University of Naples Federico II
University of Camerino
National Research Council of Italy


Doctor of Philosophy in Quantum Technologies
35th cycle

Architectures and circuits for distributed quantum computing

Daniele Cuomo

Advisors
Prof. Marcello Caleffi
Prof. Angela Sara Cacciapuoti

Coordinator
Prof. Francesco Tafuri

Academic Discipline: ing-inf/03
Years 2019 – 2023

Abstract

This thesis treats networks providing *quantum computation* based on *distributed paradigms*. Compared to architectures relying on one processor, a network promises to be more *scalable* and less *fault-prone*. Developing a distributed system able to provide practical quantum computation comes with many challenges, each of which need to be faced with careful analysis in order to create a massive integration of several components properly engineered.

In accordance with hardware technologies, currently under construction around the globe, *telegates* represent the fundamental inter-processor operations. Each telegate consists of several tasks: i) entanglement generation and distribution, ii) local operations, and iii) classical communications. Entanglement generation and distribution is an expensive resource, as it is time-consuming.

The main contribution of this thesis is on the definition of *compilers* that minimize the impact of telegates on the overall *fidelity*. Specifically, we give rigorous formulations of the subject problem, allowing us to identify the inter-dependence between computation and communication. With the support of some of the best tools for reasoning – i.e. network optimization, circuit manipulation, group theory and `ZX`-calculus – we found new perspectives on the way a distributed quantum computing system should evolve.

Contents

Chapter 1

Introduction

Distributed quantum computing is one of the most appealing applications in the panorama of quantum technologies. In fact, distributed architectures could be our bridge to step beyond the current NISQ era [1, 2, 3, 4, 5, 6]. This explains the wide interest for a large-scale integration of quantum technologies. By inter-connecting spatially distributed quantum processors, we would achieve a scalable architecture resistant to noise. The general trend [4, 7, 8, 9, 10, 11] shows a common belief in distributed (and quasi-distributed, or multi-core) architectures as physical substrate, allowing a modular and horizontal scale-up of computing resources, rather than relying on vertical scale-up, coming from single hardware advancements. On the flip side, by linking distributed quantum processors, several new challenges arise [12, 13, 14, 15, 16, 2].

Being up-to-date with technologies currently under construction around the globe is a mandatory step to create a realistic system. We hence begin – in Sec. 1.1 – by reviewing some of the most promising hardware technologies. This provide us (and the reader) with a realistic perspective of the fundamental components belonging a distributed quantum computing system. Once done that we propose – in Sec. 1.2 – a full-stack development meant to be modular and prone to future changes. The stack is indeed already an extension of one of our first proposal, available in [2].

In Ch. 2, we abstract from hardware technologies, by introducing the reader to the fundamental tools concerning distributed computing paradigms. We use the most affirmed language to express quantum computation, i.e. the standard quantum circuit model. We start with some basics on unitary *synthesis* and *decomposition*, important also for local computational. We then extend the subject to work on distributed architectures.

For the sake of completeness, In Ch. 3 we provide a framework concerning quantum noise. Handling noise is probably, to date, the hardest challenge we are facing, as any model struggle to be scalable for hardware. Hence the aim of this chapter is to give a perspective of the magnitude of the problem, which – sooner or later – will be part of a distributed computing system. The framework is strengthen by experimental results.

We conclude the thesis with what is our main contribution to research. Thanks to the knowledge gathered by us throughout the aforementioned chapters, we could minutely investigate one essential component for any system implementation of practical value. Such a component is commonly referred as *compiler*, which we formulate with mathematical rigor in Ch. 4. With the support of some of the best tools for reasoning – i.e. network optimization, circuit manipulation, group theory and `ZX`-calculus – we managed to give a first complete model for the compilation problem on distributed architectures. Every feature characterizing the problem is treated minutely. We could separate the problem in several parts, each of which is tackled with a dedicated optimizer.

We also use our model as benchmarker for different network topologies. It is indeed important, in the development of a practical system, that the project follows a *co-design* line, where each component is designed to fit at its best with the other components.

1.1 Technologies for distributed quantum computing

1.1.1 Stationary-flying transduction

Qubit-qubit interaction generally works by means of some *transducer*. A transducer can be seen as a physical interface "converting quantum signals from one form of energy to another" [17]. It is especially true, in a distributed setting, that a transducer is able to move an information stored into some *stationary qubit* – e.g. a trapped-ion, a transmon or a quantum dot – into some flying object, usually photons. A photon is therefore an information carrier or medium, able to cover a long distance. Therefore, the medium can be used to make distant qubits interact.

The ability to engineer efficient transducers allows us to rethink at quantum architectures as to be scalable and modular. Depending on the transducer [18, 17, 19, 20, 21, 10, 22, 23, 24, 9], different kinds of distributed architecture arise. For the sake of understanding qubit-qubit interaction in a distributed setting, we now consider distributed ion-traps architectures.

Scaling up a single ion-trap is challenging [25]. On the other hand, they represent a promising technology for integration within a distributed architecture, as a result of high gate fidelity [26, 27] and long life-time [28, 29]. In what comes next, we consider a cavity-based integration.

Cavities and photon emission

Considering the scenario of qubits stored on different processors, to couple them, the physical setting needs to *scatter* quantum information outside a processor and reach the other one. This can be done by means of a single photon, canalized within an optical fiber.

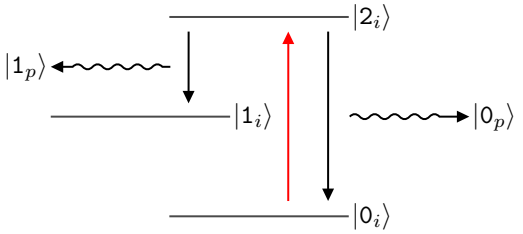


Figure 1.1: Simplified energy level structure of an ion (i) and relative photon emissions (p).

In order to achieve such a configuration, we here consider ions able to be modeled as a three-levels system – see Fig. 1.1. Such a system depicts the experimental set-up proposed in [30][1]. The specifics of the system comes from the *ion species* selected to encode quantum information [33]. By placing such an ion within a *cavity*, this creates an ion-cavity system, where now the ion interact with the cavity mode. The cavity has the role of collecting and scatter outside the system the photon emitted by the ion. Fig. 1.2 shows a pictorial representation of the ion-cavity able to do so.

The first step taking place is the excitation of the ion $|0_i\rangle \to |2_i\rangle$ – i.e. the red arrow in Fig. 1.1. Ideally, a spontaneous decay of the ion brings its energy with equal probabilities to one of the two lowest (and computationally relevant) states – i.e. $|0\rangle_i$ and $|1_i\rangle$. Furthermore, this happens with the emission of a photon which is coherent with the state of the ion.

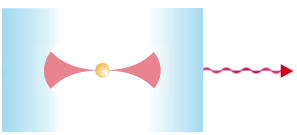


Figure 1.2: Exemplary representation of an ion-cavity system emitting a photon. For technical details the reader may start from [34].

As we anticipated, the scattered photon can be canalized within an optical fiber; the final configuration of the ion-fiber system is in the

[1]The interested reader can find other settings at Refs. [31, 32].

superposition $^1/\sqrt{2}(|0_i 0_p\rangle + |1_i 1_p\rangle)$.

A pictorial representation of a single node of the distributed architecture is shown in Fig. 1.3. The cavity is pointing at one of the ions, which is coloured differently as an ion-trap may be composed by different ion species[2], depending on whether it is meant to perform computation or communication.

To achieve the *non-local coupling* we need to consider two ion-traps generating and distributing the entanglement (at the same time), after which, a protocol called *entanglement swapping* completes the process. A *control system* will take care of accomplishing the task.

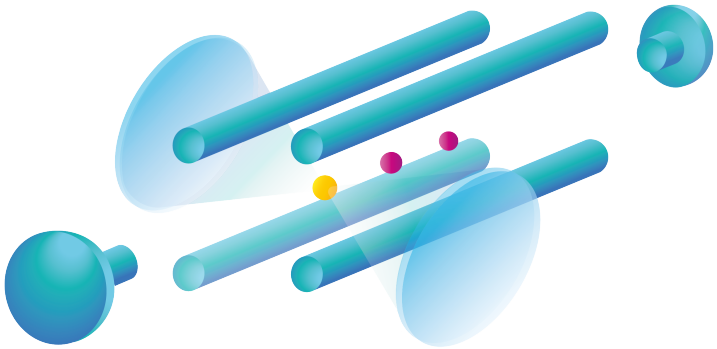


Figure 1.3: An ion-trap embedded with a cavity pointing at a single ion. Representation inspired by a linear design [33, 37].

1.1.2 Control system

Summing up, to establish entanglement, the ions are simultaneously excited to an electronic state that spontaneously decays, during which a single photon each is emitted whose polarization is entangled with the ion's internal state. These photons are collected into optical fibres using free-space optics and sent to a common *terminal*.

The terminal take care of detecting the photons by means of a probabilistic *Bell state measurement*. This projects the ions into a maximally entangled state, heralded by the coincident detection of the pair of photons – see Sec. 1.1.3 for details. This is commonly referred as *entanglement swapping*.

It is important that the ion-traps are synchronized, so that the photon reach the terminal at the same time[3]. Classical synchronization protocol would take care of this by means of a master-clock. An experimental settings is available in Ref. [39].

Ultimately, to achieve scalability we need to consider the case of several processors. In the most basic scenario, all the processors are *centralized*, in the sense the all of them are wired to a common terminal to perform bell state measurement. Such a setting is the first example of scalable distributed architectures. The problems arising from such a setting is a *scheduling* problem. A multiplexer taking deterministic choices would be enough to ensure that all the processors are carefully scheduled to not create overlaps. An experimental settings, where four processors are scheduled by means of a multiplexer, is available in Ref. [40].

[2]Which brings to the classification in *communication* and *computation* (or data) qubits [35, 5, 36, 2].

[3]Without synchronization, one can retrieve the likelihood of each photon source. This makes the photons distinguishable, causing a loss in fidelity [38].

1.1.3 Bell state analyser

To keep the discussion easy we explained the main procedures by means of three-levels systems for the ions. However, each of this state should be split to create several possible configuration. This doesn't change the whole protocol, but it has consequences on the possible outcomes obtained by measuring the photons. Different ion species and kind of measurement lead to different configurations. In general, we distinguish three main outcomes:

- One or both photons failed to be detected by the measurement. This means that the whole procedure is basically wasted time, as the nodes need to attempt again from scratch. This possibility can be a serious problem to practical computation, as a low *success rate* leads to long waiting times.
- The protocol succeeded and the ions are in one of the four Bell states $\{|\Phi^+\rangle, |\Phi^-\rangle, |\Psi^+\rangle, |\Psi^-\rangle\}$.
- The protocol partially succeeded. Namely, a superposition between two bell states has been created, e.g. $|\Phi^\pm\rangle$ or $|\Psi^\pm\rangle$. This scenario may occur for several reasons – depending also from the employed physical settings – it can be caused by *dark measurements* [38], a rare and negligible scenario. Otherwise, it may come, for example, from two *clicks* coming from the same detector.

Different Bell measurement settings brings to different sets of heralded entanglements [11, 42, 41, 43]. We here consider the quite general case of Ref. [41], as we think this case may be particularly efficient for distributed computation. A pictorial representation of the setting is reported in Fig. 1.4. In fact such a configuration brings to a 50% chance of success – i.e. two clicks on different detectors – and 50% of partial success – i.e. two clicks on the same detector. Namely, when the protocol succeeds, the final state is in $\{|\Psi^-\rangle, |\Psi^+\rangle\}$, while in case of partial success, the output has an ambiguous phase: $|\Phi^\pm\rangle$.

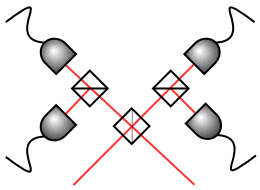


Figure 1.4: A common setting for the Bell state measurement [41].

The ideal result coming from performing entanglement generation and distribution followed by entanglement swapping is a maximally entangled state between distant qubits. In practice, this is not achievable as each of the complicated techniques we described are in general not perfect, resulting in a state slightly different from a Bell pair. One can evaluate the final distributed state in terms of *fidelity* with some target Bell state. E.g.,

$$\mathfrak{f} = \langle\Phi^+|\,\sigma\,|\Phi^+\rangle\,. \tag{1.1}$$

Where σ is the generated state. For example, consider the experiments reported in Ref. [41, 11]. The author's proposal starts with the generation of a non-maximally entangled state ion-photon

$$\sqrt{\frac{2}{3}}\,|0_i 0_p\rangle + \sqrt{\frac{1}{3}}\,|1_i 1_p\rangle\,. \tag{1.2}$$

However, after the collection into the single mode fiber, the state gets projected to the Bell pair

$$\frac{1}{\sqrt{2}}(|0_i 0_p\rangle + |1_i 1_p\rangle). \tag{1.3}$$

Once performed the entanglement swap, the ion-ion average fidelity is 0.94; a promising result. Unfortunately, in the perspective of practical computation, the fidelity needs to be some $\mathfrak{f} = 1 - \varepsilon$ with ε small enough to keep the error rate *manageable*, e.g. by means of *error correction schemes* – treated in Ch. 3. A possible solution is called *entanglement distillation* [44, 45, 46] (or purification). However, choosing the best approach is not trivial and may very depend on the architecture specifics.

1.2 Envisioning the full system

We reported several important research fields deeply related to the implementation of a first distributed and scalable architecture. We introduced the required technologies to achieve qubit-qubit interaction when these are arbitrarily far apart.

The considered literature gives a perspective on how to integrate multiple quantum processors into a scalable architecture, able to perform distributed quantum computation.

Stemming from the above overview, we now need to extract a full-stack development, by identifying the most important roles – and dependencies – and we will need to engineer an *ecosystem* [2], providing a framework for distributed quantum computation. As we are facing the early stage of quantum computation and distributed architecture, it is wise to focus on the main challenges and assigning them to a proper *entity*[4]. In fact, any proposal now is highly prone to changes, because of the continuous growing of the field [3] and the huge advancing in both technology and information theory.

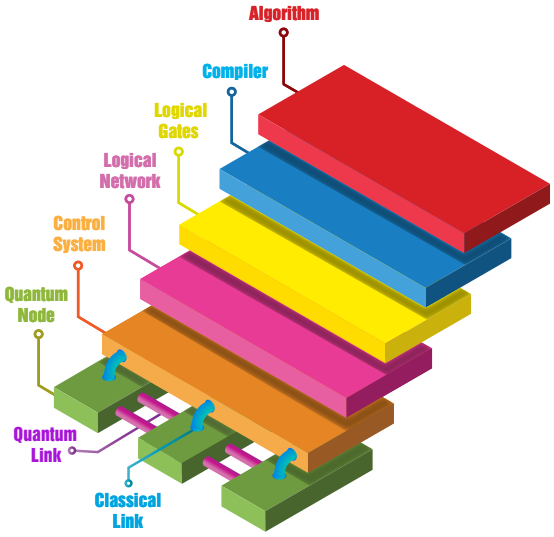


Figure 1.5: Full-stack development of a distributed quantum computing framework.

We propose a design that starts from a bottom-up reasoning, stacking up a number of layers where the lower ones provide some *resources* and flexibility which the upper layers can rely on. More precisely, consider Fig. 1.5: this shows a linear stack where each layer represents one of the fundamental subjects necessary to create a practical framework.

Level 1

Not surprisingly, the first layer is a mere pictorial representation of the distributed hardware. For the sake of clarity, we depicted a network composed by three quantum nodes. We already discussed how such a network may be achieved by focusing on ion-traps integrated with cavities, Bell state analysers and multiplexers – see Secs. 1.1 and 1.1.3.

[4] As usual in the engineer terminology, an entity is something quite abstract, which needs to be defined in terms of its roles and relations with other entities.

A quantum node refers to the full hardware set-up working locally, which includes, of course, the quantum processor. The nodes are inter-connected by **quantum links** wherein mediums carries quantum information. Such a set-up can be achieved, for example, by means of [40]:

- ion-traps, cavity-based transducers as quantum nodes and
- optical fibers, multiplexers and bell-state analysers as quantum links.

Level 2

As explained in Sec. 1.1.2, as minimum requirement for the system to be *operative*, the network needs to be carefully handled by a classical **control system**, which cares about synchronization and scheduling in real time. Because of its role, the control system is mainly physical. In fact, it is directly connected to each quantum node by means of **classical links**. The linkage will be also used to gather classical information coming from *measurement-based computation* [47].

Advancements in the physical network[5] will allow to *evolve* the layer, up to becoming a *quantum control system* – as envisioned for example in Refs. [51, 52, 53, 54] –, a quantum control system will be able to optimize the efficiency of the network, having access to a wider spectrum of resources – e.g. high-dimensional entangled states – which can be manipulated to optimize the network efficiency, by means of communication protocol fundamentally based on quantum communication theory. A quantum control system may also be responsible for the definition of a (distributed) error correction scheme – treated in Ch. 3.

Level 3

The control system deals with *real time tasks* and, if engineered properly, it is able to guarantee a **logical network** with a time domain that can be *discrete*. Specifically, a control system provides a *topology* running in well-defined *time slots*. This means that any kind of unexpected delay is negligible to the upper-layers. As central part of the stack, generating a logical network is a critical step. In fact, it is the layer where real-time tasks meet logical computation. Hence, a logical network should provide an array of logical resources which are meaningful to computation.

Level 4

As said above, the logical network provides an array of logical resources. Because of their importance a layer is dedicated to represent such resources. The set of **logical gates** will be the fundamental components upon which building a *computational paradigm*, specifically designed to work on a server farm.

Logical gates can be local – e.g. multi-qubit gates [55, 56, 57, 58] – and non-local – i.e. telegates –. In fact, while local gates operates on logical nodes, telegates operates throughout the logical network. This makes the latter resource much more expensive, which deserve a dedicated analysis, especially considering the degree of novelty in the context of computational paradigms that may arise. Refer to Sec. 2.2 for details.

[5] Up to the development of a *quantum internet* [12, 15, 48, 5, 2, 49, 14, 50]

Level 5

We entitled this layer **compiler** as its role is highly related to the already well-established branch of research in the context of local quantum computation [59, 66, 67, 68, 69, 70, 71, 72, 73, 60, 61, 62, 63, 64, 65]. A compiler has the crucial role to mask all the underlying stack to an *algorithm designer*. In fact an algorithm is generally written to solve some problem which goes beyond the architecture meant to process it. This means that the designer works in an agnostic fashion, without considering the constraints coming from the stack. Which is why a compiler is the final necessary optimizer, able to transform an abstract algorithm into a logical algorithm, compliant with the logical resources given by the bottom layers. Our research project mainly focus on the *standardization* of such a layer and we detailed our efforts in Ch. 4.

Level 6

The upper layer is simply an **algorithm**, an abstract input, which the framework takes charge of and carefully spread throughout the whole stack in order to be processed.

Once the framework is ready-to-go, it will be able to accept some *groups of algorithms*[6], up to *universal* groups. Refer to Ch. 2 for details.

References

[1] J. Preskill. "Quantum Computing in the NISQ era and beyond." In: *Quantum* 2.79 (2018) (cit. on p. 2).

[2] Daniele Cuomo, Marcello Caleffi, and Angela Sara Cacciapuoti. "Towards a distributed quantum computing ecosystem". In: *IET Quantum Communication* 1.1 (2020), pp. 3–8 (cit. on pp. 2, 4, 6, 7).

[3] Elizabeth Gibney. "The quantum gold rush". In: *Nature* 574.7776 (2019), pp. 22–24 (cit. on pp. 2, 6).

[4] Jay Gambetta. *Expanding the IBM Quantum roadmap to anticipate the future of quantum-centric supercomputing*. 2022 (cit. on p. 2).

[5] Angela Sara Cacciapuoti et al. "Quantum internet: networking challenges in distributed quantum computing". In: *IEEE Network* 34.1 (2019), pp. 137–143 (cit. on pp. 2, 4, 7).

[6] Rodney Van Meter and Simon J Devitt. "The path to scalable distributed quantum computing". In: *Computer* 49.9 (2016), pp. 31–42 (cit. on p. 2).

[7] Andrew Eddins et al. "Doubling the size of quantum simulators by entanglement forging". In: *PRX Quantum* 3.1 (2022), p. 010309 (cit. on p. 2).

[8] Yehan Liu et al. "Design of interacting superconducting quantum circuits with quasi-lumped models". In: *American Physical Society (March Meeting)*. 2022 (cit. on p. 2).

[9] Alysson Gold et al. "Entanglement across separate silicon dies in a modular superconducting qubit device". In: *npj Quantum Information* 7.1 (2021), pp. 1–10 (cit. on pp. 2, 3).

[6] Parallel-based algorithms – e.g. see Ref. [74] for a parallel algorithm solving the quantum Fourier transform – are natively meant to work on distributed architectures.

[10] David Kielpinski, Chris Monroe, and David J Wineland. "Architecture for a large-scale ion-trap quantum computer". In: *Nature* 417.6890 (2002), pp. 709–711 (cit. on pp. 2, 3).

[11] LJ Stephenson et al. "High-rate, high-fidelity entanglement of qubits across an elementary quantum network". In: *Physical review letters* 124.11 (2020), p. 110501 (cit. on pp. 2, 5).

[12] H Jeff Kimble. "The quantum internet". In: *Nature* 453.7198 (2008), pp. 1023–1030 (cit. on pp. 2, 7).

[13] Stefano Pirandola and Samuel L Braunstein. "Physics: Unite to build a quantum Internet". In: *Nature News* 532.7598 (2016), p. 169 (cit. on p. 2).

[14] Wolfgang Dür, Raphael Lamprecht, and Stefan Heusler. "Towards a quantum internet". In: *European Journal of Physics* 38.4 (2017), p. 043001 (cit. on pp. 2, 7).

[15] Stephanie Wehner, David Elkouss, and Ronald Hanson. "Quantum internet: A vision for the road ahead". In: *Science* 362.6412 (2018) (cit. on pp. 2, 7).

[16] Davide Castelvecchi. "The quantum internet has arrived (and it hasn't)". In: *Nature* 554.7690 (2018), pp. 289–293 (cit. on p. 2).

[17] Nikolai Lauk et al. "Perspectives on quantum transduction". In: *Quantum Science and Technology* 5.2 (2020), p. 020501 (cit. on p. 3).

[18] Martin JA Schütz and Martin JA Schütz. "Universal quantum transducers based on surface acoustic waves". In: *Quantum Dots for Quantum Information Processing: Controlling and Exploiting the Quantum Dot Environment* (2017), pp. 143–196 (cit. on p. 3).

[19] Benjamin M Brubaker et al. "Optomechanical ground-state cooling in a continuous and efficient electro-optic transducer". In: *Physical Review X* 12.2 (2022), p. 021062 (cit. on p. 3).

[20] Timothy P McKenna et al. "Cryogenic microwave-to-optical conversion using a triply resonant lithium-niobate-on-sapphire transducer". In: *Optica* 7.12 (2020), pp. 1737–1745 (cit. on p. 3).

[21] Emil Zeuthen et al. "Figures of merit for quantum transducers". In: *Quantum Science and Technology* 5.3 (2020), p. 034009 (cit. on p. 3).

[22] Yao-Lung L Fang, Huaixiu Zheng, and Harold U Baranger. "One-dimensional waveguide coupled to multiple qubits: photon-photon correlations". In: *EPJ Quantum Technology* 1.1 (2014), pp. 1–13 (cit. on p. 3).

[23] Changchun Zhong et al. "Proposal for heralded generation and detection of entangled microwave–optical-photon pairs". In: *Physical review letters* 124.1 (2020), p. 010511 (cit. on p. 3).

[24] Stefan Krastanov et al. "Optically Heralded Entanglement of Superconducting Systems in Quantum Networks". In: *Physical Review Letters* 127.4 (2021), p. 040503 (cit. on p. 3).

[25] Galan Moody et al. "2022 Roadmap on integrated quantum photonics". In: *Journal of Physics: Photonics* 4.1 (2022), p. 012501 (cit. on p. 3).

[26] TP Harty et al. "High-fidelity preparation, gates, memory, and readout of a trapped-ion quantum bit". In: *Physical review letters* 113.22 (2014), p. 220501 (cit. on p. 3).

[27] CJ Ballance et al. "High-fidelity two-qubit quantum logic gates using trapped calcium-43 ions". In: *arXiv preprint arXiv:1406.5473* (2014) (cit. on p. 3).

[28] Stephanie Wehner, David Elkouss, and Ronald Hanson. "Quantum internet: A vision for the road ahead". In: *Science* 362.6412 (2018), eaam9288 (cit. on p. 3).

[29] Pengfei Wang et al. "Single ion qubit with estimated coherence time exceeding one hour". In: *Nature communications* 12.1 (2021), pp. 1–8 (cit. on p. 3).

[30] C Monroe et al. "Large-scale modular quantum-computer architecture with atomic memory and photonic interconnects". In: *Physical Review A* 89.2 (2014), p. 022317 (cit. on p. 3).

[31] Shaobo Gao et al. "Optimization of Scalable Ion-Cavity Interfaces for Quantum Photonic Networks". In: *Physical Review Applied* 19.1 (2023), p. 014033 (cit. on p. 3).

[32] Will Salmon et al. "Gauge-independent emission spectra and quantum correlations in the ultrastrong coupling regime of open system cavity-QED". In: *Nanophotonics* 11.8 (2022), pp. 1573–1590 (cit. on p. 3).

[33] Colin D Bruzewicz et al. "Trapped-ion quantum computing: Progress and challenges". In: *Applied Physics Reviews* 6.2 (2019), p. 021314 (cit. on pp. 3, 4).

[34] Ümit Kaya. *The Laser Optical Cavity*. `https://chem.libretexts.org/@go/page/13651` (cit. on p. 3).

[35] Daniele Cuomo et al. "Optimized Compiler for Distributed Quantum Computing". In: *ACM Transactions on Quantum Computing* (2023) (cit. on p. 4).

[36] Marcello Caleffi and Angela Sara Cacciapuoti. "Quantum Switch for the Quantum Internet: Noiseless Communications through Noisy Channels." In: *IEEE Journal on Selected Areas in Communications* 38.3 (2020), pp. 575–588 (cit. on p. 4).

[37] Wolfgang Paul. "Electromagnetic traps for charged and neutral particles". In: *Reviews of modern physics* 62.3 (1990), p. 531 (cit. on p. 4).

[38] Laurent Stephenson. "Entanglement between nodes of a quantum network". PhD thesis. University of Oxford, 2019 (cit. on pp. 4, 5).

[39] DP Nadlinger et al. "Experimental quantum key distribution certified by Bell's theorem". In: *Nature* 607.7920 (2022), pp. 682–686 (cit. on p. 4).

[40] Daniel KL Oi, Simon J Devitt, and Lloyd CL Hollenberg. "Scalable error correction in distributed ion trap computers". In: *Physical Review A* 74.5 (2006), p. 052313 (cit. on pp. 4, 7).

[41] Norbert Lütkenhaus, John Calsamiglia, and K-A Suominen. "Bell measurements for teleportation". In: *Physical Review A* 59.5 (1999), p. 3295 (cit. on p. 5).

[42] Raju Valivarthi et al. "Efficient Bell state analyzer for time-bin qubits with fast-recovery WSi superconducting single photon detectors". In: *Optics express* 22.20 (2014), pp. 24497–24506 (cit. on p. 5).

[43] Klaus Mattle et al. "Dense coding in experimental quantum communication". In: *Physical Review Letters* 76.25 (1996), p. 4656 (cit. on p. 5).

[44] Filip Rozpedek et al. "Optimizing practical entanglement distillation". In: *Physical Review A* 97.6 (2018), p. 062333 (cit. on p. 6).

[45] Norbert Kalb et al. "Entanglement distillation between solid-state quantum network nodes". In: *Science* 356.6341 (2017), pp. 928–932 (cit. on p. 6).

[46] Xiao-Min Hu et al. “Long-distance entanglement purification for quantum communication”. In: *Physical Review Letters* 126.1 (2021), p. 010503 (cit. on p. 6).

[47] Michael A Nielsen. “Quantum computation by measurement and quantum memory”. In: *Physics Letters A* 308.2-3 (2003), pp. 96–100 (cit. on p. 7).

[48] Rodney Van Meter et al. “A quantum internet architecture”. In: *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE. 2022, pp. 341–352 (cit. on p. 7).

