CEN/CLC/JTC 22/WG 3 "Quantum Computing and Simulation"

Convenor: **PAUL Alexandra MME**

CryogenicSolidState_Draft09

Document type	Related content	Document date	Expected action
Project / Draft	Meeting: VIRTUAL 18 Mar 2026	2026-02-17	**COMMENT/REPLY** by 2026-03-18

Description

Dear members,

Please find attached the lastest draft of the Cryogenic Solid State WI.

Best,

CEN/TC XXX

Date: 20YY-XX

prEN XXXXX:20YY

Secretariat: XXX

JTC 22 WG3 Quantum Computing

Cryogenic Solid State Quantum Computing

Functional Requirement

Draft 09, 2026-02-17

CCMC will prepare and attach the official title page.

Contents

Page

European foreword

This document (prEN XXXX:20YY) has been prepared by Technical Committee CEN/TC JTC22/WG3 "Quantum Computing and simulation", the secretariat of which is held by XXX.

This document is currently submitted to the CEN Enquiry/Formal Vote/Vote on TS.

This document has been prepared under a Standardization Request given to CEN by the European Commission and the European Free Trade Association, and supports essential requirements of EU Directive(s) / Regulation(s).

[NOTE to the drafter: Add information about related documents or other parts in a series as necessary. A list of all parts in a series can be found on the CEN website: www.cencenelec.eu.]

Introduction

One of the many possible hardware architectures for quantum computing is “Cryogenic Solid State Quantum Computing”. This family of architectures include solutions based on superconducting qubits (like Transmons and Flux Qubits), semiconductor spin-qubits, topological qubits, artificial atoms in solids, etc. They have in common that their quantum devices should operate at very low temperatures in a cryostat, and that their operation is controlled from electronics outside the cryostat.

This Technical Specification elaborates on the functional descriptions and functional requirements of these architecture families.

1 Scope

This document describes the functionalities of modules for use in cryogenic solid-state quantum computers and associated functional requirements. It leaves further details about interfaces and quantification of requirements to other, future, CEN/TSs.

This document does not specify specific values, only functional requirements, and may offer informative examples that have been proven in practice. Functional requirements are mainly an enumeration of characteristics that are considered as relevant for future specification as well as a motivation why they are relevant.

Cryogenic solid-state quantum computers belong to an architecture family of which all members make use of a cryostat. The quantum device(s) within the fridge are usually controlled from outside by room-temperature control electronics, through a (huge) number of I/O channels. Examples of members within this architecture family are solutions based on superconducting transmons, superconducting flux qubits, semiconductor spin qubits, topological qubits and artificial atoms in solids.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

- CEN/CLC/TR 18202:2025 *"Layer model of Quantum Computing".*

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply

3.1 ISA
Instruction Set Architecture

3.2 QEC
Quantum Error Correction

3.3 FTQC
Fault-Tolerant Quantum Computation

3.4 TWPA
Traveling Wave Power Amplifiers

3.5 AWG
Arbitrary Waveform Generator

4 Overview

The description of cryogenic solid state quantum computing comprises the hardware and software layers of *CEN/CLC/TR 18202:2025 "Layer model for quantum computing"*, as shown within the box in figure 4.1. It involves an architecture family of which all members make use of a cryostat. The quantum device(s) within the fridge are controlled from outside by room-temperature control electronics, through a (huge) number of I/O channels. Examples of members within this architecture family are superconducting transmons, superconducting flux qubits, semiconductor spin qubits, topological qubits and artificial atoms in solids.

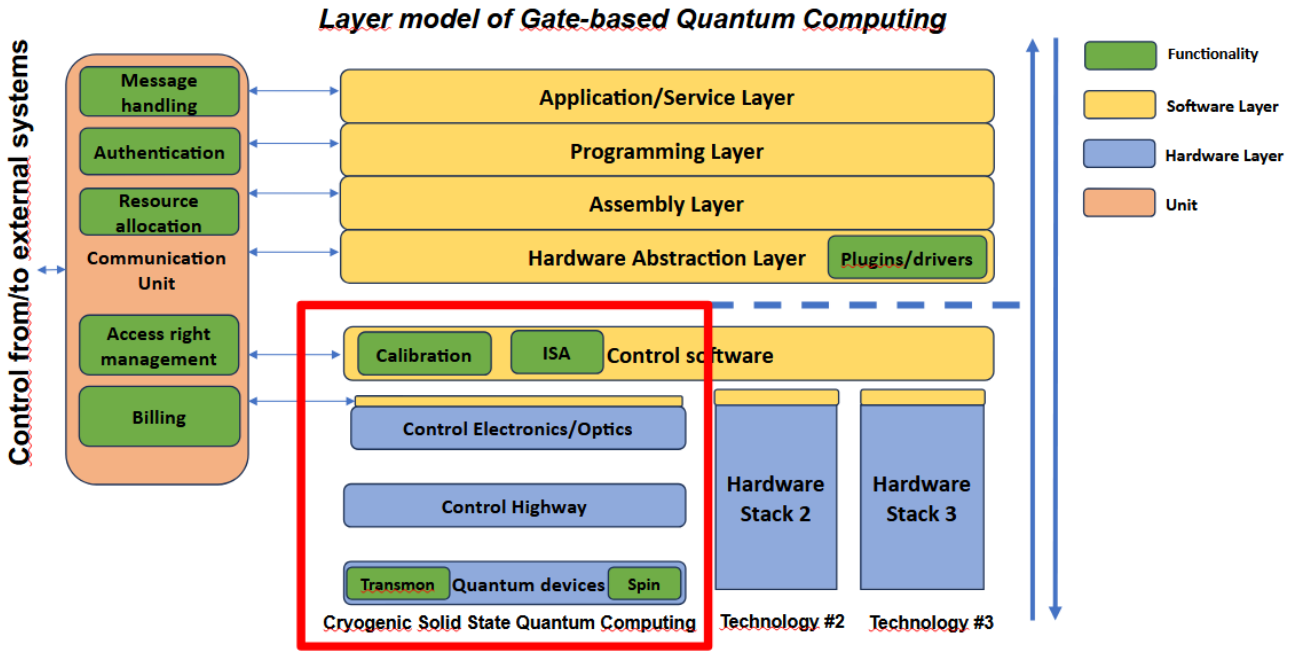

***Figure 4.1**- Overview of the layer model of quantum computing.*
Only the layers within the red box are in the scope of this document.

The description of this architecture family is organised as a stack of the following hardware and software layers, that are described in further detail in succeeding chapters :

- Control Software Layer
- Control Electronics Layer
- Control Highway Layer
- Quantum Devices Layer

A module is an implementation that may be constructed from (smaller) modules and components. It could offer the functionality of a single layer, of multiple layers, or just of a fragment of a layer. A module may also support different operating modes, such that it complies with the requirements of multiple members and/or multiple architecture families. As such, the functionality of a module may cover multiple layers and/or families and/or members.

5 Control Software Layer

The control software refers to the software systems and tools designed to manage, coordinate and optimise operations dictated by higher level languages. Thus, the software plays a crucial role in translating higher-level quantum assembly instructions into executable instructions that can be processed by quantum processors.

This layer may include an instruction set architecture (ISA), error correction means and calibration functionalities.

- **ISA** (Instruction set architecture) refers to a lower-level method of defining operations on a quantum computer. Instead of defining specific gates, this layer defines gates (or other instructions) as operations, using pulses pulsed for a certain time, on specific qubits. An example of an instruction set architecture is pulse level programming where a user can specify wave pulses on qubits instead of gates. This requires knowledge of the system's control equipment as well as the topology and qubit nature.
- **QEC** (Quantum Error correction) and **FTQC** (Fault-Tolerant Quantum Computation) refer to all low-level techniques to enable error-robust physical operations. Quantum error correction as a whole is a functionality distributed over various (higher) layers. The control software handles only low-level techniques, such as detection or simple corrections, partly autonomously and partly controlled from higher layers.
- **Calibration** refers to low-level methods to stabilise the hardware by continuous monitoring of hardware performance to maintain optimal operation.

5.1 Functional Descriptions

5.1.1 Instruction Set Architecture

The aim of an instruction set architecture (ISA) is to convert a sequence of machine-specific instructions from higher layers into commands for the control electronics, to control individual qubits. As such the ISA has full awareness of the underlying quantum hardware and its topology.

Due to the ISA's knowledge on the quantum hardware, it also has the responsibility of handling the execution timings and scheduling of individual instructions, such that higher layers should only know their sequence.

On input, the ISA may receive for instance instructions from higher layers to change the quantum state of qubits (gate-instructions, read-out qubits (measurement instructions)) or any other instructions to interact with all available qubits. These instructions can be fed to the ISA as for instance (binary) machine instructions, as (ascii) human readable instructions, or as function calls. Instructions intended for controlling one or two qubits may be fed one by one to the ISA, but it is more efficient if an ISA can handle many of them in parallel as a "vector" of instructions to interact with an ensemble of qubits simultaneously.

Higher layers may push these instructions into a buffer within the ISA each time the ISA signals to be ready for it. Alternatively, an ISA may also poll these instructions out from a buffer within higher layers each time the execution of a previous group of instruction has completed. This includes the polling of requests and instructions given by many users. In all cases, it requires a well-defined interface (API) with the above layer(s), as well as a well-defined instruction set language (such as OpenPulse [1] or Pulser [2]).

