2021 IEEE 18th International Conference on Software Architecture Companion (ICSA-C) | 978-1-6654-3910-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICSA-C52384.2021.00025

Expanding Data Encoding Patterns For Quantum Algorithms

Manuela Weigold, Johanna Barzen, Frank Leymann, Marie Salm
University of Stuttgart, Universitätsstr. 38, Germany
{weigold, barzen, leymann, salm}@iaas.uni-stuttgart.de

***Abstract*—As quantum computers are based on the laws of quantum mechanics, they are capable of solving certain problems faster than their classical counterparts. However, often algorithms which a theoretical speed-up assume that data can be loaded efficiently. In general, the runtime complexity of the loading routine depends on (i) the data encoding that defines how the data is represented and (ii) the data itself. In some cases, loading the data requires at least exponential time which destroys a potential speed-up. And especially for the first generation of devices that are currently available, the resources (qubits and operations) needed to encode the data are limited. Therefore, understanding the consequences of a particular data encoding is crucial. To capture knowledge about different encodings, we present two data encoding patterns that extend our previous collection of encoding patterns [1].**

***Index Terms*—Quantum Computing, Data Encoding, Patterns, Pattern Primitives**

I. Introduction

Recent advantages in quantum technology have led to a first generation of commercial quantum computers [2], [3]. Compared to their classical counterparts, quantum computers have the potential to solve certain problems faster [4]. For example, factoring large prime numbers [5] or unstructured search [6] can in principle be done faster by a quantum computer. These speed-ups are possible because quantum computers are based on quantum bits (qubits) and therefore can exploit *superposition* or *entanglement* which are unique characteristics of quantum mechanics. The quantum computers of this first generation have been coined *Noisy Intermediate Scale Quantum* (NISQ) devices [3] as they still have severe limitations: Their qubits are noisy and only stable for a limited amount of time until they decay. Measured by their number of qubits, the computers are of intermediate size; ranging from a few dozens to a few hundred qubits. Nevertheless, it is expected that hardware will further improve [2], [7], [8] and enable novel applications for quantum computers.

However, programming these quantum devices is challenging as their quantum nature as well as their hardware limitations must be taken into account. One key difference to classical computing is the way data is handled by quantum computers. Current quantum computers do not have access to a database or a quantum version of random access memory (RAM) [9]. Thus, in order to use data in a quantum computer, this data has to be loaded by encoding it into the state of the qubits. However, there are various data encodings that define how the data can represented by qubits. Note that the runtime of the loading routine depends on (i) the chosen data encoding, and (ii) on the data itself. It was proven that in the worst case, the loading routine is at least of exponential complexity, i.e., an exponential number of parallel operations is needed [10]. However, a common assumption of algorithms with a theoretical linear or exponential speed-up is that the process of loading data requires only logarithmic or linear time [11]. Aaronson [11] refers to these assumptions as *fine-print* of the algorithm, indicating that these are often overlooked or at least not prominent to readers. Especially in the current NISQ era understanding these runtime implications is crucial as only a certain amount of operations can be executed on noisy qubits. As developing software for quantum computers requires the interdisciplinary collaboration of e.g., physicists and computer scientists [12], a shared understanding of data loading in particular is needed.

This work is partially funded by the BMWi project PlanQK (01MK20005N).

In previous work [1], we formulated three patterns for data encoding in quantum computing to convey expert knowledge about different encodings. Patterns document proven solutions for re-occurring problems and are especially popular in the field of software [13]. In addition, fundamental building blocks used by patterns can be described as *pattern primitives* [14]. In this work, we first introduce pattern primitives to establish a common understanding and terminology of quantum computing fundamentals. Building upon these, we extend our previous collection of data encoding patterns by two new data encoding patterns. They describe a common solution (how data is encoded) that is used by at least three published quantum algorithms. With the first pattern, we explicitly document the limitations (aka fine-print) of many algorithms that use a quantum random access memory (QRAM) to load data. Since to this day, no hardware implementations of a QRAM is available, there are severe limitations, i.e., loading the data can not be guaranteed to be efficient. Our second pattern defines a different encoding that requires one qubit for each data-point.

The remainder of this paper is structured as follows: We first describe fundamentals and pattern primitives for quantum computing in Section II. We then give an overview of all encodings patterns and present the new patterns in detail in Section III. This is followed by a description of related work in Section IV. Finally, a conclusion is given in Section V.

II. Fundamentals

In this section, we first introduce patterns and pattern primitives. This is followed by an overview of quantum computing and quantum algorithms. Finally, fundamental building blocks of quantum algorithms are introduced as pattern primitives.