[49] Wojciech Kozlowski et al. *Architectural Principles for a Quantum Internet.* Internet-Draft draft-irtf-qirg-principles-03. Work in Progress. Internet Engineering Task Force, June 2021. 39 pp. (cit. on p. 7).

[50] Mihir Pant et al. “Routing entanglement in the quantum internet”. In: *npj Quantum Information* 5.1 (2019), pp. 1–9 (cit. on p. 7).

[51] Seid Koudia et al. “How deep the theory of quantum communications goes: Superadditivity, superactivation and causal activation”. In: *IEEE Communications Surveys & Tutorials* (2022) (cit. on p. 7).

[52] Jessica Illiano et al. “Quantum Internet: from Medium Access Control to Entanglement Access Control”. In: *arXiv preprint arXiv:2205.11923* (2022) (cit. on p. 7).

[53] Guus Avis, Filip Rozpedek, and Stephanie Wehner. “Analysis of Multipartite Entanglement Distribution using a Central Quantum-Network Node”. In: *arXiv preprint arXiv:2203.05517* (2022) (cit. on p. 7).

[54] Julius Wallnöfer et al. “Multipartite state generation in quantum networks with optimal scaling”. In: *Scientific reports* 9.1 (2019), pp. 1–18 (cit. on p. 7).

[55] Yao Lu et al. “Global entangling gates on arbitrary ion qubits”. In: *Nature* 572.7769 (2019), pp. 363–367 (cit. on p. 7).

[56] Jorge Casanova et al. “Quantum simulation of interacting fermion lattice models in trapped ions”. In: *Physical review letters* 108.19 (2012), p. 190502 (cit. on p. 7).

[57] Svetoslav S Ivanov, Peter A Ivanov, and Nikolay V Vitanov. “Efficient construction of three-and four-qubit quantum gates by global entangling gates”. In: *Physical Review A* 91.3 (2015), p. 032311 (cit. on p. 7).

[58] Esteban A Martinez et al. “Compiling quantum algorithms for architectures with multi-qubit gates”. In: *New Journal of Physics* 18.6 (2016), p. 063029 (cit. on p. 7).

[59] Liam Madden and Andrea Simonetto. “Best approximate quantum compiling problems”. In: *ACM Transactions on Quantum Computing* 3.2 (2022), pp. 1–29 (cit. on p. 8).

[60] Yuan-Hang Zhang et al. “Topological quantum compiling with reinforcement learning”. In: *Physical Review Letters* 125.17 (2020), p. 170501 (cit. on p. 8).

[61] Peter J Karalekas et al. “A quantum-classical cloud platform optimized for variational hybrid algorithms”. In: *Quantum Science and Technology* 5.2 (2020), p. 024003 (cit. on p. 8).

[62] Lorenzo Moro et al. “Quantum Compiling by Deep Reinforcement Learning”. In: *Nature Communications Physics* 4.178 (2021) (cit. on p. 8).

[63] Marco Maronese et al. “Quantum compiling”. In: *Quantum Computing Environments*. Springer, 2022, pp. 39–74 (cit. on p. 8).

[64] Kyle EC Booth et al. “Comparing and integrating constraint programming and temporal planning for quantum circuit compilation”. In: *28th international conference on automated planning and scheduling*. 2018 (cit. on p. 8).

[65] Davide Ferrari and Michele Amoretti. “Noise-adaptive quantum compilation strategies evaluated with application-motivated benchmarks”. In: *Proceedings of the 19th ACM International Conference on Computing Frontiers*. 2022, pp. 237–243 (cit. on p. 8).

[66] Stefan Hillmich, Alwin Zulehner, and Robert Wille. “Exploiting quantum teleportation in quantum circuit mapping”. In: *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE. 2021, pp. 792–797 (cit. on p. 8).

[67] Lukas Burgholzer, Sarah Schneider, and Robert Wille. “Limiting the Search Space in Optimal Quantum Circuit Mapping”. In: *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE. 2022, pp. 466–471 (cit. on p. 8).

[68] Dmitri Maslov, Sean M Falconer, and Michele Mosca. “Quantum circuit placement”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.4 (2008), pp. 752–763 (cit. on p. 8).

[69] Marcos Yukio Siraichi et al. “Qubit allocation”. In: *Proceedings of the 2018 International Symposium on Code Generation and Optimization*. 2018, pp. 113–125 (cit. on p. 8).

[70] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. “Mapping quantum circuits to IBM QX architectures using the minimal number of `SWAP` and `H` operations”. In: *2019 56th ACM/IEEE Design Automation Conference*. IEEE. 2019, pp. 1–6 (cit. on p. 8).

[71] Gushu Li, Yufei Ding, and Yuan Xie. “Tackling the qubit mapping problem for NISQ-era quantum devices”. In: *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019, pp. 1001–1014 (cit. on p. 8).

[72] Alwin Zulehner and Robert Wille. “Compiling $SU(4)$ quantum circuits to IBM QX architectures”. In: *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 2019, pp. 185–190 (cit. on p. 8).

[73] Toshinari Itoko et al. “Quantum circuit compilers using gate commutation rules”. In: *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 2019, pp. 191–196 (cit. on p. 8).

[74] Richard Cleve and John Watrous. “Fast parallel circuits for the quantum Fourier transform”. In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE. 2000, pp. 526–536 (cit. on p. 8).

Chapter 2

Quantum logic essentials

2.1 Quantum programming

2.1.1 Universality

In this section we report some preliminaries that explain how a quantum processor can run any algorithm by means of a restricted set of gates. Such a restricted set is called *universal* for this reason. We briefly get through the Boolean logic – which governs classical computation – and how we can express any Boolean function within the quantum framework. This gives provide the reader with a perspective of how quantum logic can express a wider group of functions.

Boolean logic

Classical computation is deeply based on Boolean logic. Since classical technologies are really advanced and benefits from many years of research on physical implementations, several Boolean operators find direct implementation as classical gates. However, our interested is narrowed to how we can express any boolean function by means of quantum operators. Hence we can restrict the discussion to a single logical operator: $\neg(\mathtt{b}_1 \wedge \mathtt{b}_2)$. In fact, for any Boolean variables $\mathtt{b}_1, \mathtt{b}_2 \in \{\mathtt{0}, \mathtt{1}\}$, the $\neg(\mathtt{b}_1 \wedge \mathtt{b}_2)$ operator[1] is universal to Boolean logic [1], hence we need to find a quantum operator able to realize it. The Toffoli operator [2] does the job. Formally, a classical state $\mathtt{b}_1 \cdot \mathtt{b}_2$ gets encoded in the quantum state $|\mathtt{b}_1\rangle \otimes |\mathtt{b}_2\rangle \otimes |\mathtt{1}\rangle$. By applying the Toffoli operator to the encoded system, the last qubit encodes the Boolean state $\neg(\mathtt{b}_1 \wedge \mathtt{b}_2)$. This is shown in Figure 2.1.

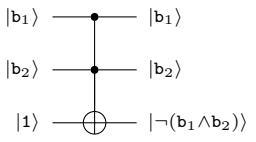


Figure 2.1: $\neg(\mathtt{b}_1 \wedge \mathtt{b}_2)$ operator by means of a Toffoli.

To date, there is no direct physical implementation for the Toffoli operator. It needs to be expressed as a composition of quantum operators physically realizable.

Quantum logic

We here report a step-by-step introduction to quantum logic and what a quantum processors should have to provide universal computing. Quantum computing works with the logic of pure states. A Hilbert space $\mathbb{H}$ of dimension d is closed under the *unitary group* of degree d. This means that for any pure state $|\varphi\rangle \in \mathbb{H}$ and any unitary operator $\mathtt{U}$, it results $\mathtt{U}\,|\varphi\rangle \in \mathbb{H}$.

A generic quantum algorithm can be expressed as a system initialized to $|\mathtt{0}\rangle^{\otimes d}$ and a unitary $\mathtt{U}$ operating on it. Figure 2.2 gives a circuit representation.

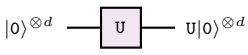


Figure 2.2: Generic algorithm expressed as a unitary, operating over a $|\mathtt{0}\rangle^{\otimes d}$ state.

Since quantum processors do not supply as primitive operator a generic unitary, this is subject to one or more steps of *decomposition* or *synthesis*. A generic decomposition is showed in circuit of Figure 2.3.

A universal operator set should be *efficient*, in the sense that the overhead caused by the decomposition from a d-degree unitary to the operator set is upper-bounded by some polynomial

[1] The value is true iff no more than one variable is true.

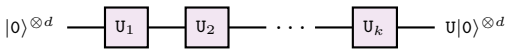


Figure 2.3: Generic decomposition of $\mathtt{U}$ into k unitaries.

function. There are several (historically) important results showing how such a requirement is achievable. We start from one coming from the work done in [3, 4], showing that, given a generic 2-degree unitary $\mathtt{U}$ – i.e. it operates over a single qubit –, the following operator is universal:

$$\wedge(\mathtt{U}) \equiv |0\rangle\langle 0| \otimes \mathbb{1} + |1\rangle\langle 1| \otimes \mathtt{U} \tag{2.1}$$

This is a controlled-$\mathtt{U}$ operator in Feynmann's notation [5]. We will use this notation throughout this thesis because of its versatility. Even if it is a bit outdated, we think it helps to highlight the logic behind the operators; it also helps us to keep consistency throughout the chapters. In the circuit model the subject operator appears as in Figure 2.4.

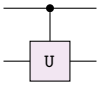


Figure 2.4: Circuit representation for the $\wedge(\mathtt{U})$ operator.

One can notice that the group of equation (2.1) is pretty compact, as it involves only 2-qubit operators where one of them act always as control. Nevertheless, some decomposition step is necessary to get closer to what real processors can actually offer. To this aim, we need to introduce some further operators. The first one is known as *special unitary*. We refer to this operator as $\mathtt{V}_{\alpha,\beta,\delta}$ and it is defined as follows [6]:

$$\mathtt{V}_{\alpha,\beta,\delta} \equiv \begin{pmatrix} e^{\mathrm{i}(\alpha+\beta)/2}\cos\delta/2 & e^{\mathrm{i}(\alpha-\beta)/2}\sin\delta/2 \\ -e^{\mathrm{i}(-\alpha+\beta)/2}\sin\delta/2 & e^{\mathrm{i}(-\alpha-\beta)/2}\cos\delta/2 \end{pmatrix} \tag{2.2}$$

The second operator represent a *global phase shift*:

$$\mathtt{G}_\gamma \equiv \begin{pmatrix} e^{\mathrm{i}\gamma} & 0 \\ 0 & e^{\mathrm{i}\gamma} \end{pmatrix} \tag{2.3}$$

It results that special unitaries composed with global phase shifts characterizes the group of unitaries. It follows the same statement in matrix form:

$$\mathtt{U}_{\gamma,\alpha,\beta,\delta} \equiv \mathtt{G}_\gamma \mathtt{V}_{\alpha,\beta,\delta} \tag{2.4}$$

As consequence, the first decomposition we can apply comes from the circuit equivalence of Figure 2.5.

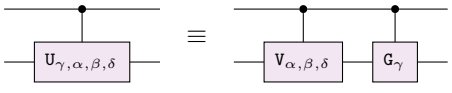


Figure 2.5: First decomposition as a result of equivalence (2.4).

A conditioned global phase shift is equivalent to a *relative phase shift*. Thus, let $\mathtt{R}_\gamma$ such an operator, defined as follows:

$$\mathtt{R}_\gamma \equiv \begin{pmatrix} 1 & 0 \\ 0 & e^{\mathrm{i}\gamma} \end{pmatrix} \tag{2.5}$$

The above statement allows us to *synthesize* $\wedge(\mathtt{G}_\gamma)$ into $\mathtt{R}_\gamma \otimes \mathbb{1}$. Figure 2.6 shows the corresponding circuit equivalence.

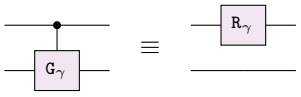


Figure 2.6: Circuit representation of the equivalence $\wedge(\mathtt{G}_\gamma) \equiv \mathtt{R}_\gamma \otimes \mathbb{1}$.

The final step to achieve universal computing through physically realizable operators is to decompose $\wedge(\mathtt{V}_{\alpha,\beta,\delta})$. To this aim, let us introduce the rotational operators over the Pauli axes. Namely, $\mathtt{X}_\gamma \equiv \mathtt{e}^{-\mathtt{i}\mathtt{X}\gamma/2}$, $\mathtt{Y}_\gamma \equiv \mathtt{e}^{-\mathtt{i}\mathtt{Y}\gamma/2}$ and $\mathtt{Z}_\gamma \equiv \mathtt{e}^{-\mathtt{i}\mathtt{Z}\gamma/2}$. Notice also that

$$\mathtt{e}^{-\mathtt{i}\mathtt{E}\gamma/2} = \cos\gamma/2 - \mathtt{i}\mathtt{E}\sin\gamma/2, \tag{2.6}$$

holds for any $\mathtt{E} \in \{\mathtt{X}, \mathtt{Y}, \mathtt{Z}\}$. Therefore, whenever $\gamma = \pi$, each rotational operator relates to the corresponding Pauli operator $\mathtt{X}$, $\mathtt{Y}$ or $\mathtt{Z}$. Formally, they are equivalent up to the global phase $\mathtt{G}_{-\pi/2} \equiv -\mathtt{i}$. E.g. $\mathtt{X}_\pi \cong \mathtt{X}$.

Now we proceed by reporting the results coming from [6]. Let $\mathtt{A}_{\alpha,\delta}$, $\mathtt{B}_{\delta,\alpha,\beta}$, $\mathtt{C}_{\alpha,\beta}$ be a triplet of special unitaries defined as follows:

- $\mathtt{A}_{\alpha,\delta} = \mathtt{Y}_{\delta/2}\mathtt{Z}_\alpha$;
- $\mathtt{B}_{\delta,\alpha,\beta} = \mathtt{Y}_{-\delta/2}\mathtt{Z}_{-(\alpha+\beta)/2}$;
- $\mathtt{C}_{\alpha,\beta} = \mathtt{Z}_{(\beta-\alpha)/2}$.

With such a triplet, together with $\wedge(\mathtt{X})$ we are able to decompose $\wedge(\mathtt{V}_{\alpha,\beta,\delta})$, according to the circuit equivalence shown in Figure 2.7.

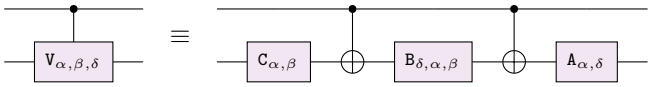


Figure 2.7: Decomposition of $\wedge(\mathtt{V}_{\alpha,\beta,\delta})$.

We can finally show the universality by composing the two results and getting a decomposition for $\wedge(\mathtt{U}_{\gamma,\alpha,\beta,\delta})$ – see Figure 2.8.

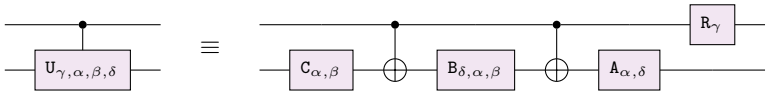


Figure 2.8: Decomposition of $\wedge(\mathtt{U}_{\gamma,\alpha,\beta,\delta})$.

Theorem. *The IBM gate set [7, 8, 9, 10] is universal.*

Proof. All the processors supplied by the IBM cloud have gate set $\{\mathtt{Z}_\gamma, \mathtt{X}_{\pi/2}, \mathtt{X}, \wedge(\mathtt{X})\}$. To prove the universality of such a set, notice that $\mathtt{R}_\gamma \cong \mathtt{Z}_\gamma$. Furthermore, any special unitary $\mathtt{V}_{\alpha,\beta,\delta}$ can be factorized as follows [6]:

$$\mathtt{V}_{\alpha,\beta,\delta} \equiv \mathtt{Z}_\alpha \mathtt{Y}_\delta \mathtt{Z}_\beta$$

Since an IBM processor can run natively a generic $\mathtt{Z}_\gamma$ gate, we only need to synthesise $\mathtt{Y}_\delta$. This can be done, by using operations from the gate set only:

$$\mathtt{Y}_\delta \equiv \mathtt{X}_{\pi/2}\mathtt{Z}_{\delta+\pi}\mathtt{X}_{3\pi/2}$$

In conclusion, since the gate set also provides $\wedge(\mathtt{X})$, any $\wedge(\mathtt{U})$ can be realized through a composition of operations coming from the `IBM` gate set. □

Not all the existing processors are universal. `D-Wave` ones are an example. In fact, the company goal is to build specific-purpose processors, meant to explore the field of optimization problems through *quantum annealing* procedures [11].

Relation between classical and quantum logic

It has been shown [12] that, to achieve quantum universality, one can start from an operator universal in the Boolean functions, i.e. the Toffoli, and by adding only a single 1-qubit operator, the operator set is quantum universal. The subject operator can be expressed as $\mathtt{X}_{\pi/2}\mathtt{Z}_{\pi/2}\mathtt{X}_{\pi/2}$[2]. As stated in [13], this "[...] can be interpreted as saying that Fourier transform is really all there is to quantum computation on top of classical", since $\mathtt{X}_{\pi/2}\mathtt{Z}_{\pi/2}\mathtt{X}_{\pi/2}$ corresponds to a Fourier transform.

2.1.2 The Clifford group

The Clifford group $\mathbb{C}$ dues its importance to its implication in fault-tolerant computation [14], simulation [15] and benchmarking [16]. Such a group is generated by 3 operators:

$$\mathbb{C} \equiv \langle \wedge(\mathtt{X}), \mathtt{X}_{\pi/2}, \mathtt{Z}_{\pi/2} \rangle. \tag{2.7}$$

$\mathbb{C}$ can be efficiently simulated by a classical computer [17]. This has as comeback that one can evaluate fault-tolerant protocols classically. As drawback, it is not universal. However to achieve universality while at the same time providing an operator set which can realize any quantum evolution efficiently one need to add a single 1-qubit operator, usually assumed to be $\mathtt{R}_{\pi/4}$[3] or the corresponding in rotational terms $\mathtt{Z}_{\pi/4}$. In fact, any unitary of dimensionality 2 can be approximated efficiently and with arbitrary precision [18, 19, 20]. Formally, by properly composing single qubit operators coming from $\mathbb{C}$ – i.e. $\mathtt{X}_{\pi/2}, \mathtt{Z}_{\pi/2}$ – together with $\mathtt{Z}_{\pi/4}$ one can achieve universality in the *dense* sense, by only introducing a polynomial overhead[4].

Since we are facing with a discrete operator set composed by even fractions of π only, we can re-state the nomenclature. Namely, the same universal operator set can be expressed in the following intuitive way:

$$\mathbb{C}^+ \equiv \langle \wedge(\mathtt{X}), \mathtt{X}^{1/2}, \mathtt{Z}^{1/2}, \mathtt{Z}^{1/4} \rangle. \tag{2.8}$$

This nomenclature stresses the logic behind the relation Pauli-rotational gates, as the power function degree says how many times one needs to apply the rotational gate to simulate a Pauli operator.

[2] As well as $\mathtt{Y}_{\pi/2}\mathtt{X}$ or as the more common *Hadamard* gate `H`. We are not going to use the latter in this thesis as we opted to keep the treating closer to real gate implementations.

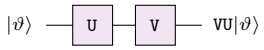
[3] Commonly referred as the `T` gate.

[4] Other extensions of $\mathbb{C}$ may be of interest. E.g. in Ref. [21] authors consider $\wedge(\mathtt{Z}_{\pi/2})$ in their generator set.

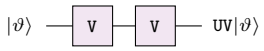
2.1.3 Programming in higher order framework

Time-ordered framework

So far we have implicitly assumed that a quantum evolution undergoes a time-ordered definition. Formally, consider two unitaries $\mathtt{U}$ and $\mathtt{V}$ and a state $|\vartheta\rangle$. Then, any time-ordered framework force us to chose whether $\mathtt{U}$ operates on $|\vartheta\rangle$ before or after $\mathtt{V}$. In circuit representation, these two cases are respectively



and



However quantum mechanics allows to think of more general frameworks, where, not only states, but also operators can be superposed to create more complex systems. Such systems may bring new non-classical advantages. Attempts in formally designing a higher-order framework is an active branch of research [22, 23], but this is out of scope. Rather, in the following, we work with a higher-order *oracle* implementing an *indefinite casual order*.

Indefinite causal orders

The *indefinite causal order* is an interesting property of quantum mechanics. In brief, it is a quantum evolution where two or more operations occur, but the order in which they occur is causally ordered by an extra quantum system. This creates a superposition of causal orders among those operations. Such a resource can be used for several purposes. In Refs. [24, 25, 26], the indefinite causal orders is considered for thermalization protocols. A big line of investigation deals with enhancing quantum communication [27, 28, 29, 30]; we gave the first experimental witness for such a resource [31][5]. As regards computation, indefinite causal order would speed up tasks that involves permuting operations [32, 33, 34]. In Ref. [35] authors implement a photonic-based discrimination protocol solved by *superposing unitaries*.

The indefinite causal orders for computation consists essentially on superposing two or more unitaries. The superposition is then coherently conditioned to an auxiliary qubit, called control qubit. For the case of 2 unitaries $\mathtt{U}, \mathtt{V}$, the superposing unitary operator $\mathtt{S}$ can be defined as follows:

$$\mathtt{S} = \begin{bmatrix} \mathtt{UV} & \mathbf{0} \\ \mathbf{0} & \mathtt{VU} \end{bmatrix}. \tag{2.9}$$

By treating the operator $\mathtt{S}$ as an oracle – i.e. it takes a unit of time to run – it brings the advantage of evaluating two different orders at the same time. This advantage comes from the fact that we are extending the standard framework – which originally could only superpose quantum states – to being able to superpose unitaries. This can be used, for example, in Information Processing to distinguish between different evolution by means of a single check [35]. This is unthinkable with standard frameworks where the order of execution of the unitaries must be defined.

However, implementing $\mathtt{S}$ does not necessarily reflect the advantage coming from theory. It is necessary to make a distinction to what is a real implementation of $\mathtt{S}$ and what is more like

[5] We report this in Ch. 3.

a *simulation*. In other words, implementing S means that the physical settings preserve the theoretical advantages. Only in this case one can treat S as an oracle. It is an open question if such a real implementation will ever be possible [36, 37]. In the attempt of finding a solution, a framework meant to superpose *gravitational fields* has been proposed [38].

Stemming from the above, what we can do now is to realize S by means of simulation. Namely, an equivalent evolution which does not preserve the speed-up advantage. An example is shown in Figure 2.9. This example immediately shows the theoretical loss, as it requires 2 use

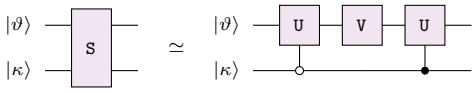


Figure 2.9: Simulation of oracle S; losing the theoretical advantage.

for one of the unitaries [39]. Specifically, since U operators run under complementary conditions, one of them does not run, meaning that one time step is always dedicated to perform an identity operation $\mathbb{1}$ – an idle time.

A simulation which looks more adapt to reflect the natural behaviour of S is represented in Figure 2.10. This circuit makes use of an auxiliary qubit initialized to $|0\rangle$. A swap conditioned on $|\kappa\rangle$ is the only potentially entangling gate between the three states. However, $|0\rangle$ has no impact on the overall system[6], up to the permutation caused by the swap operation[7]. It follows that by tracing out such a sub-system, the subject circuit implements S. Thanks to the auxiliary qubit, the two different orders can be expressed in parallel.

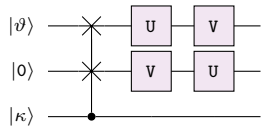


Figure 2.10: Simulation of oracle S by means an auxiliary qubit.

Nevertheless notice that it make use of a complex gate at the beginning – i.e. a controlled swap. However, to date, there is no technology providing such a gate natively, hence it is necessary to consider it as an oracle to not lose the advantage. The best we can do now is providing an efficient decomposition for it, aware of the fact that we have partial knowledge of the input – i.e. auxiliary qubit being in state $|0\rangle$. Figure 2.11 shows an optimized decomposition w.r.t. the standard one [40].

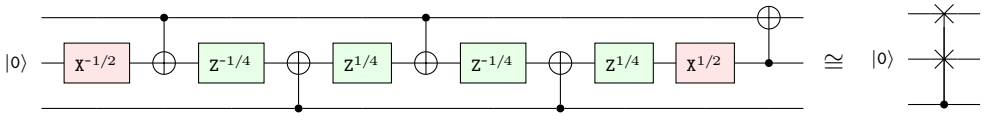


Figure 2.11: Optimized decomposition of the controlled swap operation.

Unfortunately, as long as native controlled swap are not physically implemented, the speed-up advantage – w.r.t. a time-ordered framework – is still just theoretical and it cannot be witnessed on real implementation. This is especially true when trying to superpose more than

[6] I.e., no phase shift leaks in nor out of state $|0\rangle$.

[7] Which can be retrieved through a non-destructive measurement $Z \otimes Z$. Non-destructive measurements are introduced in Chapter 3.

two unitaries. Some investigations are available in Refs. [41, 34], where the computational speed up is preserved as long as the swaps are considered oracles.

Nevertheless, a non-native implementation of indefinite causal orders may still bring some sort of quantum advantage, as it can be used for *magnitude amplification* [42, 43, 44][8]. In Section 3.3, we treat this concept – also experimentally – to enhance communication capacity.

2.2 Entanglement-based computation

Entanglement is probably the most fascinating property coming from quantum mechanics. It is also the most promising resource.

To our purpose we focus on one of the four Bell states. These states are fully entangled, meaning that their correlation is maximal and the system is said to be close. Let us introduce the $|\Phi^+\rangle$ state, defined as follow:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

Since the two system in $|\Phi^+\rangle$ present a *non-local* correlation, this state can be used to perform non-local operations. Physically speaking, this means that one need to perform what is called *entanglement generation and distribution* [46, 47, 48]:

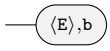
- generation; despite the non-local correlation, the generation happens between system which are in proximity one another. Generating a maximally entangled state with high fidelity is generally hard and time-consuming.
- distribution; once that the entanglement is ready, it is possible to relocate the two systems. The entanglement is, in principle, preserved.

Below we outline some of the most promising resources for distributed quantum computing, all exploiting entanglement. Most of what follows comes from the work published in [49]. Before proceeding, we need to introduce a formalism for *measurement-based computation*.

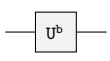
Measurement-based computation

A computing paradigm is called measurement-based whenever the quantum computation is interleaved by measurements acting on a sub-system [50]. The output of a measurement is then used to perform classical conditioned quantum operations.

A measurement over a Pauli axis `E` generates a Boolean `b`:



The output can then be used to choose whether performing or not a unitary operation `U` over another qubit:



Whenever to a Pauli gate follows a Pauli measurement, there is no need to apply the former. Precisely, instead of applying the quantum gate followed by the measurement, one can always

[8] Useful, e.g., to implement Grover's algorithm oracle [45]

perform the measurement and then applying a classical correction on the output, corresponding to the Pauli gate. Such a technique may be referred as *pushing* technique as, intuitively, one can picture this manipulation as pushing the gate beyond the measurement. See circuits in Figure 2.12 for an example.

Figure 2.12: Pushing technique to avoid quantum gates.

2.2.1 Teleportation

The first resource we report is called *teleportation* [48]. It is a protocol that, by fact, teleport quantum information from a system to another, by means of entanglement. Figure 2.13 shows the protocol steps in circuit representation. The wires are grouped by color, representing different processor, or memories. A generic processor is referred as P_i. In the case of teleportation 2 processors P_i, P_j are involved, which has the roles of *sender* and *receiver*. At the beginning of the protocol, the quantum information $|\varphi\rangle$ is located within the sender, together with half of the Bell state $|\Phi^+\rangle$. The receiver need to store the other part of the Bell state. At this point the receiver need to wait that the receiver perform a few operations meant to "inject" the information within its half part of the entangled state. Because of the entanglement, now the quantum information is already spread all over the system, which means that P_j, the receiver, also has partial knowledge of it.

