THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

A hardware signal processor for Transition Edge Sensors

Geoffrey Gordon Gillett

BSc.

A thesis submitted for the degree of doctor of philosophy at The University of Queensland in 2017

school of mathematics and physics

ARC Centre of Excellence for Engineered Quantum Systems

ARC Centre of Excellence for Quantum Computation and Communication Technology

Abstract

Transition Edge Sensors (TESs) are sensitive thermometers that exploit the sharp change in resistance with respect to temperature in a metal during the transition between the superconducting and normal states. TES sensitivity has led to the development of calorimeters and bolometers—devices that measure energy or power via a temperature change—that operate at wavelengths from near-IR to X-ray.

This thesis documents the design and validation of a hardware signal processor, implemented in a Field Programmable Gate Array (FPGA), for Tungsten based TES calorimeters with energy resolution $\sim 0.2\,\text{eV}$. When detecting monochromatic light in the near-IR and visible part of the spectrum Tungsten TES calorimeters act as photon number resolving detectors with unrivalled detection efficiency.

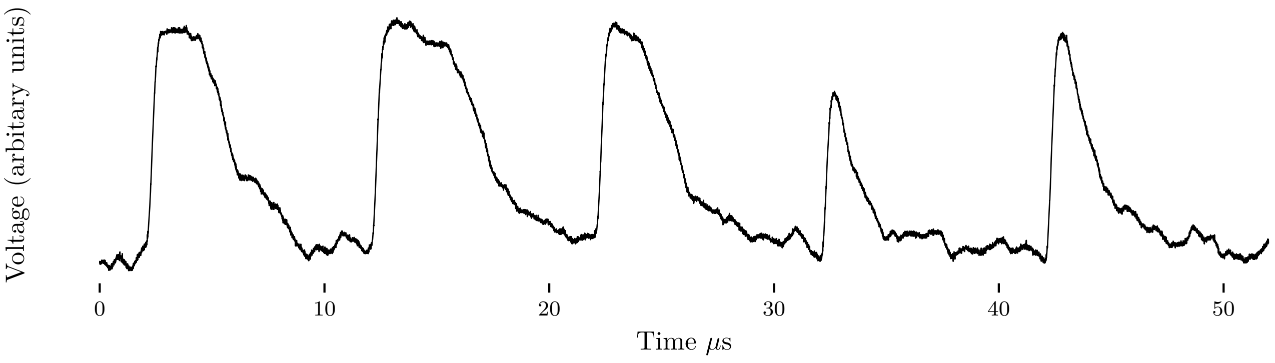


Figure 1: Raw analogue signal from a transition edge sensor (TES) after room temperature amplification, each pulse represents a detection. The optical input is from a $\sim$830 nm laser diode driven with a 100 kHz electrical pulse which produces optical pulses with average photon number $\langle n \rangle \sim 2$.

The processor takes the analogue output pulses arising from calorimeter detections and produces a time-stamped stream of *event* packets containing pulse measurements.

The hardware design is motivated by two needs; first and foremost, to produce a real time stream of detection time and photon number estimates for use in quantum optics and information experiments; and secondly to provide a testbed for exploring new strategies and algorithms for estimating the energy from a detection pulse.

Detection energy is encoded in the shape of the output pulse which changes in different energy regimes. When detecting low energies the pulse shape linearly scales with energy and as the detection energy increases this scaling becomes non-linear until the sensor eventually saturates. The pulses convey energy information in all the energy regimes but with diminishing resolution. The processor can operate in all regimes by extracting pulses and returning a trace in the event packet.

Publications during candidature

Peer-reviewed papers

Experimental Distribution of Entanglement with Separable Carriers,
A. Fedrizzi, M. Zuppardo, **G. G. Gillett**, M. A. Broome, M. de Almeida, M. Paternostro, A. G. White, and T. Paterek,
Physical Review Letters 111, 230504 (2013).

Conclusive quantum steering with superconducting transition edge sensors,
D. H. Smith, **G. G. Gillett**, M. P. de Almeida, C. Branciard, A. Fedrizzi, T. J. Weinhold, A. Lita, B. Calkins, T. Gerrits, H. M. Wiseman, S. W. Nam, and A. G. White,
Nature Communications 3, 625 (2012).

Experimental feedback control of quantum systems using weak measurements,
G. G. Gillett, R. B. Dalton , B. P. Lanyon, M. P. Almeida, M. Barbieri, G. J. Pryde, J. L. O'Brien, K. J. Resch, S. D. Bartlett and A. G. White,
Physical Review Letters 104, 080503 (2010).

Towards Quantum Chemistry on a Quantum Computer,
B. P. Lanyon, J. D. Whitfield, **G. G. Gillett**, M. E. Goggin, M. P. Almeida, I. Kassal, J. D. Biamonte, M. Mohseni, B. J. Powell, M. Barbieri, A. Aspuru-Guzik and A. G. White,
Nature Chemistry 2, 106 (2010).

Publications included in this thesis

No publications.

Contributions by others to the thesis

No contribution by others.

Statement of parts of the thesis submitted to qualify for the award of another degree

None.

Research Involving Human or Animal Subjects

No animal or human participants were involved in this research.

Acknowledgements

A personal thank you goes to my advisors; Andrew White, your open mind saw past my many faults to my strengths and your open door made PhD candidature a stimulating, rewarding and enjoyable experience; Alessandro Fedrizzi, your advice was always well considered and solid, thanks for continuing to offer it even after the occasions I failed to follow it; Marcelo Pereira de Almeida, thanks for your patience, politeness and tireless effort to help me solve problems in the lab. I'd also like to thank fellow student Alex Nicolova for her efforts in developing and implementing the host computer servers and software.

The are many, many more people I encountered during my PhD candidature who deserve a personal thank you, but I'll just say this, thanks to all; academic and administrative staff; students and postdoc's I shared rooms with or collaborated on experiments with; and workshop staff; for having the patience and taking the time to attempt to teach an old dog new tricks.

Financial support

This research was supported by an Australian Postgraduate Award (APA) scholarship.

Keywords

transition edge sensor, number resolving photon detector, field programmable gate array, signal processing, quantum optics, quantum information.

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC code:020603 Quantum Information, Computation and Communication, 40%
ANZSRC code:020604 Quantum Optics, 40%
ANZSRC code:090609 Signal Processing, 30%

Fields of Research (FoR) Classification

For code:0205 Optical Physics, 40%
For code:0206 Quantum Physics, 40%
For code:0906 Electrical and Electronic Engineering, 30%

To my family. Without your love, support and understanding this adventure into the world of professional science would not have been possible.

Contents

Thesis structure

This thesis is in two parts.

Part I covers the traditional thesis contents; introductory material; data; analysis; and conclusions.

Part II is a reference manual with technical details relating to; hardware implementation; processing pipelines; data formats; control register formats; and communication protocols.

Typography

Lower case typewriter font is used to indicate values and fields associated with the processor and are hyperlinks to the technical descriptions, some are colour coded, examples are:

- registers controlling processor behaviour: `pulse_threshold`.
- sequence derived from the digitised TES detection signal: `f`.
- event packets carrying measurements: `rise`.
- fields in event packets containing measurement information: `eflags`.
- enumerated types are strings representing a processor register settings: `f_extrema`.

VHDL generic constants that control the way the processor is implemented are typeset in upper case typewriter font, eg `CFD_DELAY`, and are not linked.

Part I

Overview

Chapter 1

Introduction

1.1 Transition edge sensors

Transition Edge Sensors (TESs) are sensitive thermometers that utilise the sharp change in resistance with respect to temperature during in the transition between the superconducting and normal metal states. The unprecedented sensitivity of TESs has been a boon in the implementation of low temperature thermal detectors, specifically bolometers and calorimeters.

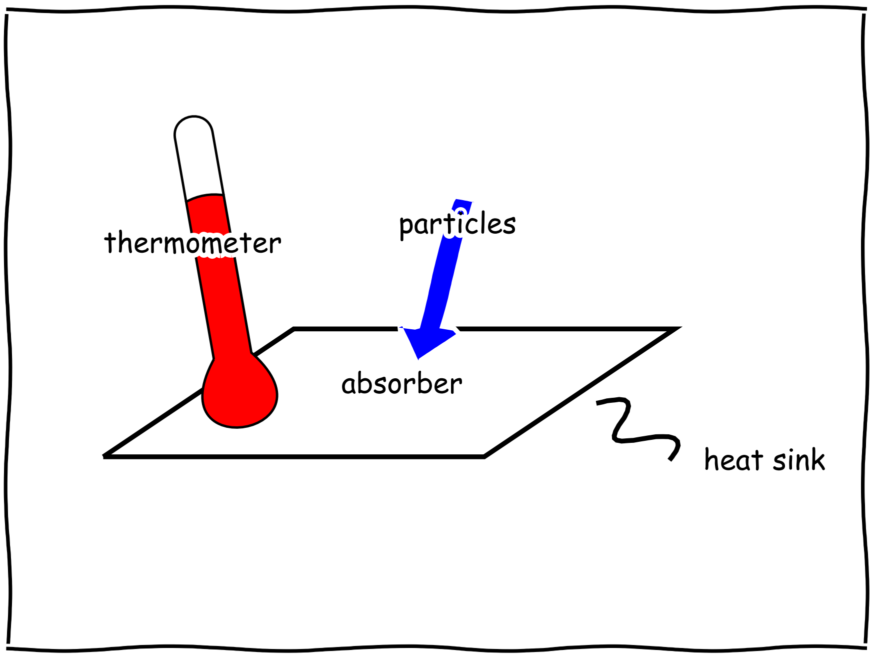


Figure 1.1: A calorimeter measures particle energy via a change in temperature. Calorimeters can be divided into 3 subsystems; an absorber for incoming particles; a thermometer to measure the absorbers internal energy change due to an absorption; and a weak link to a heat sink to cool the absorber after a detection while allowing the thermometer time to make the measurement.

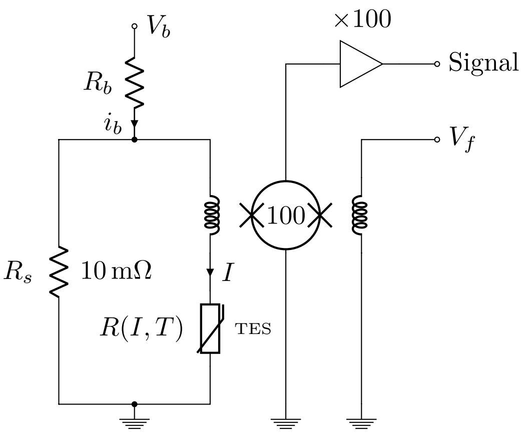
These types of detector consist of three components; an absorber of particles or incoming radiation; a thermometer, well coupled to the absorber to measure the change in internal energy from absorption events; and a weak coupling between the absorber and a cold bath to remove heat generated during detection and return it to a quiescent state (Figure 1.1). When such a detector measures power it is generally called a bolometer and when measuring energy a calorimeter. Though these terms are often used interchangeably, I'll stick to the term calorimeter since our use case is measuring photon energy.

TES based calorimeters were pioneered in Blas Cabrera's group at Stanford in the 1990's. Development was mainly aimed at astronomical use and designed to detect higher energy particles with a particular interest in detectors for the Cold Dark Matter Survey. These detectors had a separate absorber with large enough heat capacity to handle the high energies and often employed quasi-particle traps to capture athermal phonons before they thermalised. A TES was used as the thermometer that reads the internal energy change of the absorber or trap due to a particle absorption. Tungsten transition edge sensors (W-TESs) were developed for this purpose and later refined by Sae Woo Nam and his group at National Institute of Standards and Technology (NIST) Boulder to work at near-IR and optical photon energies. Tungsten was a natural choice; its phonons and electrons decouple at low temperatures providing a weak cooling link; and its superconducting transition temperature can be tuned. Tuning the transition temperature simplifies cryogenic requirements while limiting thermal noise and providing the energy resolution required to resolve near-IR photons. Tungsten has two crystal phases–α and β–by careful control of the spluttering conditions when depositing the Tungsten film the relative proportion of the phases can be controlled altering the superconducting transition temperature [LRN$^+$05].

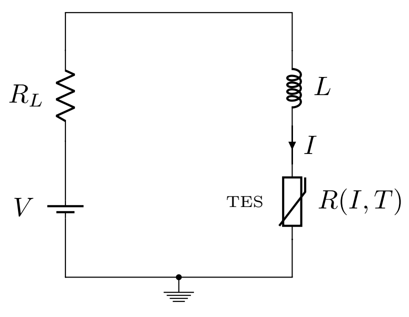
In the elegantly simple optical W-TES designed and fabricated by our collaborators at NIST [LCP$^+$10][LMN08] all three calorimeter roles are played out in a thin film of tungsten and the W-TES thermometer *is* the calorimeter. Tungsten's electron gas acts as both absorber and thermometer while it's anomalously weak low temperature electron-phonon coupling provides the cooling link to the crystal lattice acting as the cold bath. I'll drop the term TES based calorimeter and the acronym W-TES and simply use the acronym TESs or the word sensor to refer to these exquisite NIST optical calorimeters. In operation the package of fiber coupled[MLC$^+$11] TESs is placed in an adiabatic demagnetisation refrigerator (ADR) and and cooled to a bath temperature (T_b) $\sim$100 mK well below the sensors superconducting transition temperature (T_c) $\sim$150 mK. The sensor is then biased to an operating point in the transition region between the superconducting and normal metal states using the circuit in Figure 1.2(a).

1.1.1 Small signal model

In the simplest picture of a calorimeter the absorber is characterised by its heat capacity C and its weak link to the bath at temperature T_b by thermal conductivity G, instantaneous



(a) Bias and readout circuit



(b) Thevenin equivalent circuit

Figure 1.2: (a) Biasing and readout. The TES is in parallel with a small shunt resistance (R_s) forming a divider for the bias current ($i_b \sim V_b/R_b$ as $R_b \gg R_s$). The sensor current (I) is inductively coupled to an array of DC superconducting interference devices (SQUIDs) which transduce I to a voltage signal which is further amplified at room temperature. Voltage V_f sets the operating point on the SQUID voltage-flux response to maximise transduction gain while V_b biases the sensor at an operating point (I_0, T_0) in the phase transition between the superconducting and normal metal states (see Figure 1.3).
(b) Thevenin equivalent of the bias circuit used in TES modeling, $R_L = R_s R_b/(R_s + R_b)$, $V = V_b R_s/(R_s + R_b)$ and is L the inductance coupling to the SQUID.

absorption of energy E raises the absorbers temperature by $\Delta T = E/C$ which relaxes back to T_b with a natural thermal time constant $\tau = C/G$.

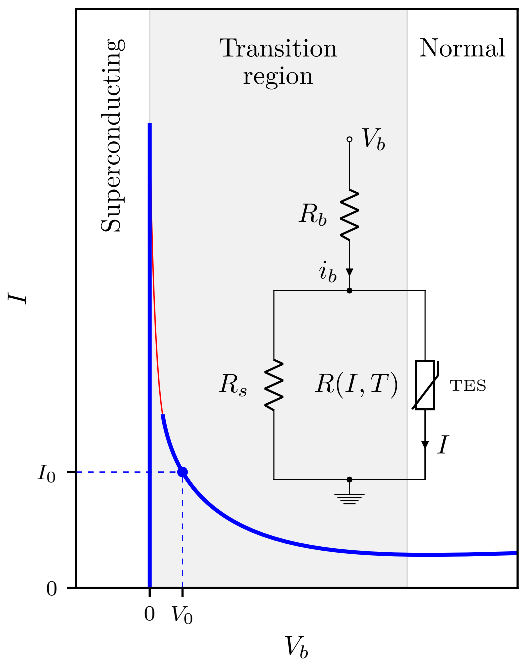
A complete picture sees the TES system as coupled electrical and thermal circuits which can be described by two differential equations in the two state variables, current (I) and temperature (T),

$$L\frac{dI}{dt} = V - I\left(R_L + R\left(I,T\right)\right) \tag{1.1}$$

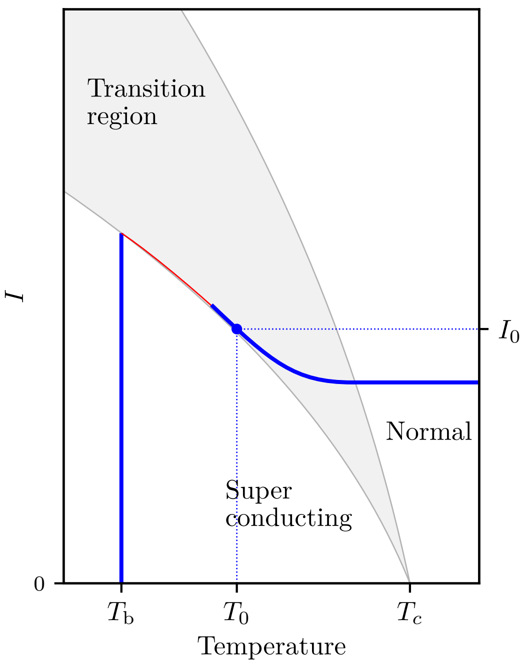
$$C\left(T\right)\frac{dT}{dt} = \underbrace{I^2 R\left(I,T\right)}_{\text{Joule heating}} - \underbrace{K\left(T^n - T_b^n\right)}_{\text{cooling power}} + \underbrace{E\delta\left(t - t'\right)}_{\text{signal power}} \tag{1.2}$$

. The electrical equation (Equation 1.1) describes the Thevenin equivalent of the calorimeter biasing circuit (Figure 1.2b) equating the voltage across the inductor L with the voltage across the rest of the circuit, $R(I,T)$ is the current and temperature dependant TES resistance. Similarly, Equation 1.2 describes the thermal circuit equating energy change to power in and out of the calorimeter, cooling to the bath is described by a power law with constant K and exponent n and the signal power term represents deposition of energy E at time t' under the assumption the event causes an instantaneous change in temperature of the absorber-thermometer system.

The cross terms in the equations describe a feedback interaction between the electrical and thermal circuits known as ETF [NCC+99][Irw95]. The sensor/thermometer is characterised by



(a) V_b-I curve



(b) Sensor state as V_b is changed

Figure 1.3: Sensor trajectory as bias voltage is changed: At 0 V bias the sensor is at the bath temperature (T_b) well below its superconducting transition temperature (T_c) and has 0 resistance. When the bias voltage (V_b) is increased above 0 V the sensor current (I) rapidly increases until it reaches the critical value and the sensor enters the transition between the superconducting and normal phases. In the transition the sensor state can be considered a mixture of the superconducting and normal phases with the proportion of normal phase and resistance increasing with temperature. Current flowing through the the non-zero resistance dissipates power increasing the sensors temperature. For any fixed bias voltage the bias current (i_b) is also fixed, variation in the sensors resistance varies the sensor current I by changing the division of i_b between the sensor and the shunt resistance R_s. As I changes so does the power dissipated which in turn changes the sensor temperature. This feedback between current and temperature is known as electro-thermal feedback (ETF). For a fixed v_b, ETF acts to stabilise the TES at some equilibrium point int the phase diagram. The blue line indicates the trajectory of this equilibrium point through the phase diagram as v_b is increased. The equilibrium temperature increases and the equilibrium current decreases as v_b increases until the sensor leaves the transition and enters the normal resistance region. The thin red segments in the figures are regions where the no stable equilibrium exists. The bias voltage is adjusted to find the equilibrium point (I_0, T_0) that achieves the highest gain for converting temperature change into a current change.

its local logarithmic sensitivity, defined as $\alpha \equiv \frac{d \ln R}{d \ln T}|_{V_q}$ by some authors and by a constant current derivative $\alpha_I \equiv \frac{\partial \ln R}{\partial \ln T}|_{I_q}$ and a constant temperature derivative $\beta_I \equiv \frac{\partial \ln R}{\partial \ln I}|_{T_q}$ by others.

These logarithmic sensitivity terms describe the shape of the transition. In the case of W-TES α is positive which when combined with voltage biasing leads to negative feedback between current and temperature. A rise in temperature increases TES resistance which reduces the current I flowing the TES arm of the circuit (Figure 1.2a) reducing the power dissipated and cooling the sensor. The reverse is true for falls in temperature. Negative ETF stabilises the TES in the transition region at a quiescent equlibruium point and reduces the relaxation time below that given by the natural time constant τ. Negative ETF acts much the same way as negative feedback acts in amplifiers to linearise and stabilise the output. The NIST W-TES operate with extreme negative ETF and the energy deposited by a photon is removed by the reduction in Joule heating while cooling power to the bath is essentially constant. In the extreme ETF regime detection energy is proportional to integral of the current pulse.

Small signal models are well developed and illuminating, they are constructed by linearising the coupled differential equations about the quiescent operating point (I_0, T_0) and expanding to first order [IH05][Lin00][McC05] following [IH05] which expands the analysis in [Lin00] and leaving the detail in the references, the linearised equations 1.1 and 1.2 can be put in a matrix form

$$\frac{d}{dt}\begin{pmatrix}\delta I\\ \delta T\end{pmatrix} = \begin{pmatrix}\frac{1}{\tau_{el}} & K_1\\ K_2 & \frac{1}{\tau_I}\end{pmatrix}\begin{pmatrix}\delta I\\ \delta T\end{pmatrix} + \begin{pmatrix}\frac{\delta V}{L}\\ \frac{\delta P}{C}\end{pmatrix}, \tag{1.3}$$

where $\delta I \equiv I - I_0$, $\delta T \equiv T - T_0$, $\delta V \equiv V - V_0$, $\delta P \equiv P - P_0$, P being the signal power and $P_0 = I_0^2 R(I_0, T_0)$. The two time constants τ_{el} and τ_I characterise the the decay of I to I_0 in the bias circuit and the decay of T to T_0 under a constant current. The two lumped constants K_1 and K_2 are dependant on the operating point and physical parameters (see [IH05] Equation 19).

Homogeneous solutions, $\delta V = \delta P = 0$, are of the form

$$\begin{pmatrix}\delta I\\ \delta T\end{pmatrix} = A_+\, e^{\lambda_+ t}\, \boldsymbol{v}_+ + A_-\, e^{\lambda_- t}\, \boldsymbol{v}_-, \tag{1.4}$$

where $\lambda_\pm$ and $\boldsymbol{v}_\pm$ are the eigenvalues and eigenvectors of the the matrix in Equation 1.3 while $A_\pm$ are unitless constants. The eigenvalues yield the rise and fall time constants for the current pulse $\tau_\pm = 1/\lambda_\pm$ and the eigenvectors give the directions for the rise and decay in the sensors phase diagram (see Figure 1.4a).

1.1.2 Large signals

As the detection energy increases the TES response is no longer described by the linear small signal model. The response becomes more and more non-linear as detection energy increases until the sensor is driven out of the transition into the normal regime and is saturated.

Although the superconducting transition is well understood theoretically the precise nature of TES resistance $R(I, T)$ in real devices outside the realm of the small signal model is not.

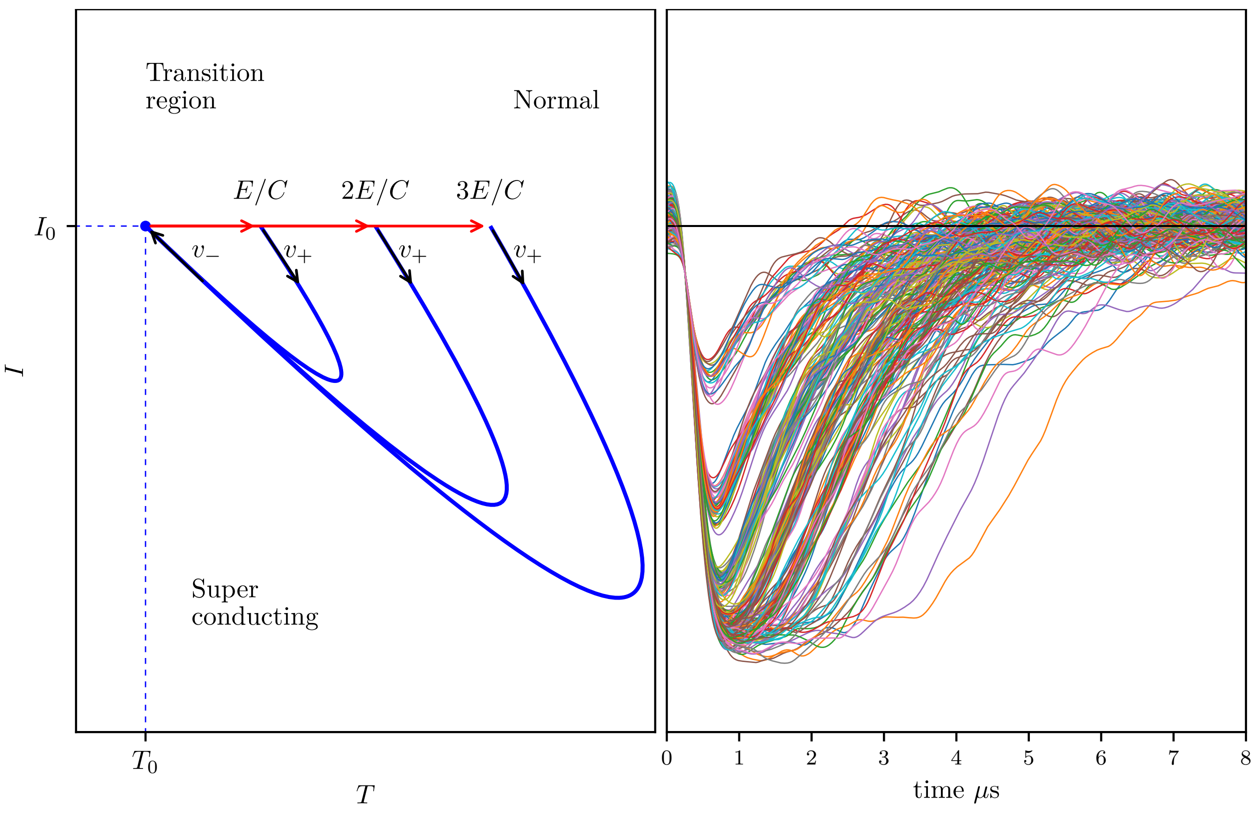


Figure 1.4: Left pane: A qualitative illustration of the sensor state trajectory during photon detection as described by the small signal model. The eigenvectors ($\boldsymbol{v}_{\pm}$) of the matrix in Equation 1.3 indicate the directions of the the rise ($\boldsymbol{v}_{+}$) and fall ($\boldsymbol{v}_{-}$) of the detection pulse while the eigenvalues are the inverse of the rise and fall time constants. The small signal model assumes that absorption of energy E instantaneously raises the sensor temperature by E/C where C is the sensors heat capacity.
Right pane: TES detection pulses after processing by hardware simulation. The optical input is a $\sim$830 nm 100 kHz pulse train with average photon number $\sim$2. The pulses are inverted to match the direction of the sensor current changes in the left pane. The collection of pulses are randomly coloured and illustrate the sensor's response in different energy regimes. For 830 nm photons these are roughly; (1-2 photons) the linear regime where the small signal model holds; (2-4 photons) the non-linear regime where scaling in of the TES response with photon number is non-linear; (5 + photons) saturation, the TES has been driven out of the transition, its resistance is no longer dependent on the current and all the enhancements to thermometry and cooling that the transition region provides no longer apply. While saturated the TES still acts like a normal cold metal calorimeter with energy information encoded in the time required to cool back into the transition. The integral of the pulse still encodes energy past saturation albeit with reduced resolution.

None-the-less several techniques have been explored for improving analysis of TES signals in the non-linear and early saturation regimes[BAB+15][FFCM+00] [LGM+14].

1.1.3 Detector metrics

As photon detectors TESs outperform competing technologies in two detector metrics:

Detection efficiency TESs have an near unity intrinsic detection efficiency for photons that reach the sensor and a system detection efficiency above 95%.

Dark counts TESs have zero intrinsic dark counts. All output pulses originate from a real detection, never from some spontaneous event in the detector or low temperature readout electronics. There are of course background counts from stray light and the high energy tail of the blackbody spectrum.

Number resolution TESs are one of the few technologies that provide *true* number resolution through the ability to resolve energy The phrase “true number resolution” distinguishes between the use of “number resolution” by technologies using multiplexed non-number resolving detectors to provide pseudo-number resolution.

. TESs are unique in combining number resolution, very high detection efficiency and low dark counts, but underperform in the remaining two detector metrics:

Timing jitter The uncertainty in the photon arrival time extracted from a TESs detection pulse is comparatively large. TES jitter can be improved by reducing inductance in the readout circuit (Figure 1.2a) which is dominated by the SQUID input coil. Reducing the inductance decreases the rise time constant (Equation 1.4) and increases the rising slope of the detection pulse. When timing is based on a threshold crossing the jitter is related to the TES noise–dominated by Johnson noise and thermal fluctuation across the link between absorber and bath–and the slope of the signal at the threshold. This improvement is demonstrated in [LLCT$^+$13] where the use of low input inductance SQUID amplifiers [DAB$^+$07] reduces the jitter to 4.1 ns at 1550 nm and 2.3 ns at 775 nm. The downside is the coax cabling required adds heat load to the cryostat and complicates wiring limiting the number of detectors that can be supported in an ADR. Limited availability of the low inductance SQUIDs has also limited deployment of these low jitter TES systems. For the sensors in our lab with relatively high inductance SQUIDs and twisted pair wiring the jitter is $\sim$ 80 ns full width half maximum.

Recovery time Also called dead-time. TESs have no intrinsic dead time and continuously detect but the long relaxation time leads to *pile-up* of detection pulses. Pile-up occurs when another photon is detected before the TES has fully recovered from the previous detection. Piled-up complicates the extraction of the detection energy and time from a pulse limits the usable TES detection rate. The recovery time can also be improved, [CLFN11] reports a four fold reduction in the fall time constant without significant loss of energy resolution. This is achieved by increasing the coupling between the TES and the cold bath using normal metal heatsinks.

I'll add an informal metric[1], ease of use. Compared to the current workhorse detector for quantum information experiments, the avalanche photo-diode, TESs are extremely difficult to use. To count photons with an avalanche photo-diode it simply needs to be powered up and connected to something that can count standard logic pulses. To count photons with a TES you must cool it to 100 mK then you must deal with its analogue output. Ease of use and accessibility to these remarkable sensors was a factor motivating this project.

