small battery-powered devices could run complex algorithms without quickly draining their battery.

However, the path is not without obstacles. The engineering challenges – ensuring adequate speed, managing complexity, and dealing with noise – mean that reversible computing might not replace conventional computing entirely but rather complement it. It is likely to first find niches where its advantages outweigh its costs. Over time, as techniques improve and if energy efficiency remains a dominant concern, its use could broaden. The history of computing teaches us that paradigm shifts (like the move from vacuum tubes to transistors, or from single-core to multi-core architectures) take time and often happen when the existing approach can no longer cope with the demands placed on it. Reversible computing may well be the paradigm shift that computing turns to as we approach the fundamental limits of current technology.

While it's too early to declare reversible computing the new norm, it is certainly poised to be a transformative influence. Even if it doesn't entirely replace irreversible logic, it will push designers to reconsider how computation is done at the most basic level, with energy considerations front and center. In an era where "greener" and more efficient technology is desperately needed, reversible computing offers a physically grounded path toward sustainable computing growth. The coming years (and perhaps decades) will be critical in determining how far and how soon this promising approach will take us.

References

[1] R. Landauer, "Irreversibility and heat generation in the computing process," *IBM Journal of Research and Development*, vol. 5, no. 3, pp. 183–191, 1961.

[2] C. H. Bennett, "The thermodynamics of computation–a review," *International Journal of Theoretical Physics*, vol. 21, no. 12, pp. 905–940, 1982.

[3] M. P. Frank, "Introduction to reversible computing: Motivation, progress, and challenges," 2005.

[4] D. Genkina, "Reversible computing escapes the lab in 2025," *IEEE Spectrum*, vol. 62, no. 1, pp. 32–37, January 2025.

[5] S. Younis, "Asymptotically zero energy computing using split-level charge recovery logic," Ph.D. dissertation, Massachusetts Institute of Technology, 1994.

[6] T. Yokoyama and R. Glück, "A reversible programming language and its invertible self-interpreter," in *Proceedings of the 2007 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation*, 2007, pp. 144–153.

[7] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge University Press, 2010.

[8] M. P. Frank, "Current status of reversible computing," April 2022, oSTI Identifier: 1869234.

[9] T. Toffoli, "Reversible computing," *Automata, Languages and Programming*, pp. 632–644, 1980.

[10] E. Fredkin and T. Toffoli, "Conservative logic," *International Journal of Theoretical Physics*, vol. 21, no. 3, pp. 219–253, 1982.

[11] C. H. Bennett, "Logical reversibility of computation," *IBM Journal of Research and Development*, vol. 17, no. 6, pp. 525–532, 1973.

[12] J. D. and R. Sivakumar, "Design and analysis of power efficient binary content addressable memory (pebcam) core cells," *Circuits, Systems, and Signal Processing*, vol. 37, 04 2018.

[13] R. M. Lewis, M. P. Frank, M. D. Henry, and N. A. Missert, "Asynchronous ballistic reversible computing using superconducting elements," no. SAND2020-10332, September 2020.

[14] C. Lutz, "Janus: a time-reversible language," 1986, *Letter to R. Landauer*.

[15] R. Glück and T. Yokoyama, "A reversible programming language and its invertible self-interpreter," *Theoretical Computer Science*, vol. 925, pp. 64–81, 2022.

[16] IEEE IRDS, "2022 ieee international roadmap for devices and systems (irds)," Online; accessed 2024-04-01, 2022, https://irds.ieee.org/roadmap-2022.