An ISA may handle gate-level instruction as well as pulse-level instructions. Both may be mixed in a single compilation pass for bypassing specific gate instructions, which can be parsed in the SDK by the user. Gate level instructions are considered to be any set of operations that can be parsed onto universal gate based quantum computers regardless of the hardware while pulse level instructions are operations that are heavily dependent on the system's physical architecture. The ISA will thus support instructions to specify the exact waveform of a pulse to be fired to a specific qubit, as well as an ensemble of pulses where each pulse has its own waveform and relative delay.

The common way of sending instructions to the ISA are via higher level layers such as the assembler or programming layer. Alternatively a user may be allowed by the communication module to access the control software layer directly or via the hardware abstraction layer, and supply ISA readable instructions directly.

On output, the ISA sends commands to the control hardware, to fire for instance pulses to qubits or to read-out their response via a measurement. This requires that the ISA is fully responsible of the timing of all these commands.

If a pulse is to be applied to a qubit, the ISA may calculate its characteristics on the fly, such as waveform / pulse-shape and magnitude. But it may also read predefined characteristics from a library created by other software units, stored somewhere in the control software layer or in the control electronics layer.

In all cases, it requires a well-defined interface (API) with the control electronics(s) as well as a well-defined command-set. Two example cases are described below:

- If a (production) user may only access the stack from the top of the assembly layer (or higher layers), the assembler compiler/interpreter will then convert assembly instructions into hardware-abstracted ones for the hardware abstraction layer. These instructions are then compiled into machine-specific code by the control software layer, where the instruction set architecture optimally schedules the user's program. It also determines the program's placement in relation to other users' tasks, ensuring efficient execution across the system.
- If a more dedicated user/designer may also access the stack from the top of the control software layer, he should be fully aware of the hardware-specific aspects of the quantum computer and generate machine-specific instructions for the ISA himself.

5.1.2 Quantum Error Correction and Fault-Tolerant Quantum Computation

The aim of quantum error correction (QEC) is to protect the quantum information stored in the quantum device and/or correct for errors in the quantum information caused by quantum operations or by decoherence. QEC is essential for fault-tolerant quantum computation (FTQC), i.e., performing a quantum computation on protected quantum information. Together, QEC and FTQC reduce the effects of decoherence, noise, faulty gates, faulty state preparation, and faulty measurements.

Different approaches to QEC include:

- Software-level QEC, emulating logical qubits from a larger redundant set of physical qubits, using redundancy to protect the quantum state and/or make errors detectable and correctable. It is sometimes called measurement-based QEC.
- Hardware-level QEC, protecting the quantum state on the hardware level, using intrinsic properties of the dedicated types of qubits.

- Circuit-level QEC, encoding quantum information redundantly and using quantum circuits to detect and correct errors.

In software-level and circuit-level QEC, logical qubits are encoded into a QEC code comprising redundant physical qubits. The minimum number of physical qubits per logical qubit depends on the code type (especially its "threshold") and the quality of the physical qubits, typically ranging from 10 to 1000.

Common examples of software-level QEC are based on stabiliser codes, such as the surface code. A stabiliser code comprises data qubits, which carry the redundant quantum information, and measurement qubits, which measure stabilisers of the code. A stabiliser measurement involves operations such as two-qubit gates between data and measurement qubits, optional single-qubit gates, and finally measuring the measurement qubit. This projects the physical state into one of several subspaces and yields *error syndromes*. These indicate whether the redundant information is consistent, or an error has occurred. A decoder uses syndromes to infer the most likely error subspace of the physical state. Based on this, the error-free state can be restored by applying single-qubit operations (typically Pauli operations). Importantly, stabiliser measurements do not reveal logical quantum information, and therefore preserve it.

In software-level QEC, the functions of defining the code, running stabiliser measurement cycles, decoding, and identifying correction operations are provided by a QEC unit implemented in higher layers (e.g., HAL). The QEC unit can comprise dedicated hardware (FPGA, CPU, GPU, …). The QEC unit connects to the ISA, which provides hardware information and measurement results to the QEC unit and executes stabiliser measurement cycles and correction operations. The interplay between QEC unit and ISA is detailed in the following steps:

- *Measurement*: A stabiliser measurement may involve resetting the measurement qubit and performing a sequence of gates and measurements. To perform such a cycle, the QEC unit instructs the ISA to execute either a circuit or a predefined macro instruction. The ISA then returns the results.
- *Decoding*: The error syndromes are input to the decoder in the QEC unit. The decoder outputs the most likely error subspace of the data qubits. This step is computationally intensive and may run on dedicated hardware.
- *Correction*: Based on the most likely error subspace, the QEC unit identifies correction operations (e.g., Pauli flips). These can be sent directly to the ISA for execution or collected in a Pauli frame and may be applied later as a cumulative correction or accounted for in the measurements.

The schedule of QEC cycles may be autonomously defined by the QEC unit, depending on qubit performance, code choice, and parameters (e.g., code distance). Alternatively, scheduling may occur at a lower level (e.g., by the ISA) or be user-defined. The entire process, including decoding, must run during QPU runtime, which imposes strict requirements on hardware speed, data throughput, and control software.

For performing logical operations on encoded quantum states, FTQC techniques are employed. Here, a logical gate or measurement corresponds to a sequence of physical operations applied to the physical qubits. These sequences must be fault-tolerant, i.e., designed so that an initial error does not spread into an uncorrectable one. The sequences may comprise magic state distillation or gate teleportation sub-circuits, among others.

An FTQC unit accepts input from a higher layer, such as a logical quantum circuit provided by the user. The FTQC unit translates each logical gate into a sequence of ISA instructions for physical gates and measurements. In some cases, these instructions may adapt dynamically based on intermediate measurement results from the QPU.

5.1.3 Calibration

Editor's Note:
Contributions are invited to fill-in this section

5.2 Functional Requirements

5.2.1 Instruction Set Architecture

The ISA is a functionality within the control software layer that can process instructions from higher layers, such as the HAL, and convert them into commands for control electronics to generate all kinds of pulses and measurements for qubits. It can process received instructions to initialise the setup, to inquire supported capabilities, to execute gates or measurements and to read-out intermediate results. However, there are several limitations that should be accounted for:

- Lane limitations: Execution often means the firing of pulses by the control electronics to the qubits, preferably multiple pulses simultaneously to different qubits. However the available hardware limits the number of pulses that can be fired simultaneously. When more qubits are to be controlled than this limit, a switching matrix is to be used after the pulse generators to reach all qubits of interest. These switches can offer so called "lanes" to connect a qubit to a particular pulse generator. To save hardware such a switching matrix may support only a restricted combination of lanes, and higher layers such as the HAL should be aware of such limitations. A convenient way to specify the supported connections between pulse generators and qubits through a switching matrix is by means of a so called "*lane matrix*".
- Adjacency limitations: Similar limitations do occur between qubits. Direct application of entangling operations between qubits can only be achieved between qubits that are adjacent to each other. It requires multiple steps to perform entangling operations between non-adjacent qubits, and therefore higher layers, such as the HAL, should be aware of such limitations. A convenient way to specify which qubits are edge-connected is by means of a so called "*adjacency matrix*".

To offer all required functionalities, the ISA should support the following groups of instructions:

5.2.1.1 Initialisation instructions

Initialisation instructions are to prepare the setup for quantum computing tasks. These instructions are typically sent at the beginning of a calculation sequence by higher layers, such as the HAL. But they may also be re-sent as often as needed. At least the following instructions should be supported by the ISA:

- "*def shapes*": construct a data structure with predefined wave shapes for pulses.
- "*def bases*": construct a data structure with predefined bases to measure individual qubits.
- "*def registers*": group the qubits into registers, and allocate indices to individual qubits. This grouping can be defined from higher layers, such as the HAL, or selected from one or more predefined register grouping.
- "*def meta instructions*": store a pre-defined sequence of instructions into a data structure, callable via a single instruction name, so that it can be played-back as if it was a new (user definable) instruction.
- "*set mapping*": construct qubit topological mapping.

- "*set lanes*": allocate numbering to lanes of pulsing channels.
- "*run calibration*": call a calibration from a set of predefined calibration procedures, to prepare a setup.

It might be possible that future systems will allow for a user-definable grouping of qubits to arbitrary registers. In that case the definitions of lane indices and mapping will also change.

5.2.1.2 Inquiring capability instructions

Inquiring capabilities is a mechanism for higher layers, such as a HAL, to identify relevant information about the setup, without performing any calculation. Most of these capabilities can be hard-coded in the software by the vendor, who has full knowledge about the hardware. But some of the capabilities can be a result stored in memory after performing initialisation instructions.

As a remark, some systems can modify their registers during runtime. When the system has this capability, it should support instructions on how they can modify the register.

5.2.1.2.1 Inquiring the organisation of qubits

Qubits can be grouped into "quantum registers" if they can be entangled, each with a unique index number to address the individual qubits. A system can support one or more of those registers, and if multiple registers are being used, each of them has a unique identifier to address the individual registers.

A quantum compiler or interpreter should have full knowledge of how the qubits are organised in order to optimize generated code. To prevent that each compiler is targeted only to a single system, it could start with inquiring these capabilities via the HAL, which in turn inquires the control software layer. The ISA should support at least the following capability instructions:

- "*Get register set*": Return a list with identifiers of each of the supported quantum registers.
- "*Get register width*": Return the number of available qubits for each individual quantum register.
- "*Get adjacency matrix*": Returns an "adjacency matrix" for each individual quantum register, to specify which qubits in the register are edge-connected. For instance, when a register has Nq qubits, then this adjacency matrix C has size Nq×Nq. The default of each element in this matrix is false, but if qx and qy are the indices of two adjacent qubits then C(qx,qy)=C(qy,qx)=true. Matrix C is therefore a symmetric matrix, since C(k,r)=C(r,k).
- "*Get lane matrix*": Returns a "lane matrix" for each individual quantum register, to specify for each pulse generator to which qubit it can be connected. For instance, when Np pulse generators can be connected to Nq qubits in that register, then the associated lane matrix L has size Np×Nq. The default of each element in this matrix is false, but if pulse generator px can be connected with qubit qy then L(px,qy)=true. Returning an empty matrix means that this limitation does not exist.