Improvements to TESs will ultimately hit limits imposed by the underlying physics and engineering constraints for producing a stable device [IHWM98]. It would appear unlikely that TESs will ever become a workhorse detector for quantum information which mostly demands efficient and rapid counting of *single* photons. The leading candidate technology for this role is based on superconducting nanowires which offer; high detection efficiency, not yet at levels offered by a TES but still improving; very low jitter; low recovery time and counting rates in hundreds of millions of counts per second; architectures that can provide pseudo-number resolution; and have higher operating temperatures requiring less expensive and more convenient continuous cycle cryogenics. See chapters 1 and 2 of [Ens05] for a review of both technologies in the context of quantum optics and information. Nonetheless, TESs find use in in quantum information experiments that exploit the sensors unique strengths. The unrivaled detection efficiency of TESs has allowed violation of quantum steering and Bell inequalities without assuming fair sampling. Number resolution, though not generally required in quantum information, allows direct examination of the consequences of assuming a source is emitting single photons and opens up higher dimensional spaces to photonic quantum information experiments.

1.2 Field programmable gate arrays

Experimental experience with TESs gained during our quantum steering project[SGdA+12] highlighted the usefulness of realtime detection information, see section 2.1 for more detail. The methods used by others to extract time and energy information from the detection pulses involve digitisation of the TES output then processing with software. While software based approaches allow arbitrarily complex processing to achieve low uncertainty for the energy estimates, they are difficult to scale past a few TES channels while still providing realtime number resolved coincidence counts. I decided to explore *hardware* processing of the signal and the subject of this thesis is a library of digital circuits designed specifically for processing TES detection pulses. The circuit library is used to create a prototype hardware signal processor which focuses on producing a digital stream of event packets with minimal latency between a detection and the event packet being available. Each detection pulse produces an event packet containing pulse measurements and a timestamp. The downside of using hardware is that processing complexity is constrained to processes realisable as circuits.

[1]A metric of particular importance to PhD students.

The modern approach to prototyping digital hardware is to use field programmable gate arrays (FPGAs) to implement circuits described in a hardware definition language (HDL). HDL descriptions can be confused with a program written for a general purpose computer but any similarity is superficial. Though HDLs are general languages capable of universal computation only a subset of statements in the language are *physical* and can be realised as a circuit. A number HDLs exist and the two most widely used emerged in the 1980's. I choose to use VHDL a language created by the United States department of defence to address the problems encountered while re-procuring hardware based on obsolete technologies. VHDL was designed to document and describe all components of a system so that new circuits could be constructed that function identically to obsolete parts. Consequently, VHDL has wide descriptive capabilities making it quite verbose but offers higher levels of abstraction than the alternative HDL from the period, Verilog.

Since the 1980's other HDLs have emerged, most notably SystemC and System-Verilog, offering levels of abstraction approaching that of general computer languages which potentially makes complicated circuits easier to verify, modify, maintain and reuse. Unfortunately tools were not available for these languages at the beginning of this project. Recent developments include tools that convert algorithms written in C into HDL circuit descriptions which may lower the entry barrier for those wanting to add to the library.

Back in the day, digital design and prototyping involved discrete components and a "bread-board" which allowed the connections between components to be adjusted to easily change the circuit. FPGAs are the evolution of this, they contain large array of components or *resources* embedded in a reconfigurable routing matrix allowing the connections between resources to be changed to create different circuits. FPGA resources can be divided into two categories; *fabric* resources are ubiquitous and arranged in cells containing fundamental low level digital components, the fabric cell array covers the majority of the FPGA chip die; *hard cores* are dedicated silicon performing a specific function, effectively internal chips. Fabric resources are general and can be used to construct *any* circuit but the many connections between the low level components required to implement a complex circuit in fabric limit the rate at which it can be clocked. Hard cores provide dedicated functionality for example, random access memory (RAM), transceivers or multiply and accumulate block for use in digital signal processing (DSP) and run at speeds comparable to discrete chips.

Digital circuits are built from two types of logic; *combinatorial* logic elements have outputs that change whenever any input changes, for example a logic gate; *sequential* logic elements have outputs that change synchronously with a *clock* edge, the canonical example being a register or flip-flop. In order to function reliably and deterministically the inputs to sequential elements must not change in a window around clock edge defined by the *setup* and *hold* time, see Figure 1.5. When all sequential elements in a circuit satisfy the setup and hold requirements the circuit is said to *meet* timing.

The HDL circuit description is processed by a toolchain which proceeds in three phases.

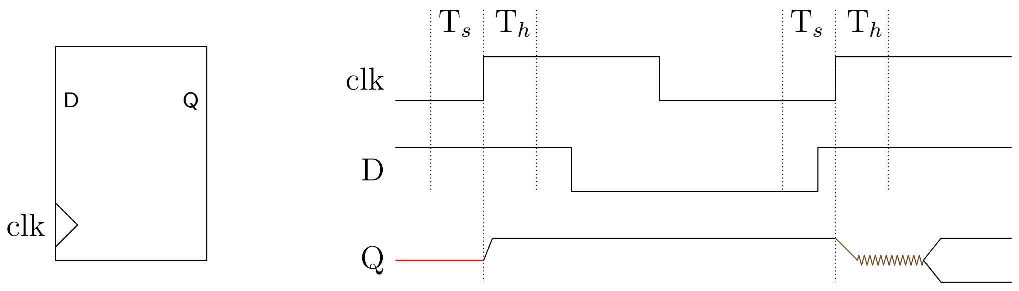


Figure 1.5: A register, also know as a D-type flip-flop, is a sequential logic element. The output (Q) is the value of the input (D) sampled at a rising transition of the clock signal (clk). To function correctly the input must be stable during the aperture that extends from the setup time (T_s) before the clock transition the the hold time (T_h) after it. If this timing constraint is violated the register can enter a metastable state which eventually decays to either Q = H or Q = L in a non-deterministic manner.

synthesis infers generic circuit elements from the HDL statements and creates a *netlist* describing the circuit. *Mapping* takes the netlist and replaces the the generic elements with the specific resources available in the particular FPGA. *Place and route* takes the mapped netlist and places each mapped element in the FPGAs resource array in such a way that the FPGAs routing matrix can realise all netlist connections. This placement is then optimised over a number a passes with the goal of ensuring the design meets its timing constraints. The placed and routed design is then converted to a bitstream that can be uploaded to the FPGA to realise the design.

I used a Xilinx 6 series FPGA (Virtex6) to implement the processor which uses a toolchain called ISE which was created at the time Xilinx was producing 3 series devices. Since I began the project Xilinx released 7 series devices and a new toolchain called Vivado. Unfortunately Vivado is not backwardly compatible and only works with 7 series devices and above. Xilinx make excellent FPGA hardware but the software leaves a lot to be desired, particularly ISE. Vivado appears to be a vastly better tool-chain. The processor is stuck at the compatibility break and needs to be ported from ISE to Vivado. ISE can only meet timing for the design when two processing channels are instantiated. Four are possible when specific placement constraints are added for the place and route optimisation but any design changes require new constraints to be found. Vivado appears to have better optimisation routines and the Virtex7 has faster routing and more resources. When the design is ported to Vivado I expect 8 channels will meet timing routinely and without extra constraints.

Chapter 2

Processor design

2.1 Motivation

My first experience using TESs in a quantum information experiment was working on a project that violated a *quantum steering* inequality while closing the detection loophole [SGdA$^+$12]. Steering is the ability to remotely prepare part of a multipartite state into different ensembles of states by performing different measurements on another part. Steering requires and serves to certify *entanglement* which is a strong *non-classical correlation* between subsystems of a multipartite quantum state.

2.1.1 The GHZ game

Non-classical correlation is perhaps best described by considering a cooperative three player game called the GHZ game[Mer85][Fin][Bac]. During play the players – Alice, Bill and Clarisse – are isolated in separate laboratories and communication between labs is *physically impossible*. Each lab contains a blank display that will show a symbol–either X or Y–during play as well as two mutually exclusive buttons labeled $+1$ and -1 that each player uses to respond to the symbol. Before being isolated the players are told the rules of the game and are free to communicate and share information and resources in an attempt to devise a foolproof strategy. Here I use the term foolproof to describe strategies that *always* win. The two rules are; when a symbol is displayed each player must respond by pressing a button; and either the symbol X will be displayed in *all* labs, or X will be displayed in exactly one lab while Y is displayed in the others. The players win the game if the product of their responses is $+1$ when all lab displays show X and -1 otherwise.

The possible games and the winning response products can be represented as a set of four equations

$$\begin{aligned} r_X^A r_X^B r_X^C &= +1 \\ r_X^A r_Y^B r_Y^C &= -1 \\ r_Y^A r_X^B r_Y^C &= -1 \\ r_Y^A r_Y^B r_X^C &= -1. \end{aligned} \tag{2.1}$$

The LHS of (2.1) is the product of terms of the form r_j^i indicating the response of player $i \in \{A(\text{lice}), B(\text{ob}), C(\text{larise})\}$ to symbol $j \in \{X, Y\}$ while the RHS is the response product required to win. Multiplying the equations together reveals a contradiction,

$$\begin{aligned} \text{product of LHS of 2.1} &= (r_X^A)^2(r_Y^A)^2(r_X^B)^2(r_Y^B)^2(r_X^C)^2(r_Y^C)^2 = +1 \\ \text{product of RHS of 2.1} &= +1 \times -1 \times -1 \times -1 = -1. \end{aligned} \tag{2.2}$$

This contradiction proves that no foolproof deterministic strategy exists, ie there are no foolproof strategies where the players have predetermined their responses to a symbol.

A similar argument proves that no foolproof non-deterministic strategy exists. In non-deterministic strategies each player i responds to symbol j by pressing $+1$ with probability p_j^i and pressing -1 with probability $1 - p_j^i$. Under the non-deterministic strategy the r_j^i terms on the LHS of (2.2) become expectation values $\langle r_j^i \rangle$ with values in the interval $[-1, +1]$. A similar contradiction occurs because the product of the squared expectation values must be greater than or equal to 0.

In the most general case the players share some correlated resource before being isolated in the labs. The correlations shared through the the resource cannot be dependant on the symbols the players see during play as the symbols are unknown when they are able to share. Resources the players share can be described as conditions on extra shared variables which can have values Λ, letting $\rho(\lambda)$ be the probability that the variables have a particular value $\lambda \in \Lambda$ a description of *every* classical strategy allowed under the rules of the game can be constructed. Equations 2.1 become,

$$\begin{aligned} \int d\lambda\, \rho(\lambda)\, r_X^A(\lambda)\, r_X^B(\lambda)\, r_X^C(\lambda) &= +1 \\ \int d\lambda\, \rho(\lambda)\, r_X^A(\lambda)\, r_Y^B(\lambda)\, r_Y^C(\lambda) &= -1 \\ \int d\lambda\, \rho(\lambda)\, r_Y^A(\lambda)\, r_X^B(\lambda)\, r_Y^C(\lambda) &= -1 \\ \int d\lambda\, \rho(\lambda)\, r_Y^A(\lambda)\, r_Y^B(\lambda)\, r_X^C(\lambda) &= -1, \end{aligned}$$

where $r_j^i(\lambda)$ is the response of player i to symbol j when the shared variables have value λ.

Since $\rho(\lambda)$ is a probability a foolproof strategy can only exist if for some particular λ

$$\begin{aligned}
r_X^A(\lambda)\, r_X^B(\lambda)\, r_X^C(\lambda) &= +1 \\
r_X^A(\lambda)\, r_Y^B(\lambda)\, r_Y^C(\lambda) &= -1 \\
r_Y^A(\lambda)\, r_X^B(\lambda)\, r_Y^C(\lambda) &= -1 \\
r_Y^A(\lambda)\, r_Y^B(\lambda)\, r_X^C(\lambda) &= -1.
\end{aligned}$$

Which raises the same contradiction (2.2) proving that no strategy based on classical resources is foolproof.

When the players have access to quantum resources a foolproof strategy *is* possible. Consider the tripartite spin-$\frac{1}{2}$ entangled state $|\text{GHZ}\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$ where 0 (1) indicates spin up (down) in the Pauli z basis[1]. If each player takes a subsystem of a $|\text{GHZ}\rangle$ state into their lab then measures it the Pauli basis corresponding to the symbol displayed and responds with the outcome they can always win.

Consider the set of commuting observables

$$\begin{gathered}
\sigma_x^A \sigma_y^B \sigma_y^C \\
\sigma_y^A \sigma_x^B \sigma_y^C \\
\sigma_y^A \sigma_y^B \sigma_x^C ,
\end{gathered} \tag{2.3}$$

though the x and y components of the spin of an individual particle anti-commute $\left\{\sigma_x^i, \sigma_y^i\right\} = \sigma_x^i \sigma_y^i + \sigma_y^i \sigma_x^i = 0$, the commutator of any pair in the set introduces an even number of anti-commutations which cancel out. $|\text{GHZ}\rangle$ is a *simultaneous* eigenstate with eigenvalue -1 of all three and the observables correspond to the three possible games in which only one lab displays the symbol X. The product of the individual players measurement outcomes is equivalent to the eigenvalue -1 so they always win games where only one lab displays X. Similarly the observable

$$\sigma_x^A \sigma_x^B \sigma_x^C \tag{2.4}$$

commutes with the previous three (2.3) and $|\text{GHZ}\rangle$ is simultaneously an eigenstate of all four. The eigenvalue for $\sigma_x^A \sigma_x^B \sigma_x^C$ is $+1$ so when all labs display X the product of the players responses is $+1$ and they always win.

Why is it so? The players are allowed arbitrary sharing and communication to establish arbitrary correlations between themselves yet a foolproof strategy does not exist if the resources they share are classical but does if they are quantum. The only constraints are that the sharing must be done in ignorance of what symbol each player will see as that information does not yet exist and once that information does exist it cannot be shared. Always winning the GHZ game is impossible classically–which is the standpoint of our common sense and intuition–what is it about quantum mechanics that allows a foolproof strategy? Perhaps the answer lies in questions about when and where information exists.

[1]Most famously analysed by Greenberger, Horne and Zeilinger [GHSZ90]

2.1.2 Elements of reality

When using classical resources in the GHZ game the information governing player responses *exists* at the time they establish correlations between themselves then waits to be revealed once they enter the lab and see a symbol. When the the information dictating player responses exists *before* the symbol is known the contradiction in (2.2) is inevitable and it is impossible to always win the game.

Quantum mechanics introduces the notion that physical properties generally have no objective reality independent of the act of observation and the act of measuring *creates* what is measured. Uncertainty relations are understood not just as a prohibition on what is co-measurable but on what is simultaneously real. Underlying this is the generally unavoidable disturbance quantum measurement has on what is measured through quantum back-action. In general the more information is gained by measurement of a quantum system the greater the disturbance to it [BMG$^+$07][GDL$^+$10].

When the players use quantum resources the information on which they base their responses does not exist when they share resources prior to entering the lab, it is brought into existence when they see the symbol and perform the corresponding measurement. The four observables (2.3-2.4) all commute and $|\text{GHZ}\rangle$ is simultaneously an eigenstate of them all. Since the corresponding eigenvalue is the product of the of the three measurement outcomes on the individual particles, measurement on any two particles determines the the outcome of the third. The rules of game are designed to perfectly exploit this remarkable non-classical correlation in the $|\text{GHZ}\rangle$ state which assures the players always respond in a coordinated way and win. A quantum resource allows players win through what Einstein dubbed "spooky action at a distance".

This notion that the act of measurement creates what is measured troubled Einstein. Pais, who often accompanied Einstein on his lunchtime walk home from the Institute of Advanced Study in Princeton, recalls[Pai79]

> We often discussed his notions on objective reality. I recall that during one walk Einstein suddenly stopped, turned to me and asked whether I really believed that the moon exists only when I look at it. The rest of this walk was devoted to a discussion of what a physicist should mean by the term "to exist".

Einsteins position was first and most famously formalised in the 1935 Einstein, Podolsky and Rosen (EPR)[EPR35], which asks "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?". The paper contains two assertions

- Quantum theory is incomplete.
- Incompatible observables cannot be simultaneous elements of reality.

and the authors argue that only one assertion can hold.

The EPR paper examines the interpretation of quantum state vectors through a thought experiment on a quantum system consisting of two particles moving away from each other such that their total linear momentum is zero. In this arrangement the position and momentum of the particles are perfectly correlated. By measuring the momentum of one particle (p) the momentum of the other can be inferred from the correlation ($-p$), similarly for position. The two systems are measured when they are well separated and in such a way that no signaling between them is possible, ie information from measurement of one particle cannot influence measurement of the other.

EPR make two assumptions without directly addressing them; *locality* prohibits distant measurements of one particle disturbing what is considered "real" for the other system; and *separability* establishes each system has a separate reality in the form of an individual physical state.

In a nutshell incompleteness argument is; Spatially separate particles have individual real physical states (separability); if particles are spatially separated measuring or not measuring one particle cannot directly affect the reality of others (locality); if quantities on separate particles are strictly correlated those quantities have definite values (often called the EPR lemma). From the lemma, the system described in the paper has simultaneous definite values of both position and momentum and since these values cannot be inferred from quantum mechanical formalism the quantum mechanical description must be incomplete.

The EPR lemma is arrived at through the slippery notion of "elements of reality" which are defined in the paper by what is now known as the EPR criterion of reality:

> If, without in any way disturbing a system, we can predict with certainty (i.e., with probability equal to unity) the value of a physical quantity, then there exists an element of reality corresponding to that quantity.

The reality criterion asserts that a quantity has a value that exists irregardless of whether it measured or not when we can predict the value of that quantity with certainty.

The $|\text{GHZ}\rangle$ state, which had not been analysed at the time of the EPR paper, makes it clear that the EPR criterion of reality is in direct conflict with the predictions of quantum mechanics. Measurement of $|\text{GHZ}\rangle$ using the observables in (2.3) yields results

$$\begin{aligned} r_X^A r_Y^B r_Y^C &= -1 \\ r_Y^A r_X^B r_Y^C &= -1 \\ r_Y^A r_Y^B r_X^C &= -1. \end{aligned} \tag{2.5}$$

where r_j^i is the outcome of measuring σ_j for subsystem i and the RHS is the eigenvalue of $|\text{GHZ}\rangle$ for the observable. We can predict the value of the outcome of a measurement on any of the subsystems with certainty by measuring the other two. By the EPR reality criterion each r_j^i corresponds to an element of reality and has a value that exists before it is measured. The product of the equations in (2.5) is $r_X^A r_X^B r_X^C = -1$ since $r_j^i = \pm 1$. This conflicts with a direct measurement of $\sigma_x^A \sigma_x^B \sigma_x^C$ which has result $r_X^A r_X^B r_X^C = +1$.

Einstein further developed, focused and clarified his incompleteness argument in later publications in which the concept of "elements of reality" thankfully falls by the wayside. These later publications draw directly on the the notions of separability and locality implicitly assumed in the original EPR paper to make a case for incompleteness. The arguments between the Einstein and Bohr camps over the completeness of quantum mechanics is considered the greatest philosophical debate of early quantum theory. Bohr published a refutation of the original EPR paper under the same title in the same journal. A less obfuscated and more accessible view opposing the realist perspective is found in a 1954 letter from Pauli to Born[EM]

> As O. Stern said recently, one should no more rack one's brain about the problem of whether something one cannot know anything about exists all the same, than about the ancient question of how many angels are able to sit on the point of a needle. But it seems to me that Einstein's questions are ultimately always of this kind.

Surprisingly Pauli and Stern were wrong, brain racking by Bohm and Bell led to results that made the EPR claims experimentally testable.

2.1.3 EPR + Bohm (EPRB) and Bell's theorem

In the late 1950's Bohm[BA57][BA60] began considering ideas presented by EPR and developing thought experiments based on incompatible measurements of spin rather that position and momentum. Inspired by Bohm's thought experiments Bell pioneered the work that led to the theorem that now bears his name. Bell's Theorem is is a collection of results proving that local realistic theories like the one proposed by EPR, also known as local hidden variable theories, are incompatible with quantum mechanical predictions. Different local hidden variable theories give different meanings to "local realistic". In Bell's 1964 paper[Bel64] the realism consisted of postulating that in addition to quantum states there exist "complete states", or hidden variables, that determine measurement results. The paper derives an inequality that bounds the correlation that can be observed in an electron spin experiment given particular local realist assumptions. Bell's original work has been generalised and refined and now a plethora of inequalities that bound correlation for local realist theories are collectively referred to as Bell inequalities. The Bell inequality derived by Clauser, Horne, Shimony and Holt (CHSH)[CHSH69] was applicable to to photon polarisation and at the time opened the possibility of more tractable experimental realisations than provided electron spin as originally analysed by Bell.

Hidden variable models are probabilistic. Let $\rho(\lambda)$ be a probability distribution over the space of complete states Λ where $\lambda \in \Lambda$ represents a particular complete state. Two parties, Alice and Bob, are well separated and each makes measurements on one particle of and entangled pair. Each party can perform one of two measurements; Alice's measurement operators are $m^A \in \left\{A_\pm, A'_\pm\right\}$ with respective measurement outcomes denoted $a_\pm$ and $a'_\pm$ which are collectively labeled r^A; similarly, Bobs measurement operators are $m^B \in \left\{B_\pm, B'_\pm\right\}$ with outcomes $b_\pm$ and $b'_\pm$ which are collectively labeled r^B. All outcomes are ± 1. The following probabilities are assumed to be well behaved $(\forall \lambda \in \Lambda)$;

$$p\left(r^A | m^A, m^B, r^B, \lambda\right) \text{ and} \tag{2.6}$$

$$p\left(r^B | m^A, m^B, r^A, \lambda\right) \tag{2.7}$$

are the probabilities of one parties outcome conditioned on the other parties outcome when the complete state is λ; and

$$p\left(r^A, r^B | m^A, m^B, \lambda\right) \tag{2.8}$$

is the probability of the outcomes of joint measurements by both parties when the complete state is λ.

Locality results from the following assumptions; *remote outcome independence* assumes one parties outcome does not depend on the outcome of other, ie that

$$\begin{aligned} p\left(r^A | m^A, m^B, r^B, \lambda\right) &\equiv p\left(r^A | m^A, m^B, \lambda\right) \text{ and} \\ p\left(r^B | m^A, m^B, r^A, \lambda\right) &\equiv p\left(r^B | m^B, m^B, \lambda\right); \end{aligned} \tag{2.9}$$

while *remote context independence* assumes that one parties outcome does not depend on the measurement choice of the other

$$\begin{aligned} p\left(r^A | m^A, m^B, \lambda\right) &\equiv p\left(r^A | m^A, \lambda\right) \text{ and} \\ p\left(r^B | m^A, m^B, \lambda\right) &\equiv p\left(r^B | m^B, \lambda\right). \end{aligned} \tag{2.10}$$

The conjunction of these assumptions is equivalent to the factorisation condition[Jar84]

$$p\left(r^A, r^B | m^A, m^B, \lambda\right) \equiv p\left(r^A | m^B, \lambda\right) p\left(r^B | m^B, \lambda\right) \tag{2.11}$$

which is often referred to as *Bell locality.* For Einstein, locality restricts influences on the "real" physical states of spatially separated systems. For Bell, locality is focused instead on influences on the outcomes of joint measurements on separated systems.

Noting that the probability measurements m^A and m^B have outcomes r^A and r^B is

$$p\left(r^A, r^B | m^A, m^B\right) = \int d\lambda \, \rho\left(\lambda\right) p\left(r^A | m^A, \lambda\right) p\left(r^B | m^B, \lambda\right) \tag{2.12}$$

and defining the correlation between measurements as the expectation value of the product of their outcomes, we have

$$c\left(m^A, m^B\right) = \sum_{i,j} r_i^A r_j^B p\left(r_i^A, r_j^B | m^A, m^B\right), \tag{2.13}$$

where the sum is over all measurement outcomes. The CHSH inequality bounds a particular combination of of measurement correlations

$$S_{CHSH} = c(A,B) + c(A,B') + c(A',B) - c(A',B'). \tag{2.14}$$

S_{CHSH} can be bounded by fixing λ and examining the sum of products

$$S = \sum_{i,j} a_i p\left(a_i|A,\lambda\right)\left[b_j p\left(b_j|B,\lambda\right) + b'_j p\left(b'_j|B',\lambda\right)\right] + \\ a'_i p\left(a'_i|A',\lambda\right)\left[b_j p\left(b_j|B,\lambda\right) - b'_j p\left(b'_j|B',\lambda\right)\right]. \tag{2.15}$$

Since that absolute value of the terms of the form $a_i p\left(a_i|A,\lambda\right)$ are bounded by by 1, the absolute value of each line in the sum is bounded by 2. When value of one of the square brackets is 2 the value of the other is 0 so the absolute value of the sum is bounded by 2. Multiplying (2.15) by $\rho(\lambda)$ and integrating over λ gives S_{CHSH}, so $|S_{CHSH}| \leq 2$. Any Bell inequality bounds correlation that can be observed under the assumptions used to construct it. In the case of the CHSH inequality the assumption is Bell locality (2.11) which is a consequence of assuming remote outcome independence (2.9) and remote context independence (2.10).

The CHSH correlation between the measurement operators $A = \frac{1}{2}(I \pm \sigma_z)$, $A' = \frac{1}{2}(I \pm \sigma_x)$, $B = \frac{1}{2}(I \pm \frac{1}{\sqrt{2}}(\sigma_x + \sigma_z))$ and $B' = \frac{1}{2}(I \pm \frac{1}{\sqrt{2}}(\sigma_x - \sigma_z))$ for the entangled state $|\Psi_-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$ is $2\sqrt{2}$, which is the maximum violation of the CHSH inequality predicted by quantum mechanics.

Any Bell test must meet a number of experimental requirements which are referred to as loopholes. Failure to address these loopholes permits local realist explanation of the observed measurement correlation. The three most significant loopholes are; *sampling* loopholes arise when there there are inefficiencies in measurement due to loss between source and detectors, particularly, inefficient detection or inefficiencies establishing the coincidence between detections used to calculate correlations; the *locality* loophole arises when there is the possibility of information from measurement on one system influencing measurement on the other; and the *freedom of choice* or *measurement independence* loophole concerns the independence of measurement choice and the internal state of the physical system being measured. Closing the locality loophole is reasonably straight forward by assuring appropriate separation between the detectors but large separations can can increase loss making simultaneously closing sampling loopholes difficult.

Heralding efficiency, the probability of detecting one particle in a pair conditioned on detecting the other, is key to closing sampling loopholes without assuming that the loss is fairly sampled. Fair sampling assumes that loss due to measurement inefficiency is *not* biased in a way that enhances measurement correlation in the cases where detection is successful. By assuming the opposite is true, that loss maximally enhances correlation, a new bound for a Bell inequality can be derived[Lar98]. For the CHSH inequality the bound becomes $\frac{4}{\eta} - 2$ where η is the heralding efficiency. A violation of the CHSH inequality by quantum mechanics requires a heralding efficiency $> 2(\sqrt{2} - 1)$. All early Bell tests using photons were performed under the fair sampling assumption. The unrivaled TES efficiency and lack of dark counts played a key role in recent Bell tests using spontaneous parametric down-conversion (SPDC) as the source of photon pairs [SMSC$^+$15][GVW$^+$15] that simultaneously close the locality and measurement independence loopholes without assuming fair sampling.

2.1.4 Quantum correlations

Mathematically, entanglement is associated with non-separable quantum states, ie states that cannot be written as a tensor product of subsystems. Non-separable quantum states can be arranged in a hierarchy by correlation strength; Bell non-local states that are capable of a Bell violation; *steerable* states are a super-set that includes the Bell non-local states and can violate a quantum steering inequality. The degree of non-classical correlation increases from non-separable to steerable to Bell non-local states.

Steering is the ability to remotely prepare one subsystem in different ensembles of states by performing different measurements on another subsystem. This concept was introduced by Schrödinger to generalise the EPR paradox for pure bipartite quantum states and is often referred to as EPR steering. Steering was put into a quantum information framework[WJD07] and inequalities developed that limit the level of correlation that can be observed without invoking quantum steering as an explanation, in a similar way that a Bell inequality limits the amount of correlation that can be explained by local realist theories. In quantum information terms, two parties that each receive a subsystem of a bipartite state can violate a steering inequality to certify they share entanglement.

Nonclassical correlation is a considered a fundamental resource powering the advantages quantum information processing provides over its classical counterpart. During the early development of of quantum information theory non-classical correlation was thought to be restricted to non-separable quantum states, ie that zero entanglement implied zero non-classical correlation. Later the concept of discord[OZ01] was introduced. Discord is the difference in two classically equivalent expressions for the mutual information between systems. Some separable mixed quantum states have non-zero discord and these states have no classical analogue. Zero discord and zero entanglement became the boundary for classical/non-classical correlation. Recent work[FP12] has compared nonclassicality criteria developed through quantum information theory and those derived from physical constraints on quantum phase space and quasi-probability distributions. It is found that the two approaches yield maximally nonequivalent nonclassicality criteria, implying that non-classical correlations exist in the absence of both entanglement and discord.

We will probe the classical/nonclassical boundary using the number resolving capabilities of TESs and my hardware processor. The entanglement and discord free nonclassical correlation can be be found in higher dimensional spaces and a resource for these correlations can be generated using SPDC. In SPDC a nonlinear crystal is *pumped* by a laser and the crystal spontaneously down-converts a single pump photon into a pair of photons with half the energy. SPDC outputs a two mode squeezed state[KLM01]

$$|\Psi\rangle = \sqrt{1-|\lambda|^2}\sum_{n=0}^{\infty}\lambda^n |n,n\rangle,$$

where λ is the squeezing parameter and $|\lambda|^2$ is proportional to the pump power. The probability of producing the Fock state $|n,n\rangle$ consisting of n photon pairs is $p(n) = (1-|\lambda|^2)|\lambda|^{2n}$ and increases with pump power. Quantum information experiments normally use SPDC to approximate a *heralded* source of single photons where detection of a photon in one mode heralds the occupation of the other. This approximation improves with decreasing pump power as the probability of the higher order terms diminishes. These higher order terms are generally avoided as they contribute to noise in a quantum circuit[WGR$^+$08]. In our search for nonclassical correlation in the absence of discord and entanglement we will pump the SPDC with as much power as possible as the nonclassical resource depends on terms of all orders.

After phase randomisation to remove any entanglement or discord the resource state is:

$$\rho_{AB} = \sum_n p(n)\,|n_A\rangle\langle n|\oplus|n_B\rangle\langle n|,$$

where $p(n)$ is the probability of producing n pairs and n_A (n_B) are the number of photons in mode A (B). The projects theorists have devised a game, similar to the GHZ game, where players with access to ρ_{AB} have an advantage over players that have only classical resources at their disposal. Unlike the GHZ state the correlations in ρ_{AB} are not perfect so no foolproof strategy exists but the quantum strategy has a higher probability of winning compared to any classical one. The number resolving capability of TESs is required for an experimental demonstration as the nonclassical correlation exists between the results of each players measurement of the number of photons they have.

