QuantERA Full Proposal

Co-Op ZX
Compilation and optimisation for near-term quantum computing using the ZX calculus

Title page will be replaced with file Front-Page
Text from previous proposal in blue

Duration: 36 months

Summary of the project

Recent investment in quantum technologies has paid off, and quantum computers are now here. Current and near-term quantum computers, known as noisy intermediate-scale quantum (NISQ) devices, have few qubits, short coherence times, and non-trivial gate error. In this regime, quantum software support – in particular focussed towards compilation and optimisation – is vital to the efficient use of scarce, noisy, hardware resources, and the development of implementable protocols that go beyond what is feasible classically.

These NISQ computers are not so much single devices, but instead patchworks of components (including classical) which vary greatly between implementations such as silicon qubits, superconducting circuits, or ion traps. Programming such devices currently requires intimate knowledge of the hardware, which is a significant barrier to the realisation of usable, scalable quantum computers, as programs must be rewritten for every new device. High-level software descriptions of quantum algorithms must be translated to low-level control instructions for quantum hardware. However, whereas classical computers have had a roughly static concept of "low-level instructions" for decades, the analogous notion for quantum hardware is constantly changing and evolving to cope with the rapid progress in quantum technology. We face a situation where the ever-multiplying range of quantum computers has minimal software support.

This project develops the ZX calculus of observables as a flexible intermediate language for quantum computation, and a core element in compilation for immediate use with NISQ devices. This intermediate language will be versatile enough to target a wide variety of hardware implementations, and simple enough to support any programming language. This project builds on recent significant formal and practical advances in completeness and optimisation of the ZX calculus. The former enables provably-correct program transformations for automatically adding error correction and performing hardware-guided optimisations, e.g. by preferring certain quantum gates over others or enforcing topological constraints. This project will develop, enrich (with annotations), and standardise the ZX language for quantum computation, integrate it into an effective stack of tools for compiling quantum programs, and develop new techniques for automated transformations that make quantum computations run better and faster.

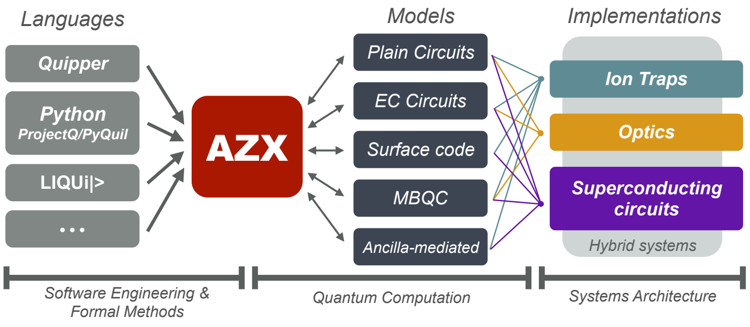
Relevance to the topic addressed in the call

The project clearly comprises "*transformative research*" that explores "*collaborative advanced interdisciplinary science and/or cutting-edge engineering with the potential to initiate or foster new lines of quantum technologies*", which is the key overall objective of QuantERA. We include several "*excellent young researchers*", including from Poland, along with well-established figures at all levels in the project, and partner with Cambridge Quantum Computing, a clearly "*ambitious high-tech SME*". In particular we address the *Quantum Computation* area of the call. The retargettable nature of the compiler supports "*new architectures for quantum computation*", in particular technologically heterogeneous implementations. The optimising aspect of the compiler will allow the "*optimisation of error correction codes*", both at the intermediate and machine level. The ability to compile multiple high-level languages will promote the "*development of novel quantum algorithms*". Machine-dependent optimisation work will contribute to the "*development of devices to realise multiqubit algorithms*". More generally, this project is an enabling technology that multiplies the impact of all the target outcomes of QuantERA and the Quantum Technology Flagship.

1 EXCELLENCE

1.1 Targeted breakthrough, baseline of knowledge and skills

Summary: Our goal is to develop a flexible intermediate representation for quantum software which enables formally verified program transformations, and based on this, to construct the back-end for a re-targetable optimising compiler for a variety of realistic quantum computer architectures.
FIGURE NEEDS RE-DRAWING



Context: High-level programming languages (HLLs) increase programmer productivity and software reliability, provided that the HLL compiler can generate machine code which runs well on the intended hardware platform. Quantum algorithm designers have the choice of several powerful quantum programming languages [15, 24, 27, 29]. However, proposed implementations of quantum computers vary greatly due to differing underlying technologies (ion traps, superconducting circuits, optics) and architectural concepts (networked vs. hybrid, measurement based, ancilla driven) [23, 26, 25], and no language takes account of the specific characteristics of any given platform. Worse, the technology is evolving quickly, none of these characteristics are stable, and no consensus has yet emerged on the best choice. For classical programs, modern compiler toolchains such as LLVM[1] decouple the HLL from the machine by using an *intermediate representation* (IR), which is independent of both. By translating to and from the IR, any HLL may be used on any platform. A quantum IR accommodating dissimilar architectures is needed to support software on rapidly shifting hardware.

Targeted breakthrough: We will define a universal intermediate representation language, called AZX, which is platform-agnostic but which facilitates program transformations appropriate to specific hardware. Specifically, AZX will provide the following (see §3 for more detail):

- An easy-to-generate universal language.
- Automated IR transformations for error correction, resource optimisation, and execution layout.
- A complete compilation pipeline from HLL to hardware for (i) superconducting transmon qubits (QuTech) [28] and (ii) optically-coupled ion traps (NQIT) [23].

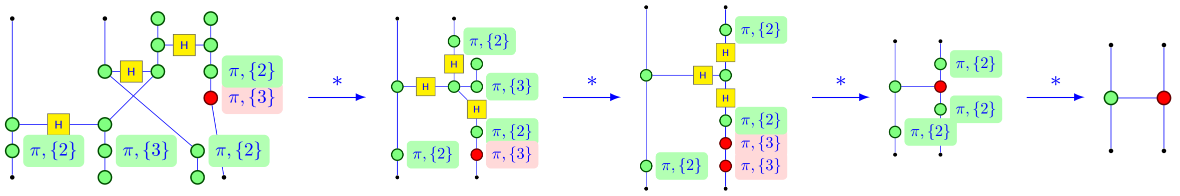
The theory, algorithms, and software architecture behind this will be flexible and extensible, thus permitting programming languages and architectures not explicitly included in the project scope to be supported. This will greatly improve the software ecosystem for quantum computers: by supporting AZX, future quantum devices may easily run existing programs, and future programming languages automatically gain support on a wide range of hardware.

Baseline of knowledge and skills: The closest extant thing to a quantum IR is QASM [7], a circuit description language. QASM is an output option for ProjectQ [16], and the input language for IBM's *Quantum Experience*[2]; the ProjectQ team describe a compilation process for quantum software involving possibly multiple IRs, but do not describe a specific one. State-of-the-art quantum programming languages such as Quipper [15] and LIQUi$|\rangle$ [29] support multiple targets; in practice the choice is between a circuit description or a classical simulation. Neither language exposes an IR.

[1]The LLVM Compiler Infrastructure, `http://llvm.org`
[2]This is a publicly accessible 5-qubit device; `http://research.ibm.com/quantum/`.

Our proposed language AZX is the successor to the ZX-calculus [5], a graphical tensor notation developed by members of this consortium during a decade of work on the mathematical foundations of quantum computing [1, 6]. The ZX-calculus can be viewed in two distinct ways: first, as a formal axiomatic theory which encodes the properties of complementary observables in categorical algebra; second, as a symbolic notation for tensor networks representing quantum states and linear operators. Terms in the ZX-calculus are labelled graphs; equations in the calculus are reified as a small number of graph rewrite rules. This equational theory is frequently more tractable than working explicitly with matrix representations. Very recently [18] members of our consortium have presented a version of the ZX-calculus which is *complete* for the Clifford+T fragment of quantum mechanics, which means that for the first time there is a finite axiomatic theory which can prove every true statement about practical quantum computation, and just before submitting this proposal a version of the ZX-calculus which is complete for universal quantum computing i.e. the Hilbert space based equational theory can be replaced with an entirely diagrammatic one [22].[3] This stunning achievement opens the door to many new possibilities for optimisation and verification of quantum computations.

The ZX-calculus has been extensively applied to quantum computation, and can easily describe computations in both the circuit and measurement-based models of quantum computation (MBQC) [26]. Due to its great flexibility and expressive power, the ZX-calculus can be used to formulate and verify quantum error correcting codes [4, 11] and quantum algorithms [14, 30]. Its graphical language is especially well-suited to systems which naturally have graph structure such as surface codes for topological cluster-states [17] and MBQC [10], where it has been used to translate [12] between the 1-way model and the circuit model. Here is an expample of such a transformation:



Furthermore, techniques for automated reasoning in the ZX-calculus were developed for the interactive theorem prover `quantomatic` [20] which was used to formally verify quantum communication protocols and error correcting codes [4, 11].

Contribution to the theme addressed We specifically address the theme of *Quantum Computation*. The goal of AZX is to make it easy to run quantum programs on any available quantum device. This will speed new quantum devices and architectures into use, and broaden the range of potential users of quantum computers.

1.2 Novelty, level of ambition and foundational character

Novelty: AZX is a totally new way of representing computation. Tensor networks specify *what* the program should do, augmented with separate instructions for *how* the hardware should do it. Unlike sequences of gates, these tensor networks have no preordained causal structure beyond the global input and output. This gives the AZX system great flexibility with respect to time-space trade-offs, and will help to achieve optimal implementations on diverse architectures. Further, the language allows transformations of tensor networks which cannot be expressed as equations between circuits. Key examples include powerful normal-form theorems for Frobenius and Hopf algebras.

A second indispensable novelty of our project, compared to existing program calculi, is that the equational theory is based on quantum mechanics rather than classical concepts. This formal underpinning implies that our program transformations are provably correct, and indeed will come with proofs of their correctness. This symbolic approach also avoids the dimensional explosion of explicit matrices.

[3]This result still needs some fine-tuning and cleaning-up, but the very fact that this can be done is an unexpected boost for the potential of the ZX-calculus.

Ambition: Developing a vertically integrated system from high-level programming languages to two different quantum computing architectures (T4.2, **??**) is ambitious in its scale and reach, and has not been done before. We take a technically ambitious approach to quantum software, reconstructing a program's implementation from its specification and the characteristics of the target machine.

Foundational Character: All other systems take the gate model of quantum computing as a given: this is entirely natural as it is the *lingua franca* of quantum computation researchers. The project proposes a new foundation for quantum software, based on a flexible, powerful, tensor-based representation combined with mathematically rigorous semantics and formal verification. The AZX system will allow quantum hardware implementors to forget the gate model, present a simpler interface to the outside world, and let the compiler bridge the gap. This in turn will facilitate the development of new architectures and technologies for quantum computing. A key example, exploited in our collaboration with NQIT, is that lattice surgery operations on surface codes do not fit into the gate model, but have natural and simple representations in the ZX-calculus [9].