Different kind of qubits can be distinguished when inquiring system specifications, such as:

- Physical qubit: A noisy quantum system in which a two-dimensional Hilbert space can be encoded.

- Data qubit: data qubits are physical qubits used for storing and processing quantum information.
- Measurement qubits: physical qubits used for stabiliser measurements of quantum error-correcting codes.
- Communication qubit: physical qubits specifically designed to perform entanglements to other communication qubits on non-local registers. They are also entangled to other qubit types within the quantum computers and are primarily used to mitigate information for distributed quantum operations.

- Logical qubit: An error-corrected quantum system whose state lies in a two-dimensional Hilbert space.

5.2.1.2.2 Inquiring supported gates

A native gate is a gate that can be executed within a "single pulse interval" using one or several simultaneous pulses. If a gate requires multiple pulses in sequence, then it is called a compound gate. If a compiler has full knowledge about which gates are native and which are compound, then it can optimize a compiled program in terms of minimal execution time.

The ISA may also support shortcuts for commonly used (sequences) of native gates as if they where single gate. Those shortcuts are called primitive gates. An example may be the well-known Pauli gates that are implemented as (a sequence) of rotations. All primitive gates that are not native are assumed to be compound gates.

The ISA should support at least the following gate inquiry instructions:

- "*Get primitive gates*": Returns an identifier list of all gates that are understood by the ISA. This includes both single qubit as well as multi qubit gates. Examples are:
  - Rx(a), Ry(b), Rz(c), X, Y, Z, etc
  - CX, CZ, SWAP, CNOT, sqrtSWAP, etc
- "*Get native gates*": Return an identifier list of a subset of primitive gates that can be executed within a single pulse interval.
- "*Get gate matrix*": Returns for each primitive gate the associated matrix defining the operation.
- "*Get gate duration*": Returns for each primitive gate the operation time.
- "*Get lane size*": Returns the number of pulses that can be fired simultaneously to a selected set of qubits.

TODO. The "*Get lane size*" instruction may be superfluous. A compiler should prevent such an overload by inquiring "*get lane matrix*". If that overload occurs, a fault should be raised by the ISA.

TODO: Elaborate on the need for possible instructions, that informs higher layers if a qubit is physical or not. If yes, it may be relevant for higher layers to know if it is a data, measurement, or communication qubit. Such an instruction might be identified as "*Get qubit properties*".

5.2.1.2.3 Inquiring qubit or register performance

Inquiring performance information is a mechanism to enable higher layers, such as the HAL, to extract information about the maximum circuit depth and other quality parameters. This means the number of time slices that can be executed before the calculation becomes unreliable. Examples are specifying the error of primitive gates, expressing a fidelity for each gate, or simply specifying a maximum circuit depth. This value is related to coherence time of the implementation and other performance parameters of each connection between qubits. Details about these mechanisms are still to be elaborated, but they should facilitate higher layers, such as the HAL, to identify one or more of the following performance parameters:

- "*Get gate performance*" such as:
  - Coherence times, e.g. (T1, T2), where "T1" refers to state decoherence time and "T2" refers to phase decoherence time.
  - Fidelity for certain gates.
  - Qubit pulse error rates.
  - Read out error to indicate how accurate a measurement can be
- "*Get instruction duration*": List of durations for each identified instruction (operations, pulses, read out times, etc.). Note that this is a generalisation of the afore mentioned "get gate duration" instruction
- "*Get qubit properties*": List of properties of each identified qubit e.g. (T1, T2, qubit type, position coordinates, etc.)

5.2.1.3 Execution instructions

Execution instructions are instructions that are meant to change the state of qubits for calculation or measurement purposes, or to prepare for such an instruction.

preparation/initialisation level:

- "*Reset calculation*": call a procedure for re-calibration the setup from a set of pre-defined calibration procedures. One should only adjust for small changes in the setup due to drift or other imperfections of the setup, which can be done in a reasonably short period of time. As such it does not refer to a full calibration as described in "initialisation instructions".
- "*Reset qubits*": Change the states of all qubits of a selected register into predefined ones, which can be identified as their "|0>" state.
- "*Reset flags*": Set all binary flags to "false" of a selected conditional register.
- "*Set flag X*": Set an individual binary flag "X" of a selected conditional register to a selected value.

pulse level:

Control electronics can fire multiple pulses in parallel, and before they are actually fired by the electronics the ISA must support instructions to prepare this for each qubit individually. However the available hardware limits the number of pulses that can be fires simultaneously. When more qubits are to be controlled than this limit, a switching matrix is to be used to reach all qubits of interest. They can offer so called "lanes" to connect a qubit to a particular pulse generator. To save hardware such a switching matrix may support only a restricted set of lanes, and higher layers such as the HAL should be aware of such limitations.

- "*select qubits*": Prepare for the connection of a selected set of qubits with available lanes, through which pulses will be transported thereafter The actual connection may be delayed until a "wait" or "fire" instruction is handled.
- "*select pulses*": Prepare for each lane of interest a predefined pulse that is to be transported thereafter. The actual firing of pulses may be delayed until a "fire" instruction is handled.
- "*fire pulses*": Fire an ensemble of the selected pulses through the selected lanes for a specified duration, when the selection of lanes and pulses is completed. This instruction will actually execute a specific native gate by changing its state into a specific phase. During this duration, the ISA can prepare for other lanes and pulses, to be used for firing a next ensemble of pules.
- "*wait*": impose a selected amount of delay before a next ensemble of pulses can be fired. During this duration, the ISA can prepare for other lanes and pulses, to be used for firing a next ensemble of pules.

gate level:

An ISA has full knowledge on what (sequence of) lanes and pulses are needed to execute primitive gates. This frees higher layers of the burden to know the exact implementation of lanes and required pulses. As such, gate execution instructions can be regarded as higher in abstraction than pulse execution instructions.

- "*select gates*": Prepare for an ensemble of gates to be executed simultaneously on selected qubits. The actual connection may be delayed until a "wait" or "fire" instruction is handled. This includes the selection of potential conditional gates.
- "*fire gates*": Fire a sequence of simultaneous pulse-delay combinations to the selected qubits in order to execute the selected native gates.

measurement level:

- "*select bases*": prepare for an ensemble of selected qubits the pulse from the "basis library", that are needed to measure the state of these qubits in a specific basis.
- "*fire measurements*": Fire a sequence of simultaneous pulse-delay combinations to measure the selected qubits in a specific basis, detects their response and store the results in the associated conditional bits.

5.2.1.4 Fault handling instructions

These instructions are about classical means to handle all kinds of "classical errors". But to avoid confusion with "quantum errors" the word "fault" is consistently used here to denote those "classical errors".

Each time a fault is raised, the setup should increment the associated fault counter. This enables higher layers, such as the HAL, to read-out these fault counters and to perform proper fault handling. Each time a fault occurs, the ISA can raise an interrupt to higher layers to inform that a fault has occurred. These interrupts can be masked individually to prevent then being raised.

Examples of such fault counters are:

- Raise exceeding pulse duration time.

- Raise parameter overflow/underflow faults (when phase is out of bounds, index of libraries, lanes or qubits are wrong, etc.).
- Raise pulse timing faults (in the event that pulses are too close together, pulsed at too short times, etc.).
- Raise runtime faults (in the event that a channel is used to pulse a qubit that is not contained in channel lane, etc.).
- Raise memory faults (instruction is too long).
- Raise connection faults (if during an active hybrid session server disconnects).
- Raise adjacency faults, if direct entanglement is requested between two non-adjacent qubits.
- Raise lane faults, if a pulse has to be send to a qubit while the associated lane cannot be build-up.

The ISA should therefore support readout and (re)setting instructions of these fault counters and flags, including:

- "*Set fault mask all*": force for all faults that the ISA will raise an interrupt to higher layers when a fault occurs.
- "*Set fault mask X*": force for fault "X" that the ISA will raise an interrupt to higher layers when such a fault "X" occurs.
- "*Get fault last*": returns the identifier (or index) of the last fault being encountered.
- "*Get fault all*": returns an array with the content of all fault counters and flags.
- "*Get fault X*": returns the content of fault counter or flag "X".
- "*Reset fault all*": Resets all fault counters and flags to zero of false.
- "*Reset fault masks*": Resets all masks to false, so that no fault will raise an interrupt to higher layers.
- "*Reset fault X*": Resets fault counter or flag "X" to zero of false.

5.2.2 Quantum error correction and Fault-Tolerant Quantum Computation

5.2.2.1 Initialisation requirements

Quantum error correction requires means from the control software layer for its initialisation. This can be achieved via initialisation instructions supported by the ISA, such as:

- Instructions for inquiring the organisation of qubits, such as "*Get register set*", "*Get register width*", "*Get adjacency matrix*", or the types of the qubits – With these instructions, the QEC unit should request information on the organisation of the qubits from the ISA. The QEC unit should use this information e.g. to provide codes suitable for the organisation of the qubits. For instance, the QEC unit should consider which qubits are assigned as data qubits and which qubits are assigned as measurement qubits by the ISA. When the ISA does not provide this information, the QEC unit should assign the roles of the qubits in the code.
- Instructions for inquiring supported gates, such as "*Get primitive gates*", "*Get native gates*", "*Get gate duration*" – With these instructions, the QEC unit should request information about the supported gates. The QEC unit should use this information e.g. to provide stabiliser

measurement cycles or subcircuits for fault-tolerant logical gates and logical measurements adapted to the Quantum Processing Unit.