2.2 Design goals

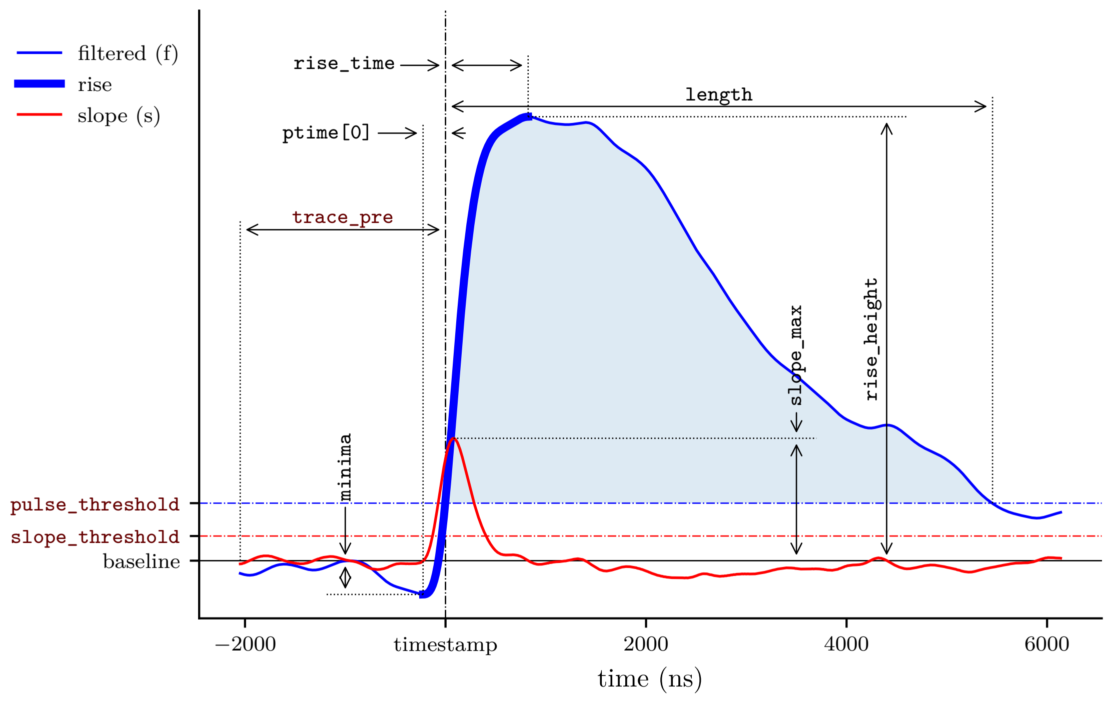
My practical introduction to TES was while working on a project that violated a steering inequality[SGdA$^+$12]. The High system detection efficiency of TESs enabled an unprecedented heralding efficiency of $\sim 62\%$ and violated the inequality by 48 standard deviations. A record that was only broken in recent Bell tests[SMSC$^+$15][GVW$^+$15]. To produce a countable signal from the TES output I used a clunky arrangement of equipment we had on hand. Consisting of an analogue constant fraction discriminator (CFD) and nuclear instrumentation module logic both pieces of equipment were designed for nuclear experiments. A long dead-time had to be imposed to ensure the CFD did not trigger on noise during the falling part of the TES pulse which reduced the conditional detection efficiency we could achieve. Despite this, the setup had the advantage of providing near real-time information useful while setting up and tuning the circuit. The standard technique for analysing TES output involves digitising entire pulses then processing them using software to extract timing and energy information. While the software approach can provide near realtime information for a few channels it is difficult to scale beyond that. The great advantage of processing the entire pulse in software is that it allows arbitrarily complex processing which can provide energy uncertainty close to the limit set by the sensors intrinsic energy resolution.

Experience gained during the steering project informed and motivated my design and the primary goal was to provide realtime detection information in a scalable way while minimising impact on the underlying sensor efficiency. My approach to achieving this goal was to process the signal in hardware rather than software and trade some energy certainty for realtime access to information from larger numbers of sensor channels all operating at the highest detection rates TESs can support. TESs have a wide dynamic range[LGM+14] with increasing energy uncertainty as the number of photons detected increases. The processor is designed for quantum information experiments which generally use “single” photon sources and to operate in the region were detections involve less than twenty 820 nm photons. The design process resulted in a library of circuit components that create and transmit a stream of event packets each containing measurements of a detection pulse and a timestamp. The processor as described in this thesis is a particular arrangement of of these library components used for evaluation and testing.

A secondary goal was to provide a platform for exploring techniques for extracting information from the TES signal. At one extreme are techniques that use a vector record of the entire pulse and at the other scalar measurements such as pulse height and area that the current processor implementation returns. Software processing of full pulses delivers the best energy resolution which translates into photon number certainty for a particular photon wavelength. Most of the single photon sources we use in the lab have a wavelength no greater than 820 nm and each photon carries relatively high energy. At these shorter wave lengths degraded energy resolution has less effect on number uncertainty due to the higher energy per photon. Preliminary analysis (subsection 3.2.3) of hardware processing indicates the simple pulse area measurement provides sufficient number certainty for the experiments we have planned. The processor can also optionally capture entire pulse records in conjunction with the scalar measurements and this information can be used to improve how the scalar measurements are implemented. The trace information can also be used to develop other hardware implementable measurements returning shorter vectors lying somewhere between the two extremes of scalar and full pulse measurement.

2.3 System overview

The analogue TES signal is digitised by an analogue to digital converter (ADC) which outputs a *sequence* of 14 bit integers called samples. The ADC sequence is labeled $\texttt{raw}_{in}$ and each sample



byte	0 1	2 3	4 5	6 7	
24	size	tflags	eflags	time	pulse header
32	area		length	trace_pre	
48	height	rise_time	minimum	ptime	rises
		⋮			
	height	rise_time	minimum	ptime	
	sample[0]	sample[1]	sample[2]	sample[3]	sequence record
		⋮			
	sample[n-3]	sample[n-2]	sample[n-1]	sample[n]	

Figure 2.1: Measurement is based on detecting rises in the filtered TES signal (f). Zero crossings of the slope (s) identify the local minimum and maximum of f. Rises are sub-sequences of f extending from a minimum to the following maximum and are considered valid when the height exceeds pulse_threshold and s has crossed slope_threshold during the rise. A pulse is a sub-sequence of f from the start of a valid rise to the next falling crossing of pulse_threshold by f. Measurements are performed on each pulse and returned in an event packet along with a timestamp and optionally a sequence record. The shaded region is returned as the area measurement. The height register controls what appears in the height field, for example it can record slope_max instead of rise_height. More details of the measurement process can be found in chapter 6 and descriptions of the registers controlling measurement in subsection A.1.8

estimates the average voltage at the ADC input over a $\sim 4\,\text{ns}$ time window and 250 million samples are produced per second. The processor operates on $\texttt{raw}_{in}$ and derives other sequences from it in order to extract detections and energy measurements. See chapter 6 for details.

The two most important derived sequences are the outputs of a reconfigurable two stage finite impulse response (FIR) filter (see section 6.3). The first stage is configured as a low pass filter to remove high frequency noise and produces the filtered sequence–labeled `f`–which is the sequence that is measured. The second stage is configured as a differentiator and produces the slope sequence–labeled `s`–which is used to identify *rises* in `f` and establish which rises constitute a *detection*.

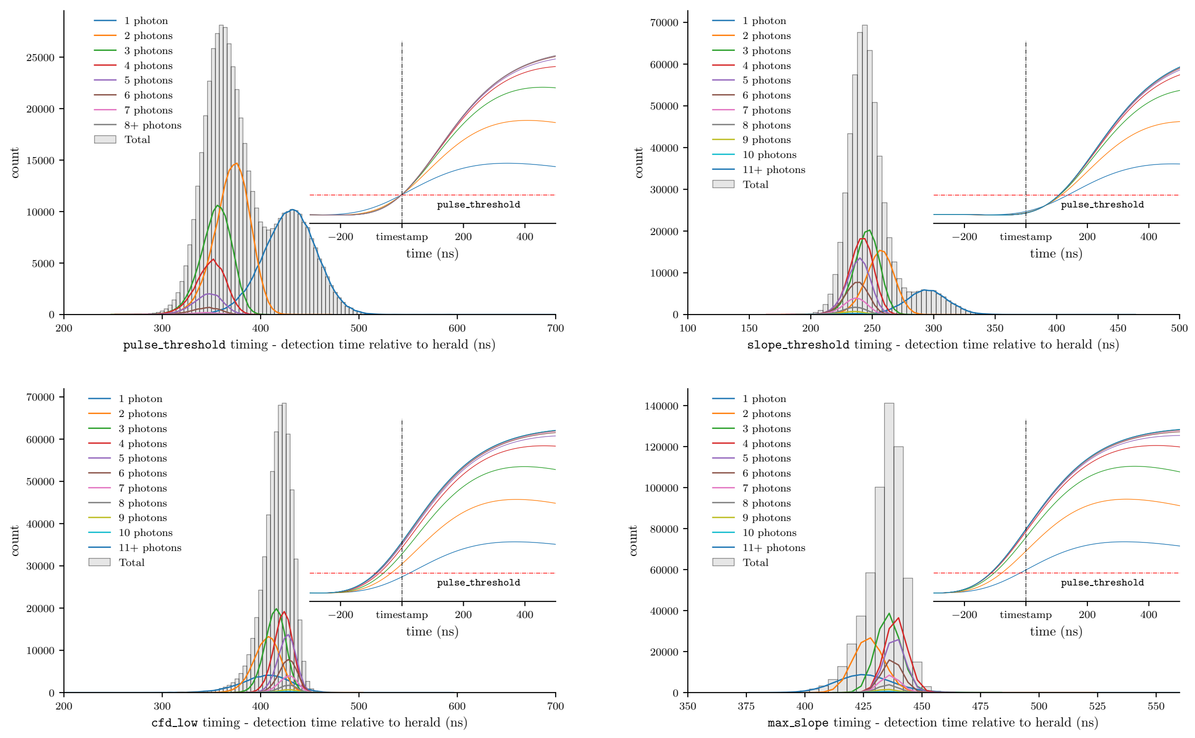


Figure 2.2: The `timing` register controls the point in a rise that is timestamped. The individual panels show histograms of the difference in timestamps from two processor channels. One channel processes the output from a TES while the other channel processes the same electrical pulse that drives the laser diode and heralds possible detections. Grey histograms show the distributions of the relative times between detections and the herald irregardless of assigned photon number while the coloured histograms show the distribution of times for detections assigned the respective photon number. The bin width of the histograms is 4 ns which is temporal resolution of the processor. The insets show the alignment of traces under the different `timing` settings, each coloured curve in the inset is the average over all captured traces assigned the respective photon number. The photon number classification process described in subsection 3.2.3. Note that these timing diagrams are not necessarily taken using the same optical inputs or the same TES. Conditions differ between diagrams and the datasets in subsection 3.2.1 and are only a guide to the different forms the number dependant timing jitter takes with different `timing` register settings.

Zero crossings of s are used to identify rises. Rises are sub-sequences of f that extend from a local minimum, identified by a rising zero crossing of s, to the next local maximum at the next falling zero crossing of s. A rise is *valid* and considered a detection when its maximum is above pulse_threshold *and* s has crossed slope_threshold during the rise. A *pulse* starts at the beginning of a valid rise and extends to the next falling crossing of pulse_threshold by f (see Figure 2.1). For each pulse the processor returns an event packet containing measurements of the pulse, a timestamp and optionally a sequence record. The sequence record, called a trace, starts trace_pre samples before the timestamp and records every trace_stride + 1 sample from the start until trace_length samples are recorded. The point at which the timestamp is generated is controlled by the timing register, see Figure 2.2. Section 5.4 describes of the contents of the different types of event packets, chapter 6 provides details of the measurement processes and Appendix A describes the registers that control the processor.

2.3.1 The multi-channel analyser (MCA)

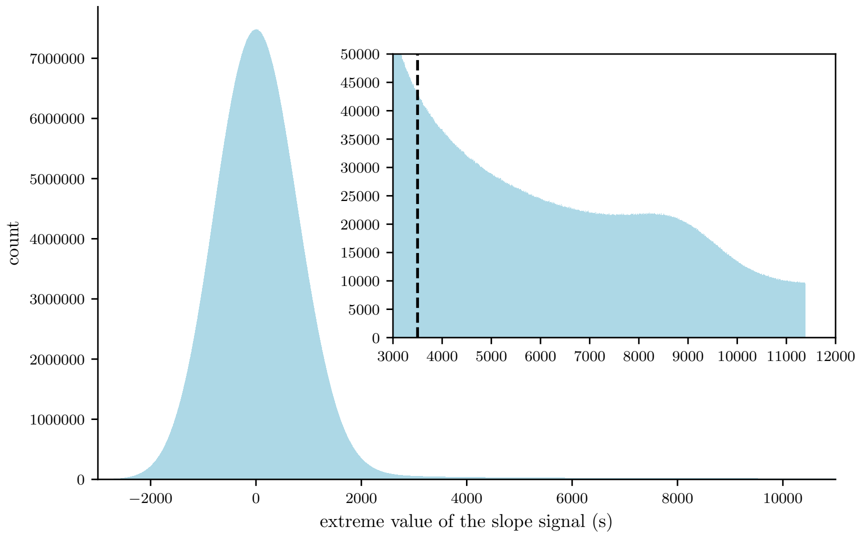


Figure 2.3: Distribution of the filtered TES sequence, f, captured by the multi-channel analyser (MCA). The MCA is capable of accumulating samples at the 250 million samples a second rate produced by the ADCs, so the histogram counts the frequency of every integer in the sequence f. The main body of the distribution represents the noise produced by the TES and associated electronics, the long tail on the right is due to the optical signal. The inset shows a zoomed view on the optical signal which continues past the abrupt termination at the last MCA bin. The vertical dashed line indicates the pulse_threshold setting.

The processor includes a MCA, which is a rather anachronistic term for a device that gathers the distribution of a measurement as a histogram. My MCA design is capable of counting every sample of a sequence while continuously transmitting the distribution of the sequence to the host computer (see Figure 2.3). It is of particular use in determining threshold register settings and descriptions of the registers controlling the MCA can be found in subsection A.1.3.

Measurements and settings are relative to a *baseline* assigned the value 0. A design assumption is that the baseline should be at the *mode* of the distribution of `f` (Figure 2.3). AC coupling to the TES output preamplifier (Figure 1.2) sets the 0 volt value at the *mean* of the `f` distribution. The position of the mean relative to the mode is dependant on the power of the optical input. This is due to the unipolar nature of the pulses which make the `f` distribution asymmetric with a long tail on one side. A simplified version of the central MCA is incorporated into each channel to track the mode of the `f` distribution and adjust the baseline accordingly (see section 6.2). This technique was not implemented to correct the baseline on short time scales but to correct for changes in output power seen when collecting experimental data. For example, when performing tomographic set of measurements the power to the sensor can vary widely with measurement setting. I discuss baseline correction further in section 4.1.

Chapter 3

Preliminary testing and analysis

3.1 Test apparatus

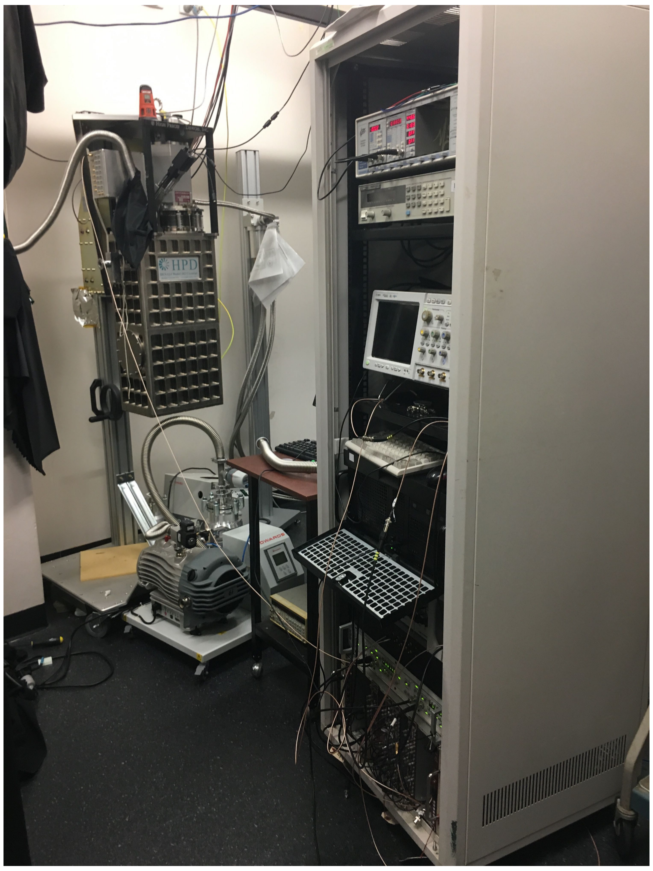


Figure 3.1: The cryostat and control electronics rack: The ADR stage and TESs are housed in the silver box labeled HPD. The rack of control electronics is in the foreground and the FPGA development board and ADC card used to implement the processor are installed in a case low in the rack. The silver pipe exiting the top of the HPD box carries fibres connected to TESs to the light tight fibre junction box out of frame on the left. In the junction box fibres from the TESs can be sliced to fibres connected to experiments.

We use an ADR to get our TESs down to operating temperature. An ADR is a final low temperature cooling *stage* usually housed in a continuous closed cycle refrigerator. Our continuous cycle refrigerator is a pulse tube refrigerator (PTR) [dW00] with a cold plate at $\sim 3\,\mathrm{K}$ and the ADR stage cools from 3 K down to the TES operating temperature. ADRs utilise the magnetocaloric effect by controlling the entropy of the magnetic moments of a paramagnetic salt housed in *pill*. Energy in the magnetic degrees of freedom of the pill is dependant on the angle of the magnetic moments relative to the direction of an external magnetic field and that fields strength. The minimum energy configuration occurs when all the magnetic moments are aligned with the field. The pill is suspended inside the PTR below a superconducting magnet which provides the field and the pill can be either thermally isolated or connected to the 3 K plate by opening or closing a mechanical *heatswitch*.

In order to cool the TESs, which are attached to the pill, down to operating temperature the ADR must first be magnetically *cycled*. During the magnetisation phase of the cycle the field is slowly increased over a period of ~ 15 minutes with the heatswitch closed[1]. As the field increases the entropy of the pill decreases as the moments align with the field. Energy stored in the misalignment of the moments moves into vibrational degrees of freedom of the salt molecules increasing the pills temperature. Entropy moves from magnetic degrees of freedom to thermal degrees of freedom. By the time the magnet reaches full field the pill temperature is a few degrees above the 3 K plate. The pill is then *soaked* at full field for at least an hour, during the soak phase heat flows from the pill through the heatswitch to the cold plate and the pill cools towards 3 K. The heatswitch is then opened isolating the pill and the field is decreased over a period of about 20 minutes in the adiabatic demagnetisation phase of the cycle. As the field decreases the magnetic moments are knocked out of alignment by thermal motion absorbing the thermal energy and cooling the pill. Entropy moves from the thermal degrees of freedom to the magnetic degrees of freedom. The temperature of the pill and attached TES can be controlled by adjusting the current in the magnet which alters the field strength. When the pill temperature reaches the desired operating temperature control of the power supply providing the magnet current is handed over to a proportional-integral-differential (PID) controller to regulate the temperature.

While the PTR has a cooling *power* capable of maintaining a temperature indefinitely under a thermal load of less power, the ADR stage has a cooling *energy* and can only maintain a temperature under any load for a finite *hold* time. With a roughly 90 minute cycle our system has approximately 8 hours of hold time at 100 mK. Although I have fully automated the cryostat by adding a motor to operate the heatswitch and developing software so the cryostat

[1] If the current in the superconducting magnet is changed too rapidly the protection circuit will activate due to the back EMF. When the voltage across the magnet leads exceeds $\sim 0.7\,\mathrm{V}$ the circuit shorts the magnet. This protects personnel and external equipment if there a loss of cooling power or another event that causes the magnet to stop superconducting while significant field current is flowing. Without the circuit the energy stored in the magnetic field would exit the cryostat via the magnet power leads in a high current and voltage pulse in a process called a magnet *quench*.

can be monitored and controlled remotely from the same script controlling an experiment, the cryostat adds an extra layer of complication to experiments involving TESs.

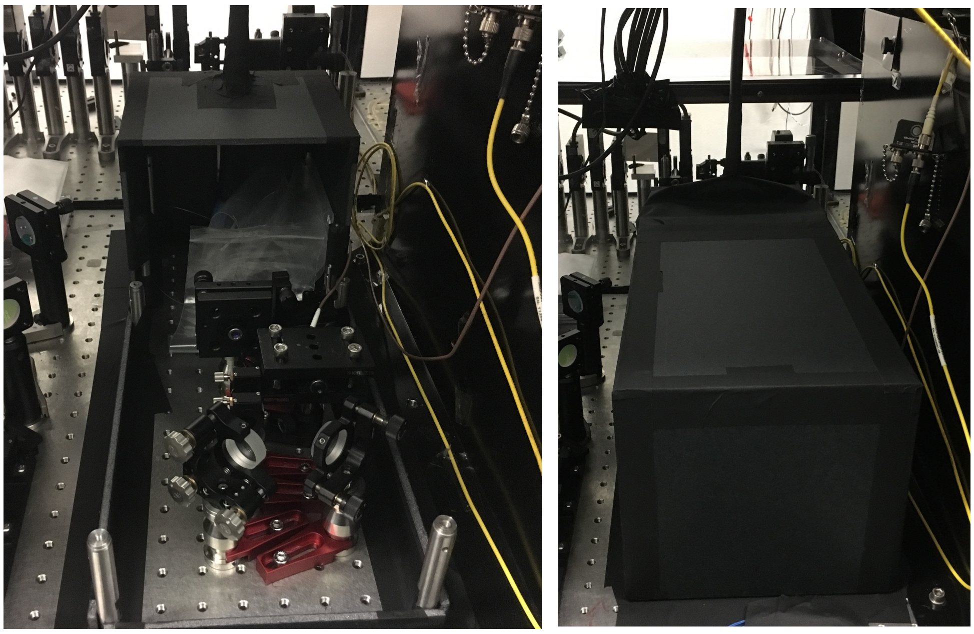


Figure 3.2: Test source: The source of weak optical pulses used for testing is a nominally 820 nm laser diode driven by a 50 ns 2.6 V electrical pulse at 10 kHz from a pulse generator. The diode and associated optics are housed in a light tight box and fibres run from the box to TES in a light tight conduit. Unused fibres connected to TESs are in the plastic bags seen at the back of the box. The diode output (middle right) passes though a 2 nm wide spectral filter centered at 820 nm (not shown). After the filter the beam is steered by two mirrors (front) through two linear polarisers, which provide variable attenuation to adjust the average photon number in a pulse, to the fibre coupler leading to the TES (middle left just in front of the plastic bags).

The optical input used to analyse the processors performance is provided by an ∼820 nm laser diode driven by a 50 ns wide 2.6 volt electrical pulse at 10 kHz. The laser output is filtered by a 2 nm wide spectral filter centered at 820 nm. The diode and associated fibre coupling optics are housed in a light tight box (Figure 3.2) and fibres to the TES run through a light tight conduit. Fine control of the pulse intensity is achieved with a pair of linear polarisers.

3.2 Preliminary performance analysis

3.2.1 Data acquisition

The TES system output from the ×100 preamp (Figure 1.2) is AC coupled to Lecroy model 1855A amplifier for further amplification before being passed to the ADCs attached to the FPGA. Data was captured during same hold cycle of the ADR with four levels of optical attenuation[2] (see Table 3.1). A second processor channel measures the electrical pulse driving

[2] The names indicate the dominant photon peak on the live MCA display as I adjusted the attenuation.

Dataset	Events captured	Model distributions	Acquisition time (s)
peak4	17834243	15	1800
peak3	17744605	14	1800
peak2	16960239	11	1800
peak1	14894211	8	1800

Table 3.1: Data capture details. The captures are of `pulse` event packets. Shorter (1 minute) captures of `single_trace`, event packets which include a record of `f`, were also taken to create figures containing traces. The model distributions column indicates the maximum photon number assignment in the measurement model for that dataset.

the laser to be used as a herald of possible detections.

The processors central MCA can capture a wider set of measurements than can be returned in event packets. In particular it is not constrained to measurements of *valid* rises or pulses as determined by the `pulse_threshshold` and `slope_threshold` registers, see Figure 2.1. The extrema sequences are the extreme value of a sequence since its previous zero crossing. The $\mathtt{f}_{extrema}$ and $\mathtt{s}_{extrema}$ sequences are used to find appropriate settings for the `pulse_threshold` and `slope_threshold` registers. See Figure 3.3 and Figure 3.4. The thresholds were set so that some noise is captured in the datasets.

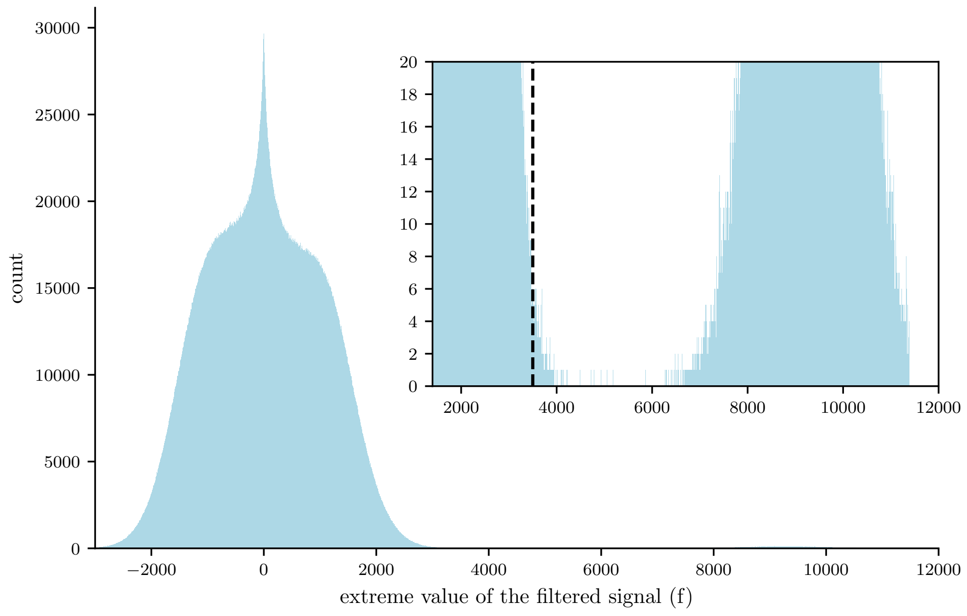


Figure 3.3: Distribution of $\mathtt{f}_{extrema}$ signal. `f` is the extreme value of `f` since its previous zero crossing but the figure only shows the distribution of the maximum. The inset shows a zoomed in view with the distribution of the noise on the left and the single photon distribution on the right. The vertical line indicates the `pulse_threshold` is set into the tail of the noise distribution.

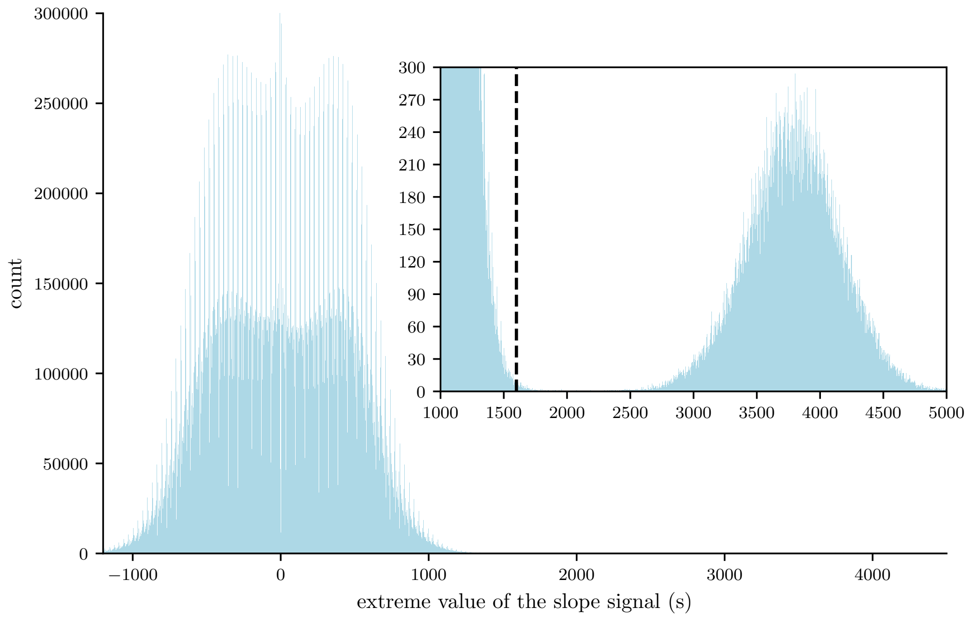


Figure 3.4: Distribution of $\texttt{s}_{extrema}$ signal. `s` is the extreme value of `s` since its previous zero crossing but the figure only shows the distribution of the maximum. The inset shows a zoomed in view with noise on the left. The peak on the right is distribution of the maximum value of `s` during detection of a single photon. The vertical line shows `slope_threshold` is set just into the noise distribution. I believe the fine structure in the distribution is because the `s` sequence is rounded to a 16.8 bit (16 bit wide with 8 fractional bits) value at the output of the differentiator. Many of the fractional bits are not significant and there is a periodic variation in the probability that the filter stages generate values over the 16 bit range. The distribution becomes smooth when MCA `bin_n` register is used to increase the histogram bin width.

3.2.2 Dark noise

In its current location in the lab our TES are often plagued by electromagnetic interference. I've done what I can to trouble shoot the problem and improve earthing without success. To collect the data I limit the signal bandwidth to 1 MHz using the analogue input filters in the Lecroy amplifier's input stage, this removes most of the interference. Unfortunately the restricted input bandwidth also limits the utility of the `s` sequence in discriminating between noise and true photon detections.

Figure 3.5 displays traces captured over 30 minutes with the laser diode off and includes events with higher energy than seen even with the laser is on. I speculate that these are cosmic ray related as generally they rise and cool a slower rate than a photon absorption which may indicate energy absorbed near the sensor heating it more slowly. Most of the events detected with the laser off are a combination of TES noise and photons from the high energy tail of the blackbody spectrum. A few photons from the lab still manage to couple into the fibre leading to the sensor, the rate is very low, ~ 1 event every 10 minutes[3].