A. *Patterns and Pattern Primitives*

In this paper, we build on the concept of patterns of Alexander et al. [15]. They define patterns as structured, human-readable documents that document a proven solution to a reoccurring problem in a given context. Pattern primitives describe fundamental elements that re-occur in patterns and have first been introduced in the context of software architecture to describe basic architectural units [14], for example ports or components. Since then, the concept has been applied to various other domains [16]–[18]. Pattern primitives are more specific than the abstract solution of a pattern [14] and are therefore especially suited to establish a common ground and base terminology for patterns within one domain [19].

B. *Quantum Computing*

Since the first proposal of a quantum computer, various quantum computing models have been established which describe computations in a different manner, e.g., the adiabatic [20], gate-based [21] or one-way model [22]. In this work, we focus on the gate-based model that serves as the basis for many current devices [23], e.g., by IBM[1], or Rigetti[2]. However, in principle our collection of quantum computing patterns can be extended with patterns or primitives for other models in the future.

For the first decades, quantum computing was mostly driven by research. Many algorithms for quantum computers were published before any commercial hardware existed [6], [24]. Thus, these first algorithms were only of theoretical relevance until the first commercial quantum computer was released in 2016 [25]. And even though until today no quantum random access memory (QRAM) [26] exists, is a common prerequisite of many algorithms [27]–[29].

Quantum algorithms are often hybrid and, therefore, consist of classical and quantum calculations. For example, Shor's algorithm [24] uses a quantum subroutine in an intermediate step. In the following, we describe a typical structure of a quantum computation and then define the main components as primitives in the next section. Fig. 1 illustrates the computation as a quantum circuit. First, the qubits of the register are initialized as $|0\rangle$. Then, an initial state is prepared. This step also includes loading data which (as a result) is encoded in the state of the register. Afterward, quantum gates perform unitary transformations on the state of the register. Finally, one or multiple qubits are measured as the result of this computation indicated by measurement gates. The overall quantum circuit is characterized by its *depth* which is the number of sequential executable gates and its *width* that equals the number of required qubits [30]).

[1]https://www.ibm.com

[2]https://www.rigetti.com

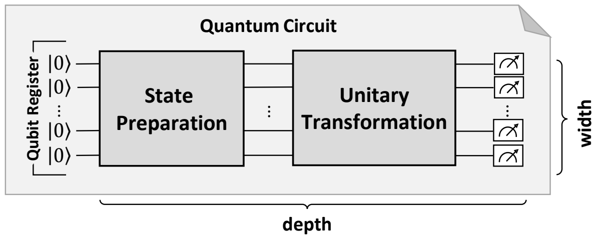

Fig. 1. Typical structure of a quantum calculation that is represented as a quantum circuit which has a defined depth and width.

C. *Pattern Primitives For Quantum Algorithms*

In this section, we present three pattern primitives for quantum computing. Each primitive has a *Name* and an *Icon* as a graphical representation. This is followed by a short *Description* and if applicable, a *Sketch*. As quantum computing is based on quantum mechanics, this is followed by a *Mathematical Definition*, and an *Example* that illustrates a concrete instance. Our primitives cover fundamental building blocks of quantum computing that serve as a basis for our data encoding patterns in Section III. Note that we do not claim that this is an exhaustive list of pattern primitives for the domain. We first introduce the most basic primitive (Qubit) and then present Qubit Register and Quantum Gate.

Qubit

***Description**:* A quantum bit (qubit) is the basic unit for information in quantum computing.

***Mathematical Definition**:* The state of a qubit is represented by a two-dimensional vector $|\psi\rangle$:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \ \text{ where } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1 \quad (1)$$

Here we used the Dirac notation for vectors where in particular

$$|0\rangle = \left(\begin{smallmatrix}1\\0\end{smallmatrix}\right) \text{ and } |1\rangle = \left(\begin{smallmatrix}0\\1\end{smallmatrix}\right).$$

The set $\{|0\rangle, |1\rangle\}$ is a basis of the two-dimensional vector space of the qubit and also referred to as the *computational basis*. The complex numbers α and β are the *amplitudes* of the quantum system and determine the possibility of a measurement outcome: with a probability of $|\alpha|^2$, a measurement in the computational basis results in $|0\rangle$ and with a probability of $|\beta|^2$, measuring results in the $|1\rangle$ state. As these are the only possible outcomes of a measurement, their possibilities ($|\alpha|^2$ and $|\beta|^2$) must sum up to 1. If $\alpha, \beta \neq 0$, the qubit is in a *superposition* - and therefore, in a linear combination - of $|0\rangle$ and $|1\rangle$.

A common visualization of the state of a qubit is the Bloch Sphere (see Fig. 2) which also inspired the icon of this primitive. Each possible state is mapped to a point on the surface of the Bloch Sphere.

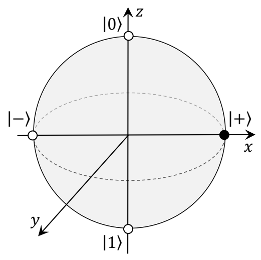

Fig. 2. Bloch Sphere Representation of a Qubit.