- Instructions for inquiring qubit or register performance, such as "*Get gate performance*", "*Get instruction duration*", "*Get qubit properties*" – With these instructions, the QEC unit can request information on qubit or register performance from the ISA. The QEC unit can use this information e.g. to choose a suitable code and/or suitable code parameters (e.g. code distance) for the given gate performance.
- Initialisation instructions, such as "*Set pre-defined instruction sequences*" – The ISA should be initialised with a QEC code specific pre-defined instruction effecting a sequence of native or primitive gates and measurements (e.g., a pre-defined instruction for effecting a stabiliser measurement, or for subcircuits of fault-tolerant quantum computation such as magic state distillation or gate teleportation schemes).

5.2.2.2 Execution requirements

Quantum error correction requires means to change the status of qubits. This can be achieved via execution instructions supported by the ISA, such as:

- Preparation/initialisation level instructions, such as "*Reset qubit*" – The ISA should be able to reset a qubit to the |0> state at any time, e.g., for initialising a measurement qubit for the next stabiliser measurement as instructed by the QEC unit or if necessary for performing FTQC subcircuits such as magic state distillation or gate teleportation schemes as instructed by the FTQC unit.
- Gate execution instructions, such as "*select gates*", "*fire gates*" – The ISA should be able to select and fire gates to perform correction operations or operations necessary for performing a stabiliser measurement as instructed by the QEC unit and gate sequences for performing logical gates or FTQC subcircuits such as magic state distillation or gate teleportation schemes as instructed by the FTQC unit.
- Measurement instructions, such as "*select bases*", "*fire measurements*" – The ISA should be able to perform a measurement on the measurement qubits for performing a stabiliser measurement as instructed by the QEC unit or for performing logical measurements or for FTQC subcircuits such as magic state distillation or gate teleportation schemes as instructed by the FTQC unit.

5.2.2.3 Latency requirements

Quantum error correction requires low latency responses from qubits through all lower layers of the stack. This is essential to perform real-time correction of quantum errors.

- The ISA may facilitate this for the control software layer by speed-optimising multi-sequence instructions that have been defined via initialisation instructions such as Set pre-defined instruction sequences.
- The control electronics layer may facilitate this via hardware support of multi-sequence instructions. It may minimise instruction overhead by firing a predefined sequence of pulses through a single command.
- The control highway layer may facilitate this via short signal channels between electronics and qubits to minimise propagation delay of signals in these channels.

5.2.3 Calibration

Editor's Note: Contributions are invited to add text on calibration

6 Control Electronics Layer

This hardware layer covers all electronics for generating, receiving, and processing microwave, RF and DC signals to control and read-out qubits via the control highway layer. Implementations may make use of routing/switching and/or multiplexing of control signals at room temperatures.

6.1 Functional Descriptions

The control electronics is typically a plurality of hardware functionalities, ranging from oscillators, arbitrary waveform generators (AWG), voltage and current sources, mixers, switches, filters, signal detectors, voltage and current meters, attenuators, etc. These hardware functionalities may be distributed over multiple stand alone devices, sometimes from different vendors, and may also be grouped together into hardware modules, which may be wired together into a coherent system for controlling qubits.

As shown in Figure 4.1, this hardware layer includes a thin sub-layer for Command Transformation in order to command the control hardware in a uniform manner. This sub-layer transforms these common commands into implementation-specific (proprietary) commands, which are fully tailored to the involved control electronics.

Figure 6.1 illustrates how multiple hardware modules can interface with a shared Command Transformation sub-layer. Each module may rely on its own proprietary software driver, which integrates into the sub-layer to provide a uniform interface to the higher layers of the stack.

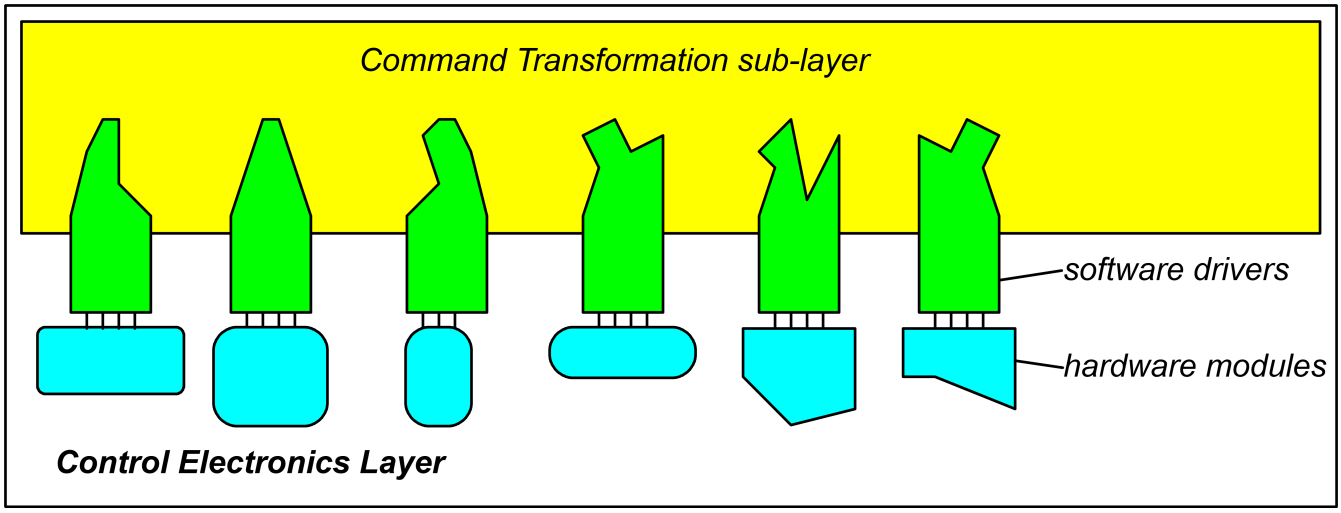

Fig 6.1 *Hard and software within the control electronics layer*

The functionality of this Command Transformation sub-layer goes far beyond a simple translation of commands. Since a lot of signals are to be generated and detected in parallel, and since many hardware devices may be involved in the handling of a single signal, a more intelligent approach has to be followed.

Higher layers will therefore communicate with the hardware modules via so called sequencers. These are local processing units for sending a sequence of proprietary commands to a variety of hardware devices, in order to fulfil a single task.

Each sequencer operates independently, comparable with a core in a CPU. Similar to how a CPU processes binary instructions, a sequencer executes micro-programs written in a proprietary low-level instruction language. Each sequencer has memory to store micro-programs and data. These micro-programs can be stored directly in binary format or compiled to binary via a dedicated (proprietary) assembly language.

The operation of sequencers usually follow the same pattern:

- prepare a sequence definition
  - allocate memory for acquisition
  - define/generate desired waveform
  - define the micro-program
- upload sequence definition to a target sequencer
- bind the sequencer to the desired inputs and outputs
- play the sequence
- read acquisition
- close and free the sequencer

A full system has many sequencers, and they all have much in common. But additional functionality can be build-in into more dedicated sequencers, such as:

- Control sequencers, to govern the generation of signals.
- Readout sequencers, to govern the generation of signals and acquisition of these signals.
- Timetag sequencers, to govern the output and acquisition of trigger events.

In principle, sequencers may be emulated to run their micro-programs directly within the command transformation sub-layer on a common CPU. But it is recommended for performance reasons that each sequencer runs its micro-program on a dedicated FPGA (Field Programmable Gate Array), integrated within the various hardware modules.

Since the qubits and transmission characteristics of the setup will gradually drift over time, the pulses that are to be fed to the qubits are to adapt regularly to the changes in the setup. Therefore the waveform, attenuation, pre-distortion with filters, etc. has to be adapted accordingly. Higher layers above the control software layer must therefore be offered with the flexibility of reprogramming sequences through a common interface.

This interfacing with higher layers may be implemented by means of a stream of commands, which allows for remote controlling the hardware via the communication unit of the quantum computer. An alternative way of interfacing is by means of function calls. This is simpler to implement but set restrictions on the programming languages being used for implementing higher layers. Both approaches can offer a standardised interface.

The specification of this interface to higher layers is left for future standards.

Editor's Note:
Above is only a functional description of a "thin"- software sublayer, but a functional description of the hardware is also needed

Think of text and diagram(s) describing an AWG to generate a pulse, an oscillator to generate a carrier, and mixer to upconvert the pulse into a modulated carrier. The resulting signal should be injected into the control highway towards the qubits
Think also of text and diagram(s) describing two mixers, that are downconverting an input signals with the same carrier frequency, but one carrier is 90 degrees shifted toward the other. Both downconverted signals are amplified and further processed.

6.2 Functional Requirements

Editor's Note:
Contributions are invited to fill-in this section
Think of a customer that can buy control electronics from different vendors, and want to identify which solutions can meet his requirement. What are the parameters, or quantities that should feed this judgement. and to identify the "best" offered solution. Most likely, it will be a judgement based on the maximum signal level, output noise floor, supported frequency range, receiver sensitivity, supported command set, flexibility to define pulses, etc. So a section on functional requirements identifies and explains all meaningful parameters and quantities, without specifying any value for them (values will be left for future standards)

7 Control Highway Layer

The control highway is a hardware module for connecting qubits in a cryogenic environment with control electronics at room temperature. It includes all I/O channels (input-output) needed for transporting downstream signals to qubits up to microwave frequencies and for returning upstream signals carrying their response.