[3] We have a more sensitive TES that has not been fully analysed, it has higher intrinsic gain and gives better

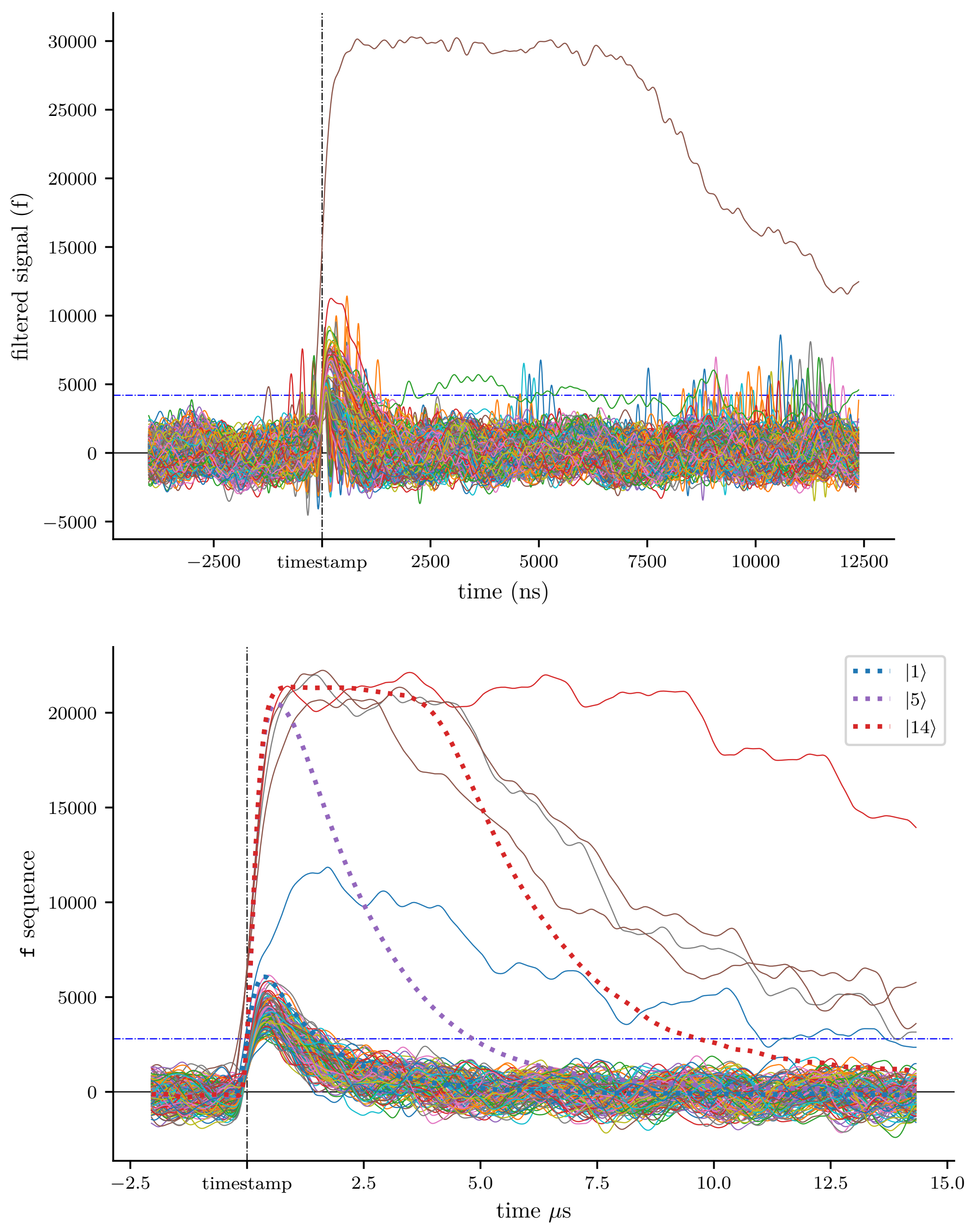


Figure 3.5: Dark traces captured over 30 minutes with 30 MHz (top) and 1 megahertz (bottom) bandwidth at the input of the Lecroy amplifier. The dash-dot blue line indicates the `pulse_threshold` setting. The thick dotted lines indicate the average pulse shape found by first assigning a photon number to traces using techniques described in subsection 3.2.3 then averaging over all traces assigned the same photon number. The majority of the 141 dark events captured during the 30 minutes are due to a combination of blackbody photons and intrinsic TES noise.

3.2.3 Statistical modeling

This analysis focuses on the pulse `area` measurement as it has the highest dynamic range. The techniques can also be applied to the other measurements such as rise height, maximum slope

discrimination between the blackbody and 820 nm photons.

etc.

If a magic box that emitted a known photon number (Fock) state at the press of a button existed, I could use it to characterise the the performance of the TES and processor. Repeatably pressing the button that emits a single photon would enable the distribution of area measurements for single photon detections to be determined. Pressing the two photon button allows the two photon measurement distribution to be estimated etc. The parameters of each distribution are dependent on the energy of the measured Fock state, the sensors energy resolution at that energy, the intrinsic TES noise and noise added by electronics and the `area` measurement process. In the absence of magic the best I can do is input an unknown Fock state superposition.

The Expectation Maximisation (EM) [DLR77] algorithm computes a Maximum Likelihood (ML) estimate in the presence of hidden data. I use it to fit a a mixture model made up of multiple component distributions to the measurement data. In this case the hidden data is the weight of each component in the mixture which originates from the unknown Fock state superposition generated by the pulsed laser diode (Figure 3.2).

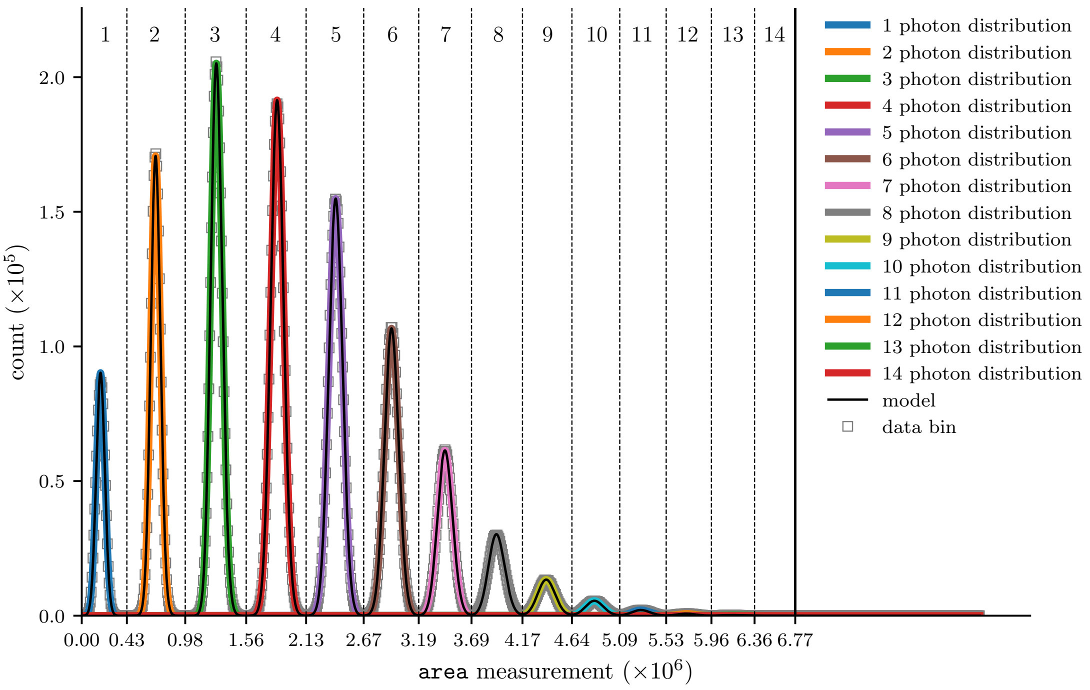


Figure 3.6: The `area` measurement model created from the peak3 dataset. The distribution of measurement values is modeled as a mixture of gamma distributions which are fitted using the expectation maximisation algorithm, see text for details. The algorithm finds the maximum likelihood estimates of each distribution's parameters and its weight in the mixture by systematically adjusting thresholds (dashed vertical lines) that partition the population of measurement values into sub-populations belonging to individual distributions. Only measurement values less than solid vertical line are fit to distributions but the total number of values is used in the calculation of the weights.

I implement the simplest form of the algorithm:

- The maximisation step takes a set of thresholds that partition the measurement values into sub-populations. Each sub-population is assumed to be sampled from one distribution in the mixture. ML estimates of the parameters for each distribution are then calculated from each sub-population and the proportion of the total measurement values each sub-population represents serves as an estimate of the hidden data.
- The expectation step uses the distribution parameters and hidden data estimated at the previous maximisation step to update the thresholds. New thresholds are set by finding the intersection of the probability density functions (PDFs) for each neighbouring distribution.

Maximisation and expectation are iterated over until the likelihood of the mixture model converges. When the individual distributions in the data are well defined and reasonable initial thresholds are supplied this simplified EM procedure performs well. A set of initial thresholds used to seed the EM algorithm is found by first constructing a histogram from the measurement value population, convolving it with a smoothing filter and then finding the peaks of the smoothed histogram. Initial thresholds are set at the midpoint between the discovered peaks with two additional thresholds added to bound the range of of measurement values used by the algorithm. An initial threshold at 0 set the lowest value used and one symmetrically placed after the last discovered peak sets the highest value used. These bounding thresholds remain fixed during the expectation maximisation process.

To compare models I use the Akaike information criterion (AIC) [Aka98],

$$\text{AIC} = 2k - 2\ln(\hat{L}),$$

where k is the number of degrees of freedom of the model and $\hat{L}$ is the maximum of the likelihood function. The AIC is an asymptotically valid estimate of the information lost by a model of some unknown distribution. When two models are compared the one with the lower AIC is preferred. The relative likelihood is,

$$\exp\left(\frac{\text{AIC}_1 - \text{AIC}_2}{2}\right),$$

where AIC_1 is the lower AIC of the two models. The relative likelihood estimates the probability that the model with AIC_2 minimises the information loss rather than the model with AIC_1.

Comparing measurement models composed of Gaussian, skewed Gaussian and gamma distributions the relative likelihood establishes that a mixture of gamma distributions is the most likely of the three model types to minimise information loss. The `area` measurement model constructed from the peak3 dataset is shown in Figure 3.6.

Counting thresholds are found using a measurement model consisting of M distributions by normalising them and calculating the intersection of the neighbouring PDFs. These $M - 1$

thresholds are labeled t_n where $n \in [1, M-1]$ and are used to assign photon number n to measurement value a. When $a \leq t_{M-1}$ it is assigned photon number n where $t_{n-1} < a \leq t_n$ and $t_0 \equiv 0$. When $a > t_{M-1}$ it is assigned the photon M_+ indicating a detection of *at least* M photons, see Figure 3.7.

Probabilities for assigning the photon numbers to a measurement value can be found from the overlap of the distributions in a measurement model, see Figure 3.8. By using the values of the cumulative density functions (CDFs) of the distributions at the modeled counting thresholds an operator that takes an input Fock state superposition to a superposition of photon number assignments (measurement outcomes) can be created. Let p_n^f be the probability of assigning photon number n to a measurement of Fock state f, then

$$p_n^f = \begin{cases} cdf_f(t_n) - cdf_n(t_{n-1}) & f \leq M, n < M \\ 1 - cdf_f(t_{M-1}) & f < M, n = M \end{cases}$$
$$p_x^f = \begin{cases} 1 & f > M, x = M \\ 0 & f > M, x \neq M \end{cases} \tag{3.1}$$

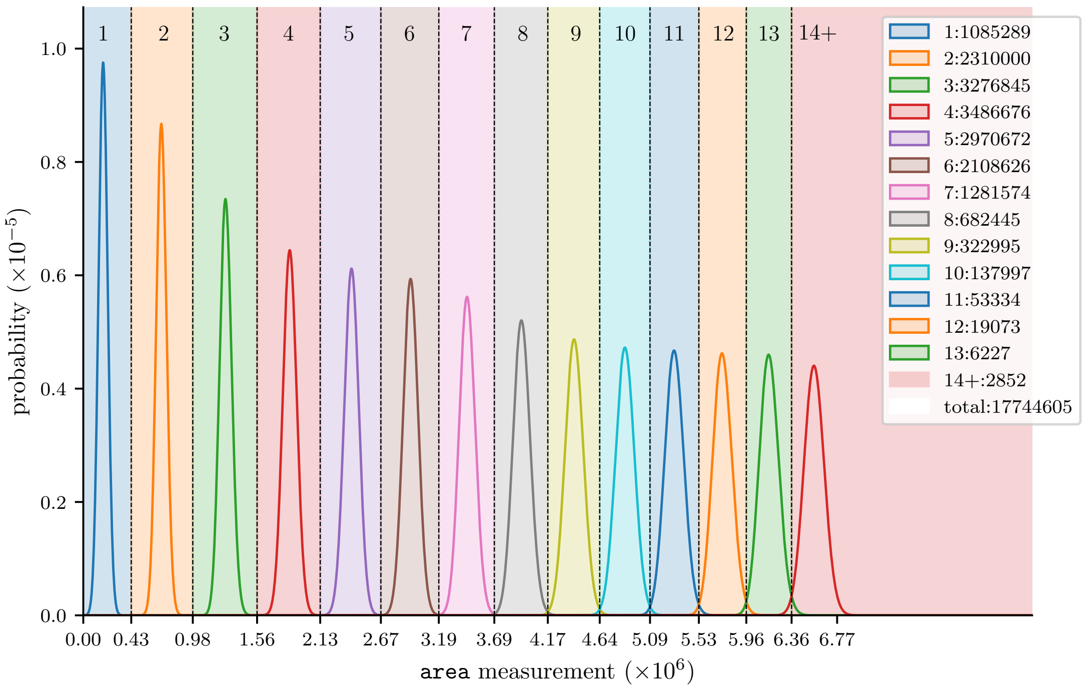


Figure 3.7: *Counting* thresholds are found by normalising each of the M distributions in the measurement model and calculating the intersection of the neighbouring PDFs. Photon number M_+ is assigned to all measurements greater than the $(M-1)^{th}$ threshold indicating detection of at least M photons, see text for details. The the figure shows the model constructed from the peak3 dataset and the numbers in the legend count the measurements assigned to each photon number.

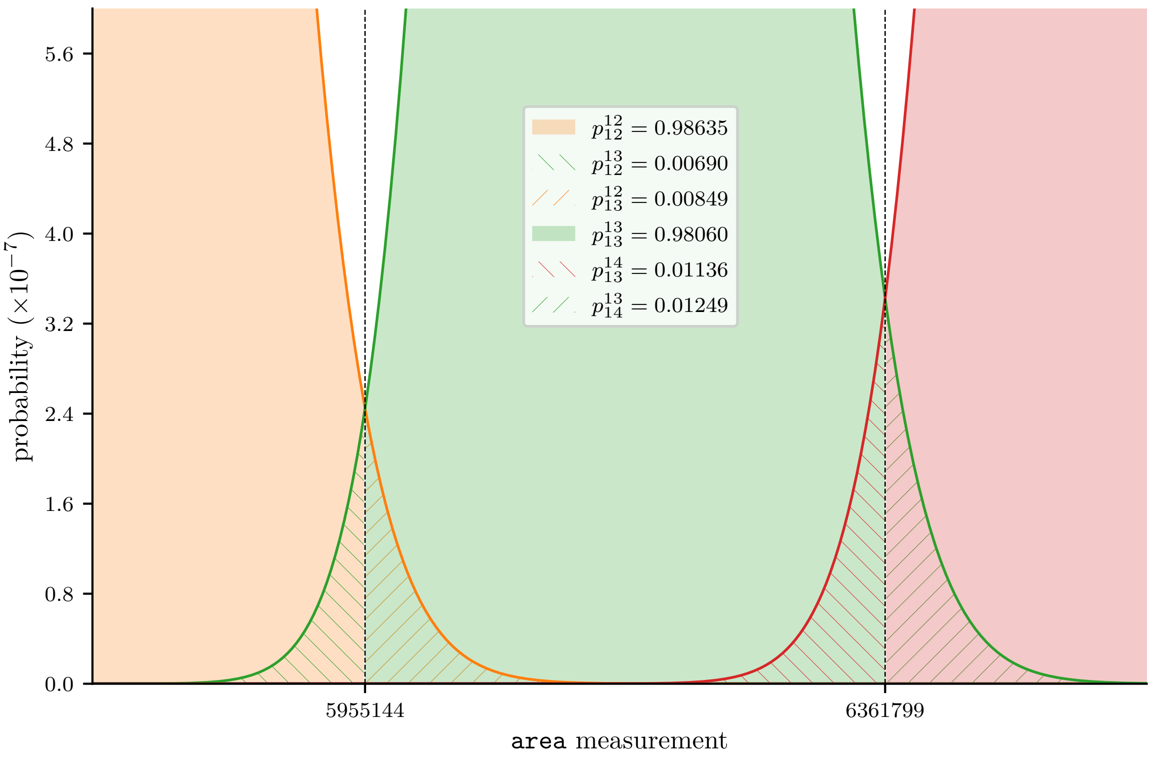


Figure 3.8: By using the values of CDFs of the distributions at the thresholds a the probabilities of assigning the wrong photon number to a measurement can be estimated. These probabilities can be used to estimate the the superposition of assigned photon numbers that arises from measuring a given Fock state superposition. The probability p_n^f is the probability of assigning photon number n to a measurement of Fock state f.

and we have

$$\sum_{i=1}^{M} p_i^f = 1 \tag{3.2}$$

These number assignment probabilities can be used to construct an operator similar[4] to a positive-operator valued measure (POVM), an operator used to describe generalised quantum measurement, which maps the state measured to measurement outcomes, see Figure 3.8.

As an operational test I reconstructed the laser's output state from the number resolved counts and the heralding information provided by the laser drive pulse. Vacuum counts are found by counting the number of heralds that are not correlated with a detection, see Figure 3.9 and Table 3.2. The laser output (Figure 3.2) is modeled as a two parameter Gaussian state

$$\rho_{\alpha,\bar{n}} = D(\alpha)\rho_{\bar{n}}D(\alpha)^\dagger, \tag{3.3}$$

where $D(\alpha)$ is the displacement operator and $\rho_{\bar{n}} = (1 - e^{-\bar{n}})\sum_n e^{-\bar{n}n}|n\rangle\langle n|$ is a thermal state with average photon number $\bar{n}$. When $\bar{n} = 0$ $\rho_{\alpha,\bar{n}}$ is a coherent state. To create the the laser

[4]There are no probabilities related to the vacuum outcome which may disqualify it as a true POVM

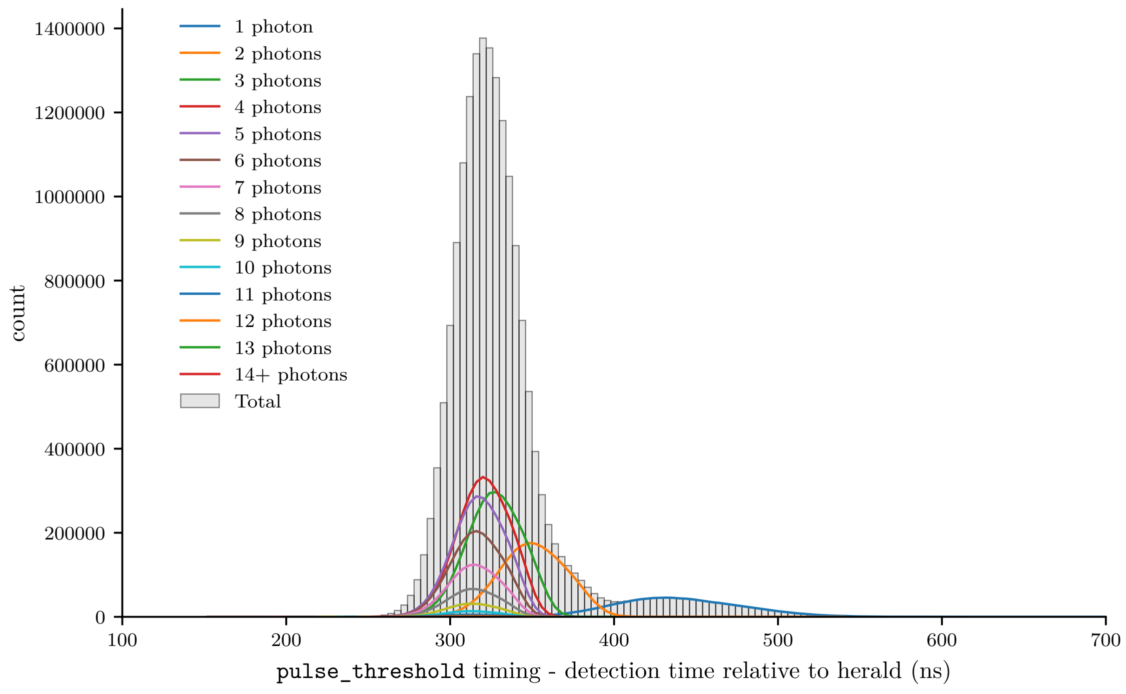


Figure 3.9: The datasets are captured with the `timing` register set to `pulse_threshold`. The histogram records the relative delay between a detection and a heralding signal derived from the electrical pulse that energises the laser diode. The probability of a vacuum detection is estimated as the number of heralding events not coincident with a detection divided by the number of heralding events in the dataset. A coincidence occurs when there is a detection between 100 ns and 700 ns after the herald.

	Correlated detections		Uncorrelated events		
	total	1-photon	heralds	1-photon	multi-photon
peak4	17833759	775469	166387	473	11
paek3	17744142	1085289	256007	458	5
peak2	16959820	2967401	1040327	416	3
peak1	14893808	5448324	3106340	399	4

Table 3.2: Temporal correlation between the heralding signal derived from the pulse driving the laser and TES detections. The photon number dependent jitter is shown in Figure 3.9. When the delay between the herald and detection is between 100 ns and 700 ns the herald and the detection are considered correlated and coincident. Uncorrelated heralds are considered vacuum detections.

output state estimates in Figure 3.10, I use the measurement model created from the peak3 datasets to analyse *all* four datasets to test how a single calibration performs with different optical inputs to the TES. I use a parametrised model of $\rho_{\alpha,\bar{n}}$ created with QuTiP[JNN13] to calculate the Fock state superposition probabilities, then apply the number assignment error

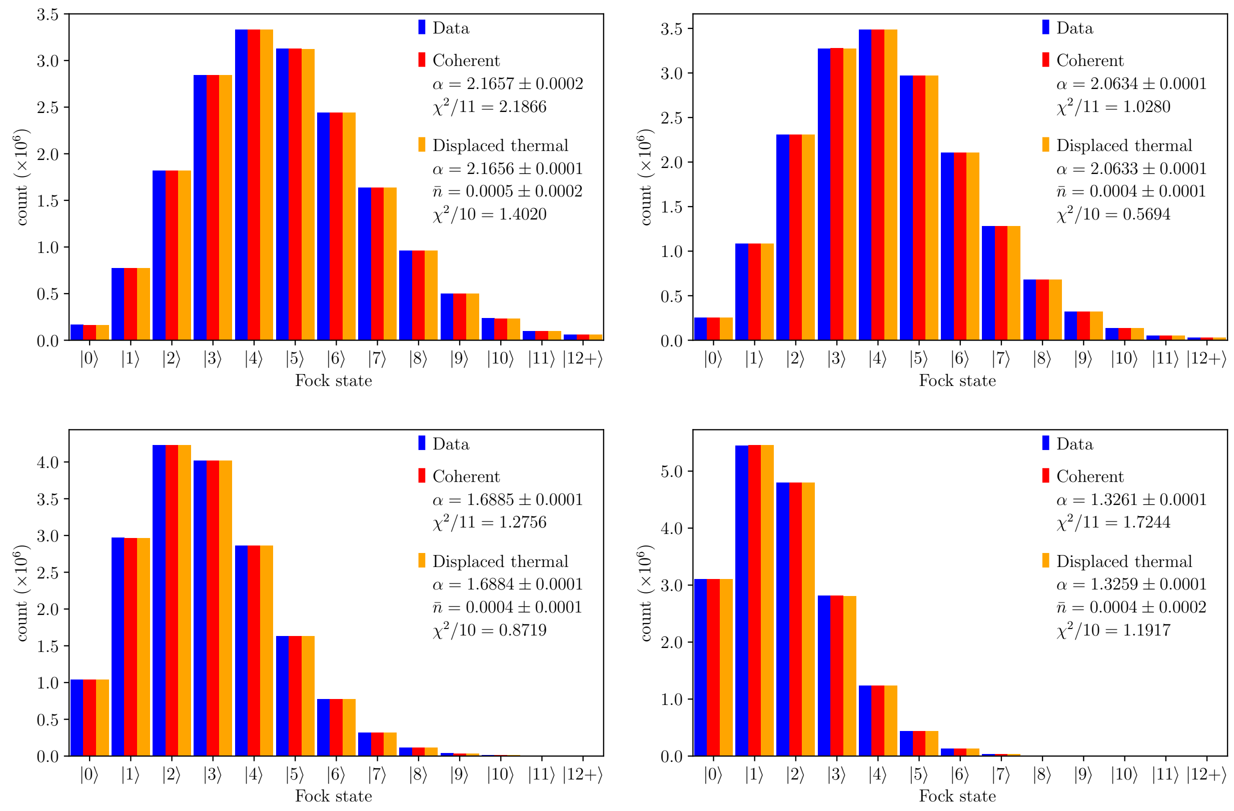


Figure 3.10: Least squares fitting of the datasets to model the optical input states. The datasets are peak4, peak3, peak2 and peak1 from left to right and top to bottom. Number assignment probabilities (Equation 3.1) and counting thresholds (Figure 3.7) from the measurement model created from the peak3 dataset are used to analyse *all* datasets. The number assignment probabilities are applied to the Fock state superposition produced by a parametrised model of a displaced thermal state ($\rho_{\alpha,\bar{n}} = D(\alpha)\rho_{\bar{n}}D(\alpha)^\dagger$) to estimate the measurement outcome probabilities. Measurement outcomes are contracted from 1-14_+ to 1-12_+ to minimise truncation effects due to the finite measurement model. The α and $\bar{n}$ parameters are estimated by non-linear least squares minimisation of the residuals between the modeled and observed measurement outcomes. The vacuum detection probability is estimated from the number of laser drive pulses not coincident with a detection, see Figure 3.9 and Table 3.2. Standard errors are calculated from an estimate of the covariance matrix created by the least squares minimisation. The coherent state model has a fixed $\bar{n} = 0$.

probabilities (POVM) in Equation 3.1 and contract the result from number outcomes 1-14_+ to 1-12_+ in order minimise truncation effects. To estimate the parameters of $\rho_{\alpha,\bar{n}}$ I perform a non-linear least squares minimisation[5] of the residuals between modeled and observed measurement outcomes. Figure 3.11 assumes the laser outputs coherent states and examines the stability of the state estimate over time. Each dataset is broken into subsets containing 100000 laser drive pulses and the α parameter is estimated as described above for each subset. Table 3.3 and

[5] Using the LMfit package which wraps the SciPy minimiser calling the MINPACK implementation of the Levenberg–Marquardt algorithm[Lev44]

Table 3.4 contain estimates of mean and standard error of the laser output state parameters from the subset populations plotted against time in Figure 3.11.

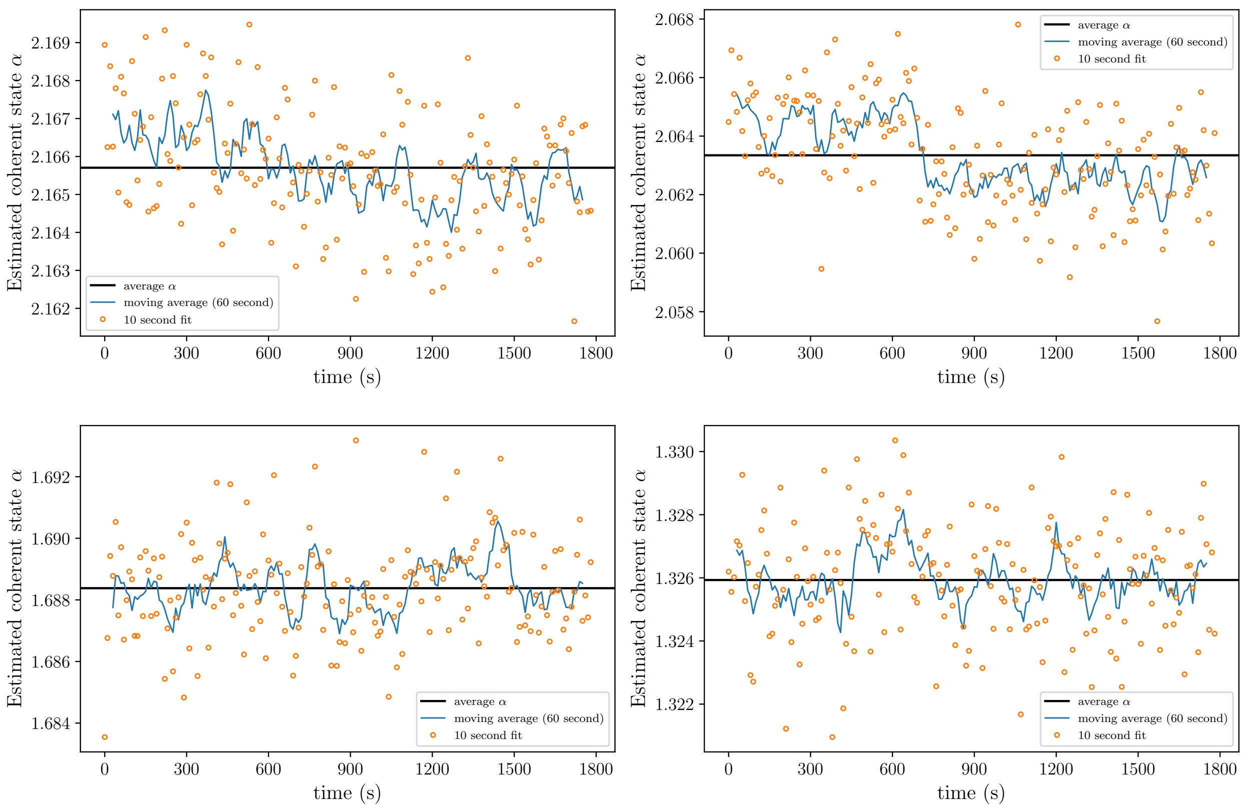


Figure 3.11: The stability of the coherent state estimate. Each data set is broken into subsets containing 100000 laser drive pulses, representing a 10 second capture time, and the subset is used to estimate α as described in the text. The blue line is a moving average over 6 of the 10 second estimates.