1.3 Concept and methodology

Our proposed AZX language is an advanced ZX-style system augmented with features needed for applications. The ZX-calculus occupies a place in quantum computation similar to the λ-calculus in classical computing: it provides a solid but austere theoretical foundation, without any niceties for practical usage. The power of AZX comes from a second layer of *annotations* on the tensor graph, describing program parameters and architectural constraints of a specific hardware configuration. This two-level design separates the specification (graph layer) from the implementation (annotation layer) of the program, and is the key to achieving our goal of supporting multiple targets.

AZX will retain the mature and effective formal tensor language of the ZX-calculus at its heart, ensuring semantic soundness, logical completeness [18, 22], and allowing techniques from earlier work (cf. `quantomatic` [20]) to be applied to AZX terms. This denotational kernel specifies the process to be carried out, independent of the target platform. Many important transformations can be performed at this platform independent level — without recourse to matrix representations of the operations involved — such as simplifying the tensor network, reducing Clifford fragments to minimal forms, and reducing T-count. Development of such techniques is a low-risk extension of earlier work, and will be done early in the project (T3.1,T3.4). Further, at this stage a program can be translated to a fault-tolerant equivalent with respect to a chosen error-correcting code.

The annotations of the second layer provide the basis of *augmented rewrites*: program transformations which are guided by the annotations to achieve particular goals, not expressible in the basic tensor language. For instance, there is an efficient algorithm [21] to find the *gflow* of a graph state; if the state has a gflow then it supports deterministic 1-way computation [3]. Annotating the graph with its gflow provides guidance for a rewrite strategy which produces an equivalent, space-optimal circuit [12]. We propose to generalise this concept to encompass other sorts of information which would inform how to transform (i.e., to re-write) a generic representation of a quantum computational procedure. For example, we would develop a system which specifies both how to represent logical operations in a particular error correcting code, and how the operations are constrained in order to satisfy basic precautions to keep the realisation fault-tolerant (T3.5). We may then attempt to re-write procedures, to minimise the number of operations subject to the constraints described by those annotations. We may further elaborate such a system of annotations by a description of the constraints and the costs involved for operations within a particular hardware platform (T4.3). For example, in hybrid architectures like NQIT, the annotations will also indicate the differing behaviour of the subsystems. Augmented rewrites will be used to find a runnable implementation of the abstract tensor for the target platform, and to optimise resource use. The development of the general theory of annotations and augmented rewrites (T3.2, T3.3), algorithms for inferring specific annotations (WP2), and rewrite strategies which exploit them (T4.4) form a major novel component of the project.

Concrete tensor networks have a fixed finite size, whereas algorithms are described in parametric fashion, e.g. varying according the input size. To accommodate this we introduce a second class of annotations to represent limited forms of iteration and recursion, yielding *parametric* AZX terms. While the hardware-derived annotations are inferred in a bottom-up fashion, the parametric structure is produced top-down, based on the original HLL program. As this information is typically erased by the circuit generation phase [15, 7] of compilation, we effectively move the boundary between AZX and the HLL above the circuit-level. This is possibly the most challenging part of the project (T2.2); however, we have experience of similar constructs from the `quantomatic` project [19, 20].

Early in the project, we will implement translations from existing quantum programming languages (T1.3). These will provide examples and test cases, and allow comparative evaluation.

The four major work packages of the project are structured into various themes: the relation between ZX and other quantum computing representations (WP1); necessary theoretical developments of ZX (WP2); optimisation strategies independent of implementations (WP3); using enriched ZX to compile and optimise for specific quantum devices.(WP4).

1.3.1 A quantum compiler stack

In the quantum setting, several powerful high-level languages (HLLs) such as Quipper [15] and LIQUi$|\rangle$ [29] have been proposed. As in the case of their classical counterparts, these HLLs are not designed to be run directly on quantum hardware, rather their compilers typically output quantum circuit descriptions. Rather than designing a new programming language, we propose a quantum IR, AZX. Unlike the languages mentioned above, AZX is not intended to be written by humans[4], instead it will be generated by a compiler front-end from programs written in existing high-level languages. Therefore it is essential to provide a robust, general framework for compilation of HLLs to AZX.

Since most existing quantum HLLs can output circuit descriptions, and since circuits can easily be represented in the ZX-calculus, we first design a simple front end for the circuit language QASM [7] in **??**. This will allow AZX terms to be produced from virtually any extant quantum HLL, albeit rather naively. Later, we will perform concrete front-end experiments using more sophisticated existing HLLs, for example *Quipper*, LIQUi$|\rangle$, or ProjectQ [27] during the long running task **??**.

This work package consists of a back-and-forth interaction between HLLs and AZX, which is why many of its tasks span the entire lifetime of the project. HLL development influences AZX by providing the control structures and other constructs that must be represented at AZX level. Conversely, AZX developments will influence HLL, as we analyse how information and requirements from the back-end flow upward in a meaningful way, and how this can be translated into compilation flags and assertions/hints to the compiler within HLL programs.

A key research challenge of this work package consists in the management of the *classical computation* and *classical information* within quantum algorithms: What computation should occur at the AZX-generation phase, and which classical parameters are passed on to the AZX terms? To help answer this question we will design a test suite (T1.4) to compare possible solutions.

The open database of tests developed in T1.4 will serve as a measuring tool for the quality of the output. The database will also be made available to the community for rating and testing future compilers or optimisation techniques. To encourage interaction from other research groups, and to support other languages, both our interface and the AZX language will be made public.

1.3.2 Representation and reasoning

Proposed and existing quantum devices differ along a variety of axes. At the most abstract level, the quantum circuit model and the 1-way model [26] have different execution concepts and primitive operations, despite their computational equivalence. More realistic models might suffer from limitation to a fixed number of qubits, a bounded total execution time, or restrictions on which qubits may interact

[4]This said, the ZX-calculus has proved a very useful notation for mathematical proofs.

directly. Primitive operations will require different amounts of time, different qubit implementations have different failure modes. In a hybrid architecture like NQIT, these properties will vary across the subsystems, with concomitant implications for the execution of the desired program.

Since the overall goal of the project is to produce a *retargetable* compiler, able to generate executables for multiple architectures, these differing characteristics must be taken into account. The ability to synthesise hardware-appropriate implementations from abstract descriptions is one of the major novel contributions of this project. Towards this objective, in WP2 we model the performance characteristics and architectural constraints of various idealised and realistic machines, and develop language features of AZX to express these properties. The goal is two-fold: to facilitate *code-generation* for a given machine from an AZX term; and to expose information needed by the *optimiser*.

Due to its novelty, we adopt an exploratory approach. Initially, and in parallel, we study the circuit model (T1.1) and the 1-way model (T1.2) because these models are well understood, stable, and have been extensively treated in the ZX-calculus literature. We will use these examples to develop prototypical AZX constructs expressing the relevant properties. This consists in three tightly related tasks: decomposing the tensor network into atomic operations; characterising runnability with respect to the model by predicates in monadic second order logic; and transforming the tensor network into an equivalent runnable version. This experience will inform the later work in WP3 and WP4.

In the second phase we switch attention to real computers. The first target is a classical simulator running on HPC hardware provided by our industrial partner Bull. This will leverage existing expertise in simulation, combined with new techniques for symbolic evaluation of AZX terms. Finally we study two concrete quantum computers based on different technologies: superconducting circuits (Delft) and optically linked ion traps (NQIT). In both cases we will interact strongly with the experimental groups working on these models, who are either members of our consortium (S. Benjamin and N. de Beaudrap for NQIT) or our advisory board (L. DiCarlo in Delft). Since these architectures are dissimilar, tackling both is an ideal demonstration of our approach. The completion of this phase will allow quantum programs represented as AZX terms to be run on real hardware.

1.3.3 Machine-independent optimisation

The formal mechanisms for transforming the AZX diagrams produced by HLL compilers into optimised, physically implementable computations are the theoretical core of this proposal, and developing effective techniques for working with AZX diagrams are a prerequisite for our success.

Recent breakthroughs in the theory of the ZX-calculus [18, 22], have shown that whenever two ZX-calculus diagrams describe the same linear operator, then one can be transformed into the other using just a finite set of local, diagrammatic transformations.

However, completeness of the ZX-calculus is just the beginning of the story for AZX. Knowing it is possible *in principle* to transform one computation (e.g. a quantum circuit) into another one doesn't say anything about efficiency or our ability to find effective optimisations. In task T3.1, we will employ theoretical and automated techniques drawn from rewrite theory to search for better presentations of the Clifford+T ZX-calculus and develop strategies for effectively simplifying ZX-diagrams. These include Knuth-Bendix completion and theory synthesis. In task T2.1, we will extend the ZX-calculus in two dimensions. The first is to expand into complete and universal qudit variations to work effectively beyond 2-level systems, and the second is to gain a deeper understanding of the role played by W-type tensors as they interact with the generators of the ZX-calculus, which are themselves of GHZ-type.

While in the early stages of the project, it will already be quite useful to study concrete diagrams of fixed size (e.g. a quantum circuit on N qubits for a previously-fixed N), in task T2.2, we will extend AZX to support parametrised families of diagrams (e.g. quantum circuits with N qubits where N can vary) mirroring the control structures present in a quantum HLL. This will enable more sophisticated, generic optimisations to be run in advance of needing any particular computational procedure.

Finally, we will construct a theoretical framework which mediates the abstract rewrite theory of

ZX-diagrams and real-world constraints coming from quantum hardware. In task T3.2, we will extend the language of AZX to express topological constraints and causal ordering. These could include a restriction to nearest-neighbour interactions for 2-qubit operations on a fixed lattice or enforcing a fixed ordering between two gates. We will then develop the formal tools for rewriting ZX-diagrams in ways that respect those constraints. Similarly, in task T3.3 we will explore methods to annotate a ZX-diagram with quantitative information such as timing, noise, or fidelity. In real-world systems, these can vary vastly between qubits interacting in different ways (e.g. neighbouring in ion trap vs. interactions mediated by optical channel [23]) or stored in different physical modes.

1.3.4 Machine-dependent optimisation

An AZX term produced by the compiler front-end is, by design, an abstract tensor network, perhaps annotated with some useful information, but generally without any preconception of how it should be executed on any particular machine. Translating such abstractly-described procedures to code that can run on realistic machines is a key objective for the project. WP4 is responsible for developing the tools to do so. This work package represents the most technically involved and multi-disciplinary component of the AZX project, requires the integration of the theoretical work of WP3 and generalisation from the specific machines considered in WP2, and will result in software forming the basis of a general-purpose quantum compiler.

We identify three main tasks: to add suitable error protection to the program; to optimise the program according to whichever resources are most appropriate for the given machine; and to layout the program for execution. Although we treat them separately, in practice these tasks will interact in non-trivial ways, and their order need not be fixed.