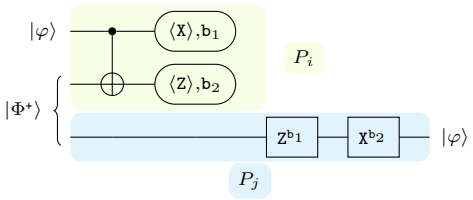


Figure 2.13: Quantum teleportation protocol between two parties.

The end of this part consists on measuring the entire system in the orthogonal basis $\mathtt{X} \otimes \mathtt{Z}$. This is a necessary step in order to ensure that the receiver will get exactly $|\varphi\rangle$. In fact, the measurement will produce two boolean values, $\mathtt{b}_1, \mathtt{b}_2$, that the receiver will need to correct its state.

To understand how such a resource can be used in computation, consider that it is often the case where two states need to interact but they are stored in such a position that does not allow them to do so, unless some middle step is performed to approach them. This means that a *routing* protocol would take care of moving those states up to a couple of qubits which are able to interact one another. This can be done by means of teleportation [51]. However, in local quantum computation, it is more common to find some smart criteria to re-arrange the state storage when necessary, by means of *swapping* protocols [52, 53, 54]. Even if this is the most common approach, it may be smart as well to instead consider teleportation as an alternative to swapping protocols.

When considering distributed architecture, this are generally assumed to deeply exploit entanglement for their interconnection. Hence, it is more likely to see in the future proposal exploiting teleportation, rather than swapping.

2.2.2 Non-local operations

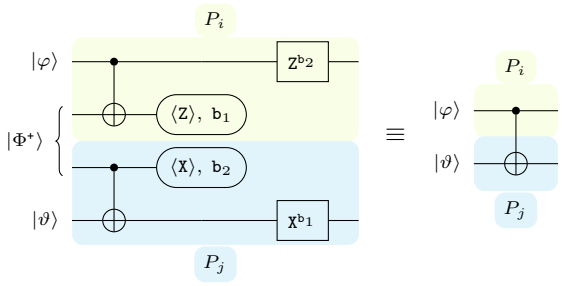


Figure 2.14: Tele-gate performing $\wedge(\mathtt{X})$ between qubits belonging different processors.

The teleportation protocol results to be a basic example of a wider class of teleporting protocols. It is in fact possible to, not only teleport state, but entire operations. For this reason the procedure we report here can be also referred as *tele-gate*.

We already discussed the importance of the $\wedge(\mathtt{X})$ operator for quantum computation in Section 2.1.1. We here report a way to perform such an operation between states belonging to different processors by means of non-local operations. In fact, since we are under the assumption the processors inter-connectivity is fundamentally based on distributed entangled states, a $\wedge(\mathtt{X})$ operator can be implemented within a few steps. These steps are shown in Figure 2.14.

Notice that performing a non-local operation is not limited to the consumption of a Bell state $|\Phi^+\rangle$. Rather, as discussed in Sec. 4.8, one can use any Bell state. For the sake of simplicity, we will always refer to $|\Phi^+\rangle$ states.

Similarly to the teleportation protocol – see Figure 2.13 –, there is a step meant two inject the control and target states within the entangled state. Thanks to this step, some quantum information is exchanged between the two parties. However, to ensure the equivalence with the subject operator, a measurement step is necessary. Hence, the state that was originally fully entangled in $|\Phi^+\rangle$ is measured in the orthogonal basis $\mathtt{Z} \otimes \mathtt{X}$. The output $\mathtt{b}_1, \mathtt{b}_2$ is then subject to a cross communication through classical channels and eventually used to perform Pauli corrections, i.e., the last step of the procedure: $\mathtt{Z}^{\mathtt{b}_2}$ over the control qubit and $\mathtt{X}^{\mathtt{b}_1}$ over the target qubit.

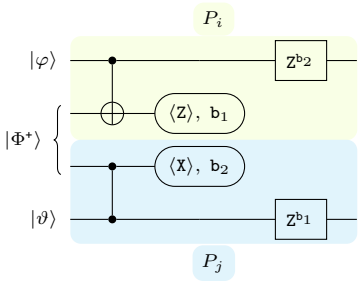


Figure 2.15: Tele-gate performing $\wedge(\mathtt{Z})$ between qubits belonging to different processors.

Mindful of Section 2.1.1, we don't need further non-local operations in order to achieve universal computation. However, it is useful to know that other tele-gates are certainly possible. E.g. Figure 2.15 shows the procedure to perform a $\wedge(\mathtt{Z})$ non-local operator.

2.2.3 Entanglement swap

Here we report a protocol known as *entanglement swap*. It is a promising procedure as it models scalable distributed architecture. Figure 2.16 shows the parties involved and their actions. Specifically, assume that two processors P_i, P_j cannot rely on a direct inter-connection

through an entanglement pair. However they both share an interconnection with a *middle* processor P_k. The middle processor has therefore stored in his memory two half of entangled pairs. By means of a local $\wedge(\mathtt{X})$ on his system, P_k inter-connects P_i with P_j. As usual, an orthogonal measurement $\mathtt{X} \otimes \mathtt{Z}$ is necessary to apply eventual corrections over P_i and P_j, which at the end of the procedure share a fully entangled state, ensuring the new inter-connection.

Depending on the time model adopted when dealing with this procedure, the inter-connectivity among the processors find different treating. For example, the work done in [55] has a more dynamic-like approach, making a distinction between *link* and *virtual link*. Such a choice probably comes from an interest in modeling a network of quantum technologies, where it is more common to deal with *on-line* combinatorial problems [56]. Instead, our focus here is to smartly model distributed architecture meant to perform algorithms. This brings us to see the inter-connectivity within a more static time model; which translates into a simpler modeling for the connectivity. We explain this in detail within next sub-section 2.2.4.

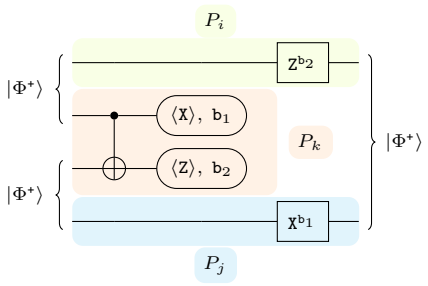


Figure 2.16: The entanglement swap protocol inter-connects two processors by means of an intermediate one, which has direct connection with both of them.

2.2.4 Entanglement paths

As our focus is on the treating of distributed quantum computing, we here provides a non-local $\wedge(\mathtt{X})$, which makes use of entanglement swaps. The circuit in Figure 2.17 comes from the combination of the basic implementation of a non-local $\wedge(\mathtt{X})$ – see Section 2.2.2 – with the entanglement swap protocol. It is important to notice that all the measurements happen at the

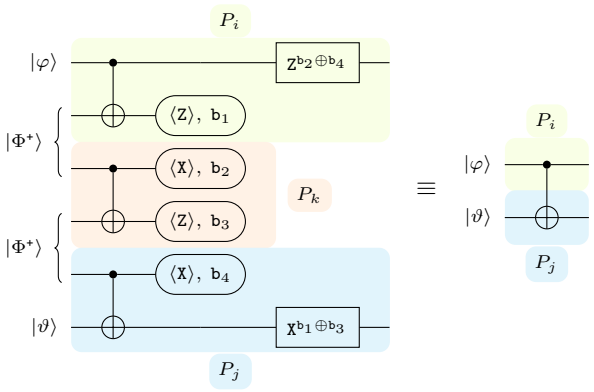
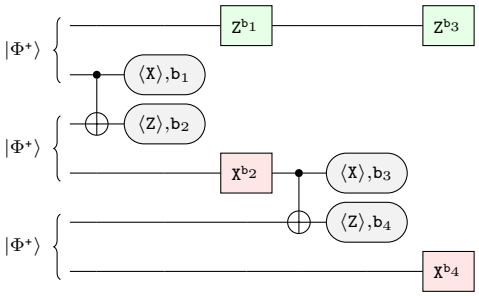


Figure 2.17: Tele-gate $\wedge(\mathtt{X})$ by means of entanglement swap.

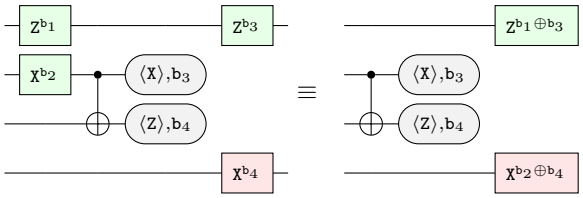
same time. Even more important to know is that this result can be generalized to any number of middle processors $P_{k_1}, P_{k_2}, \ldots, P_{k_m}$. For this reason we refer to $\{P_i, P_{k_1}, P_{k_2}, \ldots, P_{k_m}, P_j\}$ as an *entanglement path* of length $m + 2$. We now give an inductive proof for this result.

Theorem. *An entanglement path* $\{P_{i_1}, P_{i_2}, \ldots, P_{i_m}\}$ *has an implementation with depth 4.*

Proof. Consider an entanglement path of length 2. A naive realization consists on putting in strict sequence two entanglement swaps:

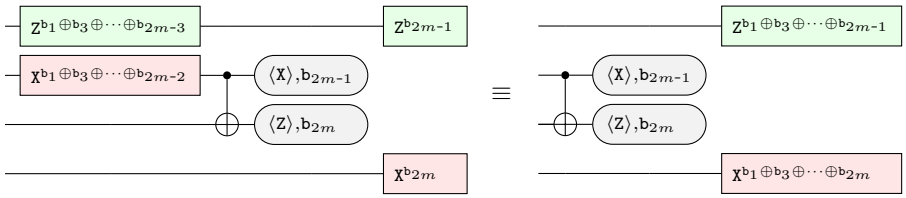


Pauli gates are the only ones we are going to optimize; since the others are independent and no optimization can be applied. What follows is the base case for the induction:



The r.h.s. of the above equation has post-processing composed by $\mathtt{Z}^{\mathtt{b}_1 \oplus \mathtt{b}_3}$ on first qubit and $\mathtt{X}^{\mathtt{b}_2 \oplus \mathtt{b}_4}$ on last qubit. Notice that the measurements are independent from other operations.

By assuming that such a shape is preserved in the inductive step, we show that this transformation can be applied to any length m:

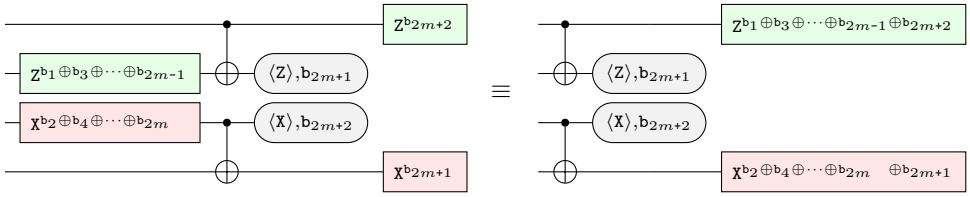


This proves that we can always consider an entanglement path $\{P_{i_1}, P_{i_2}, \ldots, P_{i_m}\}$ to have circuit depth 4. □

We just showed an efficient implementation for the entanglement path. Now we do one last step to exploit such a result and performing a generalized remote operation efficiently.

Theorem. *A tele-gate of entanglement path $\{P_{i_1}, P_{i_2}, \ldots, P_{i_{m+2}}\}$ has depth 4.*

Proof. The theorem above allows us to assume that, to perform a remote operation by using a path of length m, the computing qubits interact only with two communications qubits and depend only by Pauli operations $\mathtt{Z}^{\mathtt{b}_1 \oplus \mathtt{b}_3 \oplus \cdots \oplus \mathtt{b}_{2m-1}}$ and $\mathtt{X}^{\mathtt{b}_2 \oplus \mathtt{b}_4 \oplus \cdots \oplus \mathtt{b}_{2m}}$. We can further *propagate* such operations as follows:



In this way the measurements are independent and the depth of the circuit is not increased. □

2.2.5 Amortizing entanglement link consumption

Driven again by the aim of finding smart strategies to perform non-local gates with low consumption of entanglement links; we now investigate a way to perform multiple operations **by means of the same link**. To get a first intuition of what we are going to show, consider the following equivalence:

$$\wedge(\mathbb{1} \otimes \mathtt{X}) \cdot \wedge(\mathtt{X} \otimes \mathbb{1}) \equiv \wedge(\mathtt{X} \otimes \mathtt{X}); \tag{2.10}$$

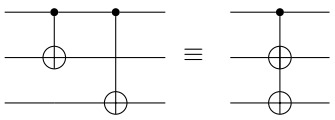


Figure 2.18: Circuit representation of the equivalence (2.10).

which has circuit representation reported in Figure 2.18. Behind its simplicity, equivalence (2.10) gives us a different perspective to implement multiple non-local operations by means of one entanglement link. This happens for the case of two processors P_i, P_j, but it can be generalized. In fact, as showed in [57, 58], any $\wedge(\mathtt{X}^{\otimes m})$ – with system of dimensionality $n > m$ and control qubit being stored in a different processor w.r.t. the target one – admits an implementation that needs one entanglement link only.

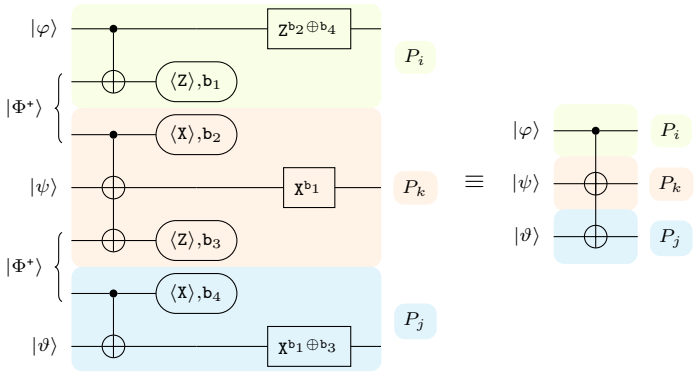


Figure 2.19: Remote implementation of $\wedge(\mathtt{X} \otimes \mathtt{X})$.

Stemming from equation (2.10), the two target systems are essentially independent, up to the common control qubit. Hence, there is no reason to restrict them to be part of the same processor. For the case under consideration – i.e. $\wedge(\mathtt{X} \otimes \mathtt{X})$ –, the maximum number of processors is three. Figure 2.19 shows the circuit protocol to perform $\wedge(\mathtt{X} \otimes \mathtt{X})$, where the system is spread over three processors.

Thanks to the entanglement path definition given in Sec. 2.2.4, such a protocol runs efficiently also for system linked only by an entanglement path.

We will use this technique in Ch. 4 to minimize entanglement links consumption, up to gain optimal solutions in some interesting scenarios.

References

[1] William Wernick. "Complete sets of logical functions". In: *The Journal of Symbolic Logic* 7.2 (1942), pp. 99–99 (cit. on p. 14).

[2] Tommaso Toffoli. "Reversible computing". In: *International colloquium on automata, languages, and programming*. Springer. 1980, pp. 632–644 (cit. on p. 14).

[3] Adriano Barenco. "A universal two-bit gate for quantum computation". In: *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences* 449.1937 (1995), pp. 679–683 (cit. on p. 15).

[4] David P DiVincenzo. "Two-bit gates are universal for quantum computation". In: *Physical Review A* 51.2 (1995), p. 1015 (cit. on p. 15).

[5] Richard P Feynman. "Quantum mechanical computers". In: *Optics news* 11.2 (1985), pp. 11–20 (cit. on p. 15).

[6] Adriano Barenco et al. "Elementary gates for quantum computation". In: *Physical review A* 52.5 (1995), p. 3457 (cit. on pp. 15, 16).

[7] IBM. *IBM Quantum backends.* `https://github.com/Qiskit/ibmq-device-information/tree/master/backends`. [Online; accessed 14-november-2022] (cit. on p. 16).

[8] IBM. *A high-fidelity, two-qubit cross-resonance gate using interference couplers.* `https://research.ibm.com/publications/a-high-fidelity-two-qubit-cross-resonance-gate-using-interference-couplers`. [Online; accessed 29-november-2022] (cit. on p. 16).

[9] Qiskit. *The cross-resonance gate for superconducting qubits implements a $Z\otimes X$ interaction.* `https://qiskit.org/documentation/stubs/qiskit.circuit.library.RZXGate.html`. [Online; accessed 29-november-2022] (cit. on p. 16).

[10] Chad Rigetti and Michel Devoret. "Fully microwave-tunable universal gates in superconducting qubits with linear couplings and fixed transition frequencies". In: *Physical Review B* 81.13 (2010), p. 134507 (cit. on p. 16).

[11] D-Waver. *What is quantum annealing?* `https://docs.dwavesys.com/docs/latest/c_gs_2.html`. [Online; accessed 14-november-2022] (cit. on p. 17).

[12] Yaoyun Shi. "Both Toffoli and controlled-NOT need little help to do universal quantum computing". In: *Quantum Information & Computation* 3.1 (2003), pp. 84–92 (cit. on p. 17).

[13] Dorit Aharonov. "A simple proof that Toffoli and Hadamard are quantum universal". In: *arXiv preprint quant-ph/0301040* (2003) (cit. on p. 17).

[14] Jonas T Anderson, Guillaume Duclos-Cianci, and David Poulin. "Fault-tolerant conversion between the steane and reed-muller quantum codes". In: *Physical review letters* 113.8 (2014), p. 080501 (cit. on p. 17).

[15] Daniel Gottesman. "Theory of fault-tolerant quantum computation". In: *Physical Review A* 57.1 (1998), p. 127 (cit. on p. 17).

[16] Easwar Magesan, Jay M Gambetta, and Joseph Emerson. "Characterizing quantum gates via randomized benchmarking". In: *Physical Review A* 85.4 (2012), p. 042311 (cit. on p. 17).

[17] Craig Gidney. "Stim: a fast stabilizer circuit simulator". In: *Quantum* 5 (2021), p. 497 (cit. on p. 17).

[18] Christopher M Dawson and Michael A Nielsen. "The solovay-kitaev algorithm". In: *arXiv preprint quant-ph/0505030* (2005) (cit. on p. 17).

[19] Vadym Kliuchnikov. "Synthesis of unitaries with Clifford+ T circuits". In: *arXiv preprint arXiv:1306.3200* (2013) (cit. on p. 17).

[20] Farrokh Vatan and Colin Williams. "Optimal quantum circuits for general two-qubit gates". In: *Physical Review A* 69.3 (2004), p. 032315 (cit. on p. 17).

[21] Andrew N Glaudell, Neil J Ross, and Jacob M Taylor. "Optimal two-qubit circuits for universal fault-tolerant quantum computation". In: *npj Quantum Information* 7.1 (2021), pp. 1–11 (cit. on p. 17).

[22] Giulio Chiribella, G Mauro D'Ariano, and Paolo Perinotti. "Quantum circuit architecture". In: *Physical review letters* 101.6 (2008), p. 060401 (cit. on p. 18).

[23] Giulio Chiribella and Hlér Kristjánsson. "Quantum Shannon theory with superpositions of trajectories." In: *Proceedings of the Royal Society A* 475.2225 (2019), p. 20180903 (cit. on p. 18).

[24] David Felce and Vlatko Vedral. "Quantum refrigeration with indefinite causal order". In: *Physical review letters* 125.7 (2020), p. 070603 (cit. on p. 18).

[25] Kyrylo Simonov et al. "Work extraction from coherently activated maps via quantum switch". In: *Physical Review A* 105.3 (2022), p. 032217 (cit. on p. 18).

[26] Tamal Guha, Mir Alimuddin, and Preeti Parashar. "Thermodynamic advancement in the causally inseparable occurrence of thermal maps". In: *Physical Review A* 102.3 (2020), p. 032215 (cit. on p. 18).

[27] Daniel Ebler, Sina Salek, and Giulio Chiribella. "Enhanced communication with the assistance of indefinite causal order". In: *Physical review letters* 120.12 (2018), p. 120502 (cit. on p. 18).

[28] Sina Salek, Daniel Ebler, and Giulio Chiribella. "Quantum communication in a superposition of causal orders." In: *arXiv preprint arXiv:1809.06655* (2018) (cit. on p. 18).

[29] Marcello Caleffi and Angela Sara Cacciapuoti. "Quantum Switch for the Quantum Internet: Noiseless Communications through Noisy Channels." In: *IEEE Journal on Selected Areas in Communications* 38.3 (2020), pp. 575–588 (cit. on p. 18).

[30] Seid Koudia et al. "How deep the theory of quantum communications goes: Superadditivity, superactivation and causal activation". In: *IEEE Communications Surveys & Tutorials* (2022) (cit. on p. 18).

[31] Daniele Cuomo, Marcello Caleffi, and Angela Sara Cacciapuoti. "Experiencing the communication advantage of the Superposition of Causal Orders". In: *2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE. 2021, pp. 181–185 (cit. on p. 18).

[32] Giulio Chiribella. "Perfect discrimination of no-signalling channels via quantum superposition of causal structures." In: *Physical Review A* 86.4 (2012), p. 040301 (cit. on p. 18).

[33] Mateus Araújo, Fabio Costa, and Časlav Brukner. "Computational advantage from quantum-controlled ordering of gates." In: *Physical Review Letters* 113.25 (2014), p. 250402 (cit. on p. 18).

[34] Timoteo Colnaghi et al. "Quantum computation with programmable connections between gates". In: *Physics Letters A* 376.45 (2012), pp. 2940–2943 (cit. on pp. 18, 20).

[35] Lorenzo M Procopio et al. “Experimental superposition of orders of quantum gates”. In: *Nature communications* 6.1 (2015), pp. 1–6 (cit. on p. 18).

[36] V Vilasini and Renato Renner. “Embedding cyclic causal structures in acyclic spacetimes: no-go results for process matrices”. In: *arXiv preprint arXiv:2203.11245* (2022) (cit. on p. 19).

[37] Giulia Rubino et al. “Experimental verification of an indefinite causal order.” In: *Science Advances* 3.3 (2017), e1602589 (cit. on p. 19).

[38] Nikola Paunković and Marko Vojinović. “Causal orders, quantum circuits and spacetime: distinguishing between definite and superposed causal orders”. In: *Quantum* 4 (2020), p. 275 (cit. on p. 19).

[39] Giulio Chiribella et al. “Quantum computations without definite causal structure.” In: *Physical Review A* 88.2 (2013), p. 022318 (cit. on p. 19).

[40] Vivek V Shende and Igor L Markov. “On the CNOT-cost of TOFFOLI gates”. In: *Quantum Information & Computation* 9.5 (2009), pp. 461–486 (cit. on p. 19).

[41] Martin J Renner and Časlav Brukner. “Computational Advantage from a Quantum Superposition of Qubit Gate Orders”. In: *Physical Review Letters* 128.23 (2022), p. 230503 (cit. on p. 20).

[42] Alessandro Zavatta, Jaromır Fiurášek, and Marco Bellini. “A high-fidelity noiseless amplifier for quantum light states”. In: *Nature Photonics* 5.1 (2011), pp. 52–56 (cit. on p. 20).

[43] MS Kim et al. “Scheme for proving the bosonic commutation relation using single-photon interference”. In: *Physical review letters* 101.26 (2008), p. 260401 (cit. on p. 20).

[44] A Zavatta et al. “Experimental demonstration of the bosonic commutation relation via superpositions of quantum operations on thermal light fields”. In: *Physical Review Letters* 103.14 (2009), p. 140406 (cit. on p. 20).

[45] Christof Zalka. “Grover’s quantum searching algorithm is optimal”. In: *Physical Review A* 60.4 (1999), p. 2746 (cit. on p. 20).

[46] Angela Sara Cacciapuoti et al. “Quantum internet: networking challenges in distributed quantum computing”. In: *IEEE Network* 34.1 (2019), pp. 137–143 (cit. on p. 20).

[47] Angela Sara Cacciapuoti et al. “When entanglement meets classical communications: Quantum teleportation for the quantum internet”. In: *IEEE Transactions on Communications* 68.6 (2020), pp. 3808–3833 (cit. on p. 20).

[48] Daniele Cuomo, Marcello Caleffi, and Angela Sara Cacciapuoti. “Towards a distributed quantum computing ecosystem”. In: *IET Quantum Communication* 1.1 (2020), pp. 3–8 (cit. on pp. 20, 21).

[49] Daniele Cuomo et al. “Optimized Compiler for Distributed Quantum Computing”. In: *ACM Transactions on Quantum Computing* (2023) (cit. on p. 20).

[50] Michael A Nielsen. “Quantum computation by measurement and quantum memory”. In: *Physics Letters A* 308.2-3 (2003), pp. 96–100 (cit. on p. 20).

[51] Stefan Hillmich, Alwin Zulehner, and Robert Wille. “Exploiting quantum teleportation in quantum circuit mapping”. In: *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE. 2021, pp. 792–797 (cit. on p. 21).

[52] Alwin Zulehner and Robert Wille. "Compiling $U(4)$ quantum circuits to IBM QX architectures". In: *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 2019, pp. 185–190 (cit. on p. 21).

[53] Bryan O'Gorman et al. "Generalized swap networks for near-term quantum computing". In: *arXiv preprint arXiv:1905.05118* (2019) (cit. on p. 21).

[54] Liam Madden and Andrea Simonetto. "Best approximate quantum compiling problems". In: *ACM Transactions on Quantum Computing* 3.2 (2022), pp. 1–29 (cit. on p. 21).

[55] Davide Ferrari et al. "Compiler Design for Distributed Quantum Computing". In: *IEEE Transactions on Quantum Engineering* 2 (2021), pp. 1–20 (cit. on p. 23).

[56] Pascal Van Hentenryck and Russell Bent. *Online stochastic combinatorial optimization*. The MIT Press, 2006 (cit. on p. 23).

[57] Pablo Andres-Martinez and Chris Heunen. "Automated distribution of quantum circuits via hypergraph partitioning". In: *Physical Review A* 100.3 (2019), p. 032308 (cit. on p. 25).

[58] Anocha Yimsiriwattana and Samuel J Lomonaco Jr. "Generalized GHZ states and distributed quantum computing". In: *arXiv preprint quant-ph/0402148* (2004) (cit. on p. 25).

Chapter 3

Quantum noise and how to handle it

3.1 Quantum noise

Assuming a good characterizing model for the noise affecting quantum information can be challenging. A highly generic one can be described as the evolution $\mathcal{N}$ on an d-dimensional system, where a state of interest σ evolves together with the environment in a state ρ [1]:

$$\mathcal{N}(\sigma) = \mathtt{Tr}_{\mathtt{env}}\big(\mathtt{U}(\sigma \otimes \rho)\mathtt{U}^\dagger\big).$$

Let $\{|e_k\rangle\}_k$ be an orthonormal basis for the environment. One can assume the environment system as being in the state $\rho = |e_0\rangle\langle e_0|$. This assumption is non-restrictive as the basis is generic and if the considered environment was not pure, one can always introduce an extra *reference* system to purify ρ. It follows that

$$\mathcal{N}(\sigma) = \sum_k \langle e_k|\,\mathtt{U}(\sigma \otimes |e_0\rangle\langle e_0|)\mathtt{U}^\dagger\,|e_k\rangle\,.$$

By defining $\mathtt{N}_k = \langle e_k|\,\mathtt{U}\,|e_0\rangle$, known as *Kraus* operator, $\mathcal{N}$ becomes

$$\mathcal{N}(\sigma) = \sum_k \mathtt{N}_k \sigma \mathtt{N}_k^\dagger. \tag{3.1}$$

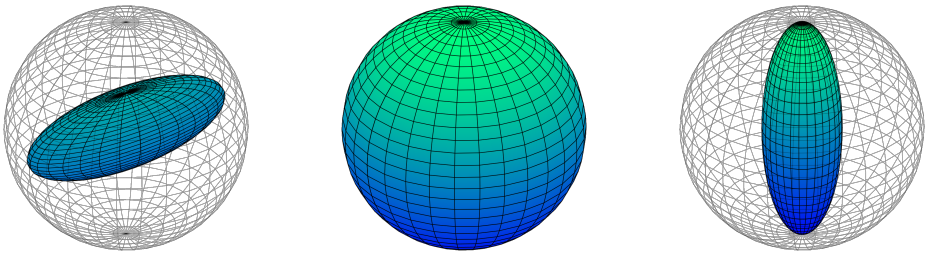
Decoherence is the most problematic noise affecting information. This can be represented with the Kraus formalism in terms of Pauli operators acting on each qubit independently [2, 3, 4], hence the set $\{\mathtt{N}_k\}_k$ as the form $\{\sqrt{\mathtt{p}_k}\,\mathtt{E}_k\}_k$, such that $\sum_k \mathtt{p}_k = 1$ and $\mathtt{E}_k$ belongs to the *Pauli group* $\mathbb{P} \equiv \{\pm\mathbb{1}, \pm\mathtt{i}\mathbb{1}, \pm\mathtt{X}, \pm\mathtt{iX}, \pm\mathtt{Y}, \pm\mathtt{iY}, \pm\mathtt{Z}, \pm\mathtt{iZ}\}^{\otimes d}$.