Example: For example, the so-called $|+\rangle$ state that is depicted in Fig. 2 is an equal superposition of $|0\rangle$ and $|1\rangle$:

$$|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$$

The probability for both measurement outcomes ($|0\rangle$ and $|1\rangle$) is equal and can be calculated as follows: $|\frac{1}{\sqrt{2}}|^2 = 0.5$.

Qubit Register

Description: Multiple qubits can form a qubit register whose state is represented by a vector in a high dimensional complex vector space.

Mathematical Definition: The state of an n-qubit register is defined as:

$$|\psi\rangle = \sum_{i=0}^{2^{n-1}} \alpha_i |i\rangle \text{ where } \sum_{i=0}^{2^{n-1}} |\alpha_i|^2 = 1. \tag{2}$$

In the equation above, $|i\rangle$ is used as a common notation for $|b_0 \dots b_n\rangle$ where $b_0 \dots b_n$ is the binary representation of i. If there are at least two amplitudes $a_i, a_j \neq 0, i \neq j$, the register is in superposition of all states with a non-zero amplitude. If each quantum system s is prepared in the state $|\psi_s\rangle$, the composite system can be described as tensor product [21]:

$$|\psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle \otimes \dots \otimes |\psi_n\rangle \tag{3}$$

Often, Eq. (3) is abbreviated to $|\psi_0\psi_1 \dots \psi_n\rangle$. If the state is not separable, i.e., it can not be expressed as a tensor product of its components, it is in an *entangled* state [21].

Example: An example of an entangled state is $|\phi^+\rangle$:

$$|\phi^+\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

Note that measuring only one of the qubits determines the measurement outcome of the other (both are measured as either $|0\rangle$ or $|1\rangle$). A state which is not entangled is for example

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle$$

as it can be rewritten as a product of the individual qubits (refer to [21] for an in-depth introduction on entanglement):

$$|\psi\rangle = |0\rangle \otimes (\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle).$$

Quantum Gate

Description: Analogue to classical logic gates (for example AND, OR, and NOT) that act on bits, quantum computers manipulate qubits with quantum gates. Often, a calculation is described as a quantum circuit (Fig. 3) that visualizes the sequence of quantum gates that are applied to each qubit. Quantum gates can act on either one or multiple qubits.

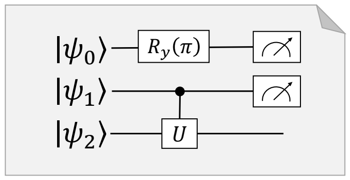

Fig. 3. Exemplary quantum circuit. Quantum gates are applied in temporal order from left to right. Quantum gates are defined for either one (for example R_y) or multiple qubits, e.g., the gate that is applied to the other two qubits.

Mathematical Definition: Except for measurement gates (i.e., the second gate on the first qubit in Fig. 3), a quantum gate is defined by a unitary matrix U for which by definition the inverse matrix $U^\dagger$ can be applied to undo the computation of U. By multiplying this matrix with a state vector, the resulting state vector can be obtained.

Example: An example of a one-qubit gate is R_y [10]:

$$R_y(2x) = \begin{pmatrix} \cos x & -\sin x \\ \sin x & \cos x \end{pmatrix}$$

This defines a rotation by an angle $2x$ around the y axis of the Bloch Sphere (refer to Fig. 2 for a visual representation). For example, rotating $|0\rangle$ around an angle of π in the Bloch Sphere can be expressed as follows:

$$R_y(\pi)|0\rangle = R_y(2\frac{\pi}{2})|0\rangle = \begin{pmatrix} \cos\frac{\pi}{2} & -\sin\frac{\pi}{2} \\ \sin\frac{\pi}{2} & \cos\frac{\pi}{2} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

which can be further simplified to:

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

and thus, results in a $|1\rangle$ state.

III. Patterns For Data Encoding

We start by describing our method for collecting patterns (Section III-A) and the pattern format (Section III-B). As both our new patterns are encoding patterns, we first describe their shared forces (Section III-C) which can also be found in our previous work [1]. We then give an overview of previous [1] and new patterns for data encoding in quantum computing (III-D) and present the new patterns (Angle and QRAM Encoding) in detail.

A. Method

Patterns are abstracted from existing solutions [15]. The patterns presented here and in previous publications [1], [31] were identified using the pattern authoring process of Fehling et al. [19] which we also describe in [1]. First, we analyzed scientific publications, books, and technical documentation to collect re-occurring solutions. If we found at least three occurrences (Coplien's rule [13]) of a pattern candidate, we authored a pattern. For loading data, a proven solution is a specific data-encoding used in various quantum algorithms. Note that given the current state of the art of quantum computing we do not require a concrete implementation. Instead, we focus on authoring patterns for writing and understanding quantum algorithms. However in future work, the patterns should be validated further in real applications.