Error correction should be performed automatically. We have extensive experience in treating error correcting codes in the ZX-calculus [11, 4, 9, 13]. Similar techniques will enable translating from "raw" AZX terms to error corrected / fault-tolerant versions of the same program; additional annotations will be added to ensure that other program transformations do not break the fault-tolerance. We identify two kinds of optimisation. First, generic, model-independent optimisations work on the raw tensor network, typically by reducing its graph complexity, or by minimising the number of non-Clifford operations in the graph. This draws on T3.1 and could be applied before the target hardwaere is known. Second, optimisations which take into account the resource models of a specific machine, expressed through annotations, cf. T3.3. This might be applied before or after layout, depending on the circumstances. Layout will be handled in a similar way to this second kind of optimisation, generalising from T4.2 and **??**.

Although we will provide specific modules for the tasks described above, the AZX system is intended to extensible, so we will also publish an open API and specification language to simplify the task of adding new architectures and error correcting schemes to the system (T1.5).

The tasks to be performed within WP4 may be broadly described in terms of how the AZX compiler will transform ZX terms produced by the front-end to obtain instructions to be realised by a quantum computer (or software quantum simulator) at the back-end. These stages are: (i) an initial round of generic, hardware-independent optimisations; (ii) application of some choice of strategy for error correction; (iii) translation to a specialised annotation system which represents the parameters and constraints of a specific hardware implementation; and finally (iv) another round of optimisation within the constraints imposed by the error correction and hardware models. In addition to the development of the tools for these stages, WP4 will develop an interface for the specification of the annotation systems used in stages (iii) and (iv) above, allowing for easy extension of the AZX compiler to arbitrary hardware systems and allowing AZX to act as a general-purpose quantum compiler.

The first stage of the compilation process represents a "generic optimisation" subroutine (T3.4), which may be applied to arbitrary ZX terms. This subroutine will re-write ZX terms into ones with fewer resources in a broadly applicable sense, such as fewer total nodes or fewer nodes which realise non-Clifford transformations (for instance, corresponding to T gates). This may be developed independently of the results of WP1 or WP2 using existing techniques (as well as incorporating any further useful techniques developed in T2.1 and T3.1).

The second stage of the compilation process is to take a generic ZX term expressing a computation on idealised quantum systems, and re-write it as a ZX term representing an equivalent transformation of error-corrected qubits (T3.5). As well as the ZX terms to translate, this will take as input a specification the particular error correction code or other fault-tolerance construction to apply.

The third stage of the compilation process attempts to map a ZX term into an equivalent ZX term which closely models the constraints of a target architecture (T4.3). This represents the core of the compilation process, taking ZX terms representing a procedure in an abstract model of quantum computation such as circuits or MBQC patterns (with or without error correction), and mapping them into a form which conforms to the physical constraints of a specific hardware implementation. Particular implementations are specified by a system of annotations provided by the development environment, consisting of an "architecture-targeted annotation" (or ArcTAn) system. ArcTAn systems will generalise the particular examples of implementation-oriented annotation systems developed in WP2, and will aim to encompass as many extant and forseeable quantum hardware platforms as possible, incorporating topological and time-ordering constraints as captured by the results of T3.2.

The fourth and penultimate stage of the compilation process (prior to translation to the machine language of the target architecture) is a final round of optimisation, this time applied to ZX terms within the constraints of the specific choice of error correction strategy and machine resources specified by the input (T4.4). This will involve the development of a theory of re-writing techniques developed in T3.2 to ArcTAn annotation systems. By performing a final round of optimisations using a theory of rewrites which apply to all ArcTAn annotation systems, we aim to make possible a reduction in the resources used in any particular hardware platform without requiring the use of bespoke techniques for each target architecture.

Annotation systems representing the hardware implementation are to be provided by the development environment, using a standardised interface. By providing public documentation for this interface (T1.5), we enable third-party developers to extend the functionality of the AZX compiler to arbitrary platforms, thus ensuring the suitability of the AZX compiler as part of a general-purpose quantum software development platform.

1.4 Interdisciplinary nature

As shown the schema at the beginning of §1.1, the ambitious vertical structure of this project requires a uniquely diverse range of expertise: from **Software Engineering & Formal Methods** at the high level, through **Quantum Computation** and logic at the mid-level, down to quantum **Systems Architecture** at the low-level. Just as AZX itself is an intermediary, this project unites those working in quantum information theory from logical and pure mathematical perspectives with those working on practical error correction, quantum hardware, and more generally programming language design and system engineering. It thus provides a unique opportunity for theoretical insight to inform future technology, and for technological problems to drive future theory. We will promote these cross-disciplinary interactions by a number of our planned activities, including holding a summer school which will provide both introductory tutorials and more advanced material on the range of techniques and methods which will be used and developed in the project, in a form accessible to both computer scientists and physicists from a wide range of backgrounds.

2 IMPACT

2.1 Expected impacts

AZX significantly advances the state-of-the-art across four of the five expected impacts.

Develop a deeper fundamental and practical understanding of systems and protocols for manipulating and exploiting quantum information — This project will take practical insights into the workings of diverse quantum technologies, along with fundamental techniques in quantum information processing, and embody them as software in the AZX toolchain. By embodying this expertise in software, practitioners can employ push-button optimisations and fault-tolerant transformations of programs without a deep understanding of the underlying theoretical techniques, effectively making these techniques available to a broader audience.

The AZX language will connect both high-level (algorithmic) and low-level (physical) representations, enabling the specifics of individual devices to be translated into constraints on the design of protocols. For instance, causal and topological structure is a crucial restriction on what can be processed in networked computing, and incorporating this into AZX will allow novel and efficient protocol design. The project also includes the ability to interface with current models of quantum computing (the circuit and one-way models), and will enable new hybrid procedures to be developed that include elements of both (as well as potentially new forms of information processing represented in AZX). The result of this project will be a step-change in our ability to describe how different quantum technologies store and manipulate quantum information, and to design protocols that use their specific abilities.

Enhance the robustness and scalability of quantum information technologies in the presence of environmental decoherence — AZX will provide a robust and scalable intermediate representation for quantum programs, which will help minimise the resource requirements in quantum technologies. AZX is also a practical tool to manage and track the resources required to realise operations on different hardware platforms. The annotation layer of AZX can model environmental noise and error rates of the target platform, allowing the compiler to make provisions to minimise decoherence as the program runs. In particular, the annotation layer of AZX can be used for fine-grained resource management for lower levels of error correction as well as idealised quantum memories, allowing integration of compilation and protection of coherence. AZX will enhance the development of error correction that is tailored to specific devices. Individual noise models and error propagation will be encoded in the language through the annotations, which can then be used for optimisation of error correction procedures. Using AZX as a design tool, specific error correction protocols can be developed for different devices, customised to the different noise models. These models will be flexible as the devices get larger, ensuring scalability of robust devices. Networked scalability can be optimised for, as well as different topologies, by encoding timing and spacial constraints in the language. As AZX will be a common representation, hybrid devices can also be optimised for. Error correction or mitigation strategies can be developed across multiple devices acting in tandem. Modelling error correction in AZX will thus enable the design of new error correction procedures, optimisation of existing ones, and give a mutually-intelligible language for error correction theorists and device technologists.

Identify new opportunities and applications fostered through quantum technologies, and the possible ways to transfer these technologies from laboratories to industries — The retargetable AZX system will make it easy to support new quantum devices, thus making the latest developments in quantum technology available to all academic and industrial users, and maximising the return on investment in quantum computing. Our consortium includes an industrial partner (Bull) to help ensure the industrial relevance of our work. We also have further industrial figures on the advisory panel. With AZX as a common intermediate language, high-level quantum languages and protocols can be designed without needing to know the underlying hardware. This will streamline the production of quantum software, opening it up to individuals and companies with limited prior knowledge of quantum computing. Quantum hardware will also be more accessible, both in academia and industry. Individual developers will not need to know the entire architecture, as different elements can be specified fully in AZX and then integrated into a large or hybrid device. This will accelerate the widespread commercial and academic development and exploitation of quantum technology.

Enhance interdisciplinarity in crossing traditional boundaries between disciplines in order to enlarge the community involved in tackling these new challenges — AZX will connect the entire range of knowledge involved in building quantum technologies, from experimental and theoretical Physics, through to quantum computing theory, and on to formal methods of Computer Science. All of these are needed to develop the language and its applications. Developing AZX is a fundamentally interdisciplinary task, and the resultant language will itself be a common method of communication between different disciplines. This opens the prospect of an acceleration in the development of quantum algorithms in a way which can then be easily ported to many different hardware platforms. For example, algorithm and protocol designers will not need to interface directly with quantum technologies: the AZX layer does all the compilation and optimisation necessary. This will allow the integration of quantum computing into mainstream Computer Science, and so the easy importing of tools (for example, techniques for optimisation or verification) that have been developed over many years. By aiding the development of intuitively accessible programming languages, AZX will also make quantum technologies accessible to a broader range of users and developers. End-users outside quantum physics and computer science will be able to build protocols for use in their own field that do not require them to understand the physical action of the hardware.

The advent of quantum computation, and the diverse set of skills needed to bring an idea from algorithm to implementation, has shown the limitations of traditional subject boundaries. The breadth of expertise of this consortium, and its thematic focus on developing a common language and methodology from quantum technologies will help overcome these limitations within the project and in the wider community.

2.2 Dissemination, exploitation of results, communication

Dissemination. The primary means of dissemination will be by publishing our results in leading journals and conferences, with a strong preference for open access venues. (We note that in computer science, the highest impact publication venues are conferences with published proceedings.) We will target:

- Specialist quantum information venues: *Quantum Information and Computation* (QIC), *Quantum Information Processing* (QIP), *Theory of Quantum Computation* (TQC), and *Quantum Physics and Logic* (QPL).
- Mainstream computer science venues: the *Journal of the ACM* (JACM), *Logic in Computer Science* (LiCS), *Principles of Programming Languages* (POPL), *Automata, Logic and Programming* (ICALP), *Tools and Algorithms for the Construction and Analysis of Systems* (TACAS).
- Mainstream physics journals: *Physical Review Letters* (PRL), *Physical Review A* (PRA), the *New Journal of Physics* (NJP), and *Communications in Mathematical Physics* (CMP), and the recently established open access journal *Quantum*.

The consortium members have a strong record of publishing in these leading venues. Other venues will targeted opportunistically in order to achieve the most timely publication of our results.

We plan three annual workshops, which will be open to any interested parties. The final workshop will include a school aimed at PhD students and potential end-users in industry. We allocate significant budget for student bursaries to maximise participation.

