electronics

MDPI

Article

QiBAM: Approximate Sub-String Index Search on Quantum Accelerators Applied to DNA Read Alignment

Aritra Sarkar [1,2,*], Zaid Al-Ars [1], Carmen G. Almudever [1] and Koen L. M. Bertels [2,3]

[1] Department of Quantum & Computer Engineering, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628 CD Delft, The Netherlands; Z.Al-Ars@tudelft.nl (Z.A.-A.); a.garciaalmudever-1@tudelft.nl (C.G.A.)
[2] QBeeX, B-3001 Leuven, Belgium; Koen.Bertels@qbee.eu
[3] Department of Electrical Engineering, Katholieke Universiteit Leuven, B-3001 Leuven, Belgium
* Correspondence: A.Sarkar-3@tudelft.nl

Abstract: With small-scale quantum processors transitioning from experimental physics labs to industrial products, these processors in a few years are expected to scale up and be more robust for efficiently computing important algorithms in various fields. In this paper, we propose a quantum algorithm to address the challenging field of data processing for genome sequence reconstruction. This research describes an architecture-aware implementation of a quantum algorithm for sub-sequence alignment. A new algorithm named QiBAM (quantum indexed bidirectional associative memory) is proposed, which uses approximate pattern-matching based on Hamming distances. QiBAM extends the Grover's search algorithm in two ways, allowing: (1) approximate matches needed for read errors in genomics, and (2) a distributed search for multiple solutions over the quantum encoding of DNA sequences. This approach gives a quadratic speedup over the classical algorithm. A full implementation of the algorithm is provided and verified using the OpenQL compiler and QX Simulator framework. Our implementation represents a first exploration towards a full-stack quantum accelerated genome sequencing pipeline design.

Keywords: accelerator architectures; associative memory; DNA read alignment; genomics; pattern matching; quantum algorithms; quantum computing; quantum search

Citation: Sarkar, A.; Al-Ars, Z.; Almudever, C.G.; Bertels, K.L.M. QiBAM: Approximate Sub-String Index Search on Quantum Accelerators Applied to DNA Read Alignment. *Electronics* **2021**, *10*, 2433. https://doi.org/10.3390/electronics10192433

Academic Editor: Lucas Lamata

Received: 9 September 2021
Accepted: 30 September 2021
Published: 7 October 2021

1. Introduction

The idea of using the fundamental physical building blocks of nature for computation [1] laid the foundation for the second quantum revolution, focusing on controlling quantum systems and engineering them for arbitrary computation, instead of a passive understanding of quantum mechanical phenomena. The current developments in this field are inspired by two directions—theoretical computer science and computer engineering.

In recent years, there has been a push by the computing industry towards heterogeneous multi-processor systems, where the general-purpose CPU offloads tasks to specialized accelerators. For example, these accelerators include graphics processing units (GPU), field-programmable gate arrays (FPGA) and digital signal processing units (DSP). Likewise, we can define a quantum accelerator as a classical processor that uses the power of a quantum processor for specific tasks [2]. Our research follows the circuit model for gate-based quantum computing. In this computing model, the classical processor interacts with the quantum processor via commands that specify a sequence of unitary gate operators and receives the collapsed state of the two-level qubit system on measurement. Quantum accelerators allow us to solve a wider complexity class of problems efficiently (the Bounded Quantum Polynomial-time class).

While quantum algorithms are often compared with their classical counterparts in terms of asymptotic complexity, these developments remain fairly independent from the

engineering efforts towards manufacturing quantum processing units. Current state-of-the-art quantum processors are limited by the number of qubits, coherence time and connectivity between these qubits. There is no clear technological winner up to this point among the competing hardware technologies (like superconductors, semiconductors, nitrogen-vacancy centers, ion traps, etc.) in terms of scalability. The development of quantum algorithms for various use-cases is nevertheless a very active field of research. These proof-of-concept implementations can be tested [3] for small (up to around 50 qubits [4]) problem instances on quantum simulators. For this research, the QX Simulator platform [5] with the OpenQL [6] language is used.

The most promising candidate applications for quantum acceleration are physical system simulation, cryptography, optimization and machine learning. Our work [7] in this article falls under the umbrella of quantum search-based algorithms, where the high dimensional state space of the qubits are harnessed to explore/search an optimization landscape faster and better. While these generic algorithms are ubiquitous in computer science and data-structures, in this research an exemplary case of DNA sequencing application is considered in depth. This is motivated by the immense application of this area in the upcoming years and its reliance on a high volume and speed of data processing. Faster DNA sequence reconstruction will assist the adoption of precision medication by accelerating the diagnostics pipeline.

Bioinformatics algorithms in use today (especially our focus here, that is, DNA sequence alignment) rely on heuristic methods to alleviate the huge volume of data. Even these heuristic approaches take days to run on supercomputing clusters, limiting their applicability to more wider use. The advantage of quantum algorithms discussed in this work is twofold: (a) they have a lower cycle time than corresponding classical algorithms; and (b) the global optima is guaranteed to be sampled with the highest probability instead of sub-optima for current heuristic algorithms. Application areas like cryptography and physical simulations are highly susceptible to noisy input/computation which makes them highly improbable to achieve good results in the low coherence quantum systems that will be available in the near-term. The application for DNA alignment tries to search for sub-optima within an acceptable threshold, so an approximate solution is more permissive for this use-case.

Quantum approaches to DNA sequencing have not been explored in much depth before. This is the first time [7] a gate-based quantum algorithm has been presented where the DNA index of the best matching sub-string is retrieved with high probability. While previous work on associative memory and phone directories provides the tools for this application, in this research a holistic gate-level description of the entire algorithm and the oracle is provided. The specific design in the context of genome sequence reconstruction takes into account the following requirements:

- In genomic sequences, since reads can contain errors, an approximate matching query is required;
- A constant Oracle is useful as compiling the Oracle differently for every read at run-time is tedious;
- The associated index in the reference needs to be retrieved, instead of the corrected query.

The proposed approach, which takes into account these requirements, is tested and verified on a quantum simulator with small artificial sequences as a proof of concept.

The rest of the paper is organized as follows: In Sections 2 and 3 the research problem in genomics and the quantum search algorithm are introduced to bridge the interdisciplinary gap for the readers. Section 4 discusses the three existing quantum algorithm designs of associative search, which are used to derive a new search approach in Section 5 that incorporates an approximate distributed search. Section 6 discusses the results in the context of the application. Section 7 concludes the paper.

2. DNA Sequence Reconstruction

In this section, we present the problem addressed in this research work, starting as an abstract algorithm, gradually developing towards the intended application in genomics. A database is defined with indices and an associated data element for each index, as illustrated with a simple example in Figure 1. A search query on this database is provided. It is possible that the exact match for the element is not present in the database. The objective of an approximate index search is to return the index of the element that is closest to the search query. In this example, by visual inspection it can easily be inferred that the closest match is the element at index 1. In fact, the key cut pattern on both the search query and the nearest matching keys are the same, a metaphor to the usefulness of the nearest association being of functional use.

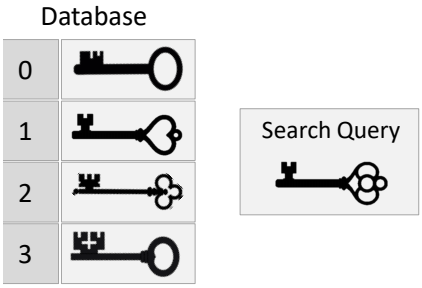

Figure 1. Simple example of associative search.

