What is Quantum Parallelism, Anyhow?

Stefano Markidis
Computer Science Department
KTH Royal Institute of Technology
Stockholm, Sweden

***Abstract*—Central to the power of quantum computing is the concept of quantum parallelism: quantum systems can explore and process multiple computational paths simultaneously. In this paper, we discuss the elusive nature of quantum parallelism, drawing parallels with classical parallel computing models to elucidate its fundamental characteristics and implications for algorithmic performance. We begin by defining quantum parallelism as arising from the superposition of quantum states, allowing for the exploration of multiple computational paths in parallel. To quantify and visualize quantum parallelism, we introduce the concept of quantum dataflow diagrams, which provide a graphical representation of quantum algorithms and their parallel execution paths. We demonstrate how quantum parallelism can be measured and assessed by analyzing quantum algorithms such as the Quantum Fourier Transform (QFT) and Amplitude Amplification (AA) iterations using quantum dataflow diagrams. Furthermore, we examine the interplay between quantum parallelism and classical parallelism laws, including Amdahl's and Gustafson's laws. While these laws were originally formulated for classical parallel computing systems, we reconsider their applicability in the quantum computing domain. We argue that while classical parallelism laws offer valuable insights, their direct application to quantum computing is limited due to the unique characteristics of quantum parallelism, including the role of destructive interference and the inherent limitations of classical-quantum I/O. Our analysis highlights the need for an increased understanding of quantum parallelism and its implications for algorithm design and performance.**

***Index Terms*—Quantum Computing; Quantum Parallelism; Quantum Data Parallelism; Fork–Join Parallelism; Amdahl's Law; Gustafson's law**

I. Introduction

Parallel computing is the simultaneous execution of threads or processes, constituting the fundamental units of computational work. In classical parallel computing, threads/processes are carried out on distinct and often specialized computational units, such as Computing Processing Units (CPU) and Graphics Processing Units (GPU) cores, executing operations serially. Historically, two different forms of parallelism have been categorized, each effectively aligned with specific problem types and formulations: *task* and *data* parallelism. Task parallelism involves the concurrent execution of potentially diverse tasks on distinct data sets, whereas data parallelism consists of executing the same task on different dataset items. Examples of these two kinds of parallelisms include pipeline parallelism, commonly employed in modern pipelined processors and Operative Systems (OS), as task parallelism, and data-parallel deep-learning training workloads, wherein identical feed-forward and back-propagation operations are performed on different training data.

Similarly, quantum computing exploits parallelism, performing calculations in the superposition of states, yet in a very distinctive and unique way. Differently from classical silicon- and electronics-based computing systems, consisting of thousands of cores completing serial tasks, a quantum computing core is inherently an elementary parallel computing unit [1], [2]: calculations can be performed in parallel by unitary transformations acting on a superposition of quantum states [3]–[6]. The READ (measurement) and WRITE (initialization) operations remain serial I/O operations as they bridge the quantum to the classical world. In this article, we explore the concept of quantum parallelism, discussing its distinctive fundamentals that render quantum parallelism such an elusive yet fascinating concept.

While the benefits and computational speed-up of parallel computing are nowadays evident and accepted, at the beginning of the first parallel computing systems, an intense diatribe of whether or not any worthwhile benefit in using parallel computing exists, e.g., what is the worth of designing more expensive and complex systems if the returned speed-up or improvement is minimal? At the basis of this discussion, parallelism and its relation to speed-up must be understood. Two influential laws, Amdahl's [7] and Gustafson's [8], [9], emerged from this controversial debate on parallelism's usefulness. The most famous Amdahl's law is summarized by the consideration that the parallel speed-up is fundamentally plagued by the algorithm serial part that asymptotically will lead to a diminishing speed-up performance. Amdahl's law is an example of the first critical reflection on parallel computing advantage. In a more positive view supporting classical parallelism, Gustafson's law responds, pointing out that an increased computational workload for each parallel task can solve larger problems and improve the speed-up.

Today, the scientific community is intensifying discussions on the advantages or the practicality of quantum computing concerning classical computing and whether, in practice, the reach of quantum advantage is achieved, will be achieved, or - in an overly pessimistic position - will ever be achieved [10]–[14]. The classical-quantum I/O bottleneck to input quantum data and need for super-quadratic speed-up are the major concerns [12]. This work aims to contribute further to this discussion by analyzing the nature of quantum parallelism and how it is used in its application, reconsidering basic classical parallelism strategies and laws, and reformulating them in the quantum context. The basic fundamental research question is: *what are the insights of applying classical parallelism con-*

cepts and laws to quantum computing in terms of performance, speed-up, and advantage?

To answer this question, we first discuss the nature of quantum parallelism. Using categories and terminology typical of classical computing, we then reconsider quantum parallelism from the classical parallelism perspective and map these concepts to quantum computing. Throughout the article, we rely on using quantum circuit and gate abstractions and pure state descriptions in an ideal set-up (no noise). This conceptual framework allows us to avoid low-level quantum computing hardware concepts, such as pulses and quantum gate implementation. To identify the limits of quantum parallelism, such as the serial part in the classical parallelism laws, we borrow a technique from classical computing to quantify the amount of parallelism. This technique relies on expressing an algorithm and its computation as a graph and then identifying the graph's critical path length [15]. We use quantum dataflow diagrams [16] to represent the quantum state superposition evolution. To show the limitations of basic quantum parallelism, we show two exemplary cases of quantum algorithms, the Amplitude Amplification (AA) [17] and Quantum Fourier Transform (QFT) [18] primitives. We show that quantum algorithms exhibit different amounts of quantum parallelism during the execution, depending on the problem input and the amount of interference. We finalize the paper by analyzing the validity of Amdahl's and Gustafson's laws in light of the results of this work.

The contributions of this work are the following:

- We provide an analysis and summary of the fundamental concepts of quantum parallelism. By discussing the fundamental differences between classical parallelism and its quantum counterpart, we offer insights into their distinct usages in applications.
- We introduce a methodology for characterizing and quantifying quantum parallelism in applications. Leveraging graph-based representations akin to classical dataflow diagrams, we enable precise quantification of parallelism. This methodology is applied to characterize the quantum parallelism inherent in the Quantum Fourier Transform (QFT) and Amplitude Amplification (AA) algorithms.
- We discuss extensions of the classical concepts of Amdahl's and Gustafson's laws to quantum computing.

II. The Concept of Quantum Parallelism

Before delving into the discussion of quantum parallelism in classical terms, it is essential to understand its fundamental nature and how it differs from classical parallelism. This understanding will help us grasp quantum parallelism's usage and potential advantages. The first pioneering work on establishing the foundation of quantum computing by Paul Benioff and Richard Feynman [19]–[22] was deeply involved in establishing computing as a physical process, discussing reversibility and dissipation of quantum gates, and exponential growth of resources and did not explicitly mention quantum parallelism. Instead, to our knowledge, the concept of quantum parallelism was first introduced by David Deutsch in 1985 in a seminal paper [23] and further developed by David Deutsch, Richard Jozsa, and Artur Ekert [3], [24]–[26].

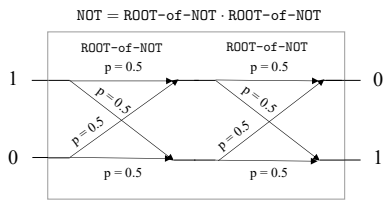

Fig. 1. A classical deterministic NOT operation can be decomposed in two quantum `ROOT-of-NOT` operations, exploiting quantum parallelism. p denotes the probability to transition from state 0 to 1 and vice versa. Note that quantum probability addition (instead of classical probability addition) results in a deterministic output. The computation is quantum parallel as it allows for a superposition of states 0 and 1 as execution paths.