Exploitation of results. We propose several direct and indirect routes to exploitation of our results. Firstly, our consortium includes an industrial partner (Bull) and members of the NQIT project (Abramsky, Benjamin, de Beaudrap). With Bull, our work will be incorporated into state of the art products for simulation of quantum systems. With NQIT, we will provide a programming framework for the networked quantum computer developed as part of that project. As part of this we will present the project results at the semi-annual NQIT Industry Forum events and the UK Quantum Technologies annual showcase. To further this aim, our postdoc in Oxford will spend 10–20% of their time working closely with the NQIT project. In both cases, our work can be exploited directly by end-users.

In addition, we have also recruited a board of advisers (see below) including the European leaders in scalable quantum devices. These experts will participate in our annual workshops to help define requirements and ensure that our work can be used by their projects. The DiCarlo group (Delft) and Rigetti have shown particular interest in exploiting our work to assist with supporting the superconducting quantum devices they are developing.

Finally, we commit to produce public APIs (see D1.5 and D4.5) for the AZX system which will allow any programming language to generate code using our system, and make it easy to add support for future hardware targets. This will enable other projects to integrate AZX into their system. To further advance this aim, the software tools developed by our project will be released on an open-source basis with a permissive license (See § 3.6.)

Communication. To communicate of our work to a wider audience, and to take advantage of wide public interest in quantum technology, we will perform a variety of outreach activities. Firstly, we will adopt “open lab-book” research, publishing work-in-progress on a project wiki, and articles aimed at a general audience on a project blog.[5] To allow end-users to experiment with AZX, Bull will provide an HPC simulator with a publicly accessible front end, which will also be incorporated into the project website.

Beyond online self-publishing we will also pitch articles to magazines aimed at a general audience in several languages. We will specifically target:

- English: *Communications of the ACM*, *IEEE Computer*, *Physics Today*, *Nautilus*, *Quanta*, *Aeon*, *Ars Technica*, *New Scientist*.
- French: *La Recherche*, *Pour la Science*, and *Interstices*.
- Dutch: *Kijk Magazine*, *Quest*, and the dutch edition of the *New Scientist*.

Several of members of the consortium have had their work featured in these publications before. In addition, the AZX system itself will also be routinely presented in any industrial or public engagement events which touch on the software tools used by NQIT.

Advisory Board: In order to ensure the maximum impact, and to complement the expertise present in the consortium, we have recruited a board of advisors who will consult with the project. The board includes several leading European researchers working on scalable quantum devices; their input will help ensure that our software stack is able to support real devices, and they form a ready base of users for the final products. The board comprises:

- Leonardo DiCarlo (QuTech / Delft),
- Jonathan Pritchard (Strathclyde),
- Peter Selinger (Dalhousie),
- Andreas Wallraff (ETH Zürich),
- Philip Walther (Vienna),
- Will Zeng (Rigetti Quantum Computing).

Letters of support from the board members are attached at the end of this document.

[5] The success of *Graphical linear algebra* (graphicallinearalgebra.net, 217K visitors since 2015) demonstrates there is a clear audience for such works.

3 IMPLEMENTATION

3.1 Work plan

The work plan has four major scientific work packages, each focusing on a different *theme* within the project. The work packages will proceed in parallel, and all will have at least some activities throughout the length of the project. (There is also a fifth work package grouping administrative and organisational activities.)

WP1 is focussed on translating from HLLs into AZX, reflecting higher level programming constructs into AZX, and building a test suite of programs.

WP2 is about modelling the properties of different machines in AZX, and translating AZX to hardware.

WP3 develops the theory behind AZX and algorithms to realise the logical ideas.

WP4 applies these advances to the creation of useful quantum software, specifically focusing on optimisation and error correction.

Each work package is divided into more specific tasks, each of which is designed to deliver a particular piece of the project: some are theoretical results, some are software functions. Broadly speaking, the tasks are sequenced in order of technical difficulty so that experience gained on earlier tasks can be applied to the more difficult ones. There are strong interactions between the tasks, and early outputs of each WP will be used in later outputs of other WPs. (Details about the tasks are found in § 3.2.)

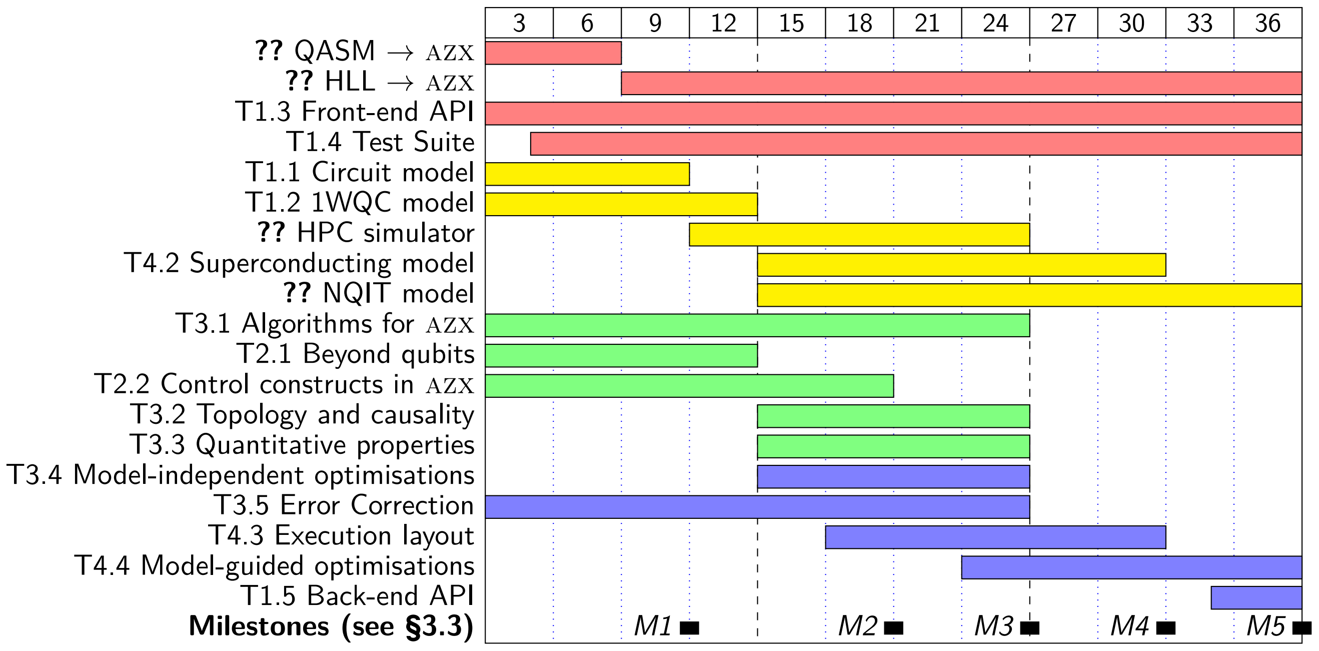


Figure 1: Approximate timings and durations of tasks (months)

The project is a single integrated whole, so there are many linkages between the work packages; these are displayed in Figure 2. As discussed in § 3.3, only some of these linkages are true dependencies, where later tasks rely on results of earlier ones. On the other hand, many tasks can influence and enhance each other as they run in parallel.

Our work plan consists of a balance of short tasks with concrete software deliverables (e.g. **??** and **??**) and longer term, more ambitious and open-ended tasks (e.g. T3.1 and T4.4) which can offer significant, but less predictable, step-changes in the state of the art.

Except for **??**, the tasks of WP1 are long and thin. That is, they are intended to work in parallel with the other WPs, with new features being integrated as they are developed.

The early tasks of WP1 and WP2 are quite practical and don't require much preparation to begin.

They will provide useful experience for the later tasks.

The first two tasks of WP3 build on a significant existing body of results and techniques for the ZX-calculus and rewrite theory in general. Hence, they can begin straight away. This will provide an ample source of theoretical work to do until the more implementation-oriented tasks T1.1 and T1.2 provide enough examples and use cases to feed into tasks T3.2 and T3.3.

The more challenging machine models of T4.2 and **??** are scheduled to begin in parallel with the more challenging theoretical tasks in WP3, anticipating a great deal of back-and-forth interaction between these two aspects of the project.

WP4 requires integrating and generalising many of the ideas of WP2 and WP3, so it is mostly scheduled toward the end of the project.

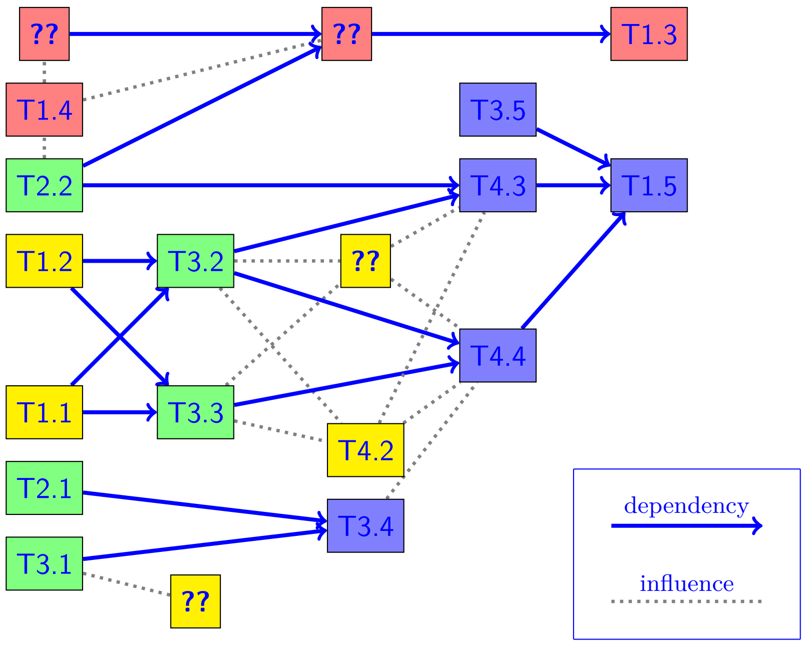


Figure 2: Dependencies and interactions between tasks

The allocation of staff to work packages is discussed in § 3.4 and §3.5. However, because of the integrated nature of the project, and the high degree of past collaboration among the consortium members, most tasks receive attention from the personnel of several sites. This degree of collaboration is a strong point of this project.

3.2 Work Packages

WP1	**A quantum compiler stack**	Start 1M	End 36M

Contribution of project partners						
Partner number	1	***2***	3	4	5	6
Total effort per partner (person*months)	0	0	0	0	0	0

Aim of the WP

This WP develops elements of ZX as in intermediate compiler language. This includes interfaces between ZX and up to known high-level quantum languages (HLL), and down to the device level of the target technologies. In the first instance we make contact between ZX and standard circuit and measurement-based models, in which all current quantum protocols are framed.

