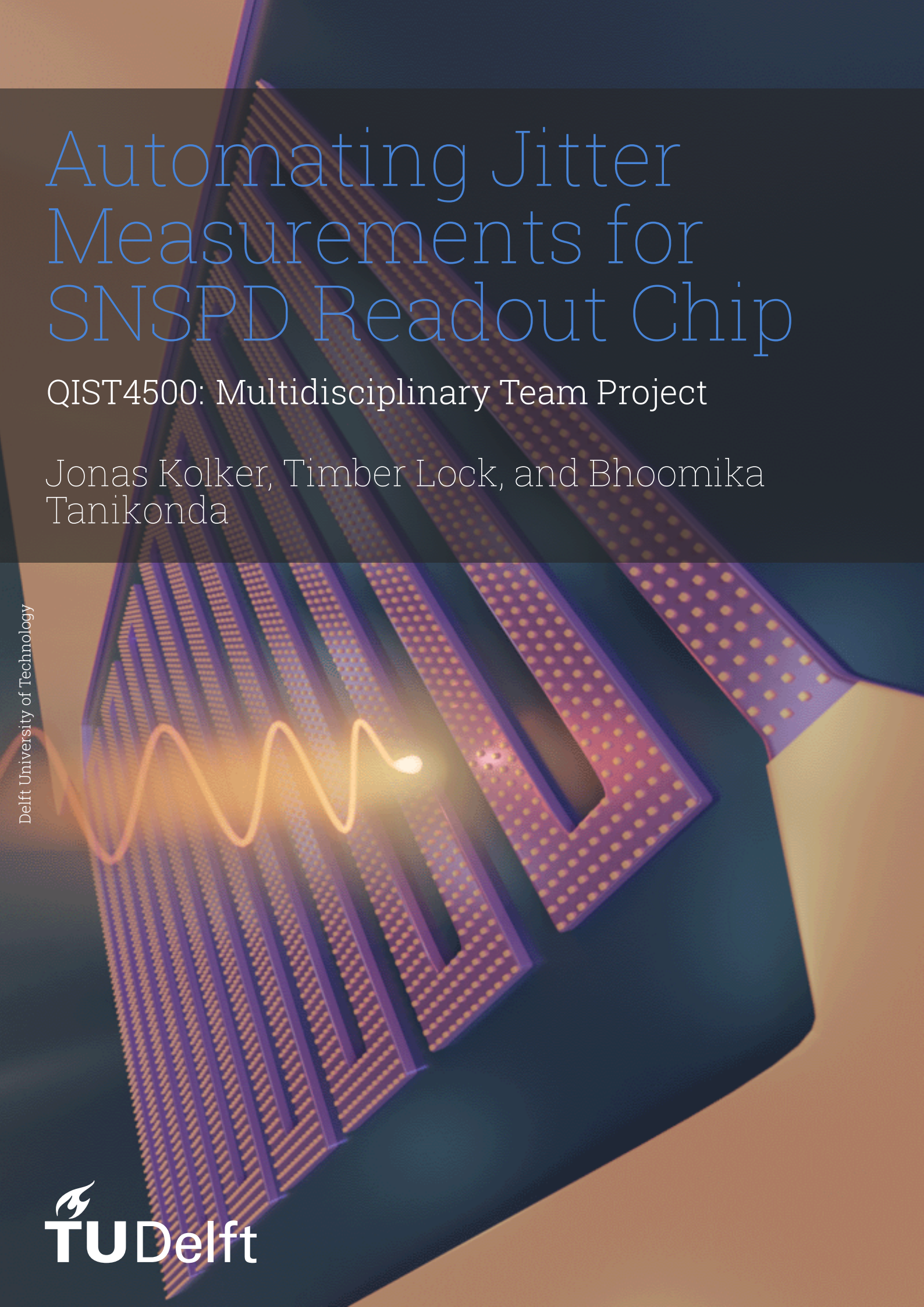
Automating Jitter Measurements for SNSPD Readout Chip

QIST4500: Multidisciplinary Team Project

Jonas Kolker, Timber Lock, and Bhoomika Tanikonda

Delft University of Technology



TU Delft

Contents

1. Problem Setting

Quantum computers (QCs) promise exponential speedups for specific tasks in cryptography, optimisation, and physical simulation. However, current QCs are not yet powerful enough for practical applications. One of the numerous roadblocks currently limiting performance is the difficulty in scaling. A robust QC needs an electronic interface to control and read out many quantum bits (qubits) simultaneously. Colour centres in diamonds, such as Nitrogen Vancancies (NVs), have emerged as a qubit platform with scaling potential thanks to their ability to interface with each other remotely through optical links. They also have higher operating temperatures (around 1 K) relative to other qubit architectures. An NV centre qubit state is read out through the emission of a single photon, necessitating an apparatus operable in cryogenic temperatures with high timing precision.

SNSPDs are one such detectors that can detect a single photon emitted from NV centre. SNSPDs offer excellent detection efficiency, sub 100 ps timing jitter, and low dark count rates, making them ideal for colour-centre qubit readout. They are superconducting nanowires which are biased just below the critical current. When the single photon falls on the SNSPD, it gets absorbed and creates a local resistive spot, which temporarily breaks the superconductivity. Hence, the current originally flowing through the superconductor is diverted to a branch of the readout circuit(load resistance) across which a voltage pulse is generated, which could be measured to detect the emission of the photon. The SNSPD then resets to the superconducting state after a recovery period which depends on the kinetic impedance of the nanowire and the load resistance across it. Depending on this load resistance, the SNSPD can recover faster if the load resistance is low or gets latched to being resistive if the load resistance is high, so here in this project, they have decided to use an active quenching circuit to prevent latching while using the high load impedance to boost the signal amplitude of readout voltage pulses.

Doing this adds a whole lot of circuitry as you can see, in Figure 1, and each of the bias currents, amplifier settings, load resistor, ctive quenching switch, timing delays etc needs to be set using digital switches(for example see Figure 2) which are programmable through a PC which sets the values digitally, send this information to the arduino which send these binary on or off of each switch to the chip.

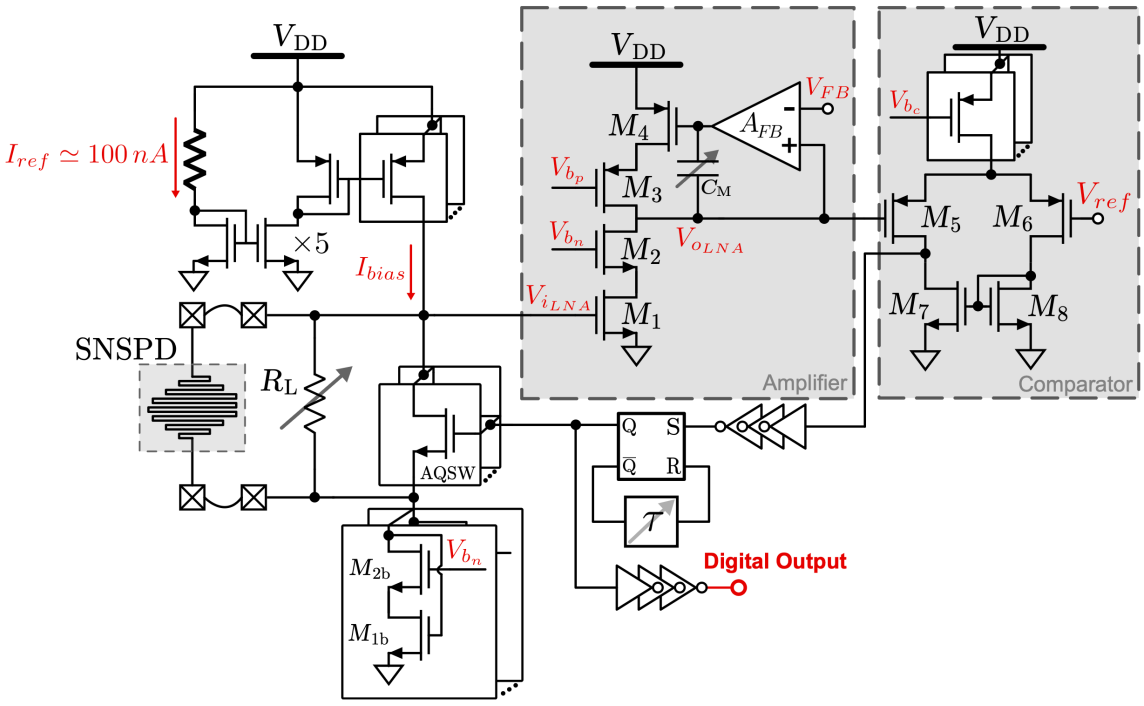


Figure 1: Schematic of the cryo-CMOS readout circuit. This is a figure from [1]

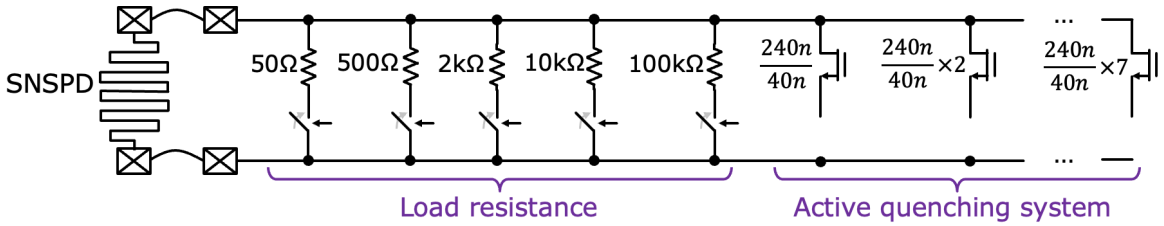


Figure 2: Schematic of load resistance R_L and AQSW of Figure 1

An essential figure of merit for SNSPD-based systems is timing jitter, which refers to the uncertainty in the detection time of a photon relative to a reference trigger(such as a laser pulse) and the detector's electrical output. Jitter arises from stochastic hotspot formation, readout noise, and electronic timing resolution. It is typically quantified by the Full Width at Half Maximum (FWHM) of the delay distribution measured over many detection events, but in this report when we say jitter we mean the standard deviation of delay (also referred to as timing offset) values. Accurately characterising jitter is essential because it directly determines timing precision, affects photon time filtering to suppress noise, enables higher count rates. In the context ofNV-centrer quantum computing, low jitter is crucial for achieving the tight timing correlations required for entanglement protocols and error correction. Moreover, jitter directly influences key system-level performance metrics such as fidelity, bit error rates, and latency.Optimising circuit parameters to minimise jitter is therefore a key step toward scalable, high-fidelity quantum readout.

As you have seen in Figure 1, there are many of these parameters, we need to sweep through each of these parameter set, so checking the jitter measured for each set of parameters is time consuming. In order to make this time efficient we need to automate sweeping through each of the chip configurations (through arduino) and then data acquisition through oscilloscope by interfacing them both(arduino and the oscilloscope) to a central PC. This eliminates changing the code manually for each sweep point of chip configuration, check the scope and repeat.

The SNSPD and readout chip are moved to a new cryostat (Quantum Design OptiCool) with a 7 T superconducting magnet. Additionally a newer iteration of chip is currently being designed, which will add the ability for a single chip to connect to multiple SNSPDs simultatiously. The jitter of this chip would also need to be characterized.

2. Problem Statement and Project Goals

2.1. Problem Statement

By the start of the project, the novel readout chip had been fabricated and its initial jitter performance benchmarked. This jitter was directly measured on an oscilloscope using its built-in edge delay features. A primary benefit of entirely on-scope measurements is speed. However, it also comes with downsides. All actual waveform data from which edge offsets are calculated is discarded by the scope. This means that there is no way to re-analyze old datasets.

An additional pitfall of non-automated measurements is the inability to efficiently sweep over different chip parameters. To iterate upon the current design and make improvements, understanding the impact of different parameter configurations on jitter is crucial. There are seven key adjustable chip parameters, each with a variable number of possible values. Obtaining jitter measurements while sweeping over all possible parameter values would be incredibly laborious by hand.

The problem statement is therefore defined as:

> ***Develop a robust and scalable testing framework that can adjust chip parameters to characterize detection jitter.***

2.2. Scope and Success Criteria

The primary scope of the project is centered around developing the automated testing framework. A key requirement emphasized by the supervisors was the need for signal waveform data to be saved alongside resultant jitter measurements. This would give researchers the freedom to return to old data and adjust their processing methods as needed.

Another direction confirmed with the supervisors to be within the project scope was designing a new printable circuit board (PCB) to support future designs. The current PCB is not suitable for chip iterations that connect with more than one SNSPD. While not directly part of experiment automation, the PCB redesign corresponds to the “robust and scalable” criteria from the problem statement.

Actually analyzing the jitter measurements was beyond the scope of the project. The goal was not to achieve a specific level of jitter reduction. The focus was solely on developing an improved measurement apparatus, *not* on drawing conclusive meaning from the results. The supervisors would handle that analysis after the system was delivered to them.

The success criteria defined at the outset were as follows:

1. Configure an experimental setup. This means gathering and properly connecting the various components required to carry out measurements.
2. On the PC, successfully read out response signals from the chip routed through an oscilloscope.
3. Establish control of the chip such that circuit parameters can be adjusted by the PC remotely.
4. Develop and implement a process to obtain jitter measurements for different chip settings.
5. Create a GUI that acts as a user-friendly frontend to the automation software.
6. Prepare for future chip implementations involving multiple SNSPDs by designing a new PCB.

These criteria were crafted roughly in the order they were intended to be accomplished. The first four points represented the main thrust of the project, while the latter two were considered secondary goals. The focus with respect to these points shifted slightly over the course of the project as various obstacles were encountered. The degree of achievement for each point is elaborated upon in the discussion section 8.

3. Experiment Layout and Design

Figure 3 shows the configuration targeted for the final experimental setup to characterize the jitter performance of the chip. The input to the system is a laser capable of simultaneously generating a trigger output when a pulse is sent. This trigger output serves as a reference signal to determine the jitter. The laser pulse itself interacts with the SNSPD, and this interaction is detected by the chip, which outputs a falling edge. Both the reference trigger signal from the laser and the output of the chip are sent to an oscilloscope. The oscilloscope then transmits its data to a PC, which analyzes the signals and calculates the jitter parameters.

To enable configuration of the chip and investigation of its influence on jitter, the chip is connected to an Arduino. The Arduino acts as a communication bridge between the PC and the chip, allowing the PC to set and modify chip parameters.

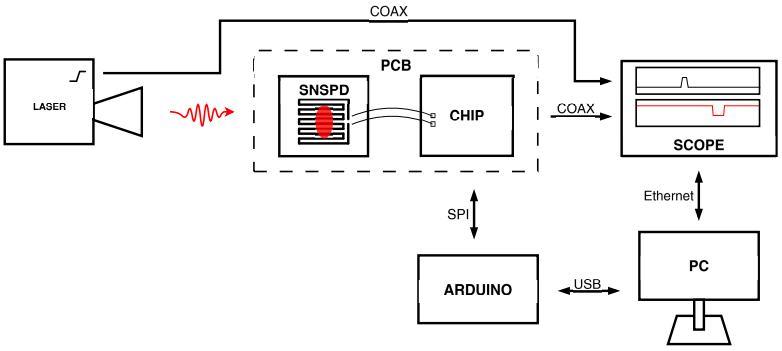


Figure 3: Ideal setup for measuring jitter.