B. Pattern Format

Depending on the domain, pattern authors use different formats for their patterns. Here, we use the pattern format of our previous work [1] that was based on the existing format of Fehling et al. [32]. Each pattern is introduced by a *Name* and an *Icon* that serves as a graphical representation of the pattern. Next to the icon, we denote the *Intent* that briefly summarizes the purpose of the pattern. If the pattern is also known under different names, these are listed as an *Alias*. Then, the problem and the circumstances of the pattern are described in the *Context* section before the *Forces* are presented. The forces are trade-offs or considerations that must be taken into account for solving the problem. The *Solution* itself is described in an abstract manner and often visualized by a *Solution Sketch*. Consequences of the solution are described as the *Result* of the solutions. This is followed by an optional section for *Variants* of the pattern. As patterns are often applied in combination or solve similar problems, we describe the connections between them in the *Related Patterns* section. Finally, we list *Known Uses* of the pattern. For encoding patterns, algorithms that use the encoding and state preparation routines are listed here. Additionally, concrete implementations of these algorithms can also be referred to in this section.

C. Forces of Data Encoding Patterns

Loading data is not a trivial task in quantum computing as a variety of data encodings can be used depending on the requirements of the proper unitary transform of the algorithm. Every data encoding is essentially a trade-off between three major forces:

(i) *the number of qubits needed for the encoding should be minimal* because current devices are of intermediate size and thus only contain a limited number of qubits
(ii) *the number of parallel operations needed to realize the encoding should be minimal to minimize the width of the quantum circuit* - ideally, the loading routine is of constant or logarithmic complexity
(iii) *the data must be represented in a suitable manner* for further calculations, e.g., arithmetic operations.

D. Overview of Data Encoding Patterns

Data encodings for quantum computing define how data is represented by the state of a quantum system. Table I gives an overview of previous [1] and new encoding patterns which are marked in bold. An excerpt of the patterns can also be found at *Quantum Computing Patterns*[3]. Each of these patterns further refines INITIALIZATION, a pattern of previous work [31] that describes the state preparation phase at the beginning of an algorithm (refer to Fig. 1). While the encodings of the first two patterns define how a single numerical data-point x_i is encoded, the three other patterns describe how a set X of n data-points can be represented. The representation of data in BASIS ENCODING is also part of two other encodings (QUAM and QRAM ENCODING). Therefore, we explain this pattern in more detail before we present the new encoding patterns (ANGLE and QRAM ENCODING) in detail.

For a BASIS ENCODING of a numerical data-point, its value is first approximated by its binary representation. The resulting bitstring $b_m \dots b_{-k}$ is then encoded by the $|b_m \dots b_{-k}\rangle$ state. Therefore, every bit of its bitstring is represented by a single qubit. Thus, BASIS ENCODING is not efficient in terms of the required number of qubits. In comparison, QUAM ENCODING uses superposition to encode a set of data-points in a qubit register of the same length (assuming that the binary representation of all values is equally long or padded with zeros). QRAM ENCODING needs $\lceil \log n \rceil$ additional qubits to represent the same data. Even more compact is the representation of data in AMPLITUDE ENCODING for which only $\lceil \log n \rceil$ qubits are needed. However, for an arbitrary data set the last three encodings of Table I cannot be realized efficiently; i.e. in constant or logarithmic number of parallel operations. While BASIS ENCODING and ANGLE ENCODING are not efficient in terms of required qubits, they can be realized in constant time (one single parallel operation). Further details and consequences of the encodings listed in Table I can be found in the corresponding patterns.

TABLE I
COMPARISON OF NEW AND PREVIOUS [1] DATA ENCODING PATTERNS.

Encoding Pattern	Encoding	Req. Qubits
BASIS ENCODING [1]	$x_i \approx \sum_{i=-k}^{m} b_i 2^i \mapsto \|b_m \dots b_{-k}\rangle$	$l = k+m$ per data-point
ANGLE ENCODING	$x_i \mapsto cos(x_i) \|0\rangle + sin(x_i) \|1\rangle$	1 per data-point
QUAM ENCODING [1]	$X \mapsto \sum_{i=0}^{n-1} \frac{1}{\sqrt{n}} \|x_i\rangle$	l
QRAM ENCODING	$X \mapsto \sum_{n=0}^{n-1} \frac{1}{\sqrt{n}} \|i\rangle \|x_i\rangle$	$\lceil \log n \rceil + l$
AMPLITUDE ENCODING [1]	$X \mapsto \sum_{i=0}^{n-1} x_i \|i\rangle$	$\lceil \log n \rceil$

[3]https://quantumcomputingpatterns.org/

Angle Encoding

Represent each data-point by a separate qubit

Alias: Qubit Encoding [33], (Tensor) Product Encoding [34]

Context: An algorithm requires an efficient encoding schema to be able to perform as many operations as possible within the decoherence time after the data has been loaded.