The model of associative memory is closely related to how learning occurs in the brain, thus finding its use in computational neuroscience. It is useful when we have noisy or incomplete knowledge of the data. An indexed memory variant is useful when we not only want to recover the nearest matching data from the database, but also the index of its occurrence that has associated semantic meaning. For example, an object detection algorithm tries to find the nearest match to objects (e.g., humans, cars, traffic lights) in a scene. After the nearest match is detected, the index of the match location can help in positioning the object in the scene, for example, if a human is on the left or right of the road. The scene here is the database and the output is the pixel coordinate of the detected object, based on which important decisions might be automated in a self-driving car.

In the rest of this paper, we would focus on one such application of approximate index search, where the data are one dimensional (thus, a sub-string search). These data are DNA sequences where finding the index of the nearest match to a query is of immense computational value in bioinformatics.

Reconstruction Using Read Alignment

DNA is a thread-like long polymer made up of nucleic molecules carrying the genetic instructions used in the growth, development, functioning and reproduction of organisms. These genetic instructions are primarily encoded in the sequence of the DNA using the four nucleic molecules, adenine (A), cytosine (C), guanine (G) and thymine (T). Adenine pairs with thymine and guanine pairs with cytosine, represented by A-T and G-C, which are referred to as base pairs (bp) in the DNA. The information in the two strands of the DNA are thus complimentary. This allows simplifying the representation as a single string with four symbols while processing it as digital data.

The length of genomes varies greatly among organisms, for example, the human genome is approximately 3.289×10^9 bp long. However, owing to this length, it is not possible to obtain the entire DNA sequence in a single readout from the sequencing machines. Instead, multiple copies of the DNA are fragmented and these short strings are stitched back together—a process called DNA sequence reconstruction. DNA sequence reconstruction is primarily of two types: (i) de novo assembly; and (ii) ab initio alignment. De novo assembly is used while sequencing a new organism, where the short reads are stitched back together based on the overlap between each pair. This is computationally

very intensive and remains intractable for classical high-performance computing except for small micro-organisms. For organisms with longer DNA, for example, humans, we prefer to use alignment. Once the DNA is constructed for a species, for example, via the Human Genome Project, this is used as a reference for further individuals of the same species. Thus, the whole genome is reconstructed by aligning the short reads on the reference genome. Thereafter, the variation from the reference DNA can be inferred to understand specific traits or abnormalities in the individual.

The problem we address in this paper is that of DNA sequence alignment, while those of quantum accelerated de novo assembly and sequence analysis are addressed in other research work of ours [8,9]. The Broad Institute's GATK DNA processing pipeline [10] is a widely used toolset for this purpose, which includes several processing stages like map-to-reference, duplicate marking and variant calling. One of the most compute-intensive processing stages is the map-to-reference stage used for aligning the reads for reference-based DNA reconstruction [11]. Due to the huge data volume of over-sampled reads, whole-genome sequencing (WGS) of a single human take days on computing clusters, limiting the applicability of WGS in personalized medicine. This motivates the demand for acceleration using quantum computation, as even a polynomial speed-up can provide huge benefits on a production environment.

Techniques currently used in the genomics industry depend heavily on heuristics to tackle the volume of data that needs to be processed. However, the heuristics used in industrial alignment algorithm, for example, BWA-MEM, are trade-offs between the quality of the solution and the tractability on the computing platform. Given access to a superior computing paradigm, the sequence reconstruction algorithm is thus built bottom-up, to achieve the best possible quality. We construct a heuristic-free quantum algorithm primitive to achieve a high performance global alignment algorithm. This is explored in this research, where we present the quantum algorithm corresponding to a naive sub-sequence alignment, which can currently be implemented as a proof-of-concept using simulators. The presented algorithm can further be refined in the future by adding a gap penalty and dataset specific heuristics when quantum processing platforms mature to the stage where these algorithms can be implemented in a quantum accelerator.

In order to map a sub-sequence of characters (or short read) to a reference sequence, the Levenshtein edit distance is commonly used as a metric for approximate matching of the sub-sequence, spanning the comparison length. The Levenshtein edit distance is upper bounded by the Hamming distance between the two sequences. In our work, we use the Hamming distance as the cost function for matching owing to its simplicity for the quantum implementation. Given the reference sequence T and a short read P of length N and M, respectively, the sub-sequence alignment problem is defined as the index $i \in N$ of T, where the alignment of P starts, which gives the minimum edit distance. The short read is matched for each of the $N - M + 1$ starting indices in the reference genome. The alignment algorithm outputs the index of the minimum Hamming distance and optionally, the nearest match in the reference. Note that this concerns the typical problem setting of linear nuclear DNA instead of circular organellar DNA like in mitochondria or chloroplasts.

An example of this naive alignment approach is illustrated in Figure 2. The four colors represent the four bases in the DNA, which can be encoded as a four level system (radix-4 number) or with 2 bits (or, qubits) each. The short read in this case can be aligned at $32 - 5 + 1 = 28$ locations on the reference genome, resulting in a Hamming distance for each match. For example, at alignment index 0, only the green (second base of the short read) matches with the reference, thus, resulting in a Hamming distance of 4. By running a classical linear search it can be inferred that the minimum Hamming distance (of 1) occurs at the index 21. It is important to note that, although the reference genome and short read are of the same species (e.g., humans), an exact match might not be found. This variation can be from two sources, read errors in the sequencing machine, as well as genetic variation from individual traits or abnormalities with respect to the reference. Thus, it is important to account for approximate matching in the quantum algorithm design.

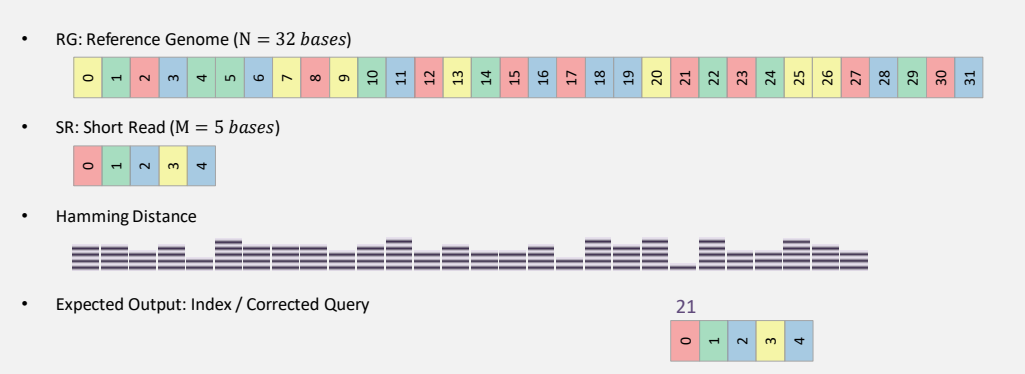

Figure 2. DNA sub-sequence alignment problem.

3. Quantum Search

In this research, we will be developing the quantum search algorithm [12,13] on an unstructured database, as proposed by Lov Grover. Grover's search offers a quadratic speedup over a classical linear search. A quadratic speedup may seem less lucrative with respect to exemplary quantum algorithms (such as Shor's factorization); however, it is the only search method possible for unstructured data and is provably optimal [14] in query complexity. Thus, under reasonable assumptions of computational complexity classes (e.g., P ! = NP), Grover's search based approach is the best algorithm in both classical and quantum domains. Near-term approaches based on Quantum Approximate Optimization Algorithm (QAOA) are now being developed to bridge the gap between Grover's search and current hardware limitations. The speedup from such quantum heuristics are yet to be theoretically proven and are heavily dependent of the hardware specifications (noise characteristics, connectivity, ansatz, gate-set, etc.). Since DNA sequence reconstruction requires more advanced quantum hardware, we take a hardware agnostics approach, focusing on a coherent protocol that preserves the Grover type speedup. It is unlikely that DNA sequence reconstruction will lead to a higher speedup with our current understanding of the encoding structure in the data. However, a polynomial speedup can prove to be path-breaking in industrial pipelines, where improvements by state-of-the-art alignment heuristics mostly progress by constant factor speedups for specific datasets.