However, for the purposes of this project, the laser and SNSPD are omitted from the setup. Because the SNSPD operates only within a cryostat, including it would significantly increase experimental complexity. Omitting these components allows development of a robust backend while maintaining the possibility of integrating the frontend later. Furthermore, the laser required for the complete setup is not yet available, as it is still awaiting delivery.

In the interim, the frontend (laser and SNSPD) is replaced by an arbitrary waveform generator (AWG). The AWG outputs a pulse that is routed both to the chip and directly to the oscilloscope as a reference signal. Figure 4 presents a schematic representation of this configuration.

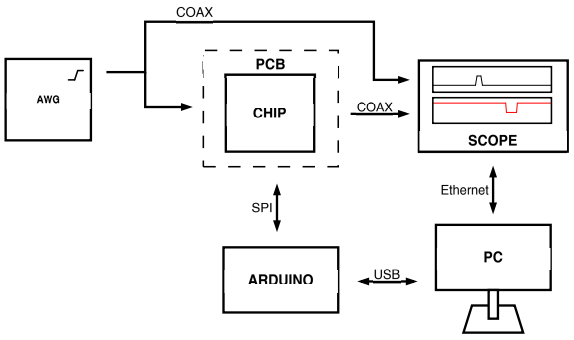


Figure 4: Non-ideal setup for measuring jitter.

Not all components of the system had to be developed from scratch. A PCB with the chip mounted and interfaced to the oscilloscope was already available, as well as the Arduino bridge. Although the existing PCB was suitable for use in the reduced setup, it would not suffice for the ideal configuration due to incompatibilities with the cryostat's physical dimensions and with newer chip generations currently in development. Therefore, it was decided to redesign this part of the system, ensuring that the final setup will be directly compatible with the ideal configuration.

4. Oscilloscope Control

The oscilloscope represents a crucial component of the testing framework. In order to resolve jitter on the order of less than 60 ps full width at half maximum (FWHM), the instrument must support a sufficiently high sampling rate [1]. The model used for this work was a Lecroy WavePro 760Zi-A oscilloscope (Figure 5), offering up to 40 GS/s sampling rate and full PC remote-control capabilities. During data acquisition, 500 samples were collected per 50 ns window, corresponding to a 10 GS/s sampling rate. An invaluable reference for this section was the MAUI oscilloscope user manual [2]. Full oscilloscope control code is provided in the Appendix.

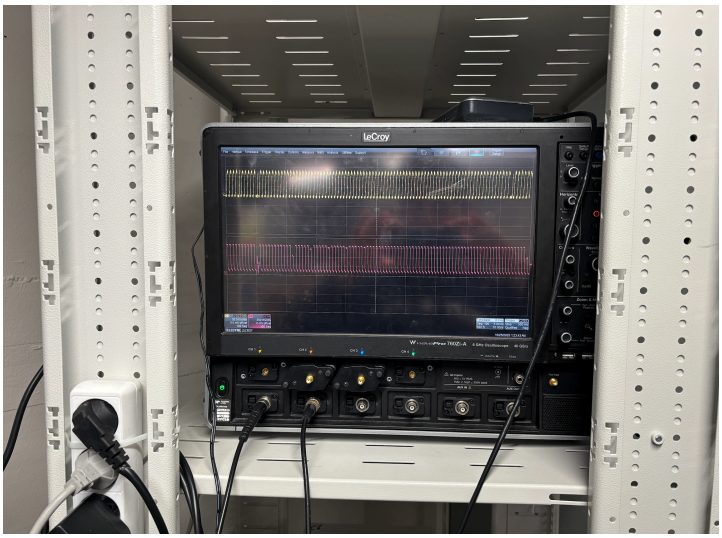


Figure 5: Lecroy WavePro 760Zi-A oscilloscope in sequence mode. Onscreen are $N = 100$ acquired waveforms. Channel 1 (top) is from the AWG, while Channel 2 (bottom) is from the chip.

4.1. Signal Data to be Acquired

Two signals are routed to the oscilloscope: the reference signal and the chip signal. The term reference refers to the source that triggers the chip's response. In a full experimental setup, this reference originates from the photodiode output of a pulsed laser directed at the SNSPD (Figure 3). To simplify testing and avoid cryogenic operation, the chip is evaluated at room temperature without the SNSPD attached. The SNSPD response is instead simulated using an arbitrary waveform generator (AWG).

For this purpose, the AWG generates 160 mV amplitude square pulses with a 105 ns high time and a 1.1 µs low time values of the same order as expected SNSPD responses. The rising edges of these pulses trigger the chip response, typically occurring around 30 ns later. The response signal from the chip appears as a falling edge with an amplitude of approximately 1.1 V. The low time of this signal is generally on the order of 10 ns, providing ample separation between successive AWG rising edges for the chip to respond and reset to its initial state.

4.2. Triggering an Acquisition

The oscilloscope continuously receives signals on both channels but records and displays data only upon being triggered. Correctly setting trigger parameters is essential to ensure that the acquired data are meaningful for jitter measurements. The control software configures the oscilloscope to trigger waveform acquisition only if a falling edge below 0.5 V occurs on Channel 2 (the chip signal) within 100 ns of a rising edge exceeding a 50 mV threshold on Channel 1 (the reference signal). This ensures that all recorded data correspond to legitimate reference events and their associated chip responses.

Two acquisition parameters play key roles: the acquisition duration and the number of samples collected within that window. A high sample count ensures precise timing resolution of edge crossings, while the acquisition window must be wide enough to capture both relevant edges. Typically, there is a delay of approximately 30 ns between the rising and falling edges on the two channels. After capturing the rising edge on Channel 1, roughly 28 ns pass before the falling edge on Channel 2 occurs. Data in

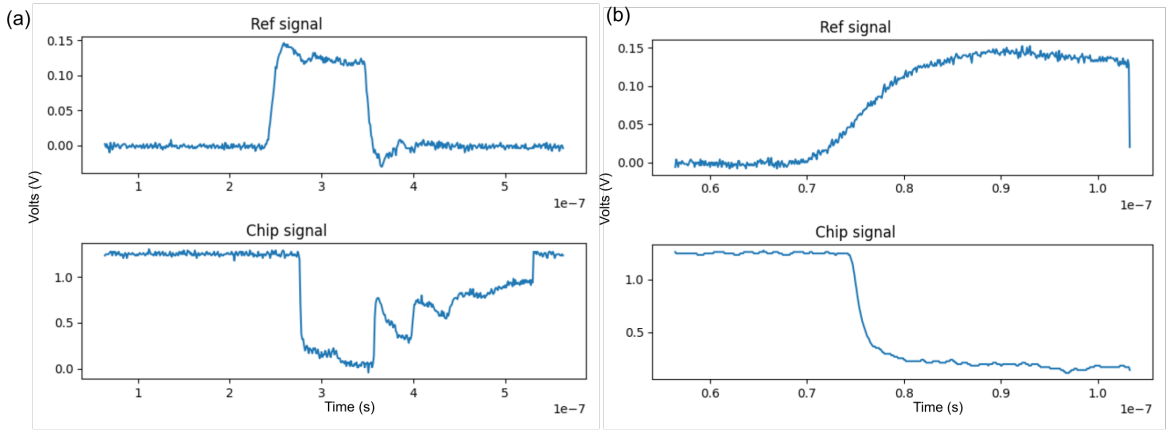


Figure 6: **(a)** A single 500-sample acquisition of the two channels without any deskew over a 500 ns window. **(b)** Another 500-sample acquisition, this time with a 30 ns deskew and a 50 ns acquisition window. The rising reference edge and falling chip edge are much more prominent.

Channel 1 during this interval are not relevant to jitter calculations.

The oscilloscope provides a deskew parameter to address this. Deskew introduces a timing offset between channels, effectively aligning them in time. With deskew set to 30 ns, the displayed delay between the rising edge in Channel 1 and the falling edge in Channel 2 is nearly eliminated. This artificial alignment does not affect jitter measurements, since only the standard deviation of delay values is significant. Consequently, the acquisition window can be shortened to focus on the regions of interest (see Figure 6).

4.3. Transmitting and Interpreting Data

Tens of thousands of waveforms are required for a single jitter measurement, as specified by the experimental supervisors. To maintain efficiency, the time required for data transfer between the oscilloscope and the control PC must be minimized. For this purpose, Sequence Mode is utilized.

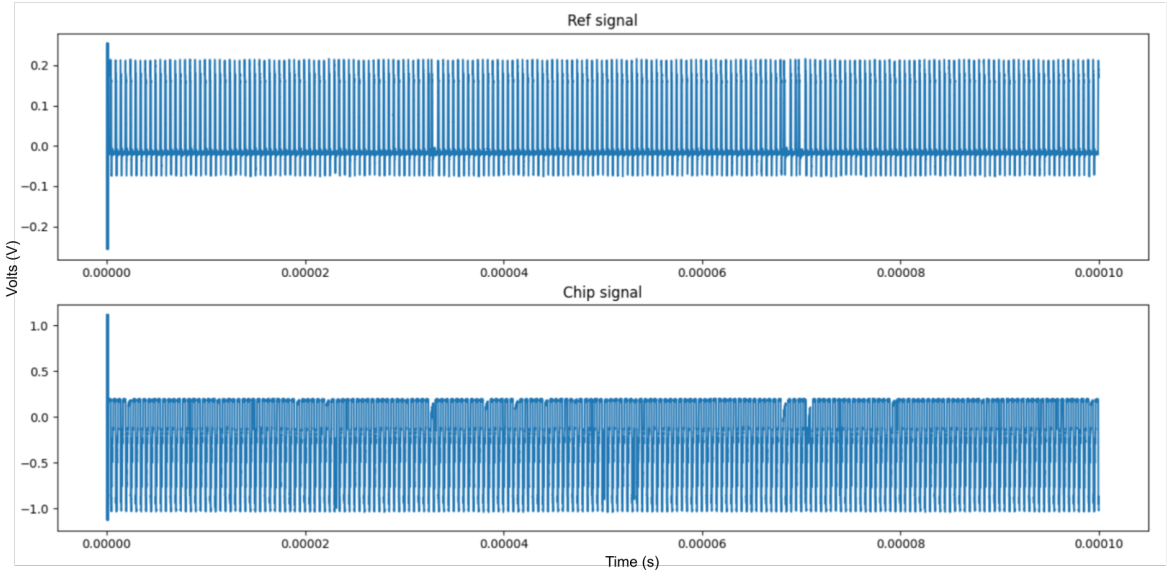


Figure 7: A sequence of 100 segments from the oscilloscope plotted without any clipping. The erratic initial behavior corresponds to bytes of metadata that were incorrectly interpreted as floats by `numpy`. In this figure, both channels were AC coupled, centering them at 0 V. Other signal plots in this report are from DC-coupled channels.

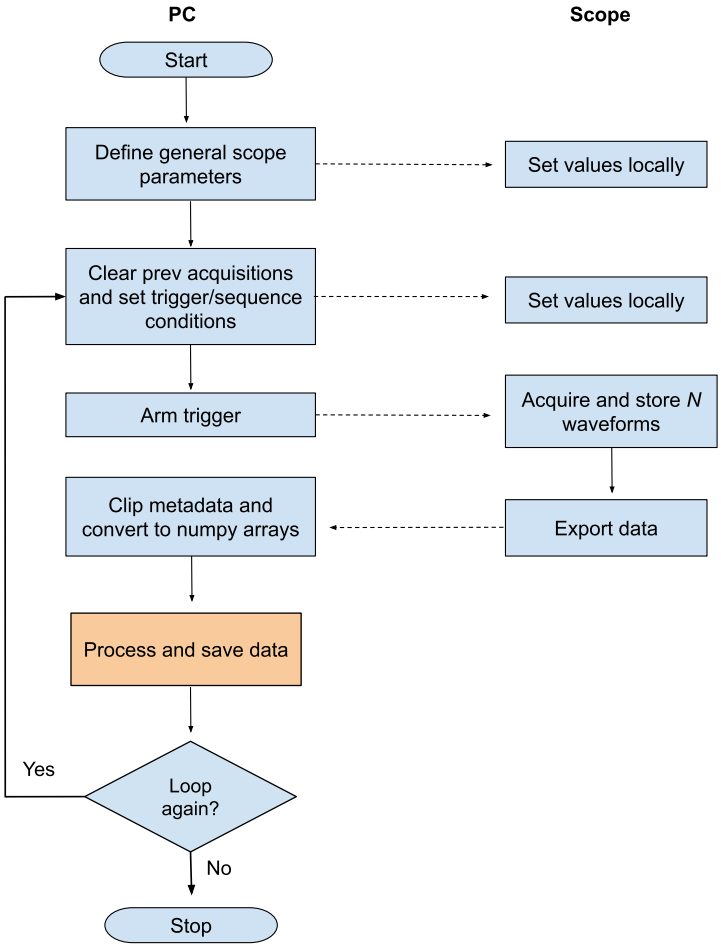


Figure 8: Overview of the waveform data transfer process between the oscilloscope and PC. Solid arrows indicate control flow, while dashed arrows represent data transmission. The process begins with the PC setting general display parameters (e.g., time and voltage divisions) on the oscilloscope. Triggering parameters are then configured, and the oscilloscope is armed. After all sequences are acquired, data are exported to the PC. Metadata are removed, and the processed data are used to extract offset values before being saved as binary files. The process is repeated according to user-specified parameters. Orange elements are described in detail in the **Processing Data** section.

In Sequence Mode, a user-defined number of samples N is stored locally on the oscilloscope. The hardware limit for N was found to be 5000. The oscilloscope then waits for N triggers, collecting the corresponding waveforms (as shown in Figure 5). After the sequence is complete, the PC requests the stored waveform data, which are transmitted as a byte stream. The oscilloscope is then ready to perform the next acquisition burst.

The transmitted data include a header containing metadata that describe acquisition parameters. In Sequence Mode, approximately $16N$ bytes are devoted to trigger-event information. These metadata bytes are removed during post-processing to prevent interference with numerical data interpretation.

For a typical jitter measurement, approximately 50,000 triggered acquisitions are required. This is achieved by repeating the process outlined in Figure 8 for ten iterations.

5. Chip Control

5.1. Arduino reads the data and echoes back

We were given a file which configures the chip parameters through a serial interface between the PC and an Arduino, which acts as the intermediary between the control software and the chip's SPI pins. To communicate configuration parameters, the PC transmits a fixed-length register frame over USB serial to the Arduino. The host side builds a 35-byte packet made of 1 preamble byte (0x00), 33 bytes of register data, and a single termination byte (0xff). Of these 33 register bytes, the first 18 correspond to actual chip control parameters, while the remaining 15 are left as vacant padding. Each of the 18 parameter fields is one byte wide and encodes a specific control setting for the chip, such as bias currents, amplifier settings, timing delays, and analog selection lines.

On send, the Arduino echoes back the trailing 34 bytes, ignoring the first byte, which is 0x00 (the 33 register bytes + the 0xff terminator). The PC then byte-for-byte compares what it sent with what it received; a full match would mean successful configuration of the chip, so then the host software we wrote(main.py) would go on for further processing; otherwise, it flags a mismatch.