Tasks	
T1.1	**Idealised quantum circuits (M1–M9; Responsible 2; Involved: 1,3,4)** Translate an ZX term to an equivalent quantum circuit with ideal gates. This will require algorithms for discovering a suitable causal ordering on the ZX term, and for decomposing it into parts that represent individual gates. We will also consider circuits with constrained width, depth and/or layout.
T1.2	**Idealised 1-Way Quantum Computation (M1–M12; Responsible 3; Involved: 1,2)** Translate a ZX term to a runnable 1WQC [26] with ideal measurements and state preparation. Since every term of the ZX-calculus can be trivially translated to a one-way program, this task focusses on finding *deterministic* programs, subject to constraints on the size and topology of the underlying graph states, and limits on the number of measurement rounds. The output language will be the Measurement Calculus [8].
T1.3	**Open API for AZX terms (M1–M36; responsible 3; involved 2,4,5)** Develop an open API for the description of AZX terms. While largly technical, it is nonetheless essential as it will be used as interface to express the benchmarks of Task T1.4 to feed to the other WPs. The API will first be built upon the existing JSON representation for the ZX-calculus. It will be expanded with the features of AZX as they become available. This task is tightly linked with T2.2.
T1.4	**Open test-suite (M3–M36; responsible 3; involved 2,4,5)** Devise a test-suite of concrete instances of circuits and algorithms to rate success of other WPs. This includes the task of protocol extraction from current known HLLs. The tests will rate various aspects of algorithms, such as controls, manipulation of classical wires, scalability, depth of circuits, topologies of resources, *etc.* The test suite will be continually expanded as the project progresses.
T1.5	**Back-end API (M24–M36 Responsible: 5; Involved: 1,2,3,4)** Open specification of an API for back-end modules, facilitating third-party development of specifications of target architectures, providing the AZX compiler with extendability to arbitrary hardware platforms.

Deliverable	**Month of delivery**	**Title of deliverable**
D1.1	M6	Preliminary front-end for QASM, and initial AZX API
D1.2	M12	Preliminary benchtests of circuits and algorithms
D1.3	M15	Module for generation of 1WQC code from AZX terms
D1.4	M30	Advanced front-end for Quipper and one other HLL – updated API
D1.5	M36	Finalized API, test-suite and front-ends
D1.6	M36	API for back-end modules, including specification language for architectures.

WP2	Representation and Reasoning in zx	Start 1M	End 36M

Contribution of project partners						
Partner number	1	2	3	4	***5***	6
Total effort per partner (person*months)	0	0	0	0	0	0

Aim of the WP

We build the theoretical foundations for ZX as an IR. This includes extending the capabilities of ZX to represent universal algorithms, control flows, and . . . . We identify and develop annotations in ZX for relevant computational resources, which will then be used for subsequent machine (in)dependent optimisation protocols.

Tasks	
T2.1	**Beyond qubits and stabilisers (M1–M12; Responsible: 2; Involved: 1,3,5)** We will exploit recently achieved completeness results for ZX-calculus to get both convenient and powerful presentations for Clifford+T and fully universal families of diagrams. We will extend the ZX tensor formalism from the qubit domain to higher dimensions.
T2.2	**Control in zx (M1–M18; Responsible: 1; Involved: 2,3,5)** Support simple control flow at the level of AZX, making it a more suitable target for compiling from a high-level language. In particular, add support for repetition and recursive definitions of diagrams, e.g. for expressing and transforming regular families of circuits.
T2.3	**Resources (M1–M18; Responsible: 1; Involved: 2,3,5)** Placeholder for stuff that's primarily Belen's.
T2.4	**Other stuff (M1–M18; Responsible: 1; Involved: 2,3,5)** Further placeholder for stuff that's primarily Belen's.

Deliverable	**Month of delivery**	**Title of deliverable**
D2.1	M12	Deliverable.
D2.2	M15	Deliverable.
D2.3	M24	Deliverable.
D2.4	M36	Deliverable.

WP3	Machine-independent optimisation	Start M1	End M36

Contribution of project partners						
Partner number	1	2	***3***	4	5	6
Total effort per partner (person*months)	0	0	0	0	0	0

Aim of the WP

We develop practical logical and algorithmic techniques for transforming "abstract" ZX terms produced from a high-level program in ways which will be required by any practical compiler, and reasoning about their properties. Examples include: resource optimisation, adding error-correction, and execution layout. This lays the groundwork for machine-dependent optimisation in the next work package.

Tasks	
T3.1	**Reduction strategies, algorithms, and complexity (M1–M24; Responsible: 5; Involved: 1,2,3)** Develop new strategies for simplifying ZX-style tensor networks and reducing to (pseudo) normal forms, with the help of automated techniques such as Knuth-Bendix completion and graphical theory synthesis. Implement these strategies in software and give bounds on computational complexity.
T3.2	**Topological and causal constraints (M13–M24; Responsible: 2; Involved: 1,3,5)** Extend AZX language and tools to express and enforce: (1) topological constaints, such as nearest-neighbour connectivity of qubits and (2) causal/temporal constraints, such as sequential ordering of measurements and classically-controlled operations.
T3.3	**Quantitative Properties (M13–M24; Responsible: 2; Involved: 1,3,5)** Extend AZX language and tools to account for several kinds of numerical annotations, e.g. timing data related to performing operations, gate fidelities, channel fidelities, and decoherence over time. Allow these to vary by location and develop techniques to maximise exploitation of resources with varying fidelities. Provide formal methods to propagate these quantities from local to global properties.
T3.4	**Generic optimisations of ZX-terms (M12–M24; Responsible: 3; Involved: 1,4,5)** Use the results of task T3.1 to develop procedures to optimise ZX-terms, in a way which is applicable for families of circuits (e.g. Clifford, Clifford+T, CNOT+T, ...) as well as measurement-based quantum computations, independently of any particular hardware implementation or approach to fault-tolerance and minimising different possible metrics (such as total size, tree-width, or number of non-Clifford subterms such as T-gates).
T3.5	**Application of Error-Correction (M1–M24; Responsible: 5; Involved: 1,2)** Develop algorithms which rewrite abstract tensor networks to equivalent tensors in codeword space of a chosen error-correcting code. This may be combined with additional constraints from the annotation system, representing the implementation of a specific approach to fault-tolerant quantum computation, which is provided by the development environment.

Deliverable	**Month of delivery**	**Title of deliverable**
D3.1	M18	ZX language constructs for basic control flow, repetition and recursion
D3.2	M24	A library of general-purpose techniques and algorithms for simplifying ZX terms
D3.3	M24	Algorithms and heuristics for optimising ZX terms, including minimisation of T gate count
D3.4	M24	Routines for adding error-correction to ZX programs
D3.5	M24	An extended ZX language which expresses topological and quantitative properties, with associated reasoning techniques

WP4	**Machine-dependent optimisation**	Start 1M	End 36M

Contribution of project partners						
Partner number	***1***	2	3	4	5	6
Total effort per partner (person*months)	0	0	0	0	0	0

Aim of the WP

We import machine-dependent specifications to ZX terms, and use this to optimise algorithms further for specific hardware constraints. We focus on the silicon quantum dot devices developing in Grenoble, the ion traps developed in Oxford, and the superconducting devices accessible through CQC and partnership with IBM. This is the culmination of all previous work packages, and feeds back into them. The final result will be .... Also machine-dependent error correction here?

Tasks	
T4.1	**Grenoble quantum dots (M13–M36 Responsible: 2; Involved: 1,5)** We will model the quantum dot device being developed in Grenoble, and extract specific annotations for ZX that describe key elements of the architecture. This will include qubit layout on wafers, network connectivity, and timing and fidelity of potential entanglement links. A suitably annotated ZX term will then be translated to an executable sequence of hardware instructions – output language to be defined in collaboration with the team at LETI.
T4.2	**Oxford ion traps (M13–M30 Responsible: 5; Involved: 1,2)** In collaboration with the Oxford ion trap group and the NQIT team, we will design an output module which generates code for a realistic model of ion trap quantum computers, including qubit losses and leakage, gate timings, and circuit layout. Output language to be defined in collaboration with hardware experts at Oxford.
T4.3	**Formatting for target systems (M15–M30; Responsible: 2; Involved: 1,3,5)** Develop algorithms which, given a collection of constraints representing a machine model (c.f. T3.2, T3.3), re-writes AZX terms to a form which can be executed on that machine model.
T4.4	**Model-guided optimisation (M21–M36; Responsible: 5; Involved: 1,2,3)** Develop procedures to optimise ZX-terms subject to a machine model, within the confines of an annotation system for a particular hardware platform employing a particular approach to fault-tolerance, informed by techniques from T3.2 and T3.3.

Deliverable	**Month of delivery**	**Title of deliverable**
D4.1	M24	Deliverable
D4.2	M24	Deliverable
D4.3	M30	General purpose layout engine
D4.4	M36	Optimising compiler for AZX, suitable for compiling to Grenoble and/or Oxford architecture
D4.5	M36	API for back-end modules, including specification language for architectures.

WP5	Administation and Communications	Start M1	End M36

Contribution of project partners						
Partner number	*1*	2	3	4	5	6
Total effort per partner (person*months)	3	3	2	1	2	0

Aim of the WP

This work package collects general administrative activities and the organisation of the project meetings. All meetings will be organised as open scientific workshops, co-located where possible with a relevant conference.

Tasks	
T5.1	**Project administration (M1–M36; responsible 1; involved 2,3,4,5)** Global administration and project coordination.
T5.2	**Creation and maintenance of project website (M1–M36; responsible 1; involved 2,3,4,5)** As part of our commitment to open science, we will create and maintain a unified website for the project, including latest scientific works, downloadable software, end-user documentation, and popularising articles aimed at a general audience.
T5.3	**Kick off meeting (M1–M2; responsible 2; involved 1)** Project workshop to define state of the art, establish plans for the next year.
T5.4	**Midpoint meeting (M17–M18; responsible 5; involved 1)** Project workshop to disseminate initial results, evaluate progress and determine next steps.
T5.5	**Final meeting and school (M33–M36; responsible 3; involved 1,5)** Project workshop and school to disseminate project results.

Deliverable	Month of delivery	Title of deliverable
D5.1	M2	Website up and running
D5.2	M3	First workshop
D5.3	M18	Second workshop
D5.4	M36	Final workshop and school

Work package overview

Partner	WP1	WP2	WP3	WP4	WP5	**TOTAL**
1. Grenoble	0	11	19	15	3	48
2. LORIA	7	27	28	22	3	87
3. Oxford	32	12	20	18	2	84
4. CQC	5	18	0	5	1	29
5. Gdansk	3	12	13	21	2	51
6. Nijmegen	3	12	13	21	2	51
TOTAL	47	83	83	75	11	299

3.3 Management structure, milestones, risk assessment

Coordinator Coordination between sites and between work packages will be overseen by R. Duncan at the Strathclyde site, which will also handle the overall administration of the project.

Sites The project will be managed by a senior scientist from each site: C. Allouche (Bull), B. Coecke (Oxford), R. Duncan (Strathclyde), E. Jeandel (LORIA), and A. Kissinger (Nijmegen). They will track global progress to ensure milestones are reached, and facilitate collaboration across tasks at their individual sites. They will be in close contact throughout the project to assure coherence and concurrence of the activities at the different sites.