A further refinement is achievable. In fact, from an information perspective, the global phase has no relevance. Hence, by considering the "quotient" group[1]

$$\mathbb{E} \equiv \{\mathbb{1}, \mathtt{X}, \mathtt{Z}, \mathtt{XZ}\}^{\otimes d} \subset \mathbb{P}, \tag{3.2}$$

one can assume $\{\mathtt{E}_k\}_k \subset \mathbb{E}$.

A comparison among Pauli noise evolution for a 2-dimensional state is given in Fig. 3.1.



$\{\mathtt{X}, \mathbb{1}\}$ and $\mathtt{p} = 0.3$. $\{\mathbb{1}\}$, i.e., a noiseless evolution. $\{\mathtt{Z}, \mathbb{1}\}$ and $\mathtt{p} = 0.3$.

Figure 3.1: Example of Pauli noise for 2-dimensional Hilbert space in Bloch sphere representation.

[1] Being aware of the equivalence $\mathtt{XZ} \equiv -\mathtt{iY}$.

3.2 Estimating an evolution

For a given evolution $\mathcal{N}$, another useful representation of its action on a state σ is through its *Choi matrix* [5] $\varsigma_{\mathcal{N}}$. When the Kraus operators of $\mathcal{N}$ are known – i.e. $\{\sqrt{\mathtt{p}_k}\ \mathtt{E}_k\}_k$ – the Choi matrix is immediately defined as [6]:

$$\varsigma_{\mathcal{N}} = \sum_{n,m} \sqrt{\mathtt{p}_n \mathtt{p}_m} |\mathtt{E}_n\rangle\rangle\langle\langle \mathtt{E}_m|, \tag{3.3}$$

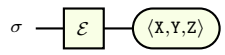
with $|\mathtt{E}\rangle\rangle$ being the *vectorization* of $\mathtt{E}$. When the evolution is *unitary* – i.e. $\mathcal{U}(\sigma) = \mathtt{U}\sigma\mathtt{U}^\dagger$ –, equation (3.3) simplifies to

$$\varsigma_{\mathcal{U}} = |\mathtt{U}\rangle\rangle\langle\langle \mathtt{U}|. \tag{3.4}$$

Working with Choi matrices allows to compute the fidelity of some unknown evolution w.r.t. a target one. For example, given a target unitary evolution $\mathcal{U}$ and an *experimental* evolution $\mathcal{E}$, both operating on d-dimensional Hilbert space. The following function is a proper fidelity for the two channels:

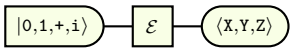
$$\mathfrak{f}(\mathcal{U}, \mathcal{E}) = \frac{1}{d} \cdot \mathtt{Tr}(\sqrt{\sqrt{\varsigma_{\mathcal{U}}}\sqrt{\varsigma_{\mathcal{E}}}})^2. \tag{3.5}$$

Estimating $\varsigma_{\mathcal{E}}$ is an expensive procedure based on a orthogonal set of measurements. A sufficient and common one is the Pauli group. In the circuit model we refer to this procedure as follows



The circuit above represent a *state tomography* – i.e., how $\mathcal{E}$ affects σ. It works by running k sets of experiments to obtain an estimator for $\{\mathtt{p}_k\}_k$.

Similarly, a *process tomography* starts from an orthogonal set of input states in order to fully characterize $\varsigma_{\mathcal{E}}$. In the circuit model we express such an estimation as



Tomography methods are intractable when considering high-dimensional systems.

3.3 Noise canceling through indefinite causal orders

Similarly to what we have done in Sec. 2.1.3, we now report the possible advantages coming from the indefinite causal order framework, applied to non-unitaries. Namely, noisy channels are superposed in order to increase the overall capacity [7, 8, 9, 10]

There are several ways to physically implement an indefinite order. Most of realizations are photonic-based [11, 12, 13, 14, 15], but it is not the only way. Indeed, within this Section we go over the work in Ref. [16]; presenting an implementation with a programmable technology, based on superconductors [17, 18].

The Indefinite Causal Orders for two evolution $\mathcal{N} \mapsto \{\mathtt{N}_n\}_n$ and $\mathcal{M} \mapsto \{\mathtt{M}_m\}_m$, is given by [19]:

$$\mathcal{S}(\sigma, \varkappa) = \sum_{nm} \mathtt{S}_{nm}(\sigma \otimes \varkappa)\mathtt{S}_{nm}^\dagger, \tag{3.6}$$

where $\varkappa$ is a *control state* and $\{\mathtt{S}_{nm}\}_{nm}$ denotes the set of Kraus operators such that

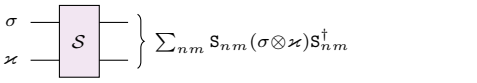
$$\mathtt{S}_{nm} = \mathtt{N}_n\mathtt{M}_m \otimes |0\rangle\langle 0| + \mathtt{M}_m\mathtt{N}_n \otimes |1\rangle\langle 1| \tag{3.7}$$

Therefore, under the assumption of errors in $\mathbb{E}$ – see Eq. 3.2 –, it is also true that

$$\mathtt{S}_{nm} = \sqrt{\mathtt{p}_n\mathtt{p}_m}\begin{bmatrix} \mathtt{E}_{nm} & \mathbf{0} \\ \mathbf{0} & \mathtt{E}_{mn} \end{bmatrix}. \tag{3.8}$$

As $\mathtt{N}_n\mathtt{M}_m = \sqrt{\mathtt{p}_n\mathtt{p}_m}\ \mathtt{E}_{nm}$ and $\mathtt{M}_m\mathtt{N}_n = \sqrt{\mathtt{p}_m\mathtt{p}_n}\ \mathtt{E}_{mn}$, with $\mathtt{E}_{nm}, \mathtt{E}_{mn}$ being in the Pauli group.

As discussed in Sec. 2.1, higher-ordered circuit frameworks are not presented within this thesis. An *oracle* is sufficient to our purpose.



Our aim here is to report our work published as in [16]. Namely, experiencing and evaluating the indefinite causal order within a *Noisy Intermediate-Scale Quantum* (NISQ) architecture [20], based on superconductors. NISQ architectures are widespread and they promise to be resources of practical interest in the next future. Furthermore, the design is likely to rapidly evolve, also by considering the Indefinite Causal Order as resource. Our hope is to enrich the knowledge on the capabilities of current quantum technologies, with the long-term goal of contributing to shape future architecture designs.

The experiment set-up is meant to witness the communication advantage, resulting from a specific case of the subject evolution. According to *quantum Shannon theory* [21], the capacity is a metric to quantify the ability for a noisy channel to convey quantum information, without destroying it. A channel with null capacity destroys the *coherence* of the quantum information, meaning that it is not possible to retrieve the original information. By superposing two or more null capacity channels, the result is a new channel with a not-null capacity, an interesting behaviour from a practical point of view [8, 9].

In the following we consider the *bit-flip* channel $\mathcal{X} \mapsto \{\mathtt{X}, \mathbb{1}\}$ and the *phase-flip* channel $\mathcal{Z} \mapsto \{\mathtt{Z}, \mathbb{1}\}$, having noise probability, respectively, $\mathtt{p}$ and $\mathtt{q}$. Ultimately, by preparing the control state to be $\varkappa = |+\rangle\langle +|$, it occurs the evolution represented in Figure 3.2 [8].

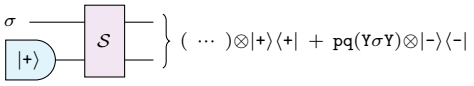


Figure 3.2: Superposition of causal orders for bit-flip and phase-flip.

According to the bottleneck inequality [21], given the composite operation $\mathcal{Z} \circ \mathcal{X}$ and let $\mathfrak{C}(\cdot)$ be the quantum capacity, the following upper-bound holds [8]:

$$\mathfrak{C}(\mathcal{Z} \circ \mathcal{X}) \leq 1 - \max\{\mathfrak{h}_2(\mathtt{p}), \mathfrak{h}_2(\mathtt{q})\}, \tag{3.9}$$

where $\mathfrak{h}_2$ denotes the binary Shannon entropy. Also, the same inequality holds for $\mathcal{X} \circ \mathcal{Z}$.

Whenever both $\mathtt{p}$ and $\mathtt{q}$ are equal to $1/2$, we have that both the configurations are characterized by a null capacity, i.e., $\mathfrak{C}(\mathcal{Z} \circ \mathcal{X}) = \mathfrak{C}(\mathcal{X} \circ \mathcal{Z}) = 0$.

Let us now superpose the two noises. Accordingly, with probability $\mathtt{pq}$ the output of circuit 3.2 is given by the second addendum, namely, $\big(\mathtt{Y}\sigma\mathtt{Y}\big) \otimes |\mathtt{-}\rangle\langle\mathtt{-}|$. As consequence the capacity of $\mathcal{S}$ is lower-bounded by $\mathfrak{C}(\mathcal{S}) \geq 1/4$, as shown in [9]. Specifically, σ occurs to pass through $\mathcal{Y}(\sigma) = \mathtt{Y}\sigma\mathtt{Y}$, coherently with control state being $|\mathtt{-}\rangle\langle\mathtt{-}|$. Therefore, it is possible to exploit the control state to gain a heralded unitary evolution $\mathcal{Y}$ via post-selection through the occurrence of $|\mathtt{-}\rangle\langle\mathtt{-}|$. Since $\mathcal{Y}$ is unitary, it is also reversible, therefore we can restore the information, gaining a perfect transmission of σ, i.e.,

$$\mathcal{Y} \circ \mathcal{Y}(\sigma) = \mathtt{YY}\sigma\mathtt{YY} = \sigma. \tag{3.10}$$

3.3.1 Quantum simulation

In this section we present our steps to realize $\mathcal{S}$ of Figure 3.2. We already discussed that the state of the art on quantum technologies doesn't allow any native implementation of the indefinite causal orders. Hence our goal is to witness the communication advantage through quantum simulation.

When considering a non-unitaries, as in the case of our interest, the overhead grows up w.r.t. superposing unitaries. The reason is that non-unitaries are naturally harder to engineer and need to be simulated as well. Specifically, for any Hilbert space $\mathbb{H}$, the corresponding set of density states lies in the convex map $\mathcal{C}(\mathbb{H})$.

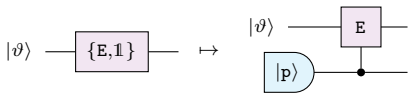


Figure 3.3: Stinespring dilation simulating a generic Pauli noise $\mathtt{E}$.

According to the *Stinespring dilation* [22], one can always associate to a non-unitary evolution $\mathcal{N} : \mathcal{C}(\mathbb{H}) \to \mathcal{C}(\mathbb{H})$ a unitary one, $\mathcal{A}_{\mathcal{N}}$, defined as follows:

$$\mathcal{A}_{\mathcal{N}} : \mathcal{C}(\mathbb{H}) \otimes \mathcal{C}(\mathbb{A}) \to \mathcal{C}(\mathbb{H}) \otimes \mathcal{C}(\mathbb{A}) \tag{3.11}$$

where $\mathcal{C}(\mathbb{A})$ is an auxiliary system with associated basis $\big\{\, |a_v\rangle\langle a_w| \,\big\}_{vw}$. Since $\mathcal{A}_{\mathcal{N}}$ is unitary, it has direct realization with the circuit algorithm. To simulate $\mathcal{N}$ from a realization of $\mathcal{A}_{\mathcal{N}}$, one need to *discard* the auxiliary system. In terms of operations, discarding the auxiliary system means applying a *partial trace* $\mathtt{Tr}_2 : \mathcal{C}(\mathbb{H}) \otimes \mathcal{C}(\mathbb{A}) \to \mathcal{C}(\mathbb{H})$. Specifically, for a generic state $\sigma = \sum_{ijvw} c_{ijvw}\big(\, |\vartheta_i\rangle\langle\vartheta_j| \otimes |a_v\rangle\langle a_w| \,\big)$, the partial trace outputs the following [23]:

$$\mathtt{Tr}_2(\sigma) = \sum_{ijvw} c_{ijvw}\, |\vartheta_i\rangle\langle\vartheta_j|\, \langle a_w|a_v\rangle. \tag{3.12}$$

Since $\mathbb{H}$ and $\mathbb{A}$ are taken to be generic systems, equation (3.12) has a direct generalization to the form $\mathtt{Tr}_{i_1,\ldots,i_k}$, tracing out subsystems indexed by $i_1, \ldots, i_k$.

In summary, we just outlined a method to realize an operation $\mathcal{N}$, involving two steps:

1. realizing the circuit $\mathcal{A}_{\mathcal{N}}$;
2. discarding the auxiliary system with a partial trace $\mathtt{Tr}_2$.

To our purpose, we apply this method, restricted to evolution $\{\mathtt{E}, \mathbb{1}\}$. In circuit representation this is shown in Figure 3.3. To superpose them we need an extra qubit, which encodes the control system. The final realization is shown in Figure 3.4.

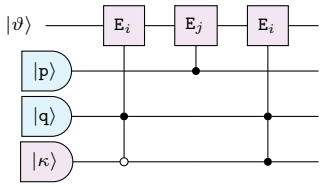


Figure 3.4: Circuit implementing an indefinite causal orders between two Pauli noise $\mathtt{E}_i, \mathtt{E}_j$.

3.3.2 Physical setting

Starting from circuit 3.4, we can do a step closer towards the real physical setting meant for us to witness the communication advantage of the indefinite causal order. To this aim we need to simulate the evolution from Figure 3.2, for the case of i.i.d. probabilities – i.e. $p = q = 1/2$. Figure 3.5 shows the physical setting we used for our experiments. Notice that we added specific state preparations and measurements for information and control qubits. This update express a process tomography settings, explained in detail in Section 3.2. Second and third qubits are not measured, rather, their final output is ignored, which naturally express trace out over those systems – i.e. $\mathtt{Tr}_{2,3}$.

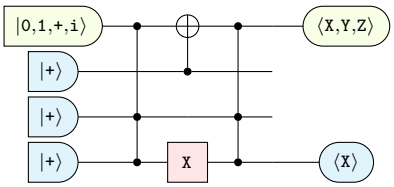


Figure 3.5: Circuit representation of the quantum experiment setting we used to witness the communication advantage.

At the end each run a post-selection occurs. Namely, coherently with our discussion of Section 3.3, we only keep those outputs where the control qubit results in the state $|-\rangle$. To the given set of outputs, we then apply a **classical bit-flip** in case the information qubit was subject to a measurement $\langle \mathtt{X} \rangle$ or $\langle \mathtt{Z} \rangle$. In fact, aware of the fact that the communication advantage comes from the post-processing of equation (3.10), which is a Pauli operation, this can be computed classically and has an effect only when measuring $\langle \mathtt{X} \rangle$ and $\langle \mathtt{Z} \rangle$.

Circuit decomposition and optimization

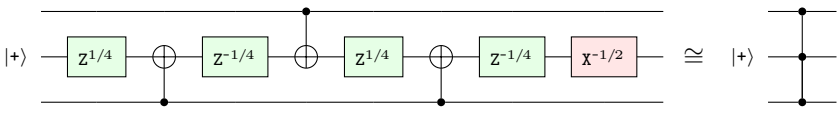


Figure 3.6: Whenever the second wire takes $|+\rangle$ as input, the gate r.h.s. is equivalent, up to a global phase, to the decomposition l.h.s.

As already discussed in Section 2.1.3, it is often the case that a real quantum technology doesn't supply natively a gate. For our setting of Figure 3.5, this is the case of the 3-qubits

gates, which have expensive decomposition [24].

To witness the communication advantage we managed to do some optimization on the first occurrence of such a gate. The rationale behind the optimization is that the gate decomposition could be simpler in case we have some knowledge of the input. We indeed know there is at least one qubit prepared in the state $|+\rangle$. This is enough to use the decomposition in Figure 3.6 instead of the standard one.

The final result is shown by plotting the characterization of the channel as a bloch sphere – See Figure 3.7. It represents the experimental characterization for the physical setting of Figure 3.5. Gray sphere represents the ideal sphere, corresponding to a set of pure states. The inside coloured sphere is the deformation induced by the imperfections caused by the employed technology, which in this case is the `santiago` processor provided by `IBM`[2], which has a *quantum volume* [25] of 16.

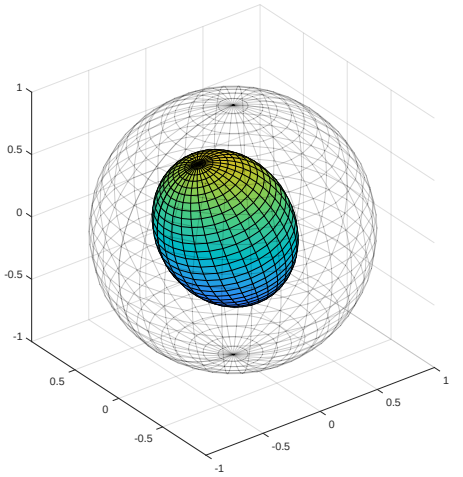


Figure 3.7: Bloch sphere representation of the experimental characterization for the physical setting of Figure 3.5.

3.4 Modeling faulty gates

According to the Pauli-Lindblad master equation [26, 4, 27, 28], a faulty operation can be modeled as $\mathcal{U} \circ \mathcal{N}$. In words, the model allows to think of a faulty operator as the composition of an ideal operator $\mathcal{U}$ preceeded by some Pauli-noise $\mathcal{N}$ – see Fig. 3.8.

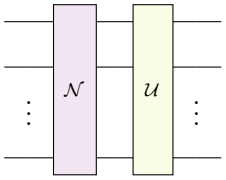


Figure 3.8: Faulty operation $\mathcal{U}$ expressed as the composition $\mathcal{U} \circ \mathcal{N}$.

For numerical evaluations, a common assumption is that a quantum evolution undergoes a depolarization [29] $\mathcal{D}$, which can be expressed as a mixture of Pauli errors:

$$\mathcal{D}(\sigma) = \sqrt{1 - \frac{3\lambda}{4}}\mathbb{1}\sigma\mathbb{1} + \sqrt{\frac{\lambda}{4}}(\mathtt{X}\sigma\mathtt{X} + \mathtt{Z}\sigma\mathtt{Z} + \mathtt{Y}\sigma\mathtt{Y}). \tag{3.13}$$

Starting from equation (3.13), it is possible to relate λ to a triplet of probabilities for the Pauli errors `X`, `Y` and `Z`. A method to do that is outlined in Ref. [30]. Furthermore, according to Ref. [2], the approximation

$$\mathcal{D}(\sigma) \approx (1-\mathtt{p})^2\sigma + \mathtt{p}(1-\mathtt{p})(\mathtt{X}\sigma\mathtt{X} + \mathtt{Z}\sigma\mathtt{Z}) + \mathtt{p}^2\mathtt{XZ}\sigma\mathtt{XZ}. \tag{3.14}$$

is a good model for decoherence and it applies independently to each qubit of the system – i.e., $\mathcal{D}^{\otimes d}$, for a d-dimensional system.

The joint result of equation (3.14) together with the Pauli-Lindblad assumption, allows to

[2]The processor has been retired at the time of writing.

invastigate circuit as a composition of ideal operators affected by some single qubit Pauli error that affect the logic of computation. Such errors propagate among the circuit coherently with the logical operators. As basic example, consider the operator $\wedge(\mathtt{X})$. According to equation (3.14), a Pauli error may precede its execution and propagates through the system as in Figure 3.9.

Despite the simplicity of the example, the two propagation rules shown in Figure 3.9 are complete, as $\wedge(\mathtt{X})$ is the only non-single qubit necessary for universal computation[3].

Figure 3.9: Propagation over $\wedge(\mathtt{X})$ operator.

As regards single-qubit operators, consider operators $\mathtt{Z}^{1/2}, \mathtt{X}^{1/2}$, necessary to generate the Clifford group as section 2.1.2. In such a case, orthogonal Pauli errors invert their logic, as in circuits of Figure 3.10

Figure 3.10: Pauli noise affecting orthogonal single-qubit operators.

3.5 Error correction and logical computing

3.5.1 Code functions

Similarly to what is done in classical information [31], quantum information can be protected through the introduction of an *error correction* scheme [3]. This starts with the definition of a *code* function Γ, which introduce redundancy to the system and exploit it to restore the original information, in case an error occurs. Formally, a code function $\Gamma_{k,m} : \mathbb{H}^{\otimes k} \to \mathbb{H}^{\otimes(k+m)}$, able to encode k qubits into $k+m$ with $m > 0$. The code is said to have ratio $\frac{k}{k+m}$. When defining a code, a *desideratum* is to achieve a high ratio. Another likewise important metric is the *distance* of a code. Specifically, $\Gamma_{k,m}$ creates a *codeword* space s.t. $|\mathtt{Im}(\Gamma_{k,m})| = 2^{k+m}$ where only 2^k elements are valid codewords. Generally speaking a highly sparse valid space allows for higher distance among valid codewords. On contrary a dense valid space makes fuzzier the identification of a couple of valid-invalid codewords. This issue is treated more formally in Section 3.8, while now it is just important to observe that distance and ratio are, by their nature, inversely proportional, creating a challenging trade-off to handle.

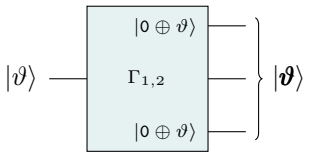


Figure 3.11: Example of code function $\Gamma_{1,2} : \mathbb{H} \to \mathbb{H}^{\otimes 3}$.

[3] We discussed this in detail in Section 2.1

Redundancy through entanglement

As an example, consider a code function which takes in input a generic single qubit $|\vartheta\rangle = \alpha|0\rangle + \beta|1\rangle$ and generates a logical qubit

$$|\boldsymbol{\vartheta}\rangle \equiv \alpha|0\rangle + \beta|1\rangle \equiv \alpha|000\rangle + \beta|111\rangle . \tag{3.15}$$

Such code can be implemented as in Fig. 3.11.

Coherently with the definition of code, the system has 3 qubits with $2^k = 2$ possible outcomes: $|000\rangle$ and $|111\rangle$. After the application of $\Gamma_{1,2}$, some errors become detectable. For example, assume that the bit-flip noise $\mathcal{X} \mapsto \{\mathbb{1}, \mathtt{X}\}$ affects each qubit, with i.i.d. error probability $\mathtt{p}$. One can perform a projection over the even space and the odd space for qubits 1 and 2, and then the same for qubits 2 and 3. This can be done by performing the *non-destructive measurement*[4]

$$\mathtt{Z}^{\otimes 2} = (|00\rangle\langle 00| + |11\rangle\langle 11|) - (|01\rangle\langle 01| + |10\rangle\langle 10|). \tag{3.16}$$

The eigenvalues are ± 1 and the detectable errors are represented in the table below.

Syndrome	Detection
$+1, +1$	$\mathtt{I}\otimes\mathtt{I}\otimes\mathtt{I}$
$-1, +1$	$\mathtt{X}\otimes\mathtt{I}\otimes\mathtt{I}$
$-1, -1$	$\mathtt{I}\otimes\mathtt{X}\otimes\mathtt{I}$
$+1, -1$	$\mathtt{I}\otimes\mathtt{I}\otimes\mathtt{X}$

Notice that $\mathtt{Z}^{\otimes 2}$ has no effect on $\alpha|000\rangle + \beta|111\rangle$, whatever the target qubits are. For this reason its measurement can be used to detect some error, without affecting the original state.

This error correction scheme works for $\mathtt{p} < \frac{1}{2}$. Without the scheme the minimum fidelity is

$$\mathtt{f} = \min_{\forall|\vartheta\rangle} \sqrt{\langle\vartheta|\,\mathcal{X}(|\vartheta\rangle\langle\vartheta|)\,|\vartheta\rangle} = \sqrt{1-\mathtt{p}}.$$

The probability of getting an error on the scheme is given by $\mathtt{p}_e = 3\mathtt{p}^2(1-\mathtt{p}) + \mathtt{p}^3$, so it is required that $\sqrt{1-\mathtt{p}_e} > \sqrt{1-\mathtt{p}}$, which happens when $\mathtt{p} < \frac{1}{2}$.

3.6 Stabilizer codes

Generally speaking, defining an efficient code is a hard task. Here, we review a fundamental family called *stabilizer codes*, which is particularly helpful as it can relate to linear codes coming from classical error correction.

A code function $\Gamma_{k,m}$ is a stabilizer code if its image is characterized by an abelian subgroup[5] $\mathbb{S} \subseteq \mathbb{E}$ as follows

$$\mathtt{Im}(\Gamma_{k,m}) \equiv \{|\boldsymbol{\vartheta}\rangle \;:\; \mathtt{S}\,|\boldsymbol{\vartheta}\rangle = |\boldsymbol{\vartheta}\rangle\,, \forall \mathtt{S} \in \mathbb{S}\}$$

$\mathbb{S}$ must be abelian in order to stabilize a non-trivial code. Consider the case $[\mathtt{S}_1, \mathtt{S}_2] = 1$, then $\mathtt{S}_1\mathtt{S}_2|\boldsymbol{\vartheta}\rangle = -\mathtt{S}_2\mathtt{S}_1|\boldsymbol{\vartheta}\rangle = -|\boldsymbol{\vartheta}\rangle$, but also $\mathtt{S}_1\mathtt{S}_2|\boldsymbol{\vartheta}\rangle = |\boldsymbol{\vartheta}\rangle$; hence $|\boldsymbol{\vartheta}\rangle = -|\boldsymbol{\vartheta}\rangle = 0$ and $\mathbb{S}$ stabilizes the trivial code $\mathtt{Im}(\Gamma_{k,m}) = \{0\}$.

[4] A possible implementation of non-destructive measurement is is given in Sec. 3.7.
[5] A group of which components commute one another. I.e. $[\mathtt{S}_i, \mathtt{S}_j] = 0$ holds for any $\mathtt{S}_i, \mathtt{S}_j \in \mathbb{S}$.

For 2^k possible outcomes, a stabilizer group $\mathbb{S}$ has 2^m elements and, since it is abelian, it can be specified by m generators $\{\mathtt{S}_1, \mathtt{S}_2, \ldots, \mathtt{S}_m\}$. The benefit of using generators is that to check whether a state vector is stabilized by $\mathbb{S}$ or not, one needs only to check it for the generators.

To see how the correction strategy works, consider an error operator $\mathtt{E} \in \mathbb{E}$. Let us analyse how $\mathtt{E}$ relates with a generator $\mathtt{S}_i$.

1. $\exists \mathtt{S}_i : \mathtt{E}\mathtt{S}_i = -\mathtt{S}_i\mathtt{E}$, then $\mathtt{S}_i\mathtt{E}\,|\boldsymbol{\vartheta}\rangle = -\mathtt{E}\mathtt{S}_i\,|\boldsymbol{\vartheta}\rangle = -\mathtt{E}\,|\boldsymbol{\vartheta}\rangle$. Therefore, $\mathtt{E}\,|\boldsymbol{\vartheta}\rangle$ is a -1 eigenvector of $\mathtt{S}_i$ and $\mathtt{E}$ can be detected by measuring $\mathtt{S}_i$.

2. Otherwise $\mathtt{E}\mathtt{S}_i = \mathtt{S}_i\mathtt{E}\ \forall \mathtt{S}_i$[6], then if $\mathtt{E} \in \mathbb{S}$, it clearly doesn't corrupt the state. So the problem arises when $\mathtt{E} \notin \mathbb{S}$, making $\mathtt{E}$ *undetectable*.