Each parameter in the 18-byte register block has a defined range and bit width. For example, DC-compensate is a 3-bit value ranging from 0 to 7, DFBamp is 4 bits (0–15), and several others, such as DSNSPD, DAQSW, and Dbias_ampNMOS, are 7-bit values (0–127). Bias parameters like Dbias_NMOS and Dbias_PMOS are encoded as full 8-bit values, while control lines such as Analoga are encoded as bit fields corresponding to predefined analog sources. The given file also enforces these ranges before transmitting data, ensuring that only valid configurations reach the Arduino.

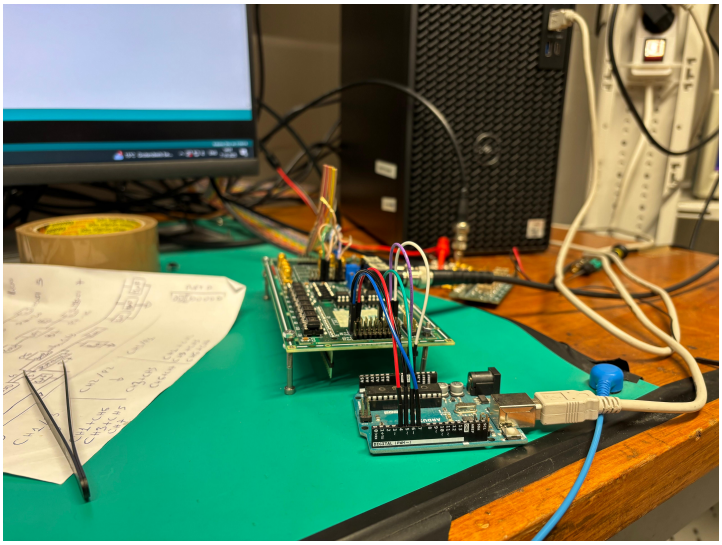


Figure 9: Setup showing PC connected to Arduino, which is connected to SPI pins, which transfers data onto the chip.

The communication port used by the system is stored in a simple text file, which allows the control scripts to automatically identify the correct Arduino COM port without manual intervention.

5.2. More on experimental run in main.py script

The main.py script constructs and manages the register frame, and sweeps through parameter values systematically. In a typical experimental run(main.py), all parameters are initialised to a default operating point, after which a single parameter is varied across its defined range while the others are held constant.

During each parameter sweep, main.py orchestrates both chip configuration and oscilloscope acquisition in a single loop. For a given parameter, the script first assembles the 35-byte configuration frame and transmits it over the Arduino's serial link. The Arduino then writes these values to the chip via SPI and echoes back the trailing 34 bytes for verification. Only when this echo check succeeds does the script arm the oscilloscope and initiate an acquisition burst in Sequence Mode, ensuring that ev-

ery dataset is paired with a confirmed register state (see the preceding Oscilloscope Control and Chip Control subsections).

The scope collects N triggered waveforms locally, then main.py retrieves the raw bytes, discards the initial metadata block, and forwards the two voltage traces per segment to the analysis routine that extracts edge times and computes delays/jitter (with the detailed post-processing described in the Processing Data section) and plots the jitter per parameter sweep for each of the parameters. Resulting parameter values, per-segment delays, and summary statistic (i.e, histograms) are written to disk alongside waveform data so that each acquisition can be traced back to its exact register frame and instrument settings. This is further explained in the Processing data section. The overall flow of that main.py follows is summarized schematically in Figure 10. The code files themselves can be found in the Appendix.

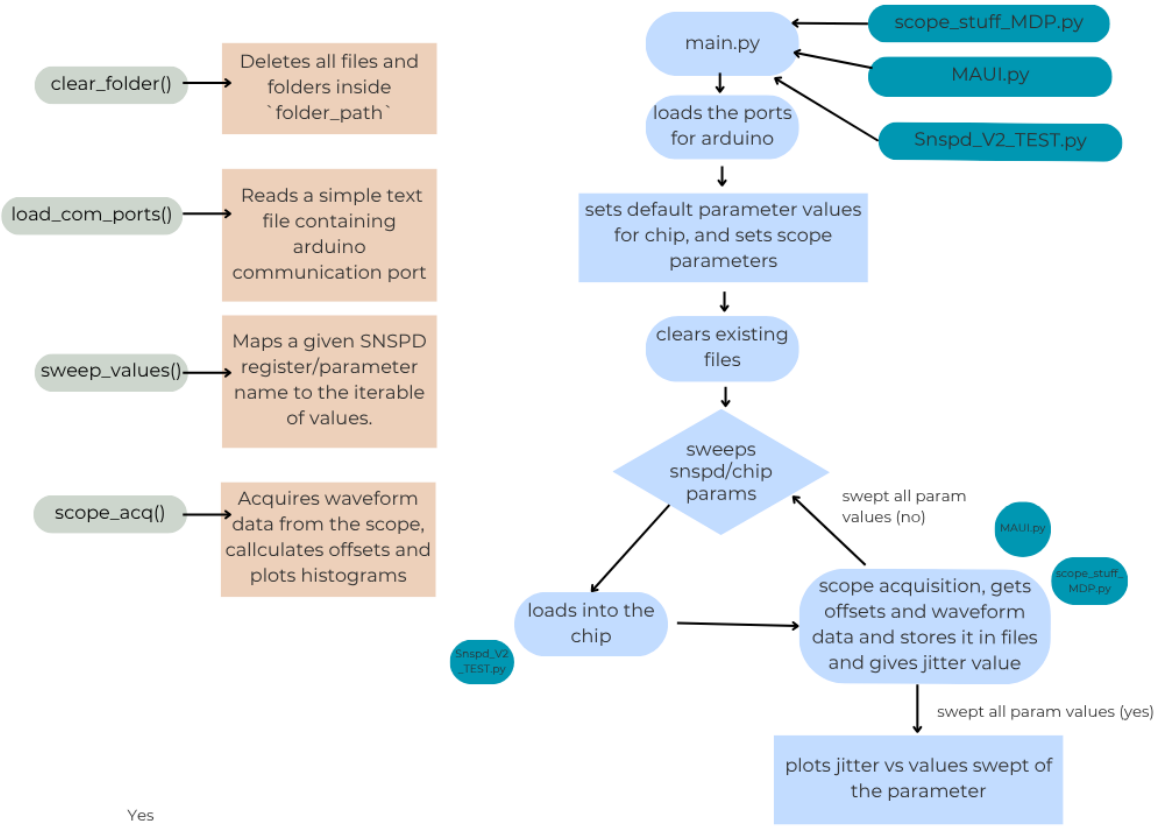


Figure 10: An overview of the experimental run, i.e, how we begin by changing parameters to control the chip to collect data from the scope to analyse jitter, the details of how scope acquisition is done and then processed to get offsets and data storage in files, refer to Figure 8 above

6. Processing Data

With the scope-PC interface established and the signal waveforms successfully transferred, the task still remains to actually calculate the delays between the many rising and falling edges. When plotted in a histogram, these offsets should form a Gaussian distribution. The FWHM (or standard deviation, depending on convention) of this Gaussian is the jitter we ultimately seek to measure.

A key constraint from the supervisors was that this delay calculation be done entirely by the PC. The scope has its own delay measurement functionalities, but processing raw waveforms gives users more choice in how detection events are defined. This also means handling the inevitable complications from noise or hardware quirks manually. Two situations were identified that may lead to errors in delay calculations: mismatches in the number of edges between channels, and a drift in the centering of the detected edges that occasionally led to outliers.

6.1. Handling Mismatches in Number of Edges

At this point, the PC has a sequence of rising/falling edge waveforms from both channels. The naive approach to finding the time offsets between them would then be just to find the times in the Channel 1 data where a rising edge goes above 50 mV, find the times in the Channel 2 data where a falling edge goes below 500 mV, and calculate the difference between these times. The code attempts to do just this. If exactly N edges are detected in each channel, the delay calculation is straightforward.

What happens more often than not – particularly for large N – is that the number of detected edges in each channel differs. In such cases, the waveform sequence arrays are broken into N individual segments. An entire sequence should have αN datapoints, where N is the number of triggered acquisitions and α is the number of samples per acquisition. The sequence is divided into individual segments by taking a numpy array of the shape $(\alpha N,)$ and reshaping it to (N, α). This creates N subarrays of α datapoints.

These are iterated over to get individual edge delays for each. In instances where a crossing is detected in one channel but not at all in another, that segment is disregarded.

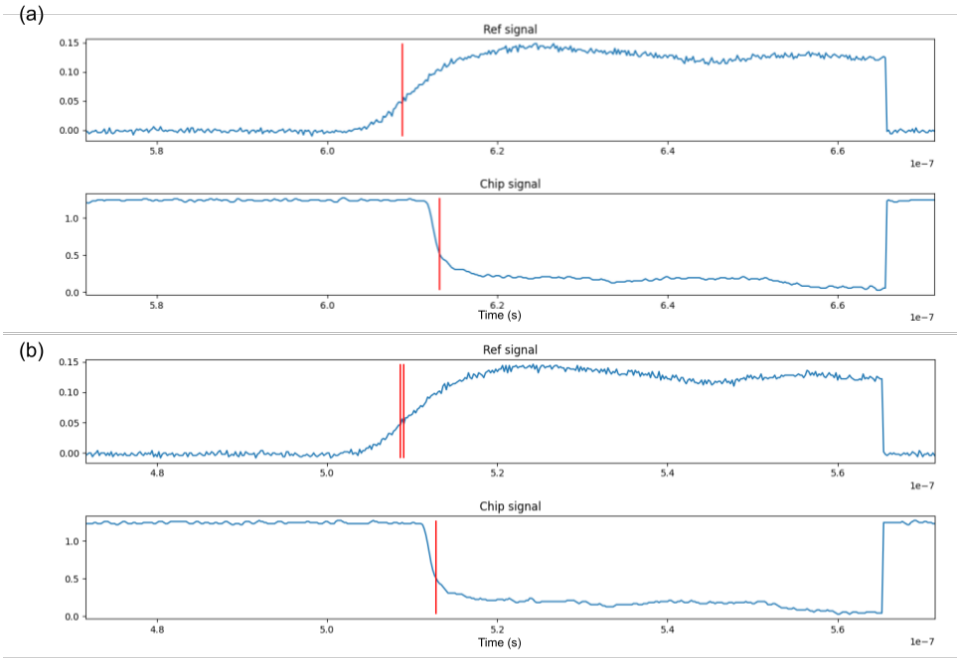


Figure 11: A single segment with detected crossing times marked. The discontinuities at the far right mark the end of one scope acquisition window and the beginning of another. These are included due to the window drift outlined in Section 6.2. **(a)** A single crossing is correctly detected in each channel. **(b)** A separate segment. In this noise in the reference signal leads to a mistaken second crossing detected.

Both signals, the AWG in particular, are prone to noise that registers as multiple edges. Such events are handled by setting the ”true” crossing time to be the average of multiple collected ones. This method is effective so long as the multiple crossings are in close proximity to each other as in Figure 11). These cases make up the large majority of multi-crossing events. In the rare instances where multiple crossings are detected far from each other, the corresponding delay will be filtered out via a standard deviation cutoff process described subsequently.

6.2. Segment Window Drift and Discarding Outliers

Another occasional issue arises when a sequence is broken into constituent segments. There will sometimes be more than α bytes per segment in a sequence. This is a quirk of the scope the team doesn't fully understand (perhaps some per-segment metadata). It means the segments aren't perfectly, periodically divided. This is outlined in Figure 12. It isn't an issue so long as edges in each channel maintain their relative distances from each other. But it can cause an occasional crop of dramatic outliers that are ultimately filtered out after-the-fact.

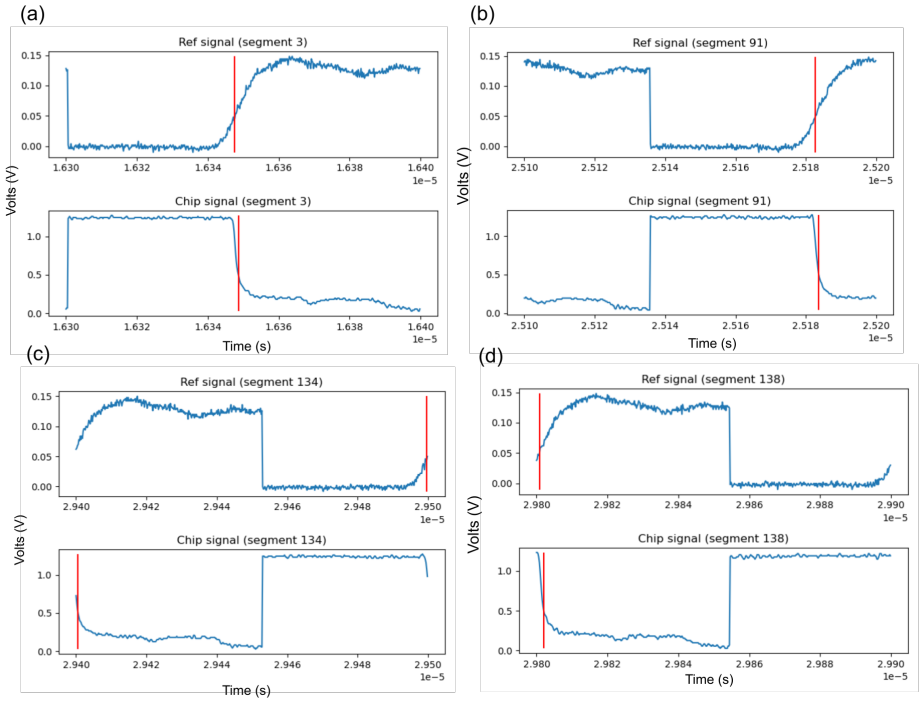


Figure 12: Segments from a larger sequence of signals. Large discontinuities mark the transition from one scope acquisition to the next. As the segment number increases, the threshold crossings shift right due to deviations in samples per segment. For **(a)**, **(b)**, and **(d)** this has no impact on delay calculations. However **(c)** yields an incorrect, larger offset time of 100 ns that is discarded later in processing.

After calculating all timing offsets between edges, data will follow a largely Gaussian distribution. Due to the drift phenomena described above, there will be collection of outliers bunched around a delay roughly the width of the acquisition as in Figure 12c. Let data including such outliers be referred to as ”total data”. The standard deviation of the total data is σ_{tot}. These outliers are infrequent enough to be many σ_{tot} away from the mean, but substantial enough to alter the fitted σ we ultimately use for our jitter measurement.