7.1 Functional Description of the Control Highway

The control highway facilitates the transportation of downstream and upstream signals between control electronics, operating at room temperature, and quantum devices, operating at cryogenic temperatures.

Figure 7.1 shows an example diagram of channels in a possible control highway dedicated to a quantum computer using transmons. The I/O channels of spin-qubit quantum computers may be different, but this example alone may be sufficient to get basic understanding of various functional requirements for future standardisation.

In this example, the I/O of each qubit is handled via three downstream channels: one for microwave control signals, another one for flux control and a third one for read-out. The response signals of two or more qubits may share a common upstream read-out channel to reduce the overall number of channels. Travelling wave amplifiers may be used for amplifying these response signals, and may need an extra TWPA pump channel for powering. As such, a 50 qubit transmon quantum computer may have 102 or more I/O channels.

The involved I/O channels may be build-up from a chain of building blocks, for instance from transmission lines, attenuators, directional couplers, low-pass filters, infra red filters, DC-blocks, superconducting sections, amplifiers, isolators, circulators as well as thermalization means and vacuum feed-throughs.

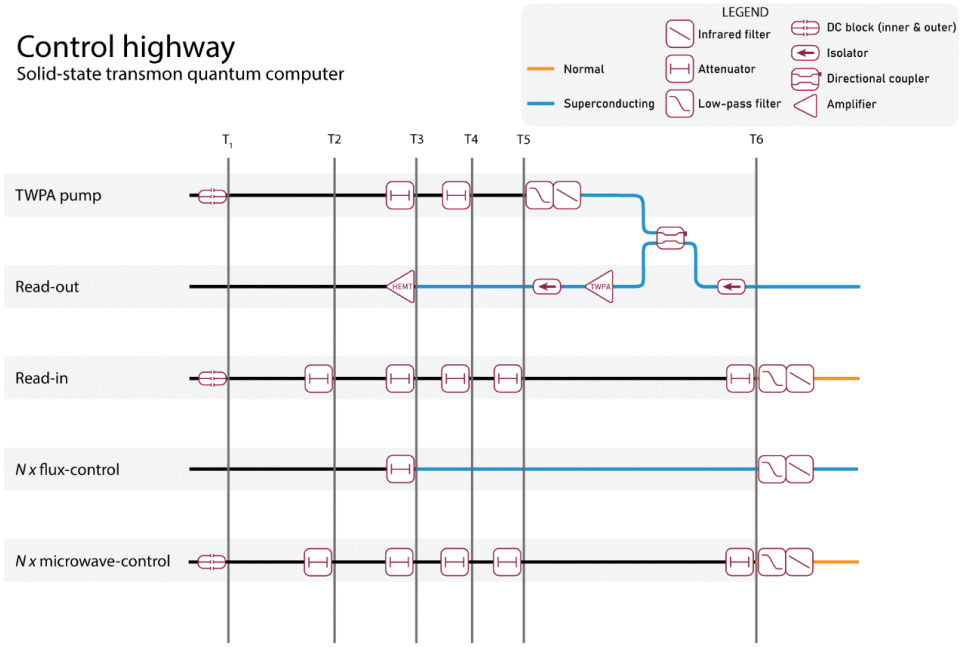

Figure 7.1 An example functional diagram of a control highway applicable to a particular transmon architecture.

A design of a typical implementation usually starts with a functional diagram, including the length of each temperature stage, and position of desired components in each chain. Furthermore, realistic values for those functional requirements that are relevant for the application.

Figure 7.2 shows an example implementation of how a control highway module with 256 signal channels may look like. The room temperature node on the left has 256 coaxial connections in this implementation. A few bundles of flex lines will transport all those signals down to the milli-Kelvin stage at the very right of the figure. These flex lines are thermalised at each cooling plate within the cryostat to sink the heat. Passive components like attenuators and filters are integrated within the flex lines at dedicated temperature levels, and are hardly visible. A flange just below the room temperature node enables a vacuum feed-through for all flexible striplines entering the cryostat.

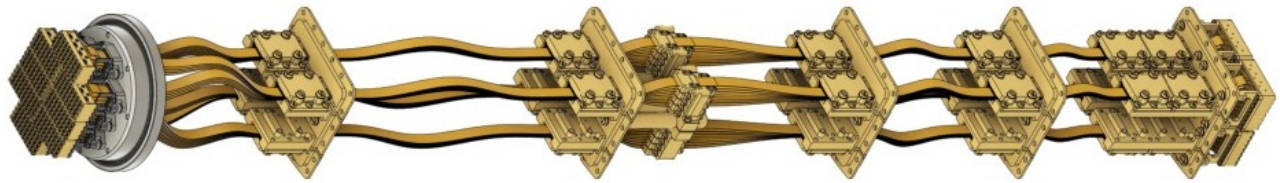

Figure 7.2. An example implementation of a control highway module with 256 channels

Figure 7.3 is a photograph of an example room temperature node with 64 coaxial connections. Due to its modularity, this number can be increased, for instance as shown in above figure 7.2.

Figure 7.3. An example room-temperature node with 64 coaxial connections

7.2 Functional Requirements of the control Highway

The transmission requirements on the control highway are to be defined in detail, and these requirements are highly dependent on the specific architecture and use case. The same applies to various interconnection and footprint requirements. But there are more issues of relevance that are to be specified, which may be less obvious. Their relevance is explained below.

7.2.1 Transmission requirements

Primary requirements on the transmission may include:

Masks for pass-band frequencies. These are design values (target) as well as masks for upper and lower limits of the transmission in the desired pass-band of interest when the chain is terminated by a specified impedance. This could be offered for the full chain, as well for each stage and/or segment/component.

Masks for out-of-band frequencies These are masks for upper limits on low-pass filtering for out-of-band frequencies. These masks may be specified up to one or two decades above the highest pass-band frequencies, to reduce out-of-band noise.

An effective reducing of out-of-band noise up to even infra-red frequencies might be of importance for qubits that are sensitive for it. For instance, if pulses are to be modulated on an 8 GHz carrier, these masks might be specified up to 100GHz or beyond. It is not obvious to achieve that with a low-pass microwave filter. This is because of the distributed nature of filters with transmission lines that have the fundamental limitation of passing signals above the break frequency. For instance, a 7th-order stripline filter up to 10 GHz, can have an out-of-band passband between 30 and 60 GHz.

Therefore the use of a microwave filter with an additional “IR-filter” may be essential to be compliant with out-of-band specifications. Such IR-filters are usually based on transmission lines with very lossy dielectric materials (like metal-powder filters), which can offer an extra slope (expressed in dB/GHz) up to 100GHz or more.

DC/low-frequency characteristics. These are requirements for DC-currents and low-frequency characteristics, for instance to separate bias currents from signals. When applicable, the maximum DC currents that may flow should be specified.

This requirement may become extra relevant when the current has to flow through a super conducting section in the chain. Such section will loose its super conductivity when that DC current a critical current.

Another reason to specify a maximum current is dissipation in lossy elements (cabling, attenuators, etc). Dissipation may heat-up the channel and exceeds the cooling capacity of the cryostat.

Step and/or impulse response These are requirements on step and/or impulse response of the full chain, when the chain is terminated by a realistic impedance. This could involve rise-time, overshoot, and ringing. Note that when the line is terminated with an impedance of a quantum device that is quite different from 50 ohm, it may not be useful to specify response under 50 ohm conditions.

7.2.2 Reflection requirements

Reflections, due to mismatched elements, can cause rippling in transfer functions and distortion of pulses at the end of the chain. But this does not mean that the existence of reflections will always harm. Reflections against filters will always occur for stop-band frequencies, and the same applies for mismatched components. However, reflected pulses will quickly fade-away in lossy lines and will be fully absorbed in matched attenuators.

These lossy conditions are quite common in cryogenic chains. Therefore reflection requirements for *individual* components in the chain should not be over-specified if the impact of these reflections are hardly visible in the transmission properties of the *full*-chain.

What really matters is specifying limits to rippling in the transfer function and deformation of pulse and step responses of the full chain under realistic termination conditions. What occurs internally is hardly of any relevance.

7.2.3 Crosstalk requirements

Crosstalk between channels will cause that a pulse intended for one qubit, will also appear (in weakened form) at other qubits. One should distinct between two different measures of crosstalk: NEXT and EL-FEXT.

- NEXT, or Near End Crosstalk, between two channels is measured by injecting a signal into one side of a channel and observing the crosstalk at the same side in another channel. NEXT will then be the ratio between crosstalk and injected signal levels.
- EL-FEXT, or Equal-Level Far End Crosstalk, is measured by injecting a signal into one side of a channel and observing at the other side what signal level will arrive and what crosstalk will arrive in another channel. EL-FEXT will then be the ratio between arriving crosstalk and arriving signal.

EL-FEXT to the cold side of I/O channels is of primary relevance, due to the typical signal levels that occur in these I/O channels. If a pulse is injected to control a particular qubit, then the EL-FEXT may disturb the quantum state of other qubits. It also may result in a false response signal from qubits in general.

NEXT at the warm side of upstream channels is also if primary relevance for the same reason. The amplified response of a qubit in an upstream channel may still be very weak and can easily be disturbed by pulses injected in downstream channels.

Again, one should not over specify crosstalk requirements. Crosstalk can also occur in the transmission lines within the quantum device. These are often shaped as a microstrip structure, and crosstalk between microstrips is usually much higher than crosstalk between stripline or coaxial structures. Crosstalk between channels in a quantum device put an upper limit to what crosstalk limits are reasonable within a control highway as a whole.