The set of undetectable errors is given by $C_{\mathbb{E}}(\mathbb{S}) \setminus \mathbb{S}$, where C is the *centralizer* function. Nevertheless, a noise $\mathcal{N}$ with some undetectable operators, may still be *correctable*. Formally, a generic noise $\mathcal{N}(\sigma) = \sum_k p_k \mathtt{E}_k \sigma \mathtt{E}_k$ is correctable if any two operators $\mathtt{E}_i, \mathtt{E}_j \in \{\mathtt{E}_k\}_k$, differ in syndrome or have the same syndrome but differ by a stabilizer, i.e.

$$\mathtt{E}_i\mathtt{E}_j \in \mathbb{S} \cup \big(\mathbb{E} \setminus C_{\mathbb{E}}(\mathbb{S})\big).$$

3.7 Relation with classical binary codes

Instead of expressing the error detection as the measurement operator $\mathtt{Z}^{\otimes 2}$, we can use the math coming from classical linear codes to define a quantum error correction scheme, e.g.

$$H = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}.$$

H can be used as parity-check matrix to detect and correct a bit flip occurring in one of the three physical qubits of $|\boldsymbol{\vartheta}\rangle$ – see Fig. 3.12. Clearly, this is true as long as ancillary qubits undergoes a negligible noise .

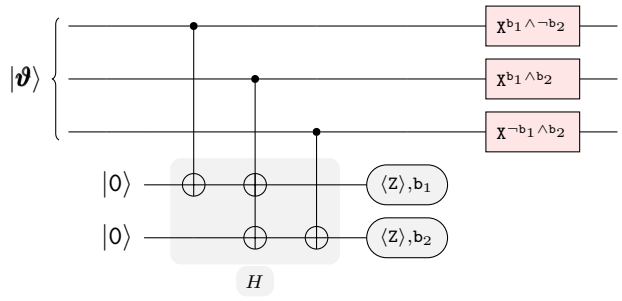


Figure 3.12: The gray coloring helps to visualize how the classical matrix representation H of the detection scheme is implemented by means of auxilliary qubits.

Ancillary qubits allows to abstract from the hardware, while the real implementation of this scheme very depends on what kind of measurements the quantum processor supplies[7]. Since $\mathtt{E} \in \mathbb{E}$, one can represent a generic error as a $2(k+m)$-dimensional binary vector $(e_{\mathtt{x}}|e_{\mathtt{z}})$,

[6] Because any couple of $\mathbb{E}$ either commutes or anti-commutes.

[7] See for example [32] for an experimental implementation, based on ancillary qubits to get non-destructive measurements.

such that if $e_{i,\mathtt{x}} = 1$, $\mathtt{E}$ has an $\mathtt{X}$ operator affecting the i-th qubit, or the identity otherwise. Symmetrically, for $e_{i,\mathtt{z}}$ the subject operator is $\mathtt{Z}$.

With the same criteria, let $(s_{i,\mathtt{x}}|s_{i,\mathtt{z}})$ be the binary vector representing a generator $\mathtt{S}_i$. Thus one can construct an $m \times 2(k+m)$ matrix H such that

$$H \equiv \begin{pmatrix} s_{1,\mathtt{x}}|s_{1,\mathtt{z}} \\ \vdots \\ s_{m,\mathtt{x}}|s_{m,\mathtt{z}} \end{pmatrix} \tag{3.17}$$

To check if the noise $\mathcal{N}$ is fully correctable, for any couple $(e_{i,\mathtt{x}}|e_{i,\mathtt{z}}), (e_{j,\mathtt{x}}|e_{j,\mathtt{z}})$, related to the operators $\mathtt{E}_i, \mathtt{E}_j$, the following holds:

$$H(e_{i,\mathtt{x}} + e_{j,\mathtt{x}}|e_{i,\mathtt{z}} + e_{j,\mathtt{z}})^{\intercal} \neq 0.$$

To get a stabilizer code starting from a classical linear code function[8] $\Sigma_{k,m} : \mathbb{F}_2^k \to \mathbb{F}_2^{(k+m)}$, with $0 < m < k$ and parity-check matrix H_Σ. The corresponding matrix for the quantum paradigm is given by

$$H_\Gamma \equiv \begin{pmatrix} H_\Sigma & \mathbf{0} \\ \mathbf{0} & H_\Sigma \end{pmatrix} \tag{3.18}$$

If the code is self-orthogonal – i.e. $\mathtt{Im}(\Sigma_{k,m})^\perp \subseteq \mathtt{Im}(\Sigma_{k,m})$ –, H_Γ relates to a stabilizer group $\mathbb{S}$ – in accordance with matrix (3.17) – for $k-m$ logical qubits and $k+m$ physical qubits[9]. Formally, following this procedure leads to a quantum code $\Gamma_{k-m,m}$ such that

$$\mathtt{Im}(\Gamma_{k-m,m}) \equiv \{|\boldsymbol{\vartheta}\rangle \ : \ \mathtt{S}\,|\boldsymbol{\vartheta}\rangle = |\boldsymbol{\vartheta}\rangle\,, \forall \mathtt{S} \in \mathbb{S}\}.$$

A slightly more general definition considers a couple of classical linear codes $\Sigma_{k_1,m_1}, \Sigma_{k_2,m_2}$. As long as $\mathtt{Im}(\Sigma_{k_2,m_2})^\perp \subseteq \mathtt{Im}(\Sigma_{k_1,m_1})$ and $k_1 + m_1 = k_2 + m_2$ hold, we can perform the parity check, in accordance with matrix (3.18). The result is a quantum code $\Gamma_{k,m}$ with $k = k_1 + k_2 - m_1 - m_2$ and $m = m_1 + m_2$. $\Gamma_{k,m}$ is called CSS code because of their creators Calderbank, Shor and Steane [33, 34].

3.8 Distance and bounds

3.8.1 Classical bounds

Consider a codeword set s.t. $|\mathtt{Im}(\Sigma_{k,m})| = 2^k$, defined in a 2^{k+m}-dimensional Hamming space. An error relates to a codeword u, creating a new word $\tilde{u} = u + e$. Generally speaking $\tilde{u}$ can relate to more than one u or e, making the definition of a good code a hard task. A good code should be able to relate any error e to one and one only codeword u. In this sense, such a code relates to each u a lattice sphere, centered in u with radius r. Each word $\tilde{u}$ in the sphere is such that $d(u,\tilde{u}) \leq r$ and $d(v,\tilde{u}) > r$ for any other codeword v. In order to avoid any overlap, the radius is upper-bounded by $r \leq \lfloor d/2 \rfloor$, where d is the minimum distance of the code. From

[8] $\mathbb{F}_2^n$ is a binary n-dimensional Hamming space, i.e. a vector of n binary values.
[9] Notice the loss in the ratio – i.e. $m/(k+m)$ – caused by "quantizing" a classical code.

this the *Hamming bound* follows:

$$\sum_{i=0}^{\lfloor d/2 \rfloor} \binom{k+m}{i} \leq 2^m$$

Where 2^m is the maximum number of words orthogonal to the code and $\binom{k+m}{i}$ is the number of errors involving i bits. Whenever the spheres saturate this bound, without creating any overlap, the code is said to be *perfect*[10].

The minimum possible distance to not run into overlaps is 3. To see that consider the basic case of two codewords `01`, and `11`. A parity-check with even result on the first codeword is indistinguishable from an odd result on the second codeword. It is possible to define a group of perfect codes[11] of distance $d = 3$. Namely, by setting $k = 2^h - h - 1$ and $m = h$ it results

$$\sum_{i=0}^{1} \binom{2^h - 1}{i} = 2^h$$

Notice how the ratio – i.e. $1 - \frac{h}{2^h - 1}$ – rapidly grows to 1 as h grows. More in general, for any high-dimensional code $\Sigma_{k,m}$ and minimum distance $d \propto k + m$, it is possible to prove [3] that the ratio $1 - \mathfrak{h}_2(d/(k+m))$ is achievable , where $\mathfrak{h}_2$ is the binary entropy.

3.8.2 Quantum bounds

For a given noise $\mathcal{N}(\sigma) = \sum_k p_k \mathtt{E}_k \sigma \mathtt{E}_k$, let $\{\mathtt{E}_{k_i}\}_i \subseteq \{\mathtt{E}_k\}_k$ be the set of undetectable errors. Then, the code distance is given by

$$d = \mathtt{min}\big\{\mathfrak{w}(\mathtt{E}) \ : \ \mathtt{E} \in \{\mathtt{E}_{k_i}\}_i\big\},$$

where $\mathfrak{w}$ is the weight function, counting the number of single-qubit components differing from the identity operator `I`. If the code $\mathtt{Im}(\Gamma_{k,m})$ is characterized by a stabilizer group $\mathbb{S}$, then

$$d = \mathtt{min}\big\{\mathfrak{w}(\mathtt{E}) \ : \ \mathtt{E} \in C_{\mathbb{E}}(\mathbb{S}) \setminus \mathbb{S}\big\}. \tag{3.19}$$

Symmetrically to classical codes, the Hamming bound related to a code $\Gamma_{k,m}$ is given by

$$\sum_{i=0}^{\lfloor d/2 \rfloor} 3^i \binom{k+m}{i} \leq 2^m.$$

The new factor 3^i expresses the possible error combinations from $\mathbb{E}$ involving any i qubits. Saturating the Hamming bound establishes a perfect code only if this is non-degenerate.

Consider, as an example, the following stabilizer set

$$\mathbb{S} = \langle \mathtt{X}_{1,4}\mathtt{Z}_{2,3}, \mathtt{X}_{2,5}\mathtt{Z}_{3,4}, \mathtt{X}_{1,3}\mathtt{Z}_{4,5}, \mathtt{X}_{2,4}\mathtt{Z}_{1,5} \rangle$$

$\mathbb{S}$ generates a code $\Gamma_{k,m}$ such that $m = \log|\mathbb{S}| = 4$ and, as the operators are defined over a 5-dimensional system, $k = 1$. Ultimately, according to (3.19), the distance is given by any $\mathtt{E}$

[10] Despite its name, the code is still unable to apply correction whenever $u_1 + e_1 = u_2 + e_2$ occurs, with u_1, u_2 being codewords and e_1, e_2 being errors.

[11] Known as Hamming codes.

operating on at least 3 qubits, coming from the centralizer group $C_{\mathbb{E}}(\mathbb{S})$, e.g. $\mathtt{E} = \mathtt{Z}_{1,2,4}\mathtt{X}_4$. To see that, consider any 1- or 2-qubits operator and check that it anti-commutes with some element from $\mathbb{S}$. Hence $d = \mathfrak{w}(\mathtt{E}) = 3$. The sphere coverage is

$$\sum_{i=0}^{1} 3^i \binom{5}{i} = 16 = 2^m$$

and, therefore, the code is perfect.

3.9 The role of stabilizers in computing

There is a tight relation between communication and computing scenarios, as regards error correction. Namely, in communication, noise affects information conveyed through a physical channel. Similarly, in computing, noise affects information during the life-time of an algorithm. Both scenarios run under the same noise model $\mathcal{N}(\sigma) = \sum_k p_k \mathtt{E}_k \sigma \mathtt{E}_k$. However, during computation, logical states demand for the definition of *logical operators*.

A unitary operator $\mathbf{U}$ is a logical operator for a code Γ stabilized by $\mathbb{S}$ if, for any logical state $|\boldsymbol{\vartheta}\rangle$ and any $\mathtt{S} \in \mathbb{S}$, the following holds:

$$\mathbf{U}\,|\boldsymbol{\vartheta}\rangle \in \mathtt{Im}(\Gamma), \quad \mathbf{U}\mathtt{S}\mathbf{U}^\dagger \in \mathbb{S}.$$

To prove that, it is sufficient to show that

$$\mathbf{U}\mathtt{S}\mathbf{U}^\dagger\mathbf{U}\,|\boldsymbol{\vartheta}\rangle = \mathbf{U}\mathtt{S}\,|\boldsymbol{\vartheta}\rangle = \mathbf{U}\,|\boldsymbol{\vartheta}\rangle$$

holds for all $\mathtt{S}$ in the generator of $\mathbb{S}$. Notice that $\mathbf{U}\mathtt{S}\mathbf{U}^\dagger$ stabilizes $\mathbf{U}\,|\boldsymbol{\vartheta}\rangle$.

Defining a stabilizer code from scratch

From the above emerges a general way to build a stabilizer code by defining together a code function $\bar{\Gamma}_{k,m}$ and its stabilizer group $\bar{\mathbb{S}}$. Formally consider the state $|\boldsymbol{\vartheta}\rangle \equiv |\vartheta\rangle \otimes |0\rangle^{\otimes m}$, which is (trivially) stabilized by the group $\mathbb{S} \equiv \langle \mathtt{Z}_{k+1}, \mathtt{Z}_{k+2}, \ldots, \mathtt{Z}_{k+m} \rangle$. Let $\mathtt{U}$ be a unitary operator mapping $\mathbb{S}$ to itself[12], then

$$\mathtt{Im}(\bar{\Gamma}_{k,m}) = \{\mathtt{U}\,|\boldsymbol{\vartheta}\rangle \;:\; \bar{\mathtt{S}}\mathtt{U}\,|\boldsymbol{\vartheta}\rangle = \mathtt{U}\,|\boldsymbol{\vartheta}\rangle\,, \forall \bar{\mathtt{S}} \in \bar{\mathbb{S}}\},$$

where $\bar{\mathbb{S}} \equiv \langle \bar{\mathtt{S}}_1, \bar{\mathtt{S}}_2, \ldots, \bar{\mathtt{S}}_m \rangle$ and $\bar{\mathtt{S}}_i \equiv \mathtt{U}\mathtt{Z}_{k+i}\mathtt{U}^\dagger$.

If the code is meant for computation, the only missing ingredient is the set of logical operators, which is

$$\left\{\mathbf{Z}_{\boldsymbol{i}}, \mathbf{X}_{\boldsymbol{i}} \;:\; \mathbf{Z}_{\boldsymbol{i}} \equiv \mathtt{U}\mathtt{Z}_i\mathtt{U}^\dagger,\ \mathbf{X}_{\boldsymbol{i}} \equiv \mathtt{U}\mathtt{X}_i\mathtt{U}^\dagger\right\}_{1 \le i \le k}.$$

Achieving fault-tolerant computing

Let Γ be any stabilizer code satisfying *self-duality*[13] and being *doubly-even*[14]. Then the Clifford group generators $\wedge(\mathtt{X}), \mathtt{X}^{1/2}$ and $\mathtt{Z}^{1/2}$ relate to *fault-tolerant* logical operators $\boldsymbol{\wedge(\mathbf{X})}, \mathbf{X}^{1/2}, \mathbf{Z}^{1/2}$.

[12] This may be any operator coming from the Clifford group, as it satisfies the closure over the Pauli group.
[13] $\mathtt{Im}(\Gamma) = \mathtt{Im}(\Gamma)^\perp$.
[14] Any codeword has Hamming weight divisible by 4.

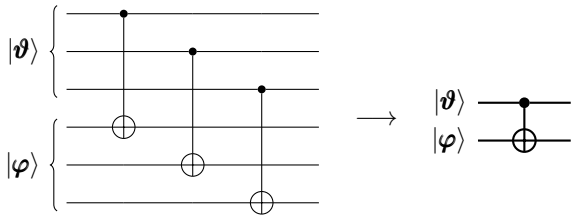


Figure 3.13: Transversal implementation of a logical $\wedge(\mathtt{X})$.

These operators can be claimed to be fault-tolerant because they admit (in principle) a so-called *transversal* implementation, which is very efficient in terms of circuit depth and error propagation. Figure 3.13 shows an example of transversal implementation of $\wedge(\mathtt{X})$ between two logical qubits $|\vartheta\rangle$ and $|\varphi\rangle$. Its efficiency is given by the fact the each physical operator acts on independent pairs of physical qubits. For the same reason, also the noise does not mix up among the physical qubits.

As regard fault-tolerance for the universal gate set $\mathbb{C}^+$ – see Sec. 2.1.2 – and especially for the non-Clifford operator $\mathtt{Z}^{1/4}$; there are some proposal for transversal implementation for the logical operator $\mathbf{Z^{1/4}}$, but these usually do not relate to any stabilizer group. Hence, in literature, two main branches of research emerged:

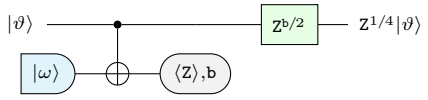


Figure 3.14: Example of $\mathtt{Z}^{1/4}$ gate *injection*.

- circuit manipulation with the goal of minimizing $\mathtt{Z}^{1/4}$ occurrences [35, 36];
- design of $\mathbf{Z^{1/4}}$ by means of *injection* protocols [37, 38, 39].

A basic example of $\mathtt{Z}^{1/4}$ *injection* is shown in Figure 3.14; this performs the injection by introducing one auxiliary qubit to the processor, prepared in the state

$$|\omega\rangle = \frac{1}{\sqrt{2}}(|0\rangle + e^{\frac{i\pi}{4}}|1\rangle). \tag{3.20}$$

Normal forms – see Sec. 2.1.2 – for universal circuits are also possible. An interesting result in this sense is available in Ref. [40], where authors showed that non-Clifford operators can be pushed to the beginning of the circuit.

3.10 Conclusion

With this chapter we covered many important topics to know when dealing with quantum computation. Especially considering the current state-of-the-art of quantum technologies, which are commonly referred as Noisy Intermediate-Scale Quantum (NISQ) architectures [20]. With the incoming Ch. 4, we will focus on optimizing circuits by means of *circuit compilation*, which preserves the circuit logic, while aiming to circuits more compliant to the hardware limitations. For practical reasons, we will consider circuit optimization without error correction schemes, as at the current stage of quantum technologies, these are to be considered at an early stage, where the gain promised by the theory struggle to be witnessed in real implementations. In fact, such schemes usually demands for more resources than the actual availability.

In accordance with our full-stack development proposed in Ch. 1, we expect error correction scheme implementations to show up at the *control system* level. Hence, a scheme will be in charge of providing a logical view of the physical resources. Because of such an organization, the compiler should not affect the logic on which the scheme relies on. This observation lead to think of new challenges specific to distributed architectures.

Before proceeding with the investigation of distributed compilers, we conclude the chapter by making some observation on the complication arising when trying and embedding error correction schemes within a distributed system.

3.10.1 Open challenges

As already shown in Ch. 2, telegates work by means of the generation and distribution of Bell states. Implementing a *transversal logical telegate* would result into a high parallelism of each required task. However, it is not straightforward to *import* the quantum error correction schemes into the distributed paradigm. To understand why, consider Fig. 3.15, it shows all the telegate steps but the last one – i.e. the post-processing. The bold representation refer to a transversal implementation, according to the formalism we introduced – see Fig. 3.13. Unfortunately, such an implementation is not logical in general, as the Bell states may affect the system which can end up *outside* the code scheme. For example, assuming:

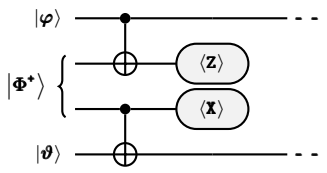


Figure 3.15: Transversal telegate, without post-processing.

$$\left|\boldsymbol{\Phi^+}\right\rangle \equiv \left|\Phi^+\right\rangle^{\otimes k+m}. \tag{3.21}$$

In such a case the Bell states are independent one another and because of this, there is a time lapse during which the system lays outside the code scheme and, for this reason, it is vulnerable to undetectable errors. The time lapse starts when applying the transversal operators $\wedge(\mathtt{X})$ and can only be restored by the post-processing, hoping that no error occurred in the meantime. In other words, the only detectable errors would be those caused by the post-processing, which, however, from a hardware perspective, results to be the most reliable step. Hence, this should not be considered as a practical way to proceed.

A possible way round is to generate and distribute a maximally entangled system, e.g., the generalized GHZ state:

$$\left|\boldsymbol{\Phi^+}\right\rangle \equiv \frac{1}{\sqrt{2}}\left(\left|0\right\rangle^{\otimes k+m} + \left|1\right\rangle^{\otimes k+m}\right). \tag{3.22}$$

Since the distributed system is now fully entangled, such a system can be used not only to perform the non-local operator $\boldsymbol{\wedge(\mathtt{X})}$, but it results that the measurement outcome can be used to detect errors, combining so the syndrome with the post-processing.

The proposed distributed encoded system – Eq. (3.22) – seems to be the solution to the problem. However, it has significant drawbacks from an hardware perspective. It should be already clear, this far, that generating a Bell state with high fidelity and within a reasonable time lapse is very challenging. This gets even harder, when thinking of higher degree systems, as the one of Eq. (3.22)[15].

[15]E.g. the smallest transversal code has $k = 1$ and $m = 6$ [41], resulting in the generation and distribution of

In conclusion, to model a fault-tolerant scheme for $\wedge(\mathbf{X})$, Eq. (3.22) may be an assumption too strong to be practical. A more clever approach would be to define the generation and distribution protocol as to be **part of the encoding**. Very little has been done in this direction experimentally. An inspiring set-up can be found in Ref. [42], where authors managed to create a Shor code [43] by means of photons paired in Bell states. As being photon-based, this may bring to future experimental settings where *stationary-flying hybrid systems* are considered.

References

[1] Michael A Nielsen and Isaac L Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2010 (cit. on p. 31).

[2] Eric Dennis et al. "Topological quantum memory". In: *Journal of Mathematical Physics* 43.9 (2002), pp. 4452–4505 (cit. on pp. 31, 36).

[3] Stefano Mancini and Andreas Winter. *A Quantum Leap in Information Theory*. World Scientific, 2020 (cit. on pp. 31, 37, 41).

[4] Ewout van den Berg et al. "Probabilistic error cancellation with sparse Pauli-Lindblad models on noisy quantum processors". In: *arXiv preprint arXiv:2201.09866* (2022) (cit. on pp. 31, 36).

[5] Man-Duen Choi. "Completely positive linear maps on complex matrices". In: *Linear algebra and its applications* 10.3 (1975), pp. 285–290 (cit. on p. 32).

[6] Jarn de Jong. "Fault-tolerant quantum computation: Implementation of a fault-tolerant SWAP operation on the IBM 5-qubit device". MA thesis. Delft University of Technology, 2019 (cit. on p. 32).

[7] Daniel Ebler, Sina Salek, and Giulio Chiribella. "Enhanced communication with the assistance of indefinite causal order". In: *Physical review letters* 120.12 (2018), p. 120502 (cit. on p. 32).

[8] Sina Salek, Daniel Ebler, and Giulio Chiribella. "Quantum communication in a superposition of causal orders." In: *arXiv preprint arXiv:1809.06655* (2018) (cit. on pp. 32, 33).

[9] Marcello Caleffi and Angela Sara Cacciapuoti. "Quantum Switch for the Quantum Internet: Noiseless Communications through Noisy Channels." In: *IEEE Journal on Selected Areas in Communications* 38.3 (2020), pp. 575–588 (cit. on pp. 32, 33, 34).

[10] Seid Koudia et al. "How deep the theory of quantum communications goes: Superadditivity, superactivation and causal activation". In: *IEEE Communications Surveys & Tutorials* (2022) (cit. on p. 32).

[11] Lorenzo M Procopio, Amir Moqanaki, Mateus Araújo, et al. "Experimental superposition of orders of quantum gates." In: *Nature Communications* 6.1 (2015), pp. 1–6 (cit. on p. 32).

[12] Giulia Rubino et al. "Experimental verification of an indefinite causal order." In: *Science Advances* 3.3 (2017), e1602589 (cit. on p. 32).

[13] Yu Guo et al. "Experimental transmission of quantum information using a superposition of causal orders". In: *Physical Review Letters* 124.3 (2020), p. 030502 (cit. on p. 32).

a maximally entangled system of 14 qubits.

[14] Joseph F Fitzsimons, Jonathan A Jones, and Vlatko Vedral. "Quantum correlations which imply causation." In: *Scientific Reports* 5.1 (2015), pp. 1–7 (cit. on p. 32).

[15] Kaumudibikash Goswami et al. "Indefinite causal order in a quantum switch". In: *Physical Review Letters* 121.9 (2018), p. 090503 (cit. on p. 32).

[16] Daniele Cuomo, Marcello Caleffi, and Angela Sara Cacciapuoti. "Experiencing the communication advantage of the Superposition of Causal Orders". In: *2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE. 2021, pp. 181–185 (cit. on pp. 32, 33).

[17] *IBM Quantum*. `https://quantum-computing.ibm.com/`. 2021 (cit. on p. 32).

[18] D. Castelvecchi. "IBM's Quantum Cloud Computer goes Commercial." In: *Nature* 543.7644 (Mar. 2017), p. 159 (cit. on p. 32).

[19] Giulio Chiribella et al. "Quantum computations without definite causal structure." In: *Physical Review A* 88.2 (2013), p. 022318 (cit. on p. 32).

[20] J. Preskill. "Quantum Computing in the NISQ era and beyond." In: *Quantum* 2.79 (2018) (cit. on pp. 33, 43).

[21] Mark M Wilde. *Quantum Information Theory*. Cambridge University Press, 2013 (cit. on p. 33).

[22] W. Forrest Stinespring. "Positive Functions on C*-Algebras". In: *Proceedings of the American Mathematical Society*. 6.2 (1955), pp. 211–216 (cit. on p. 34).

[23] Eleanor G Rieffel and Wolfgang H Polak. *Quantum computing: A gentle introduction*. MIT Press, 2011 (cit. on p. 34).

[24] Vivek V Shende and Igor L Markov. "On the CNOT-cost of TOFFOLI gates". In: *Quantum Information & Computation* 9.5 (2009), pp. 461–486 (cit. on p. 36).

[25] Andrew W Cross et al. "Validating quantum computers using randomized model circuits". In: *Physical Review A* 100.3 (2019), p. 032328 (cit. on p. 36).

[26] Heinz-Peter Breuer, Francesco Petruccione, et al. *The theory of open quantum systems*. Oxford University Press on Demand, 2002 (cit. on p. 36).

[27] Maximilian A Schlosshauer. *Decoherence: and the quantum-to-classical transition*. Springer Science & Business Media, 2007 (cit. on p. 36).

[28] Angela Sara Cacciapuoti and Marcello Caleffi. "Toward the quantum Internet: A directional-dependent noise model for quantum signal processing". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 7978–7982 (cit. on p. 36).

[29] Michael A Nielsen. "Quantum computation by measurement and quantum memory". In: *Physics Letters A* 308.2-3 (2003), pp. 96–100 (cit. on p. 36).

[30] Craig Gidney. *Decorrelated Depolarization*. `https://algassert.com/post/2001` (cit. on p. 36).

[31] D J Baylis. *Error Correcting Codes: A Mathematical Introduction*. Routledge, 2018 (cit. on p. 37).

[32] Julia Cramer et al. "Repeated quantum error correction on a continuously encoded qubit by real-time feedback". In: *Nature communications* 7.1 (2016), pp. 1–7 (cit. on p. 39).

[33] Robert Calderbank and Peter Shor. "Good quantum error-correcting codes exist". In: *Physical Review A* 54.2 (1996), p. 1098 (cit. on p. 40).

[34] Andrew Steane. "Multiple-particle interference and quantum error correction". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 452.1954 (1996), pp. 2551–2577 (cit. on p. 40).

[35] Peter Selinger. "Quantum circuits of T-depth one". In: *Physical Review A* 87.4 (2013), p. 042302 (cit. on p. 43).

[36] Matthew Amy, Dmitri Maslov, and Michele Mosca. "Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33.10 (2014), pp. 1476–1489 (cit. on p. 43).

[37] Mithuna Yoganathan, Richard Jozsa, and Sergii Strelchuk. "Quantum advantage of unitary Clifford circuits with magic state inputs". In: *Proceedings of the Royal Society A* 475.2225 (2019), p. 20180427 (cit. on p. 43).

[38] Ying Li. "A magic state's fidelity can be superior to the operations that created it". In: *New Journal of Physics* 17.2 (2015), p. 023037 (cit. on p. 43).

[39] Xinlan Zhou, Debbie W Leung, and Isaac L Chuang. "Methodology for quantum logic gate construction". In: *Physical Review A* 62.5 (2000), p. 052316 (cit. on p. 43).

[40] John van de Wetering. "Constructing quantum circuits with global gates". In: *New Journal of Physics* 23.4 (2021), p. 043015 (cit. on p. 43).

[41] "[[7, 1, 3]] Steane code". In: *The Error Correction Zoo*. Ed. by Victor V. Albert and Philippe Faist. 2022. URL: `https://errorcorrectionzoo.org/c/steane` (cit. on p. 44).

[42] Rui Zhang et al. "Loss-tolerant all-photonic quantum repeater with generalized Shor code". In: *Optica* 9.2 (2022), pp. 152–158 (cit. on p. 45).