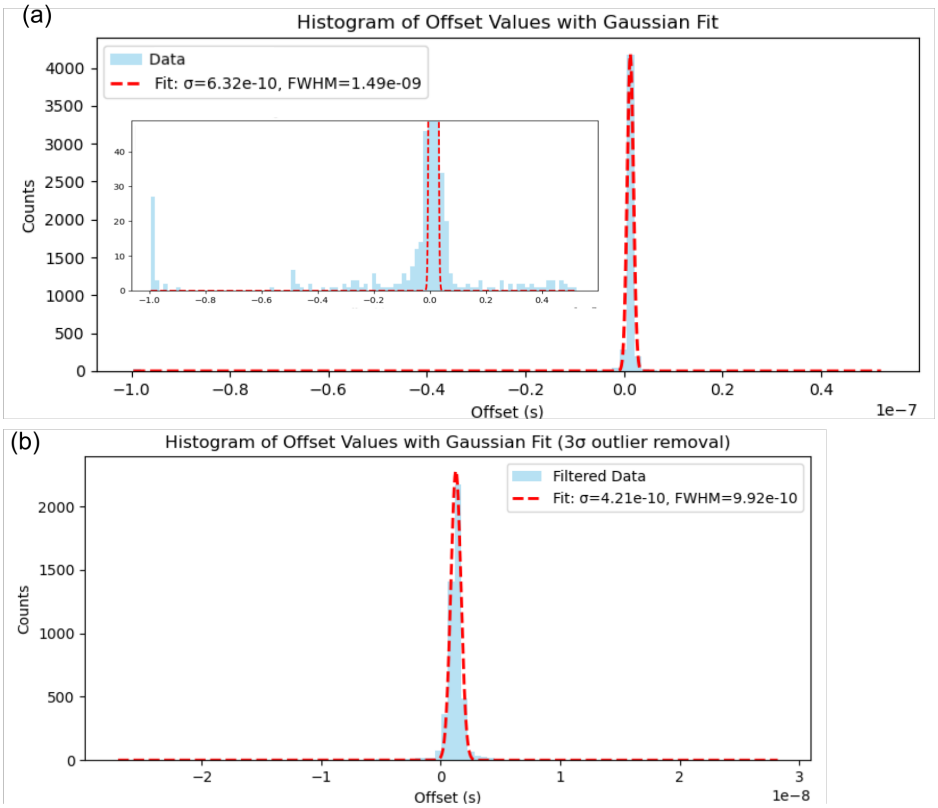


Figure 13: Offset histograms for a dataset with 10,000 triggered acquisitions. The mean offset time appears smaller than in reality it because deskew has been set to 30 ns. **(a)** No outliers are filtered from the calculated edge offsets. The inset zooms in such that outliers are visible. The leftmost spike of counts corresponds to window drift miscalculations. **(b)** Offsets from the same dataset, but with all datapoints more than $3\sigma_{\text{tot}}$ from the mean.

After consulting with the supervisors, it was decided the best option was to remove data more than $3\sigma_{\text{tot}}$ from the mean, as there was a clear of understanding of why outliers were present and that they didn't represent meaningful phenomena.

6.3. Saving the data

Once all timing offsets are extracted, cleaned, and filtered according to the procedures described above, the resulting dataset is stored together with the full set of experimental waveform data. This is done directly within the main control script `main.py`, which coordinates both the parameter sweep and scope acquisition. For each parameter value in a sweep, the program records the corresponding jitter data, acquisition settings, and the exact register frame that was sent to the Arduino prior to acquisition. By saving the register state(particularly parameter name and its value), every data file can be traced unambiguously to the chip configuration under which it was collected due to the naming of the files saved.

For each sweep point, the calculated offset times between the reference and chip signals are stored as arrays, together with the fitted jitter metrics (mean, standard deviation, and FWHM) derived from the Gaussian fits of the offset histograms. Data such as the default parameter values, the number of acquisitions N, and the oscilloscope settings (deskew, trigger levels, and sample rate, etc) are stored alongside the data in the parameters.txt file.

All acquisition outputs are organised under a single top-level directory, `C:\LeCroy\ScopeData`(for example), which is cleared at the start of each run to avoid mixing results from different experiments. For every sweep point (i.e., each value of the currently swept parameter), `main.py` creates parameters.txt file and three subfolders named after the parameter and its value: `ReferenceWaveforms_{param}{val}`, `ChipWaveforms_{param}{val}`, and `OffsetVals_{param}{val}`.

Within each loop of a sequence acquisition, the reference and chip waveforms are saved as NumPy

arrays with zero-padded loop indices `ref_data_{000..}.npy` and `chip_data_{000..}.npy` inside their respective folders. These arrays contain the time axis and voltage samples returned from the scope interface. Before saving, the script adds a deterministic time offset to each loop so that time stamps remain unique across the full burst. In parallel, the per-loop edge-offset results (the computed delay values for that loop's sequences) are written as plain text files named `offset_vals_{param}{val}_{000..}.txt` inside `OffsetVals_{param}{val}`. After all loops finish for that sweep point, the script concatenates every per-loop offset file, preserving their numeric order into a single combined list stored at the top level as `offset_values_all_{param}{val}.txt`, then deletes the per-loop text files and their now-empty offset folder to save up space on the computer. Then builds a histogram, rejects outliers (e.g., cutoff at 3σ), fits a Gaussian to extract the jitter statistics, and saves the figure as `hist_{param}{val}.png` in the top-level directory. Once a parameter's sweep is complete, the script also saves a summary plot of jitter versus parameter value where the main.py file is located, saved as `jitter_vs_{param}.png`, with error bars derived from the fit. All of the description made until now can be seen in Figure 14

This final step closes the experimental loop illustrated in Figure 10 in the report. After the chip is configured and verified, the scope acquires and transfers waveform data, the PC extracts and processes jitter information, and the results are automatically logged. This minimises operator error and ensures that every sweep point is paired with its correct configuration and analysis output, which is especially important for large sweeps involving tens of thousands of triggered acquisitions.

Figure 14: Clear hierarchical file and folder structure of how the data gets saved.

7. PCB Design

To interface the chip with external devices, a dedicated PCB with suitable connectors was designed. In the room-temperature setup, these connectors are directly linked to the oscilloscope and Arduino. In the cryogenic setup, the connectors will interface with the cryocooler and use the preinstalled wiring of the cooling system.

The PCB must be compatible with multiple generations of the chip. This allows testing and debugging of the entire system using the currently available chip, while ensuring future compatibility with newer chip iterations.

Connections to the chip include both high-speed signals (such as SPI) and low-speed DC power lines. FFC connectors are used for DC signals, while UMCC connectors are employed for high-speed RF connections. Additionally, certain signals must be shared between multiple chips on the board. To facilitate this, a two-PCB stack configuration was chosen.

The overall setup must also be compatible with several cryostats. The design was based on the most space-constrained model under consideration, ensuring compatibility with other, larger systems.

Figure 15 shows the two PCB configurations. The bottom PCB houses the chip, while the top PCB carries the connectors used for RF multiplexing and DC interfacing.

The following sections discuss these design constraints in more detail. The design process progresses from implementing mechanical requirements, to selecting the stackup, defining the schematic, and completing the layout. Although presented sequentially, this process is iterative in practice, with multiple refinement cycles before finalizing the design.

7.1. Mechanical Configuration

Mechanical constraints are the primary design drivers for the PCB. The reference cooling system is the OptiCool optical cryostat, which includes a removable sample pod for mounting experimental setups, as shown in Figure 16. The sample pod accommodates circular samples up to 60,mm in diameter.

In this design, the PCB is mounted on a copper riser (Figure 15a) rather than directly on the sample pod. The riser includes a central copper protrusion providing a direct thermal interface for the SNSPD. This feature aligns the SNSPD height with the adjacent chip, enabling direct wire bonding between them. The bottom of the PCB has exposed copper to minimise thermal resistance between the pod and the board.

Stackup and Design Rules

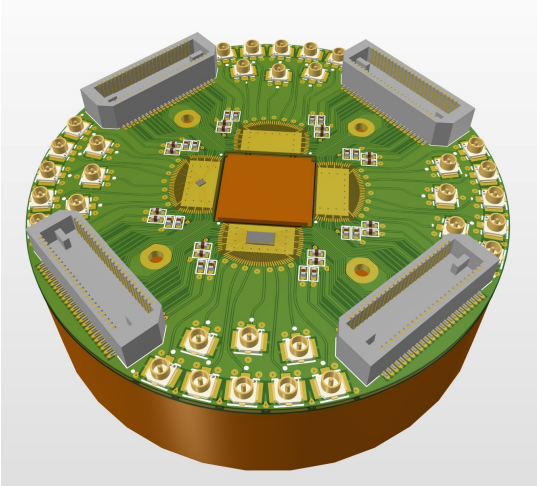
The next key design decision concerns the PCB stackup, which defines the copper layer configuration and dielectric spacing. The stackup also dictates the design rules, such as minimum trace width, clearance, and via dimensions, as determined by the manufacturer (Eurocircuits).

While modern PCB fabrication supports advanced technologies such as microvias and fine traces, these are typically cost-effective only for large-scale production. Since this project involves low-volume manufacturing, a conventional four-layer stackup was selected to balance cost and performance.

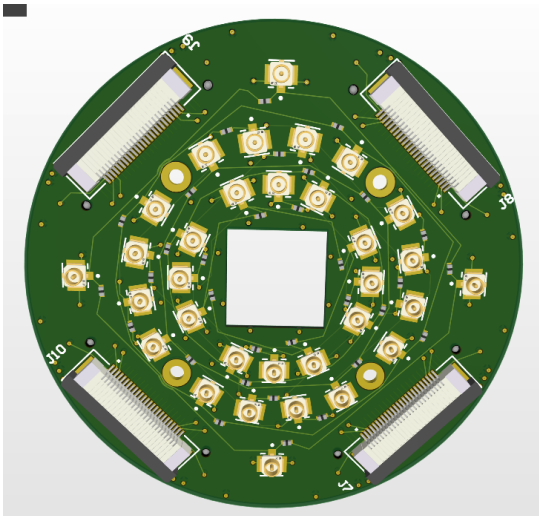
The design does not use microvias (vias connecting only adjacent layers). As discussed above, the bottom layer of the bottom PCB is exposed copper and grounded. Each through-hole via therefore contacts the grounded copper mount, meaning only ground vias can be implemented. Consequently, all signal routing must occur on the top layer. This constraint prevents direct interconnection of signals between chips on the same board, a limitation resolved by using a second PCB in the stack for inter-chip routing. This top PCB uses the top layer, 2nd layer and the 4th(bottom) layer for signal routing and the 3rd layer was made to be ground.

For the bottom PCB although only the top and bottom layers are used for active routing, a four-layer stackup was selected. This enables well-controlled 50 Ω impedance traces, which would be too wide to route efficiently on a two-layer board. Of the four layers, three serve as ground planes and only the top layer carries signal traces. Impedance-controlled RF lines are discussed further in Section 7.3.

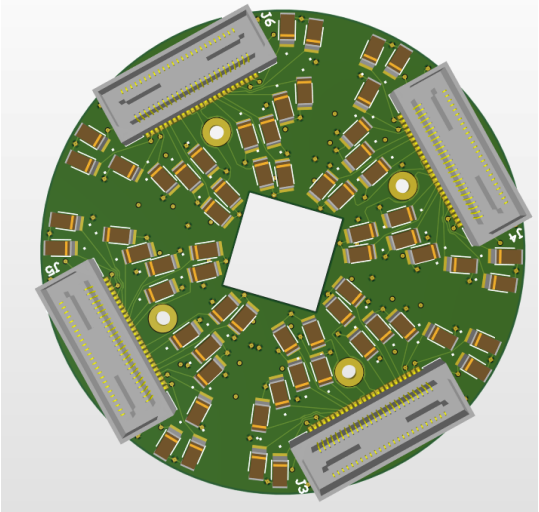
For the top PCB, only the bottom layer has the 50 Ω impedance traces; the trace width on the top layer and the 2nd layer is too thin. But since there are traces connecting multiple RF connectors on



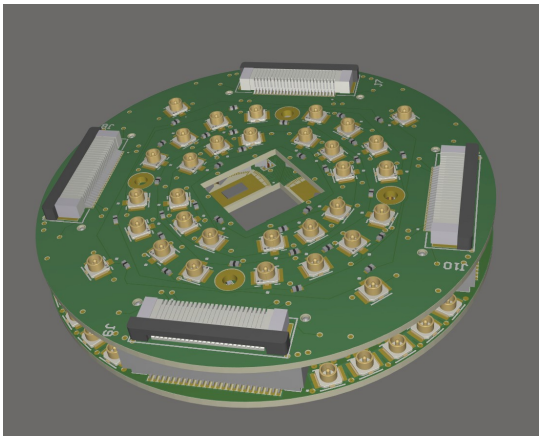
(a) Bottom PCB showing two generations of chip mounted. Two of the chip pads are intentionally left unpopulated. The central PCB cutout exposes a copper mount, where the SNSPD will be directly attached and wire-bonded to the chip.



(b) Top layer of top PCB. The central PCB cutout is made to leave space for light to enter to the SNSPD. This layer has additional RF connectors which could be connected to the bottom PCB with added configurability.



(c) Bottom side of top PCB. This layer has Decoupling capacitors connected to the pins of board-to-board connector.



(d) Both PCBs assembled on top of each other.

Figure 15: (a) and (b) show the two PCB configurations: (a) bottom PCB with chip mount, and (b) and (c) top PCB top and bottom sides respectively, (d) stacked PCBs with interconnects.

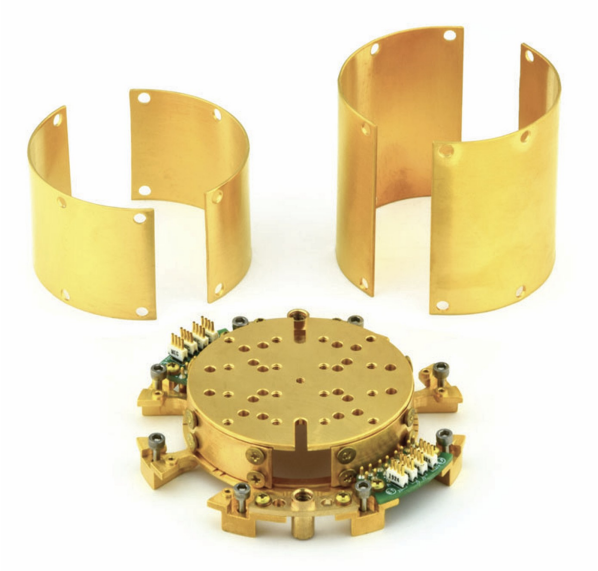


Figure 16: Sample pod of the OptiCool cryostat [3].

the top layer, causing reflections anyway, we have relaxed this condition for the top PCB, assuming this condition doesn't add too many reflections. Additionally, connecting multiple RF connectors will inherently create stubs, which also adds to the reflections. The length of the stubs are around 60mm. Using the rule maximum of 7.62mm of stub per Gbps digital signal will allow a maximum digital signal of $127Mbps$.

7.2. Schematic

The schematic of both the PCBs is provided in Appendix B. It defines all components and interconnections that are realised during layout. For the bottom PCB, the chip and its footprint are defined as separate schematic symbols, allowing multiple chip generations to be supported. This approach clarifies which pins are actively connected and which remain unused. For the top PCB, the schematic is defined in two layouts, one for the routing/placement of components in the top layer and the other for the bottom.

UMCC connectors (TE Connectivity) and 50-pin board-to-board BTS connectors were chosen to meet the tight space constraints.

The pinout of the first-generation chip is given in Table 1. The next-generation chip is designed to interface with multiple SNSPDs. Instead of requiring one chip per SNSPD, a single chip can now connect to several SNSPDs and output results via the SPI interface. Consequently, the number of PCB connections does not increase significantly, although the number of wire bonds scales with the number of SNSPDs. It has been chosen to support up to 32 pads to the SNSPD, which is up to 16 SNSPD connections.