	100000 heralding events (10s)		running average (60s)	
	Coherent state α	$\chi^2/11$	Coherent state α	$\chi^2/11$
peak4	2.166 ± 0.002	1.0 ± 0.4	2.1657 ± 0.0008	1.0 ± 0.1
peak3	2.063 ± 0.002	1.0 ± 0.4	2.063 ± 0.001	1.0 ± 0.2
peak2	1.688 ± 0.002	1.0 ± 0.5	1.6884 ± 0.0007	1.0 ± 0.2
peak1	1.326 ± 0.002	1.0 ± 0.6	1.3257 ± 0.0007	1.0 ± 0.2

Table 3.3: Coherent state estimates based on the statistics of subsets containing 100000 drive pulses, values are recorded as (mean)±(standard deviation) from the population of estimates plotted against time in Figure 3.11.

	100000 heralding events (10s)		
	α	$\bar{n}$	$\chi^2/12$
peak4	2.165 ± 0.002	0.001 ± 0.001	1.0 ± 0.4
peak3	2.063 ± 0.002	0.001 ± 0.001	1.0 ± 0.5
peak2	1.688 ± 0.002	0.001 ± 0.001	1.1 ± 0.5
peak1	1.326 ± 0.002	0.001 ± 0.001	1.1 ± 0.7

Table 3.4: Thermal state estimates from the statistics of subsets of each dataset containing 100000 laser drive pulses. Values are recorded as (mean)±(standard deviation) of the subset population.

Chapter 4

Discussion and conclusions

4.1 Discussion

The accuracy of the scalar measurements the processor currently makes hinge on the accuracy of the baseline estimate. While capturing the peak4 dataset, the baseline shifted by approximately 400 from its value when the laser was off. The estimate of $\alpha = 2.166$ for the weak coherent pulses captured in peak4 implies an average rate 46920 photons per second. Though the dynamic correction described in subsection 2.3.1 was active (`baseline.dynamic`=True) the correction mechanism operates on comparatively long time scales. Baseline error effects the `area` measurement in an outcome dependent way. It adds area noise proportional to the length of the TES detection pulse which is correlated with the measured Fock state. Two major factors influence the quality of the laser output state estimations in Figure 3.10, Figure 3.11, Table 3.3 and Table 3.4, one is baseline error the other is errors in discriminating photon detections from noise.

Baseline correction through monitoring the mode of the `f` sequence is novel but was only intended as a first approximation to be used in the initial peak finding pipeline. A second order estimation can be made for each detection by adding some extra delay and pipelining enabling averaging over the `f` sequence over a fixed time range before the timing point of each pulse. The current prototype would be used to establish where and for how long the average should be acquired and what improvement in measurement uncertainty could be achieved.

Figure 3.11 displays the stability of the estimate of the laser output state over time. Each dataset in the figure was captured over a 30 minute period in the following order peak4, peak3, peak2 then peak1, time flows left to right top to bottom. Given that there is no temperature stabilisation of the laser diode, the drift in intensity seen in Figure 3.11 is consistent with a drift in the laser's wavelength as it thermalises. Wavelength drift is converted into drift in the average photon number per pulse by the 2 nm wide filter. In this scenario, the abrupt drop in α in the middle of the peak3 dataset can be attributed to a mode hop. An alternative explanation is that the drift in the estimated pulse intensity is due to drift in the baseline estimate. Though I favour the former explanation, the true picture most likely involves a combination of the two.

Further investigation is required and the processor is capable of capturing `pulse` packets, as was done to collect the datasets, while simultaneously capturing distributions streamed from the MCA. These two datastreams can be correlated in time to investigate the stability of the baseline estimate in more detail.

In general, the use of thresholds in classification is not considered best practice as they represent a hard binary choice between two classes, above and below, whereas other techniques can offer a fuzzier probabilistic classification. Processing using hardware circuits imposes considerable constraints on the processing that can be performed and the use of thresholds appeared to be the only tractable design choice. Discrimination of photon detections from noise is the main task that needs to be examined in this light.

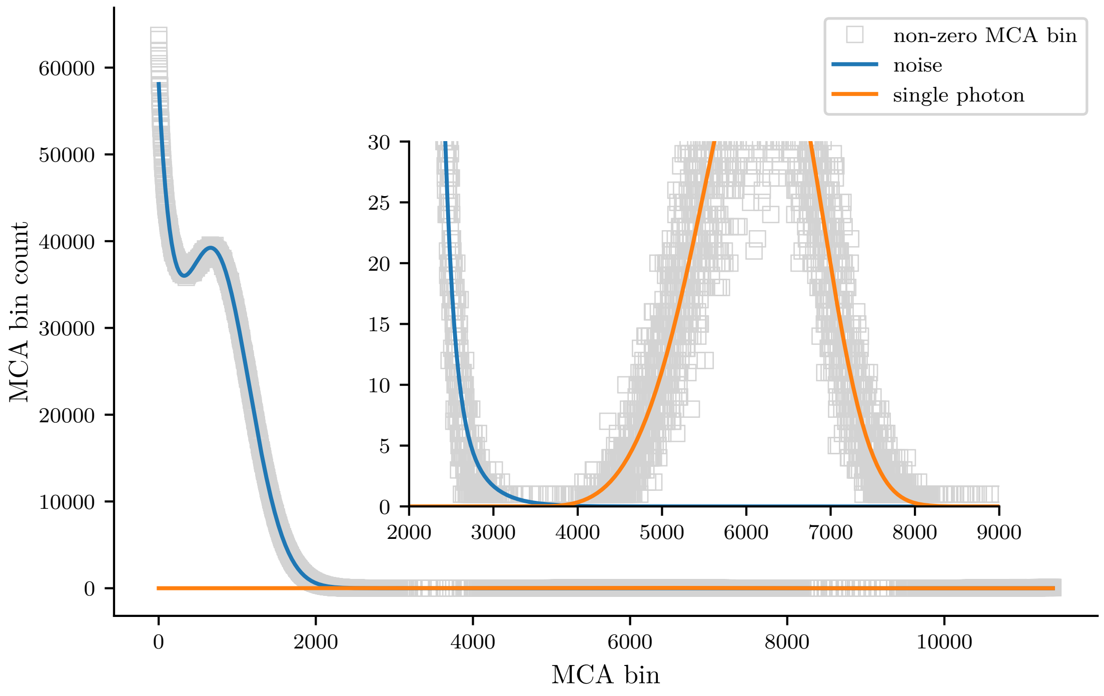


Figure 4.1: The distribution $\mathtt{f}_{extrema}$ sequence captured for 1 minute showing the distribution of the maximum value the filtered TES signal ($\mathtt{f}$) obtains between baseline (zero) crossings. The inset shows a zoomed in view. On the left is the noise distribution which can be modeled as a mixture of two Weibull distributions (blue line) and on the right is the single photon distribution which can be modeled as a single Weibull distribution (orange line). This information can be used to estimate the probabilities of error in number assignment between the vacuum and single photon term and in turn estimate the single photon loss and dark count (noise) probabilities for a given `pulse_threshold` setting. These probabilities can then be used to complete the POVM, see Equation 3.1. A realtime display could be created that estimates the overlap of the noise distribution and the single photon distribution as an aid in tuning the TES biasing, see Figure 1.2.

In the analysis presented in subsection 3.2.3, the discrimination between signal and noise is achieved using two different thresholds each on different but correlated sequences. A rise is the detection signal is classified as a photon detection when the filtered TES signal (the `f` sequence) exceeds the `pulse_threshold` setting *and* the maximum value of the slope of `f` (the `s` sequence) exceeds the `slope_threshold` setting during the rise. These threshold settings trade off between efficiency, or loss of a photon detection, and counting noise as photon detection, or dark counts. The settings used to capture the datasets analysed in subsection 3.2.3 favour efficiency and are set into the noise distributions, see Figure 3.3 and Figure 3.4. Table 3.2 contains correlated and uncorrelated counts for the datasets and shows that the dark count rate is roughly 800-900 per hour. Generally dark counts will be assigned a photon number of 1 and loss will be generally of detections that would have been assigned photon number 1. The measurement outcome dependence of these errors degrades the quality of the laser output state estimations. Estimation of the loss of single photon detections is not addressed in my current analysis. Direct accounting of the loss maybe possible using the techniques of correlated photon metrology[Mig08] or with a calibrated attenuator[LGPM15]. The MCA provides an alternative method for estimating the dark count and single photon loss probabilities that I have not analysed in detail.

Figure 4.1 shows the distribution of the $\mathtt{f}_{extrema}$ sequence which is the maximum value the filtered TES signal reaches between baseline crossings. The distribution of the TES noise can be modeled as a mixture of two Weibull distributions and the single photon response as another Weibull distribution. The overlap of the noise and single photon distribution can be used to estimate the single photon loss and dark count probabilities in a similar way to Figure 3.8 and be used to add the missing POVM elements in Equation 3.1. The overlap of the noise and single photon distributions depends on the gain of the particular sensor which is subject to fabrication variability and the TES biasing. A realtime display can be constructed using the MCA that would allow the biasing to be adjusted to minimise this overlap. The probabilities of single photon loss and single photon noise are dependent on the threshold settings, which can be used to tune the trade off between dark counts and detection loss.

Another processor feature that can aid in discrimination of signal from noise but has not been analysed, is the dot product measurement returned by the `dot_product` event packet. The dot product module in each channel operates on the same sequence record that would be returned as a trace as specified by the `trace_length`, `trace_stride` and `trace_pre` register settings, see Figure 2.1 and section 2.3. The vector dot product operation is performed on the sequence record and a stored template with the same sequence length then recorded in the `dot_product` field of the event packet. It was shown in [LGM$^+$12] that the dot product provides respectable energy discrimination. I expect it also improves discrimination of signal from noise as it incorporates some pulse shape characteristics. By using the average single photon response as a template my expectation, based on these preliminary results, is that a combination of `dot_product` measurement, the `pulse_threshold` and `slope_threshold` settings and perhaps the `rise_time` measurement will significantly improve discrimination of signal from noise.

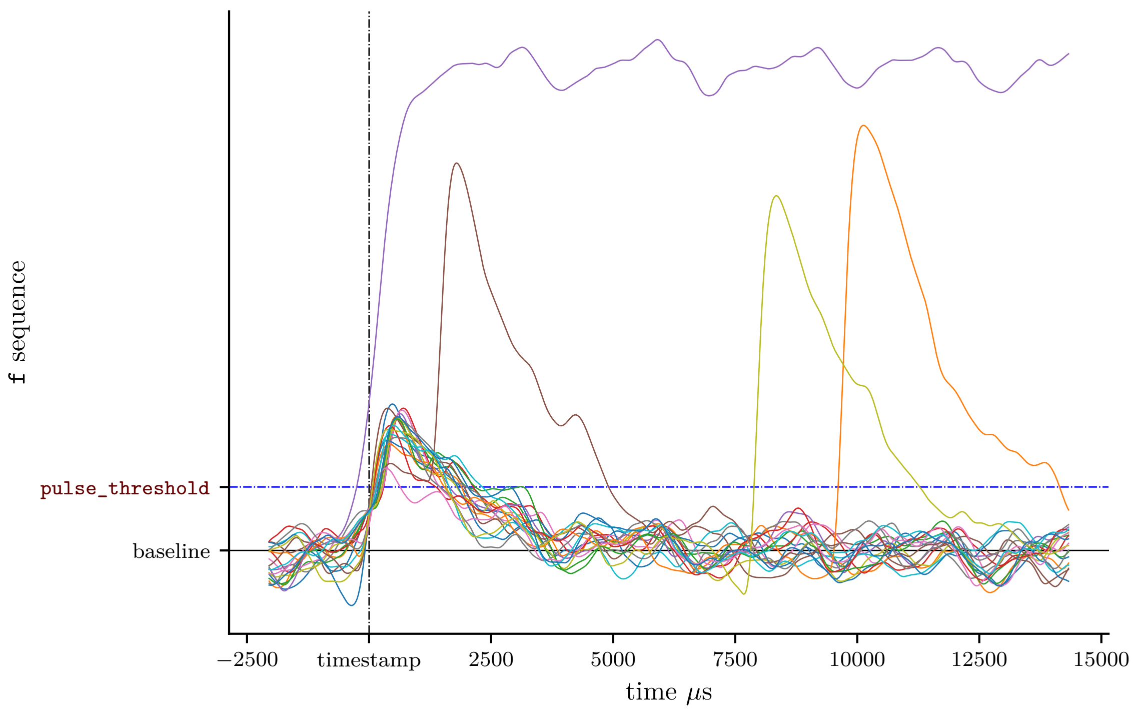


Figure 4.2: Uncorrelated traces captured over 1 minute with `cfd_low` timing. The peaks that are delayed in the traces are due to a noise event triggering the trace capture which was followed by a photon photon event during the time the trace was captured. When capturing only point measurements in a `pulse` event packet two event are seen as separate if the signal falls below `pulse_threshold` between them. In this case, two `pulse` event packets with be returned. For the brown photon event that occurs closest to the timestamp the signal does not fall below the `pulse_threshold` after the noise event triggers the sequence recording. In this case the `pulse` packet will contain two rise records, see Figure 2.1. Multi-rise information has not been used in the analysis presented in this thesis.

It should be noted that the testing and analysis in chapter 3 is performed under non optimal conditions, perhaps even approaching worst case. I choose the sensor that displayed the worst intrinsic gain (due to fabrication variation) to analyse. I biased the sensor "by eye" using an oscilloscope and the live MCA display of the distribution of $\mathtt{f}_{extrema}$ signal (without any curve fitting in Figure 4.1) so it is unlikely to be optimally biased. The bandwidth is limited to 1 MHz at the input of the Lecroy amplifier effecting the utility of slope in discriminating signal from noise. The analysis does not use the heralding signal to post select any noise events only to estimate the vacuum count.

4.2 Conclusions

This thesis describes the design development and testing of a hardware circuit library for use with the remarkable Transition Edge Sensor. The library is written in VHDL and a prototype signal processor based on its components is implemented in a Field Programmable Gate Array for testing. The prototype makes “point” measurements of a TES pulse such as its area, height and length. The goal of the hardware approach is to provide realtime, time and number resolved coincidence counting across multiple channels at the highest rates the sensors can deliver. Existing techniques capture the entire sequence record of a detection pulse and process it in software. Software based approaches allow arbitrarily complex processing which delivers low energy uncertainty but can have potential scalability issues, difficultly delivering information in realtime and trouble capturing long high detection rate records without dropping detection events.

Though testing is preliminary and the analysis is rudimentary the initial results are very encouraging. My optimism is based on the quality of estimates of the laser output state in Figure 3.10, Figure 3.11, Table 3.3. The quality of the estimate is reasonable across the timescales examined. Photon number assignment for all the datsets is based solely on the measurement model derived from the peak3 dataset indicating the characterisation is reasonably stable across different pulse intensities. Least squares and the χ square statistic are not really the most appropriate indicators and more rigorous likelihood based analysis is warranted, this will follow in a peer reviewed publication. Based on the estimate of the likelihood from the least squares minimisation use of the POVM elements (Figure 3.8) does not significantly change the parameter estimates but does improve the likelihood of the fit and therefore the reduced χ square. This indicates the estimated POVM elements are reasonable. I believe the increase in variance of the reduced χ square with decreasing α seen in Table 3.3 is due to baseline error. In the peak4 dataset most of the electrical pulses that drive the laser result in a detection whereas far less drive pulses produce a detection for the peak1 dataset, see Table 3.2. Under the conditions of the peak1 dataset there are more occasions where the TES cools for 2 or 3 drive pulse periods and this may increase baseline errors as the baseline estimate is based on averaging over longer time scales. Section 4.1 has details of how a second order, per detection baseline estimation can be implemented which would reduce energy uncertainty at low average photon numbers.

The displaced thermal state model for the laser output was introduced during early testing of the prototype. The first data I captured with the processor was produced by a $\sim 830\,\mathrm{nm}$ laser diode driven by a $5.6\,\mathrm{V}$ $2\,\mathrm{ns}$ electrical pulse at $100\,\mathrm{khz}$ and under those conditions the displaced thermal model was the best estimate of the laser output state. For the data analysed in this thesis using a $\sim 820\,\mathrm{nm}$ diode driven at $100\,\mathrm{khz}$ by a $2.6\,\mathrm{V}$ $50\,\mathrm{ns}$ pulse the thermal character has effectively disappeared and laser output is best modeled using a coherent state. The AIC for both models based on the likelihood estimate from the least squares minimisation does not significantly favour one model over the other. The estimates of $\bar{n}$ the average number of photons in the thermal state before displacement are consistent with $\bar{n} = 0$, see Table 3.4. Further work is required to establish if the thermal nature seen in earlier data is optical or due to measurement error related to baseline estimation. The output estimates should be investigated under different drive repetition rates, pulse widths and voltages. Optical responses of laser diodes under these drive conditions are not well described in the literature.

Parts of my analysis are still qualitative but I expect that the the processor has low enough number uncertainty, as is, for number resolved coincidence counting for the quantum information experiments we have planned with 820 nm photons. There is a clear program of development that will reduce the current energy uncertainty and should make the processor more useful at longer wavelengths where the energy per photon is lower. I don't have any of the standard software techniques implemented in our lab and time pressure prevented me visiting NIST Boulder to see them in action so quantitative comparison of the software and hardware processing approaches is difficult. Ideally a head to head comparison of the two approaches operating on the output from the same sensor should be performed. Many processor features described in Part II are yet to be tested or examined in detail.

The realtime capabilities of hardware processing aid in the setup and optimisation of experiments. My MCA design is capable of operating continuously and without loss at the 250 MHz ADC sampling frequency and opens a novel view on TES signals via capturing the probability distributions of various signal statistics. Use of the MCA may also lead to improved TES biasing procedures (Figure 4.1) and, if the biasing voltages were computer controllable, would enable full automation of TES tuning.

Not described in this thesis is the capture server software that runs on a host computer. Though designed by myself, in parallel with the circuit library, the server software was developed and implemented by fellow PhD student Alexandrina Nickolova. The server captures the packet stream produced by the processor and produces daughter streams for realtime display. This includes the counting of number coincidence patterns. The server software is capable of capturing a full record of the eventstream produced by the processor as a number of indexed files allowing easy access to the data. Data analysed in subsection 3.2.3 was captured in this way. The server streams and the processors registers can be accessed from any network device capable of running Python which allows full automation of data acquisition and control the experimental circuit using python scripts or Jupyter notebooks. I'll leave detailed explanation of the capture server software to Alexandrina's thesis.

There are a number of bugs that need to be addressed, a clear program of improvements to be made, and more rigorous testing and analysis are required but all this is typical of a prototype at this stage of development. Though the design, hardware development, circuit verification, testing and software development for the circuit library was a long and often painful process, I am uncharacteristically pleased with the result.

Part II

Implementation Details

Chapter 5

Stream processing

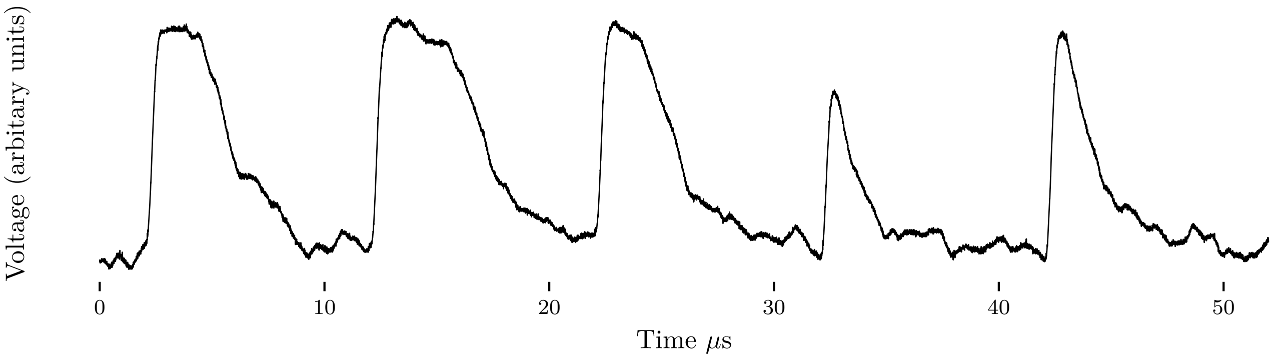


Figure 5.1: TES output after room temperature amplification. The optical input is from a ∼830 nm laser diode driven with a 100 kHz electrical pulse which produces optical pulses with average photon number $\langle n \rangle \sim 3$.

Each processor channel converts the stream of detection pulses output by a TES depicted in Figure 5.1 into a stream of *event* packets called the *eventstream* (section 5.2). Event packets contain pulse measurements, made in the channels measurement pipeline (chapter 6), and a timestamp. Processor *registers* (appendix Appendix A) control how the measurements are made and which measurements appear in the event packet. Each channels eventstream is merged into a single timestamped eventstream (section 5.3) which is packaged for transport to the host.

See Figure 5.2 for a schematic overview of the prototype. The schematic is hyperlinked, each pipeline element links to a description of what it does and the description links to registers that control it.

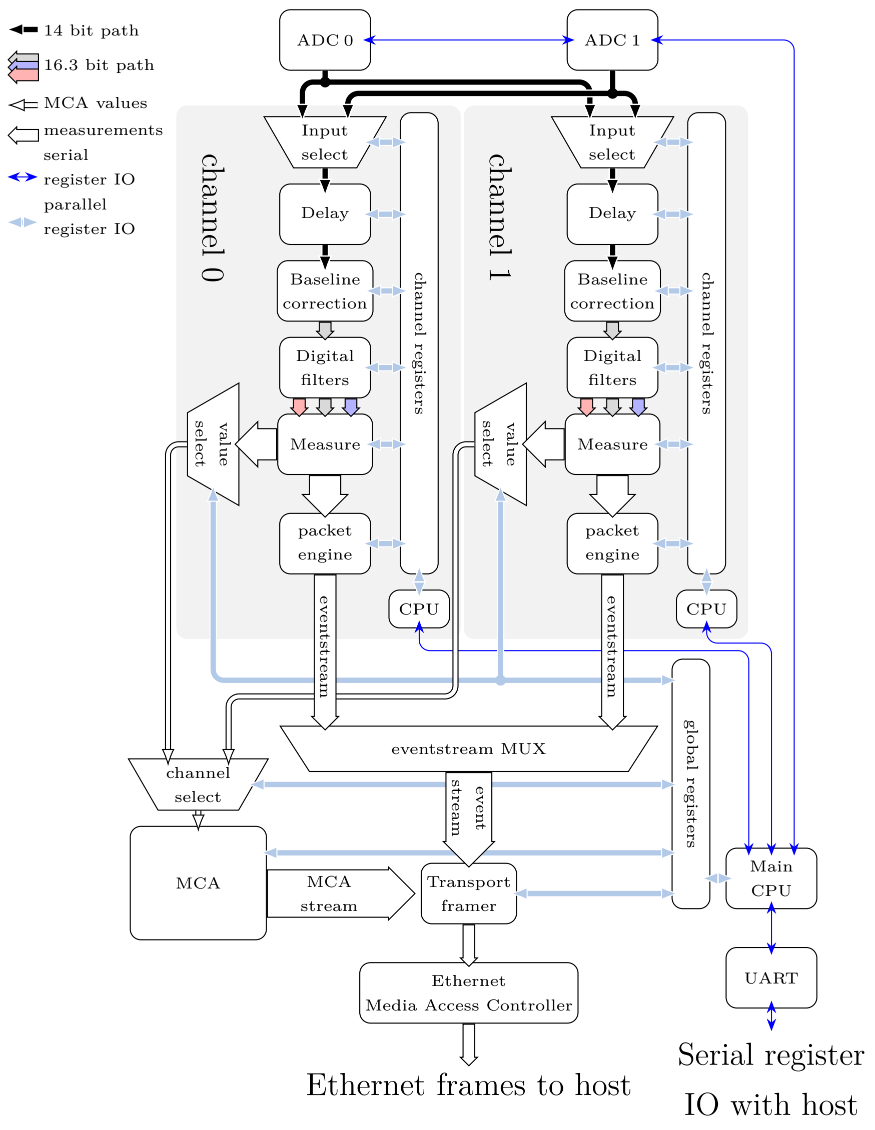


Figure 5.2: Hardware prototype: The amplified analogue TES signal is digitised by 14 bit 250 MHz ADCs then processed by channel pipelines which condition the signal, extract detections and make measurements to be included in an event packet. The stream of event packets, called the *eventstream*, produced by each channel are merged by the eventstream multiplexer (MUX) which adds timing information. The merged event stream is then transported to the host computer as Ethernet frames over a dedicated point to point connection. The processor contains a collection of internal registers controlling how measurements are made, what appears in packets and other processor functions. Register IO with the host is mediated by embedded 8 bit central processing units (CPUs) which handle serialisation/de-serialisation, address decoding and serial protocols required to communicate with external chips. For clarity, two channels are shown though eight are possible with the current eventstream MUX design. The MCA captures the distribution of a selected measurement as a histogram. Note this figure is hyperlinked, each subsystem links to a more detailed description.

5.1 Notation and terminology

Streams, packets and frames

A *packet* is an ordered collection of *fields* carrying data and there are a number of packet types. A *frame* is a similar structure used to transport packets to the host containing a header, with fields dictated by the particular transport protocol, and a payload. Each transport protocol defines a maximum frame length called the maximum transmission unit (MTU). Frame payloads in this design fall into two classes; an array of short event packets of the same type and size; or a packet *fragment* if the packet is larger than the MTU. Packets carrying a *trace*, which is a fixed length record of a *sequence*, or a histogram from the MCA are the only types large enough to fragment.

A *stream* is a sequential flow of information between a *source* and a *sink* managed by a protocol. All streams in the design comply with the advanced extensible interface (AXI) stream protocol which is part of the widely used advanced micro-controller bus architecture (AMBA) open-standard. An AXI stream uses three signals to control the *transfer* between source and sink[1].

The *ready* handshake is asserted by the sink when it can accept data.
The *valid* handshake is asserted by the source when it has data to transfer.
The *last* signal is used by the source to indicate the last transfer in a packet.

Transfer occurs when both handshakes are asserted and amount of data in each transfer is called the width of the stream, all streams in this design are 64 bits wide.

Packets and frames are assembled by a *framer*, an entity I created that has random access for writing and exposes an stream interface for reading, a cross between RAM and a first in first out (FIFO) queue. The framer solves a problem that occurs when information arrives in a different order to the ideal transmission order, for example, transport frames usually have a field containing the frames length but the length is usually not known until the entire frame is processed. The framer easily allows the length to be in a header field at the beginning of the frame so the host can allocate a buffer to hold the payload. Framers are critical in creating event packets because the field information does not arrive in the order required to optimise memory alignment. The framer in a *packet engine* generates the eventstream for each processor channel, see section 5.2.

I have attempted to define all other terms at first use and the document contains a hyper-linked glossary, if the term is idiosyncratic or specialised it should have an entry.

[1]The signals are labeled TREADY, TVALID and TLAST in the specification while the source is called master and the sink called slave.

Sequences

An ADC quantises a continuous time-dependant voltage in both voltage and time to produce a *sequence* of integers representing the voltage at a discrete point in time. The integers are called samples and a sample is added to the sequence each tick of the sample clock. Sequences are typeset $\mathtt{y}_{sub}$ where *sub* may be omitted. Samples in a sequence are indexed by discrete time, $\mathtt{y}[n]$ is the n^{th} sample of $\mathtt{y}$.

I use the term *signal*, where possible, to specify a 1 bit binary or Boolean sequence such as a crossing signal. Crossings signals for sequence $\mathtt{y}$ are denoted $\mathtt{y}^{\pm}_{thresh}$ where $+$ $(-)$ indicates the rising (falling) crossing signal for the threshold *thresh*. I use *after* to refer to the clock cycle following a crossing. I'll abuse notation a little and use $\mathtt{y}[\mathtt{w}^{\pm}_{thresh}]$ to indicate the sample of $\mathtt{y}$ *captured* at the crossing of *thresh* by $\mathtt{w}$.

Crossings are strict, a rising crossing signal for sequence $\mathtt{y}$ is *true* at the first time where $\mathtt{y}$ is strictly greater than the threshold since it was last strictly below the threshold, conversely for a falling crossing. For example, the sequence $\{-1, 0, 0, 0, 1\}$ has a rising zero crossing at the value 1 but $\{1, 0, 0, 0, 1\}$ has no zero crossings.

The notation $\mathtt{y}|^{\pm}_{thresh}$ refers to the index of *previous* crossing of *thresh* by $\mathtt{y}$. At time n in the sequence $\mathtt{y}$, $\mathtt{y}|^{\pm}_{thresh} = c$ where c is the largest index such that $c < n$ and $\mathtt{y}^{\pm}_{thresh}[c]$ is *true*. The sample at the crossing is $\mathtt{y}[\mathtt{y}|^{\pm}_{thresh}]$. The difference between $\mathtt{y}[\mathtt{w}|^{\pm}_{thresh}]$ and $\mathtt{y}[\mathtt{w}^{\pm}_{thresh}]$ is that when $\mathtt{w}^{\pm}_{thresh}$ is *true*, ie at a crossing, $\mathtt{y}[\mathtt{w}|^{\pm}_{thresh}]$ is the sample at the *previous* crossing while $\mathtt{y}[\mathtt{w}^{\pm}_{thresh}]$ is the sample at *this* crossing. When $\mathtt{w}^{\pm}_{thresh}$ is *false* $\mathtt{y}[\mathtt{w}|^{\pm}_{thresh}]$ and $\mathtt{y}[\mathtt{w}^{\pm}_{thresh}]$ are the same sample.

The key sequences in the design are:

`raw` is the raw detection sequence captured by an ADC.

`f` is the filtered detection sequence, `raw` after processing by a low-pass digital filter.

`s` is the slope sequence, `f` after processing by a digital filter configured as a differentiator.

The `f` and `s` sequences are used to identify *rises* and *pulses*. Rises are sub-sequences of `f` that *start* at a rising zero crossing of `s` and *end* at falling zero crossing of `s`. I use terms like *after* the rise or *after* the end of the rise to refer to the clock cycle after the falling zero crossing of `s`. The zero crossings can be considered to be at the local minima and maxima of `f`. It should be noted that these local minima and maxima are not necessarily at the locally extreme values of `f` as the zero crossings and the differentiator are not exact. A *first rise* is a rise that starts below `pulse_threshold`. A rise is a detection when at the end of the rise `f` is above the `pulse_threshold` setting and during the rise `s` crossed the `slope_threshold`. Detections generate an event packet. A *pulse* is a sub-sequence of `f` that starts at the start of a valid first rise and ends at the next falling crossing of `pulse_threshold` by `f`. Pulses may contain more that one valid rise and when they do are called a *piled-up* pulse.

Fixed point binary numbers

I use the notation $w.f$ bit to indicate a fixed point precision, w is the width or total number of bits in the fixed point number and f is the number of bits in the fractional part. Numbering bits starting with 0 for the least significant bit, $\mathrm{bit}[f] = 2^0$, $\mathrm{bit}[f-1] = 2^{-1}$ and $\mathrm{bit}[f+1] = 2^1$ etc.

other

All usages of *time* refer to discrete time measured by the sample clock, time fields and registers are integers counting clock pulses. *Host* refers to the computer communicating with the FPGA.