Grover's search consists of three main steps between state initialization and measurement, as shown as the quantum circuit blocks in Figure 3. The algorithm creates a uniform superposition of all basis states by applying the Hadamard gate on all qubits in the all-zero state. The black box Oracle marks the solution state. Then the amplitude of the solution state is amplified by an inversion-about-mean operation by the Grover diffusion gate. Repeating the last two steps a quadratic number of steps with respect to the number of qubits, leads to a high probability of measuring the marked solution state. Thus, the search reduces the query complexity to the Oracle by a quadratic factor compared to a classical linear search.

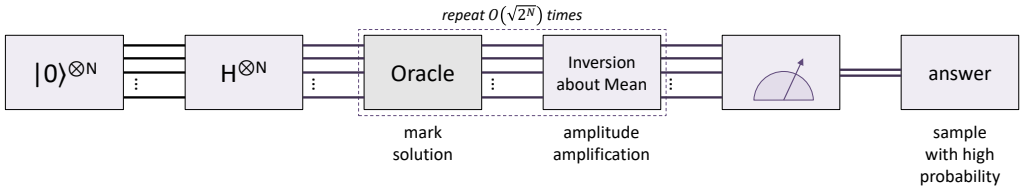

Figure 3. Grover's search steps.

Grover's search was enhanced by subsequent research that will allow us to apply this algorithm in our context. These improvements are:

- Multiple known number of solutions [15];
- Arbitrary distribution of initial amplitude [16];

- Multiple unknown number of solutions by randomizing iterations over multiple runs [17];
- Multiple unknown number of solutions by a priori counting the number of solutions [18].

The encoding of an application [19,20] to the Oracle is not explicitly described in these papers, and Grover's original paper assumes the execution of the Oracle in constant time, for an overall polynomial speedup. However, a complete description of an algorithm in the circuit model needs an explicit representation of the construction of the Oracle with quantum gates as described in this work.

4. Related Algorithms

Associative memory, also called content-addressable memory (CAM), is a type of memory organization where instead of the index of the element to be retrieved (similar to a random-access memory, RAM), a partial description of an element is passed as the input query. As introduced in Section 2, the element in the memory with the nearest match to the query is retrieved. Here, we review the differences and application of these approaches, without the detailed proofs of the quantum circuit construction from the original articles.

4.1. Quantum Associative Memory

The idea of quantum associative memory (QuAM) [21–25] was developed under the umbrella of quantum neural networks (QNN). Intuitively, the entire parallel search operation is reduced to operations on a superposition of states (memories). This results in either an exponential increase in the capacity of the memory, or a reduction in the number of comparisons to constant time.

The algorithm consists of two major blocks, a pattern store and a pattern recall. The pattern store starts from an all-zero initial state as is standard in all gate-based quantum algorithms. The information of the reference text string or DNA, T, is encoded as a superposition database of smaller substrings. This set T_M of length M has substrings $T_M(i)$ (where $i \in \{0 \cdots (N - M + 1)\}$) made from T, each starting from a consecutive index. This is compared with a recall pattern, P representing the query. If an exact match in the stored database is found, it is retrieved as the measurement output. However, if a partial or approximate version of P is queried, that is, some characters of the string are not known exactly, the algorithm returns a random output. Since in practice, P can be inaccurate, our algorithm should be able to retrieve the most similar string from the stored database.

4.2. Quantum Associative Search

A major improvement for the quantum associative memory is the use of distributed queries [26]. Using this concept, the associative memory solves the pattern completion problem; that is, it can restore the full pattern when initially presented with a partial pattern such that the known parts exactly coincide with some part of a valid full pattern. This allows the associative memory to also retrieve valid memory items when presented with noisy versions of a partial pattern. This improvement solves the problem of associative search for which no part of the input stimulus is guaranteed to be noise-free. It is desirable to retrieve the memory state, which is most similar to the given stimulus. This kind of memory is called a pattern correcting associative memory.

For strings containing only 0 s and 1 s (binary alphabet), this corresponds to finding the minimum Hamming distance between the query and the memory states. Amplitudes are distributed in the distributed query such that the maximal value occurs for some definite state p (the provided search query pattern) and the amplitudes of the other states x decrease monotonically with Hamming distance $h(p, x)$. The binomial distribution matches the required query model. p is the query center of the binomial distribution. Let $d = |x|$ be the number of qubits required to store the memory states. For all $x \in \{0 \cdots (2^d - 1)\}$, let

$$|b_p^x\rangle = \sqrt{\gamma^{h(p,x)}(1-\gamma)^{d-h(p,x)}}$$

where γ incorporates a metric into the model, which tunes the width of the distribution permitting the comparison of the similarity of the stimulus and the retrieved memory at a variable scale. The unitary Oracle transformation can be formed as

$$O = I_{2^d \times 2^d} - 2\left|b_p\right\rangle \left\langle b_p\right|.$$

Further modification [27] to the model of a distributed query is carried out by merging the concept of the memory state Oracle with the binomial function based Oracle. This is depicted in Figure 4. After the pattern is marked by the binomial Oracle, the entire superposition for stored memories are marked and amplified. Thereafter, the standard Grover iteration is carried out. The reconstructed pattern from among the stored memories (entries in the database) is retrieved with high probability once the qubits are measured. While the search function is similar to our use-case, the Oracle in this case needs to change for each short read and the reads needs to be indexed with respect to the reference.

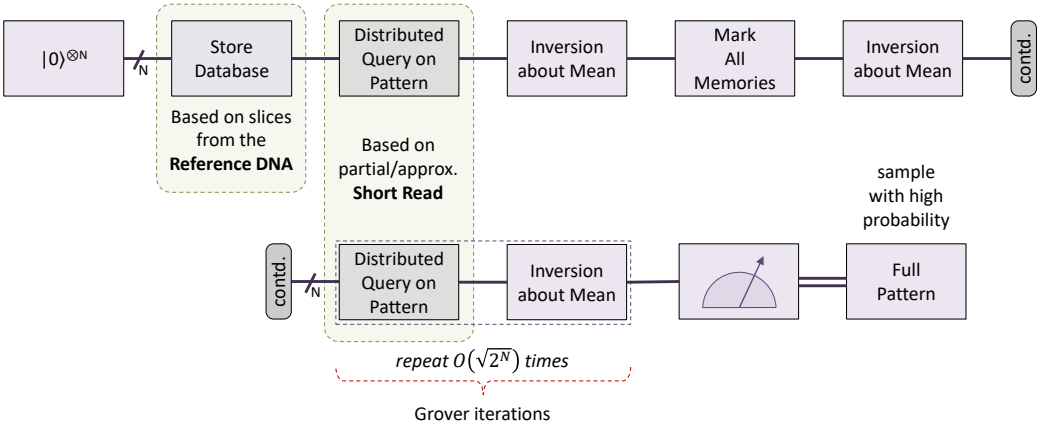

Figure 4. Quantum associative search algorithm with distributed query.

4.3. Quantum Indexed Memory

The location in the reference database of an exact or closest match to a query pattern is the alignment index with the minimum Hamming distance between the sequences. This method [28] was developed for a similar problem of amino-acid sequence matching. A block diagram of the algorithm is shown in Figure 5.