Solution: First, normalize all data-points that should be encoded to the interval $[0, 2\pi]$. Each value x_i is then represented by a single qubit as follows (Fig. 4): a rotation around the y-axis of the Bloch Sphere (refer to Fig. 2) is applied. Hereby, the angle for the rotation depends on the data value (see Section II-C for a more detailed description of the operation).

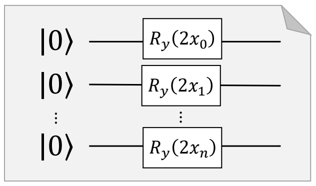

Fig. 4. Quantum circuit for loading data in Angle Encoding based on Leymann and Barzen [34].

Result: The resulting quantum state for this encoding is separable, i.e. the qubits are not entangled:

$$|\psi\rangle = \begin{pmatrix}\cos x_0\\ \sin x_0\end{pmatrix} \otimes \begin{pmatrix}\cos x_1\\ \sin x_1\end{pmatrix} \otimes \ldots \otimes \begin{pmatrix}\cos x_n\\ \sin x_n\end{pmatrix} \qquad (4)$$

The main advantage of this encoding is that it is very efficient in terms of operations: Only a constant number of parallel operations is needed regardless of how many data values need to be encoded [33]. However, the number of data values affects how many qubits are needed: One qubit is required to encode one component of the input vector. Thus, as only single-qubit rotations are required the state preparation routine is highly efficient while the number of qubits for this encoding is not optimal [35].

Related Patterns: This pattern refines Initialization.

Variants: [33] present *dense angle encoding* as an alternative encoding that exploits an additional property of qubits (relative phase) to use only $\frac{n}{2}$ qubits to encode n data points.

Known Uses: [33] and [35] present a classifier based on this encoding. The encoding is also used in quantum image processing: In the so-called flexible representation of quantum image (FRQI), one qubit represents the color information of a pixel whereas another register represents the position [36]. In the context of quantum neural networks, a qubit using this encoding has been referred to as a quantum neuron (quron) [37]. PennyLane provides a state preparation routine for angle encoding [38] for which the axis of the rotation can be specified (x, y, or z). If no loading routine is provided, a state preparation routine can be constructed with standard qubit rotations in a straightforward manner [34].

Quantum Random Access Memory (QRAM) Encoding

Use a quantum random access memory to access a superposition of data values at once

Context: An algorithm requires random access to the data values of the input.

Solution: A classical RAM that receives an address with a memory index, loads the data stored at this address into an output register. QRAM provides the same functionality, but the address and output register are quantum registers [9]. As a result, both the address and the output register can be in a superposition of multiple values. Fig. 5 illustrates the basic functionality of a QRAM [26] that receives a superposition of the first two addresses ($\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|01\rangle$) as input and loads the corresponding data values into an empty output register. This leads to the following state of the overall quantum system:

$$|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle|010\rangle + \frac{1}{\sqrt{2}}|01\rangle|110\rangle \qquad (5)$$

In general, loading m of n data values with a QRAM can therefore be described as the following operation [10]:

$$\frac{1}{\sqrt{m}}\sum_{i=0}^{m-1}|a\rangle_i|0\rangle \xrightarrow{QRAM} \frac{1}{\sqrt{m}}\sum_{i=0}^{m-1}|a\rangle_i|x_a\rangle \qquad (6)$$

where the first register is the address register that is in a superposition of all m addresses to be loaded and the second register is the output register. Further, $|a\rangle_i$ specifies the address of the i-th data value to be loaded and x_a is the data value associated with this address. The QRAM loads each data value $|x_a\rangle$ into the output register such that $|a\rangle_i|x_a\rangle$ is contained in the combined state of both registers. Depending on the data values, this state may be entangled.

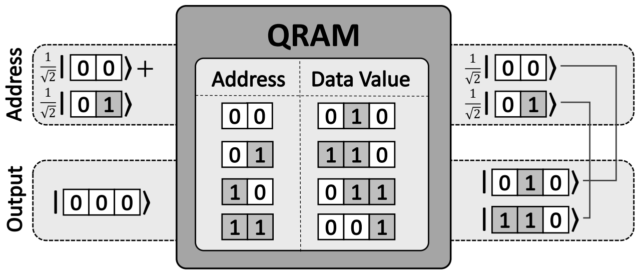
QRAM
Address
Data Value
Address
Output
$\frac{1}{\sqrt{2}}|00\rangle +$
$\frac{1}{\sqrt{2}}|01\rangle$
$|000\rangle$
00 010
01 110
10 011
11 001
$\frac{1}{\sqrt{2}}|00\rangle$
$\frac{1}{\sqrt{2}}|01\rangle$
$|010\rangle$
$|110\rangle$

Fig. 5. Basic functionality of a QRAM based on [9]. Given an address register that is in a superposition of addresses ($|00\rangle$ and $|01\rangle$), QRAM creates a superposition of addresses and their data values: $\frac{1}{\sqrt{2}}|00\rangle|010\rangle + \frac{1}{\sqrt{2}}|01\rangle|110\rangle$.