Since its inception, quantum parallelism has been the capability of computing systems to perform parallel computation, exploiting the quantum systems' capability of being in a superposition of states. The first articles on the fundamentals of nature quantum parallelism highlight two important points. First, that computing, intended as a physical process, is inherently quantum parallel. To convince ourselves, we can go back to the example of implementing a classical NOT on a classical bit as a sequence of two `ROOT-of-NOT` (see Fig. 1), as discussed in Refs. [1], [27]. In fact, this classical deterministic NOT operation on a classical bit can be implemented using quantum parallelism with two quantum `ROOT-of-NOT` operations (half-NOT operations) and produce a classical output. Even the most basic example of one qubit and NOT operation can exploit parallelism when framed in a quantum computing formulation that is a more fundamental, powerful, and comprehensive theory of classical computing.

A second important point to note, especially for readers from High-Performance Computing (HPC), is that quantum parallelism, in its simplest form, does not directly lead to computational speed-up. Rather, it is a fundamental feature of quantum computing systems. The true source of speed-up lies in the inherent *quantumness* of superposition, which manifests in several critical aspects, summarized as follows:

1. Quantum parallelism follows the principles of interference, which can either be destructive or constructive. Interference phenomena are rooted in using the squared amplitude of complex numbers to denote probability in quantum computing. A qubit, the fundamental unit of information in two-state quantum systems, can exist in one of two states upon measurement: $|0\rangle$ or $|1\rangle$, corresponding to the standard or computational basis states. Here, $|0\rangle$ represents classical bit 0 and $|1\rangle$ represents classical bit 1. A qubit ψ can also exist in a superposition state, expressed as $\psi = \alpha\,|0\rangle + \beta\,|1\rangle$, where α and β are complex numbers. These complex numbers can be expressed either in Cartesian form, such as $a + bi$ (where a and b are the real and imaginary components, respectively), or in polar form, $\rho\exp(i\phi)$, where ρ is the magnitude and ϕ is an angle. In quantum mechanics, the probabilities of measuring the

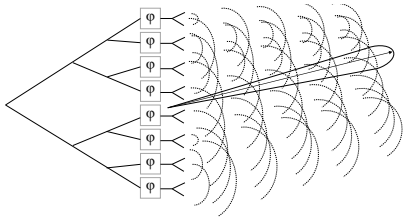

Fig. 2. Some of the original quantum algorithms were inspired by antenna arrays, where several antennae emit electromagnetic waves with different phases and amplitudes to create directed single or multiple beams. Parallelism arises from utilizing the signal from different antennas. Similarly to quantum computing, beamforming employs parallelism and interference phenomena to cancel wave propagation in undesired directions.

quantum states $|0\rangle$ or $|1\rangle$ are given by α^2 and β^2, respectively. When a transformation results in a qubit with $\alpha \neq 0$ and $\beta \neq 0$, quantum parallelism is utilized with two quantum threads running simultaneously. The use of complex numbers influences how these parallel threads interact: probabilities of quantum threads do not obey the additive rule of classical probability laws; instead, they can cancel out, decreasing the probabilities, via destructive interference. An example of this is evident in the second `ROOT-of-NOT` transformation in Fig. 1, where parallelism reduces from two threads to one. Quantum algorithms designed to detect solutions among a limited number of answers must exploit destructive interference to identify the correct answer.

There exists a connection between quantum computing and electromagnetics and antenna theories. It is unsurprising that some quantum algorithms draw inspiration from techniques used in beamforming with antenna arrays [28]: manipulating phases and amplitudes to direct electromagnetic beams in specific directions. Similarly, quantum computing can manipulate the phases and amplitudes of quantum states to achieve the correct solutions through interference. The important point lies in the fact that quantum parallelism is essentially an interference pattern resulting from the interaction of quantum states: each quantum state contributes to a collective interference pattern analogous to the combining of signals in antenna arrays. However, unlike classical parallelism, where multiple independent processes run concurrently, quantum parallelism is characterized by a complex web of interference. The notion of reduced quantum parallelism highlights the subtle balance between constructive and destructive interference in quantum algorithms. While constructive interference amplifies certain computational pathways, destructive interference selectively suppresses others, ultimately guiding the system toward the correct solutions. Thus, optimizing interference within quantum algorithms emerges as a critical optimization strategy akin to fine-tuning the amplitude and phase relationships in beamforming applications [29].

We might then ask: what is the main difference between classical and quantum interference? The difference lies within the nature of quantum mechanics, where we leverage the complementary nature—also known as *duality*—of particles capable of exhibiting both wave-like and particle-like behavior [30]. This unique property of quantum systems is not exploited in classical systems. In quantum physics, particles such as electrons and photons can manifest as waves under certain conditions, leading to interference phenomena. Even larger entities like atomic nuclei and molecules exhibit wave-like behavior and interference and can be exploited in quantum computing systems to perform computations.

2. Quantum parallelism exploits the composition of elementary probabilistic systems. As highlighted in one of the seminal articles on quantum computing by Richard Feynman [21], constructing quantum computing systems involves composing probabilistic systems, leading to an exponential increase in the quantum states of the composite system. Consider two probabilistic systems which do not need to be quantum: A with two possible states ($[A_0, A_1]$) and an associated transition or adjacency matrix M_A, and B with three possible states ($[B_0, B_1, B_2]$) and an associated transition or adjacency matrix M_B [31]. The combined states are obtained by taking the tensor product of the two systems' states: $[A_0B_0, A_0B_1, A_0B_2, A_1B_0, A_1B_1, A_1B_2]$. Here, A_0B_0 represents the probability of concurrently observing A in state A_0 and B in state B_0, and so forth for other elements of the tensor product. Similarly, the combined transformation associated with the assembled state is obtained by taking the tensor product of the adjacency matrices $M_A \otimes M_B$. The tensor product allows for the assembly of probabilities across different systems. When employing two-state systems, such as qubits, the assembly of N elementary systems results in a combined state with a size of 2^N, and transitions between different states are expressed by a $2^N \times 2^N$ matrix system. If we used only one classical non-probabilistic system, such as an antenna array, we would require 2^N antennae. Instead, we can leverage the assembly properties of probabilistic systems to scale available resources exponentially. This aspect of quantum parallelism is intriguing: the quantum parallelism available in a multi-qubit system scales exponentially with the number of qubits, with a 300-qubit system offering parallelism greater than the number of particles in the entire universe.

An important insight into the power of quantum computing arises from the so-called *principle of local operations*, as discussed in Refs. [26], [32], [33], wherein a local transformation to a qubit is carried out throughout the entire system. Specifically, the operation U on a certain qubit in a multi-qubit system is expressed as $I \otimes ... \otimes I \otimes U \otimes I .... \otimes I \otimes I$, where I is the identity matrix. See Fig. 3 for a visual representation. This single operation, applied across the entire system, would necessitate the product of a $2^N \times 2^N$ matrix with a vector of size 2^N on a classical computer. While the exponential expansion of space arises from the probabilistic composition of single states, it is crucial to note a significant distinction between classical probabilistic systems and quantum computing systems. In classical probabilistic systems, all events, such as transitions to different states, are mutually exclusive, whereas quantum computing systems simultaneously explore and process multiple computational paths. The assembly of elemental quantum computing systems can thus be viewed as a form of

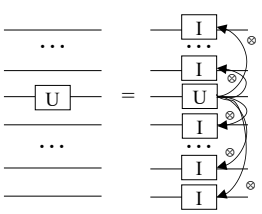

Fig. 3. A local gate operation U on a qubit is effectively carried out across all 2^N quantum states. While U corresponds to a 2×2 matrix operation in a single-qubit system, it extends to a $2^N \times 2^N$ matrix operation in a multi-qubit system. This principle is often referred to as the *principle of local operations*.