Work packages Each work package will be lead by a responsible PI who will coordinate research activity between sites to ensure that deliverables are met, achieve WP-specific objectives, and organise collaboration meetings as needed. **WP1**: B. Valiron (LORIA), **WP2**: S. Abramsky (Oxford), **WP3**: B. Coecke (Oxford), **WP4**: A. Kissinger (Nijmegen), **WP5**: R. Duncan (Strathclyde).

Software Integration Responsibility for the overall software architecture and integration of components will be shared between C. Allouche (Bull), A. Kissinger (Nijmegen), and B. Valiron (LRI), who make the technical decisions regarding interfacing of modules and global design.

Monitoring and advisory board Every six months there will be a meeting of both the work package leaders and the site leaders, either electronically (e.g. via Skype) or at a project event. At these meetings progress towards research objectives will be evaluated, and any new opportunities will also be discussed. These meetings will be organised by the coordinator.

To assist in monitoring and evaluating progress, we have recruited a board of external advisors who are both experts and potential end-users of the project. The advisory board consists of: Leonardo DiCarlo (QuTech / Delft University of Technology), Jonathan Pritchard (University of Strathclyde), Peter Selinger (Dalhousie University), Andreas Wallraff (ETH Zürich), Philip Walther (University of Vienna), and Will Zeng (Rigetti Computing). The entire project and advisory board (see §2) will meet once a year to evaluate progress, set priorities, and plan next steps.

List of milestones The milestones of the project are conceptually simple: at each milestone we will deliver a functioning piece of software. With each milestone, we add more, and more advanced, functionality. By delivering the software incrementally, we follow best practice in the industry: by regularly integrating parts from all work packages, we reduce risk and improve communication across the consortium.

Milestone	Delivery Month	WP involved	Title
M1	9	WP1,WP2,	Minimal QASM→QASM circuit optimiser
M2	18	All	Quipper→MBQC compilation pipeline
M3	24	All	Simulator back-end with parametric AZX support
M4	30	All	Integrate Delft back-end
M5	36	All	Integrate NQIT back end

Critical risks for implementation The project is overall quite high-risk, in the sense that what we propose is in the most part entirely novel, and might fail. The rewards for success would be correspondingly great. However, we have designed the project to survive the failure of may (or even most) of its tasks and still deliver value.

We mix low and high risk activities. Success in low risk activities (T1.3, T3.4, T3.5, T1.1, T1.2) will still deliver significant progress towards our overall objective. WP2 is the most critical; here we mitigate the risk by (i) developing experience on easier objectives before addressing more demanding ones and (ii) consulting our board of advisors to foresee problems. Due to its high dependence on other tasks, we consider WP4 to be highest risk.

Description of risk	Likelihood	WPs involved	Proposed mitigation measures
Can't hire suitable post-docs	Low	All	We have already identified several suitable candidates for each role.
NQIT or Delft machine info unavailable or not detailed enough	Low	WP2	(i) We have members of the consortium from NQIT, and letters of support from DiCarlo who agrees to help. (ii) We can approach other members of our board of advisors, or other friendly experimentalists. (iii) We can target an more abstract model of the machine
Some desired technique or algorithm isn't found	High	All	All tasks where this is a risk have been structured as a collection of related goals; if some part doesn't succeed then the finished system will be will have fewer features, or worse performance. However it's very unlikely that an entire task will fail in a way that jeopardises the project.
Software integration issues	Medium	All	(i) We will establish a common API in D1.1 to allow loose coupling of the software components (ii) We appoint the most experienced software developers in the consortium (Allouche, Kissinger, Valiron) to act as "integration triumvirate" and ensure the global design is good. (iii) As noted above, we will integrate often and deliver new features more than once a year.

3.4 Consortium as a whole

The members of the consortium are chosen to provide the best combination of skills to deliver this project, having been the pioneers of the fields that provide the foundations for this project, namely the invention of ZX-calculus and its further development, and also having developed applications of ZX-calculus to quantum technologies. Many members also have a long history of collaboration. They also contributed greatly to community building. Several members are part of are part of the NQIT Quantum Technologies Hub.[6] We now provide details on each of these.

Expertise on the theoretical aspects underpinning the project is provided by the project leader Duncan and Oxford site leader Coecke who jointly invented the ZX-calculus [5]. Kissinger, Perdrix, Horsman and De Beaudrap and many of their students are experts in its further development and use, for example in translation between different computational models [12, 17], error-correction [4] and lattice surgery [9]. Abramsky and Coecke are pioneers of high-level methods for quantum computing more generally. Duncan, Kissinger and Vicary have pioneered automation of diagrammatic reasoning (`quantomatic` and `Globular` respectively), which also will play a key role in this project.

We include pioneers in quantum programming languages (Valiron) and important contributors to the theory of MBQC (Perdrix, de Beaudrap, Benjamin, and Duncan) and quantum circuits (Jeandel). The consortium includes experts in classical simulation (Allouche), quantum error correction (Horsman, Benjamin), and in quantum architecture (Benjamin, de Beaudrap), whose expertise is bolstered by the members of our world-leading advisory board.

We include experienced co-ordinators of large multi-site projects (Abramsky, Benjamin, Coecke, Jacobs), and in particular, Benjamin and Coecke have led large-scale project in quantum computing.

Of utter importance is the alignment with Networked Quantum Information Technologies Hub (NQIT) at the Oxford site, which means that several members of the consortium have already direct expertise with interacting with quantum hardware (Benjamin, de Beaudrap, Horsman). The NQIT is the largest of the four Hubs in the UK National Quantum Technology Programme, a 270 GBP million investment by the UK government to establish a quantum technology industry in the UK. Concretely, the most important aspect is the fact that the modular architecture motivated using lattice surgery on surface codes for the logical operations, and that these are in effect ZX-operations [9]. This will certainly make the ambition here much more achievable.

Bull brings expertise in high performance simulation as well as industrial guidances in software.

The consortium has also been instrumental in community building, for example with the QPL conference series which now attracts well over 100 participants every year and approx. 75 paper submissions on foundational and structural research in the area of quantum computing. It also has organised several schools e.g. the QiCS School[7] and the CAP Spring School,[8] and a substantial talks archive is maintained.[9]

[6] nqit.ox.ac.uk

[7] www.cs.ox.ac.uk/people/bob.coecke/QICS_School.html

[8] www.cs.ox.ac.uk/ss2014/

[9] www.youtube.com/user/OxfordQuantumVideo

3.5 Description of the consortium

Partner 1 Project Coordinator	University of Strathclyde Department of Computer and Information Sciences

Expertise: Strathclyde University is one of the UK's leading technological universities, and is a member of all four of the UK's Quantum Technology hubs, collecting world leading expertise in all areas of quantum information. In the Computer and Information Science department, the Mathematically Structured Programming group contains internationally recognised leaders in the areas of programming language design and implementation (Conor McBride, Robert Atkey) and in categorical semantics (Neil Ghani, Clemens Kupke). In the Physics department, which was ranked No. 1 in the UK in REF 2014, the Optics Division contains numerous leading academics and researchers across all areas of theoretical and experimental quantum technology, including Andrew Daley, John Jeffers, Stefan Kuhr, Daniel Oi, Marco Piani, Jonathan Pritchard, and Luca Tagliacozzo. This unique range of expertise across all areas of project the makes Strathclyde an ideal site to host an interdisciplinary project like AZX.

Dr. Ross Duncan is Lecturer in Computer and Information Sciences, having previously held personal research awards as Chargé de Recherche (Université Libre de Bruxelles) and EPSRC Postdoctoral Fellow (University of Oxford). He is the co-inventor of the ZX-calculus (with B. Coecke) (1), and made key contributions to its theory (2,3) and its application to quantum computation (4). He initiated the `Quantomatic` project[a] and pioneered its use for the formal verification of quantum error correcting codes (5). He has organised many major events, most recently *Quantum Physics and Logic 2016*, and is co-organiser of the workshop series *Quantum Information Scotland*, and *Categories Logic and Physics*. ***Publications:*** *(1) B. Coecke and —. Interacting quantum observables: Categorical algebra and diagrammatics. New J. Phys, 13(043016), 2011. (2) — and K. Dunne. Interacting Frobenius algebras are Hopf. In LiCS 2016, ACM Transactions, 2016. (3) Bob Coecke, —, Aleks Kissinger, and Quanlong Wang. Strong complementarity and non-locality in categorical quantum mechanics. LiCS 2012. IEEE Computer Society Press, 2012. (4) — and Simon Perdrix. Rewriting measurement-based quantum computations with generalised flow. In ICALP 2010, Springer LNCS 6199. (5) Liam Garvie and —. Verifying the smallest interesting colour code with quantomatic. In Proceedings of QPL 2017, to appear.*

Role in Project:
As the coordinating site, Strathclyde will handle the overall management of the project. With expertise in the ZX-calculus, and graphical rewriting generally, Duncan will be involved in all the work packages, especially WP3, and tasks T1.2 and T3.5. The postdoc employed at Strathclyde will focus on WP3 (15 months), WP4 (12 months) and WP2 (9 months).

[a] `http://quantomatic.github.io`

Partner 2	University of Oxford Department of Computer Science

Expertise: The now well over 50 members Quantum Group at the Department of Computer Science, founded and led by Abramsky and Coecke has been the world-leading group in the development of high-level computer science methods for quantum computing. It is also the birthplace of ZX-calculus, where most of the completeness result where proven, and where `quantomatic` was mostly developed. Previously they coordinated the FP6 FET Open STREP QICS. The group is part of the NQIT Quantum Technologies Hub and has hosted 8 long-term EPSRC fellowships in the area of Quantum Computing. The Computer Science Department at Oxford is currently ranked within the world top 3.
Prof. Bob Coecke pioneered categorical and diagrammatic methods for quantum computing (1), and ZX-calculus in particular (with Duncan) (2). He is/has supervised approx. 40 PhD students, which include Ng and Wang who recently proved universal completeness of the ZX-calculus [22], and included Backens who proved stabiliser completeness [2]. He co-authored *Picturing Quantum Processes* (3), which presents diagrammatic methods for quantum computing to a broader audience. ***Publications:*** *(1) Samson Abramsky and —. A categorical semantics of quantum protocols. In LICS 2004. IEEE Computer Society, 2004. (2) Bob Coecke and Ross Duncan. Interacting quantum observables: Categorical algebra and diagrammatics. New J. Phys, 13(043016), 2011. (3) — and A. Kissinger. Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning. Cambridge University Press, 2017.*
Prof. Samson Abramsky is Christopher Strachey Professor of Computing and a Fellow of the Royal Society. He pioneered high-level methods for quantum computing (see (1) above). He received the IEEE LiCS test of time award, is a fellow of the ACM for his pioneering work in computing and was awarded the BCS Lovelace Medal in 2013.
Prof. Simon Benjamin is Professor of Quantum Technologies and Associate Director and head of the "Architectures" WP of the NQIT Quantum Technologies Hub. He is an expert in design and architecture of quantum hardware. ***Publications:*** *(1) Architectures and materials for robust and scalable quantum technologies, Nature Comm. 4, 1756. (2) N. H. Nickerson, J. F. Fitzsimons, and —. Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links. Phys. Rev. X, 4:041041, 2014.*
Dr. Sam Staton is Associate Professor and Royal Society University Research Fellow specialised in programming languages, and has expertise in quantum programming languages.
Dr. Jamie Vicary is a Senior Research Fellow specialised in graphical reasoning and automation thereof, in particular having produced the Globular software, an online proof assistant for higher-dimensional rewriting (http://globular.science).
Dr. Dominic Horsman is a visiting researcher at Oxford and co-inventor of the well-known lattice surgury technique for fault-tolerant computation. He has contributed greatly to applications of ZX-calculus, including the study of non-circuit quantum computational models, error-correction, and lattice surgery. ***Publications:*** *(1) Surface code quantum computing by lattice surgery. —, Fowler, Devitt, Van Meter New Journal of Physics 14 (12), 123011.* **Dr. Niel de Beaudrap** is a post-doctoral researcher involved in the NQIT project. He developed the first efficient algorithms to recover annotation systems to re-write MBQC procedures to the unitary circuit model. ***Publications:*** *(1) —. Finding flows in the one-way measurement model. Phys. Rev. A 77 (022328), 2008.*