Since wire bonding across the chip package can lead to shorts, only one side of the chip's pads is used for SNSPD connections. The PCB, therefore, provides connectors for most pins, allowing flexibility for the chip designer to decide which pads to use or leave unconnected.

7.3. Layout

During the layout stage, the schematic is translated into a physical PCB design. Figure 17 shows the layout of both PCBs. The bottom PCB's layout includes fan-outs from four chip footprints to four board-to-board connectors and four groups of eight RF connectors. For the top PCB, all the signal layers are shown. The top layer has all the RF Connectors, which connect to the bottom board and the FCCs, which connect to the board-to-board connectors. The bottom layer has the decoupling capacitors that connect to the pins of the board-to-board connector. Layer 2 has some more signal routings for the RF connectors, which couldn't be done on the top layer; all the placements and the routing are well

Table 1: Pin mapping and connections for the IC.

Pin No.	Pin Function	Explanation	Connection Type
1	NC	Not connected	–
2	NC	Not connected	–
3	Vref comp	Reference voltage	DC
4	Vref FB	Reference voltage feedback	DC
5	Vbulk	Bulk voltage	DC
6	Input+	Positive connection to SNSPD	Wirebond
7	Input–	Negative connection to SNSPD	Wirebond
8	Iref	Reference current	DC
9	Digital out	Output of the chip	RF
10	SR Din	SPI digital input	RF
11	Vss	Ground	DC
12	SR Load	SPI signal to load registers	RF
13	SR CLK	SPI clock signal	RF
14	SR Dout	SPI digital output	RF
15	NC	Not connected	–
16	NC	Not connected	–
17	NC	Not connected	–
18	Vdd digi 1.1	Digital power supply (1.1 V)	DC
19	Vdd analog 1.1	Analog power supply (1.1 V)	DC
20	Vdd 2.5	Power supply (2.5 V)	DC
21	NC	Not connected	–
22	Vdd amp 1.1	Amplifier power supply (1.1 V)	DC
23	Vdd meas 1.1	Measurement power supply (1.1 V)	DC
24	Analog Out	Analog output (unused)	–

thought out to fit in the small over footprint that is allowed.

Chip Footprint

Section 7.2 introduced the pin configuration of the first-generation chip, which supports up to 32 pads on the long side. For this design, 16 pads per short side were chosen. This number provides sufficient connections for DC lines while maintaining a compact footprint with a practical aspect ratio.

The footprint design accounts for several manufacturing and assembly constraints. The minimum clearance of 100,μm, required by the manufacturer, results in the footprint being significantly larger than the chip itself. This spacing necessitates wire bonds that fan out radially, which can lead to overlapping or shorting if pads are placed in a strictly linear arrangement. To avoid this, the bonding pads are positioned along an arc, reducing the chance of overlap and minimising differences in wire bond length, which also benefits signal integrity.

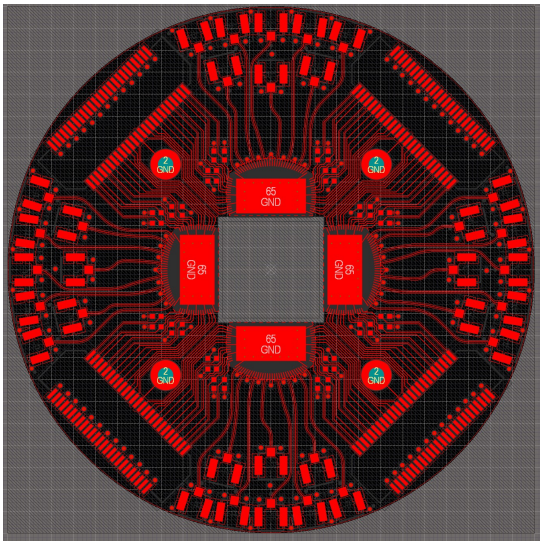
Figures 18a and 18b show the footprint implementation for both the first-generation chip and a potential next-generation version. Both chips are compatible with the same footprint, ensuring reusability and flexibility for future designs.

Connector Layout

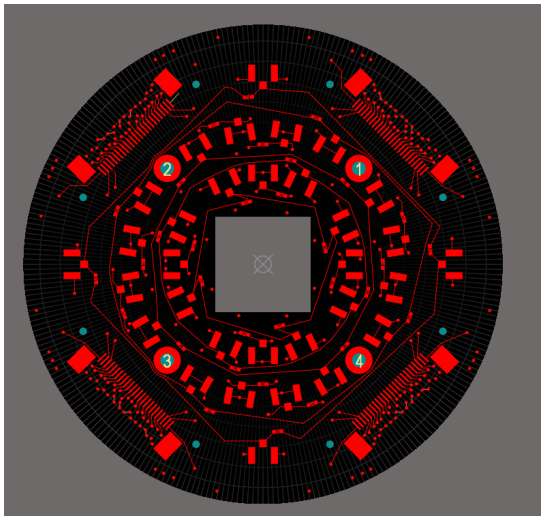
The connector layout was designed to accommodate both DC and high-frequency RF connections while fitting within the cryostat's spatial constraints. The top PCB contains the FFC connectors for DC power and control signals, while UMCC connectors are used for high-speed RF lines. The relative placement of connectors minimises trace length and crosstalk while maintaining accessibility for wire bonding and assembly.

RF Connections

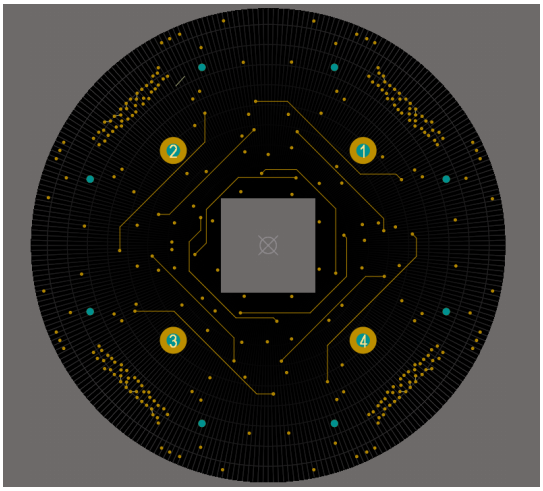
At low frequencies, DC and quasi-static signals can be modelled as having instantaneous potential changes across a trace, neglecting propagation delay. However, as frequency increases and the signal wavelength approaches the physical length of the transmission line, this assumption becomes in-



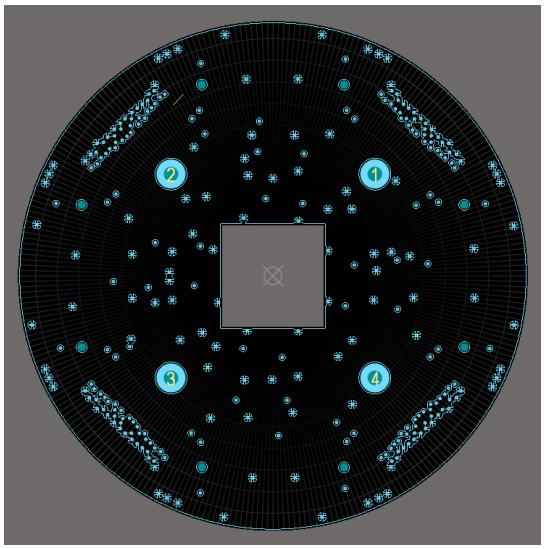
(a) Layout of the bottom PCB, showing the top layer. The design includes fan-outs from four chip footprints to four board-to-board connectors and four sets of eight RF connectors.



(b) Layout of the top PCB, showing the top layer. The design includes RF connectors and jumpers used to make connections between the RF connectors, which then connect to the bottom PCB via cables. An FCC connector, which connects to a board-to-board connector on the bottom layer, is also present on this layer.

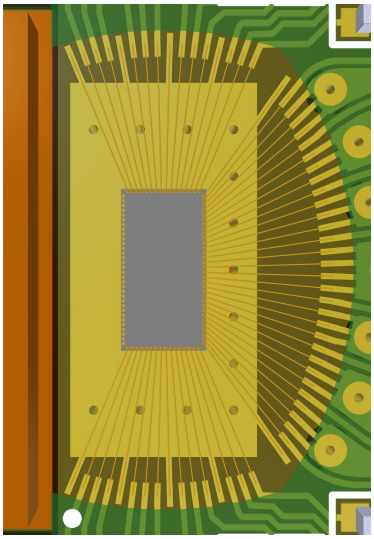


(c) Layout of the top PCB, showing the 2nd layer. This layer has some of the routings connecting different RF connectors and jumpers.

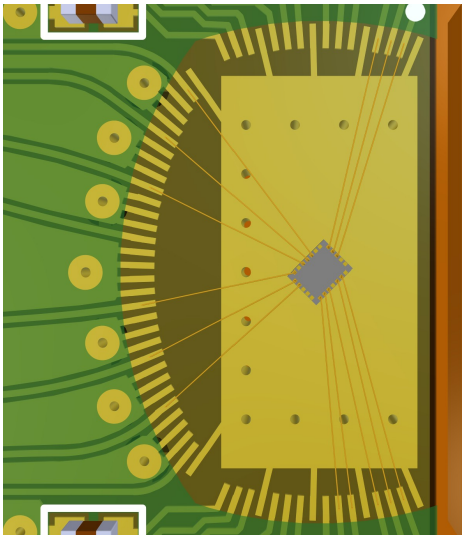


(d) Layout of the top PCB, showing the bottom layer. This layer includes connections to pins of board-to-board connector with decoupling capacitors.

Figure 17: Layout of both the PCBs.



(a) Footprint showing the largest chip supported by the PCB in terms of pin count.



(b) Footprint with the first-generation chip mounted, demonstrating backward compatibility.

Figure 18: (a) and (b) show two versions of the chip footprint on the PCB, demonstrating compatibility with different chip generations.

valid. In these cases, traces must be treated as transmission lines, which can exhibit reflections due to impedance mismatches.

Reflections occur at points where the impedance changes along the line. By maintaining a constant characteristic impedance of 50 Ω throughout the transmission path—including cables, connectors, and terminations—reflections can be minimised.

On the PCB, 50 Ω impedance is achieved using microstrip traces, where a signal trace runs above a continuous ground plane. The impedance depends on the trace width and the dielectric thickness between the trace and the ground plane. These parameters were calculated using both ECAD tools and the manufacturer's online impedance calculator. The manufacturer's calculator is considered more reliable, as it accounts for fabrication tolerances and process variations specific to their production methods.

Decoupling Capacitors

Decoupling capacitors are placed close to each power pin to reduce noise and counteract the inductive effects of the power distribution network. In previous designs, 200pF capacitors were used. Similarly, several 200pF capacitors are placed on the bottom PCB.

However, due to limited space, not all power pins can be decoupled on the bottom board. Therefore, an additional 100nF decoupling capacitors are included on the top PCB to ensure sufficient power supply stability across all voltage rails.

8. Outcome and Discussion

The central objective of the project was the development of an improved measurement apparatus, encompassing both the automation process and the printed circuit board (PCB) design. Unlike a conventional research paper, the focus here lies not on the measurement results themselves, but on the setup and methodology used to obtain them. As outlined by the project supervisors, detailed analysis of jitter measurements falls outside the intended scope of this work.

Nevertheless, successful jitter sweeps are presented as a proof of functionality, accompanied by a reflection on the project's progression upon completion.

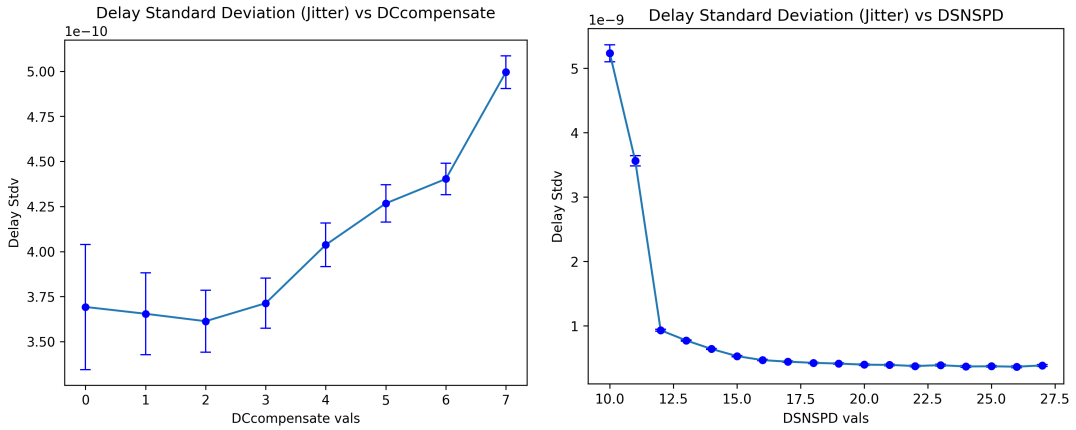
8.1. Results

Figure 19 summarises the output of the automated jitter measurement framework following full system integration. Each subplot corresponds to a parameter sweep or representative dataset collected using the procedures detailed in Sections 4–6. Specifically, subplots (a–g) show the measured standard deviation of delay, defined operationally as jitter, as a function of one of the programmable chip parameters. The automation scripts configured the chip via the Arduino interface (Section 5), triggered the oscilloscope in sequence mode (Section 4.2), and processed the resulting waveforms entirely on the control PC. For each configuration – i.e. for every data point in Figure 19 – approximately 50,000 triggered acquisitions were collected, filtered, and fitted to Gaussian distributions to extract the corresponding jitter statistics. This number isn't exactly 50,000 due to discarded outlier data.

This dataset demonstrates the capabilities of the automation code. In future applications, the same framework could be adapted to explore permutations of parameter values in order to identify the most optimal configuration for minimising jitter.