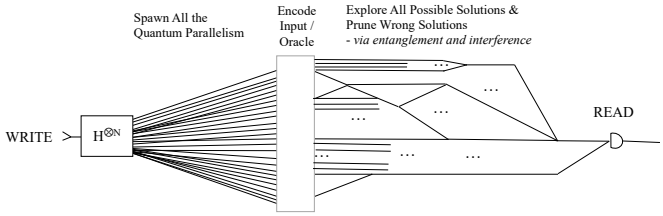

Fig. 4. Diagram illustrating a prototypical quantum application workflow. Traditional quantum algorithms typically begin by initializing a classical state, followed by the parallel spawning of quantum parallelism by applying Hadamard gates ($H^{\otimes N}$). Subsequently, the input data is encoded, typically in the quantum state's amplitude and phases, or an oracle is applied. Computational processes then occur in superposition before concluding with a READ operation (measurement). Notably, while the initial phase of the algorithms maximizes quantum parallelism, extracting meaningful results often relies on pruning erroneous outcomes via destructive interference.

superposition creating quantum parallelism. Another critical difference stems from using complex numbers rather than real numbers to express probabilities. In quantum computing, transitions are described by unitary transformations, which, along with reversibility, ensure that the total probability sums to one (unitary transformations *preserve the geometry* of the space in which they operate).

Finally, another significant consequence of assembling basic quantum computing units is evident when considering classical interconnects, such as current HPC networks. Without leveraging probabilistic computing, they cannot exploit exponential parallelism growth. Instead, a quantum interconnect is needed to exponentially increase the system's available quantum parallelism.

3. Entanglement is crucial for quantum parallelism to explore states and threads that are otherwise inaccessible. As discussed previously, the composition of single-qubit gates generates a state vector that can be produced by composing single-qubit states via the tensor product, thereby excluding states that cannot be represented in this manner. These excluded states, known as *entangled* states, necessitate an entanglement operation for their creation. The fundamental operations that provide entanglement are the controlled operations.

An important aspect concerning entangled states and classical HPC systems, particularly in their role as quantum computer simulators, is that quantum entangled states pose challenges for representation on classical computer systems, requiring exponential resource growth. While non-entangled states, or *separable* states, can be encoded in basic single qubits, allowing for the reconstruction of the full state vector via the tensor product, this encoding method fails when representing an entangled quantum state. This challenge is also evident in tensor-network-based quantum computer simulators, which can efficiently compress quantum states with low entanglement but struggle with expressing quantum systems with high entanglement [34].

Multi-Core Quantum Computing. A multi-qubit system can be likened to a quantum single-core capable of providing 2^N quantum parallelism. However, the effort to pack and scale more qubits into a monolithic quantum core presents technological challenges. These challenges include dense wiring requirements for control systems, increased crosstalk between qubits, and the necessity for increased qubit connectivity to enable multi-qubit gate operations, among others. To address the obstacle of scaling the number of qubits, a viable solution is to adopt modular or multi-core quantum architectures [35], wherein different quantum single-cores are interconnected via a quantum network. For example, to realize a 200-qubit Quantum Processing Unit (QPU), ten 20-qubit quantum cores could be combined, circumventing the complexity associated with implementing a single 200-qubit core, which is technologically challenging.

The underlying mechanism of quantum networks and communication relies fundamentally on entanglement. While entanglement in quantum parallelism enables the exploration of quantum state configurations inaccessible via single-gate quantum transformations, in quantum networks, entanglement – in particular, Einstein–Podolsky–Rosen (EPR) pairs and quantum teleportation – enables intimately correlated qubits and allows communication to sidestep the constraints of the non-cloning theorem. Analogous to classical computing, inter-core communication in quantum systems is time-consuming and necessitates additional resources, such as dedicated quantum state communication cores (distinct from computational quantum cores) and qubit highways [36]. Moreover, quantum networks are highly sensitive to latency, given the limited lifetime of qubits. Consequently, performance models have been developed to evaluate quantum communication costs across the network [37], and message-passing programming models [38], [39], including collective communication primitives, have been devised to program quantum distributed algorithms. While our discussion primarily focuses on quantum single-core parallelism, the framework presented in this paper can be extended to encompass quantum distributed applications.

III. The Role of Quantum Parallelism in Quantum Applications

When developing quantum algorithms that, by intrinsic nature, are quantum parallel, it is important to rethink how quantum parallelism is used in quantum applications. In fact, quantum algorithms exploit parallelism very differently than

classical ones. Classical parallelism is set to do useful work: each thread is set to do an operation that contributes to the overall solution of the problem, e.g., the problem's solution in a certain sub-domain of the overall problem's domain. Instead, quantum algorithms have been set to work differently. A prototypical quantum application is shown in Fig. 4. They exploit Michelangelo's *forza of levare* (means of removal): they spawn maximum parallelism available to explore all the solutions (even the wrong ones) and prune to a small number of threads containing the correct solutions via quantum interference. Quantum algorithms consist of three fundamental phases: *i)* All the available quantum parallelism is spawned out of a classical input (WRITE), typically using parallel execution of `H` gates and the input is encoded via an encoding or an oracle is queried exploiting all the available quantum parallelism. *ii)* All the possible solutions are explored via quantum parallelism. The amplitude of correct solutions is amplified via constructive interference, while the amplitudes of incorrect solutions are reduced via destructive interference. Entanglement allows us to explore a larger number of possible solutions that are not expressed by using only single-qubit gate operations. *iii)* A READ or measurement operation is carried out to collapse the superposition to a classical case. Small quantum parallelism before the measurement is a convenient feature for many quantum applications, as it will result in fewer outcomes.

Current Noisy Intermediate-Scale Quantum (NISQ) [40] algorithms, such as variational quantum eigensolvers [41], [42] and, in general, parametrized quantum circuits [43], do not follow the traditional quantum application pattern, as exemplified in Fig. 4. NISQ applications are slightly different as they typically do not include initialization of the application that brings the quantum systems into a full equal superposition of the available quantum parallelism. Instead, the initial superposition is provided by the encoding of the input into the quantum states: phase (or angle) encoding and amplitude encoding have the potential of using all the available superpositions, while basis state encoding provides only limited quantum parallelism. In particular, the NISQ algorithms lack the initial phase, where the available quantum parallelism is spawned via `H` gates. Instead, the superposition creation depends on the particular encoding scheme or feature mapping and usage of `ROTX(`θ`)` and `ROTY(`θ`)` quantum gates (and their controlled versions) in the *ansatz* or parametrized quantum circuit [44]. Because superposition is connected with the encoding scheme, there is likely a link between quantum parallelism and the expressivity of certain quantum neural network architecture [45]–[47].

IV. Quantum Parallelism in Classical Parallelism Terms

If we identify the number of quantum states in superposition in the computational basis as a metric of quantum parallelism, then quantum parallelism can be viewed as a form of *data parallelism*. This analogy arises from the fact that quantum transformations operate simultaneously on the superposition states. Notably, quantum parallelism aligns with the concept of SIMD (Single Instruction Multiple Data) parallelism, where a single instruction operates on multiple processing elements, each handling distinct data elements [48], [49]. As discussed previously, traditional quantum application algorithms typically undergo an initial phase where they harness all available parallelism, compute in superposition, manipulate phases, and perform measurements. During this initial phase, the quantum algorithm maximizes parallelism. However, parallelism diminishes as the application progresses due to interference phenomena, reducing the quantum parallelism.

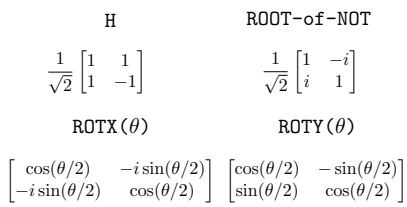

Fig. 5. Examples of single-qubit gate operations capable of generating quantum parallelism from a classical state, where no superposition exists. In classical terms, these operations correspond to a fork operation, akin to spawning tasks across multiple processing units.

In any quantum algorithm, mechanisms for spawning quantum parallelism from a classical input state and mechanisms for reducing data parallelism to identify the correct answer are essential. In classical parallelism, these operations are enabled by fork-join operations. For instance, in classical fork-join paradigms, OpenMP achieves this through opening and closing parallel regions [50], while Cilk utilizes `spawn` and `sync` operations [51]. In quantum computing, the fundamental mechanism for creating an equal superposition of basis states, starting from pure basis states, involves using quantum gates with associated dense transformation matrices that yield dense outputs. Examples of such quantum unitary transformations are depicted in Fig. 5. The Hadamard gate (`H`), widely regarded as the gateway to quantum parallelism, is the most renowned and ubiquitous gate. Additionally, the previously mentioned `ROOT-of-NOT` gate, as well as the `ROTX(`θ`)` and `ROTY(`θ`)` gates, find extensive application in parametrized quantum circuits and variational quantum eigensolvers.