Role in Project: As the group where ZX-calculus originated [5], Oxford will continue the fundamental further development of the calculus. Oxford is also the central institution in the NQIT project, which is the largest national quantum computation hardware project in the United Kingdom. The participation of researchers involved with NQIT, and their interaction with quantum technologies specialists in Oxford and elsewhere in Europe, will bring to the AZX project a wealth of expertise in quantum hardware technology projects. The Oxford postdoc will split their time equally between WP2, WP3, and WP4, including spending 10–20% of their time working closely with the NQIT project.

Partner 3	Université de Lorraine / CNRS / INRIA LORIA (UMR 7503) LRI (UMR 8623) (Université Paris-Sud / CNRS)

Expertise: LORIA (UMR 7503) is a research unit common to the CNRS, the University of Lorraine and Inria. Its missions mainly deal with fundamental and applied research in computer sciences. Bolstered by the 500 people working in the lab, LORIA is today one of the biggest research units in Lorraine, and one of the biggest computer science labs in France. The Inria project team Carte, led by Prof. Emmanuel Jeandel is expert in models of quantum computation, quantum information theory and in particular ZX-calculus.
Emmanuel Jeandel is Professor at Université de Lorraine, leader of the Inria project team Carte. He did a PhD in quantum computing, he is also an expert in dynamical systems (tiling, cellular automata). He contributed to the development of the ZX-calculus (2) and, together with Simon Perdrix and Renaud Vilmart, also at LORIA, they recently proved the completeness of the ZX-calculus for a universal Clifford+T fragment of quantum mechanics (3). ***Publications:*** *(1) —. Universality in Quantum Computation. In ICALP 2004, Springer LNCS 3142. (2) —, S. Perdrix, R. Vilmart, and Q. Wang. ZX-calculus: Cyclotomic supplementarity and incompleteness for Clifford+T quantum mechanics. MFCS 2017. (3) —, S. Perdrix, and R. Vilmart. A complete axiomatisation of the ZX-calculus for Clifford+T quantum mechanics. arXiv:1705.11151, 2017.*
Simon Perdrix is researcher at CNRS (CR1), having previously held positions at LIG (Grenoble) as a charge de recherche, and at OUCS (Oxford), LFCS (Edinburgh) and PPS (Paris) as Postdoc. He is an expert of ZX-calculus introducing several new axioms to the language (1,2,3). He is also an expert of measurement-based quantum computing, introducing in particular a graphical characterisation of determinism in the model (4,5). He leads the Quantum Computation French network (GT IQ at CNRS GdR IM) and is board of the CNRS Quantum Technology network (GdR IQFA). ***Publications:*** *(1) R. Duncan and —. Graph states and the necessity of Euler decomposition. In CiE 2009, Springer LNCS 5635. (2) — and Q. Wang. Supplementarity is Necessary for Quantum Diagram Reasoning. In MFCS 2016. LIPIcs, Dagstuhl, Germany, 2016. (3) R. Duncan and —. Rewriting measurement-based quantum computations with generalised flow. In ICALP 2010, Springer LNCS 6199. (4) D. E. Browne, E. Kashefi, M. Mhalla, and —. Generalized flow and determinism in measurement-based quantum computation. New J. Phys, 9(250), 2007. (5) M. Mhalla and —. Finding optimal flows efficiently. In Automata, Languages and Programming, In ICALP 2008, Springer LNCS 5125.*
Benoît Valiron (Assistant Prof. CentraleSupélec) from LRI (UMR 8623) is also included within the LORIA site. He obtained his Ph.D. In Mathematics at the University of Ottawa (Canada) in 2008. He is currently assistant professor at CentraleSupélec and researcher at LRI (laboratoire de recherche en informatique), Orsay. His research topics on interests include quantum computation, semantics of programming languages and models of computations, he is in particular co-inventor of the Quipper language. ***Publications:*** *(1) A. S. Green, P. L. Lumsdaine, N. J. Ross, P. Selinger, and —. Quipper: A scalable quantum programming language. PLDI 2013. (2) A. Scherer, —, S.-C. Mau, Scott Alexander, E. van den Berg and T. E. Chapuran. Concrete resource analysis of the quantum linear-system algorithm used to compute the electromagnetic scattering cross section of a 2D target. Quantum Information Processing 16:60, 2017 (3) —, N. J. Ross, P. Selinger, D. S. Alexander and Jonathan M. Smith. Programming the Quantum Future. Communications of the ACM, Vol. 58 No. 8, 2015. (4) M. Pagani, P. Selinger, —. Applying quantitative semantics to higher-order quantum computing. In POPL 2014.*

Role in Project: LORIA will develop the front-end compilation of HLLs into AZX terms. As one of the main contributors to the ZX-calculus and expert of 1WQC, LORIA will also play a key role in the development of AZX taking into account the different models of computation. The site provies expertise both in quantum programming languages and in the ZX-calculus. The postdoc at LORIA will spend half of their time on WP1, and divide the remaining time equally between the other WPs.

Partner 4	Bull ATOS Quantum Lab

Expertise: Bull SAS, part of ATOS group, is the European industrial leader in HPC and Big Data. It has extensive expertise of machine learning, in particular in the context of high performance computing. It has also a strong expertise in Quantum Computing, with a dedicated team of 10 FTE, with activities relevant to the project : quantum software, optimization of quantum circuits, high performance simulation of quantum algorithms, theoretical physics.
Cyril Allouche, PhD, is the director of Quantum Computing R&D of Atos. An industrial member of the High Level Steering Committee of the Quantum Flagship, he had strongly contributed to the definition of Pillar 3, "compute". Relevant product is the "Atos Quantum Learning Machine", first commercial product dedicated to quantum programming and high performance simulation of quantum algorithms

Role in Project:
With their expertise in high-performance computing Bull, will focus on the HPC simulation of AZX terms (**??**) however they will also contribute to the front end (WP1) and compilation techniques (WP4). With expertise in building large scale software Bull will also contribute the definition of the APIs that the AZX toolchain will use.

Partner 5	Radboud Universiteit Nijmegen Institute for Computing and Information Sciences

Expertise: Situated within the largest digital security group in the Netherlands (51 members), the Radboud Quantum Group offers strong expertise in the formal mathematical structures underpinning both quantum theory and classical programming languages. It consists of two full-time academics, one postdoc, and six PhD students. The Quantum Group furthermore maintains active relationships with the security group as a whole, including prominent members of the classical and post-quantum cryptography communities (e.g. Joan Daemen, co-author of the renowned AES cipher; and Peter Schwabe, whose post-quantum key exchange protocol NewHope was recently trialled by Google[a]).
Dr Aleks Kissinger is an Assistant Professor of Quantum Structures and Logic in Radboud's Institute for Quantum and Information Sciences (iCIS). For the past 10 years, he has been instrumental in the development of the diagrammatic approach to quantum theory, notably developing the theory of classical and quantum interaction for general process theories (1), classification of strong complementarity, and the ZW calculus (2). He also co-authored the canonical textbook for the field (3). Since 2006, he has also lead development on the Quantomatic tool (4), which serves as the platform for the software and automated techniques in this proposal. ***Publications:*** *(1) B. Coecke, C. Heunen, and —. Categories of quantum and classical channels. Quantum Information Processing, 15(12), 2016. (2) B. Coecke and —. The compositional structure of multipartite quantum entanglement. In Proceedings of ICALP, 2010. Springer LNCS 6199. (3) B. Coecke and —. Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning. Cambridge University Press, 2017. (4) Quantomatic: A Proof Assistant for Diagrammatic Reasoning. Proceedings of Conference on Automated Deduction (CADE) 2015. Springer LNCS 9195. quantomatic.github.io*

Prof Bart Jacobs is a Professor of Software Security and Correctness at Radboud, the holder of an ERC Advanced Grant in Quantum Computation, Logic, and Security, and a member of the National Cybersecurity Council. He has won several prestigious awards including the 2012 Huibregtsen award for Science and Society, the SURF Security and Privacy Award 2017, and is a recipient of the Dutch order of chivalry: Officer of Orange-Nassau. He has made major contributions to the formal theory of computation, including prominent textbooks on type theory (1) and coalgebra (2). He furthermore pioneered a new approach to modelling the logic of probabilistic and quantum systems, initiating the field of Effectus Theory (3). ***Publications:*** *(1) —. Categorical Logic and Type Theory. Number 141 in Studies in Logic and the Foundations of Mathematics. North Holland, Amsterdam, 1999. (2) —. Introduction to Coalgebra. Towards Mathematics of States and Observations. Cambridge University Press, 2017. (3) —. New directions in categorical logic, for classical, probabilistic and quantum logic. Logical Methods in Computer Science, Vol. 11, 2015.*

Role in Project:
The Radboud site will coordinate all aspects of the project dealing with automation and tool development, focussed primarily in WP4. It will furthermore contribute strongly to WPs 2 and 3, and in the case of WP2, will serve as an important point of contact with the extensive quantum hardware groups working within the Netherlands, notably the DiCarlo group in Delft. The *postdoc* working at this site will focus on the development of automated tools and techniques for quantum program transformation and engage with nearby quantum hardware groups to target short- and long-term applications of those techniques.

[a]Nick Stratt. Google is working to safeguard chrome from quantum computers. The Verge, July 2016.