5.2 The streams

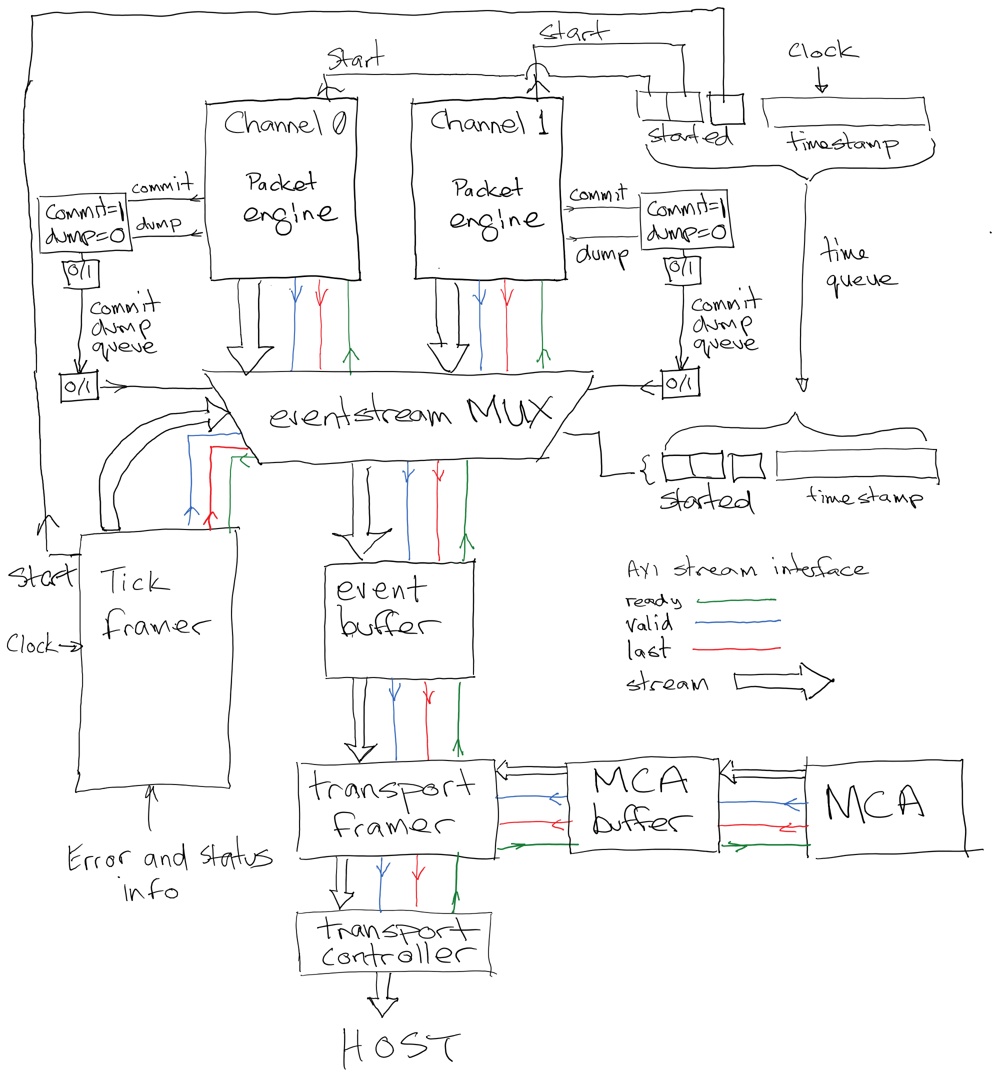


Figure 5.3: Stream pathways from source to the transport controller communicating with the host, see text for details.

Each channel processes the digitised TES output, the sequence $\texttt{raw}_{in}$, in a measurement pipeline; first performing a baseline correction, centering the sequences noise band on zero; then applying a two stage FIR digital filter. The first stage is configured as a low-pass filter to smooth `raw` and produces the filtered detection sequence `f`, while the second stage differentiates `f` to produce the slope sequence `s`. The measurement subsystem uses `raw`, `f` and `s` to generate measurement sequences which a *packet engine* (section 5.4) uses to produce the eventstream. See chapter 6 for detailed description of the measurement pipeline.

The eventstreams generated by the packet engines in each channel are merged by the eventstream MUX (section 5.3). The MUX timestamps each packet with the relative time since the previous event and the timestamp saturates at 16 bits. The MUX also periodically inserts `tick` packets (subsection B.3.1) that contain the full 64 bit system time and processor status information, a `tick` is considered and event in the relative timestamp calculation so selecting an appropriate period using the `tick_period` register acquires a continuous time resolved detection record.

5.2.1 Stream transport

The merged eventstream then passes through the *event buffer* to the *transport framer* which prepares packets for transport to the host via the *transport controller* for the communication channel. *Back pressure* from saturation of the communication channel travels up the event stream though the AXI ready signals (Drawing 5.3). When the communication channel saturates the transport controller signals it is *not* ready to the transport framer and frames begin to accumulate in its frame memory, when the frame memory is full the framer signals it is not ready. Similarly, not ready propagates upstream to the packet engines in each channel, but when their memory fills they begin dumping events. Each `tick` packet contains a count of total packets lost and flags indicating which channels dumped packets since the previous `tick`.

Event packets fall into two classes (section 5.4); multiple short packets can fit into a single transport frame in the form of an array; while *trace* packets that contain a sequence record may extend over multiple frames each containing a *fragment*. The transport framer guarantees the payload of short packet frames are array addressable by the host by ensuring that each frame contains complete packets of identical type and size. If two channels request different event packet types then no single frame will contain event packets from both channels. Trace fragments a always transported in sequential frames. Additionally the framer always puts a `tick` in a separate frame even when there are consecutive ticks with no intervening events.

Packets are designed for efficient access by ordering fields to optimise alignment in the host's memory. The transport framer also arbitrates between the eventstream and the *MCAstream* carrying histograms generated by the MCA (subsection A.1.3). Arbitration favours the eventstream as back pressure at the MCA causes the current histogram to accumulate for another tick and no information is lost.

5.3 The eventstream multiplexer

Individual channel eventstreams are merged in the eventstream MUX which preserves temporal order and inserts the relative timestamp. Preserving temporal order reduces the burden on the host by removing buffering and reordering required in order to establish coincidence or reconstruct a fully time-resolved detection record. Out of order event packets occur because some packets take longer to construct than others which can occur even for packets of the same type. Consider a three photon detection in one channel closely followed by a single photon detection in another, the single photon event packet is available before the three photon event packet because the TES takes less time to recover from the one photon absorption.

Temporal order is maintained using queues, a *time queue* common to all channels and a 1 bit *commit-dump queue* for each channel, see Drawing 5.3. Each packet engine directs the process through three signals

`start` is asserted at the at the point the time stamp for this event should be applied.

`commit` is asserted when a packet is complete and committed to the packet engine framer.

`dump` is asserted when a packet was started but the detection pulse did not meet requirements and was not committed to the packet engine framer, eg when `area` is less than `area_threshold`. `dump` is also asserted when the framer's internal RAM fills while creating a packet.

At regular intervals, set by the `tick_period` register, the tick framer commits a `tick` packet and asserts `start`. The `tick` packet contains a 64 bit timestamp and the error/status information accumulated since the previous `tick`.

When the time queue receives a `start` signal from any of the channels or the tick framer a 64 bit timestamp and a vector of `start` bits is enqueued. The the bit vector indicates which channels simultaneously asserted `start` and if there is a tick at the timestamp. When a packet engine asserts `commit` or `dump` a bit is enqueued to the commit-dump queue for the corresponding channel with 0 representing `dump` and 1 `commit`.

The MUX processes the head of the time queue by checking whether any started channel's commit-dump queue is not empty which indicates the channel has completed. When a channel has completed with `commit` the MUX connects the eventstream from the packet engine for the channel to the MUX's output eventstream until the last transfer of the packet in the packet engine completes. The MUX then marks that channel as *handled.* When a channel completes with `dump` it is marked as handled with no other action. When all channels are handled and the tick bit in the started vector is set the stream from the `tick` framer is connected to the MUX output until the last transfer of the `tick` packet completes.

Every type of event packet has a common structure for bytes 4 to 7 which are in the first transfer of a packet

byte	0	1	2	3	4	5	6	7
0					eflags		time	

This allows the MUX to insert the relative timestamp in the `time` field as well as basic coincidence information. The `new` bit in `eflags` field is set when `time` is greater than the `window` setting.

5.4 The packet engine and event packets

Each packet engine contains a framer that assembles the event packets that form the eventstream for the channel. Framer memory is 64 bits wide and consists of 4 16 bit *chunks* with individual write enables. The write enables allow a field to be written in some chunks(s) without overwriting data already in other chunks. Finite state machines (FSMs) in each packet engine fill packet fields when the information becomes available and send the `start`, `commit` and `dump` signals to the eventstream MUX (section 5.3). On commit the framers internal write port address is advanced by the length of the packet and the committed packet is available in the packet engine's eventstream. When a packet is dumped the framer overwrites the old data. The packet engines are controlled by registers in the `event` group and the `packet` and `trace` registers set the type of event packet generated. Unless `packet`=`trace` the `trace` setting is ignored.

5.4.1 Event flags

Each event packet has a 2 byte `eflags` field containing register settings used to create the packet and additional information about the event.

first byte

bit	7	4	3	2	0
	rise_number		rel2min	channel	

second byte

bit	7	6	5	4	3	2	1	0
	trace		height		packet		tick	new

where the fields are:

`rise_number`: is set to 0 at the start of a pulse and incremented *after* the end of each rise, ie `rise_number`=0 for first rises.

`rel2min`: the `rel2min` setting that controlled CFD behaviour,see Equation 6.1.

`channel`: the channel that processed the event.

`timing`: the `timing` setting that controlled the point in the rises that the timestamps refer to, for packets carrying traces this also the *trigger* point, see Equation 6.4.

`height`: the `height` setting that determined how the `height` field was calculated, see Equation 6.5.

`packet`: the `packet` setting, ie the type of this event packet.

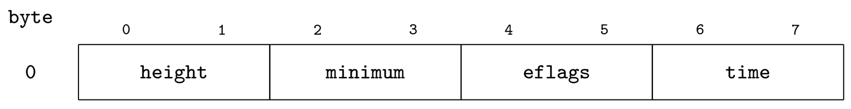
`tick`: *true* if this event packet is a tick. When true the values of `packet` and `trace` are undefined.

`new`: *true* if this event marks the start of a new coincidence window (see section 5.3).

5.4.2 Short event packets

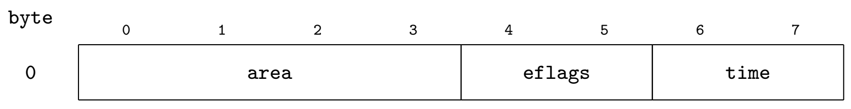
The rise event packet

The `rise` event packet captures individual valid rises. The piled-up pulse in Figure 6.2 would generate two rise packets the first with `rise_number`=0 and the second with `rise_number`=1.



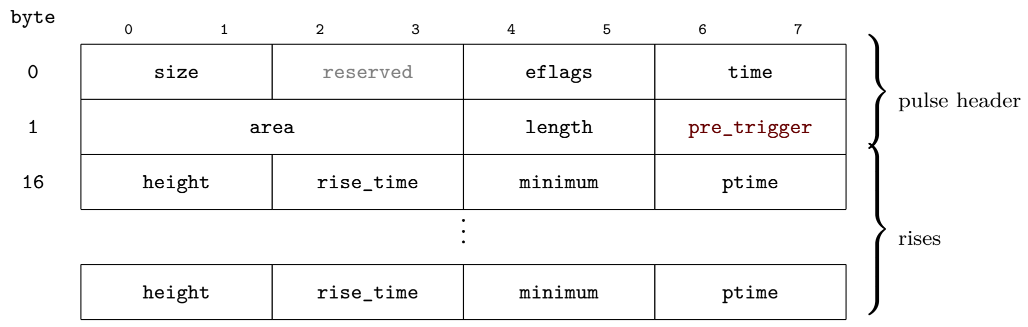
The area event packet

The `area` packet captures the area of a valid pulse The piled-up pulse in Figure 6.2 would generate a single area packet with `rise_number`=2 indicating that the pulse has two rises.



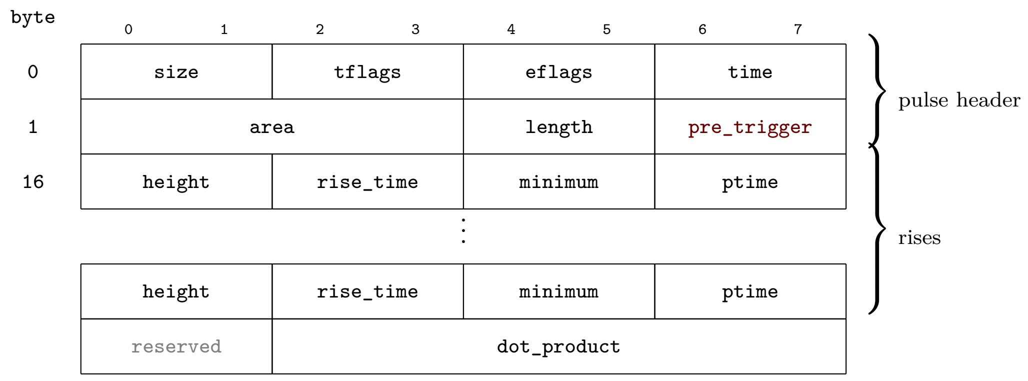
The pulse event packet

The `pulse` packet combines fields from the `rise` and `area` packets. The pulse packet contains a fixed number of slots, controlled by the `max_rises` setting, to record rises. The `rise_number` field indicates how many of rise slots were filled.



The dot product event packet

Setting packet=trace and trace=dot_product adds a dot product field to a `pulse` packet. The packet does not carry a sequence record but uses the trace subsystem to calculate the mathematical dot product of a trace with a *template* created by averaging over many pulses, see section 5.4.3 for details.



5.4.3 Event packets carrying traces

All packets in this section use the packet=trace setting and carry a record of a sequence called a *trace*. The following registers control how the sequence is recorded

sequence selects the sequence to record.

stride sets the number of samples to skip in the sequence between each recorded sample.
For example, setting stride=1 records every second sample of sequence.

length sets the number of samples to record.

pre_trigger sets the number of samples to record prior to the timing point.

Trace flags

Each trace packet contains a two byte `tflags` field with details of the trace and settings used during capture.

first byte

bit	7	6	5	4 … 0
	resv	Mrise	Mpulse	stride

second byte

bit	7 … 6	5 … 4	3 … 0
	trace	sequence	offset

where

`Mrise`: is *true* if more than one rise was detected in the trace.

`Mpulse`: is *true* if more than one pulse was detected in the trace.

`stride`: the `stride` setting used.

`trace`: the `trace` setting, ie the type of this trace.

`sequence`: the `sequence` setting, ie which sequence is recorded in the trace.

`offset`: the offset from the start of the packet to the first sample in the trace. The offset in bytes is 8*`offset`.

The average trace template

Setting `trace` to `average` uses the *dot product* entity to accumulate an average of the selected `sequence` over 2^{16} traces. The number of pulses averaged over is $2^{\texttt{AVERAGE\_N}}$ where `AVERAGE_N` is a VHDL generic and can be changed. Any traces that contain multiple rises are not included in the averaging process. The average template can be used in techniques that reduce the uncertainty in photon number analysing traces, see section 4.1.

The average template in retained in memory and can be used with `trace`=`dot_product` or `dot_product_trace` event packets to return the dot product of a trace with the stored template. The result is returned in the `dot_product` field

$$\texttt{dot\_product} = \sum_{n=0}^{\texttt{length}-1} \texttt{a}[n]\texttt{y}[n], \tag{5.1}$$

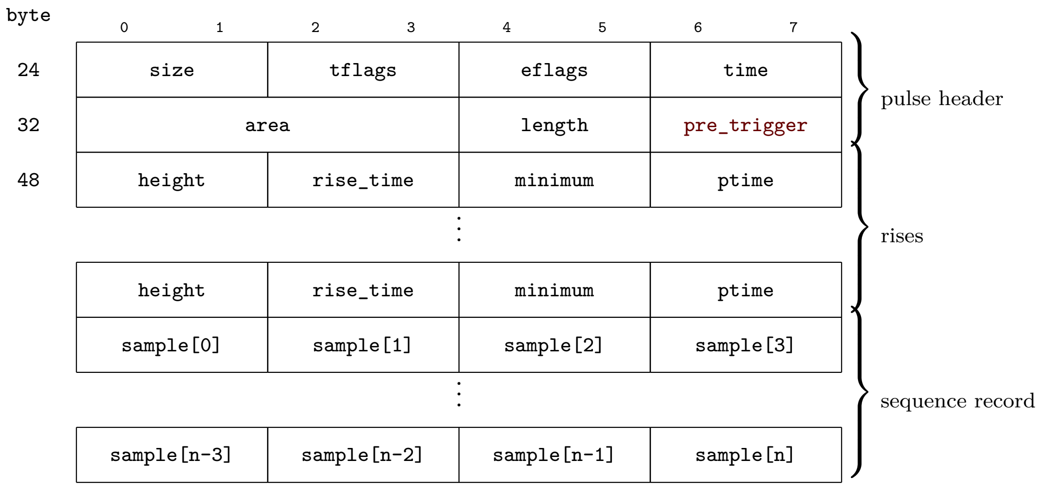
where `length` sets the length of the trace, `a` is the stored average template and `y` is selected with the `sequence` setting.

byte	0 – 1	2 – 3	4 – 5	6 – 7	
0	size	tflags	eflags	time	Average header
8	multi_pulse		multi_peak		Average header
16	sample[0]	sample[1]	sample[2]	sample[3]	Average sequence
	⋮	⋮	⋮	⋮	Average sequence
	sample[n-3]	sample[n-2]	sample[n-1]	sample[n]	Average sequence

The fields `multi_pulse` and `multi_peak` count the number of traces rejected because multiple `pulse`$_{start}$ or `rise`$_{start}$ signals occurred while capturing the sequence.

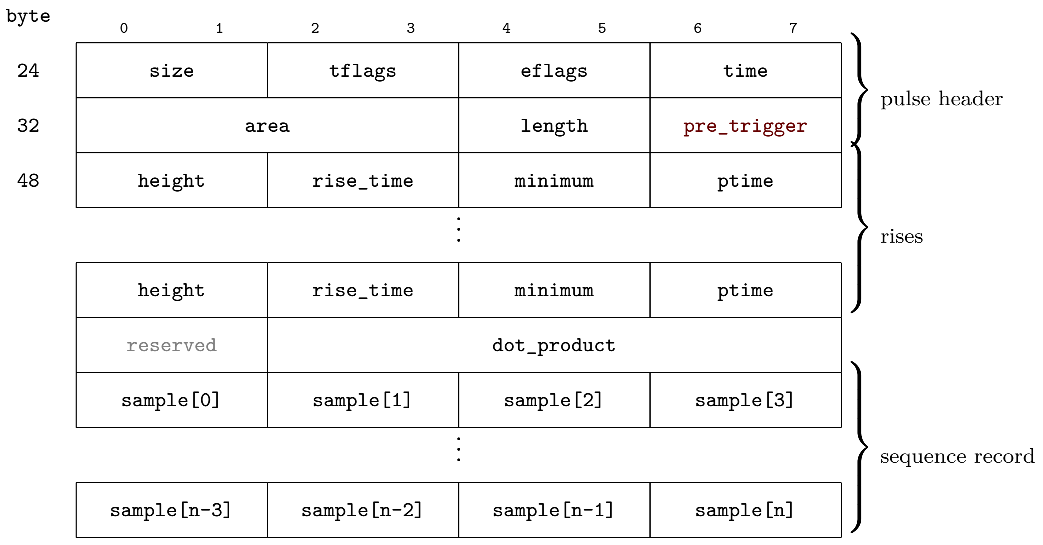
The pulse trace

Setting `trace`=`pulse` adds a sequence record to a `pulse` event packet.



The dot product trace

Setting `trace`=`dot_product_trace` adds a sequence record to a `dot_product` event packet.



5.4.4 Event packet fields

Timer sequences

Timer sequences are 16 bit and saturate at $2^{16} - 1$.

$\texttt{pulse}_{timer}$ is set to 1 after each $\texttt{pulse}_{start}$ otherwise it is incremented each tick of the sample clock. It counts time since the start of the pulse.

$\texttt{rise}_{timer}$ is set to 1 after each $\texttt{stamp}_{rise}$ otherwise it is incremented each tick of the sample clock. It counts time since the timestamp in the `time` field.

Fields

`size`: the size of the event packet in bytes.

`tflags`: flags relating to traces, see section 5.4.3.

`eflags`: flags relating to all events, see subsection 5.4.1.

`time`: the relative timestamp for this event, see section 5.3.

`area`: ($\texttt{pulse\_area}[\texttt{f}^-_{pulse}]$) the area of the pulse above `pulse_threshold`, see Equation 6.6. Note that event packets with an `area` field will be dumped when $\texttt{pulse\_area}[\texttt{f}^-_{pulse}]$ is less than or equal to the `area_threshold` setting.

`length`: ($\texttt{pulse}_{timer}[\texttt{f}^-_{pulse}]$) the length of the pulse ie the time between $\texttt{f}^+_{pulse}$ and $\texttt{f}^-_{pulse}$.

`pre_trigger`: the `pre_trigger` setting used to capture the trace. Undefined for event packets without a trace.

`height`: ($\texttt{height}[\texttt{s}^-_0]$) the height of the rise. How the measurement is made is dependent on the `height` setting, see Equation 6.5.

`rise_time`: the time from the `timing` point to the measurement in the `height` field, `rise_time` is dependant on the `height` setting.

$$\texttt{rise\_time} = \begin{cases} \texttt{f}[\texttt{s}^-_0] & \texttt{height} = \texttt{peak} \\ \texttt{f}[\texttt{f}^+_{high}] & \texttt{height} = \texttt{cfd\_high} \text{ or } \texttt{cfd\_height} \\ \texttt{f}[\texttt{s}^+_{max}] & \texttt{height} = \texttt{max\_slope} \end{cases}$$

`minimum`: ($\texttt{minima}[\texttt{s}^+_0]$) the effective value of `f` at the start of the rise, see Equation 6.3.

`ptime`: used in `pulse` based packets which can measure multiple rises, `ptime` is the $\texttt{pulse}_{timer}$ value when $\texttt{stamp}_{rise}$ was *true* for the rise. The timestamp for each rise is $\texttt{time} + (\texttt{ptime} - \texttt{ptime0})$, where `ptime0` is the value of the `ptime` field for the first rise.

`dot_product`: the dot product of the trace with the average template stored in RAM, see section 5.4.3

Chapter 6

The measurement pipeline

6.1 Input stage

The input stage is controlled by registers in the `channel` group.

The input multiplexer, controlled by the `adc_select` register, can connect any ADC output to the input of any channel pipeline. Before entering the pipeline the selected ADC sequence can be delayed by `delay` clock periods to temporally align it with other channels. The processor expects *rising* detection pulses the `invert` register can be used to multiply the selected ADC sequence by -1 if required. The output sequence for the stage is:

$\mathtt{raw}_{in}$ (14 bit signed) Delayed and polarity corrected ADC output selected to drive this processing channel.

6.2 Baseline correction

Pulse measurements (Figure 2.1) are relative to a baseline assigned the value 0. The unipolar nature of the pulses makes averaging, including AC coupling, inaccurate and dependant on the optical input power. Registers in the `baseline` group control a minimal implementation of a dual buffered MCA that tracks the mode of the $\mathtt{raw}_{in}$ sequence which estimates the baseline value. The baseline MCA can only track the mode when it is within 2040 of 0, the `offset` register is used to bring the mode close to 0. The input for the baseline MCA is $\mathtt{raw}_{DC} = \mathtt{raw}_{in} - \mathtt{offset}$.

In order to track changes in the distribution of $\mathtt{raw}_{DC}$ the buffers must be regularly cleared, one buffer is cleared every `time_constant` clocks and the buffer that is cleared is alternated. Each buffer exposes the value of the bin with the highest frequency (`mode`) and its frequency (`count`). A buffers estimate of the baseline ($\mathtt{raw}_{bc}$) is updated to `mode` whenever count changes and `count` $>$ `count_threshold`, if `new_only` is *true* then `mode` must also have changed from it previous value for the update to occur. The baseline estimate $\mathtt{raw}_{BC}$ is the average of the individual $\mathtt{raw}_{bc}$ from each buffer.

The 14.0 bit $\texttt{raw}_{BC}$ is sign extended to 16 bit the left shifted 3 bits to form the 16.3 bit output sequence:

`raw` The raw baseline corrected TES detection signal.

6.3 Digital filtering

A useful property of a linear time invariant system is that its behaviour is completely described by its response to an impulse. The system output given *any* input is the convolution input and the systems impulse response. In signal processing this property is exploited to create filters by engineering an impulse response, called the filter kernel, that describes an system with some desired frequency response. linear time invariant system theory is well developed and underpins DSP. Many algorithms and software solutions exist for the design and analysis of digital filters.

For discrete time FIR filters convolution is a sum

$$y[n] = \sum_{k=0}^{N-1} h_k x[n-k]$$

where $x[n]$ is the input sequence, $y[n]$ the output sequence and h_k the coefficients of a kernel of length N, these types of filters have a linear phase response and a group delay of $K = \frac{N-1}{2}$ samples.

Many FPGAs contain multiply and accumulate resources designed to multiply the signal by the filter coefficients and accumulate the partial products of the convolution which makes implementation in hardware relatively straight forward. Xilinx calls these resources DSP48E and there are columns og these hard cores through out the FPGA. Fast routing exists between each DSP48E in a column so filters that can be implemented within a single column can be clocked at higher frequencies which makes timing closure easier to achieve. The kernel lengths have been chosen to take advantage of this.

6.3.1 DSP stage

The current implementation uses two chained FIR filter stages with kernels that can be reconfigured from the host without reconfiguring the FPGA.

The first stage FIR filter is configured as a low-pass filter to remove high frequency noise from `raw`, has $N = 142$ and expects a symmetric kernel consisting of 24.28 bit coefficients. The raw first stage 48.31 bit output sequence is rounded to 16.3 bits (convergent rounding to even) and on overflow or underflow saturates. This rounded sequence forms input into the second stage and a copy is delayed to compensate for the second stage group delay to become the first stage output sequence.

The second FIR filter stage has $N = 23$ and expects an anti-symmetric kernel with 25.25 bit coefficients configured to differentiate its input and its raw 46.28 bit output sequence rounded to 16.3 bits (convergent rounding to even) with overflow and underflow leading to saturation.

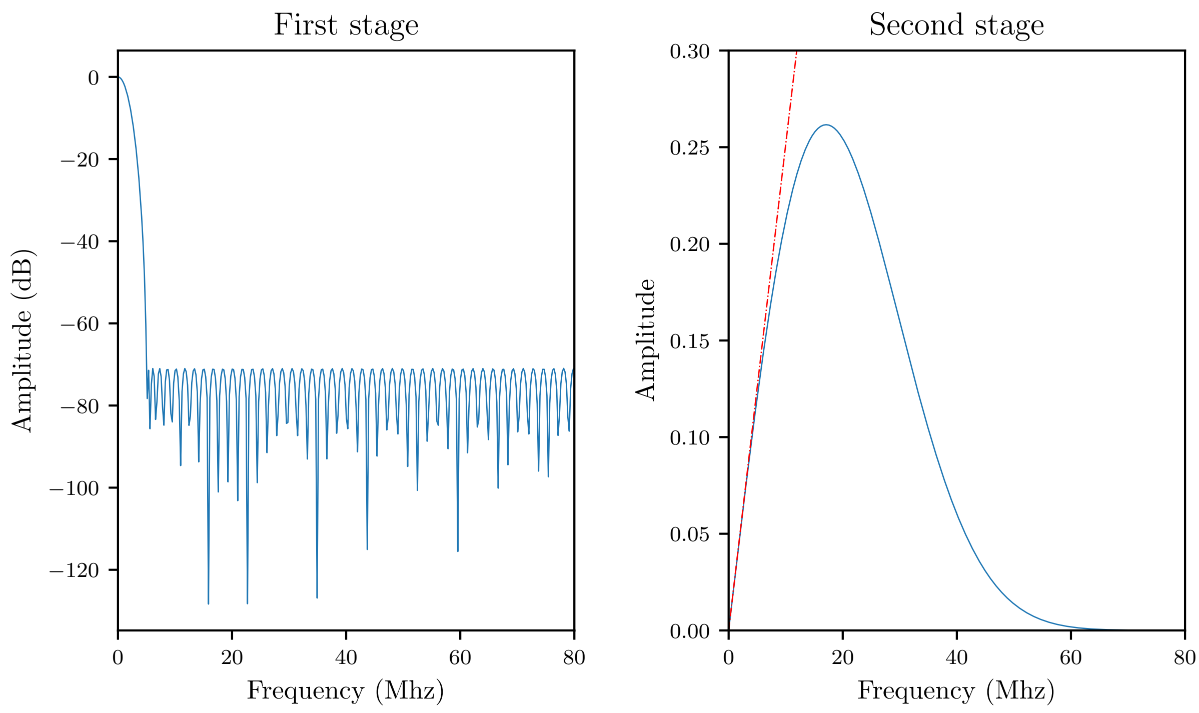


Figure 6.1: Default filter amplitude responses used during the preliminary testing, see chapter 3: The first stage is an equi-ripple low pass filter with a 3 db cutoff frequency of ~1.5 MHz. The second stage is configured as a smooth noise robust differentiator [Hol08] designed for a smooth transition between the ideal differentiator response (red dashed line) at low frequencies and an open circuit response (x-axis) at high frequencies to suppress noise.

The amplitude responses of both stages configured with the default kernels are shown in Figure 6.1. The stage outputs include a copy of `raw`$_{BL}$ delayed to match the group delay of both filters, the first stage output is also delayed to compensate for the second stage delay, so that the output sequences have the correct temporal alignment. The output sequences are:

`raw` The raw TES detection sequence.

`f` The filtered detection sequence. This is `raw` after processing by the the first stage filter.

`s` The slope of `f`. This is `f` after processing by the second stage filter.

6.4 Measurement

Measurement is controlled by registers in the event and cfd groups. The measurement subsystem processes f and s in two phases. Phase one identifies rises in f using zero crossing of s and generates the $\texttt{first}_{rise}$, armed and above signals (Figure 6.2). Phase one also generates the constant fraction sequences $\texttt{cf}_{low}$, $\texttt{cf}_{high}$ and the maximum slope sequence $\texttt{s}_{max}$. At the end of each rise in phase one sequences and signals are *captured* and added to a FIFO queue (Figure 6.3).