7.2.4 Heat flow requirements.

A cryogenic fridge cools the setup in multiples stages, with temperatures from T1, T2, T3, and so on, down to the lowest temperature; usually down to the milli-Kelvin range. The control highway has to bridge a temperature drop of about 300K, and these channels will leak heat from room temperature into the fridge down to the quantum device. This will challenge the cooling mechanism of the fridge, and

may prevent desired temperatures at the quantum device. Most of the heat flows through the metallic parts of the cabling, mainly through the shielding of coaxial cabling or ground planes of stripline cabling.

To minimise that heat flow, the cabling should have low thermal conductance, and should be thermalised at each temperature stage. Most of the heat flow through the cabling will then flow via the thermal anchors into the cooling mechanism. The residual heat flow to a next stage in the fridge will then be minimised.

A superconducting transmission line at one of the bottom sections may be used to reduce the heat flow even further. Superconductors tend to combine low thermal conductance with high electrical conductance, which is the opposite behaviour of metals.

Due to the large number of channels, this thermal leakage cannot be ignored and puts limits on the lowest temperatures that can be achieved since the cooling capacity of the fridge is limited. This puts a maximum on the number of channels.

This explains the need of various thermal requirements on the control highway as a whole.

Another source of heat is dissipation of signals in attenuators. They will heat up, and if all attenuation is concentrated at the lowest temperature stage, it may challenge the cooling capacity of the fridge as well as increasing the thermal noise generated within these attenuators. A good solution for restricting the hot spot temperature at the resistors of the attenuator is by mounting these attenuators close to a thermal anchor. An even better way of cooling is to integrate the attenuator in a flat transmission line (like strip lines) and to drain the heat away by mounting directly between the plates of a thermal anchor.

Therefore heat flow requirements could involve:

- Maximum passive heat flow through an I/O channel.
- Requirements on superconducting sections, for reducing the heat flow.
- Maximum signal dissipation in each stage (attenuators), at given signal power, to prevent that the resulting active heat flow overloads the cooling capacity per stage.
- Transversal thermal conductivity of an I/O channel near thermalisation clamps that are installed for draining away unwanted heat flow into the cooling system.
- Transversal thermal conductivity of attenuators to minimise raise of hot-spot temperatures. This increase of hot-spot temperature in the resistors of the attenuator will increase the thermal noise generated by these resistors.

7.2.5 Noise requirements

Each I/O channel suffers from adding extra noise to the signal. Even passive lines generate thermal noise, because the introduce loss.

Without any loss in a channel, and (hypothetical) noise-free control electronics, this noise level would be at least the thermal noise of a 50 ohm resistor at room temperature. Therefore, attenuators are to be placed at different temperature stages, to achieve noise temperatures that are only slightly above the temperature of each stage. Attenuation values between 40 to 80 dB distributed over the chain are not uncommon.

The lowest achievable noise level (in absence of any signal) occurs when all attenuation is concentrated in the stage with the lowest temperature.

Under operational conditions, however, signals will be dissipated in the attenuators, which is the reason why attenuation has to be distributed. This can be explained as follows.

At first, the dissipation of signal in the attenuator results in more heat power that should flow away via the cooling mechanism of the fridge. Since this cooling capacity is limited, with the lowest capacity at the coolest stages, this dissipation can easily overload the cooling. This is one reason why attenuation has to be distributed, a measure that also increases the noise at the end of the chain.

Secondly, the dissipation of signal power in the attenuator will increase the hot-spot temperature of the internal resistors. That temperature will raise above the outside temperature of the attenuator, which is usually thermalised at the stage temperature. This raise increases the thermal noise as well, which will be most pronounced by the last attenuator at the lowest temperature. Preventing all dissipation at a single spot by proper distribution of attenuation will reduce this noise. So even with infinite cooling capacity, attenuation has to be distributed because of noise.

The increase of hot-spot temperature can be reduced by effective hot-spot cooling. It requires attenuators with high thermal conductance between internal hot spots and external thermalisation points. Unfortunately, the thermal conductance of many materials is low at cryogenic temperatures, which challenges effective hot-spot cooling.

This may illustrate that effective hot-spot cooling and distribution of attenuation is essential to minimise the noise at the end of the I/O chain. The optimum distribution is use-case dependent, such as available cooling capacity of the fridge and used signal powers.

Therefore noise requirements on the control highway as a whole could involve:

- Requirements on maximum thermal noise temperatures at the end of downstream I/O channels, under passive conditions (in absence of any signal).
- Requirements on hot-spot cooling and distribution of attenuators to restrict the raise of noise temperatures.
- Requirements on noise generated within cryogenic amplifiers.

When specifying noise requirements, one should also account for the noise floor of the control electronics.

7.2.6 Vacuum requirements

A vacuum is needed as heat insulation to reach the low temperatures for cryogenic quantum devices. Once a vacuum pump has achieved the desired vacuum level, leakage from outside will gradually raise this level. It may be obvious that this puts strong vacuum requirements on the feed-throughs between outside and inside the fridge.

In addition, materials inside vacuum, and cavities within constructions, may suffer from out-gassing. This will gradually fill the vacuum with unwanted particles. And even when this out-gassing stops after a while, it may occur again after reopening the fridge when materials and cavities act like a sponge.

Outgassing is also strongly temperature dependent. At low temperatures almost all outgassing is stopped since most materials will freeze at cryogenic temperatures. This may suggest that outgassing is mainly a room-temperature issue.

However the main problem with leaks and outgassing is that the gases may condense and freeze at the colder parts of a cryostat which dissipates energy and uses part of the available cooling power.

Therefore vacuum requirements on the control highway as a whole could involve:

- Leakage requirements on the vacuum feed-through.
- Out-gassing requirements of the used materials and constructions.

7.2.7 Interconnection requirements

The more qubits that are to be controlled the more challenging it will become to connect them all to room-temperature electronics. Therefore interconnection requirements on the control highway could involve

- Interconnection between I/O chains and quantum devices. This may be performed by specifying preferred connectors or by specifying geometries to make a more permanent interconnection between cabling and these devices. This becomes even more important when connection are to be made directly to the quantum chips.
- Interconnection between I/O chains and control electronics. This may be performed by specifying preferred (bus) connectors.
- Means for organising a massive number of wiring between fridge and control electronics. It could be by specifying lengths of cabling outside the fridge or preferred intermediate (bus) connectors at some patch panel outside the fridge.

7.2.8 Footprint requirements

Different cryostat implementations are commercially available, and each of them may have different dimensions. Vacuum feed-throughs should fit on holes in the fridge, space near these holes outside a fridge is limited which may trouble access to connectors, cabling modules should fit inside these fridges, and the number of modules that can be mounted in a fridge is limited.

Therefore footprint requirements on the control highway as a whole could involve:

- Mechanical/dimensional requirements on vacuum feed-throughs.
- Mechanical/dimensional requirements on thermalisation clamps around cabling.
- Mechanical/dimensional requirements on holes in the plates on each stage.
- Ways to organize thousands of channels for controlling > 1000 qubits in a single fridge.

7.2.9 Vibration requirements

A cryostat may use a powerful pump which may induce mechanical vibrations in the setup as a whole. This energy might disturb the control of qubits.

One possibility is that the wiring itself is sensitive for these vibrations and may convert this mechanical energy into (weak) signals inside the cabling. Another possibility is that these vibration changes the

transmission properties and modulate the signals flowing into the channels. Bending a cable may make it a bit longer and will then increase the delay through that channel.

Therefore vibration requirements could involve:

- limits to induced signals as can be observed at the end of the channels
- limits to induced variations of transmission properties

7.2.10 Shielding and magnetic requirements

These requirements could involve:

- Non-magnetic requirements of dedicated connectors and other devices.
- Shielding around (groups) of I/O channels and components.
- Residual magnetic fields allowed in shielded environment.
- Maximum external magnetic fields to avoid saturation of shields.

8 Quantum Devices Layer

The quantum devices in the hardware layer are modules with qubits that are operating at cryogenic temperatures and may be implemented as chips and connected by PCBs (Printed Circuit Boards). Different solutions have been developed over time on how to implement the qubits in these devices, including transmons, flux qubits, semiconductor spin qubits topological qubits and NV centers in diamonds. Functional descriptions and functional requirements of these devices are described below.

8.1 Functional Description

The following members have been identified within this architecture family:

- Transmons;
- Semiconductor spin qubits;
- Flux qubits;
- CAT qubits
- Topological qubits;
- Artificial atoms in solids.
- Molecular spins

8.1.1 Functional description of Transmons

A transmon is one implementation of the unit of quantum information. A number of transmons on a chip forms a superconducting quantum processor unit (QPU), which is typically controlled using microwave and DC signals. The microwave signals are used to initialize and manipulate the state of the qubit, to perform quantum operations and readout. The DC signals are typically used to change the qubit's working frequency.

A single transmon qubit is a non-linear LC resonator with a narrow frequency in the bandwidth between 4 and 8 GHz. It has several input and output lines:

- A direct drive line that is used for injecting pulses modulated on a microwave carrier. These pulses are used to perform single qubit operations, such as initialize and manipulate the state of the qubit. The frequency of the pulse is at the same frequency of the corresponding qubit, so within the 4-8 GHz bandwidth, while their phase, amplitude and duration govern the amount of X and Y rotation that the qubit will acquire in the Bloch sphere. Typical values for the amplitude and duration of direct drive signals are -110 dBm (at PCB interface) and <30 ns, respectively.
- A flux line that is used to tune the individual qubit frequency to its desired value via a current. This DC current can be static (qubit at fixed frequency) or can be varied over time to move a qubit in or out of resonance with respect to other qubits, for example during 2 qubit gate operations. Moreover, a low frequency modulation can be over imposed to the DC current, to allow for a more accurate control of the trajectory of the qubit in its frequency and phase. The DC component of the flux signals have a maximum of around 2 mA, while the low frequency modulation might have a bandwidth up to 800 MHz and an amplitude of -110 dBm.
- A readout resonator (typically 6-7 GHz) is used to detect the state of the qubit. A readout tone at the frequency of the readout resonator is injected into resonator for a time between 300 ns up to a micro second depending on the desired read out fidelity. The signal reflected from the resonator has a shift in frequency that depends on the state of the qubit.