All single-qubit gate operations illustrated in Fig. 5, along with their associated controlled versions, can generate parallelism from a classical state via quantum interference. The matrices corresponding to these transformations are dense. These operations correspond to a *fork* operation in the classical parallelism context. Due to the unitarity of quantum gates, it is always possible to revert from a superposition of states to a classical state – transitioning from parallel quantum threads to a single classical thread – by applying the inverse gates. For example, in the case of `ROTX(`θ`)` and `ROTY(`θ`)`, rotating the quantum state by a $-\theta$ angle enables the return to the classical state, thereby reducing the superposition to a single classical state. This reduction in data parallelism via the inverse operation is called the *join* operation. The

reduction operation corresponds to a destructive interference phenomenon, resulting in the vanishing probability of a particular path. The H gate is arguably the most prominent bridge for transitioning back and forth between computational basis states and an equal superposition thereof. Although the H gate can be constructed atop ROTX(θ) and ROTY(θ) gates if not implemented as a standard basic gate, it differs from ROTX(θ) and ROTY(θ) in that it serves as its inverse. This characteristic grants the H gate a critical role in all quantum algorithms: the same gate can perform fork and join operations.

In this work, we introduce the concept of measuring quantum parallelism using quantum dataflow diagrams, analogous to classical Directed Acyclic Graphs (DAGs) used in parallel computing [15] [1]. Two key metrics quantify classical parallelism: work (T_W) and span or critical path length (T_∞). Work, T_W, represents the total number of nodes required to execute all operations in the graph, while span, T_∞, represents the longest path of dependencies in the graph in terms of nodes. The classical parallelism P is then calculated as $P = T_W/T_\infty$. To extend this concept to quantum computing, we introduce the quantum dataflow diagram, a graphical tool similar to DAGs but tailored for quantum systems [16]. This diagram represents a weighted graph where vertices correspond to quantum operations, and edges symbolize quantum data movement. By leveraging this representation, we can visualize and quantify quantum parallelism. In the quantum dataflow diagram, each quantum thread is weighted by its probability throughout the execution of different quantum gates. The quantum parallelism quantity enables us to calculate the quantum parallelism efficiency as the fraction of quantum parallelism over the total available parallelism ($\eta_P = P/2^N$), as well as the efficiency of destructive interference in determining correct answers ($\eta_{DI} = 1 - \eta_P$).

To explain the concept of quantum dataflow diagrams, we can focus on expressing the basic fork–join operation obtained with the H gate acting on a qubit. This is shown in Fig. 6. Note that the H gate exhibits distinct behaviors when applied to the $|0\rangle$ and $|1\rangle$ states, manifesting as a phase difference while sharing identical amplitudes. On the left side of Fig. 6, theH gate can bring a computational basis state, $|0\rangle$ or $|1\rangle$, into an equal superposition of them. As discussed before, differently from the ROTX(θ) and ROTX(θ), it is essential to note that the inverse of the H gate is H itself and therefore it can both perform both fork and join operations.

While quantum dataflow diagrams can be manually designed for simple circuits, more complex circuits require simulation tools like Qiskit [52] or Cirq [53], [54] to determine state vectors in polar coordinates, representing amplitudes and phases. The development of automated tools within quantum computing software frameworks can streamline the construction of quantum dataflow diagrams for intricate quantum circuits.

[1]Charles Leiserson, *What the $#@! is Parallelism, Anyhow?* https://www.cprogramming.com/parallelism.html

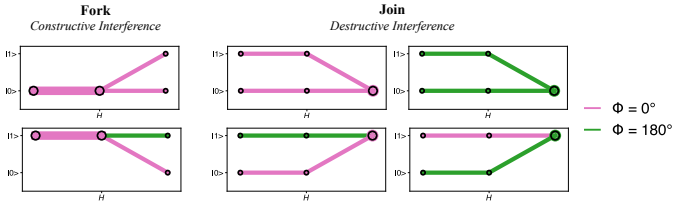

Fig. 6. The H gate acts as a fork–join operator that allows for creating superposition (fork) and *merging* two quantum states via destructive interference (join). The H gate is the key gate for creating quantum data parallelism with fork-join operations.

V. Quantum Dataflow Diagrams Use Cases

In this section, we apply the methodology of the quantum dataflow diagrams to two important quantum primitives, the QFT and the AA iteration.

A. Quantum Fourier Transform

The first example of quantum algorithms we want to analyze for quantum parallelism is the QFT. This method was first devised in 1994 by David Coppersmith [55]. QFT allows us to calculate the Discrete Fourier Transform (DFT) of a signal encoded in the phase of the quantum states in an equal superposition: it takes the signal included in the quantum state phases and gives as output the frequency of the signal encoded in the amplitude. QFT can also be seen as a powerful way to transform information encoded in the phases into amplitude that can be directly measurable. Besides the QFT direct usage for spectral analysis, QFT and its inverse iQFT are omnipresent primitives in quantum computing. They are valuable computing primitives that are critical building blocks for Shor's algorithms, Quantum Phase Estimation, and Harrow–Hassidim–Lloyd (HHL) [56] algorithms, to cite a few prime examples.

As for the Fast Fourier Transform (FFT), the QFT is based on recursive or iterative execution of smaller DFTs, consisting of controlled phase (C_ϕ) and H gates [57], and a QFT of one qubit being the H [58], with a final qubit reversal, consisting of SWAP gates. When investigating the circuit complexity, the number of gates grows only as $\mathcal{O}(N^2)$, where N is the number of qubits. Conversely, the classical FFT number of operations grows with the number of input bits n as $\mathcal{O}(n2^n)$. However, the $\mathcal{O}$ notation obfuscates the comparison of FFT and QFT in terms of performance. As noted by several works [12], [59], there exists a performance crossover point in terms of the number of qubits, after which the QFT becomes more computationally favorable than FFT. In fact, QFT becomes advantageous for more than 22 qubits or about four million input sample sizes. FFT is more convenient than QFT for less than four million input samples. This is one of the reasons why quantum computing is often said to be favorable for solving big-data problems.

When designing a quantum dataflow diagram for the QFT, for clarity of exposition, we choose a four-qubit system and associate quantum circuit, as shown in Fig. 7. We encode a signal of frequency two in the phase of the quantum

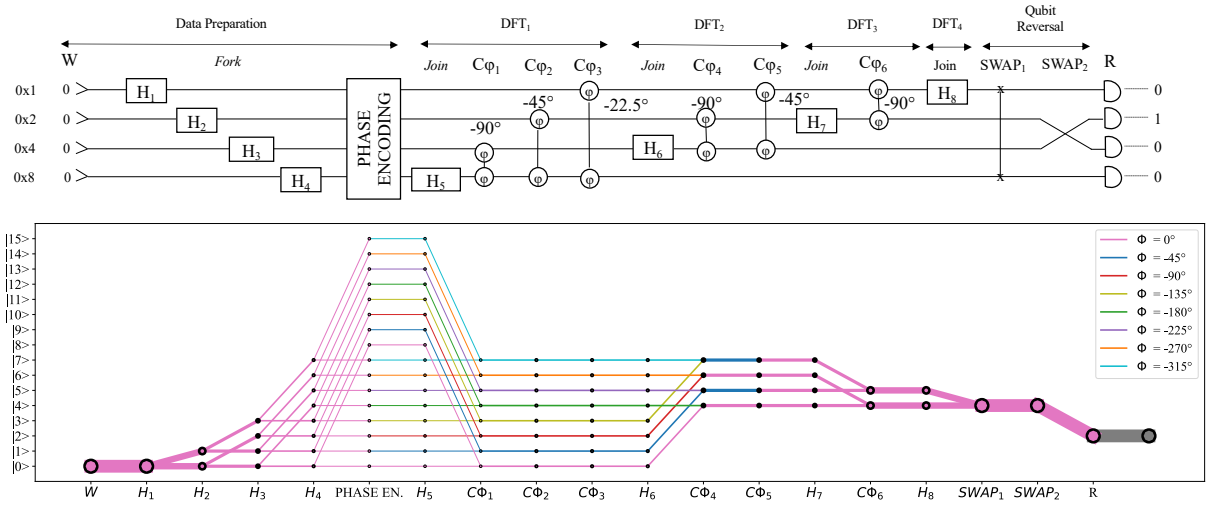

Fig. 7. Four-qubit QFT circuit and associated quantum dataflow diagram for the signal of a frequency of two encoded in quantum state phases.