3.6 Consortium agreement principles

The consortium commits strongly to Open Science. All scientific works produced in the course the project will be made available to the general public free of charge. All articles will be published in Open Access venues.

We aim at making our software available to the widest community so, with one exception, all APIs and source code for the software artefacts produced during the project will be freely available to the public, under a permissive Open Source license (e.g. the BSD license[10].)

However, the use of Bull HPC (see §3.7) requires proprietary Bull technologies for the HPC simulation back-end module (**??**). This module will not be open source. However, we will also provide a generic reference implementation of this module, which will be open source.

3.7 Significant facilities and large equipment available to the consortium to perform the project

Bull (Partner 4) will provide high performance computing (HPC) facilities which will be accessible by VPN to all project participants. The available HPC facilities include:

- One Atos bullion in-memory x86 server, 384 cores and 16 Tbytes RAM
- Nvidia P100 GPUs, as required.
- Altera Stratix FPGA boards, as required.

3.8 Link with ongoing projects

- **Oxford**: we have a very strong connection to the NQIT project, the United Kingdom's largest quantum hardware technology project. Benjamin is the leader of the NQIT "Architectures" work-package, and is an Associate Director of the NQIT project as a whole. Abramsky and de Beaudrap work on the "Quantum/classical interface and emulation" workpackage of NQIT, and are responsible for guiding the development of a compiler for the NQIT architecture. This gives the AZX project unique access to the expertise and information regarding the NQIT, and will help to ensure that the activities of the AZX project are directed at goals which are of direct practical relevance to practical quantum computational platforms.
- **Oxford**: we have a connection with the £2M Quantum Causal Structures JTF grant (with Coecke as PI), which includes the study of causal aspect of quantum computing.
- **Oxford**: ongoing work on quantum and probabilistic programming languages, funded through a Royal Society University Research Fellowship (Principled Foundations of Programming Languages, Staton), an EPSRC Project (EP/N007387/1: Quantum Computation as a Programming Language, Staton) and a grant from the Korean Government (Staton). That ongoing work is currently on the theoretical side of quantum programming languages and their semantics, and this proposal adds a new direction because it bridges the gap towards practice.
- **Strathclyde:** the Carnegie Trust awarded a PhD Scholarship to Mr Joseph Collins for the project "Infinite Dimensional Categorical Quantum Mechanics". This contributes to the later parts of T2.1.
- **Radboud**: ERC advanced grant "Quantum Computation, Logic, and Security" (QCLS) held by Jacobs. This 5-year 2.5M euro project will conclude during year 1 of the AZX project. Main outcomes: a new mathematical formalism, *effectus theory*, that covers Boolean, probabilistic, and quantum computation in a single framework and an associated Python tool EfProb for modelling probabilistic and quantum systems. The accrued tools and expertise will greatly enhance our effectiveness in providing concrete tools related to WP3 and WP4.

[10] `https://opensource.org/licenses/BSD-3-Clause`

3.9 Financial plan (1 page)

Personnel This is a large and multifaceted project, which will require significant work to deliver. A full-time post doc at each academic site is needed, under supervision of the site lead. In addition, At all the academic sites there is a large amount of time donated to the project by senior scientists with relevant expertise. At Bull, our industrial partner, several engineers will contribute to the project.

- *Post-doctoral researchers*: We request 36 months of salary for post-docs at Strathclyde, Oxford, LORIA, and Nijmegen. They will be hired as soon as possible by the site leads at each site.
- *Coordinator*: We request 10% contribution the Duncan's salary at Strathclyde to cover the time spent managing the project. This is reduced from 20% upon negotiation with Stratclyde. (Duncan will contribute an additional 8.4 months of time as a researcher, see below.)
- *Principle investigators and other named staff*: Spread across the four academic sites, more than 110 person-months of time will be contributed by the named staff on the project, all of which is supported by other sources. This means that almost 40% of the research effort of the project is funded from elsewhere.
- *PhD Students*: at LORIA, Renaud Vilmart will contribute approx 9 months to the project; this is funded from other sources.
- *Engineers*: At Bull, we request 29 person-months for research engineers, familiar with Bull's HPC systems, and experienced in making commercial software products. They will be essential in achieving our planned impacts.

Workshops Project workshops serve a key role in intra-project communication, dissemination, and outreach. We plan one workshop each year, in Oxford, Nijmegen, and Nancy respectively. Our entire advisory board will be invited to these workshops, which increases the cost beyond the usual expenses of venue hire and speakers' expenses. We have budgeted €6.5k for the first one[11] (which is planned to be smaller) and €12k for the second and third.

Travel and subsistence Since many of the personnel have expertise relevant to more than one work package, we request substantial budget for travel, for quantERA reporting meetings, for formal project meetings, smaller more frequent WP meetings, and also to present our work at conferences.

Equipment

- We request 25% of the cost of new server blades to upgrade our dedicated HPC facility. This facility will be available to the entire project via VPN, and will be used across several tasks, most crucially for **??**.
- We also request laptop computers for each of the postdocs, and replacement laptops for some staff at LORIA. These are necessary because of the frequent need to travel and work at another site and/or present work at conferences or workshops.

3.10 Ethical issues

No ethical issues foreseeable.

[11] Included in the Oxford site's consuambles budget.

References

[1] Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science: LICS 2004*, pages 415–425. IEEE Computer Society, 2004.

[2] Miriam Backens. The zx-calculus is complete for stabilizer quantum mechanics. *New Journal of Physics*, 16(9):093021, 2014.

[3] Dan E. Browne, Elham Kashefi, Mehdi Mhalla, and Simon Perdrix. Generalized flow and determinism in measurement-based quantum computation. *New J. Phys*, 9(250), August 2007.

[4] Nicholas Chancellor, Aleks Kissinger, Stefan Zohren, and Dominic Horsman. Coherent parity check construction for quantum error correction. *arXiv.org preprint*, 2016.

[5] Bob Coecke and Ross Duncan. Interacting quantum observables: Categorical algebra and diagrammatics. *New J. Phys*, 13(043016), 2011.

[6] Bob Coecke and Aleks Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press, 2017.

[7] Andrew W. Cross, Lev S. Bishop, John A. Smolin, and Jay M. Gambetta. Open quantum assembly language. `https://github.com/IBM/qiskit-openqasm/blob/master/spec/qasm2.pdf`, January 2017.

[8] Vincent Danos, Elham Kashefi, and Prakash Panangaden. The measurement calculus. *Journal of ACM*, 54(2), 2007.

[9] Niel de Beaudrap and Dominic Horsman. The ZX calculus is a language for surface code lattice surgery. arXiv:1704.08670, 2017.

[10] Ross Duncan. A graphical approach to measurement-based quantum computing. In Chris Heunen, Mehrnoosh Sadrzadeh, and Edward Grefenstette, editors, *Quantum Physics and Linguistics: A Compositional, Diagrammatic Discourse*, chapter 3. Oxford University Press, 2013.

[11] Ross Duncan and Maxime Lucas. Verifying the Steane code with Quantomatic. In Bob Coecke and Matty Hoban, editors, *Proceedings 10th International Workshop on Quantum Physics and Logic (QPL 2013)*, volume 171 of *Electronic Proceedings in Theoretical Computer Science*, pages 33–49, 2014.

[12] Ross Duncan and Simon Perdrix. Rewriting measurement-based quantum computations with generalised flow. In S. Abramsky, C. Gavoille, C Kirchner, F. Meyer auf der Heide, and P. G. Spirakis, editors, *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Proceedings Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 285–296. Springer, 2010.

[13] Liam Garvie and Ross Duncan. Verifying the smallest interesting colour code with quantomatic. In *Proceedings of QPL 2017, to appear*, 2017.

[14] Stefano Gogioso and Aleks Kissinger. Fully graphical treatment of the quantum algorithm for the hidden subgroup problem. *arXiv.org*, (1701.08669), 2017.

[15] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. Quipper: A scalable quantum programming language. In *Programming language design and implementation (PLDI'13)*, volume 48 of *ACM SIGPLAN Notices*, pages 333–342, 2013.

[16] Thomas Häner, Damian S. Steiger, Krysta Svore, and Matthias Troyer. A software methodology for compiling quantum programs. *arXiv.org*, (1604.01401), 2016.

[17] Clare Horsman. Quantum picturalism for topological cluster-state computing. *New J. Phys.*, 13(095011), September 2011.

[18] Emmanuel Jeandel, Simon Perdrix, and Renaud Vilmart. A complete axiomatisation of the ZX-calculus for Clifford+T quantum mechanics. *arXiv preprint*, 2017.

[19] Aleks Kissinger and Vladimir Zamdzhiev. Equational reasoning with context-free families of string diagrams. In Francesco Parisi-Presicce and Bernhard Westfechtel, editors, *Graph Transformation*, volume 9151 of *Lecture Notes in Computer Science*, pages 138–154. Springer International Publishing, 2015.

[20] Aleks Kissinger and Vladimir Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning. In P. Amy Felty and Aart Middeldorp, editors, *Automated Deduction - CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, pages 326–336. Springer, 2015.

[21] Mehdi Mhalla and Simon Perdrix. Finding optimal flows efficiently. In *Automata, Languages and Programming, Proceedings of ICALP 2008*, volume 5125 of *Lecture Notes in Computer Science*, pages 857–868. Springer, 2008.

[22] Kang Feng Ng and Quanlong Wang. A universal completion of the zx-calculus. arXiv:1706.09877, 2017.

[23] Naomi H. Nickerson, Joseph F. Fitzsimons, and Simon C. Benjamin. Freely scalable quantum technologies using cells of 5-to-50 qubits with very lossy and noisy photonic links. *Phys. Rev. X*, 4:041041, Dec 2014.

[24] Jennifer Paykin, Robert Rand, and Steve Zdancewic. QWIRE: A core language for quantum circuits. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, POPL 2017, pages 846–858, New York, NY, USA, 2017. ACM.

[25] Timothy J. Proctor, Erika Andersson, and Viv Kendon. Universal quantum computation by the unitary control of ancilla qubits and using a fixed ancilla-register interaction. *Phys. Rev. A*, 88:042330, Oct 2013.

[26] R. Raussendorf and H. J. Briegel. A one-way quantum computer. *Phys. Rev. Lett.*, 86:5188–5191, 2001.

[27] Damian S. Steiger, Thomas Häner, and Matthias Troyer. ProjectQ: An open source software framework for quantum computing. *arXiv.org*, (1612.08091), 2016.

[28] R. Versluis, S. Poletto, N. Khammassi, N. Haider, D. J. Michalak, A. Bruno, K. Bertels, and L. DiCarlo. Scalable quantum circuit and control for a superconducting surface code. *arXiv.org*, (1612.08208), 2016.

[29] Dave Wecker and Krysta M. Svore. Liqui|>: A software design architecture and domain-specific language for quantum computing. February 2014.

[30] William Zeng. *The Abstract Structure of Quantum Algorithms*. PhD thesis, Oxford University, 2015.