[43] "[[9, 1, 3]] Shor code". In: *The Error Correction Zoo*. Ed. by Victor V. Albert and Philippe Faist. 2022. URL: `https://errorcorrectionzoo.org/c/shor_nine` (cit. on p. 45).

Chapter 4

Circuit compilers on distributed architectures

As outlined in Ch. 1, a full-stack development of a distributed system for quantum computation requires to be carefully engineered. The proposed stack allows a circuit designer to focus on the logic of its algorithm, without necessary consider all the issues coming from the physical infrastructure that will take care of computing it.

In this chapter we consider the layer interfacing with the algorithm (written in circuit model). This takes care of *optimizing the circuit*, adapting it to the constraints given by the underlying layers. Such a layer is commonly referred as *compiler* and the corresponding optimization problem is called the *compilation problem*. This is a generally tough task to solve, even on single processor, and for which an NP-hardness proof is available [1]. An ever growing literature arises with a variety of proposals for local computation [2, 9, 10, 11, 12, 13, 14, 15, 16, 3, 4, 5, 6, 7, 8] and for distributed computation [17, 19, 20, 21, 22, 23, 24, 25, 26, 18].

Even if quantum processors are already available, distributed architectures are at an early stage and must be discussed from several perspectives. A key concept is that of *telegates* as the fundamental inter-processor operations [27, 28, 29]. We already discussed in Ch. 2 how telegates works, but we report below the main steps.

Each telegate can be decomposed into several tasks, that we group as follows: (i) the generation and distribution of entangled states among different processors, (ii) local operations and (iii) classical communications. These tasks makes the telegate an expensive resource, in terms of running time and/or fidelity. As a consequence, they have critical impact on the performance of the overall computation. In contrast to such a limit, telegates offer remarkable opportunities of parallelization. In fact, much circuit manipulation is possible to keep computation independent from telegate's tasks. Therefore, we aim to model an optimization problem that embeds such opportunities.

Fig. 4.1 provides the reader with a conceptual map concerning the main scientific contributions.

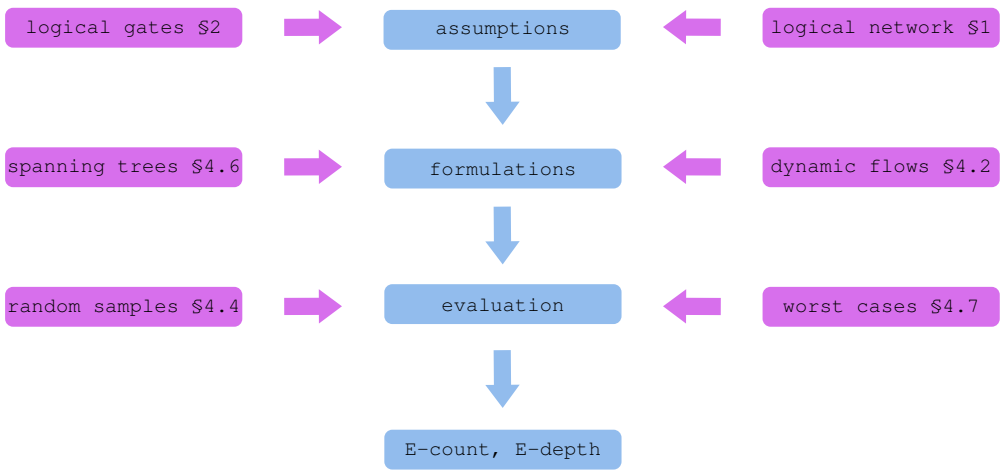


Figure 4.1: Overview of the main contribution; blue blocks denote the main steps in the problem modeling, scanned by blue arrows. The violet blocks are the main ingredients to the entry blue blocks.

4.1 Mathematical modeling

According to the current trend on quantum technologies – reported in Ch. 1 – we can give a mathematical description. Formally, let $\mathcal{N} = (V, P, F)$ be a network triple representing the architecture. $V = Q \cup C$ is a set of nodes describing qubits, therefore it is the disjoint union of computation qubits $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$ and communication qubits $C = \{c_1, c_2, \ldots, c_{|C|}\}$. We can represent n processors by partitioning V into $P = \{P_1, P_2, \ldots, P_n\}$. Therefore, a sub-set P_i characterizes a processor as its set of qubits/nodes.

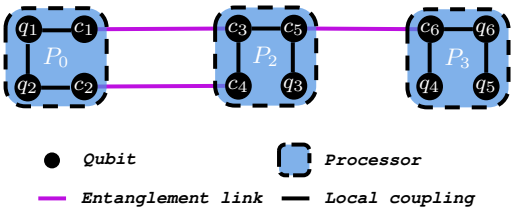


Figure 4.2: Toy distributed quantum architecture with 3 processors.

$F = L \cup R$ is as a set of undirected edges. L represents the local couplings, therefore

$$L \subseteq \bigcup_i P_i \times P_i.$$

Notice that there is no particular assumption on connectivity nor cardinality within processors. This keeps the treating hardware-independent and it allows for heterogeneous architectures.

R represents entanglement links. Since entanglement links connect only communication qubits, we introduce, for each processor, a set of those qubits only; i.e., $C_i = C \cap P_i$. Therefore, we have

$$R \subseteq \bigcup_{i,j \,:\, i \neq j} C_i \times C_j.$$

Fig. 4.2 shows an exemplary architecture, with three processors in P, six computation qubits in Q, six communication qubits in C, three entanglement links in R and ten local couplings in L.

Concerning minimal assumptions, we only care about architectures actually able to perform any operation. This translated into a simple connection assumption.

4.2 Distributed quantum circuit compilation problem

Usually, in the literature dealing with compiler design [15, 16, 13, 21], a circuit is encoded as a set of *layers*. Formally, a layer is a set ℓ of independent operators, meaning that each operator in ℓ acts on a different collection of qubits. A circuit is an enumeration of layers $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_{|\mathcal{L}|}\}$, where the cardinality is also commonly referred as circuit *depth*.

A quantum programmer writes a logical circuit, abstracting from the real architecture and assuming that qubits are fully connected, i.e., any couple of qubits can perform a $\wedge(\mathtt{X})$ operation – defined as in Eq. (2.1) – directly. Such an abstraction holds also when stepping to distributed architectures.

However, NISQ architectures do not always provide full coupling. This is especially true in the distributed scenarios. As a consequence, there must be an interface – namely, a compiler – able to map an abstract circuit to an equivalent one, but meeting the available coupling. In general, such a mapping implies overhead in terms of circuit depth. Therefore, finding a

mapping with minimum depth overhead is an optimization problem. We refer to it as the *quantum circuit compilation* problem (QCC), which is proved to be NP-hard [1]. Its version on distributed architectures, which we refer to as the *distributed quantum circuit compilation* problem (DQCC), is likely to be at least as hard as QCC. In fact, while in QCC we deal with local connectivity restrictions, in DQCC local connectivity stands alongside with remote connectivity – i.e., the entanglement links –, which is less dense than the local one[1]. Furthermore, performing a remote operation is much more time consuming than a local operation. Just consider that a remote operation relies on communication of both quantum and classical information.

The above reasons make telegates the bottleneck in distributed computing. Therefore, they are worth of dedicated analysis to minimize their impact.

4.2.1 Objective function

To optimize a circuit, the first thing we need to do is choosing an objective function to rate the expected performance of a circuit. A common approach is to evaluate only those operators which are somehow a bottleneck to computation. Considering the universal gate set $\mathbb{C}^+$ – defined in Sec. 2.1.2, in the context of fault-tolerant quantum computing [30], the bottleneck is the $\mathtt{Z}^{1/4}$ operator [31, 32], since error correction protocols are designed for the Clifford group $\mathbb{C}$. Conversely, on current NISQ technologies, the bottleneck lies in the interaction between qubits – as for the case of $\wedge(\mathtt{X})$. The relevant metric can either be the number of occurrences of some operator O, namely the O-count, or the number of layers containing O at least once, namely the O-depth. To rate a compiled circuit on distributed architectures, we do something along the lines of this approach. Specifically, the bottleneck are the non-local $\wedge(\mathtt{X})$ (and $\wedge(\mathtt{Z})$) operators, each of which implies one occurrence of entanglement generation and distribution stage. We refer to such a stage as the E operator. Therefore, we will rate a circuit by means of its E-depth and E-count.

4.2.2 Modeling the time domain

It should be clear that E has central interest in our treating. In fact, we are also going to model the time by scanning it as E occurs. Specifically, notice that link generations among different couples of qubits are independent. For this reason we assume that all the possible links generate simultaneously and, as soon as all the states are measured, a new round of simultaneous generations begins.

Clearly, after that a measurement generates a boolean b, there is at least one post-processing operator that need to wait for that boolean to arrive. Generally speaking, the longer the path the more time b takes to reach its destination. We need to account for that by a proper model. To this aim, we do some observations.

Remark. *Consider a generic single-qubit unitary operator U. The time required to perform $U^{\mathtt{b}}$ is given by the sum of the travel time of b plus the time to perform U. However, the traveling of b is independent from computation and any operation preceding $U^{\mathtt{b}}$ can run. Hence, we compactly refer to the post-processing waiting-time as $\Delta_{U^{\mathtt{b}}}$. A second observation is that the travel of b is also independent by entanglement link creations, which we assume to take time Δ_E. It is, therefore, also reasonable to assume $\Delta_{U^{\mathtt{b}}} \lesssim \Delta_E$ because of the following observation: even if*

[1] Because the more communication qubits there are, the less computing resources are available.

`b` need to cover a longer distance than the one covered by `E`, `b` relies on classical technologies, which are way more efficient[2] than entanglement generation and distribution protocols. For this reason, in our treating we neglect $\Delta_{\mathtt{b}}$, since it happens in parallel with $\Delta_{\mathtt{E}}$. Furthermore, in Secs. 4.4 and 4.6 we will focus on groups of circuits where all the post-processing operations are fully separated from the quantum computation.

Stemming from this, we can model the time domain as a discrete set of steps $\tau \in \{1, 2, \ldots, d\}$, where d is an unknown time horizon, which is also the `E`-depth. At the beginning of each time step τ, the whole set of entanglement links is available for telegates. Notice that most of the local operators are expected to run during the creation of the links. Because we relate them to the following inequality

$$\Delta_{\mathtt{E}} \gg \Delta_{\wedge(\mathtt{U})}, \Delta_{\mathtt{U}}, \tag{4.1}$$

where `U` is a single-qubit unitary operator. Therefore, since `E` is independent from local operators, we can always attempt to run these while `E` is running – and also while classical bits `b` are traveling, as explained in Sec. 2.2.4.

4.2.3 Modeling the distributed architecture

In light of the above observations, it is reasonable and convenient to consider the whole processor as a network node, and define a function c that provides the number of available links between two processors. Specifically, we first formalized a distributed architecture as the network graph $\mathcal{N} = (V, P, F)$ introduced in subsection 4.1; this step was important to understand the interior behavior of remote operations from a qubit perspective. However, now it is useful to restate it to a more compact encoding, which highlights the main bottleneck of a distributed quantum architecture, the entanglement links. Formally speaking, we will consider a *quotient graph* of $\mathcal{N}$.

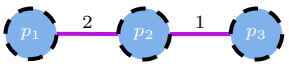


Figure 4.3: Quotient graph derived from Fig. 4.2. The processors become the nodes, the entanglement links between a couple of processors gather into one edge, with capacity equal to the number of original links [33].

To not further weigh down the formalism, we re-model the instance, by considering as main nodes, the processors, corresponding to an enumeration for the partition P, i.e., $P = \{p_1, p_2, \ldots, p_n\}$. All the entanglement links, connecting the same couple of processors, now collapse to an only edge with integer capacity c, describing how many parallel entanglement links the two processors supplies[3]. We refer to this sets of edges as

$$E \subseteq \bigcup_{i,j \,:\, i \neq j} p_i \times p_j.$$

Hence, the new undirected graph is $\mathcal{Q} = (P, E, c)$. With this reformulation a remote operation will refer to a *control processor* and a *target processors* – e.g., $\wedge(\mathtt{X})_{u,v}$ with $p_u, p_v \in P$.

In Fig. 4.3 we show the quotient graph related to the toy architecture of Fig. 4.2.

[2]The design of a distributed quantum architecture can easily adapt to satisfy requirements coming from assumptions on classical technologies, since these are very advanced.

[3]Notice that this has no impact when the quantum processor is based has a full-connected topology, as in the case of Ion-traps.

4.2.4 Single layer formulation

Consider a basic circuit expressed as the singleton $\mathcal{L} = \{\ell\}$. Assume that in ℓ there occur k $\wedge(\mathtt{X})$ – or $\wedge(\mathtt{Z})$ – operators. From a logical perspective, all the k operators can run in parallel, by definition of layer. In other words, if the architecture connectivity had infinite capacity – i.e., $c(e) = \infty$, $\forall e \in E$ – we could run $\mathcal{L}$ with $\mathtt{E}$-depth 1, that is optimal. As the capacity values decrease, the optimal $\mathtt{E}$-depth value grows, up to $\mathtt{E}$-depth k in the worst-case.

Let us formulate an optimization problem for the single-layer case – we will introduce a generalization to any circuit in subsection 4.2.5. Specifically, the *quickest multi-commodity flow* [34] wraps this basic scenario.

In brief, the goal is to find a flow over time which satisfy the constraints imposed by a set of so-called commodities, which are going to represent the $\wedge(\mathtt{X})$ of a quantum circuit. The less time the flow takes, the better. To formalize this problem, one can directly model an objective function that evaluates a flow by the time it takes. This is an approach employed in Ref. [35], but for single commodity. Alternatively, authors in Ref. [34] propose to start from a formulation of the *multi-commodity flow* problem over time $\mathtt{MCF}_d$, where d is a given *time horizon*[4], namely a maximal number of time steps in which the flow is constrained. We prefer this latter way because dynamic flows like $\mathtt{MCF}_d$ has been deeply studied since long time ago [36, 37]. Furthermore, even if this approach has an important drawback, explained at the end of this sub-section, it does not apply to our scenario.

Commodities

To formulate $\mathtt{MCF}_d$, first, we enumerate the occurrences of two-qubit remote operators in $\mathcal{L}$ as a set of commodities $[k] = \{1, 2, \ldots, k\}$. A set of couples source-sink nodes associates to the commodities. To do that, let $\mathbf{s} = (s_1, s_2, \ldots s_k)$ and $\mathbf{t} = (t_1, t_2, \ldots t_k)$ be two vectors induced by the operators $\wedge(\mathtt{X})$ – or $\wedge(\mathtt{Z})$ – in $\mathcal{L}$[5] such that,

$$\wedge(\mathtt{X})_{s_i,t_i} \in \ell \iff \exists i \in [k] \ : \ p_{s_i}, p_{t_i} \in P.$$

Namely, p_{s_i} (p_{t_i}) is the processor where the control (target) qubit of operation i occurs.

Decision variables

The decision variables of the optimization problem are the time-dependent functions $f_{e,i}(\tau) \in \{0, 1\}$, indicating the flow on edge $e \in E$ dedicated to operation $i \in [k]$ at time τ. The function has a binary co-domain because an operation i uses at most one entanglement link.

Constraints

As usual, the first constraint we introduce is the *flow conservation* constraint. Formally, $\forall i \in [k]$, $\forall \tau \in [d]$ and $\forall p_j \in P \setminus \{p_{s_i}, p_{t_i}\}$ the following holds:

$$\sum_{e \in \delta^-(p_j)} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_j)} f_{e,i}(\tau) = 0 \tag{4.2}$$

[4]The choice of using letter d should highlight that the time horizon is going to be the $\mathtt{E}$-depth.
[5]We need to use vector notation to admit repetitions.

where $\delta^-, \delta^+ : P \to E$ are the standard functions outputting the set of entering and exiting edges of the input node, respectively.

Since a flow $f_{e,i}(\tau) = 1$ identifies the usage of an entanglement link in e to perform i, we need to guarantee that the flow going through intermediate links of a path does not stop there. Conversely, whenever an end point of the path occurs in the control or target processor – i.e., p_{s_i} or p_{t_i} –, the *operation demand* – or *commodity demand* – constraint holds instead of the conservation constraint. Namely, $\forall i \in [k]$, this can be written as:

$$\sum_{e \in \delta^-(p_{s_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_{s_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) = -1 \tag{4.3}$$

$$\sum_{e \in \delta^-(p_{t_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_{t_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) = +1 \tag{4.4}$$

The above constraint explicitly requests that a flow dedicate to i reaches its target p_{t_i}, without exiting. Symmetrically, it leaves its control processor p_{s_i} without returning.

Notice that constraint (4.2) forces the operation demand to be satisfied within a single time-step.

The last constraint ensures that, at any time step, the number of operations does not exceed the entanglement resources. Hence, $\forall e \in E$ and $\forall \tau \in [d]$, we introduce a *capacity bound*:

$$\sum_{i \in [k]} f_{e,i}(\tau) \leq c(e) \tag{4.5}$$

Ultimately, the objective function is the total flow $f = \sum_{e \in E} \sum_{i \in [k]} \sum_{\tau} f_{e,i}(\tau)$.

By gathering the above equations, we obtain the Integer Linear Programming formulation (4.6), which models $\texttt{MCF}_d$. A flow f perfectly matches a set of entanglement paths used by the telegates.

$$\begin{aligned}
\text{minimize} \quad & f = \sum_{e \in E} \sum_{i \in [k]} \sum_{\tau \in [d]} f_{e,i}(\tau) \\
\text{subject to} \quad & \sum_{e \in \delta^-(p_j)} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_j)} f_{e,i}(\tau) = 0 && \forall i \in [k], \forall \tau \in [d], \forall p_j \in P \setminus \{p_{s_i}, p_{t_i}\}, \\
& \sum_{e \in \delta^-(p_{s_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_{s_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) = -1 && \forall i \in [k], \\
& \sum_{e \in \delta^-(p_{t_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) - \sum_{e \in \delta^+(p_{t_i})} \sum_{\tau \in [d]} f_{e,i}(\tau) = +1 && \forall i \in [k], \\
& \sum_{i \in [k]} f_{e,i}(\tau) \leq c(e) && \forall e \in E, \forall \tau \in [d]
\end{aligned} \tag{4.6}$$

Notice that solutions with cycles are in general feasible, but are senseless in our scenario. By expressing the problem as a minimization of f, a solver will avoid any cycle and will try to use as few entanglement links as possible. Once defined a solver for $\texttt{MCF}_d$, we just need to use it as proposed in Ref. [34], namely the solver occurs as sub-routine within a binary research on the minimum time where a feasible solution exists. Since the research space is over time, the algorithm is, in general, pseudo-logarithmic.

Specifically to our case, we already know that the worst solution is where all the operations run in sequence – i.e., E-depth equal to the amount k of telegates. Therefore, the time horizon is upper-bounded by k and the binary search has $\log k$ calls to the sub-routine. Algorithm 1 shows the steps. Notice that it makes use of an undetermined solver for $\texttt{MCF}_d$. Since we are facing an NP-hard problem, this means that a real implementation would generally look for sub-optimal solutions.

Algorithm 1: Quickest multi-commodity flow

Input: $\mathcal{Q}, [k]$
Output: d
$L \leftarrow 1, R \leftarrow k$
while $L \leq R$ **do**
  $\bar{d} \leftarrow \lfloor \frac{L+R}{2} \rfloor$
  $S \leftarrow \texttt{MCF}_{\bar{d}}(\mathcal{Q}, [k])$
  if S **is feasible then**
    $d \leftarrow \bar{d}$
    $R \leftarrow \bar{d} - 1$
  else
    $L \leftarrow \bar{d} + 1$

Unfortunately, standard $\texttt{MCF}_d$ cannot catch the whole features of DQCC when $\mathcal{L} = \{\ell_1, \ell_2, \dots, \ell_{|\mathcal{L}|}\}$; we need to consider that operations in $[k]$ are somehow related each other by a logic determined by $\mathcal{L}$. Hence, in Sec. 4.2.5 we will model such relations by introducing extra constraints.

Transformation to direct graph

Since the literature dealing with multi-commodity flows usually assume a direct graph, we here report a mapping method from an undirected graph to an equivalent one with direct edges. This would bring just a constant overhead in the space, while it would not affect any approximation factor which a solver would rely on. Fig. 4.4 comes from [38]. It is a fast approach to map an undirected multi-commodity flow problem to a directed one. Specifically, for each couple of nodes p_i, p_j connected by an edge with capacity c, one have to introduce two extra nodes, say $p_{i'}, p_{j'}$ and connect them with the direct edge $(p_{i'}, p_{j'})$ of capacity c. The last step is creating directed cycles of infinite capacity, where the only bottleneck is c.

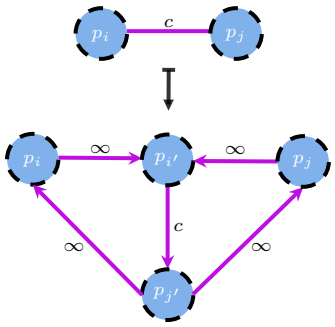


Figure 4.4: Mapping from an undirected graph to a directed one working for any multi-commodity flow problem. The transformation undergoes with a constant overhead in the number of nodes and edges.

4.2.5 Any layer formulation

As mentioned, the formulation we just gave is not enough to model the DQCC problem to any $\mathcal{L} = \{\ell_1, \ell_2, \dots, \ell_{|\mathcal{L}|}\}$, because a circuit generally follows a logic which is related on the order of occurrence given by $\mathcal{L}$. Therefore, even if it might happen that two operations could run in any order, in general this is not true. One needs to define an order relation which is consistent with the logic of the circuit. From an optimization point of view, a critical matter is to choose an order relation that either wraps most of the good solutions or is prone to optimization algorithms. For this reason and for the sake of clarity, we here refer to a generic, irreflexive,

order relation $\prec$ defined over $[k]$, without giving it a unique definition. Formally, for any $i, j \in [k]$, $j \prec i$ means that to run i we need to ensure that j already ran. Starting from $\prec$, we can define a constraint to add to formulation (4.6). Namely, $\forall i \in [k], \forall e \in \delta^-(p_{t_i})$ the following holds:

$$f_{e,i}(\tau) \leq \min_{j \prec i} \sum_{\bar{\tau}<\tau} f_{e,j}(\bar{\tau}) \tag{4.7}$$

The right part of the inequality is a value in $\{0, 1\}$ and takes value 1 only if all the operations logically preceding i already ran. Notice that constraint (4.7) is linear, as it takes the minimum value among linear functions, and it can be easily mapped to a set of independent constraints $f_{e,i}(\bar{\tau}) \leq \sum_{\bar{\tau}<\tau} f_{e,j}(\bar{\tau})$, $\forall j : j \prec i$.

The formulation now models `DQCC`. But we will refine inequality (4.7) to get a better solution space – see Sec. 4.3.

4.3 Increasing the parallelism

As before, from an optimization point of view, we are interested in considering as many good solutions as possible. To this aim, we propose an interesting approach which should enlarge the space of good solutions. Specifically, we notice that even if two operations $i, j \in [k]$ are such that $i \prec j$, this does not necessarily mean that they must run at different time steps. They, indeed, may run at the same time step and still respecting the logic imposed by $\prec$.

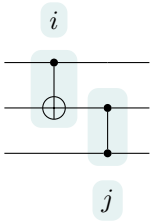


Figure 4.5: Telegates in logical conflict.

Consider the example from Fig. 4.5. Since operations i and j operates over a common qubit, they are in logical conflict. Hence, it is reasonable to think that $i \prec j$ should hold. However, when considering i and j in their extended form – i.e., where communication qubits are explicit – we notice that their logical conflict does not map over all the operations involved. As Fig. 4.6 shows, the left part of the equivalence is a naive implementation of i followed by j, where the extended form completely inherits the logical conflict. Instead, the right part of the equivalence is way more efficient and it is still an implementation of circuit of Fig. 4.5. As consequence, even if i and j are in logical conflict, they can run at the same time step. We refer to this property as *quasi-parallelism*. For this reason we introduce a new binary relation between operations in $[k]$, which we refer to with the intuitive symbol $\shortparallel$.

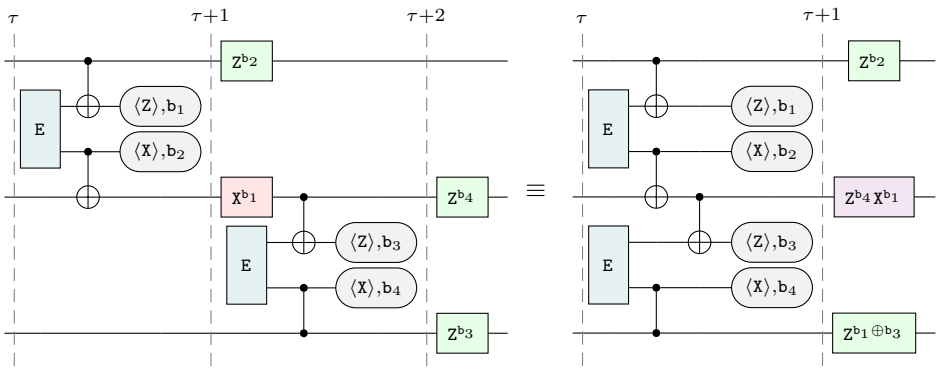


Figure 4.6: Example of how to achieve quasi-parallelism for two telegates in logical conflict.

As before, we do not give here a unique definition of $\shortparallel$. Specifically, for any $i, j \in [k]$, we write $i \shortparallel j$ to mean that operations i and j can run at the same time step, but we did not fix a criterion to establish when $\shortparallel$ holds. Clearly, operations $i, j \in [k]$ which can run in full parallelism, are a special case of quasi-parallelism and $i \shortparallel j$ holds.

We can now split the constraint (4.7), by discriminating between operations which can run in quasi-parallelism and the ones which cannot. Formally, $\forall i \in [k], \forall e \in \delta^-(p_{t_i})$ we introduce two new constraints

$$f_{e,i}(\tau) \leq \min_{j \prec i \wedge j \nshortparallel i} \sum_{\bar{\tau}<\tau} f_{e,j}(\bar{\tau}) \tag{4.8}$$

$$f_{e,i}(\tau) \leq \min_{j \prec i \wedge j \shortparallel i} \sum_{\bar{\tau}\leq\tau} f_{e,j}(\bar{\tau}) \tag{4.9}$$

To sum up, we propose (4.10) as Integer Linear Programming formulation of the DQCC problem. $\mathcal{C}$ is the set of constraints coming from the standard MCF formulation given in (4.6). In what follows we propose a characterization for relation $\shortparallel$.

$$\begin{aligned}
\text{minimize} \quad & f = \sum_{e\in E} \sum_{i\in[k]} \sum_{\tau\in[d]} f_{e,i}(\tau) \\
\text{subject to} \quad & \mathcal{C}, \\
& f_{e,i}(\tau) \leq \min_{j \prec i \wedge j \nshortparallel i} \sum_{\bar{\tau}<\tau} f_{e,j}(\bar{\tau}) \quad \forall i \in [k], \forall e \in \delta^-(p_{t_i}), \forall \tau \in [d], \\
& f_{e,i}(\tau) \leq \min_{j \prec i \wedge j \shortparallel i} \sum_{\bar{\tau}\leq\tau} f_{e,j}(\bar{\tau}) \quad \forall i \in [k], \forall e \in \delta^-(p_{t_i}), \forall \tau \in [d]
\end{aligned} \tag{4.10}$$

For an exemplary characterization of $\shortparallel$, please refer to [33] where we introduced a computationally efficient predicate, which states whenever two remote operations i, j can run in quasi-parallelism. We opted to not report this part and rather focus on ways of getting rid of the extra constraints – see Sec. 4.4. To this aim we investigate different groups of algorithms and what normal forms they offer. This allows to give a much wider perspective on what kind of shapes the compiler can get in input.

4.4 The role of Clifford group in distributed architectures

In our model, we showed that by postponing the Pauli-corrections, we get the combined advantage of (i) parallelizing remote operations and (ii) delaying the correction, which amortizes the impact of the traveling time that a boolean value takes to reach its destination(s). An ideal result would be to push all the corrections to the end of the circuit. In fact, as already discussed in Sec. 2.2, if the corrections reach the end of the circuit, they could be replaced by classical computation. Driven by this goal, we now investigate the properties of quantum circuits to find when such a condition is satisfied, starting from the Clifford group $\mathbb{C}$.