states: the phase rotates twice across quantum states, and the result is deterministic and equal to $|2\rangle$, when measured. More precisely, this signal encodes 16 values with relative phases corresponding to two full anti-clockwise rotations (0°, -45°, -90°, ...) into the quantum states $|0\rangle$ ... $|15\rangle$. We assume that the encoding operation is only one operation while the encoding phase (such as a simple signal can be encoded by using a quantum circuit). After putting the system in a superposition of states and encoding the input vector with a quantum circuit, four quantum DFTs are applied in succession, with a final qubit reversal of the quantum results using `SWAP` operations.

When analyzing the QFT dataflow diagram, we have a serial WRITE operation during the data preparation, which is the interface between classical and quantum worlds. We then have four Hadamard gates (H_1 – H_4) to put in a superposition using a fork operation. H_1 – H_4 spawn the maximum available parallelism, which is 16 quantum threads. The QFT algorithm consists of four quantum DFT blocks with `H` gates (the `H` gate itself can be considered as a DFT of size one), providing destructive interference and controlled phase operations that strategically alter the phases to allow for more interference. In particular, the Hadamard gates H_5 – H_8 halve the quantum parallelism, similarly to the *decimation* transformation in FFTs. After H_8, the process is inherently serial (or classical) with the qubit reversal operation via `SWAP` gates, required to provide the correct results. We want to point out, though, that the final `SWAP` gates are critical to providing entanglement, necessary entangled states that could not be provided by other means. For instance, consider a signal that consists of a superposition of constant (equivalent to frequency 0) value and a frequency equal to 15: the QFT output is an entangled state, only achievable via the final `SWAP` gates.

It is important to acknowledge that varying the input leads to changes in the amount of parallelism. For instance, an input signal featuring multiple frequencies results in higher parallelism and diminished destructive interference. The quantum parallelism is intimately tied to the input and encoding methodologies employed.

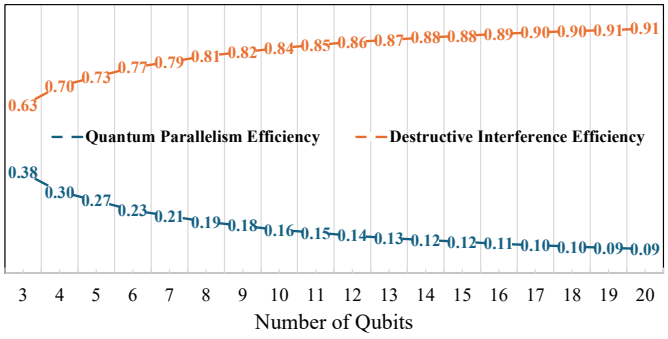

Fig. 8. Quantum parallel and destructive interference efficiencies varying the number of qubits in a QFT acting on a signal with a frequency of two.

An intriguing observation arises from the reversibility of quantum transformations: the quantum dataflow diagram for the inverse QFT (iQFT) can be interpreted in the reverse direction. For instance, one can trace the path from a frequency of two to create a signal encoded in the phase of quantum states in equal superposition.

The QFT dataflow diagram provides insights into the quantum parallelism inherent in the algorithm. Simplifying the analysis to focus on the parallelism after data preparation, including the serial W operation and the initial encoding of the input signal, we can calculate the total number of nodes ($T_W = 68$) and the critical path length ($T_\infty = 14$). This results in a total parallelism of $P = T_W/T_\infty = 4.86$. With corresponding values of $\eta_p = 0.3$ and $\eta_{DI} = 0.7$, we observe that the quantum parallelism accounts for only 30% of the available parallelism of 16, indicating effective destructive interference. By fixing the QFT input signal to a frequency of two, we can derive an expression to calculate the quantum parallelism for varying numbers of qubits as follows:

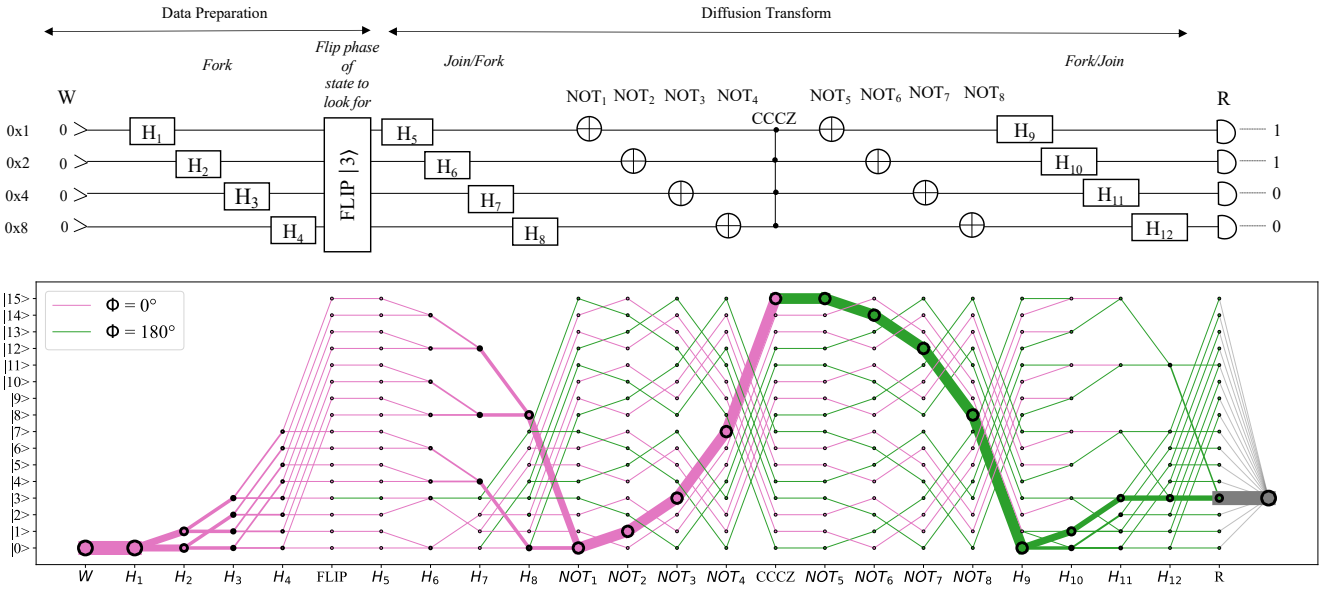

Fig. 9. Quantum dataflow diagram for one AA iteration in a four-qubit system. In this example, the flip operation marks the relative phase of the quantum state $|3\rangle$.

$P = (N2^N + \lfloor N/2 \rfloor + 2)/(\frac{1}{2}N(N+1) + \lfloor N/2 \rfloor + 2)$. As the number of qubits increases, the quantum parallelism efficiency decreases while the efficiency of the destructive interference increases. This trend is depicted in Fig. 8.

B. Amplitude Amplification Iteration

The AA iteration (also called *diffusion transform* or *inversion above the average*) is a critical quantum computing primitive, first designed and expressed in the formulation we know today by Lev Grover in 1996 [17], [60], [61]. Lev Grover applied AA iterations to the problem of finding an item in an unordered set, the so-called *searching for a needle in a haystack* problem [60] and pointed out that the basic AA iteration can also be applied to several other applications, including neural networks and associative memories. With N_I items, Grover's algorithm relies on $\sqrt{N_I}$ AA iterations (including the query/ call to the oracle) to achieve the maximum probability of measuring the correct item. In the classical case, identifying an item would take $N_I - 1$ queries in the worst-case scenario or $N_I/2$ queries on the average. For this reason, Grover's algorithm provides only a quadratic speed-up, when compared to the classical case.