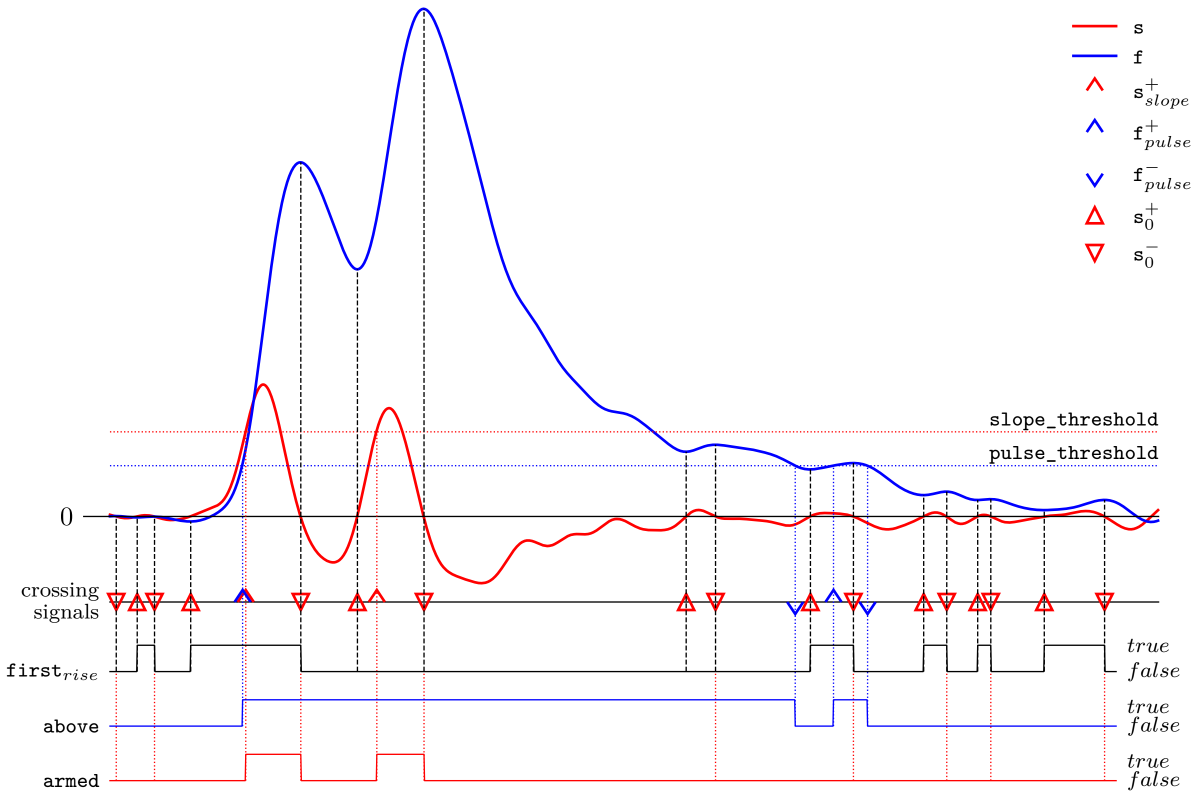


Figure 6.2: Rises: Zero crossings of s are used to identify rises in f. A *rise* is sub-sequence of f that starts at $\texttt{s}^{+}_{0}$ and ends at the next $\texttt{s}^{-}_{0}$. A $\texttt{first}_{rise}$ is a rise that starts below pulse_threshold. The detector is armed when s crosses slope_threshold and is reset *after* the end of the rise. The above sequence is *true* when f is above pulse_threshold. pulse_threshold. Markers on the crossing signals line indicate where the crossings are true.

Phase two acts on delayed versions of f and s and dequeues the values captured in phase one at the start of the delayed rises. This provides phase two with *look-ahead* information about the whole rise at its start. Look-ahead allows *level* crossings in phase two based on CFD thresholds that are constant fraction of the rise height and a level crossing at the maximum slope during the rise. A level crossing is *true* at the first time in a rise that the sequence is greater than or equal to the threshold. With Look-ahead phase two can determine if a rise is a *detection* at its start and prepare the packet engine FSMs to fill fields in the event packet.

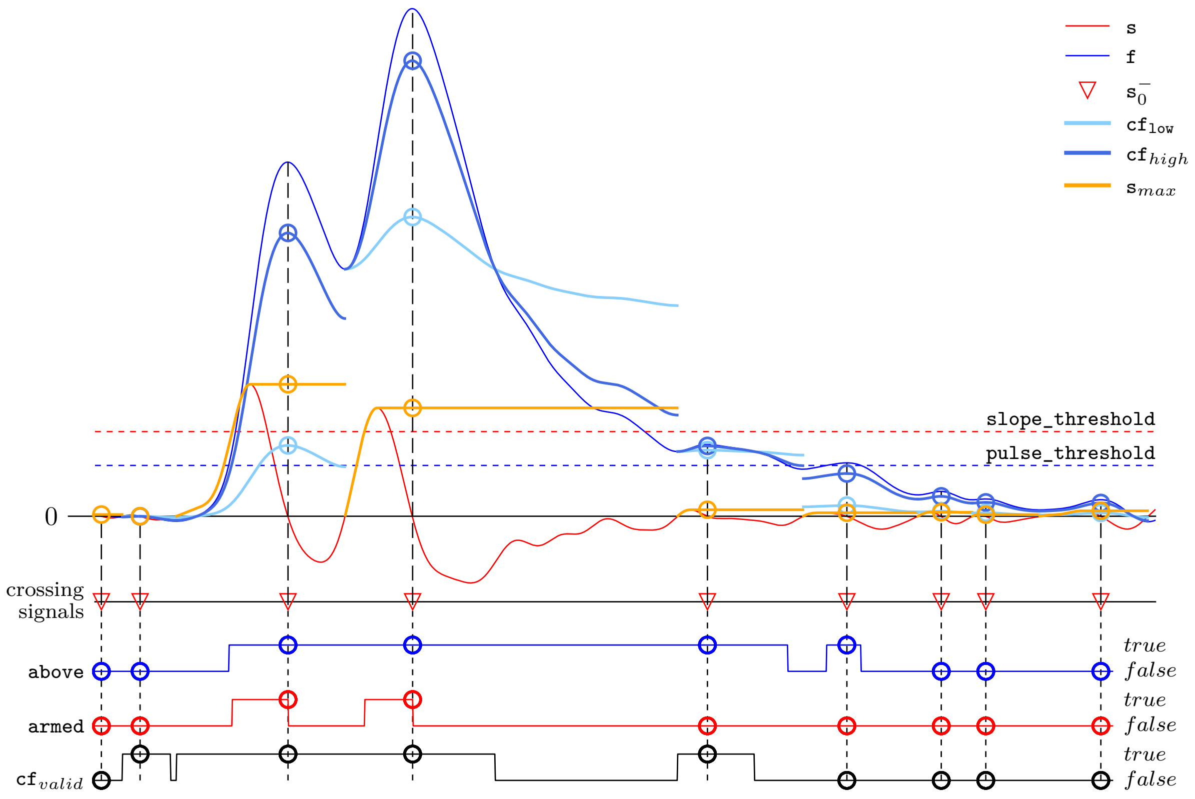


Figure 6.3: Phase one measurements: Two constant fraction sequences, $\mathtt{cf}_{high}$ and $\mathtt{cf}_{low}$, and a maximum slope sequence, $\mathtt{s}_{max}$ are generated (see text). At the end of each rise values are captured (circle markers) and input into a queue to be used in phase two (Figure 6.4).

Generation of the constant fraction sequences, $\mathtt{cf}_{low}$ and $\mathtt{cf}_{high}$, is controlled by the `fraction` and `rel2min` settings.

$$\mathtt{cf}_{high}[n] = (\mathtt{f}[n] - \mathtt{minima}[n]) \times (1 - \mathtt{fraction}) + \mathtt{minima}[n] \tag{6.1}$$

$$\mathtt{cf}_{low}[n] = (\mathtt{f}[n] - \mathtt{minima}[n]) \times \mathtt{fraction} + \mathtt{minima}[n] \tag{6.2}$$

where

$$\mathtt{minima}[n] = \begin{cases} 0 & \text{when } \mathtt{first}_{rise}[n] \text{ and not } \mathtt{rel2min} \\ \mathtt{f}[\mathtt{s}_0^+] & \text{otherwise} \end{cases} \tag{6.3}$$

When $\mathtt{first}_{rise}$ is *true* and `rel2min` is *false* a *minima* of 0 is used in the constant fraction calculation this can generate $\mathtt{cf}_{low}$ sequences that are less than `f` for the entire rise, producing phase two thresholds that can never be crossed. The signal $\mathtt{cf}_{valid}[n] = \mathtt{cf}_{low}[n] \geq \mathtt{f}[n]$ monitors this condition. When `rel2min` is *true* the true minima, $\mathtt{f}[\mathtt{s}_0^+]$, is used and the phase two thresholds can always be crossed.
The maximum slope sequence is

$$\mathtt{s}_{max}[n] = \begin{cases} \mathtt{s}[n] & \mathtt{s}_0^{\pm}[n] \\ \max(\mathtt{s}[n], \mathtt{s}_{max}[n-1]) & \text{otherwise} \end{cases}$$

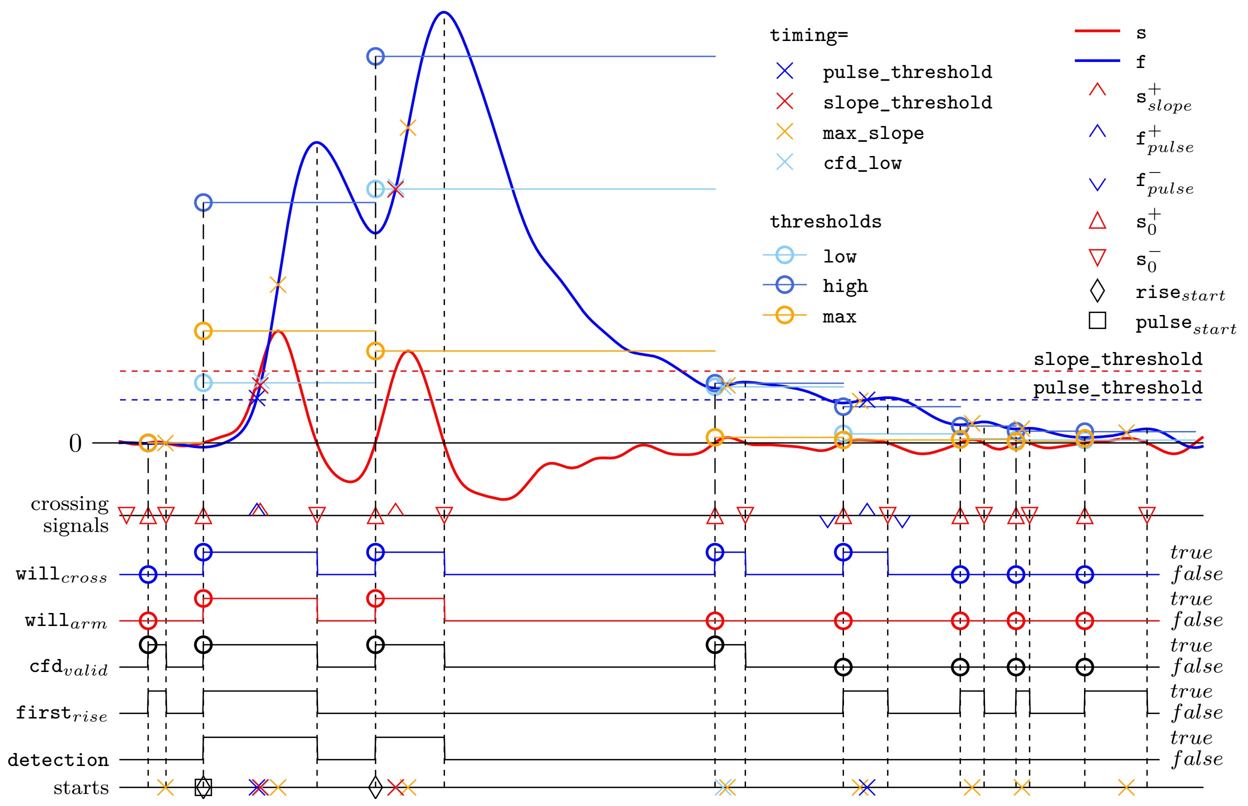


Figure 6.4: Measurement phase two: Phase two operated on delayed versions of f and s. Values captured in phase one are de-queued at the start of each delayed rise (circle markers) creating the $will_{cross}$, $will_{arm}$ and cfd_{valid} signals and the low, high and max thresholds. The point during a rise that is timestamped is determined by the timing setting and the various timing points appear as x's on the starts line. A rise is a detection and can generate an event packet when at its start both $will_{cross}$ and $will_{arm}$ are *true*, if timing=cfd_low cfd_{valid} must also be *true*. The $rise_{start}$ signal is generated at the start of each detection and the $pulse_{start}$ signals the start of a detection that is also a first rise.

Phase two operates on delayed versions of f, s the delay time is set by the VHDL generic CFD_DELAY. At the start of each rise in phase two values captured and queued at the end of the rise in phase one are de-queued. When the time between the start and end of a rise exceeds CFD_DELAY the values have not been captured by phase one when phase two tries to de-queue them. This generates a cfd_overrun which is flagged in the next tick and counted as a lost event.
The following de-queued signals are *false* outside of a rise:

$will_{cross}$ is the value above will have at the end of the rise.
$will_{arm}$ is the value armed will have at the end of the rise.
cfd_{valid} is the value cf_{valid} will have at the end of the rise.

The de-queued thresholds only change at the start of a rise:

low is the value cf_{low} will have at the end of the rise and is the low CFD threshold with level crossing signal f^+_{low}.

high is the value $\texttt{cf}_{high}$ will have at the end of the rise and is the high CFD threshold with level crossing signal $\texttt{f}^{+}_{high}$.

max is the maximum slope during the rise and the level crossing signal is $\texttt{s}^{+}_{max}$.

The timing setting selects which signal drives $\texttt{rise}_{stamp}$ indicating when the timestamp should be applied during the rise (see Figure 6.4).

$$\texttt{rise}_{stamp} = \begin{cases} \begin{cases} \texttt{f}^{+}_{pulse} & \text{when } \texttt{first}_{rise} \\ \texttt{f}^{+}_{low} & \text{otherwise} \end{cases} & \texttt{timing} = \texttt{pulse\_threshold} \\ \texttt{s}^{+}_{slope} & \texttt{timing} = \texttt{slope\_threshold} \\ \texttt{s}^{+}_{max} & \texttt{timing} = \texttt{max\_slope} \\ \texttt{f}^{+}_{low} & \texttt{timing} = \texttt{cfd\_low} \end{cases} \tag{6.4}$$

timing=pulse_threshold is a special case because the threshold can only be crossed by rises that start below it, ie a first rise.

The height sequence is captured by the packet engine as the height field in event packets

$$\texttt{height} = \begin{cases} \texttt{f} & \texttt{height} = \texttt{peak} \\ \texttt{high} & \texttt{height} = \texttt{cfd\_high} \\ \texttt{high} - \texttt{low} & \texttt{height} = \texttt{cfd\_height} \\ \texttt{max} & \texttt{height} = \texttt{max\_slope} \end{cases} \tag{6.5}$$

The pulse_area sequence is captured by the packet engine as the area of f above pulse_threshold in the area field.

$$\texttt{pulse\_area}[n] = \sum_{m=\texttt{f}|^{+}_{pulse}}^{n-1} \texttt{f}[m] - \texttt{pulse\_threshold} \tag{6.6}$$

Two measurements are performed on f and s that don't appear in an event packet but can be captured by the MCA, the area and extreme value of the sequences between zero crossings.

$$\texttt{y}_{area}[n] = \sum_{m=\texttt{y}|^{\pm}_{0}}^{n-1} \texttt{y}[m]$$

$$\texttt{y}_{extrema}[n] = \begin{cases} \texttt{y}[n] & \texttt{y}^{\pm}_{0}[n] \\ \text{extreme}(\texttt{y}[n], \texttt{y}_{extrema}[n-1]) & \text{otherwise} \end{cases}$$

where

$$\text{extreme}(a,b) = \begin{cases} \max(a,b) & \texttt{y}|^{+}_{0} > \texttt{y}|^{-}_{0} \\ \min(a,b) & \text{otherwise} \end{cases}$$

and `y` is either `f` or `s`, creating $\texttt{f}_{area}$, $\texttt{f}_{extrema}$, $\texttt{s}_{area}$ and $\texttt{s}_{extrema}$ sequences available for input into the MCA. When triggered at the appropriate zero crossings the MCA outputs the distribution of areas or extreme values of the selected sequence between zero crossings.

Appendix A

Registers

Communication with the FPGA registers is via a RS232 serial connection. The low level serial protocol is described in section A.2 and the underlying FPGA address map is presented in section A.3.

A.1 Python Interface

High level register access is provided by a client written in Python. This client can either be used remotely, by connecting to the ZeroMQ socket for the register server running on the FPGA host computer, or directly from the host via a serial port. The client communicates with the FPGA using the protocol described in section section A.2 to access the hardware address map in section section A.3.

To use the client it must first be imported from the `tes` package and instantiated

```
>>> from tes.registers import Registers
>>> r = Registers('server address')
```

the variable `r` now references an instance of the client object.

Registers are arranged in functional groups and `r.group.register` accesses a register in the group, with the exception of registers in the `global` group which are accessed without a group name `r.register`.

All groups except the `global` and `mca` group support indexing in order access a register for a particular channel, both slicing and fancy indexing are supported and if the index is omitted *all* channels are referenced. The indexed groups also provide Pythons *iterable* interface allowing entire register groups to be accessed using dictionaries.

For example,

```
>>> r.event[0].pulse_threshold=1200
```

sets the event.pulse_threshold register from the event group for channel 0 to 1200 and

```
>>> r.event[0].pulse_threshold
1200
```

reads event.pulse_threshold for channel 0.

Omitting the index references *all* channels,

```
>>> r.event.pulse_threshold=23
>>> r.event.pulse_threshold
[23, 23, 23, 23, 23, 23]
```

first sets event.pulse_threshold for all channels to 23 then reads all channels returning a python list of register values. A list can also be used to to set registers with

```
r.event.pulse_threshold=[1, 2, 3, 4, 5, 6, 7, 8]
```

setting each channel to the corresponding value in the list. A subset of channels can be referenced using slicing or fancy indexing

```
>>> r.event[0:2].pulse_threshold=42 # slicing
>>> r.event[0,1].pulse_threshold=42 # fancy indexing
```

both set the event.pulse_threshold for channels 0 and 1 to 42, similarly

```
>>> r.event[0:2].pulse_threshold=[23, 42]
```

sets channel 0 to 23 and channel 1 to 42.

An entire group can be read as a dictionary and

```
>>> dict(r.event[0,1])
{'area_threshold': [1808, 10000],
 'enable': [False, False],
 'height': [<Height.peak_height: 0>, <Height.peak_height: 0>],
 'max_peaks': [0, 0],
 'packet': [<Event.peak: 0>, <Event.peak: 0>],
 'pulse_threshold': [23, 42],
 'slope_threshold': [512, 512],
 'timing': [<Timing.cfd_low: 2>, <Timing.cfd_low: 2>]}
```

returns a dictionary of register values from the `event` group for channels 0 and 1, this mechanism can be used to duplicate settings,

```
>>> r.event[1]=dict(r.event[0])
```

copies the settings for the `event` group from channel 0 to channel 1.

The client also provides an iterator over all registers and

```
>>> dict(r.all)
```

outputs a dictionary with an entry for every register. The `tes.register` module also contains two methods for loading and and saving register dictionaries as human readable YAML files

```
>>> from tes.registers import load,save
>>> save(dict(r.all), filename)
```

A.1.1 Notation

Register descriptions have the following format

`group.register` access, binary type:size (python type)
Register description.

Where access is either Read-only or strobe and if no access is given read-write is implied. A strobe ignores the data written to it but performs some action, for example see `update`. The binary type:size field refers to attributes of the underlying hardware register and links to it in the address map in section A.3. The python type for the register is given in the description and maybe a *enumerated type* that subclasses `tes.base.VhdlEnum` which is a subclass of `IntEnum`. The description of *enumerated types* has the following format:

0: `name0` Description.

1: `name1` Description.

2: ...

and the register can be set using the `int` value, the `name` as a string or an instance of a `VhdlEnum` subclass, reading the register always returns an instance of the `Vhdlenum` subclass. The python type field indicates the package path for importing the `VhdlEnum` subclass if required.

A.1.2 Global registers

The global group has no indexing.

`hdl_version` read-only, unsigned:24 bit (`str`)
: Returns a string containing the hexadecimal value of the short SHA-1 for the git commit of the HDL code.

`cpu_version` read-only, unsigned:32 bit (`str`)
: Returns a string containing the date and time the central CPU code was assembled as "day-month-year hour:minute".

`channel_count` read-only, unsigned:8 bit (`int`)
: Returns an int containing the number of processing channels instantiated in the FPGA.

`adc_chips` read-only, unsigned:8 bit (`int`)
: Returns an in containing the number of dual channel ADC chips attached attached to the FPGA.

`adc_enable` unsigned:8 bit (`int`)
: An integer that enables the ADC channels corresponding to the set bits, disabled channels are put in low power mode. See the `adc.enable` register to enable on a per channel basis.

`event_enable` unsigned:8 bit (`int`)
: An int event transmission for the channels corresponding to the set bits. See the `event.enable` register to enable on a per channel basis.

`tick_period` unsigned:32 bit (`int`)
: An int that sets the time (4ns clock pulses) between tick events.

`tick_latency` unsigned:32 bit (`int`)
: An int that contains the maximum time (4ns clock pulses) to wait after a tick before flushing the event buffer up to the next tick. This effectively sets a maximum latency for tick events in the output stream.

`window` unsigned:32 bit (`int`)
: An int setting the time window (4 ns clock pulses) used in setting the `new` bit in `eflags`

as an aid in determining coincidence between events.

`mtu` unsigned:32 bit (`int`)
: An int setting the maximum number of bytes in transmitted Ethernet frames (max 1496).

`ad9510_status` 1 bit (`bool`)
: Boolean controlling the status pin on the AD9510 clock generator chip on the FMC108 (see the AD9510 data sheet).

`vco_power_en` 1 bit (`bool`)
: Boolean controlling the power enable pin on the AD9510 clock generator chip (see AD9510 data sheet).

`fmc` read-only, 1 bit (`bool`)
: Boolean indicating that the FMC108 digitiser card is connected to the FPGA mezzanine connector (FMC) connector and recognised.

`fmc_power` read only, 1 bit (`bool`) Indicates FMC108 card is present and powered up.

`fmc_internal_clock` 1 bit (`bool`) The state of internal clock pin on the FMC108 (see the AD9510 datasheet).

`mmcm_locked` Read only, 1 bit (`bool`) Indicates FPGA multi-mode clock manager (MMCM) tile is locked to FMC108 clock from the first AD9510.

`iodelay_ready` Read only, 1 bit (`bool`) Indicates the FPGA IODELAY controller for inputs from ADCs is initialised.

A.1.3 MCA group

The MCA group has no indexing.

`mca.lowest_value` signed:32 bit (`int`)
: Values < `lowest_value` are placed in the underflow bin (bin[0]).

`mca.ticks` unsigned:32 bit (`int`) Sets how many ticks each histogram is accumulated over, see `tick_period`.

`mca.value` unsigned:4 bit (`tes.mca.Value`)
: Selects the MCA input sequence.

0: `disabled` Disables the MCA.

1: `f` (`f`) the filtered TES sequence.

2: `f_area` ($\mathtt{f}_{area}$) the area of `f` between zero crossings.

3: `f_extrema` ($\mathtt{f}_{extrema}$) the extreme value of `f` between zero crossings

4: `s` (`s`) the slope of `f`.

5: `s_area` ($\mathtt{s}_{area}$) the area of `s` between zero crossings.

6: `s_extrema` ($\mathtt{s}_{extrema}$) the extreme value of `s` between zero crossings.

7: `pulse_area` (`pulse_area`) the area of `f` above the `pulse_threshold` setting.

8: `raw` (`raw`) the raw unfiltered TES sequence.

9: cfd_high ($\mathtt{cfd}_{high}$) the high CFD threshold.

10: pulse_timer ($\mathtt{pulse}_{timer}$) the time since the rising zero crossing of s at the start of a pulse, see section 5.4.4.

11: rise_timer $\mathtt{rise}_{timer}$ the time since the timestamp was applied to a rise.

mca.trigger unsigned:4 bit (tes.mca.Trigger)

The sequence selected by value is sampled into the histogram when trigger and qualifier are both *true*.

0: disabled disables the MCA.

1: clock always trigger.

2: pulse_t_pos ($\mathtt{f}^{+}_{pulse}$) rising crossings of pulse_threshold by f.

3: pulse_t_neg ($\mathtt{f}^{-}_{pulse}$) falling crossings of pulse_threshold by f.

4: slope_t_pos ($\mathtt{s}^{+}_{slope}$) rising crossings of slope_threshold by s.

5: f_0xing ($\mathtt{f}^{\pm}_{0}$) zero crossing of f.

6: s_0xing ($\mathtt{s}^{\pm}_{0}$) zero crossings of s.

7: s_0xing_pos ($\mathtt{s}^{+}_{0}$) rising zero crossings of s.

8: s_0xing_neg ($\mathtt{s}^{-}_{0}$) falling zero crossings of s.

9: cfd_high ($\mathtt{f}^{+}_{high}$) rising crossings of the high CFD threshold by f.

10: cfd_low ($\mathtt{f}^{+}_{low}$) rising crossings of the low CFD threshold by f.

11: max_slope ($\mathtt{s}^{+}_{max}$) the first point in the rise with maximum slope.

mca.qualifier unsigned:4 bit (tes.mca.Qualifier)

The sequence selected by value is sampled into the histogram when trigger and qualifier are both *true*.

0: disabled always *false*, no triggers qualify.

1: all always true all triggers qualify.

2: valid_rise ($\mathtt{valid}_{rise}$) *true* during rises that qualify as a detection, see Figure 6.4.

3: above_area *true* when pulse_area > area_threshold.

4: above (above) *true* when f > pulse_threshold.

5: will_cross ($\mathtt{will}_{cross}$) *true* during a rise were f will cross pulse_threshold.

6: armed (armed) becomes *true* when s crosses slope_threshold during a rise and remains *true* till the end of the rise.

7: will_arm ($\mathtt{will}_{arm}$) *true* during rises where s will cross slope_threshold.

8: rise *true* from a rising zero crossing of s to the next falling zero crossing.

9: valid_rise1 *true* during a valid first rise.

10: valid_rise2 *true* during rises that have rise_number=1, ie a valid second rise during a pulse.

11: valid_rise3 *true* during rises that have rise_number=2, ie a valid third rise in a pulse.

mca.channel unsigned:3 bit (int)

The channel to analyse.

mca.bin_n unsigned:5 bit (int)

Sets the width of histogram bins to $2^{\texttt{bin\_n}}$.

mca.last_bin unsigned:14 bit (int)

The bin used for overflows, the number of bins in the histogram is last_bin + 1 including the overflow and underflow bins.

mca.update strobe (bool)

Update the MCA registers at the next tick where a buffer is free.

mca.update_on_completion strobe (bool)

Update the MCA registers after the current buffer has completed its ticks and a buffer is free.

A.1.4 ADC group

The number of channels in the ADC group is 2 ∗ adc_chips and this value is accessable through the client instance variable adc_channels.

adc.enable 1 bit (bool)

Sets or clears the bit corresponding to this channel in the adc.adc_enable register. When False the ADC is put in a low-power mode.

adc.pattern unsigned:3 bit (int)

Sets the type of ADC test pattern. See the ADS62P49 datasheet.

adc.pattern_low unsigned:8 bit (int)

The low byte of the of the custom test pattern See the ADS62P49 datasheet.

adc.pattern_high unsigned:6 bit (int)

The upper 6 bits of the custom test pattern. See the ADS62P49 datasheet.

adc.gain unsigned:4 bit (int)

The gain of the ADC input stage, 0 to 6 dB in 0.5 dB steps. See the ADS62P49 datasheet.

A.1.5 Channel group

The number of channels in the channel group is channel_count and this value is accessible through the client instance variable dsp_channels.

channel.cpu_version Read only, unsigned:32 bit (str)

Date and time the channel CPU code was assembled. Same format as the main_cpu register.

channel.adc_select unsigned:8 bit int

Selects the ADC input for the channel.

`channel.invert` 1 bit (`bool`)

Multiply the selected ADC output by −1.

`channel.delay` unsigned:16 bit The delay applied (4ns clock pulses) the selected ADC output is delayed before entering the measurement pipeline.

A.1.6 Baseline group

The number of channels in the baseline group is `channel_count` and this value can be accessible through the client instance variable `dsp_channels`.

`baseline.offset` signed:16 bit (`int`)

The value of the `offset` register is subtracted from the selected ADC output to correct any DC offset.

`baseline.time_constant` unsigned:32 bit (`int`)

The period (4ns clock pulses) that buffers in the baseline MCA are cleared the buffer that is cleared alternates.

`baseline.threshold` signed:16 bit (`int`)

`offet` corrected ADC values that are greater than `threshold` are not included in the baseline estimate. (may not work use 0xFFFF to be safe)

`baseline.count_threshold` unsigned:32 bit (`int`)

The most frequent bin of the baseline MCA is only included in the baseline estimate when its frequency is greater than `count_threshold`.

`baseline.new_only` 1 bit (`bool`)

When *true* the baseline MCAs most frequent bin is included in the baseline estimate only when the bin changes, otherwise it is also included when it's frequency changes.

`baseline.dynamic` 1 bit (`bool`)

When *true* the baseline estimate is subtracted from the `offset` corrected ADC signal. This effectively turns the baseline correction on and off.

A.1.7 CFD group

`cfd.fraction` unsigned:17 bit (`float`)

The constant fraction used in the CFD process to calculate $\texttt{cf}_{low}$ and $\texttt{cf}_{high}$.

`cfd.rel2min` 1 bit (`bool`)

When `False` the calculation of $\texttt{cf}_{low}$ and $\texttt{cf}_{high}$ use the true minimum, $\texttt{f}[\texttt{s}|_0^+]$. Otherwise a minimum of 0 is used for first rises.

A.1.8 Event group

`event.enable` 1 bit (`bool`)

Sets/clears the bit corresponding to this channel in the global `event_enable` register and enables/disables event packet transmission from this channel.