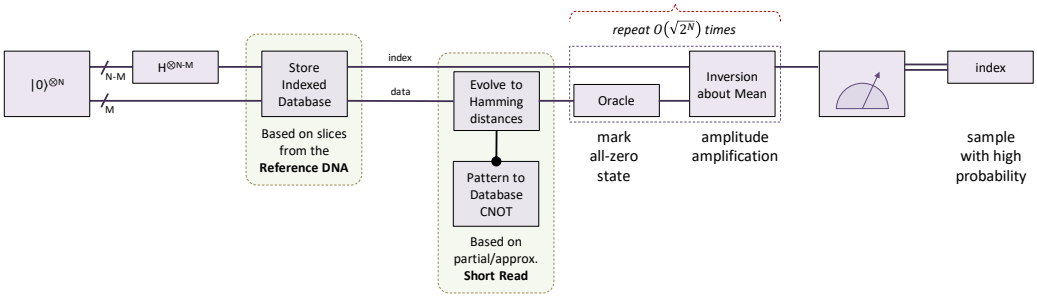

Figure 5. Quantum phone directory algorithm.

The initial state is composed of two quantum registers of $N - M$ and M qubits; the index and the pattern forming the quantum phone directory (QPD)—similar in architecture to a phone directory with name and number. Initially, the index is set to a full superposition. Then, based on the index, the data is stored in the database. For each tagged index, a subsequence of the reference DNA is stored as a basis state of the quantum superposition. Essentially, the set of patterns are sorted into an ordered list due to the second register of the database that tags the data. The initial state is described as:

$$|\psi_0\rangle = \frac{1}{\sqrt{N-M+1}} \sum_{i=0}^{N-M} \Big(|T_M(i)\rangle \otimes |i\rangle \Big),$$

where $T_M(i)$ represents a sub-sequence of the reference T of length M starting at the position i.

The next step in the algorithm evolves the data qubits to their Hamming distances with respect to the search pattern P, which in the case of DNA reconstruction is the short read from the sequencing machines. This operation can be done on the entire superposed state highlighting the parallel transformation power of quantum operators. A set of CNOT gates with the query pattern P as the control on the data qubits results in the data register evolving to the superposition of Hamming distances between each original data and the query pattern. The black-box nature of the Oracle function is thus simplified. For a perfect match, the Oracle now needs to mark the states with the value of 0, thus making it a fixed function with no dependence on either the reference or the search pattern. Once the state is amplified according to the modified Grover's algorithm (for an unknown number of solutions), the location of the sequence in the database can be determined by making a measurement on the second part of the entangled register, that is, the tag qubits. Note that both the index and data qubits need to be part of the Grover iterations as the two quantum registers are entangled.

For approximate matching, the Oracle needs to be modified such that it finds the minimum value of the Hamming distance, instead of an exact 0. We propose an algorithm that merges the improved distributed query approach on the indexed quantum data structure, to retrieve the index of the closest match.

5. Proposed Algorithm: QiBAM

The algorithm presented here inherits some of the features from the approaches highlighted in the previous sections. It is a novel quantum pattern matching algorithm specifically designed for the context of genome sequence reconstruction. These requirements for the quantum algorithm are:

- Approximate query matching to handle read errors;
- A constant Oracle to prevent run-time quantum circuit compilation;
- Retrieval of the associated index in the reference along with the corrected query.

Our proposed algorithm meets these three requirements. The quantum circuit blocks for the proposed quantum indexed bidirectional associative memory (QiBAM) algorithm is depicted in Figure 6.

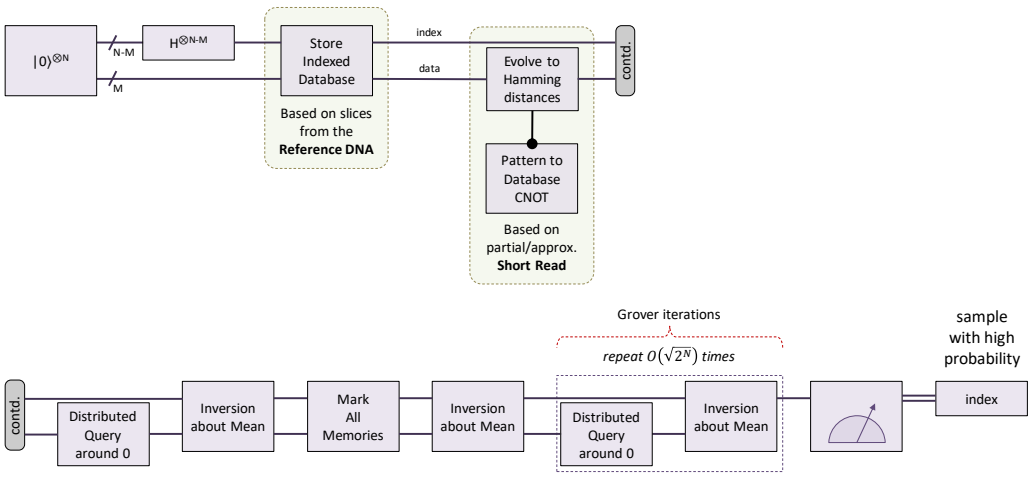

Figure 6. Quantum circuit blocks of the proposed QiBAM algorithm.

The initialization of the algorithm follows the design as described in Section 4.3. The tag qubits encode the pattern index, while the data qubits form the associative memory. Thus, the pattern store step in the associative memory algorithm (refer Section 4.1) is modelled as a quantum phone directory encoding—which allows the recall of the tagged index corresponding to the query pattern completion/correction. Once the data are encoded, the Hamming distance evolution is carried out. This solves the black-box nature of Grover's marking Oracle.

A distributed query is then defined over the associative memory with the query center at zero Hamming distance. This is based on the quantum associative search, now modified with $p = 0$, such that,

$$|b_0^x\rangle = \sqrt{\gamma^{h(0,x)}(1-\gamma)^{d-h(0,x)}}.$$

The value of γ is empirically set to 0.25 based on the quantum simulation results. In principle, this free parameter for the application of DNA sequence reconstruction needs to be tuned based on the error rate of the sequencing machine which generates the short read patterns that need to be aligned to the reference DNA.

Thereafter, the minimum Hamming distance (the evolved data string with the largest number of zeros in the basis string in the superposition) is amplified by the process of distributed quantum associative search. Finally, the index qubits are read out to sample with high probability the index where the search pattern best matches the reference.

While the societal relevance of the application presented in this research is enormous, it is important to restate that currently available state-of-the-art quantum processors are not yet capable to implement a proof-of-concept of this algorithm due to the limitations in the qubit multiplicity, error rates and connectivity. Additionally, since the reference DNA needs to be accessed for each run, as with most quantum algorithms, we assume the availability of a QRAM. Efficient realizations of QRAM is a separate research topic. Additionally, we propose that, if an efficient QRAM implementation is not possible, multiple copies of the QiBAM algorithm can be executed in parallel based on the multiplicity of the qubit, since the result for each search pattern is independent of another. In this context of parallelism, we envisage a multi-core quantum processor where each subset of qubits would be executing the QiBAM on a specific DNA search string similar to how single-instruction-multiple-data (SIMD) is implemented within the cores on a GPUs. Based on an initial version of this research, an extension to Multiple Sequence Alignment has been carried out by [29].

In further sections, we show a proof-of-concept on classical simulation of quantum computation instead of experimenting with NISQ hardware. It is hard to predict the timeline of quantum processors that will be able to implement this algorithm; however, given the current research thrust and development, a 5–10 years estimate is reasonable [3]. A quantum computer architectural stack aids in developing a quantum algorithm while being agnostic to the underlying hardware, such that, the programs implementing the QiBAM algorithm can readily be ported to any quantum processor once the technological maturity is reached.

5.1. Quantum Indexed Multi-Associative Memory

If both the quantum registers are accessible for gate operations, the associative memory can be operated (searched) based on either of the registers thus allowing a bidirectional associative search and retrieval. We can search with the index (in a RAM mode), or by the data (in a CAM mode).