At its fundamentals, AA transforms the difference between the relative phases into the amplitude magnitude, making the quantum state that, for instance, we marked with a flipped relative phase, identifiable through a measurement. For the sake of simplicity, we flip the phase of a given quantum state by 90°. However, this can be implemented with an oracle based on phase logic.

Fig. 9 illustrates the quantum circuit for the AA iteration on a four-qubit system and marks the relative phase of the quantum state $|3\rangle$ by flipping its phase. The circuit consists of data preparation, including spawning all available parallelism and encoding the input as a flipped relative phase in the quantum state, and the diffusion transform quantum circuits. The outcome of the AA quantum primitive transforms the relative phase difference into an amplitude difference, enhancing the probability of measuring state $|3\rangle$.

When investigating the AA quantum dataflow diagram, we observe that, as for the QFT, the write operation is serial. After the data preparation step (corresponding to all the available parallelism and marking an item by flipping its corresponding quantum state phase), and the H_5 – H_8 transformations, AA exhibits a quantum thread with a larger amplitude (corresponding to thicker lines) than others. We might be tempted to prune the quantum dataflow diagram to one quantum thread, de-quantize it, and make it serial to all the quantum states resulting from the sequence NOT_1 – NOT_8. However, this would not be the correct method to use, and it would break the correctness of the algorithms. The AA relies critically on all the 16 quantum threads – albeit they have a low probability of occurring – to achieve the correct answer. To convince ourselves of that, hypothetically, we can merge all the quantum threads into the dominant one between NOT_1 and NOT_8. However, H_9 – H_{12} acting only one state quantum will create an equal superposition of states with no discernible result when measuring. In other words, the unbalanced superposition across all the 16 quantum states is critical for the correct code functioning, and therefore, we cannot prune the different threads. We note that the AA iteration and Grover's algorithm use all the available parallelism by design, resulting in a non-deterministic outcome.

As it was done for the four-qubit QFT, we proceed to evaluate the quantum parallelism inherent in the AA iteration. Disregarding the data preparation phase, in the context of a four-qubit system, we observe a total of 242 nodes (T_W)

and a span of 19 (T_∞), resulting in a quantum parallelism of $P = T_W/T\infty = 242/19 = 12.7$. A quantum parallel efficiency of $\eta_P = 12.7/16 = 0.8$ indicates that the AA iteration uses 80% of the available parallelism, showcasing a higher utilization of available quantum parallelism when compared to the QFT. Moreover, the influence of destructive interference with a $\eta_{DI} = 0.2$ appears to be relatively limited in the AA iteration scenario. As for the QFT, the efficiency of quantum parallelism diminishes with an increasing number of qubits, while the efficiency of destructive interference shows a proportional increase.

VI. Quantum Parallelism Laws

An intriguing consideration in the context of quantum computing is the applicability and formulation of classical parallelism laws, such as Amdahl's and Gustafson's laws, in quantum computing. Amdahl's law states that the overall performance improvement gained by optimizing a single part of a system is limited by the fraction of time of the serial [7]. In other words, the maximum speed-up achievable (i.e., overall performance improvement) depends on the percentage of the code that can be parallelized, regardless of the amount of available parallelism. In quantum algorithms, parallelism is an inherent characteristic associated with the quantum circuit and depends on both the algorithm and the input data.

A. Reconsidering Amdahl's Law for Quantum Computing

The Amdahl's law describes the potential speed-up of a classical parallel computing system as a function of the proportion of the program that can be parallelized, and it can be expressed as follows:

$$S = \frac{1}{F + (1 - F)/P}, \tag{1}$$

where S is the parallel speed-up, F is the fraction of the threads/processes that must be run serially (that cannot be parallelized), and P is the number of threads/processes. In the limit of infinite parallelism ($P \to \infty$), the speed-up of a program with a finite serial part is asymptotically equal to $1/F$. For instance, if the serial portion of a program constitutes 50% of the whole program, even with an infinite number of processes, the speed-up is only two.

While Amdahl's law provides valuable insights into the limitations of parallelism in classical systems, its direct application to quantum computing faces challenges due to the unique characteristics of quantum algorithms. In quantum computing, parallelism is an inherent characteristic associated with the quantum circuit, and it depends on both the algorithm and the input data. However, the speed-up achieved does not directly depend on quantum parallelism and may even decrease with it. This is because increased quantum parallelism often leads to less efficient interference patterns, limiting the effectiveness of parallelism. Additionally, Amdahl's law is formulated with strong scaling in mind, where the problem size remains constant, and parallelism is increased. However, in quantum computing, increasing parallelism automatically increases the workload as more qubits are used. Therefore, the concept of strong scaling does not directly apply to quantum computing.

While we argued that the speed-up does not depend on quantum parallelism, the fundamental limitation related to the serial portion of the algorithm still holds, e.g., a serial portion in a parallel algorithm can nullify computing mechanisms to achieve speed-up. All the quantum algorithms have an inherently serial part, which is the classical quantum I/O, and in general, quantum data preparation and retrieval are needed to initialize the quantum simulation [12]. For instance, when solving a linear system with the HHL algorithm [56], the matrix must be loaded from classical data and encoded into the quantum state. The classical-quantum I/O wall fundamentally limits the quantum advantage, regardless of the speed-up achieved with quantum computing. For this reason, as pointed out by several previous works, the serial fraction due to the I/O wall has to be tackled as one of the priorities. Another more fundamental factor limiting the quantum advantage is the lack of algorithm destructive interference, which is key to eliminating the wrong, incorrect solutions.

B. Reconsidering Gustafson's Law for Quantum Computing

Conversely to Amdahl's law, Gustafson's law [8] focuses on scaling the size of the problem as the number of processors increases. It assumes that the workload and problem size can be scaled up to use additional processors in a setup called *weak scaling*. Gustafson's law expresses the speed-up as $S = P - F(P - 1)$. In the classical context, Gustafson's law effectively removes the limitation of the serial part, making the speed-up close to linear with the number of processors if the non-parallelizable fraction is relatively small. In quantum computing, the inherent serial part is the classical quantum I/O, which is not a constant factor, but instead, it is a function of the number of quantum states used for encoding the input. Performance models for describing the classical-quantum I/O wall are needed further to develop Gustafson's law to the quantum regime.

Another interesting point related to Gustafson's law and weak scaling is the quite narrow weak scaling window in terms of qubits, where quantum computing is effectively advantageous for classical computing. For relatively small problem sizes, classical computing is faster. We saw that we need at least an input size of 2^{22} for the QFT to make it advantageous. In addition, given the exponential growth of quantum states with the number of qubits, quantum computing indeed provides an unparalleled number of superpositions of quantum states, and the available quantum parallelism (albeit not the quantum parallelism efficiency) increases exponentially with the number of qubits. This presents a fundamental departure from the classical case, where achieving significant advances in parallelism requires substantial technological leaps, and still, scientific workloads can be tailored to leverage the available parallelism efficiently. In quantum computing, neglecting the problem of the factorization of very large numbers, we find ourselves in a new regime where the amount

of parallelism surpasses the needs of even the largest problem sizes. In such a paradigm, we witness the end of Gustafson's law, as the problem size can no longer effectively utilize the potential of potentially unlimited parallelism offered by quantum computing because of *limited classical* problem sizes.