Result: For this encoding, l qubits are needed to encode the data values using Basis Encoding. The address register requires $\lceil log(n)\rceil$ additional qubits for a maximum of n addresses. The computational properties are similar to those of Basis and QuAM encoding: As a superposition of the encoded data values is prepared, data can be processed in parallel (quantum parallelism) and arithmetic operations such as addition or multiplication can be used.

An algorithm that uses QRAM, assumes that an efficient procedure exists that can be used to perform state preparation in logarithmic runtime [10]. As long as there are no hardware implementations of QRAM, a state preparation routine must be used that mirrors the loading process of a QRAM. The main drawback of QRAM encoding is that in general, no state preparation routine for arbitrary input data exists that is as efficient as a QRAM. A promised speed-up of an algorithm that relies on QRAM can only be realized if data can be encoded efficiently by a known state preparation routine.

Related Pattern: This pattern further refines INITIALIZATION and uses BASIS ENCODING. As the address and output register of QRAM ENCODING are often entangled, CREATING ENTANGLEMENT [31] is used.

Known Uses: An algorithm for state preparation is given by circuit family #3 of [39]. An alternative approach is presented in [40]. This encoding is used by algorithms for solving semi-definite programs [41]. Various other algorithms exist that require or are improved upon QRAM [26]–[29]. The so-called HHL algorithm for solving linear equations [42] uses QRAM ENCODING to represent eigenvalues in an intermediate step [41].

IV. RELATED WORK

The encoding patterns presented in this work complement encoding patterns [1] and patterns for quantum algorithms [31] of our previous works. While various other publications [43]–[45] introduce "quantum patterns", none of these confirm to patterns in the sense of Alexander et al. [15]. Therefore, to our best knowledge, no other pattern for quantum computing have been published so far.

Besides the encodings mentioned in this paper, various other encodings have been used in quantum algorithms. An overview of data encodings with focus on quantum machine learning can be found in [10], [33], [35], while [36] describes encodings for quantum image processing. In [33], the authors show that using ANGLE ENCODING as input data for a quantum classifier restricts the decision boundaries that the model can learn to a simple sine-function. [46] further generalizes these findings and show that the expressive power of a model can be increased by repeating the encoding. [34] draws general conclusions for data loading and various encodings in the NISQ era. In contrast to our approach, none of the previously mentioned overviews uses patterns to provide easy access to knowledge about data encodings.

In the context of quantum machine learning, the process of encoding data is also referred to as applying a quantum feature map ϕ [47], [48]. The authors point out that quantum feature maps and kernels in machine learning are related: Each feature map implicitly defines a quantum kernel. After applying the quantum feature map (and thus, loading the data), the data can be analyzed in high-dimensional space. Therefore, each encoding that we presented in this paper gives rise to a quantum feature map ϕ and defines a quantum kernel.

QRAM is an essential component in larger quantum computers [26]. Several architectures have been proposed and demonstrated on a small scale (see for example [49]). However, building a larger QRAM remains an open technical challenge for hardware providers [50], [51] which we also emphasize in our QRAM ENCODING pattern.

As current NISQ devices are limited by their hardware, various tools estimate or determine the resources required by a quantum algorithm (in particular, qubits and operations). Microsoft introduced the *Quantum Resources Estimator* [4] for implementations in their quantum programming language Q#. The *NISQ Analyzer* of Salm et al. [52] considers concrete input data of quantum algorithms and suggests suitable implementations and quantum computers for execution. The knowledge about data encodings captured in our patterns can be used to further improve these estimations.

V. CONCLUSION

In this paper, we extended our previous collection of patterns [1], [31] by two new data encoding patterns. In our overview of all existing encoding patterns, we compared them regarding their forces (in particular in terms of required qubits). Especially in the NISQ area, a specific encoding may simply be preferred because it is very efficient in terms of qubits (e.g., AMPLITUDE ENCODING) or operations for state preparation (e.g., ANGLE ENCODING). Nevertheless, understanding the limitations of QRAM ENCODING is crucial for many quantum algorithms. Besides the encoding patterns, we also introduced three pattern primitives that serve as fundamental building blocks for our patterns.

In the future, we will collect more quantum computing patterns and share them in our pattern repository [53], [54] as part of the PlanQK[5] project. We also plan to add concrete implementations to the patterns for further validations. The encoding patterns will further contribute to improve the resource estimation for quantum algorithms [52] in the future.

ACKNOWLEDGMENT

This work is partially funded by the BMWi project PlanQK (01MK20005N).