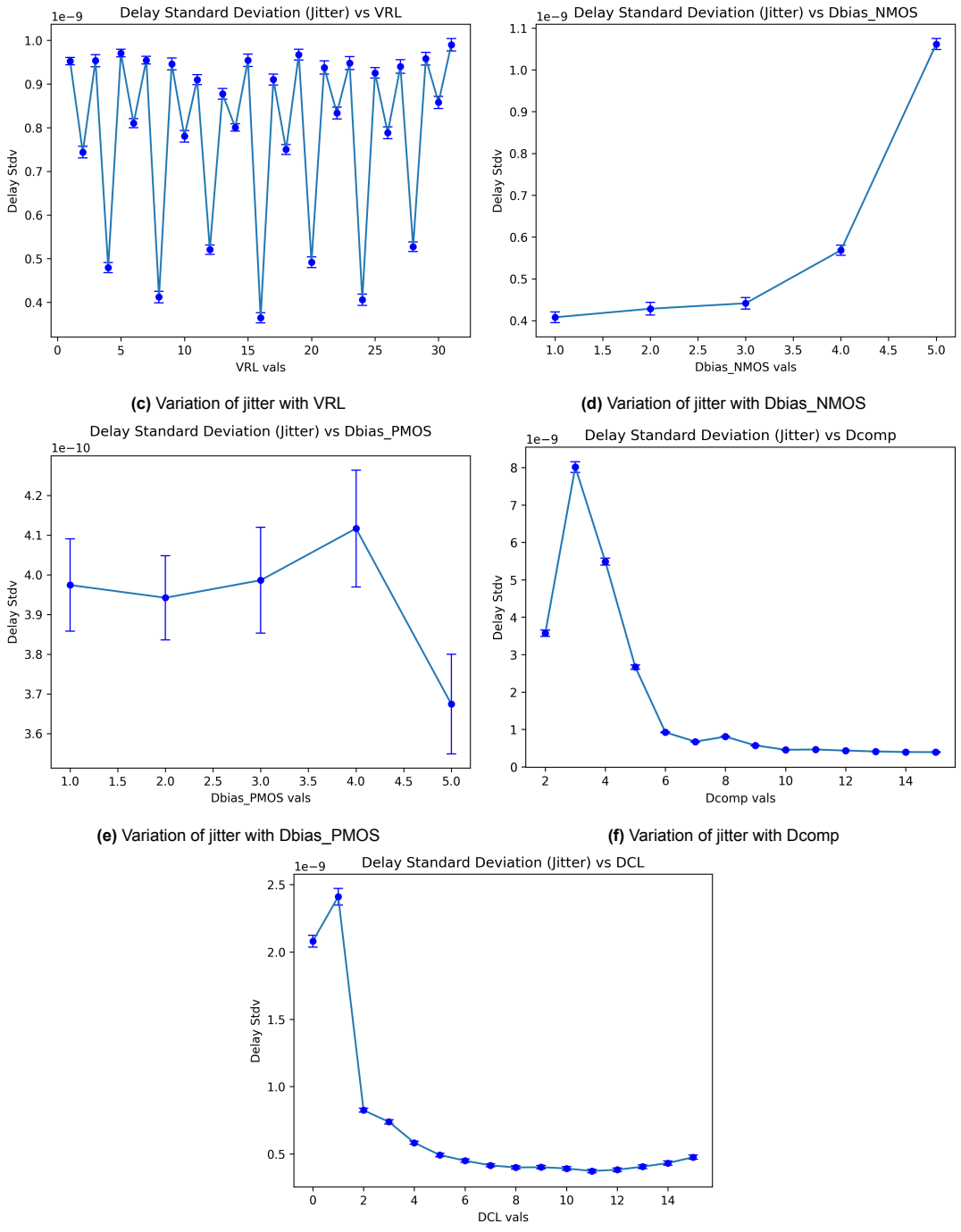
Overall, the plots validate the complete automation workflow, spanning from chip configuration through to jitter extraction and visualisation. While absolute jitter values are not analysed here, since their interpretation lies beyond the project scope, the consistency and reproducibility of the data confirm the robustness of the developed system. The observed trends illustrate the sensitivity of timing precision to analogue biasing, and highlight the strong dependence of jitter magnitude on threshold positioning.

The jitter values measured here are notably higher than the 60 ps FWHM (25.5 ps standard deviation) reported in prior studies. However, similar values were obtained using the oscilloscope's internal measurement functions. The project supervisors confirmed that discrepancies between literature values and these results will be addressed separately.



(a) Variation of jitter with DCcompensate

(b) Variation of jitter with DSNSPD



(c) Variation of jitter with VRL

(d) Variation of jitter with Dbias_NMOS

(e) Variation of jitter with Dbias_PMOS

(f) Variation of jitter with Dcomp

(g) Variation of jitter with DCL

Figure 19: Measured jitter dependence (units of seconds) on key programmable parameters of the cryo CMOS readout chip. Each subplot (a–g) shows the standard deviation of delay (jitter) as a function of one digital control variable. Parameters correspond to distinct functional blocks in the circuit shown in Figure 1.

8.2. Reflection on Success Criteria

The success criteria, as outlined in the Problem Statement and Project Goals section 2, were as follows:

1. Configure an experimental setup, including the assembly and connection of all components required to carry out measurements.
2. Successfully read out response signals from the chip via the oscilloscope and PC.
3. Establish remote control of the chip such that circuit parameters can be adjusted from the PC.

4. Develop and implement a process to obtain jitter measurements for different chip settings.
5. Create a graphical user interface (GUI) to serve as a user friendly front end for the automation software.
6. Prepare for future chip implementations involving multiple SNSPDs by designing a new PCB.

The first four criteria represent the fundamental elements of a functional experimental setup, all of which were achieved. Points 1–3 were relatively straightforward, as the software interfaced effectively with the relevant devices once they became available. A minor issue arose concerning oscilloscope firewall settings that temporarily blocked PC communication, but this was resolved promptly once identified.

Point 4 proved more demanding. Although establishing communication and data transfer between devices was simple, integrating these components into a cohesive and reliable process presented significant challenges. Early success with points 1–3 gave way to unforeseen technical issues, such as misconfigured acquisitions yielding unusable waveforms, undiagnosed data processing errors, and specific chip parameters causing unstable PC connections. These obstacles were anticipated given the project's technical complexity, and most were ultimately resolved. As illustrated in Figure **??**, the software now reliably performs parameter sweeps and plots jitter values with Gaussian fit uncertainty indicated by error bars. Point 4 was therefore successfully met.

The fifth criterion, relating to GUI development, was only partially achieved. The initial recommendation was to use Qt Creator, an industrial grade C based design platform unfamiliar to the project team. With the assistance of the AI programming tool Cursor, a prototype GUI was created during the first half of the project. However, because the tool generated much of the code automatically, addressing minor bugs and implementing new features became impractical. Consequently, GUI development was deprioritised in favour of stabilising the measurement and automation systems. A functional GUI was eventually produced using Python's Tkinter package, though response times remain slow when updating parameter fields. The aim is to deliver a refined version of this GUI to stakeholders in the following project phase.

The final criterion, concerning PCB development, was largely fulfilled. Two PCB designs were completed and verified to meet all manufacturing tolerances, ensuring cost effective production. Full validation, however, requires testing with chips wire bonded to the PCBs, an activity planned for future work.

8.3. Teamwork

Three primary areas of focus were identified in the project: the oscilloscope to PC interface, the chip to PC interface, and the PCB design. With three team members, responsibilities were naturally divided across these areas.

Timber, with an electrical engineering background and access to Altium design software, led the PCB design efforts. Bhoomika, also from an electrical engineering background, initially focussed on chip control and interfacing due to limited access to Altium and licensed workstations. Jonas, with an applied physics background, concentrated on the oscilloscope hardware interface and subsequent data processing.

As the project progressed, these roles began to overlap. Once the oscilloscope and chip interfaces were functioning independently, the work of Jonas and Bhoomika became increasingly integrated as the systems were combined. Bhoomika also contributed additional effort to PCB design. Timber's PCB work remained consistent throughout the project and, when required, he assisted with troubleshooting automation and processing issues.

This three pronged approach proved effective overall. Earlier integration of individual work areas could potentially have accelerated progress, but by the project's conclusion, interdisciplinary collaboration had been successfully established.

8.4. Navigating Hurdles

The initial phase of the project involved several administrative and logistical challenges. Difficulties with the formatting of laboratory access requests initially delayed work, but these were resolved through

in person meetings and email correspondence, an experience that underscored the importance of proactive communication and self advocacy.

Access to essential equipment presented another challenge. Up until Week 5, the oscilloscope was in use by another research group, leaving the team unable to begin data acquisition. This period, while frustrating, led to increased focus on PCB design and preparation. Time was also used productively to study oscilloscope control documentation and command structure, which later enabled rapid implementation once access to the instrument was granted. This preparatory work was instrumental in achieving success criteria related to device communication and control.

9. Conclusion

The objective of the project was to develop a testing framework that enables adjustment of chip parameters for jitter measurements in a robust and scalable manner. A series of success criteria was defined as benchmarks for progress, and all were met, with the partial exception of the graphical user interface objective.

The experimental setup was simplified to avoid the need for cryogenic operation, allowing for faster and more streamlined development. Although access to certain components was delayed, thorough preparation and organisation during the early stages of the project enabled rapid advancement once the required equipment became available.

The most technically demanding aspect of the work involved integrating chip control, oscilloscope control, and data processing into a unified system. Recurring errors in the data processing function were identified and corrected. A filtering procedure was implemented to remove outlier data caused by these errors, ensuring that the jitter measurements were based exclusively on valid and meaningful data.

Two PCBs were designed to satisfy the requirements for connecting multiple generations of chips to external devices. Final verification of the design will be performed once the PCBs have been manufactured and tested with the connected chips.

References

[1] G. Carboni et al. “Cryo-CMOS Readout of Single-Photon Detectors for Color-Center Quantum Computers”. In: *Proc. ESSERC* (2025).

[2] *MAUI Oscilloscopes Remote Control and Automation Manual*. Teledyne LeCroy, Inc, 2024. URL: https://cdn.teledynelecroy.com/files/manuals/maui-remote-control-and-automation-manual.pdf.

[3] Quantum Design GmbH. *OptiCool – Optical Sample Pods*. Datasheet: “1318-016 Rev. A0”. Quantum Design GmbH. Breitwieserweg 9, D-64319 Darmstadt, Germany, 2025. URL: https://qd-europe.com/fileadmin/Mediapool/products/quantum-design/opticool/_pdf/OptiCool_sample_pods.pdf.

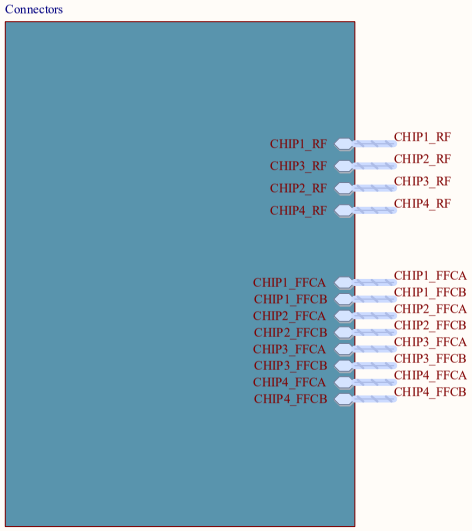
A. Automation Code

All code used for this report can be found at the following public repository: `https://github.com/jonas-kolker/SNSPD_Chip_Jitter_Automation`

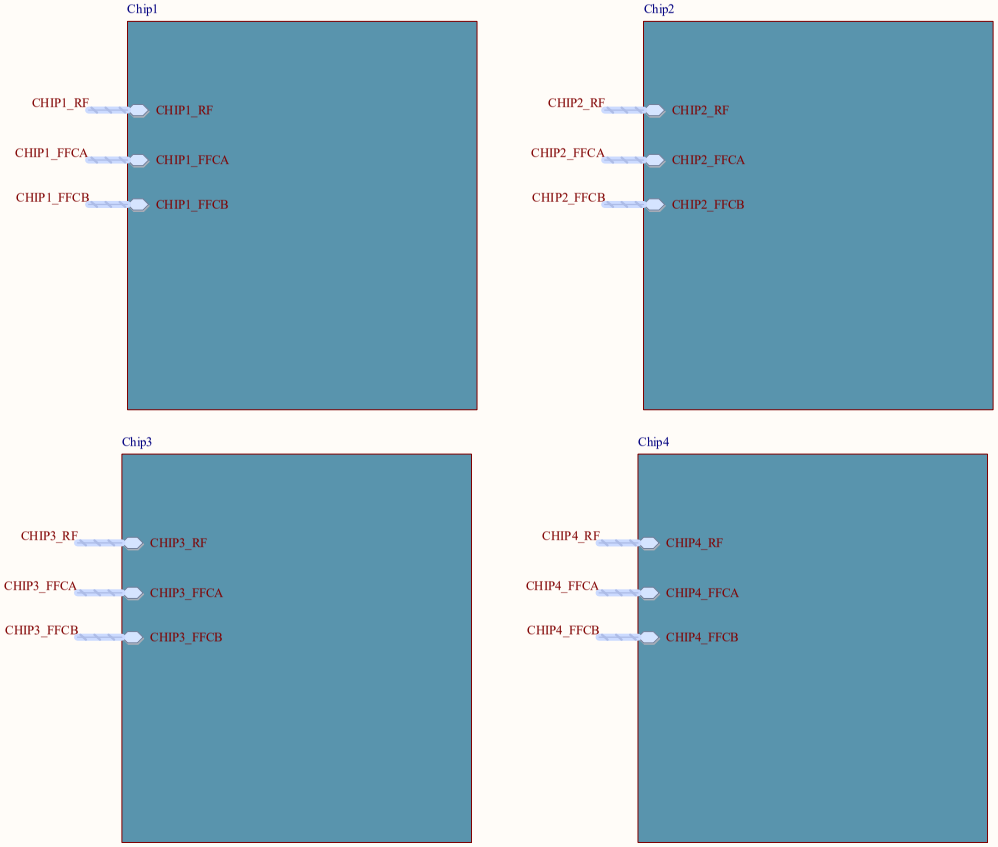
B. Schematics Chip PCB

The following pages contain schematic documents for the PCB.