VII. Conclusion

This work investigated the fundamental questions surrounding quantum parallelism in quantum applications, exploring its implications for actual speed-up and challenging classical parallelism paradigms in the context of quantum computing. Quantum parallelism, emerging from the interaction of 2^N quantum states, exhibits a characteristic interference pattern reminiscent of waves or antennas. By leveraging classical parallel computing concepts within quantum computing, we underscored the importance of fork-join operations, which enable both constructive and destructive interference. To quantify parallelism, we introduced quantum dataflow diagrams, providing a visual tool for measuring quantum parallelism in practical applications such as the QFT and AA iterations. Additionally, we introduced metrics such as quantum parallelism and destructive interference efficiencies to evaluate the utilization of available parallelism in quantum algorithms. Our study revealed that in the QFT and AA iterations, quantum parallel efficiency tends to diminish with an increased number of qubits for the analyzed applications. Finally, drawing parallels with classical computing principles, we revisited Amdahl's and Gustafson's laws. This led us to reconsider the role of the serial portion of quantum applications, the classical-quantum I/O bottleneck, and highlighted the significance of a narrow window of problem sizes that render quantum computing advantageous.

An inherent challenge in defining quantum parallelism lies in its entanglement – pun intended – with the interpretation of quantum mechanics, which leads to philosophical complexities regarding the actual realization of quantum parallelism and computation [62]–[64]. Different interpretations of the physical reality of quantum mechanics can give rise to distinct *conceptual* implementations of parallelism. To navigate this challenge, we adopted a practical yet shallow approach, circumventing philosophical obstacles as much as possible. However, it is worth noting that one of the most intuitive and elegant interpretations is Hugh Everett's many-worlds interpretation of quantum mechanics and the multiverse hypothesis [65], [66]. According to this interpretation, time is envisioned as a many-branched tree, where every possible outcome of quantum parallelism is actualized in a separate branch or universe. This interpretation suggests that each computational path exists simultaneously across different branches of reality. This concept aligns with the notion of quantum parallelism, implying that all potential outcomes of quantum computation occur in parallel across multiple universes. One important implication of the multiverse theory is its ability to support quantum parallelism surpassing the number of particles in the observable universe ($>$300 qubits). In such scenarios, multiple parallel universes can concurrently accommodate computational processes unfolding across different branches without being constrained by the particle count within a single universe [66].

This work offers an intuitive perspective on quantum parallelism, clarifying its fundamental aspects. Nevertheless, it is crucial to acknowledge that our exploration merely scratches the surface of quantum parallelism. Our investigation primarily revolves around quantum parallelism within the computational or standard basis, aligning with the canonical approach directly corresponding to classical bit concepts. However, using alternative bases would necessitate a reconfiguration of the quantum dataflow diagrams. Moreover, we focused on quantum gate abstractions, state vectors, and pure state formulations. Alternative and more comprehensive quantum computing abstractions and approaches, such as Hamiltonian parameterization [67] or the exploration of continuous-variable quantum computing paradigms [28], [68], require further investigations. Such approaches could offer even deeper insights into quantum parallelism.

From the practical point of view, to quantify the extent of quantum parallelism, we proposed the quantum dataflow diagram, akin to classical dataflow graphs [69]. This tool enables the visualization and measurement of quantum parallelism, revealing critical distinctions between classical and quantum parallelism and facilitating the development of quantum performance models. For simplicity, we have assumed a uniform work associated with each gate transformation, where each node has an execution time of one. However, quantum dataflow diagrams can be used as starting models to develop quantum application performance models. A more accurate model of the quantum application can be achieved by weighting each node differently based on its computational cost. Moreover, we can incorporate additional overheads, such as latencies from quantum hardware, into the dataflow diagram to construct more sophisticated quantum application performance models.

This paper aimed to pave the way for expressing quantum concepts within the framework of classical parallelism. While this approach holds promise for practical applications in quantum programming models—especially those built upon classical programming paradigms and tools—it remains uncertain whether it can be directly applied to specific hardware concepts. This uncertainty arises because the paper primarily deals with high-level abstractions, such as quantum gates and circuits, which are agnostic to specific hardware implementations. Nevertheless, this work offers a perspective that can stimulate further research and engage the HPC community on the elusive concept of quantum parallelism. By bridging classical parallelism frameworks with quantum computing principles, the paper opens doors for exploring methodologies, developing quantum programming models, and advancing our understanding of quantum algorithms.

References

[1] D. Deutsch, A. Ekert, and R. Lupacchini, "Machines, logic and quantum physics," *Bulletin of Symbolic Logic*, vol. 6, no. 3, pp. 265–283, 2000.

[2] S. Lloyd, "Ultimate physical limits to computation," *Nature*, vol. 406, no. 6799, pp. 1047–1054, 2000.

[3] D. Deutsch and R. Jozsa, "Rapid solution of problems by quantum computation," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 439, no. 1907, pp. 553–558, 1992.

[4] D. C. Marinescu, "The promise of quantum computing and quantum information theory-quantum parallelism," in *19th IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2005, pp. 3–pp.

[5] I. L. Chuang and Y. Yamamoto, "Simple quantum computer," *Physical Review A*, vol. 52, no. 5, p. 3489, 1995.

[6] E. G. Rieffel and W. H. Polak, *Quantum computing: A gentle introduction*. MIT Press, 2011.

[7] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, spring joint computer conference*, 1967, pp. 483–485.

[8] J. L. Gustafson, "Reevaluating Amdahl's law," *Communications of the ACM*, vol. 31, no. 5, pp. 532–533, 1988.

[9] Y. Shi, "Reevaluating Amdahl's law and Gustafson's law," *Computer Sciences Department, Temple University (MS: 38-24)*, 1996.

[10] R. Babbush, J. R. McClean, M. Newman, C. Gidney, S. Boixo, and H. Neven, "Focus beyond quadratic speedups for error-corrected quantum advantage," *PRX quantum*, vol. 2, no. 1, p. 010103, 2021.

[11] M. E. Beverland, P. Murali, M. Troyer, K. M. Svore, T. Hoefler, V. Kliuchnikov, G. H. Low, M. Soeken, A. Sundaram, and A. Vaschillo, "Assessing requirements to scale to practical quantum advantage," *arXiv preprint arXiv:2211.07629*, 2022.

[12] T. Hoefler, T. Häner, and M. Troyer, "Disentangling hype from practicality: On realistically achieving quantum advantage," *Communications of the ACM*, vol. 66, no. 5, pp. 82–87, 2023.

[13] A. K. Fedorov, N. Gisin, S. M. Beloussov, and A. I. Lvovsky, "Quantum computing at the quantum advantage threshold: a down-to-business review," *arXiv preprint arXiv:2203.17181*, 2022.

[14] M. Schuld and N. Killoran, "Is quantum advantage the right goal for quantum machine learning?" *Prx Quantum*, vol. 3, no. 3, p. 030101, 2022.

[15] C. E. Leiserson and I. B. Mirman, "How to survive the multicore software revolution (or at least survive the hype)," *Cilk Arts*, vol. 1, p. 11, 2008.

[16] Y. Vandriessche, E. D'Hondt, T. Van Cutsem, and T. D'Hondt, "A highly-parallel formulation of quantum computing simulation through fine-grained dataflow." in *Workshop on Data-Flow Execution Models for Extreme Scale Computing*. Citeseer, 2013.

[17] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[18] D. Camps, R. Van Beeumen, and C. Yang, "Quantum Fourier transform revisited," *Numerical Linear Algebra with Applications*, vol. 28, no. 1, p. e2331, 2021.

[19] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical hamiltonian model of computers as represented by turing machines," *Journal of statistical physics*, vol. 22, pp. 563–591, 1980.

[20] ——, "Quantum mechanical hamiltonian models of turing machines," *Journal of Statistical Physics*, vol. 29, pp. 515–546, 1982.

[21] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, no. 6/7, 1982.

[22] ——, "Quantum mechanical computers," *Optics news*, vol. 11, no. 2, pp. 11–20, 1985.

[23] D. Deutsch, "Quantum theory, the Church–Turing principle and the universal quantum computer," *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, vol. 400, no. 1818, pp. 97–117, 1985.

[24] R. Jozsa, "Characterizing classes of functions computable by quantum parallelism," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 435, no. 1895, pp. 563–574, 1991.

[25] A. Ekert and R. Jozsa, "Quantum computation and Shor's factoring algorithm," *Reviews of Modern Physics*, vol. 68, no. 3, p. 733, 1996.