This idea of associative memory can be generalized to multiple quantum registers holding different attributes of the data that needs to be analyzed. This generalization of QiBAM is called Quantum indexed multi-associative memory (QiMAM). For example, the quality value of the reads can be stored in register 2, and the chromosome number of the read in register 3, in addition to the index and the pattern. More complex search queries can be formed based on this entangled quantum database, for example, a search for the index of a specific noisy query pattern among high quality reads in a particular

chromosome. Such a database is particularly useful for applications like Gene Ontology, Sequence Ontology and Genome Wide Association Studies.

5.2. Qubit and Gate Complexity

Three important parameters are used as metrics for a quantum algorithm. The space and time complexity of classical algorithms, correspond respectively, to the scaling behavior of the number of required qubits and the number of required gates for our quantum computation model. The detailed derivations of the complexity are presented in [7] and here we present the final results. The probability of reading out the desired solution is another important metric though it is dependent on the specific data. Currently, however, it is not possible to use real DNA sequence ensembles in the implementation due to limitations in simulation and available hardware.

The qubit (space) complexity is the aggregate of the qubits used for encoding the data register, tag register and ancilla. We do not consider overheads for error-correction, mapping, routing or other factors besides the algorithm logic. For QiBAM, the qubit complexity is the same as the algorithm in Section 4.3.

Let the number of qubits required for the data and tag registers be $q_d = \lceil log_2(A) \rceil M$ and $q_t = \lceil log_2(N - M) \rceil$, respectively. The total number of qubits is thus $Q = q_d + q_t + 1$, yielding a typical estimate for the DNA alphabet, the human genome and read length (e.g., from Illumina sequencers) $A = 4$, $N = 3 \times 10^9$ and $M = 50$, as 133 fully-connected logical qubits. While this is beyond the reach of current NISQ era hardware, we note that the number of required qubits to achieve quantum advantage is considerably less than the exemplary Shor's algorithm for factorization for the RSA coding in the cryptography context.

The gate complexity depends on the choice of the universal gate set. Here, the gate set used consists of $\{H, Ry, Rz, C_cX, \}$, where $c = 0$ is the X-gate, $c = 1$ is the CNOT gate, $c = 2$ is the Toffoli gate, and so on. Higher-order controls can be decomposed with ancilla qubits [30]. The translation of the gate complexity to the run-time for a specific quantum processor platform would depend on the native gate set available on the hardware. Most modern quantum compilers can convert between universal gate sets in linear compile time and polynomial gate overhead. Thus, the universal gate set considered here is without reference to any specific quantum processor platform. The initialization kernel is first decomposed. First, q_t Hadamard gates are used on the tag qubits to create a superposition of solution states. Then, conditioned on each tag, the corresponding shifted sub-string of the reference is encoded. The binary encoding of the tag requires half the controls as inverted on average, requiring X-gate dressing totaling $q_t 2^{q_t}$. We can use results from the statistical distribution of the nucleotide frequencies to estimate the typical case complexity of the quantum circuits. The Chargaff's rules [31] state that the DNA nucleotides are distributed approximately $1/4$ in each sub-string. This requires $q_d/2$ targets for each tag encoded sub-string. Thus, the total initialization and Hamming evolution require $q_t H + q_t 2^{q_t} C_0 X + q_t q_d / 2 C_{q_t} X$ gates.

The distributed query step depends entirely on the chosen decomposition method for the unitary and the native gate set. The unitary decomposition method using Quantum Shannon Decomposition (QSD) [32] has a complexity of $3(4^{n-1} - 2^{n-1})$, where n is the dimension of the unitary. The mark memory operation would evolve the states in the initial quantum database. This requires the controlled-Z (also called, CPhase) quantum logic gate over the tag and data qubits for each of the 2^{q_t} memories of which $N - M + 1$ are memories from the reference genome. The data qubits, following Chargaff's rule, would have half the bits of 1, thus using a total of M qubits in average for the compute and uncompute. The tag qubits would follow the same behavior as the initialization phase, with average X-dressing of $q_t 2^{q_t} C_0 X$ gates. Thus, the total for the marking memory is $(2^{q_t})\{2H + (M + q_t)C_0X + C_{q_d+q_t-1}X\}$. Note that this is the Oracle complexity of the quantum search, which for all practical implementations, always needs to be considered in addition to the polynomial speedup of query complexity. Finally, the Grover diffusion

operator is decomposed to $\{2(q_d + q_t) + 2\}H + 2(q_d + q_t)C_0X + C_{q_d+q_t-1}X$ gates. The details of our implementation of the unitary decomposition algorithm can be found in [33].

While the exponential reduction in space (qubit) complexity is easy to visualize as proposed in the quantum associative memory architecture, the polynomial speedup is not so pronounced. This is due to the exponential terms in the worst-case analysis for the QSD and binary encoding. Many of these gates can be scheduled in parallel in a quantum processor flattening the complexity. The focus of this research is on the correctness of the algorithm for the presented application, while retaining the Oracle query complexity of the Grover search. However, since the Oracles are deterministically computable, they can be aggressively optimized by the compiler before run-time.

5.3. Run-Time Architecture

While accelerating a classical algorithm on classical data, the interactions with a quantum computer will ultimately require inter-conversion of the input and output to and from quantum data. Besides, the control electronics of a quantum processor are classical circuits. In the initial days of a quantum computer (what is called as the pre-universal quantum computing era), we are most likely to use quantum computers for very specialized applications, where we can use the power of superposition and entanglement for a computational advantage. Thus, even though a quantum algorithm can do everything a classical algorithm can (within polynomial factors), it is not likely that quantum computers will be standalone systems. In computer engineering terms, quantum systems will be an accelerator, specialized computing processors like GPU (for graphics and parallel simple computations), NPU (for neural networks), FPGA (for hardware optimized circuits), DSP (for digitizing analog signals), and so forth. The current quantum processors are much less powerful in terms of space (number of qubits), time (coherence), accuracy (noise) and connectivity (qubit interactions). This makes us really selective in choosing which part of the application algorithm we want to offload to the quantum accelerator.

The quantum accelerator is architecturally layered; this is called the stack [34]. The stack, as shown in Figure 7, is inspired by the currently existing classical computers. The stack layers from the application layer to the physical substrate are:

- The application can be described in hybrid quantum-classical logic as mathematical state evolution designed to perform the desired task. They need to be decomposed into programming constructs as input to the compiler;
- To ease development of algorithms, many compilers now offer libraries which consist of an arsenal of primitives that help a quantum algorithm developer;
- Compiler and programming language (like OpenQL) is the interface for the algorithm designer to precisely define the quantum operators and state in abstracted high-level constructs;
- The compilation process generates an assembly level code (common-QASM) specifying the gate operations;
- Quantum runtime unit is responsible for scheduling the operations required for the compiler code. This includes quantum error correction (QEC) and qubit logical to physical mapping. The input for this is provided as hardware parameters. The executable-QASM is generated via this process;
- Quantum Instruction Set Architecture defines the runtime operations of both classical control and quantum parts of the algorithm. It encapsulates the hardware dependence;
- Micro-architecture takes into account the precise timing controls and the instruction pipelines;
- Quantum-classical interface comprises of ADC and DAC and their controls for interacting with the physical qubits. Finally, the quantum processing unit houses the physical qubits. This can be superconductors, semiconductors, or other types of competing qubit technologies;
- Hardware agnostics application development (as discussed in this paper) can bypass to directly interface the cQASM with the simulator (which in turn runs on the classical

CPU). The simulated qubits are perfect in nature for testing the functionality of the algorithm.