CONNECTORS

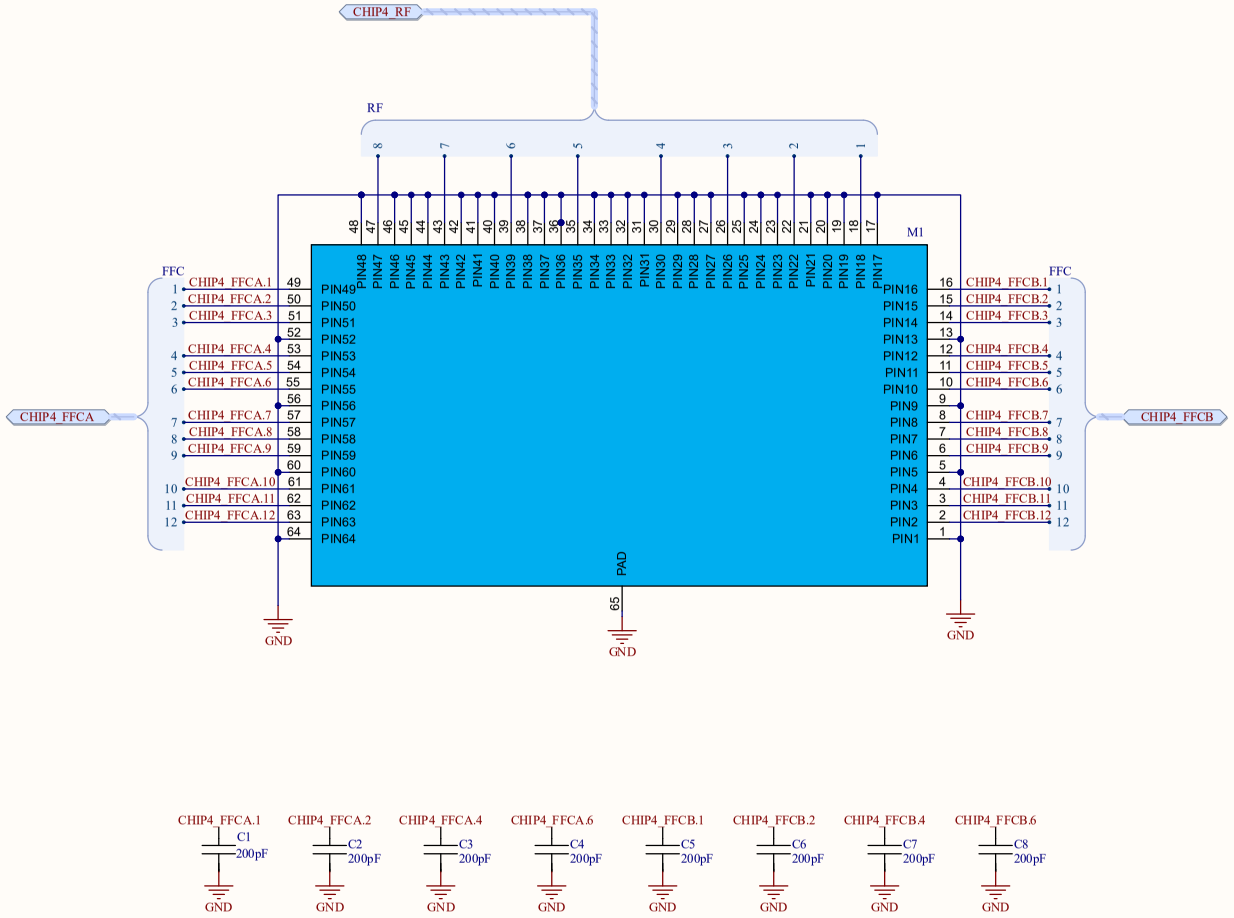


CHIPS



Title:	SNSPD Readout+Chip		
Drawn By:	Timber Lock	Revision:	V3.0
Date:	19/10/2025	Sheet	1 Of 6
Project:	SNSPD Readout+Chip V3.0.PrjPcb		

CHIP FOOTPRINT

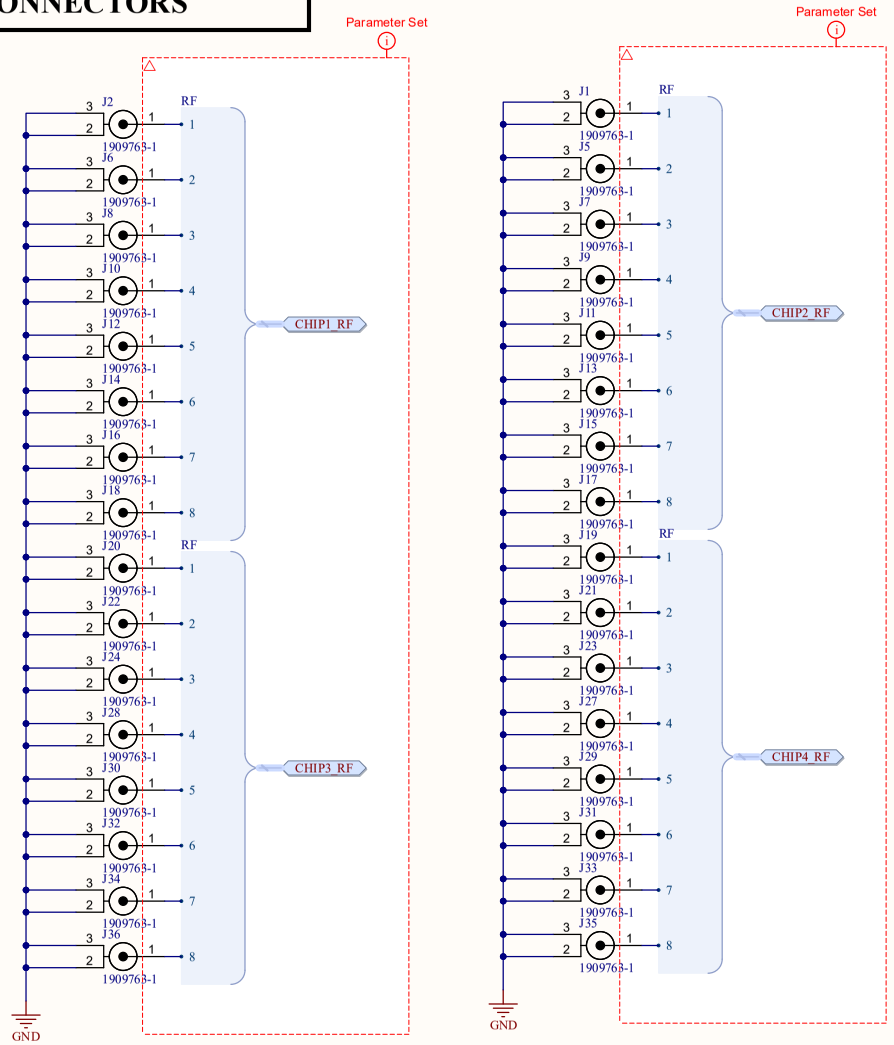


CHIP

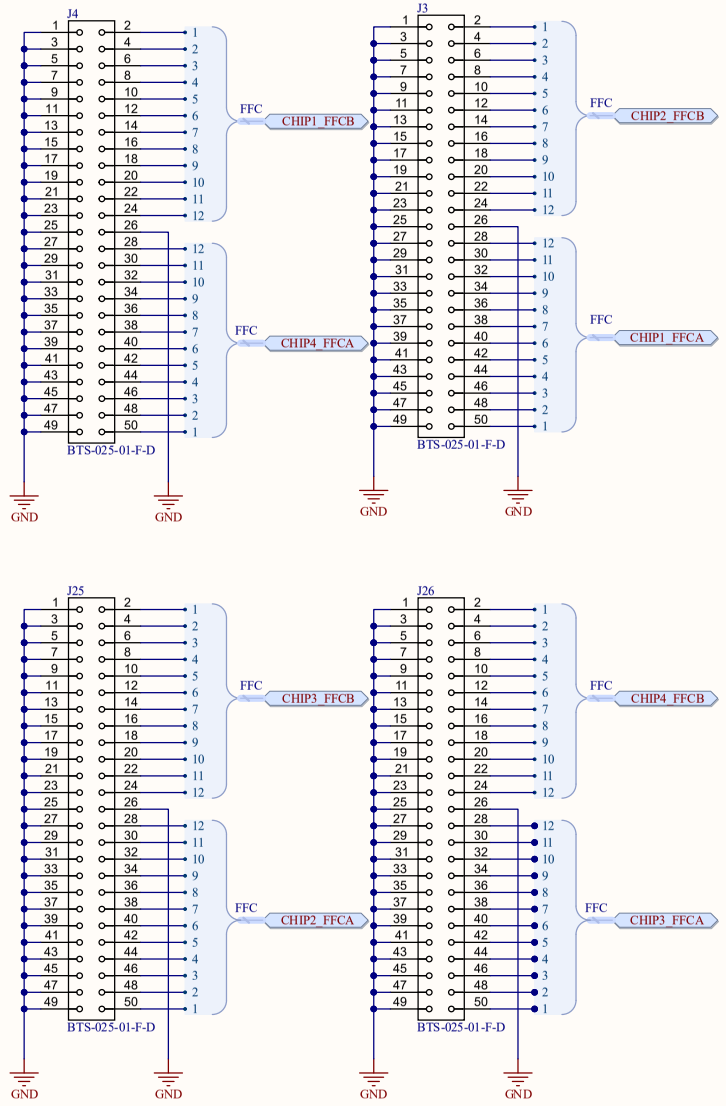
Title:	*		
Drawn By:	*	Revision:	*
Date:	27/10/2025	Sheet	2 Of 6
Project:	SNSPD Readout+Chip V3.0.PrjPcb		

QuTech

RF CONNECTORS



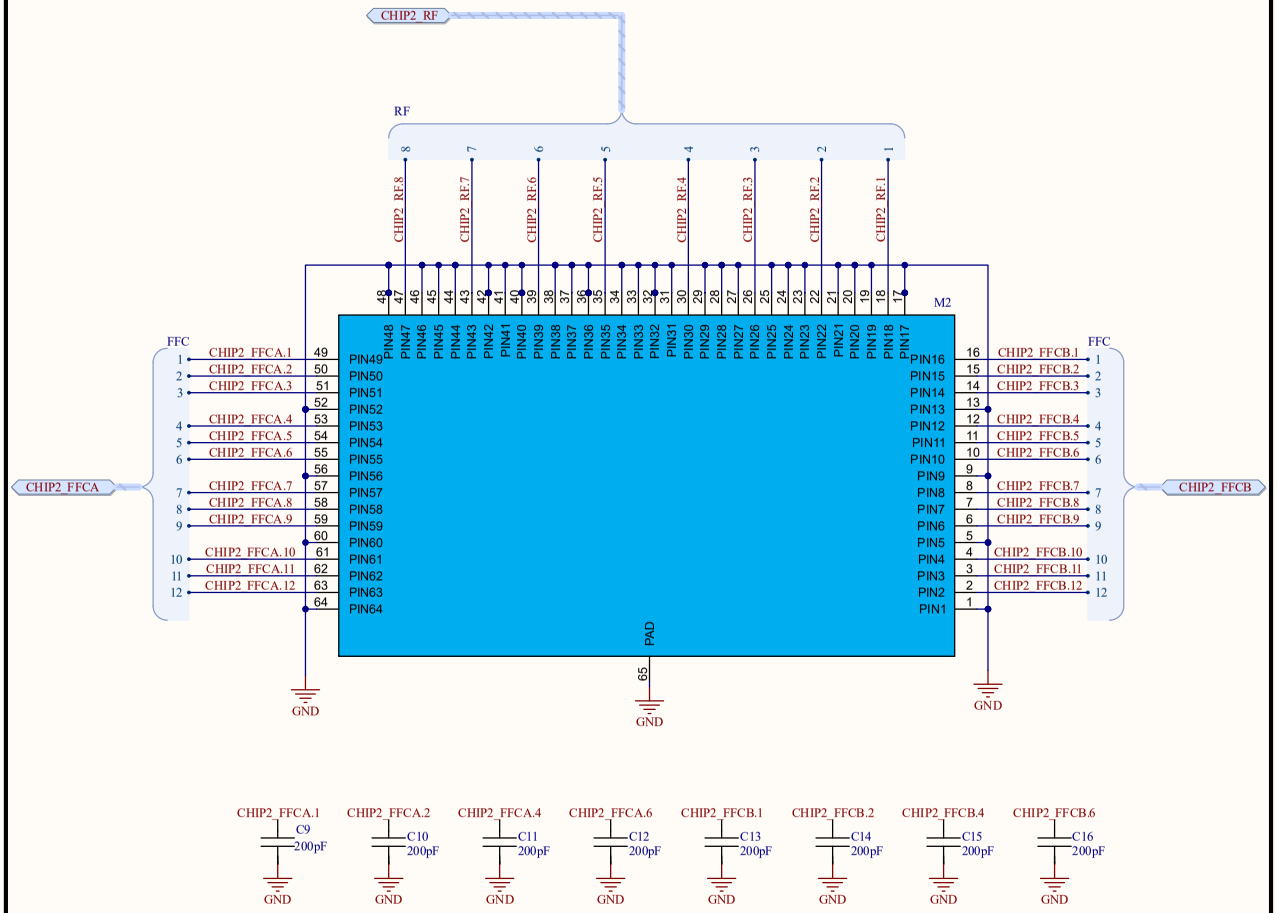
FFC CONNECTORS



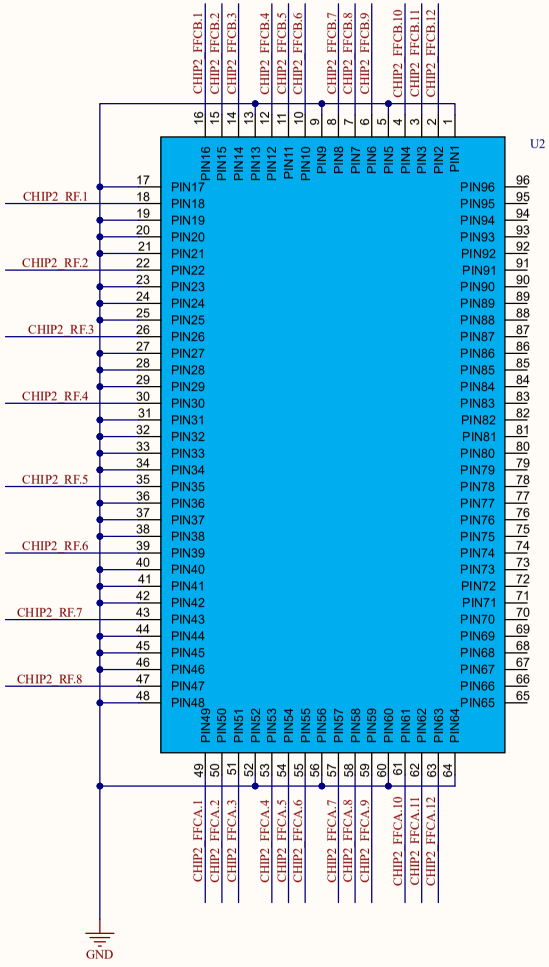
Title:	SNSPD Readout+Chip		
Drawn By:	Timber Lock	Revision:	V3.0
Date:	19/10/2025	Sheet	3 Of 6
Project:	SNSPD Readout+Chip V3.0.PrjPcb		

QuTech

CHIP FOOTPRINT

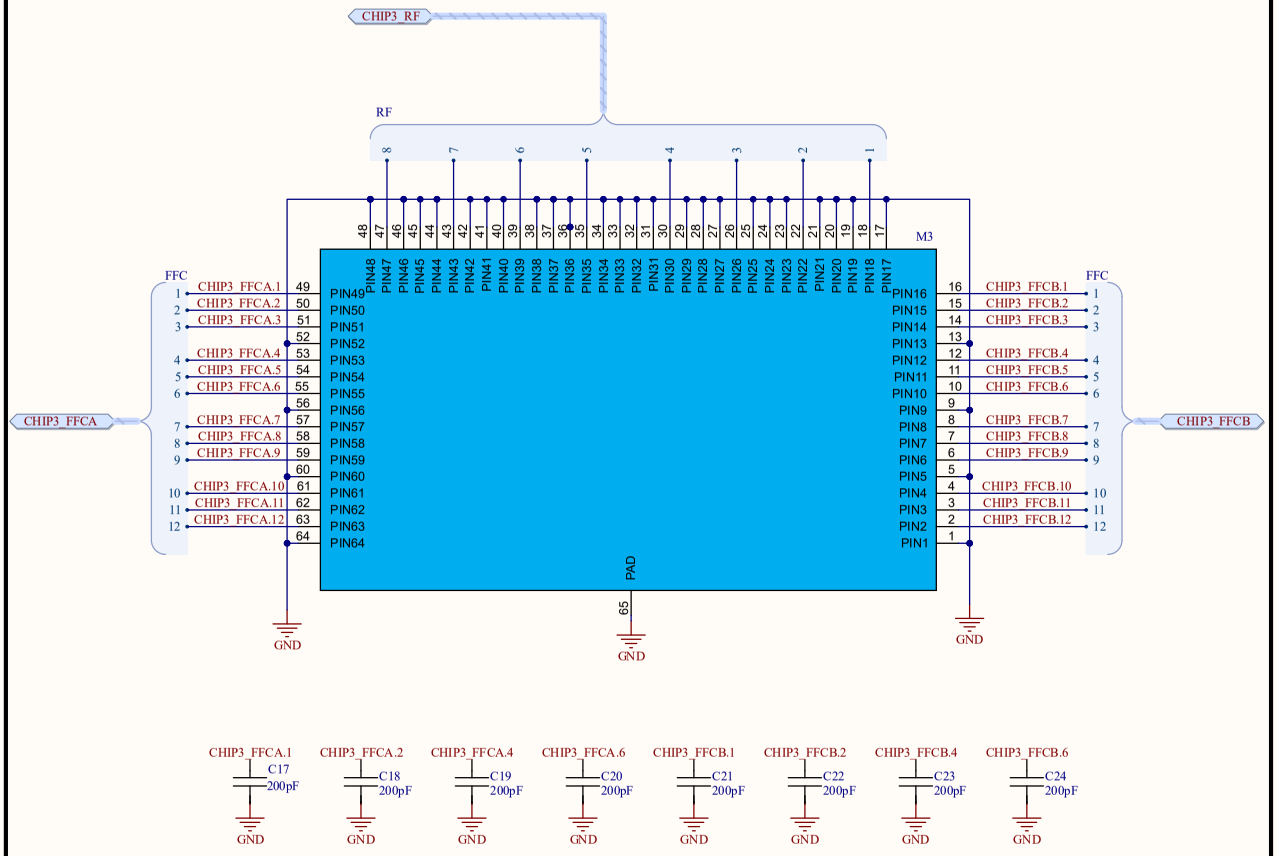


CHIP



Title:	*		
Drawn By:	*	Revision:	*
Date:	14/10/2025	Sheet	4 Of 6
Project:	SNSPD Readout+Chip V3.0.PrjPcb		