The interest in the Clifford group derives from the fact that it covers a wide spectrum of circuits and, to be universal, it needs only one extra operator. In Sec. 2.1.2, we referred to such an extension as the group $\mathbb{C}^+ \equiv \langle \wedge(\mathtt{X}), \mathtt{X}^{1/2}, \mathtt{Z}^{1/2}, \mathtt{Z}^{1/4} \rangle$. For this reason, it makes sense to represent an arbitrary circuit in terms of a Clifford circuit plus as little $\mathtt{Z}^{1/4}$ as possible. This is, indeed, an active branch of research [31, 32].

4.4.1 Circuit normal forms and implications on the post-processing

As said at the beginning of Sec. 2.1.2, important benefits could be achieved by postponing the post-processing to the end of the circuit, where they can be computed classically. An attempt in this direction is available in Ref. [40], where authors delay Pauli operations together with non-Pauli ones. Instead, our approach is to show that the result can always be achieved on the Clifford group, by relying on the **normal forms** [39, 41, 42, 43, 44, 45]. Such a form results particularly useful for distributed computing and, more in general, for *measurement-based* computation. It has been shown in [39] that any Clifford gate acting on a Pauli state can be represented in the normal form depicted in Fig. 4.7. This normal form is of practical interest as it can be obtained starting from any Clifford circuit, which is in general not in normal form. Such a result comes from the employment of a *ZX-calculus* reasoner, e.g. [46]. ZX-calculus [39, 47] is a graphical language, arisen as an optimizer for quantum circuits, that translates a quantum circuit into a ZX-*diagram*. The main difference between the diagram and the original circuit is that the former works with ZX-*rules*, which serve as a reasoning tool to smartly generate a new circuit, equivalent to the original one. ZX-calculus was introduced in the literature in 2007 [48], with the main objective of minimizing a circuit gate-depth, and its potentiality is still being explored, raising increasing interest for its versatility. In fact, we use it here to perform architecture-compliant optimization.

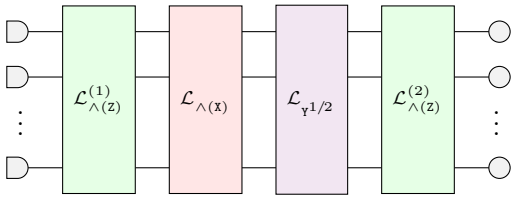


Figure 4.7: Normal form coming from the ZX-rules applied in Ref. [39].

Coming back to Fig. 4.7, we use the circuit symbol ⊐— to express a generic Pauli state preparation. Similarly, the symbol —◯ expresses a generic Pauli measurement. $\mathcal{L}_{\mathtt{O}}$ is a set of layers where only the O operator occurs. For example $\mathcal{L}_{\wedge(\mathtt{Z})}$ encodes a circuit composed by $\wedge(\mathtt{Z})$ operators.

For the subject normal form we need to define only a few *pushing rules*. As regard the circuit $\mathcal{L}_{\wedge(\mathtt{X})}$, the following rules always apply:

- $\wedge(\mathtt{X}) \cdot \mathtt{X}^{\mathtt{b}} \otimes \mathbb{1} \equiv \mathtt{X}^{\mathtt{b}} \otimes \mathtt{X}^{\mathtt{b}} \cdot \wedge(\mathtt{X})$
- $\wedge(\mathtt{X}) \cdot \mathbb{1} \otimes \mathtt{Z}^{\mathtt{b}} \equiv \mathtt{Z}^{\mathtt{b}} \otimes \mathtt{Z}^{\mathtt{b}} \cdot \wedge(\mathtt{X})$
- $\wedge(\mathtt{X}) \cdot \mathbb{1} \otimes \mathtt{X}^{\mathtt{b}} \equiv \mathbb{1} \otimes \mathtt{X}^{\mathtt{b}} \cdot \wedge(\mathtt{X})$
- $\wedge(\mathtt{X}) \cdot \mathtt{Z}^{\mathtt{b}} \otimes \mathbb{1} \equiv \mathtt{Z}^{\mathtt{b}} \otimes \mathbb{1} \cdot \wedge(\mathtt{X})$

Similarly, for $\mathcal{L}_{\wedge(\mathtt{Z})}$ circuits, we can use the following rules:

- $\wedge(\mathtt{Z}) \cdot \mathtt{X}^{\mathtt{b}} \otimes \mathbb{1} \equiv \mathtt{X}^{\mathtt{b}} \otimes \mathtt{Z}^{\mathtt{b}} \cdot \wedge(\mathtt{Z})$
- $\wedge(\mathtt{Z}) \cdot \mathtt{Z}^{\mathtt{b}} \otimes \mathbb{1} \equiv \mathtt{Z}^{\mathtt{b}} \otimes \mathbb{1} \cdot \wedge(\mathtt{Z})$

Finally, the last single layer circuit $\mathcal{L}_{\mathtt{Y}^{1/2}}$ can be handled as follows:

- $\mathtt{Y}^{1/2} \cdot \mathtt{X}^{\mathtt{b}} \cong \mathtt{Z}^{\mathtt{b}} \cdot \mathtt{Y}^{1/2}$
- $\mathtt{Y}^{1/2} \cdot \mathtt{Z}^{\mathtt{b}} \equiv \mathtt{X}^{\mathtt{b}} \cdot \mathtt{Y}^{1/2}$

Remark. *By means of the above rules, all the post-processing operations can be pushed forward, up to end of the circuit and they can be computed efficiently by a classical computer. Furthermore, since no post-processing occurs during quantum computation, the entanglement path length has negligible impact to the running-time (thanks to the non-locality of the operations).*

The normal form suggests that the problem can be separated into three parts, corresponding to $\mathcal{L}^{(1)}_{\wedge(\mathtt{Z})}$, $\mathcal{L}_{\wedge(\mathtt{X})}$ and $\mathcal{L}^{(2)}_{\wedge(\mathtt{Z})}$. For two of them – i.e., $\mathcal{L}^{(1)}_{\wedge(\mathtt{Z})}$ and $\mathcal{L}^{(2)}_{\wedge(\mathtt{Z})}$ – the order relation is trivial (as all $\wedge(\mathtt{Z})$ commute), and therefore we can use any quickest multi-commodity flow solver to get a feasible compilation. On the contrary, the optimal characterization of the order relation for $\mathcal{L}_{\wedge(\mathtt{X})}$ is more complex. Indeed, a set of relations with minimal size may not be the best characterization from a practical point of view, if many of the relations involve remote qubits. The topic of optimal $\wedge(\mathtt{X})$ order relations deserves a dedicated analysis. Hence we first evaluate what we achieved so far, by evaluating our model on $\mathcal{L}_{\wedge(\mathtt{Z})}$, while we investigate later $\mathcal{L}_{\wedge(\mathtt{X})}$ circuits. – see Sec 4.6.

Let us emphasize the importance of $\mathcal{L}_{\wedge(\mathtt{Z})}$ circuits, by pointing out some facts from Ref. [43]. The authors therein introduce the *Boolean degrees of freedom* as a way to count how many different algorithms can be implemented with a class of gates, and show that a generic $\mathcal{L}_{\wedge(\mathtt{Z})}$ "has roughly half the number of the degrees of freedom" compared to a generic $\mathcal{L}_{\wedge(\mathtt{X})}$, and roughly a quarter compared to the Clifford group.

$\mathcal{L}_{\wedge(\mathtt{Z})}$ circuits represent also an important group for efficient syntheses [49, 50, 51, 52], where these are used to maximize the efficiencies by means of parity check sequences.

We validate our compiler performance by solving $\mathcal{L}_{\wedge(\mathtt{Z})}$ circuits on different architectures in Sec. 4.6. So, being able to exploit normal forms to isolate two highly expressive blocks $\mathcal{L}^{(1)}_{\wedge(\mathtt{Z})}$ and $\mathcal{L}^{(2)}_{\wedge(\mathtt{Z})}$ that can be compiled without recurring to order relations, is a very relevant result.

4.4.2 Analysis on the upper-bounds and future perspective

There is a fair doubt arising from the employment of normal forms for compilation: do we know the overhead cause by mapping any Clifford circuit to some normal form? If yes, is it reasonable?

The answer is positive to both questions. By working with normal forms, we are not only able to work with a circuit with known shape, but we can also upper-bound the overhead for the number of introduced operations. Depending on whether or not ancillae are considered, the system get more complex in terms of space or run-time. In Ref. [50], authors treat both cases, and prove linear upper-bounds.

Normal forms unlock also better opportunities from an hardware perspective. Specifically, dealing with well defined circuit allows to extend the gate set with more practical operators, as the $\wedge(\mathtt{X}^{\otimes m})$ introduced in Ch. 2. In Sec. 4.6 we refer to these gates as *fan-in* (and *fan-out*) gates. From a hardware perspective, these are also commonly referred as *global gates*, as they may act over a large m simultaneously [53, 54, 55, 56]. Citing [53]: "It has been suggested that polynomial or exponential speedups can be obtained with global [gates]".

Other results in terms of overhead can be found in Ref. [57], where authors proved that any n-qubit Clifford circuit can be synthesised to $4n - 6$ global gates and any n-qubit circuit with $\dot{n}$ non-Clifford gates can be synthesised with no more than $2\dot{n} + \mathcal{O}(n/\log n)$ global gates.

Ultimately, our choice to employ the normal form of Fig. 4.7 has several benefits, besides the ones we already discussed:

- It is practical, as the open-source `pyzx` [46] provides the tools to perform the mapping.
- It is efficient, as the `pyzx` engine works to minimize the number of two-qubit gates.
- It has a good shape, as $\mathcal{L}_{\wedge(\mathtt{Z})}$ circuits are generally easier than $\mathcal{L}_{\wedge(\mathtt{X})}$ ones.

Let us make a final remark on `ZX`-calculus. We introduced it in the context of the Clifford group, but it is designed to work more broadly with any circuit [58, 59, 60, 61]. Therefore, we aim to expand our analysis in future works, by investigating normal forms for universal circuits. An interesting result in this direction is available in Ref. [62], where authors split a universal circuit into the following three steps:

1. the system is prepared in a *non-Clifford state* [57], this involves auxiliary qubits which will do the work of injecting non-Clifford phases;
2. an $\mathcal{L}_{\wedge(\mathtt{X})}$ circuit;
3. a measurement-based sequence of Clifford operations – which can still be treated with `ZX`-calculus [63].

4.5 Commuting circuits compiler

As distributed quantum architectures are still at an early stage, it is hard to predict with confidence what kind of connectivity and resources they will supply. It is therefore of interest to investigate on what kind of topology a distributed architecture should have. For this reason we now report a compiler for $\mathcal{L}_{\wedge(\mathtt{Z})}$ and use it to evaluate the performance given by different topologies. Thanks to this evaluation, we could choose the topology most performing with our experiments[6].

Here we evaluate the *rectangle lattice* topology – see Figs. 4.8a, 4.8c – and comparing it with a *hexagon lattice* topology – see Fig 4.8b. We therefore verify the compiler performance for both the lattices in terms of:

- solution quality;
- robustness to scale-up.

We conclude the comparison with the possible implications of the results.

4.5.1 An approximation-based implementation

We already discussed in Sec. 4.2.4 how to tackle `DQCC` as a particular case of quickest multi-commodity flow. In this way we managed to reduce the problem on the resolution of one or more static instances of `MCF`. In Refs. [65, 66] it has been shown that whenever each commodity is a source (or a target) for any other node, then solving it through `LP`-relaxation outputs an optimal solution to `MCF`. This result can be of interest when treating *fully entangling circuits*.

[6]The same topology is also employed in [64], where authors deal with unreliable optical links to create entanglement and dynamically choose a multi-path solution in order to maximise the entanglement success-rate.

To keep the compiler more general, we opted to investigate algorithms with approximation boundary guaranteed [67, 68, 69]. Specifically, we implemented the pseudo-code outlined in Ref. [70]. This is followed by a proof on the approximation quality for the case of capacity $c = 1$ and $c > 1$. We focus on the case $c = 1$, but it can be extended to $c > 1$.

Algorithm 2: Iterative compiler

Input: $\mathcal{Q}, [k]$
Output: d
$S \leftarrow [k]$
$d \leftarrow 0$
while $S \neq \varnothing$ **do**
  $\bar{S} \leftarrow \texttt{MCF}(\mathcal{Q}, S)$
  $S \leftarrow S \setminus \bar{S}$
  $d \leftarrow d + 1$

By using our formalism, the approximation algorithm aims to run as many non-local operators – i.e. satisfying commodities demand – as possible. A computed solution is a sub-set $S \subseteq [k]$. The optimal solution is $S^* \subseteq [k]$ and $|S| \leq |S^*|$. It follows the (optimal) approximation boundary [68, 70]:

$$|S| \geq \frac{|S^*|}{\mathcal{O}(\sqrt{m})}, \; m = |E| \tag{4.11}$$

Notice that the solution quality is inversely proportional to the number of entanglement links. It means that we cannot estimate an optimal solution to the `DQCC`, as for a given time horizon, this affects the quality of the solution space. Furthermore, the time-expansion increases the number of edges and so does the distance $|S^*| - |S|$. Ultimately, even if the allocated space by the time-expansion grows at most linearly with the number of non-local operations – see Sec. 4.2.4 –, this can seriously affect the performance when such an amount is very big[7]. On contrary, it is possible to keep the time-expansion *abstract* and compiling iteratively as many operations as possible at each time-step. This method is detailed in Algorithm 2. Notice that each iteration guarantees the boundary of equation (4.11) and, above all, since the instance decreases in size, the distance $|S^*| - |S|$ tends to decrease as well.

4.5.2 Set-up

To compare the compiler performance on different topologies, we make use of a *generator factor* $g \in \mathbb{N} \setminus \{0\}$. The number of nodes and edges of each lattice will be expressed as a function of g. Because the two lattices differ by definition, it is not trivial to settle a fair comparison. To do that, we first generate[8] a sample of hexagon lattices $\mathcal{H}$ such that

$$|P| = 1/2 \cdot g^2 + 3g + \mathcal{O}(1), \; |E| = 3/4 \cdot g^2 + 7/2 \cdot g + \mathcal{O}(1). \tag{4.12}$$

We compare $\mathcal{H}$ with two rectangle lattices, say $\mathcal{R}_\blacktriangledown$ and $\mathcal{R}_\blacktriangle$, that have sizes respectively lower and higher than $\mathcal{H}$ for each g – see Fig. 4.8. Hence, $\mathcal{R}_\blacktriangledown$ is such that

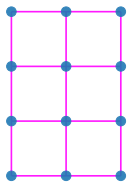
$$|P| = 1/4 \cdot g^2 + 3/2 \cdot g + \mathcal{O}(1), \; |E| = 1/2 \cdot g^2 + 2g + \mathcal{O}(1). \tag{4.13}$$

while $\mathcal{R}_\blacktriangle$ is such that

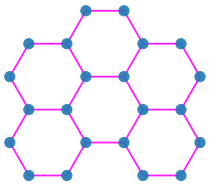
$$|P| = 2g^2 + 2g, \; |E| = g^2 + 2g + \mathcal{O}(1). \tag{4.14}$$

[7] Better upper-bounds for the worst-case solution should be investigated.
[8] With the help of the `networkx` library [71].

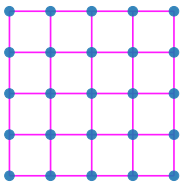
We show in the next subsection that $\mathcal{R}_{\blacktriangle}$ and $\mathcal{R}_{\blacktriangledown}$ perform better than $\mathcal{H}$ in terms of resulting E-depth. This implies that the rectangle lattice is a better design for distributed quantum computers, assuming that our compiler performs equally well on different topologies.



(a) rectangle lattice $\mathcal{R}_{\blacktriangledown}$.



(b) Hexagon lattice $\mathcal{H}$.



(c) rectangle lattice $\mathcal{R}_{\blacktriangle}$.

Figure 4.8: Example of lattices used for the experimental evaluation; they all come from generator $g = 4$.

Since we use Algorithm 2, capacities are assumed to be 1. We already pointed out that such an algorithm can be extended to the case $c > 1$.

Notice that different node degrees imply different assumptions on the processor units P_i. The hexagon lattice has node degree upper-bounded by 3 and lower-bounded by 2, which means that P_i has 2 to 3 communication qubits. Similarly, the rectangle lattice has degree upper-bounded by 4. Hence, the communication qubits per unit are 2 to 4. Since our focus here is on distributed compilation, we will assume that P_i has 1 computation qubit. This is especially reasonable when considering that real implementation of distributed architecture may use most of their local resources as *auxiliary qubits*, meant to keep the computation fault-tolerant.

For the numerical evaluation we use a generating vector $\mathbf{g} = (1, 2, \ldots, 11)$. Hence, when the generator is fixed to 11, the size of $\mathcal{H}$ reaches $|P| = 96$ and $|E| = 131$, $\mathcal{R}_{\blacktriangledown}$ reaches $|P| = 49$ and $|E| = 84$, while $\mathcal{R}_{\blacktriangledown}$ reaches $|P| = 144$ and $|E| = 264$.

We generate three samples classified by their size (or number of occurring operators). Each sample is composed by 10 random circuits in order to average the results. The size of the samples are 256, 512 and 1024.

4.5.3 Architecture evaluation

To evaluate the results we used the `matlab` environment [72]. The employed architecture is a MacBook Air - M1, 2020, 8GB RAM.

The first result – shown in Figs. 4.9a, 4.9b and 4.9c – is a comparison on the solution quality, a.k.a. the E-depth.

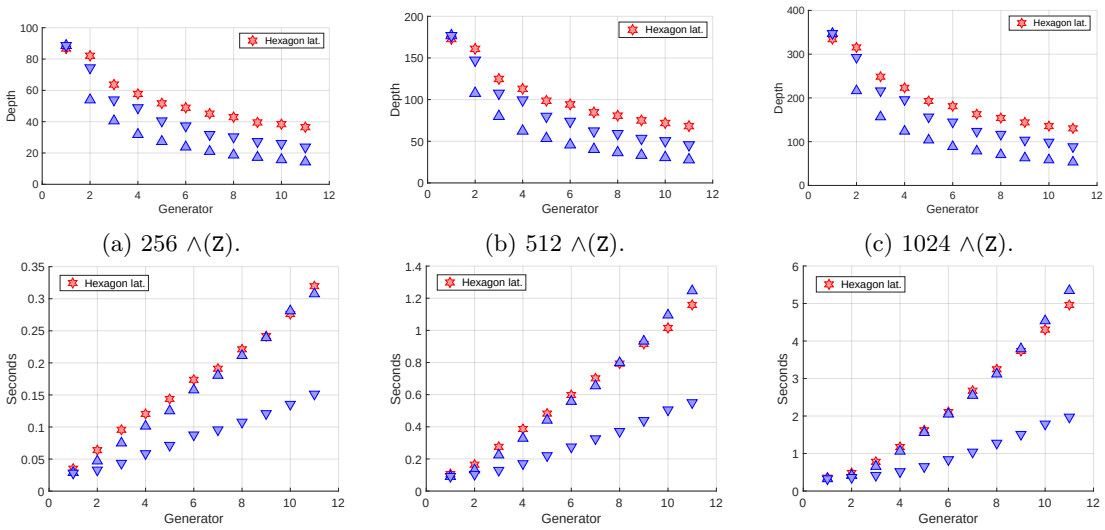
As anticipated, the plots show that a rectangle lattice gives better solutions, for any problem size. We can relate this behavior to the *ratio edges-to-nodes*. Formally, let $\mathfrak{r}_{\mathcal{Q}} = \frac{|E|}{|P|}$ be such a ratio for a graph $\mathcal{Q}$. Then it results that rectangle lattices have ratio:

$$\lim_{g \to \infty} \mathfrak{r}_{\mathcal{R}} = 2. \tag{4.15}$$

Instead hexagon lattices have a lower ratio:

$$\lim_{g \to \infty} \mathfrak{r}_{\mathcal{H}} = 3/2. \tag{4.16}$$

This suggests that the bigger the ratio, the better the solutions. The plots also show that



(a) 256 $\wedge(\mathtt{Z})$. (b) 512 $\wedge(\mathtt{Z})$. (c) 1024 $\wedge(\mathtt{Z})$.

(d) 256 $\wedge(\mathtt{Z})$. (e) 512 $\wedge(\mathtt{Z})$. (f) 1024 $\wedge(\mathtt{Z})$.

Figure 4.9: Quality and time scale comparison.

the depth achieved by the different lattices may be ruled by the same polynomial function (up to some constant factor). This is in line with the intuition that a more connected topology allows for shorter depth. Furthermore, we already mentioned in Sec. 4.5.1 that, even if the approximation algorithm depends on the edges size, this is called as a subroutine that performs better and better at each iteration. All this may mean that the compiler has a convergence to an optimal depth. On contrary, if the compiler was affected by the number of edges, the functions should swap at some point, but we never observed such phenomenon.

To conclude our evaluation, we took the average times for each sample. The results are shown in Figs. 4.9d, 4.9e and 4.9f. Differently from what we got in the solution quality evaluation – where we noticed a similar behaviour for each architecture – the time-scale gives new perspectives in the lattices comparison. In fact, $\mathcal{H}$ and $\mathcal{R}_{\blacktriangle}$ seems to need approximately the same time to compile any circuit, with $\mathcal{R}_{\blacktriangle}$ performing slightly worse, which is coherent with the size difference between the twos. Instead, $\mathcal{R}_{\blacktriangledown}$ outperforms the others lattices. Furthermore, it seems that it is more resistant to scale-up as the scaling seems to follow a lower degree function.

Thanks to the given experimental evaluation, we could give a first evaluation of our model, by implementing it for an important group of commuting circuits – i.e. the $\mathcal{L}_{\wedge(\mathtt{Z})}$. At the same time we could find a good topology for our next step, which is compiling Clifford circuits.

4.6 Clifford circuits compiler

In Sec. 4.5 we evaluated our model on $\mathcal{L}_{\wedge(\mathtt{Z})}$ circuits. Even if they represent an important group of circuits, both theoretically [43] and practically [50, 51, 52, 49, 57] – see Sec. 4.5– we now aim to extend our model to cover any Clifford circuit. Hence, since we managed to split the Clifford compilation in 3 independent problems – see Sec. 4.4 – the final step is to define a compiler for $\mathcal{L}_{\wedge(\mathtt{X})}$ circuits.

4.6.1 Parity check circuits

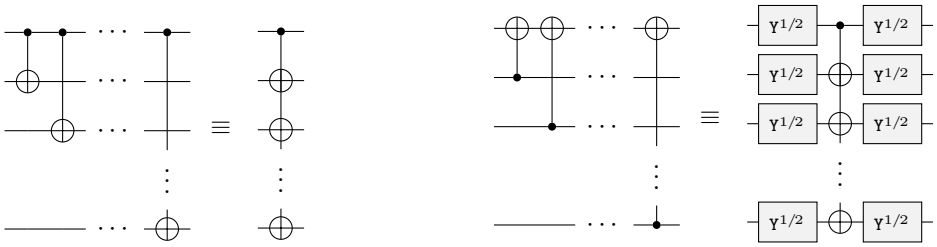
Any $\mathcal{L}_{\wedge(\mathtt{X})}$ can be interpreted as a *parity check* circuit, e.g. [39]:

$$|\mathtt{b}_1, \mathtt{b}_2, \mathtt{b}_3, \mathtt{b}_4\rangle \;\mapsto\; |\mathtt{b}_1 \oplus \mathtt{b}_2, \mathtt{b}_1 \oplus \mathtt{b}_3, \mathtt{b}_4, \mathtt{b}_3\rangle\,.$$

We highlight such a relation to make more intuitive how we are now going to envision a generic $\mathcal{L}_{\wedge(\mathtt{X})}$. It is common in parity check circuits to layering the circuit in operations of two kinds: *fan-in* or *fan-out* [49]. These two kinds are shown in circuit representation in Figs. 4.10a and 4.10b. Hence, one can see an $\mathcal{L}_{\wedge(\mathtt{X})}$ circuit as a sequence of $\wedge(\mathtt{X}^{\otimes m})$ operations, eventually interleaved by a layer of local gates $\mathcal{L}_{\mathtt{Y}^{1/2}}$. m denotes the number of target qubits and respects $m < n$, with n being the total number of qubits.

We already showed in Sec. 2.2.5 how to efficiently implement – in terms of $\mathtt{E}$-count – an $\wedge(\mathtt{X} \otimes \mathtt{X})$ distributed over three processors. But we didn't detailed the general case $\wedge(\mathtt{X}^{\otimes m})$; because it depends on the connectivity of the network.

Assuming to work with rectangular lattices $\mathcal{R}_{\blacktriangledown}$ – since we showed in Sec. 4.5 having the best performance – it is convenient to generalize the definition of entanglement path – given in Sec. 2.2.5 – to the concept of *entanglement tree*.



(a) Fan-in (b) Fan-out

Figure 4.10: Generic fan-in and fan-out operations

4.6.2 Entanglement trees

While it is always possible to find an entanglement path passing through all the qubits involved by a $\wedge(\mathtt{X}^{\otimes m})$ operators, this may not be the best approach to run the subject operator. In fact, the shortest entanglement path does not necessarily finds the minimal $\mathtt{E}$-count. In alternative, consider that a fan-in (fan-out) operator is rooted in the control (target) qubit and it doesn't actually constrains the targets (controls) to be covered by a single entanglement path. Hence, we can opt to compute an *entanglement tree*, which may result in better solutions. In fact, the search for a tree covering $\wedge(\mathtt{X}^{\otimes m})$ can be expressed as a generalization of the *minimum spanning tree* problem [73, Chapter 4.5]. Such a generalization is known as *minumum Steiner tree* problem [74], which, in general, is not tractable. Nevertheless efficient approximation ratio have been achieved [75, 76, 77] and it can be used for any topology. Since a lattice is a special case of the Euclidean plane, a further interesting result is that, for such a group of topologies, the problem admits a *polynomial-time approximation scheme* [78]. Finally, we report a result important to us within the following remark [79, 80]:

Remark. *The minimal Steiner tree on rectangular lattices $\mathcal{R}$ can be found in polynomial time.*

(a) Rectangular lattice $\mathcal{R}_{\blacktriangledown}$ with generator $g = 6$.

(b) A Steiner tree with E-count of 13.

Figure 4.11: A Steiner tree computed from a network in the rectangular lattice topology. The tree is able to perform any non-local $\wedge(\mathtt{X}^{\otimes m})$, spread over thirteen processors.

The above remark tells us that we can find, for any fan-in (fan-out) operation, the minimum E-count possible, by constructing an entanglement tree treated as a minimum Steiner tree problem. Fig. 4.11 shows an example computed with the method provided by the `networkx` library [71].

4.6.3 Circuit construction and partitioning

To prove the quality of our proposal, we start by considering *dense fan-in (fan-out) circuits*, where with dense we mean that the layers of a circuit $\mathcal{L}_{\wedge(\mathtt{X})} = \{\ell_1, \ell_2, \ldots, \ell_d\}$ are such that each $\ell_{\dot{d}} \in \mathcal{L}_{\wedge(\mathtt{X})}$ is a fan-in (fan-out) involving as many qubits as possible[9,10]. The rationale behind this choice is that, when writing a parity circuit, it is convenient to aggregate $\wedge(\mathtt{X})$ operators to form fan-in operators, i.e. $\wedge(\mathtt{X}^{\otimes m})$. This does not restrict the kind of circuits the compiler will be able to process, since also the two-qubit operator $\wedge(\mathtt{X})$ is a basic case of fan-in. By proceeding this way, we can push our compiler to solve the most complex parity check circuits, in terms both of number of operations and commutativity. Specifically, the hardest fan-in circuit has $\binom{n}{2}$ operations and $n-1$ non-commuting layers. For this reason, the E-depth corresponds to the number of layers, while we can optimize the E-count by computing the minimal Steiner tree for each layer. Circuit in Fig. 4.12 shows the criterion.

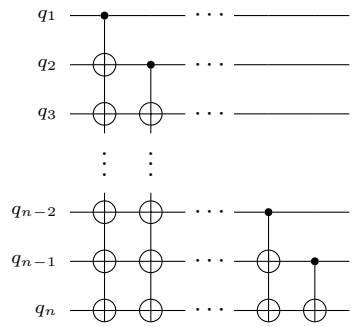


Figure 4.12: Hardest fan-in circuit composed by $\binom{n}{2}$ telegates, with n being the number of qubits/processors.