REFERENCES

[1] M. Weigold, J. Barzen, M. Salm, and F. Leymann, "Data encoding patterns for quantum computing," in *Proceedings of the 27th Conference on Pattern Languages of Programs*. The Hillside Group, 2021, in press.

[2] National Academies of Sciences, Engineering and Medicine, *Quantum Computing: Progress and Prospects*. Washington, DC: The National Academies Press, 2019.

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

[4] R. Horodecki, P. Horodecki, M. Horodecki, and K. Horodecki, "Quantum entanglement," *Reviews of modern physics*, vol. 81, no. 2, p. 865, 2009.

[5] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM review*, vol. 41, no. 2, pp. 303–332, 1999.

[6] L. K. Grover, "A fast quantum mechanical algorithm for database search," *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, 1996.

[4] https://docs.microsoft.com/th-th/quantum/user-guide/machines/resources-estimator

[5] http://planqk.de/

[7] "Ibm's roadmap for scaling quantum technology," https://www.ibm.com/blogs/research/2020/09/ibm-quantum-roadmap/, 2020.

[8] "Scaling ionq's quantum computers: The roadmap," https://ionq.com/posts/december-09-2020-scaling-quantum-computer-roadmap, 2020.

[9] E. R. Johnston, N. Harrigan, and M. Gimeno-Segovia, *Programming Quantum Computers: Essential Algorithms and Code Samples*. O'Reilly Media, Incorporated, 2019.

[10] M. Schuld and F. Petruccione, *Supervised Learning with Quantum Computers*, ser. Quantum Science and Technology. Springer International Publishing, 2018.

[11] S. Aaronson, "Read the fine print," *Nature Physics*, vol. 11, pp. 291–293, 2015.

[12] B. Weder, J. Barzen, F. Leymann, M. Salm, and D. Vietz, "The Quantum Software Lifecycle," in *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software (APEQS 2020)*. ACM, Nov. 2020, Workshop, pp. 2–9. [Online]. Available: https://dl.acm.org/doi/10.1145/3412451.3428497

[13] J. O. Coplien, *Software Patterns*. SIGS Books & Multimedia, 1996.

[14] U. Zdun and P. Avgeriou, "Modeling Architectural Patterns Using Architectural Primitives," in *Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA 2005)*. ACM, Oct. 2005, pp. 133–146.

[15] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, Aug. 1977.

[16] U. Zdun, C. Hentrich, and S. Dustdar, "Modeling process-driven and service-oriented architectures using patterns and pattern primitives," *ACM Trans. Web*, vol. 1, no. 3, p. 14–es, Sep. 2007.

[17] C. Endres, U. Breitenbücher, M. Falkenthal, O. Kopp, F. Leymann, and J. Wettinger, "Declarative vs. Imperative: Two Modeling Patterns for the Automated Deployment of Applications," in *Proceedings of the 9th International Conference on Pervasive Patterns and Applications (PATTERNS 2017)*. Xpert Publishing Services, Feb. 2017, pp. 22–27.

[18] D. Schumm, J. Barzen, F. Leymann, and L. Ellrich, "A Pattern Language for Costumes in Films," in *Proceedings of the 17th European Conference on Pattern Languages of Programs (EuroPLoP 2012)*, C. Kohls and A. Fiesser, Eds. New York NY USA: ACM, 2012.

[19] C. Fehling, J. Barzen, U. Breitenbücher, and F. Leymann, "A Process for Pattern Identification, Authoring, and Application," in *Proceedings of the 19th European Conference on Pattern Languages of Programs (EuroPLoP 2014)*. ACM, Jan. 2014.

[20] D. Aharonov, W. Van Dam, J. Kempe, Z. Landau, S. Lloyd, and O. Regev, "Adiabatic quantum computation is equivalent to standard quantum computation," *SIAM review*, vol. 50, no. 4, pp. 755–787, 2008.

[21] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge and New York: Cambridge University Press, 2010.

[22] R. Raussendorf and H. J. Briegel, "A one-way quantum computer," *Physical Review Letters*, vol. 86, no. 22, p. 5188, 2001.

[23] R. LaRose, "Overview and Comparison of Gate Level Quantum Software Platforms," *Quantum*, vol. 3, p. 130, Mar. 2019.

[24] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th annual symposium on foundations of computer science*. Ieee, 1994, pp. 124–134.

[25] "IBM makes quantum computing available on ibm cloud to accelerate innovation," https://www-03.ibm.com/press/us/en/pressrelease/49661.wss, 2016.

[26] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Physical Review Letters*, vol. 100, no. 16, p. 15, 2008.

[27] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical review letters*, vol. 113, no. 13, p. 130503, 2014.

[28] N. Wiebe, A. Kapoor, and K. M. Svore, "Quantum deep learning," *arXiv:1412.3489*, 2015.