A QPU, which consists of an array of transmon qubits may have additional input and output lines such as:

- An (optional) control line for tunable coupling, in QPUs where the qubit-qubit interaction is mediated by a controllable element. Similarly to the flux bias lines for qubits, the DC component of the tunable coupler signals have a maximum of around 2 mA, while the low frequency modulation might have a bandwidth up to 800 MHz and an amplitude of -110 dBm.
- The read-out signals at the feedline output are to be amplified first with dedicated ultra-low noise amplifiers operating at cryogenic temperatures. Commonly used solutions are Traveling Wave Power Amplifiers (TWPA). Typical TWPA-pump signals are narrowband tones with a power of around -40 dBm at around 8 GHz.

8.1.2 Functional description of Spin Qubits

A spin qubit defines an implementation of a unit of quantum information. A multitude of connected spin qubits on a chip define a quantum processing unit (QPU). A spin qubit based QPU is typically controlled by a combination of dc, baseband, and microwave signals. The dc signals are used to bias the QPU to the right operating regime. The baseband signals can be used to perform quantum operations, as well as perform qubit initialisation and readout. The microwave signals can be used to perform quantum operations, as well as reflectometry for qubit readout.

Different implementations of spin qubits in quantum dots exists, but in general the quantum information is encoded in one or more electron/hole spins confined in one or more semiconductor quantum dots. The simplest example is a single electron/hole in a quantum dot, where spin-up and spin-down states span the qubit Bloch sphere.

Typically, the spin-qubit is operated in a cryostat within the bore of a cryogenic magnet, the latter to provide a static magnetic field. This can either be a single-axis or multi-axis (vector) magnet.

A single spin qubit is typically controlled by electric signals send to the different input terminals of the qubit. Effective control signals can also consist of components on several input lines, for example to offset crosstalk within the QPU. Typical control input lines of the QPU are defined by electrostatic gates which define a very-high impedance load.

Typically control signals control different physical parameters of the spin-qubit system:

- Charge occupation within the quantum dot system. By shifting the potential levels within the quantum dot system, carriers can be moved between quantum dots.
- Coupling between the different quantum dots. By shifting the potential level of the barriers between different quantum dots in the quantum dot system, the interaction between different quantum dots can be controlled.

From a signal perspective, the control signal inputs can typically be categorised as follows:

- DC signals required to bias the spin-qubit into the correct working regime. These can be applied both to the charge occupation and coupling control parameters. Typical amplitudes of these signals are -2<Vdc<2 V.
- Baseband signals (100 Hz < f < 1 MHz), used to switch between different operation regimes, such as initialisation, manipulation, and readout of the qubit. Typically, these signals consist out of extended plateaus where the same voltage is held, connected with ramped transitions. These can be applied both to the charge occupation and coupling control parameters. Typical amplitudes of these signals are 100-500 mV at the QPU.

- Baseband signals (1 MHz < 1 GHz), used to perform qubit operations. Depending on the encoding and/or operation mode of the qubit, these signals are used to either perform single or multi-qubit gate operations.
    - For some qubit encodings where single qubit operations are performed using baseband signals, these typically consist of ramped rectangular pulses on the charge occupation parameters, with durations of 1-100 ns. The selection between an X or Z rotation is done in this case through a separate control parameter controlling the coupling. Changing the voltage applied to the coupling control line, will tilt the axis around which the qubit will rotate during the baseband pulse, thus switching between e.g. X or Z rotations.
    - Two-qubit interactions can also be performed by applying a ramped rectangular pulse with typical durations of 1-100 ns to the correct control parameter.
    - In addition, these pulses can be shaped differently (e.g. Gaussian pulse) for higher performing qubit operations.
    - Typical signal amplitude might be 10-100 mV at the QPU.
- Microwave signals (100 MHz < f < 20 GHz), used to perform qubit operations. Depending on the encoding and/or operation mode of the qubit, these signals are used to either perform single or multi-qubit gate operations.
    - To perform resonant single qubit operations, these signals consist of short bursts of microwave signals, with a frequency matching the qubit frequency. The duration of the burst controls the angle of rotation, while the relative phase between different bursts controls the rotation axis. As an example, phase shifting a microwave burst by pi/2 with respect to a previous burst will produce a Y rotation, as compared to an X rotation for a burst with phase equal to 0.
    - In addition, these pulses can be shaped differently (e.g. Gaussian pulse) for higher performing qubit operations.
    - Typical amplitude of these signals might be -30 dBm at the QPU.
- Readout signals (100 MHz < 8 GHz), used to perform qubit readout through a resonant readout resonator. Depending on the implementation of the readout, such resonator can be connected to an electrostatic gate or ohmic contact. A readout tone is injected into the resonator for a time of 1-10 microseconds. The reflected signal will have a shift in frequency or amplitude, depending on the exact implementation of readout.
    - Typically, the feed signal for the readout resonator is injected through a directional coupler. The readout signals from the resonator output are amplified with a chain of dedicated low-noise amplifiers, both at cryogenic and room temperature.

Control signals in different frequency domains can sometimes be combined into a single input line, making use of bias-tees/diplexer circuits at different cryogenic stages. It is important to ensure the time-integral of the baseband signals equals zeros in this case, to prevent charging and drift of the bias tee circuitry.

8.1.3 Functional description of Molecular spin systems

Molecular spin systems and in particular molecular nanomagnets (MNMs) [3] have not yet reached the degree of development of the technologies described above but offer peculiar features and competitive advantages that make them a promising platform for general purpose quantum computing. Indeed, several MNMs are multi-level quantum systems, thus potentially offering many well characterized and coherent degrees of freedom that could be exploited for quantum information processing (QIP) [4] at

cryogenic temperatures. Another crucial difference from established technologies is the possibility of engineering the energy spectrum and eigenvectors of MNMs at the synthetic level.

This molecular design can provide many low-energy levels potentially protected from decoherence, which could be exploited to develop algorithms that go beyond the binary logic, where qubits are replaced by d>2 quantum systems called qudits. The latter approach could significantly simplify algorithms (such as quantum simulation) and thus lead to important advantages in the current NISQ era. Along this line, the first proof-of-concept quantum simulator was recently implemented on a molecular spin qudit manipulated by radiofrequency pulses [5]. Moreover, d>2 quantum systems can host error-protected logical units, where Quantum Error Correction (QEC) is embedded within single objects. Compared to multi-qubit encodings, the possibility of building a quantum processor with individual elementary physical units, the molecules, each encoding an error resilient logical qubit represents a potential important advantage. Indeed, it considerably simplifies the practical implementation of QEC, by eliminating nonlocal operations, and reduces the number of resources needed to carry out any computation.

Recently, the encoding of an error-correctable logical qubit in a nuclear spin qudit was demonstrated experimentally [6]. The description of these architectures and associated low-level layers is to be developed in future.

8.2 Functional Requirements

8.2.1 Functional requirements of Transmons

The operation of superconducting qubits can easily be deteriorated by external disturbances from outside the device. Examples are disturbing quantities like temperature, static magnetic fields, EM-fields, vibrations, infrared stray photons, background radiation and cosmic activity. But there may be more issues of relevance that are to be specified, which may be less obvious. Their relevance is explained below.

8.2.1.1 Thermal requirements

Superconducting qubits typically operate at very low temperatures, typically a few milliKelvin, which is close to absolute zero. This is necessary to reduce thermal noise and decoherence in the qubits, which can cause errors in quantum computations.

The low temperatures are achieved using a cryogenic cooling system, which typically consists of a dilution refrigerator with a pulse-tube cryocooler. The cooling system cools the qubits and their local control interconnections to very low temperatures (15 - 50mK), while also providing some level of electromagnetic and mechanical isolation.

Overall, the low temperature needed to operate superconducting qubits and the proper thermalisation of all the components are critical factors that must be carefully managed in order to achieve reliable and accurate quantum computations.

8.2.1.2 Static Magnetic field requirements

Stray magnetic fields can also be a significant source of noise and can cause decoherence in the QPU. Therefore, it is important to minimise the levels of magnetic interference in the cryostat.

One approach to reducing magnetic interference is to use shielding materials to block external magnetic fields from entering the cryostat. This can be achieved by using materials such as mu-metal or conductive foils to create a preferential magnetic path around the experimental volume. Additionally, careful placement of the cryostat and the orientation of the sample can help to minimise the amount of magnetic interference that arrives at the QPU.

Another approach is to use superconducting magnetic shields, based on the Meissner effect, that expels stray magnetic fields from the experimental volume below its superconducting transition temperature.

In general, the choice of non-magnetic materials in proximity of the QPU is to be avoided. Static magnetic fields can be compensated per qubit (by flux bias compensation) but is not a scalable solution and requires fully superconducting wiring to the QPU to avoid local dissipation.

8.2.1.3 EM field requirements

Electromagnetic interference can also be a significant source of noise and can cause decoherence in the QPU. Therefore, it is important to minimise the levels of electromagnetic interference (EMI) in the cryostat.