CHIP FOOTPRINT

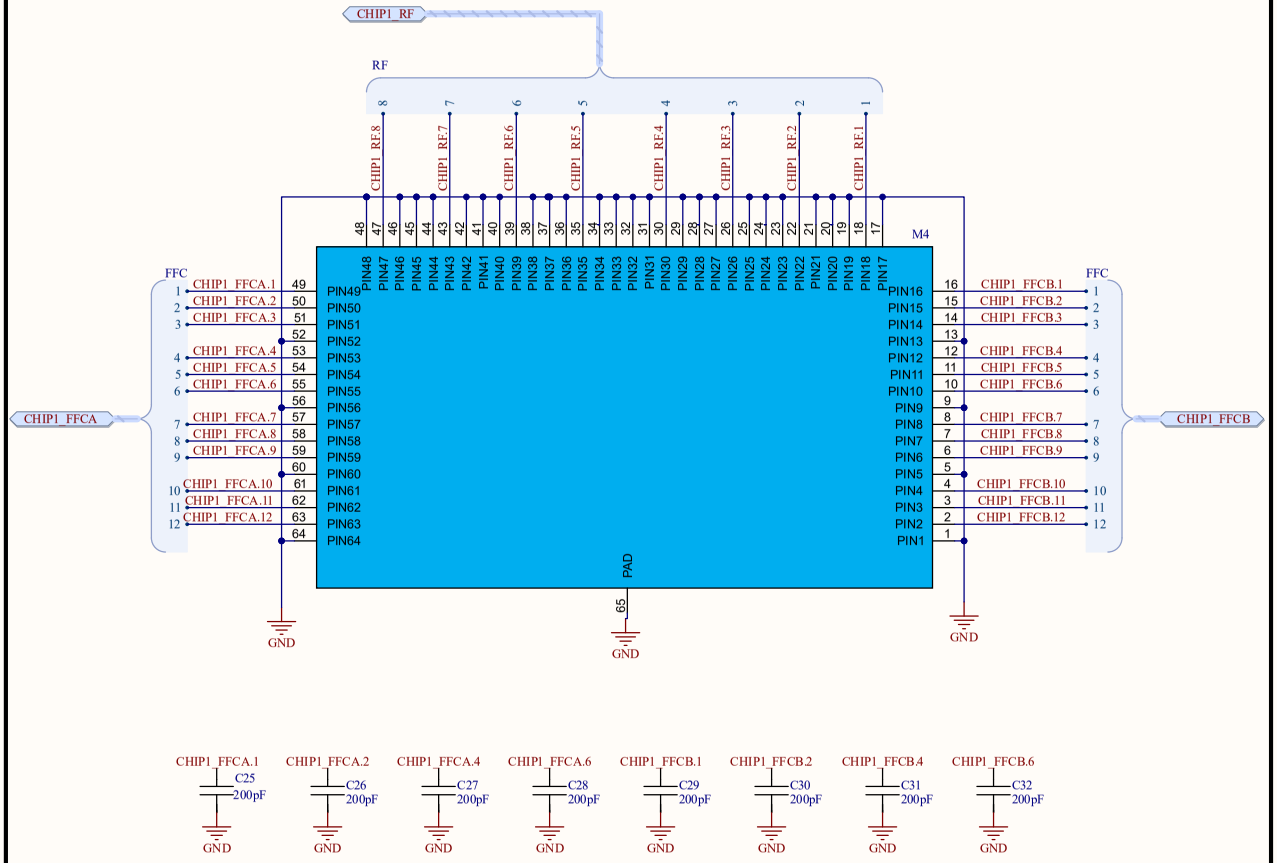


CHIP

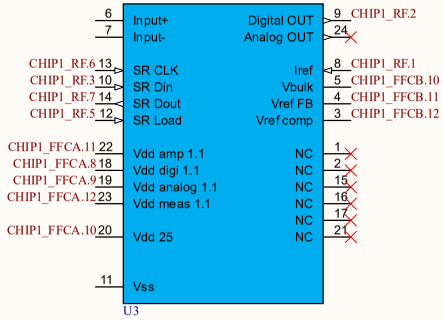
Title:	*		
Drawn By:	*	Revision:	*
Date:	14/10/2025	Sheet	5 Of 6
Project:	SNSPD Readout+Chip V3.0.PrjPcb		

QuTech

CHIP FOOTPRINT



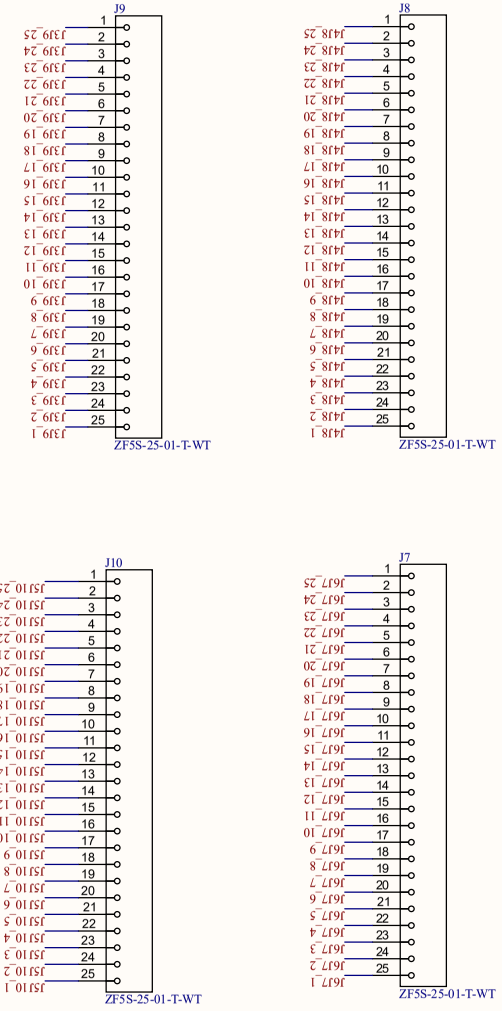
CHIP



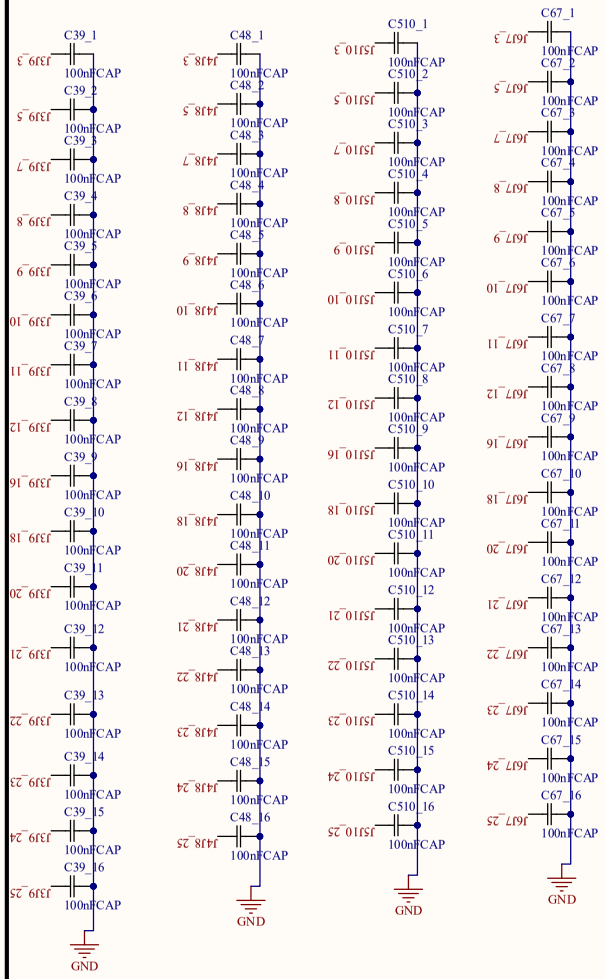
Title:	*		
Drawn By:	*	Revision:	*
Date:	19/10/2025	Sheet	6 Of 6
Project:	SNSPD Readout+Chip V3.0.PrjPcb		

QuTech

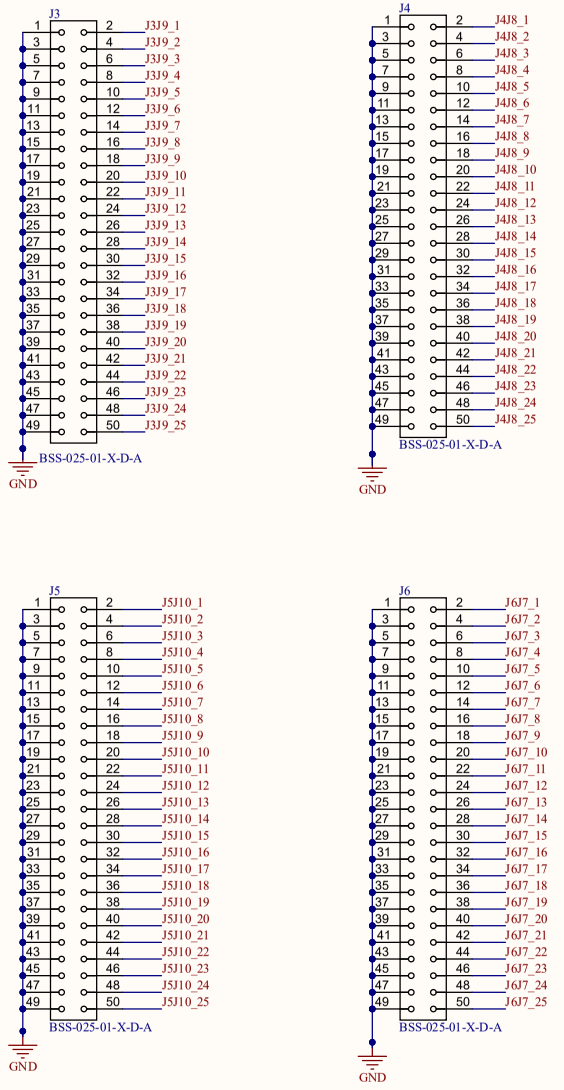
FFC CONNECTORS



DECOUPLING CAPACITORS

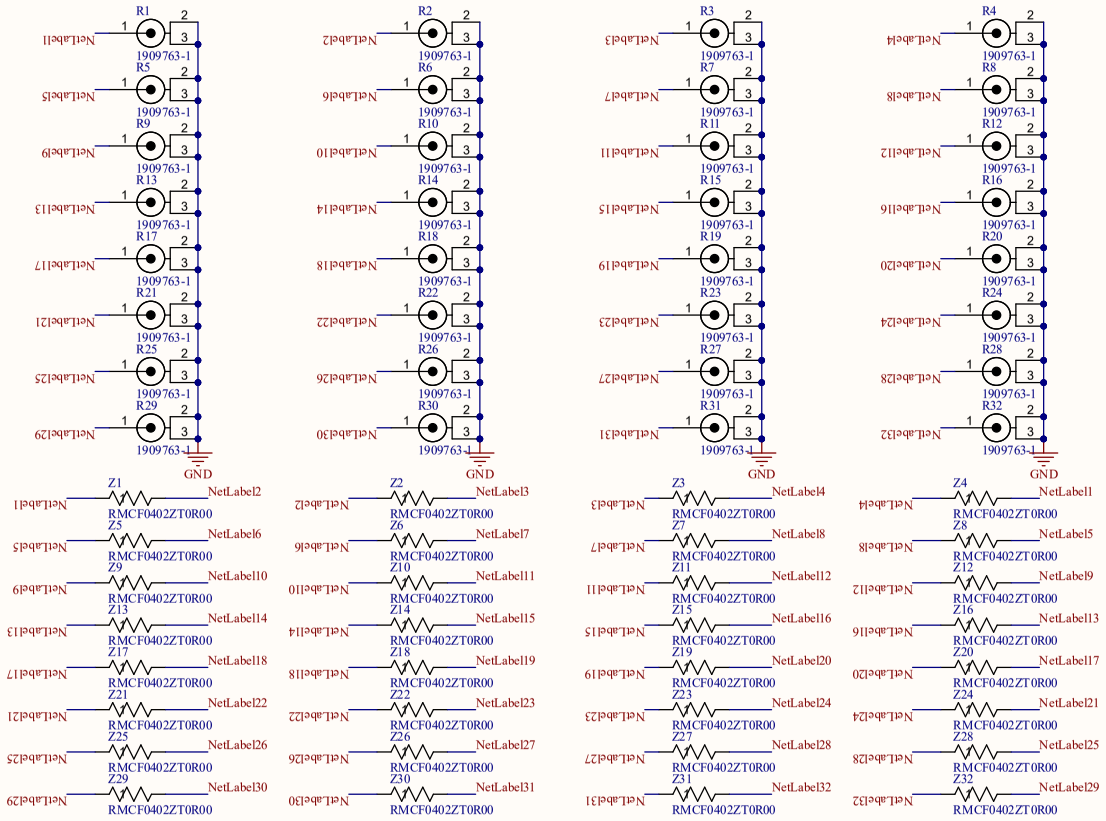


BSS CONNECTORS



Title:	Connectors to Decoupling Capacitors		
Drawn By:	Bhoomika Tanikonda	Revision:	V3.0
Date:	31-10-2025	Sheet	1 Of 2
Project:	connector_PCB.PrjPcb		

QuTech



Title:	RF Connectors on the top Board		
Drawn By:	Bhoomika Tanikonda	Revision:	*
Date:	31-10-2025	Sheet	2 Of 2
Project:	connector_PCB.PrjPcb		

QuTech