[29] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nature Physics*, vol. 10, no. 9, pp. 631–633, 2014.

[30] J. L. F. W. B. Salm, Marie; Barzen, "About a Criterion of Successfully Executing a Circuit in the NISQ Era: What $wd \ll 1/\epsilon_{\text{eff}}$ Really Means," in *Proceedings of the 1st ACM SIGSOFT International Workshop on Architectures and Paradigms for Engineering Quantum Software (APEQS 2020)*. ACM, Nov. 2020, Workshop, pp. 10–13. [Online]. Available: https://dl.acm.org/doi/10.1145/3412451.3428498

[31] F. Leymann, "Towards a pattern language for quantum algorithms," in *Quantum Technology and Optimization Problems*, ser. Lecture Notes in Computer Science (LNCS), vol. 11413. Cham: Springer International Publishing, 2019, pp. 218–230.

[32] C. Fehling, F. Leymann, R. Retter, W. Schupeck, and P. Arbitter, *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*. Springer, Jan. 2014.

[33] R. LaRose and B. Coyle, "Robust data encodings for quantum classifiers," *arXiv:2003.01695*, 2020.

[34] F. Leymann and J. Barzen, "The bitter truth about gate-based quantum algorithms in the NISQ era," *Quantum Science and Technology*, pp. 1–28, Sep. 2020. [Online]. Available: https://doi.org/10.1088/2058-9565/abae7d

[35] E. Grant, M. Benedetti, S. Cao, A. Hallam, J. Lockhart, V. Stojevic, A. G. Green, and S. Severini, "Hierarchical quantum classifiers," *npj Quantum Information*, vol. 4, no. 1, pp. 1–8, 2018.

[36] F. Yan, A. M. Iliyasu, and S. E. Venegas-Andraca, "A survey of quantum image representations," *Quantum Information Processing*, vol. 15, no. 1, pp. 1–35, 2016.

[37] M. Schuld, I. Sinayskiy, and F. Petruccione, "The quest for a quantum neural network," *Quantum Information Processing*, vol. 13, no. 11, pp. 2567–2586, 2014.

[38] "Templates," https://pennylane.readthedocs.io/en/stable/introduction/templates.html, 2020.

[39] J. A. Cortese and T. M. Braje, "Loading classical data into a quantum computer," *arXiv:1803.01958*, 2018.

[40] A. Prakash, "Quantum algorithms for linear algebra and machine learning." Ph.D. dissertation, EECS Department, University of California, Berkeley, Dec 2014.

[41] K. Mitarai, M. Kitagawa, and K. Fujii, "Quantum analog-digital conversion," *Physical Review A*, vol. 99, 01 2019.

[42] 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.

[43] A. Gilliam, C. Venci, S. Muralidharan, V. Dorum, E. May, R. Narasimhan, and C. Gonciulea, "Foundational patterns for efficient quantum computing," *arXiv:1907.11513*, 2019.

[44] Y. Huang and M. Martonosi, "Statistical assertions for validating patterns and finding bugs in quantum programs," in *Proceedings of the 46th International Symposium on Computer Architecture*, ser. ISCA '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 541–553.

[45] S. Perdrix, "Quantum patterns and types for entanglement and separability," *Electron. Notes Theor. Comput. Sci.*, vol. 170, p. 125–138, Mar. 2007.

[46] M. Schuld, R. Sweke, and J. J. Meyer, "The effect of data encoding on the expressive power of variational quantum machine learning models," *arXiv:2008.08605*, 2020.

[47] M. Schuld and N. Killoran, "Quantum machine learning in feature hilbert spaces," *Physical review letters*, vol. 122, no. 4, p. 040504, 2019.

[48] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, "Supervised learning with quantum-enhanced feature spaces," *Nature*, vol. 567, no. 7747, pp. 209–212, 2019.

[49] M. Blencowe, "Quantum ram," *Nature*, vol. 468, no. 7320, pp. 44–45, 2010.

[50] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, 2017.

[51] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig, "Quantum machine learning: a classical perspective," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 474, no. 2209, p. 20170551, 2018.

[52] M. Salm, J. Barzen, U. Breitenbücher, F. Leymann, B. Weder, and K. Wild, "The NISQ Analyzer: Automating the Selection of Quantum Computers for Quantum Algorithms," in *Proceedings of the 14th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2020)*. Springer International Publishing, Dec. 2020, pp. 66–85. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-64846-6_5

[53] F. Leymann and J. Barzen, "Pattern Atlas," *arXiv:2006.05120 [cs]*, 2020.

[54] M. Weigold, J. Barzen, U. Breitenbücher, M. Falkenthal, F. Leymann, and K. Wild, "Pattern Views: Concept and Tooling of Interconnected Pattern Languages," in *Proceedings of the 14th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2020)*. Springer International Publishing, Dec. 2020, pp. 86–103. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-64846-6_6