One approach to reducing EMI is to use shielding materials to block external electromagnetic fields from entering the cryostat. This can be achieved by using materials such as mu-metal or conductive foils to create a Faraday cage around the cryostat. Additionally, careful placement of the cryostat and other equipment can help to minimise the amount of EMI that enters the cryostat.

Another approach is to use filters and other signal conditioning techniques to reduce the impact of EMI on the QPU. This can include using low-pass filters to remove high-frequency noise, or using notch filters to remove specific frequencies that are known to cause interference.

To ensure that the levels of EMI are within the required limits, it is important to measure and monitor the EMI levels in the cryostat using electromagnetic field sensors. This can help to identify any sources of EMI and enable corrective measures to be taken.

8.2.1.4 Vibration requirements

Cryostat vibration can be a significant source of noise and can cause decoherence in the QPU. Therefore, it is important to minimise the vibration levels in the cryostat.

One approach to reducing vibration is to use a mechanical support system that isolates the QPU from external vibrations. This can be achieved by using a combination of passive and active vibration isolation techniques. Passive isolation techniques include using rubber pads or springs to decouple the QPU from the cryostat, while active isolation techniques use sensors and actuators to cancel out vibrations in real-time.

Another approach is to use a cryocooler that operates at a higher frequency than the natural frequency of the cryostat, which can help to reduce vibration levels. Additionally, careful placement of the cryocooler and other equipment in the cryostat can help to minimise vibration.

Vibrations in the cables have been shown to induce triboelectric noise in qubits. To ensure that the vibration levels are within the required limits, it is important to measure and monitor the vibration levels in the cryostat using accelerometers or other sensors. This can help to identify any sources of vibration and enable corrective measures to be taken.

8.2.1.5 Infrared stray photon requirements

Stray photons in the infrared range can also be a significant source of noise and can cause decoherence in the qubits. Therefore, it is important to minimise the levels of infrared radiation that reach the qubits.

One approach to reducing infrared radiation is to use optical filters to block out infrared photons. This can be achieved by using filters that are designed to block out specific frequencies of infrared radiation. Additionally, careful placement of the qubits in light tight enclosure is necessary to minimise the amount of infrared radiation that reaches the qubits.

Another important consideration is that the waveguides used to interconnect the QPU typically also contain dielectrics that are transparent to IR radiation, therefore is essential to use IR filters or waveguides that are transparent to microwave frequencies but opaque to infrared radiation to create a shield around the qubits. This can be achieved by using dielectric materials such as copper or magnetic powders mixed in an epoxy matrix.

Overall, protecting the qubits from infrared stray photons is an important factor in achieving reliable and accurate quantum computations, and requires careful consideration of IR filtering and material selection.

8.2.1.6 Shielding requirements against background radiation and cosmic activity

Background radiation and cosmic activity can also be a significant source of noise and can cause decoherence in the qubits by the creation of phononic and quasiparticle excitations. Therefore, it is important to minimise the levels of background radiation and cosmic activity that reach the qubits.

One approach to reducing background radiation is to use shielding materials to block out radiation from the environment. This can be achieved by using materials such as lead or tungsten to create a radiation shield around the QPU to reduce the gamma ray component. With careful choice of radio pure materials in close proximity to the QPU, one can mitigate the amount of local radioactivity and the generation of induced secondary radioactivity.

Ultimately, to further reduce the impact of cosmic activity in the form of neutrons and muons, it is advisable to locate the cryostat in an underground facility under several tens of meters or more of ground.

8.2.2 Functional requirements of Spin Qubits

The operation of spin qubits can easily be deteriorated by external disturbances from inside or outside the device. Examples are disturbing quantities like temperature, magnetic fields, electric fields, vibrations. But there may be more issues of relevance that are to be specified, which may be less obvious. Their relevance is explained below.

8.2.2.1 Thermal requirements

Quantum dot spin qubits operate at cryogenic temperatures, typically between 20 and 1000 mK. This is necessary to reduce thermal broadening of the electron states to improving the performance of commonly used readout techniques. Furthermore, low operation temperatures can aid in minimizing charge noise originating from defects in the QPU material stack that can negatively impact qubit coherence. The low temperatures are achieved using a cryogenic cooling system, which typically consists of a dilution refrigerator with a pulse-tube cryocooler. The cooling system cools the qubits and

their local control interconnections to very low temperatures (phonon temperature of <50 mK), while also providing some level of electromagnetic and mechanical isolation. To also achieve equally low electron temperatures, proper thermalisation of all components and filtering of relevant lines should be carefully managed.

8.2.2.2 Magnetic field requirements

Magnetic field fluctuations and/or distortions can be a source or decoherence or operation errors with the QPU. It is therefore important to minimise the levels of magnetic interference in the cryostat. As an example, careful placement of the cryostat in relation to sources of magnetic noise, such as transformers or large-current carrying wires can help to minimise magnetic interference. To minimise the distortion of applied magnetic fields, care should be taken to avoid the use of magnetic materials in the vicinity of the QPU, avoiding magnetisation because of the applied magnetic fields.

8.2.2.3 Electric field requirements

Electric field fluctuations can be a source of decoherence or operation errors within the QPU. In particular, for spin-qubit implementations where a coupling between electric and magnetic fields is implemented for rapid qubit control, e.g. through spin-orbit interaction or the presence of micromagnetic structures. In such case, fluctuations of the applied control voltages can directly impact the qubit energy splitting and lead to decoherence. As a result, the qubit coherence can be impacted by noise with a wide range of frequencies, including slow (near dc) fluctuations. Therefore, it is important to minimise the levels of electric interference and noise on the control lines. One approach to reduce electric noise is by filtering and/or attenuating the control lines going into the cryostat. Exact filter bandwidths and attenuation values depend on the requirements of each specific line and should be tailored to the application. Another approach is to use control electronics that have a low output noise, in particular within the required output bandwidth, as this is particularly difficult to filter.

8.2.2.4 EM field requirements

Electromagnetic interference can also be a significant source of noise and can cause decoherence in the QPU. Therefore, it is important to minimise the levels of electromagnetic interference (EMI) in the cryostat. One approach to reducing EMI is to use shielding materials to block external electromagnetic fields from entering the cryostat. This can be achieved by using materials such as conductive foils to create a Faraday cage around the cryostat. Additionally, careful placement of the cryostat and other equipment can help to minimise the amount of EMI that enters the cryostat. Finally, when the bandwidth requirements of certain control lines permit, using twisted pair wiring can aid in reducing EMI pickup in the control highway.

8.2.2.5 Vibration requirements

Cryostat vibration can be a significant source of noise and can cause decoherence in the QPU. While the qubits themselves are not affected by the vibrations, noise can couple into the system through the generation of effective electric of magnetic fields. Vibrations in the cables have been shown to induce triboelectric noise. Vibrations of the qubit with respect to the cryogenic magnet can lead to effective magnetic noise. Therefore, it is important to minimise the vibration levels in the cryostat. One approach to reducing vibration is to use a mechanical support system that isolates the QPU from external vibrations. This can be achieved by using a combination of passive and active vibration isolation

techniques. Passive isolation techniques include using rubber pads or springs to decouple the QPU from the cryostat, while active isolation techniques use sensors and actuators to cancel out vibrations in real-time. Another approach is to use a cryocooler that operates at a higher frequency than the natural frequency of the cryostat, which can help to reduce vibration levels. Additionally, careful placement of the cryocooler and other equipment in the cryostat can help to minimise vibration.

8.2.2.6 Signal integrity requirements

Depending on the specific implementation, the accuracy of the quantum operations can be sensitive to the exact pulse shape arriving at the QPU. The usage of various components with different bandwidths and (effective) filter characteristics in the control highway will lead to a deformation (in the time domain) of the input signal at room temperature. One approach to reduce these effects is by measuring the deformation and predistortion of applied signals to correct for the imperfections in the control highway. Another approach is to design the control highway in such a way that within the relevant bandwidth, minimal signal distortion is present.

Secondly, cross talk between different control lines can lead to unwanted quantum operations and/or decoherence. One approach to overcome this is by minimising crosstalk between different control lines, through careful design and implementation of proper shielding/grounding of the control highway. Another approach to overcome this is by applying correctional operations to the QPU, to undo effects caused by the crosstalk of previous operations. To ensure that levels of crosstalk are within the required limits, it is important to measure and monitor crosstalk between various control lines. This can help to identify any sources of crosstalk and enable corrective measurements to be taken.

Bibliography

[1] Cross, Andrew, et al. “OpenPulse Grammar — OpenQASM Live Specification Documentation.” Openqasm.com, 2019, openqasm.com/language/openpulse.html. Accessed 6 Sept. 2024.

[2] “Pulser — Pulser 0.19.0 Documentation.” Readthedocs.io, Pulser Development Team, 2022, pulser.readthedocs.io/en/stable/. Accessed 6 Sept. 2024.

[3] A. Gaita-Ariño, F. Luis, S. Hill, E. Coronado, Molecular spins for quantum computation, Nature Chemistry 11, 301-309 (2019)

[4] A. Chiesa, P. Santini, E. Garlatti, F. Luis, S. Carretta, Molecular nanomagnets: a viable path toward quantum information processing?, Rep. Progr. Phys. 87 (3) (2024) 034501

[5] S. Chicco, G. Allodi, A. Chiesa, E. Garlatti, C. D. Buch, P. Santini, R. De Renzi, S. Piligkos, S. Carretta, Proof-of-Concept Quantum Simulator Based on Molecular Spin Qudits, Journal of the American Chemical Society 146, 1 (2023)

[6] S. Lim, M. V. Vaganov, J. Liu, A. Ardavan, Physical Review Letters 134, 070603, 2025