[26] ——, "Quantum algorithms: entanglement–enhanced information processing," *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 356, no. 1743, pp. 1769–1782, 1998.

[27] D. Deutsch and A. Ekert, "Beyond the quantum horizon," *Scientific American*, vol. 307, no. 3, pp. 84–89, 2012.

[28] S. Lloyd and S. L. Braunstein, "Quantum computation over continuous variables," *Physical Review Letters*, vol. 82, no. 8, p. 1784, 1999.

[29] R. Cleve, A. Ekert, C. Macchiavello, and M. Mosca, "Quantum algorithms revisited," *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, vol. 454, no. 1969, pp. 339–354, 1998.

[30] V. Kendon and B. C. Sanders, "Complementarity and quantum walks," *Physical Review A*, vol. 71, no. 2, p. 022307, 2005.

[31] N. S. Yanofsky and M. A. Mannucci, *Quantum computing for computer scientists*. Cambridge University Press, 2008.

[32] D. Deutsch, A. Ekert, R. Jozsa, J. Cirac, P. Zoller, and J. Poyatos, "Concepts of quantum computation," in *The Physics of Quantum Information: Quantum Cryptography, Quantum Teleportation, Quantum Computation*. Springer, 2000, pp. 93–132.

[33] R. Jozsa, "Quantum effects in algorithms," in *NASA International Conference on Quantum Computing and Quantum Communications*. Springer, 1998, pp. 103–112.

[34] ——, "On the simulation of quantum circuits," *arXiv preprint quant-ph/0603163*, 2006.

[35] H. Jnane, B. Undseth, Z. Cai, S. C. Benjamin, and B. Koczor, "Multicore quantum computing," *Physical Review Applied*, vol. 18, no. 4, p. 044064, 2022.

[36] H. Zhang, K. Yin, A. Wu, H. Shapourian, A. Shabani, and Y. Ding, "Compilation for quantum computing on chiplets," *arXiv preprint arXiv:2305.05149*, 2023.

[37] S. Rodrigo, S. Abadal, C. G. Almudéver, and E. Alarcón, "Modelling short-range quantum teleportation for scalable multi-core quantum computing architectures," in *Proceedings of the Eight Annual ACM International Conference on Nanoscale Computing and Communication*, 2021, pp. 1–7.

[38] T. Häner, D. S. Steiger, T. Hoefler, and M. Troyer, "Distributed quantum computing with QMPI," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2021, pp. 1–13.

[39] T. Nguyen, Y. Shi, S. Stein, T. Stavenger, M. Warner, M. Roetteler, T. Hoefler, and A. Li, "A reference implementation for a quantum Message Passing Interface," in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 2. IEEE, 2023, pp. 292–293.

[40] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[41] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien, "A variational eigenvalue solver on a photonic quantum processor," *Nature communications*, vol. 5, no. 1, p. 4213, 2014.

[42] J. Tilly, H. Chen, S. Cao, D. Picozzi, K. Setia, Y. Li, E. Grant, L. Wossnig, I. Rungger, G. H. Booth *et al.*, "The variational quantum eigensolver: a review of methods and best practices," *Physics Reports*, vol. 986, pp. 1–128, 2022.

[43] M. Schuld, I. Sinayskiy, and F. Petruccione, "An introduction to quantum machine learning," *Contemporary Physics*, vol. 56, no. 2, pp. 172–185, 2015.

[44] J. Hidary, *Quantum computing: an applied approach*. Springer, 2019, vol. 1.

[45] M. Schuld, R. Sweke, and J. J. Meyer, "Effect of data encoding on the expressive power of variational quantum-machine-learning models," *Physical Review A*, vol. 103, no. 3, p. 032430, 2021.

[46] M. Schuld, "Supervised quantum machine learning models are kernel methods," *arXiv preprint arXiv:2101.11020*, 2021.

[47] M. Schuld, F. Petruccione, M. Schuld, and F. Petruccione, "Quantum models as kernel methods," *Machine Learning with Quantum Computers*, pp. 217–245, 2021.

[48] G. Hager and G. Wellein, *Introduction to high performance computing for scientists and engineers*. CRC Press, 2010.

[49] T. Sterling, M. Brodowicz, and M. Anderson, *High performance computing: modern systems and practices*. Morgan Kaufmann, 2017.

[50] T. Häner, V. Kliuchnikov, M. Roetteler, M. Soeken, and A. Vaschillo, "Qparallel: Explicit parallelism for programming quantum computers," *arXiv preprint arXiv:2210.03680*, 2022.

[51] R. D. Blumofe, C. F. Joerg, B. C. Kuszmaul, C. E. Leiserson, K. H. Randall, and Y. Zhou, "Cilk: An efficient multithreaded runtime system," *ACM SigPlan Notices*, vol. 30, no. 8, pp. 207–216, 1995.

[52] Qiskit contributors, "Qiskit: An open-source framework for quantum computing," 2023.

[53] C. Developers, "Cirq," Jul. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.8161252

[54] S. V. Isakov, D. Kafri, O. Martin, C. V. Heidweiller, W. Mruczkiewicz, M. P. Harrigan, N. C. Rubin, R. Thomson, M. Broughton, K. Kissell *et al.*, "Simulations of quantum circuits with approximate noise using qsim and cirq," *arXiv preprint arXiv:2111.02396*, 2021.
[55] D. Coppersmith, "An approximate Fourier transform useful in quantum factoring," *arXiv preprint quant-ph/0201067*, 1994.
[56] A. W. Harrow, A. Hassidim, and S. Lloyd, "Quantum algorithm for linear systems of equations," *Physical review letters*, vol. 103, no. 15, p. 150502, 2009.
[57] R. R. Tucci, "Quantum fast Fourier transform viewed as a special case of recursive application of cosine-sine decomposition," *arXiv preprint quant-ph/0411097*, 2004.
[58] R. Cleve and J. Watrous, "Fast parallel circuits for the quantum Fourier transform," in *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE, 2000, pp. 526–536.
[59] E. R. Johnston, N. Harrigan, and M. Gimeno-Segovia, *Programming quantum computers: essential algorithms and code samples*. O'Reilly Media, 2019.
[60] L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack," *Physical review letters*, vol. 79, no. 2, p. 325, 1997.
[61] ——, "Quantum computers can search rapidly by using almost any transformation," *Physical Review Letters*, vol. 80, no. 19, p. 4329, 1998.
[62] A. Duwell, "The many-worlds interpretation and quantum computation," *Philosophy of Science*, vol. 74, no. 5, pp. 1007–1018, 2007.
[63] A. Duwell, M. Cuffaro, and S. Fletcher, "How to make orthogonal positions parallel: Revisiting the quantum parallelism thesis," *Physical perspectives on computation, computational perspectives on physics*, pp. 83–102, 2018.
[64] M. Lanzagorta and J. Uhlmann, "Is quantum parallelism real?" in *quantum information and computation VI*, vol. 6976. SPIE, 2008, pp. 172–178.
[65] D. Wallace, *The emergent multiverse: Quantum theory according to the Everett interpretation*. Oxford University Press, USA, 2012.
[66] D. Deutsch, *The fabric of reality*. Penguin UK, 1998.
[67] Y. Peng, J. Young, P. Liu, and X. Wu, "SimuQ: A framework for programming quantum hamiltonian simulation with analog compilation," *Proceedings of the ACM on Programming Languages*, vol. 8, no. POPL, pp. 2425–2455, 2024.
[68] N. Killoran, T. R. Bromley, J. M. Arrazola, M. Schuld, N. Quesada, and S. Lloyd, "Continuous-variable quantum neural networks," *Physical Review Research*, vol. 1, no. 3, p. 033063, 2019.
[69] K. Wongsuphasawat, D. Smilkov, J. Wexler, J. Wilson, D. Mane, D. Fritz, D. Krishnan, F. B. Viégas, and M. Wattenberg, "Visualizing dataflow graphs of deep learning models in tensorflow," *IEEE transactions on visualization and computer graphics*, vol. 24, no. 1, pp. 1–12, 2017.