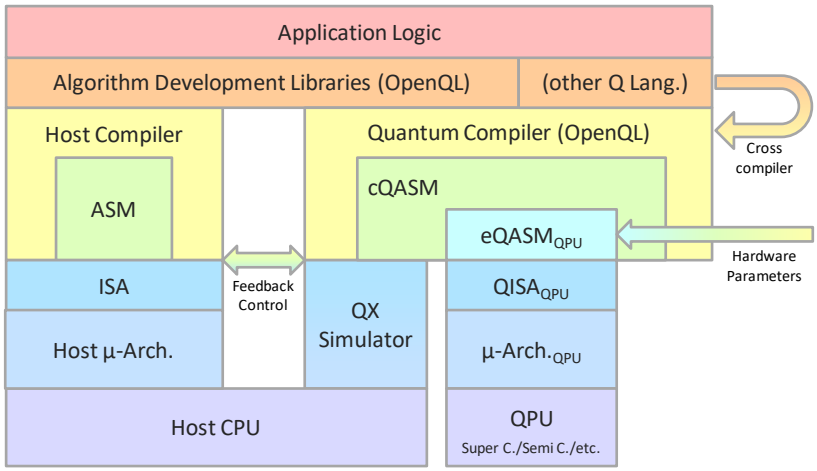

Figure 7. Quantum architectural stack.

An isolated discussion of a specific quantum algorithm is not sufficient for near-term implementation. The quantum algorithm would have interfaces with other software modules running in parallel as shown in Figure 8.

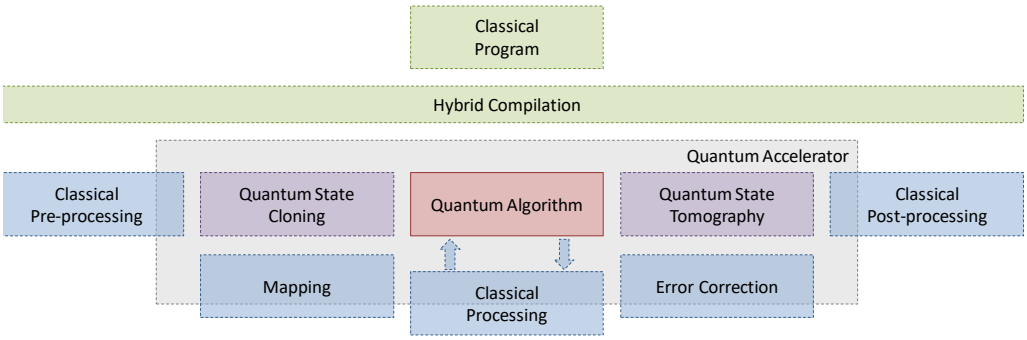

Figure 8. Quantum Algorithm (block with solid outline) and interfacing software architecture.

There are three factors that contribute to the overall run-time of a general quantum algorithm:

- Algorithm: This pertains to the core algorithm running on a simulator, where the internal state vector can be accessed. It refers to the inherent gate complexity of the algorithm and other classical pre/post-processing involved;
- No-cloning: If the internal state vector cannot be accessed (like in real quantum processors), the experiment needs to be repeated multiple times and the measurement is aggregated. Most algorithm demands a statistical estimate of the state's probability distribution. The central tendency of these measurements is the resultant output from the quantum algorithm.
  Quantum state tomography is an active area of research. Advanced methods based on linear inversion, linear regression, maximum likelihood, Bayesian, compressed-sensing and neural networks [35] exists for estimating the state with fewer tomographic trials.
- Experimental: For algorithm development (using perfect qubits) and proof-of-concept testing, a simulator platform is preferable, such as the QX Simulator used for this research. After sufficient confidence in the logic is established, it needs to be ported to an experimental quantum processing unit [36]. This adds complexity overhead for topological mapping [37] and quantum error correction cycles [38].

Thus, every quantum algorithm that depends on a probabilistic result in a noisy environment needs to be repeated, adding a multiplicative factor to the inherent gate complexity.

$$O(f(\text{experimental}) \times g(\text{no-cloning}) \times h(\text{algorithm}))$$

6. QiBAM Results on DNA Sequences

The implementation of this algorithm is carried out in OpenQL [6] and the QX Simulator [5]. The OpenQL framework allows hybrid quantum-classical coding in Python or C++, compiling and optimizing quantum code to produce the intermediate Common QASM (cQASM) [39] and the compiled Executable QASM (eQASM) for various target platforms (superconducting qubits, spin-qubits, NV-centers, etc.) The Qxelarator library allows the execution of the compiled QASM on the QX binary and receives the measurement outcomes in the high-level OpenQL code encapsulating the quantum architecture (in this case, a simulator), allowing interleaving classical and quantum code blocks in a single program. QX is a universal quantum computer simulator that takes as input a cQASM file and provides thorough aggressive optimization, high simulation speeds for qubit state evolution. Our experimental setup (with 28 HT cores, at 2.00 GHz and 384 GB memory) can simulate ≈35 qubits if the states and operations are non-sparse. For this algorithm, in the development stage, perfect qubits are used without any error model in the QX simulator (like depolarizing model) and the unitary operations have full fidelity. However, measurement aggregate is used instead of accessing the internal state vector (as allowed for ease of development in most quantum simulators), as this divergence from the ideal distribution has a considerable impact on the algorithm metrics.

In the following example, we will show the results of implementing the QiBAM algorithm on an actual DNA sequence. We show how to search for a pattern of length 2 over the DNA alphabet (*A, C, G, T*). A minimum-length super-string that includes all possible length-2 DNA substrings is *AATTGTCTAGGCGACCA*. This minimal-length super-string helps with verifying the correctness of the quantum algorithm exhaustively. To test the distributed query capabilities of the algorithm for mismatches in the reference sequence, the last memory, *CA* is not encoded, making the reference genome as *AATTGTCTAGGCGACC*. This is encoded as the input database shown in Figure 9. The radix-4 symbols of the DNA alphabet are encoded in binary as $00, 01, 10, 11$, while the tag is encoded as a 4 bit binary coded decimal value. Thus, the database (of the first two columns in Figure 9) is encoded in a quantum superposition as: $|\text{tag\_data}\rangle = a(|0000\_0000\rangle + |0001\_0011\rangle + |0010\_1111\rangle + |0011\_1110\rangle + |0100\_1011\rangle + |0101\_1101\rangle + |0110\_0111\rangle + |0111\_1100\rangle + |1000\_0010\rangle + |1001\_1010\rangle + |1010\_1001\rangle + |1011\_0110\rangle + |1100\_1000\rangle + |1101\_0001\rangle + |1110\_0101\rangle + |1111\_\text{xxxx}\rangle)$, where the amplitude $a = \frac{1}{\sqrt{16}}$ Since the tag qubits are in a full superposition, the extra tags (e.g., 1111 in this case) are allocated any arbitrary value (xxxx) and measurements of index beyond the valid range can be ignored.

Now the search query is chosen as *CA*. The search pattern conditionally toggles the database to evolve it to the Hamming distance. Since *CA* is not present in memory, we expect the nearest patterns (approximate matches) to have a higher probability of detection, which are *{AA, TA, CG, CC}* (with a Hamming distance of 1 in the encoding). This results in the superposition: $|\text{tag\_dist}_{\text{Ham}}\rangle = a(|0000\_0100\rangle + |0001\_0111\rangle + |0010\_1011\rangle + |0011\_1010\rangle + |0100\_1111\rangle + |0101\_1001\rangle + |0110\_0011\rangle + |0111\_1000\rangle + |1000\_0110\rangle + |1001\_1110\rangle + |1010\_1101\rangle + |1011\_0010\rangle + |1100\_1100\rangle + |1101\_0101\rangle + |1110\_0001\rangle + |1111\_\text{xxxx}\rangle)$. It can be verified for this case that, at indices $0000, 0111, 1011, 1110$, the dist_{Ham} quantum register has the minimum number of 1 s and thus will be amplified by a distributed query around 0000 followed by Grover diffusion for the required number of iteration. The estimated trend for a higher solution probability should be in line with decreasing Hamming distance, as plotted in Figure 10, with the tag on the *X*-axis and the Estimate Amplification on the *Y*-axis.