In terms of our formulation (4.10) of DQCC, this means that we are fixing the order relation for any pair $\wedge(\mathtt{X})_{s_i,t_i} \in \ell_{\dot{d}}$ and $\wedge(\mathtt{X})_{s_j,t_j} \in \ell_{\ddot{d}}$ to satisfy the following statement:

$$i \prec j \iff \dot{d} < \ddot{d} \tag{4.17}$$

[9] Notice that including further operators would turn up to just create repeating pairs of $\wedge(\mathtt{X})$, which cancel out each other.

[10] One may then generalize the definition to let more than one fan-in (or fan-out) within the same layer, as long as they operate on different qubits.

Since we are facing a circuit composed by $\binom{n}{2}$ non-local operations, this is also a lower-bound on the optimal E-count. One last step is required to make the compiler reach this lower-bound. Specifically, for a given set of qubits $Q = \{q_1, q_2, \ldots, q_n\}$ and processors $P = \{P_1, P_2, \ldots, P_n\}$[11], then $q_i \in P_i,\ \forall i$. A remark follows:

Remark. *The compiler hits the lower-bound of $\binom{n}{2}$ for the hardest fan-in (fan-out) circuit, which means the E-count is optimal.*

From this result we also notice that we can apply the same idea to solve $\mathcal{L}_{\wedge(\mathtt{Z})}$ circuits, achieving even better results. Specifically, first notice that fan-in and fan-out circuits do not limit to $\wedge(\mathtt{X}^{\otimes m})$ operator, but also to $\wedge(\mathtt{Z}^{\otimes m})$. Furthermore, since $\mathcal{L}_{\wedge(\mathtt{Z})}$ are commuting circuit, one can always reduce it to a dense fan-in circuit, by proceeding in a greedy fashion outlined below.

1. For each qubits, compute the number of $\wedge(\mathtt{Z})$ in which they are involved;
2. Enumerate the qubits by decreasing order and partition them such that $q_i \in P_i$;
3. iterate over the enumeration and associate the maximal fan-in coming from $\mathcal{L}_{\wedge(\mathtt{Z})}$;
4. solve each layer with the spanning tree based compiler.

In this way, any $\mathcal{L}_{\wedge(\mathtt{Z})}$ circuit is expressed as a dense fan-in circuit, for which we can compute the optimal E-count. In other words:

Remark. *The minimal E-count for any $\mathcal{L}_{\wedge(Z)}$ circuit can be computed efficiently with at most $n-1$ fan-in layers.*

Such a result relates to the same upper-bound of $n-1$ fan-in (fan-out) given in Ref. [57].

Notice that, with this last remark, we now have all the ingredients to construct an efficient compiler for Clifford circuits[12]. In fact, even if so far we proceeded under the assumption that each processor has only one computational qubit and link capacity one. This helped us to let out the potentiality of distributed architectures. We indeed managed to show very promising results. However, this approach does not limit the compiler to only distributed operations and can be extended to differ between local and non-local fan-in (fan-out) and even mixtures. Also the capacity can be generalized as the minimum spanning tree problem already works to minimize the total capacity.

It is also important to remember that the normal form we showed in Sec. 4.4 is only one of several [39, 41, 42, 43, 44, 45]; each with implications on the structure of $\mathcal{L}_{\wedge(\mathtt{Z})}$ and $\mathcal{L}_{\wedge(\mathtt{X})}$ circuits. Hence, despite the appealing properties of the spanning tree approach, whenever the circuit is naturally composed by a few gates, a Steiner tree over fan-in layers may still be beaten by a quickest multi-commodity flow solver.

As an example, consider that for any $\mathcal{L}_{\wedge(\mathtt{X})}$ circuit, it has already been discovered an algorithm to compute the minimum number of $\wedge(\mathtt{X})$ operators [81]. Even if the minimum number of $\wedge(\mathtt{X})$ is an important result that, in perspective, closes positively the problem, this does not mean that optimal E-depth and E-count reside here. Hence, other techniques may be more practical – e.g. see [57].

[11] Please refer to Sec. 4.1 for the employed formalism.
[12] Which may also be expanded to universal circuits with methods as the one outlined in [57].

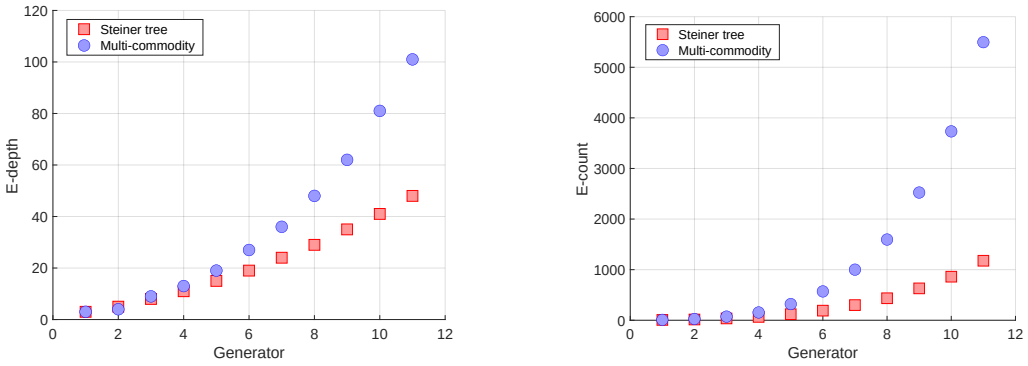
The conclusion of our analysis is that a good compiler should be *tunable*; meaning that it is able to use one or the other approach depending on the circuits in input[13]. For this reason we think a good conclusion of our investigation is a comparison between the two approaches. But before that, let us sum up the steps of a *tunable compiler*:

1. Transform the circuit in input into some normal form.
2. Split the problem into $\mathcal{L}_{\wedge(\mathtt{X})}$ and $\mathcal{L}_{\wedge(\mathtt{Z})}$ sub-circuits.
3. Analyse the structure of the sub-circuits (fan-in density, number of operators, etc.).
4. For each sub-circuit, eventually reduce to a fan-in structure.
5. For each sub-circuit, solve with the multi-commodity flow or the spanning tree approach.

4.7 Multi-commodity flow vs. Steiner trees

Thanks to this new perspective for circuits – i.e. as dense fan-in (fan-out) circuits – we can now evaluate the performance of our previous compiler – see Algorithm 2 – with the new one. In fact, while with the quickest multi-commodity flow we focused on minimizing the E-depth, we are now minimizing the E-count. By means of a careful construction of dense fan-in circuits, we can't do much optimization on the E-depth, while we can still work on the E-count.

To this aim, we focus on $\mathcal{L}_{\wedge(\mathtt{Z})}$ circuits, where both compiler works at their best. However, we consider again the worst case scenario – i.e., fan-in circuits of $\binom{n}{2}$ operators – where we expect the multi-commodity flow approach suffering the most. We already showed the optimality in the E-count for the spanning tree approach, but let us see now how the multi-commodity flow approach work in such a hard case, compared to Steiner trees.



(a) E-depth comparison.

(b) E-count comparison.

Figure 4.13: This is a comparison between a compiler based on multi-commodity flow with one based on Steiner trees.

The plot in Fig. 4.13a shows that the E-depths achieved are still comparable, which preserve a side interest into employing the multi-commodity flow approach. However, from the plot in Fig. 4.13a emerges a neat advantage in terms of E-count. This is probably related to the

[13]It may be also convenient to further refine the Steiner tree approach, e.g. whenever a single layer has more than one fan-in (fan-out) acting on independent qubits, then the layer can be solved by computing the Steiner three of the induced sub-graphs.

formulation of the multi-commodity flow, which has primal interest into minimizing the E-depth, while the E-count is minimized only in the case this is convenient in terms of E-depth. Such an analysis brings the attention to different formulations, called *path-based*, while the one we gave – i.e. formulation (4.10) – is *edge-based*. An interesting branch of research in this direction can be found in Ref. [82].

4.8 Conclusion: the importance of a compiler

According to our envision of the full-stack development – see Sec. 1.2 – a compiler will take care of creating a logical circuit which is compliant with the hardware. To understand how helpful a compiler can be, it is important to keep in mind that the word "logical" should be interpreted in two different ways.

1. The lower layers guarantee a reliable abstraction on which the compiler can operate logically.

2. The compiler lighten the lower layers from all the tasks which are merely logical.

Specifically to the second interpretation, consider that each layer of the stack has important and complex tasks to care about. Lower layers are more prone to treat real-time problems. In fact, the scheduler cares about synchronization and connectivity maximization. The hardware layer cares of being efficient in terms of, e.g., pushing the technologies at their maximal performance to get high fidelities and high success rates. The above tasks are for sure hard. For this reason, identify what can be **delegated** to a logical reasoner – i.e. the compiler – may bring critical advantages to the overall architecture.

To get a practical intuition of what we are stating, consider for example our assumption for the E operator. We used this assumption to provide the reader with a model relatively easy to understand. But with a careful knowledge of the underlying architecture from an information theory perspective, something better can be achieved. As an example, consider the stationary-flying system – see Ch. 1 – generating and distributing entangled states. When it succeeds, it generally means that a heralded Bell state has been produced – i.e. $\{|\Psi^+\rangle, |\Psi^-\rangle, |\Phi^+\rangle, |\Phi^-\rangle\}$ –.

Now; if the reader believe, as we do, that delegating to the compiler to analyse post-processing will bring advantages to the general performance of the quantum computation, then it is clear that this approach does not stop to the model we created throughout this chapter. As a matter of fact, Bell states differ one another by Pauli corrections, which are usually treated at the hardware layer to provide the upper layers with $|\Phi^+\rangle$. But, even if local operations are very efficient, when it comes to multiple repeated steps – as for the case of quantum computation – every single gate avoided has a positive impact on the final fidelity. For the case we are considering now, the presence of a compiler enable the hardware to delegate the corrections, which now includes them to the big set of logical instructions to optimize. As basic example consider the circuit in Fig. 4.14, where a $\wedge(\mathtt{X})$ runs with a different Bell state.

Let us see numerically what happens without delegating the correction to the compiler. Consider a single-qubit gate error of probability $\mathtt{p} = 10^{-n}$. Then, for $10 \cdot m$ non-local operations, the probability get worse of m orders of magnitude, i.e. 10^{-n+m}. On contrary, the compiler eliminates all the corrections, preserving the error probability to 10^{-n}. The same reasoning applies to the running-time.

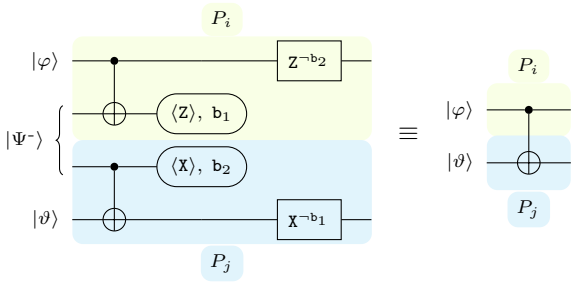


Figure 4.14: Non-local $\wedge(\mathtt{X})$ performed by means of $|\Psi^-\rangle$. This example shows how to avoid Pauli corrections.

With our models we covered several interesting group of circuits[14], for which the compiler eliminates every single correction from the quantum computation. The Bell state correction is no different. Specifically, instead of performing Pauli-corrections, the compiler keeps track of the logical error propagation caused by avoiding it. At the end of the computation the compiler provides the classical computer the necessary bit-flip corrections to get the right final state, just like we have done so far, by means of the rules introduced in Sec. 4.4.1.

References

[1] Adi Botea, Akihiro Kishimoto, and Radu Marinescu. "On the complexity of quantum circuit compilation". In: *Eleventh annual symposium on combinatorial search*. 2018 (cit. on pp. 49, 51).

[2] Liam Madden and Andrea Simonetto. "Best approximate quantum compiling problems". In: *ACM Transactions on Quantum Computing* 3.2 (2022), pp. 1–29 (cit. on p. 49).

[3] Yuan-Hang Zhang et al. "Topological quantum compiling with reinforcement learning". In: *Physical Review Letters* 125.17 (2020), p. 170501 (cit. on p. 49).

[4] Peter J Karalekas et al. "A quantum-classical cloud platform optimized for variational hybrid algorithms". In: *Quantum Science and Technology* 5.2 (2020), p. 024003 (cit. on p. 49).

[5] Lorenzo Moro et al. "Quantum Compiling by Deep Reinforcement Learning". In: *Nature Communications Physics* 4.178 (2021) (cit. on p. 49).

[6] Marco Maronese et al. "Quantum compiling". In: *Quantum Computing Environments*. Springer, 2022, pp. 39–74 (cit. on p. 49).

[7] Kyle EC Booth et al. "Comparing and integrating constraint programming and temporal planning for quantum circuit compilation". In: *28th international conference on automated planning and scheduling*. 2018 (cit. on p. 49).

[8] Davide Ferrari and Michele Amoretti. "Noise-adaptive quantum compilation strategies evaluated with application-motivated benchmarks". In: *Proceedings of the 19th ACM International Conference on Computing Frontiers*. 2022, pp. 237–243 (cit. on p. 49).

[14] In Sec. 4.4 we also gave a perspective on how our approach would eventually bring us to cover universal groups as well.

[9] Stefan Hillmich, Alwin Zulehner, and Robert Wille. "Exploiting quantum teleportation in quantum circuit mapping". In: *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE. 2021, pp. 792–797 (cit. on p. 49).

[10] Lukas Burgholzer, Sarah Schneider, and Robert Wille. "Limiting the Search Space in Optimal Quantum Circuit Mapping". In: *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE. 2022, pp. 466–471 (cit. on p. 49).

[11] Dmitri Maslov, Sean M Falconer, and Michele Mosca. "Quantum circuit placement". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 27.4 (2008), pp. 752–763 (cit. on p. 49).

[12] Marcos Yukio Siraichi et al. "Qubit allocation". In: *Proceedings of the 2018 International Symposium on Code Generation and Optimization*. 2018, pp. 113–125 (cit. on p. 49).

[13] Robert Wille, Lukas Burgholzer, and Alwin Zulehner. "Mapping quantum circuits to IBM QX architectures using the minimal number of `SWAP` and `H` operations". In: *2019 56th ACM/IEEE Design Automation Conference*. IEEE. 2019, pp. 1–6 (cit. on pp. 49, 50).

[14] Gushu Li, Yufei Ding, and Yuan Xie. "Tackling the qubit mapping problem for NISQ-era quantum devices". In: *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*. 2019, pp. 1001–1014 (cit. on p. 49).

[15] Alwin Zulehner and Robert Wille. "Compiling $SU(4)$ quantum circuits to IBM QX architectures". In: *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 2019, pp. 185–190 (cit. on pp. 49, 50).

[16] Toshinari Itoko et al. "Quantum circuit compilers using gate commutation rules". In: *Proceedings of the 24th Asia and South Pacific Design Automation Conference*. 2019, pp. 191–196 (cit. on pp. 49, 50).

[17] Robert Beals et al. "Efficient distributed quantum computing". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 469.2153 (2013), p. 20120686 (cit. on p. 49).

[18] Ranjani G Sundaram, Himanshu Gupta, and CR Ramakrishnan. "Efficient Distribution of Quantum Circuits". In: *35th International Symposium on Distributed Computing*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik. 2021 (cit. on p. 49).

[19] Mariam Zomorodi-Moghadam, Mahboobeh Houshmand, and Monireh Houshmand. "Optimizing teleportation cost in distributed quantum circuits". In: *International Journal of Theoretical Physics* 57.3 (2018), pp. 848–861 (cit. on p. 49).

[20] Omid Daei, Keivan Navi, and Mariam Zomorodi-Moghadam. "Optimized Quantum Circuit Partitioning". In: *International Journal of Theoretical Physics* 59.12 (2020), pp. 3804–3820 (cit. on p. 49).

[21] Davide Ferrari et al. "Compiler Design for Distributed Quantum Computing". In: *IEEE Transactions on Quantum Engineering* 2 (2021), pp. 1–20 (cit. on pp. 49, 50).

[22] Eesa Nikahd et al. "Automated window-based partitioning of quantum circuits". In: *Physica Scripta* 96.3 (2021), p. 035102 (cit. on p. 49).

[23] Davood Dadkhah, Mariam Zomorodi, and Seyed Ebrahim Hosseini. "A New Approach for Optimization of Distributed Quantum Circuits". In: *International Journal of Theoretical Physics* 60.9 (2021), pp. 3271–3285 (cit. on p. 49).

[24] Omid Daei, Keivan Navi, and Mariam Zomorodi. "Improving the Teleportation Cost in Distributed Quantum Circuits Based on Commuting of Gates". In: *International Journal of Theoretical Physics* 60.9 (2021), pp. 3494–3513 (cit. on p. 49).

[25] Moein Sarvaghad-Moghaddam and Mariam Zomorodi. "A general protocol for distributed quantum gates". In: *Quantum Information Processing* 20.8 (2021), pp. 1–14 (cit. on p. 49).

[26] Mariam Zomorodi-Moghadam, Zohreh Davarzani, Ismail Ghodsollahee, et al. "Connectivity matrix model of quantum circuits and its application to distributed quantum circuit optimization". In: *Quantum Information Processing* 20 (2021) (cit. on p. 49).

[27] LJ Stephenson et al. "High-rate, high-fidelity entanglement of qubits across an elementary quantum network". In: *Physical review letters* 124.11 (2020), p. 110501 (cit. on p. 49).

[28] Daniele Cuomo, Marcello Caleffi, and Angela Sara Cacciapuoti. "Towards a distributed quantum computing ecosystem". In: *IET Quantum Communication* 1.1 (2020), pp. 3–8 (cit. on p. 49).

[29] Rodney Van Meter and Simon J Devitt. "The path to scalable distributed quantum computing". In: *Computer* 49.9 (2016), pp. 31–42 (cit. on p. 49).

[30] Daniel Gottesman. "Theory of fault-tolerant quantum computation". In: *Physical Review A* 57.1 (1998), p. 127 (cit. on p. 51).

[31] Peter Selinger. "Quantum circuits of T-depth one". In: *Physical Review A* 87.4 (2013), p. 042302 (cit. on pp. 51, 57).

[32] Matthew Amy, Dmitri Maslov, and Michele Mosca. "Polynomial-time T-depth optimization of Clifford+T circuits via matroid partitioning". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33.10 (2014), pp. 1476–1489 (cit. on pp. 51, 57).

[33] Daniele Cuomo et al. "Optimized Compiler for Distributed Quantum Computing". In: *ACM Transactions on Quantum Computing* (2023) (cit. on pp. 52, 57).

[34] Lisa Fleischer and Martin Skutella. "The quickest multicommodity flow problem". In: *International Conference on Integer Programming and Combinatorial Optimization*. Springer. 2002, pp. 36–53 (cit. on pp. 53, 54).

[35] Maokai Lin and Patrick Jaillet. "On the quickest flow problem in dynamic networks – A parametric min-cost flow approach". In: *Proceedings of the 26th annual ACM-SIAM symposium on discrete algorithms*. SIAM. 2014, pp. 1343–1356 (cit. on p. 53).

[36] Lester R Ford Jr and Delbert Ray Fulkerson. "Constructing maximal dynamic flows from static flows". In: *Operations research* 6.3 (1958), pp. 419–433 (cit. on p. 53).

[37] Lester R Ford Jr and D.R. Fulkerson. "A suggested computation for Maximal Multi-Commodity Network Flows". In: *Management Science* 5.1 (1958), p. 97 (cit. on p. 53).

[38] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. "Network flows". In: (1988) (cit. on p. 55).

[39] Ross Duncan et al. "Graph-theoretic Simplification of Quantum Circuits with the ZX-calculus". In: *Quantum* 4 (2020), p. 279 (cit. on pp. 58, 64, 66).

[40] Daniel Litinski. "A game of surface codes: Large-scale quantum computing with lattice surgery". In: *Quantum* 3 (2019), p. 128 (cit. on p. 58).

[41] Scott Aaronson and Daniel Gottesman. "Improved simulation of stabilizer circuits". In: *Physical Review A* 70.5 (2004), p. 052328 (cit. on pp. 58, 66).

[42] Jeroen Dehaene and Bart De Moor. "Clifford group, stabilizer states, and linear and quadratic operations over GF (2)". In: *Physical Review A* 68.4 (2003), p. 042318 (cit. on pp. 58, 66).

[43] Dmitri Maslov and Martin Roetteler. "Shorter stabilizer circuits via Bruhat decomposition and quantum circuit transformations". In: *IEEE Transactions on Information Theory* 64.7 (2018), pp. 4729–4738 (cit. on pp. 58, 59, 63, 66).

[44] Marc Bataille. "Reducing stabilizer circuits without the symplectic group". In: *arXiv preprint arXiv:2012.09224* (2020) (cit. on pp. 58, 66).

[45] Sergey Bravyi and Dmitri Maslov. "Hadamard-free circuits expose the structure of the Clifford group". In: *IEEE Transactions on Information Theory* 67.7 (2021), pp. 4546–4563 (cit. on pp. 58, 66).

[46] Aleks Kissinger and John van de Wetering. "PyZX: Large Scale Automated Diagrammatic Reasoning". In: Proceedings 16th International Conference on *Quantum Physics and Logic*. Vol. 318. Open Publishing Association, 2020, pp. 229–241 (cit. on pp. 58, 60).

[47] John van de Wetering. "ZX-calculus for the working quantum computer scientist". In: *arXiv preprint arXiv:2012.13966* (2020) (cit. on p. 58).

[48] Bob Coecke and Ross Duncan. *A graphical calculus for quantum observables*. `https://zxcalculus.com/publications.html`. 2007 (cit. on p. 58).

[49] Dmitri Maslov and Yunseong Nam. "Use of global interactions in efficient quantum circuit constructions". In: *New Journal of Physics* 20.3 (2018), p. 033018 (cit. on pp. 59, 63, 64).

[50] Sergey Bravyi, Dmitri Maslov, and Yunseong Nam. "Constant-cost implementations of Clifford operations and multiply controlled gates using global interactions". In: *arXiv preprint arXiv:2207.08691* (2022) (cit. on pp. 59, 63).

[51] Nikodem Grzesiak et al. "Efficient quantum programming using EASE gates on a trapped-ion quantum computer". In: *Quantum* 6 (2022), p. 634 (cit. on pp. 59, 63).

[52] Pascal Baßler et al. "Synthesis and compilation with time-optimal multi-qubit gates". In: *arXiv preprint arXiv:2206.06387* (2022) (cit. on pp. 59, 63).

[53] Yao Lu et al. "Global entangling gates on arbitrary ion qubits". In: *Nature* 572.7769 (2019), pp. 363–367 (cit. on p. 59).

[54] Jorge Casanova et al. "Quantum simulation of interacting fermion lattice models in trapped ions". In: *Physical review letters* 108.19 (2012), p. 190502 (cit. on p. 59).

[55] Svetoslav S Ivanov, Peter A Ivanov, and Nikolay V Vitanov. "Efficient construction of three-and four-qubit quantum gates by global entangling gates". In: *Physical Review A* 91.3 (2015), p. 032311 (cit. on p. 59).

[56] Esteban A Martinez et al. "Compiling quantum algorithms for architectures with multi-qubit gates". In: *New Journal of Physics* 18.6 (2016), p. 063029 (cit. on p. 59).

[57] John van de Wetering. "Constructing quantum circuits with global gates". In: *New Journal of Physics* 23.4 (2021), p. 043015 (cit. on pp. 59, 60, 63, 66).

[58] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. "A complete axiomatisation of the ZX-calculus for Clifford+T quantum mechanics". In: *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*. 2018, pp. 559–568 (cit. on p. 60).

[59] Titouan Carette et al. "Completeness of Graphical Languages for Mixed State Quantum Mechanics". In: *ACM Transactions on Quantum Computing* 2.4 (2021), pp. 1–28 (cit. on p. 60).

[60] Miriam Backens. "The ZX-calculus is complete for stabilizer quantum mechanics". In: *New Journal of Physics* 16.9 (2014), p. 093021 (cit. on p. 60).

[61] Aleks Kissinger and John van de Wetering. "Reducing T-count with the ZX-calculus". In: *arXiv preprint arXiv:1903.10477* (2019) (cit. on p. 60).

[62] Luke E Heyfron and Earl T Campbell. "An efficient quantum compiler that reduces T count". In: *Quantum Science and Technology* 4.1 (2018), p. 015004 (cit. on p. 60).

[63] Ross Duncan. "A graphical approach to measurement-based quantum computing". In: *arXiv preprint arXiv:1203.6242* (2012) (cit. on p. 60).

[64] Mihir Pant et al. "Routing entanglement in the quantum internet". In: *npj Quantum Information* 5.1 (2019), pp. 1–9 (cit. on p. 60).

[65] D Kleitman et al. "A matching theorem for graphs". In: *Journal of Combinatorial Theory* 8.1 (1970), pp. 104–114 (cit. on p. 60).

[66] Daniel J Kleitman. "An algorithm for certain multi-commodity flow problems". In: *Networks* 1.1 (1971), pp. 75–90 (cit. on p. 60).

[67] Bernhard Korte and Jens Vygen. "Multicommodity Flows and Edge-Disjoint Paths". In: *Combinatorial Optimization: Theory and Algorithms*. Springer, 2006 (cit. on p. 61).

[68] Maren Martens. "A simple greedy algorithm for the k-disjoint flow problem". In: *International Conference on Theory and Applications of Models of Computation*. Springer. 2009, pp. 291–300 (cit. on p. 61).

[69] Petr Kolman and Christian Scheideler. "Improved bounds for the unsplittable flow problem". In: *SODA*. Vol. 2. 2002, pp. 184–193 (cit. on p. 61).

[70] Li Fei. *Multicommodity Flows and Disjoint Paths Problem*. https://cs.gmu.edu/~lifei/teaching/cs684spring17/lec8.pdf. 2017 (cit. on p. 61).

[71] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring network structure, dynamics, and function using networkx". In: *Proceedings of the 7th Python in Science Conference*. Pasadena, CA USA, 2008, pp. 11–15 (cit. on pp. 61, 65).

[72] MATLAB. *R2021b*. Natick, Massachusetts: The MathWorks Inc., 2021 (cit. on p. 62).

[73] Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson Education India, 2006 (cit. on p. 64).

[74] Dietmar Cieslik. *Steiner minimal trees*. Vol. 23. Springer Science & Business Media, 2013 (cit. on p. 64).

[75] Lawrence Kou, George Markowsky, and Leonard Berman. "A fast algorithm for Steiner trees". In: *Acta informatica* 15.2 (1981), pp. 141–145 (cit. on p. 64).

[76] Piotr Berman, Marek Karpinski, and Alexander Zelikovsky. "1.25-approximation algorithm for steiner tree problem with distances 1 and 2". In: *Workshop on Algorithms and Data Structures*. Springer. 2009, pp. 86–97 (cit. on p. 64).

[77] Miroslav Chlebık and Janka Chlebıková. "The Steiner tree problem on graphs: Inapproximability results". In: *Theoretical Computer Science* 406.3 (2008), pp. 207–214 (cit. on p. 64).

[78] Jaroslaw Byrka et al. "An improved LP-based approximation for Steiner tree". In: *Proceedings of the forty-second ACM symposium on Theory of computing*. 2010, pp. 583–592 (cit. on p. 64).

[79] Marcus Brazil et al. "Minimal Steiner trees for 2k× 2ksquare lattices". In: *journal of combinatorial theory, Series A* 73.1 (1996), pp. 91–110 (cit. on p. 64).

[80] Marcus Brazil et al. "Minimal Steiner trees for rectangular arrays of lattice points". In: *journal of combinatorial theory, Series A* 79.2 (1997), pp. 181–208 (cit. on p. 64).

[81] Ketan N Patel, Igor L Markov, and John P Hayes. "Optimal synthesis of linear reversible circuits." In: *Quantum Inf. Comput.* 8.3 (2008), pp. 282–294 (cit. on p. 66).

[82] Julian Rabbie et al. "Designing quantum networks using preexisting infrastructure". In: *npj Quantum Information* 8.1 (2022), pp. 1–12 (cit. on p. 68).