Tag	Input Database	Input Encoding	XOR with CA = 10	Hamming Distance	Estimate Amplification
0	AA	00	10	1	3
1	AT	03	13	3	1
2	TT	33	23	3	1
3	TG	32	22	2	2
4	GT	23	33	4	0
5	TC	31	21	2	2
6	CT	13	03	2	2
7	TA	30	20	1	3
8	AG	02	12	2	2
9	GG	22	32	3	1
10	GC	21	31	3	1
11	CG	12	02	1	3
12	GA	20	30	2	2
13	AC	01	11	2	2
14	CC	11	01	1	3

Figure 9. Quantum database for search pattern *CA* and reference string *AATTGTCTAGGCGACC*.

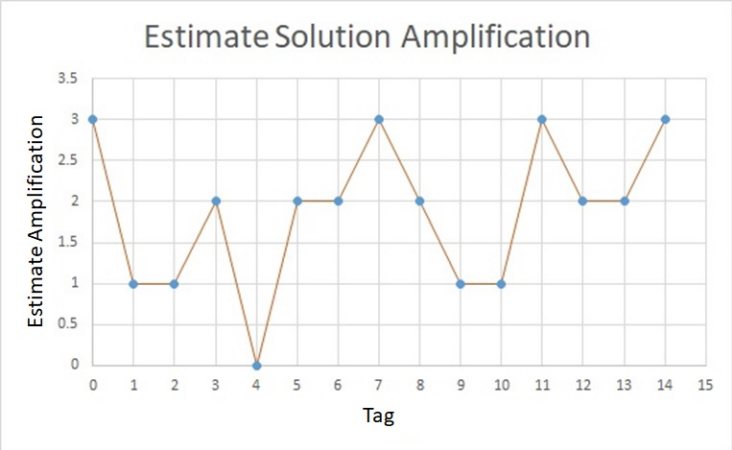

Figure 10. Estimate of the solution probability trend as a numerical estimate of expected results of a sample run.

The OpenQL algorithm is executed with the Qxelarator library, returning the internal state vector. The reference sequence and the search query is hardcoded in the Python program for this test but can be streamlined to be directly read from an industry-standard file like the FASTQ format from commercial DNA sequencers. The result from the run is plotted in Figure 11. The left vertical axis shows the staircase state curve for the tag qubits, while the right vertical axis shows the measurement probability of each individual state. There are four tag qubits and four data qubits (2 Radix-4 numbers for a DNA search pattern of length 2). Thus, the total state space is $2^8 = 256$. The double-precision floating point naive state vector simulation of this algorithm requires 32 Kb to store the state space, while each of the ≈230 gates requires a matrix of 8Mb. The states with prominent probabilities are the memory states. The envelope of these states (ignoring the spurious memories) gives the same trend as our estimate in Figure 10, verifying the correctness of our implementation.

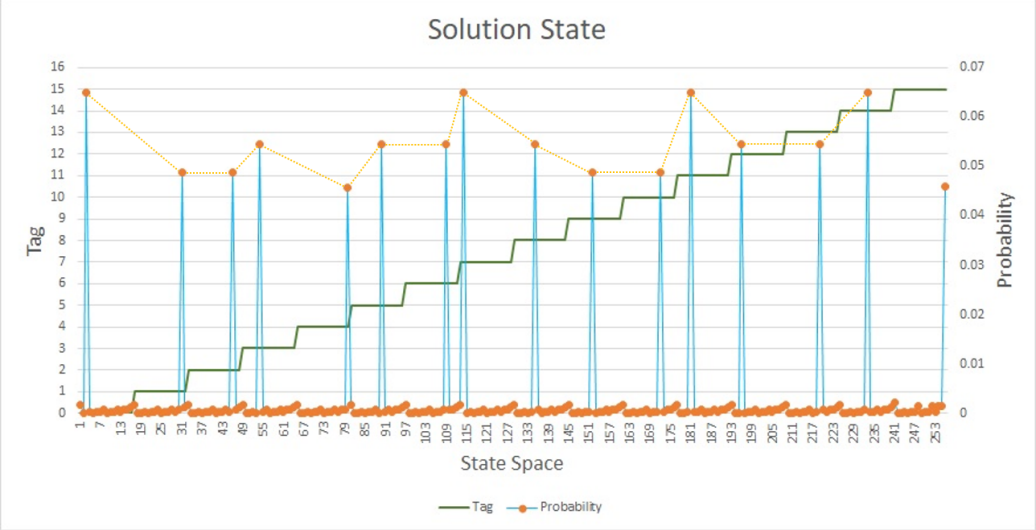

Figure 11. Results of a sample execution of QiBAM on QX Simulator, which matches the results derived analytically in Figure 10.

7. Conclusions

This research is motivated by the bottleneck of DNA sequence reconstruction in genomics, and explores how quantum acceleration can be applied in this domain. This is the first time a quantum pattern matching algorithm is specifically designed, keeping in mind genomic sequences.

The idea of associative memory is extended to an indexed directory of DNA sequences spliced from the reference genome. In addition to taking into account the DNA alphabet, since reads can contain errors, a distributed query for approximate matching is designed. This is applied over the superposition of a quantum state, thereby storing an exponential number of patterns. A constant Oracle is designed based on minimizing the Hamming distances. This eliminates the bottleneck of compiling the query differently for every short read at run-time. The associated index in the reference is retrieved, instead of the corrected query by entangling the index with the sequence database.

This paper also discussed the complexity of the algorithm taking into account system parameters as well. This algorithm is generalized to a generic quantum data structure for multi-dimensional search. The algorithm is implemented and verified in the OpenQL environment with the QX Simulator as the backend.

This research is the first exploration [7] towards a roadmap project [34] undertaken in the Quantum Computer Architecture lab at the Delft University of Technology, to design a *full-stack quantum accelerator architecture that is domain-specific for genomics*. While the computer application community awaits a large quantum processor capable of real-world problem size execution; our research is carried out on high-performance simulator platforms to test the functional proof-of-concept execution pipeline on small DNA test patterns. Further research is currently being carried out to adapt the algorithm for near-term quantum computers using parameterized quantum-classical hybrid variational approaches.

Author Contributions: Conceptualization, A.S., K.L.M.B. and Z.A.-A.; methodology, A.S., K.L.M.B., C.G.A. and Z.A.-A.; software, A.S.; validation, A.S. and Z.A.-A.; formal analysis, A.S.; investigation, A.S., K.L.M.B. and Z.A.-A.; writing—original draft preparation, A.S.; writing—review and editing, A.S., Z.A.-A. and C.G.A.; visualization, A.S., K.L.M.B. and Z.A.-A.; supervision, Z.A.-A., C.G.A. and K.L.M.B.; project administration, Z.A.-A. and K.L.M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are openly available on GitHub at DOI: 10.5281/zenodo.5482659 (accessed on 9 September 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Feynman, R.P. There's plenty of room at the bottom: An invitation to enter a new field of physics. In *Handbook of Nanoscience, Engineering, and Technology*, 3rd ed.; CRC Press: Boca Raton, FL, USA, 2012; pp. 26–35.
2. Riesebos, L.; Fu, X.; Moueddenne, A.; Lao, L.; Varsamopoulos, S.; Ashraf, I.; van Someren, J.; Khammassi, N.; Almudever, C.; Bertels, K. Quantum Accelerated Computer Architectures. In Proceedings of the 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 26–29 May 2019; pp. 1–4. [CrossRef]
3. Bertels, K.; Sarkar, A.; Ashraf, I. Quantum Computing—From NISQ to PISQ. *IEEE Micro* **2021**, *41*, 24–32. [CrossRef]
4. Smelyanskiy, M.; Sawaya, N.P.; Aspuru-Guzik, A. qHiPSTER: The quantum high performance software testing environment. *arXiv* **2016**, arXiv:1601.07195.
5. Khammassi, N.; Ashraf, I.; Fu, X.; Almudever, C.G.; Bertels, K. QX: A high-performance quantum computer simulation platform. In Proceedings of the 2017 Design, Automation & Test in Europe Conference & Exhibition (DATE), Lausanne, Switzerland, 27–31 March 2017; pp. 464–469. [CrossRef]
6. Khammassi, N.; Ashraf, I.; Someren, J. v.; Nane, R.; Krol, A. M.; Rol, M. A.; Lao, L.; Bertels, K.; Almudever, C. G. OpenQL: A Portable Quantum Programming Framework for Quantum Accelerators. *arXiv* **2020**, arXiv:2005.13283.
7. Sarkar, A. Quantum Algorithms for Pattern-Matching in Genomic Sequences. Master's Thesis, Delft University of Technology, Delft, The Netherlands, 2018.
8. Sarkar, A.; Al-Ars, Z.; Bertels, K. QuASeR: Quantum Accelerated de novo DNA sequence reconstruction. *PLoS ONE* **2021**, *16*, e0249850.
9. Sarkar, A.; Al-Ars, Z.; Bertels, K. Estimating Algorithmic Information Using Quantum Computing for Genomics Applications. *Appl. Sci.* **2021**, *11*, 2696. [CrossRef]
10. Broad Institute GATK Best Practices Pipeline. Available online: https://gatk.broadinstitute.org/hc/en-us (accessed on 9 September 2021).
11. Houtgast, E.J.; Sima, V.M.; Bertels, K.; Al-Ars, Z. Hardware acceleration of BWA-MEM genomic short read mapping for longer read lengths. *Comput. Biol. Chem.* **2018**, *75*, 54–64. [CrossRef]
12. Grover, L.K. A fast quantum mechanical algorithm for database search. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 1 July 1996; pp. 212–219. [CrossRef]
13. Grover, L.K. Quantum mechanics helps in searching for a needle in a haystack. *Phys. Rev. Lett.* **1997**, *79*, 325. [CrossRef]
14. Zalka, C. Grover's quantum searching algorithm is optimal. *Phys. Rev. A* **1999**, *60*, 2746. [CrossRef]
15. Boyer, M.; Brassard, G.; Høyer, P.; Tapp, A. Tight bounds on quantum searching. *Fortschr. Der Phys. Prog. Phys.* **1998**, *46*, 493–505. [CrossRef]
16. Biham, E.; Biham, O.; Biron, D.; Grassl, M.; Lidar, D.A. Grover's quantum search algorithm for an arbitrary initial amplitude distribution. *Phys. Rev. A* **1999**, *60*, 2742. [CrossRef]
17. Brassard, G.; Høyer, P.; Tapp, A. Quantum counting. In *International Colloquium on Automata, Languages, and Programming*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 820–831. [CrossRef]
18. John, M. Sampling with quantum mechanics. *arXiv* **2003**, arXiv:quant-ph/0306181.
19. Viamontes, G.F.; Markov, I.L.; Hayes, J.P. Is quantum search practical? *Comput. Sci. Eng.* **2005**, *7*, 62–70. [CrossRef]
20. Mateus, P.; Omar, Y. Quantum pattern matching. *arXiv* **2005**, arXiv:quant-ph/0508237.
21. Ventura, D.; Martinez, T. Quantum associative memory with exponential capacity. In Proceedings of the 1998 IEEE International Joint Conference on Neural Networks Proceedings, IEEE World Congress on Computational Intelligence (Cat. No.98CH36227), Anchorage, AK, USA, 4–9 May 1998; Volume 1, pp. 509–513. [CrossRef]
22. Ventura, D. Artificial associative memory using quantum processes. In Proceedings of the International Conference on Computational Intelligence and Neuroscience, Draper, UT, USA, 18–22 October 1998; Volume 2, pp. 218–221.
23. Ventura, D.; Martinez, T. Initializing the amplitude distribution of a quantum state. *Found. Phys. Lett.* **1999**, *12*, 547–559. [CrossRef]
24. Ventura, D.; Martinez, T. A quantum associative memory based on Grover's algorithm. In *Artificial Neural Nets and Genetic Algorithms*; Springer: Vienna, Austria, 1999; pp. 22–27. [CrossRef]
25. Ventura, D.; Martinez, T. Quantum associative memory. *Inf. Sci.* **2000**, *124*, 273–296. [CrossRef]
26. Ezhov, A.; Nifanova, A.; Ventura, D. Quantum associative memory with distributed queries. *Inf. Sci.* **2000**, *128*, 271–293. [CrossRef]
27. Njafa, J.P.T.; Engo, S.N.; Woafo, P. Quantum associative memory with improved distributed queries. *Int. J. Theor. Phys.* **2013**, *52*, 1787–1801. [CrossRef]
28. Hollenberg, L.C. Fast quantum search algorithms in protein sequence comparisons: Quantum bioinformatics. *Phys. Rev. E* **2000**, *62*, 7532. [CrossRef] [PubMed]

29. Giannakis, K.; Papalitsas, C.; Theocharopoulou, G.; Fanarioti, S.; Andronikos, T. A Quantum-inspired optimization Heuristic for the Multiple Sequence Alignment Problem in Bio-computing. In Proceedings of the 2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA), Patras, Greece, 15–17 July 2019; pp. 1–8. [CrossRef]
30. Gidney, C. Constructing Large Controlled Nots. Available online: https://algassert.com/circuits/2015/06/05/Constructing-Large-Controlled-Nots.html (accessed on 9 September 2021).
31. Yamagishi, M.E.B. *Mathematical Grammar of Biology*; Springer: Berlin/Heidelberg, Germany, 2017.
32. Shende, V.V.; Bullock, S.S.; Markov, I.L. Synthesis of quantum-logic circuits. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2006**, *25*, 1000–1010. [CrossRef]
33. Krol, A.; Sarkar, A.; Ashraf, I.; Al-Ars, Z.; Bertels, K. Efficient decomposition of unitary matrices in quantum circuit compilers. *arXiv* **2021**, arXiv:2101.02993.
34. Bertels, K.; Sarkar, A.; Hubregtsen, T.; Serrao, M.; Mouedenne, A.; Yadav, A.; Krol, A.; Ashraf, I. Quantum computer architecture: Towards full-stack quantum accelerators. In Proceedings of the 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2020; pp. 1–6. [CrossRef]
35. Torlai, G.; Mazzola, G.; Carrasquilla, J.; Troyer, M.; Melko, R.; Carleo, G. Neural-network quantum state tomography. *Nat. Phys.* **2018**, *14*, 447–450. [CrossRef]
36. Fu, X.; Rol, M.; Bultink, C.; Van Someren, J.; Khammassi, N.; Ashraf, I.; Vermeulen, R.; De Sterke, J.; Vlothuizen, W.; Schouten, R.; et al. An experimental microarchitecture for a superconducting quantum processor. In Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture, Cambridge, MA, USA, 14 October 2017; pp. 813–825. [CrossRef]
37. Lao, L.; Manzano, D.M.; van Someren, H.; Ashraf, I.; Almudever, C.G. Mapping of quantum circuits onto NISQ superconducting processors. *arXiv* **2019**, arXiv:1908.04226.
38. Varsamopoulos, S.; Bertels, K.; Almudever, C.G. Decoding surface code with a distributed neural network–based decoder. *Quantum Mach. Intell.* **2020**, *2*, 1–12. [CrossRef]
39. Khammassi, N.; Guerreschi, G.G.; Ashraf, I.; Hogaboam, J. W.; Almudever, C.G.; Bertels, K. cQASM v1.0 towards a Common Quantum Assembly Language. *arXiv* **2018**, arXiv:1805.09607.