`event.packet` unsigned:2 bit (`tes.base.Detection`)

Sets the type of event packet generated by this channel, see section 5.4

0: `rise` a `rise` packet.

1: `pulse` a `pulse` packet.

2: `area` an `area` packet.

3: `trace` the packet is determined by the `trace` setting.

`event.trace` unsigned:2 bit (`tes.base.TraceType`)

Sets the type of sequence recorded. Note `packet` must be set to `trace`, see subsection 5.4.3.

0: `pulse` a `pulse` packet that includes a record of `trace_sequence`.

1: `average` accumulates a template that is the average of a number of pulses then returns the template (see section 5.4.3).

2: `dot_product` a `pulse` packet that includes a dot product field containing the dot product of `trace_sequence` with the average template. NOTE: this packet does not contain a sequence record. (see section 5.4.3).

3: `dot_product_trace` same as `dot_product` but includes a record of `trace_sequence`.

`event.trace_sequence` unsigned:2 bit (`tes.base.Sequence`)

Selects the sequence captured in traces.

0: `default` `f` the filtered detection sequence.

1: `raw` `raw` the raw detection sequence.

2: `f` `f` the filtered detection sequence.

3: `s` `s` the slope `f`.

`event.trace_stride` unsigned:5 bit (`int`)

Sets the number of samples skipped when recording `trace_sequence`. For example, if `stride`=1 the trace will contain {`sample[0]`, `sample[2]`, `sample[4]`, ...} of `trace_sequence`. NOTE: the sample at the `timing` point is always included.

`event.trace_pre` unsigned:5 bit (`int`)

Sets the number of samples in a trace prior to the `timing` point.

`event.trace_length` unsigned:13 bit (`int`)

The number of samples to capture in a trace.

`event.timing` unsigned:2 bit (`tes.base.Timing`)

Selects the point a the rise that generates the timestamp (see Equation 6.4 and Figure 6.4)

0: `pulse_threshold` when `f` crosses `pulse_threshold` for first rises otherwise when `f` first equals or exceeds the low CFD threshold during the rise.

1: `slope_threshold` when `s` crosses `slope_threshold`.

2: `cfd_low` when `f` first equals or exceeds the low CFD threshold during the rise.

3: `slope_max` when `s` first equals its maximum value in the rise.

`event.max_rises` unsigned:4 bit (`int`)

The number of rise slots in a `pulse` based packet is `max_rises` + 1.

`event.height` unsigned:2 bit (`int`)

Selects the the sequence that drives the `height` sequence. The `height` sequence is captured into the `height` field of an event packet by the packet engine.

0: `peak` `height` = `f`

1: `cfd_high` `height` = `high`. Note this is the high CFD *threshold*.

2: `cfd_height` `height` = `high` − `low` the difference between the high and low CFD thresholds.

3: `max_slope` `height` = `max`.

`event.pulse_threshold` unsigned:16 bit (`int`)

For a rise to be considered a detection and produce an event packet `f` must be above `pulse_threshold` at the end of the rise and `s` must have exceeded `slope_threshold` during the rise.

`event.slope_threshold` unsigned:16 bit (`int`)

For a rise to be considered a detection and produce an event packet `f` must be above `pulse_threshold` at the end of the rise and `s` must have exceeded `slope_threshold` during the rise.

`event.area_threshold` unsigned:32 bit (`int`)

The `area` sequence must be greater than or equal to `area_threshold` at the falling crossing of `pulse_threshold` by `f` for the packet engine to produce an event packet that contains an `area`.

A.2 Serial IO protocol

Register IO is currently transported over a RS232 serial connection via the Silicon Labs CP210x USB to UART bridge on the ML605 development board. The protocol is compatible with the AMBA AXI Lite specification, adopted by Xilinx and used widely in their IP cores, making porting to a higher bandwidth transport bus reasonably painless.

Serial IO commands and responses are encoded as ASCII hex *characters* `(0-9,A-F)` and terminated with a carriage return \r `(0x0D)`.

Commands

Commands are 19 characters in length including the terminator

`VVVVVVVVAAAAAAAA0C`\r

where the 8 hex characters `VVVVVVVV` represent the 32 bit value to be written, `AAAAAAAA` the 32 bit register address and `C` one of the following command op-codes.

1	write register
2	read register
3	reset (warm reboot of FPGA)

The value part is ignored in a read command but must be present, and must be `00000000` for a valid reset.

Responses

After a command is processed a 3 or 11 character response is returned

`RC`\r	write response
`VVVVVVVVRC`\r	read or reset response

where `C` is the op-code being responded to, `R` the response code and `VVVVVVV` the returned value. The 4 bit response code indicates errors during the command

3	2	1	0
non hex	bad length	AXI response	

the `non hex` bit indicates that illegal characters were found in the command, the `bad length` bit indicates the received command was not 19 characters long, and the 2 bit `AXI response` flags any processing errors

00	OKAY
10	SLVERR
11	DECERR

where `DECERR` indicates an unknown address, and a `SLVERR` a command error, eg writing to a read-only register. The value in a reset response is the FPGA features register.

Addresses

The most significant byte of the address is used by the main CPU to route the command to various sub-systems within the design.

bit	31	28	27	24	23	0
	system		channel		24 bit sub-system address	

the system nibble is one of the following

0	channel CPUs
1	global registers
2	serial peripheral interface (SPI) bus

the `channel` nibble is ignored unless the system nibble is `0`.

bit	31–28	27–24	23–0
	`0`	`N`	`24 bit address in channel N`

where `N` is the channel number.

bit	31–28	27–24	23–0
	`1`	`X`	`24 bit global address`

The SPI bus communicates with the chips on FMC108 mezzanine board, four ADS62P49 ADC chips and the AD9510 clock distribution chip. SPI address are of the form

bit	31–28	27–24	23–13	12–8	7–0
	`2`	`X`	`reserved`	`MOSI`	`SPI address`

The SPI address is chip register address given in the data sheet and the master out slave in (MOSI) field selects which chip(s) to communicate with

bit 8	ADS62P49 chip 0 (ADC channels 0 and 1)
bit 9	ADS62P49 chip 1 (ADC channels 2 and 3)
bit 10	ADS62P49 chip 2 (ADC channels 4 and 5)
bit 11	ADS62P49 chip 3 (ADC channels 6 and 7)
bit 12	AD9510 chip

A.3 Register address map

`0x0N000000 channel CPU version (read-only)`

31 0
channel.cpu_version

The 32 bit return value is formatted `YMDDHHmm` where each character represents a 4 bit nibble.

`Y`	Year mod 16
`M`	Month
`DD`	Day
`HH`	Hour (24-hour)
`mm`	Minute

`0x0N000001 event type`

31 10	9 8	7 4	3 2	1 0
reserved	H	MP	T	P

H	`event.height`
MP	`event.max_peaks`
T	`event.timing`
P	`event.packet`

`0x0N000002 pulse threshold`

31 16	15 0
reserved	event.pulse_threshold

`0x0N000004 slope threshold`

31 16	15 0
reserved	event.slope_threshold

0x0N000008 constant fraction

31	17	16 0
R	reserved	cfd.fraction

R cfd.rel2min

0x0N000010 area threshold

31 0
event.area_threshold

0x0N000020 channel delay

31 10	9 0
reserved	channel.delay

0x0N000040 baseline offset

31 16	15 0
reserved	baseline.offset

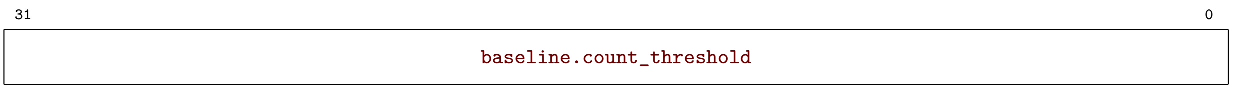
0x0N000080 baseline time constant

31 0
baseline.time_constant

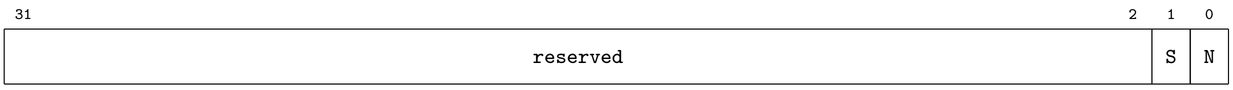
0x0N000100 baseline threshold

31 16	15 0
reserved	baseline.threshold

0x0N000200 baseline count threshold

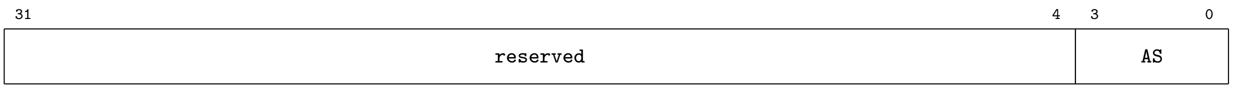


0x0N000400 baseline control



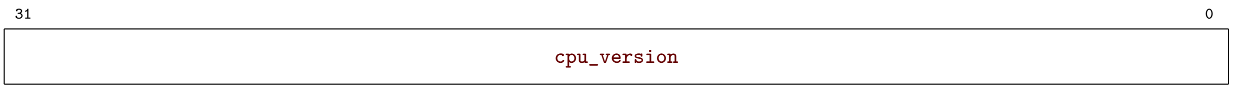
N	`baseline.new_only`
S	`baseline.subtraction`

0x0N000800 channel input select



AS	`channel.adc_select`

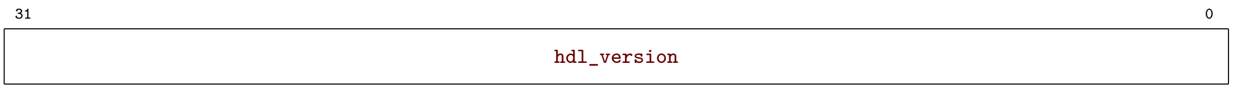
0x10000000 main CPU version (read-only)



The 32 bit return value is formatted `YMDDHHmm` where each character represents a 4 bit nibble.

Y	Year mod 16
M	Month
DD	Day
HH	Hour (24-hour)
mm	Minute

0x10000001 HDL version (read-only)



0x10000002 MCA

31 30	29 16	15 11	10 8	7 4	3 0
resv	LB	BN	C	T	V

LB	mca.last_bin
BN	mca.bin_n
C	mca.channel
T	mca.trigger
V	mca.value

0x10000004 MCA lowest value

31 0
mca.lowest_value

0x10000008 MCA ticks

31 0
mca.ticks

0x10000010 MTU

31 0
mtu

0x10000020 tick period

31 0
tick_period

0x10000040 tick latency

31 … 0
tick_latency

0x10000080 ADC enables

31 … 8	7 … 0
reserved	adc_enable

0x10000100 event enables

31 … 8	7 … 0
reserved	event_enable

0x10000200 FMC108 control

31 … 2	1	0
reserved	P	C

C	fmc_internal_clock
P	vco_power_en

0x10000400 window

31 … 0
window

0x10000800 MCA qualifier

31	7 4	3 0
reserved	mca.qual1	mca.qual0

0x10002000 MCA update strobes

31 3 2	1	0
reserved	U	C

C	mca.update_on_completion
U	mca.update

0x10800000 FPGA features (read-only)

31 21	20	19	18	17	16	15 8	7 0
res	I	M	S	P	F	adc_chips	channel_count

F	fmc
P	fmc_power
S	ad9510_status
M	mmcm_locked
I	iodelay_ready

Appendix B

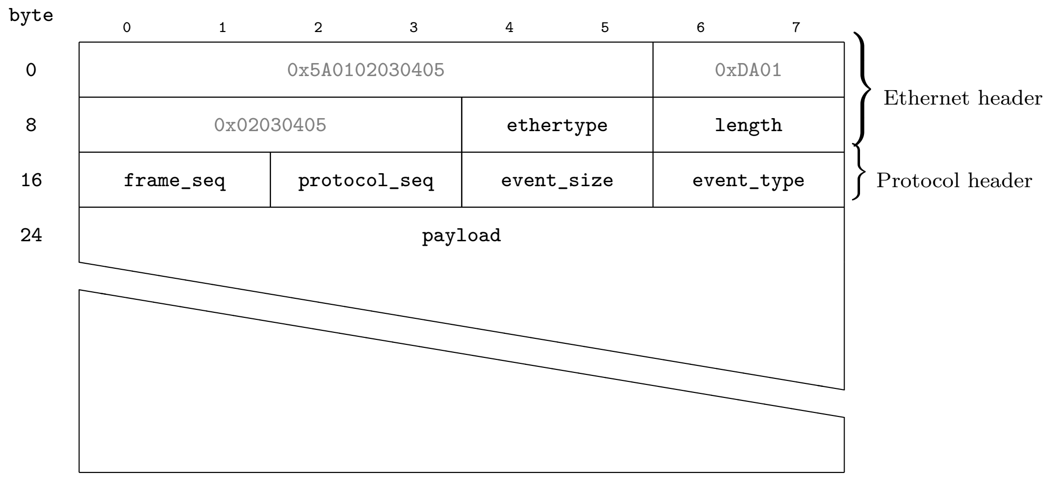
Ethernet protocols

B.1 Ethernet frames

Measurement information is sent to the host computer from the FPGA over a dedicated point to point Ethernet connection as raw Ethernet frames. A standard 16 byte Ethernet header is present containing spoofed source and destination addresses which can safely be ignored. The 14 bytes before the `length` field are always in network byte order (big endian) while the byte order of rest of the frame can be changed to match that of the host and is little endian by default.

Two distinct streams are transmitted by the FPGA; the MCAstream carries the distributions of measurement values as histograms output from the MCA; and the eventstream carries event packets containing timestamped measurements of TES detection pulses. The `ethertype` field indicates which stream the frame carries. An 8 byte protocol header follows the Ethernet header and contains sequence numbers and fields describing the payload for frames in the event stream.

Ethernet Frame



Ethernet header fields

`ethertype`: (big endian).

`0x88B5` the frame is in the event stream.

`0x88B6` the is in the MCA stream.

`length`: length in bytes of the valid part of the frame. Note this maybe less than the length of the Ethernet frame which has a minimum length of 60 bytes.

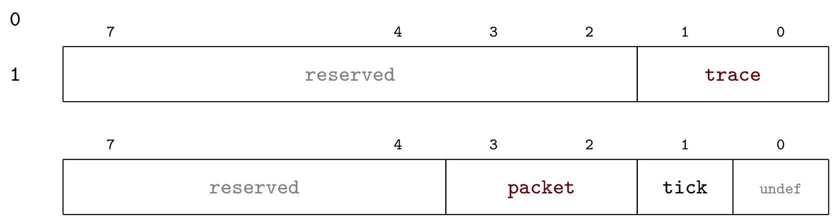
Protocol header fields

`frame_seq`: incremented for each frame sent by the FPGA.

`protocol_seq`: independently incremented for each stream type and set to zero for header frames.

`event_size`: the size in 8 byte chunks of the event packets forming the payload array.

`event_type`: (2 bytes) the type of event packet in the payload array.

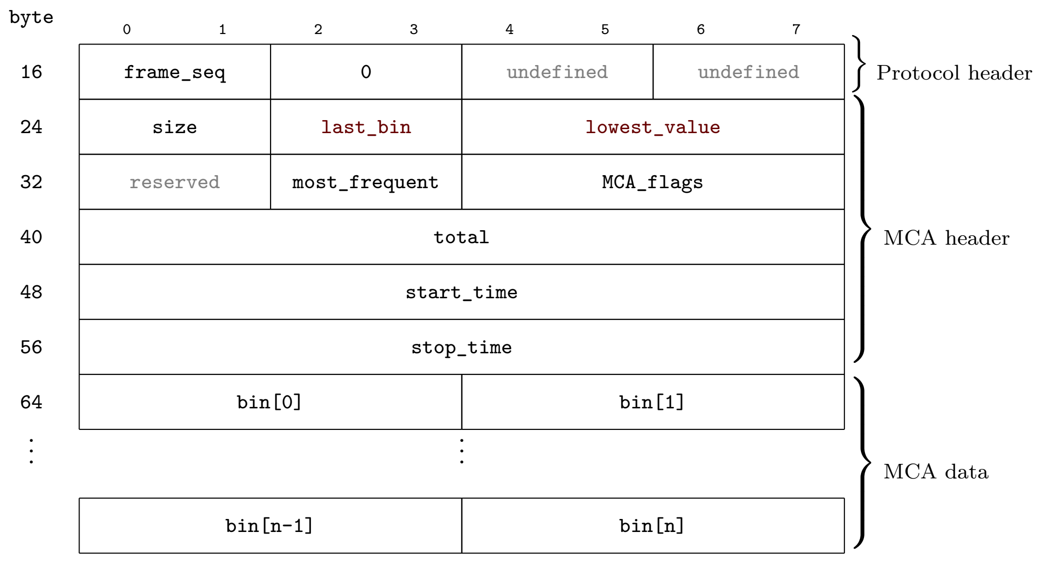


When the `tick` bit is set the payload is a `tick` packet carrying a time stamp and error information and the other fields are undefined. Otherwise packet and trace are the `event.packet` and `event.trace` settings that generated the packets in the payload.

Both `event_type` and `event_size` are only valid for frames in the eventstream.

B.2 The MCAstream `ethertype=0x88B6`

A complete histogram from the MCA can extend over multiple frames and eventstream frames may be interspersed during the transmission of the histogram. Frames in the MCAstream which have `protocol_seq=0` are MCA header frames that contain register settings and other information for the captured histogram. The MCA header is followed by histogram bin data that generally extends over multiple frames. In each frame after the header the bin data starts at byte 24 and `protocol_seq` is incremented.



B.2.1 MCA header fields

`size`: The total size of the histogram in bytes including the header and all bins.

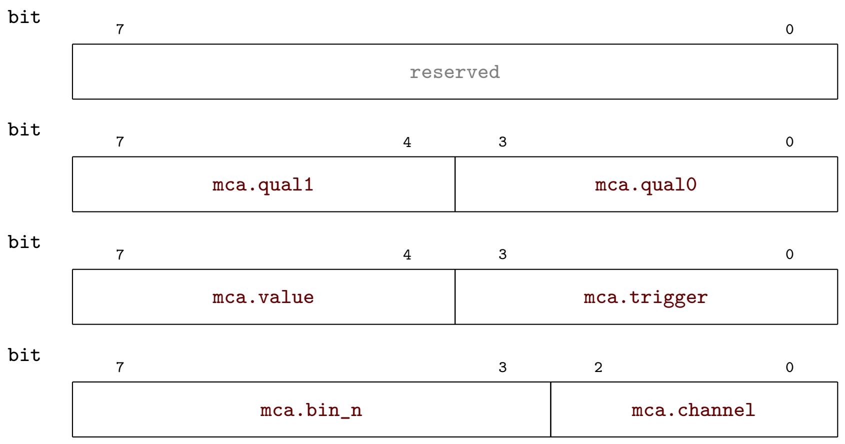
`last_bin`: the `mca.last_bin` setting.

`lowest_value`: the `mca.lowest_value` setting.

`most_frequent`: The bin with the highest count.

`MCA_flags`: (4 bytes) contains register settings used to capture the histogram.

The bytes in transmission order are:



`total`: The sum of all bins.

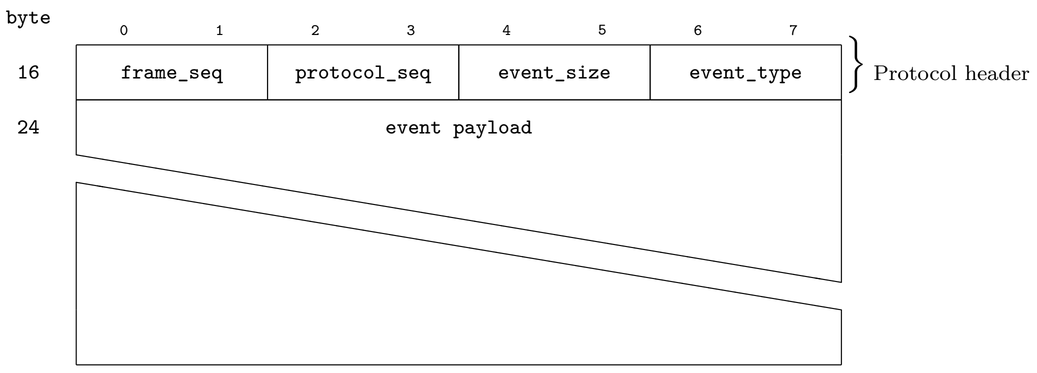
`start_time`: a 64 bit timestamp indicating when accumulation started.

`stop_time`: a 64 bit timestamp indicating when accumulation stopped.

B.3 The eventstream (ethertype=0xBB85)

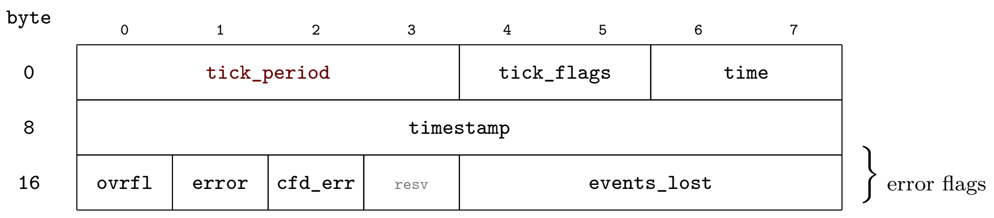
Frame in the eventstream carry event packets (section 5.4) in three payload formats; the *tick* payload contains a single `tick` packet; *measurement* payloads consist of an array of event packets each containing measurements of a TES detection pulse; and *trace* payloads begin with a single event packet as a header which is followed by a sequence record. Trace packets may extend over multiple *sequential* frames that will never be interspersed with MCAstream frames The header frame that starts with abn event packet is indicated by a 0 in the `protocol_sequence` field. Each payload type can be identified by inspecting the `event_type` field for the frame.

The transport framer (section 5.2) guarantees each measurement payload array consists of event packets of the same type and size and the packets are designed for optimum memory alignment.



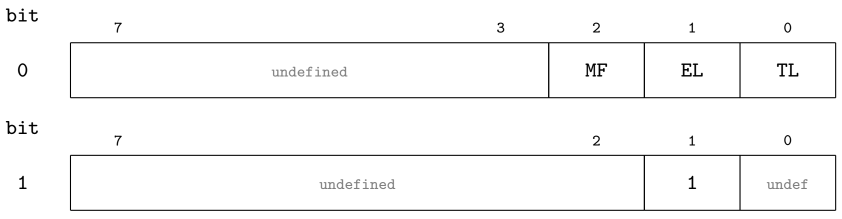
B.3.1 The `tick` packet

Each event packet includes a 16 bit `time` field which records the number of 4 ns clocks since the previous event in the stream, `time` saturates at $2^{16} - 1$. By setting the `tick_period` register to less than 2^{16} a continuously time-stamped detection record is acquired.



`tick_period` (unsigned 32 bit) The `tick_period` setting for this tick.

`tick_flags`: (2 bytes)



`MF`: the eventstream MUX time queue overflowed.

`EL`: the `events_lost` field is not zero.

`TL`: the tick framer overflowed.

`time` (unsigned 16 bit) The time since the previous event (saturates), see the `time` field.

`timestamp`: (unsigned 64 bit) The system time when this tick was generated.

`ovrfl`: 1 bit flags indicating that the packet engine for that channel overflowed since the last tick. LSB is channel 0.

`error`: 1 bit flags indicating that the packet engine for that channel had a framing error since the last tick. LSB is channel 0.

`cfd_err`: 1 bit flags indicating that CFD for that channel overran since the last tick. LSB is channel 0.

`events_lost`: (unsigned 32 bit) The number of events lost since the previous tick.

Glossary

AD9510 Clock generation and distribution chip from Analogue devices. 72, 79, 91

ADS62P49 Dual channel 14 bit 250 Mhz ADC chip from Analogue devices. 74, 79, 91

detection a rise that ends with `f` above `pulse_threshold` and `s` has crossed `slope_threshold` during the rise. 23, 77, 91

entity A VHDL functional unit fulfilling a similar role to a software subroutine or class in that it enables code to be reused. It is analogous to a discrete chip with input and output pins called ports. 49

event packet A packet containing measurement fields from a detection. , 8, 22, 24, 29, 42, 43, 54, 57, 58, 59, 65, 66, 77, 86, 88, 89, 91

eventstream An AXI stream of event packets carrying measurements of TES detection pulses. 47, 48, 49, 52, 53, 54, 86, 87, 88, 89, 92

fabric Ubiquitous FPGA resources arranged as a large array of cells covering most of the FPGA chip's die. Each cell contains low level digital components like registers (flip-flops) and MUXs. Fabric resources are the most general an *any* circuit can be realised using them. The large number of routing connections between the low level components in a complex circuit built from fabric resources can limit maximum clock speed. 9

field A entry in a packet representing a value. , 49, 54, 55, 57, 59, 66, 91, 92

first rise a rise that starts below the `pulse_threshold` register setting. 50, 54, 65, 66, 73, 76

FMC108 An ADC mezzanine card containing 4 dual channel ADS62P49 ADCs and a AD9510 clock management chip. It is connected to the ML605 FPGA development board's high pin count FMX connector. 72

frame A structure used to transport packets to the host. A frame may contain multiple packets or a fragment of a single packet. 49, 88, 89

framer A cross between a RAM and a FIFO, the input port allows random writing of fields an a frame. The output port implements an AXI stream interface and when the frame is committed it streams out, asserting last at the end of the frame. 49, 53, 54, 89, 92

generic A VHDL constant used during synthesis to control how an entity is implemented. , 57, 65

hard core dedicated silicon resources in a FPGA performing a specific function. These are effectively internal discrete chips performing functions like RAM, multiply and accumulate for DSP filters, etc. 9

last An AXI stream signal asserted by the stream source to indicate the last transfer in a packet. 49

MCAstream The AXI stream carrying histograms from the MCA. 52, 86, 87, 89

packet An ordered collection of fields carrying data, one field represents the length of the packet. Packets are transported in which have a maximum length set by the MTU. Multiple packets can be transported in a single frame or a single packet can be fragmented across multiple frames depending on the packet length and the MTU. 49, 53, 55, 88, 91

packet engine An entity containing a framer and controlled by FSMs, it creates event packets from measurements of TES output to produce an eventstream. 49, 52, 53, 54, 66, 77, 90

pile-up The long TES relaxation time leads to pile-up of detection pulses when another photon is detected before the TES as fully recovered from a previous detection. 7

pulse a sub-sequence of `f` from the start of a valid rise to the next falling crossing of `pulse_threshold` by `f`. 24, 29, 50, 59, 73

ready An AXI stream handshake signal asserted by the stream sink to indicate it is ready to accept data. 49, 93

rise a sub-sequence of the `f` sequence that extends from a local minima to the following local maxima. 22, 23, 24, 29, 41, 43, 50, 55, 57, 59, 73, 77, 91, 92

RS232 Ancient but simple protocol for serial communication. 68, 77

sequence An ordered list of integers. In this thesis sequences originate from quantisation of a continuous voltage signal by an ADC. , 21, 22, 23, 24, 29, 30, 40, 41, 42, 49, 50, 60, 76, 89, 92

trace A sequence record of fixed length. 49, 76

transfer Data is transferred between between a source and a sink in a stream when the valid and ready handshakes are both asserted. the amount of data transferred is the width of the stream. 49, 54

valid An AXI stream handshake signal asserted by the stream source to indicate it has valid data to write. 49, 93

VHDL A hardware definition language developed by the U.S. department of defence. , 9, 43, 57, 65

Xilinx Market leading manufacturer of FPGAs and developer of infuriating software tools. 10, 61, 77

ZeroMQ a high-performance asynchronous messaging library, with bindings for many popular programming languages. 68

Acronyms

ADC analogue to digital converter. 21, 23, 24, 26, 28, 45, 48, 50, 60, 71, 72, 74, 75, 79, 91, 92

ADR adiabatic demagnetisation refrigerator. 2, 7, 26, 27, 28

AIC Akaike information criterion. 33, 44

AXI advanced extensible interface. 49, 52, 77, 91, 92, 93

CDF cumulative density function. 34, 35

CFD constant fraction discriminator. 20, 55, 63, 65, 66, 73, 75, 76, 77, 90

CPU central processing unit. 48, 71, 74

DSP digital signal processing. 9, 61, 92

EM Expectation Maximisation. 32, 33

EPR Einstein, Podolsky and Rosen. 14, 19

ETF electro-thermal feedback. 3, 4, 5

FIFO first in first out. 49, 63, 92

FIR finite impulse response. 23, 52, 61

FMC FPGA mezzanine connector. 72

FPGA field programmable gate array. 9, 10, 26, 28, 51, 61, 68, 72, 86, 87, 91, 92, 93

FSM finite state machine. 54, 63, 92

HDL hardware definition language. 9, 10

MCA multi-channel analyser. 24, 25, 28, 29, 30, 41, 42, 43, 45, 48, 49, 52, 60, 66, 67, 72, 73, 75, 86, 87, 92

ML Maximum Likelihood. 32, 33

MMCM multi-mode clock manager. 72

MOSI master out slave in. 79

MTU maximum transmission unit. 49, 92

MUX multiplexer. 48, 52, 53, 54, 89, 91

NIST National Institute of Standards and Technology. 2, 5, 45

PDF probability density function. 33, 34

PID proportional-integral-differential. 27

POVM positive-operator valued measure. 35, 37, 41, 42, 44

PTR pulse tube refrigerator. 27

RAM random access memory. 9, 49, 53, 59, 92

SPDC spontaneous parametric down-conversion. 19, 20

SPI serial peripheral interface. 78, 79

SQUID superconducting interference device. 3, 7

TES transition edge sensor. , 1, 2, 3, 4, 5, 6, 7, 8, 11, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 30, 31, 32, 36, 40, 41, 42, 44, 45, 47, 48, 52, 53, 61, 62, 72, 86, 89, 91, 92

W-TES Tungsten transition edge sensor. 2, 5

Bibliography

[Aka98] Hirotogu Akaike. Information Theory and an Extension of the Maximum Likelihood Principle. In *Selected Papers of Hirotugu Akaike*, pages 199–213. Springer New York, New York, NY, 1998.

[BA57] D Bohm and Y Aharonov. Discussion of Experimental Proof for the Paradox of Einstein, Rosen, and Podolsky. *Phys. Rev.*, 108(4):1070–1076, November 1957.

[BA60] D Bohm and Y Aharonov. Further discussion of possible experimental tests for the paradox of Einstein, Podolsky and Rosen. *Il Nuovo Cimento*, 17(6):964–976, September 1960.

[BAB$^+$15] S E Busch, J S Adams, S R Bandler, J A Chervenak, M E Eckart, F M Finkbeiner, D J Fixsen, R L Kelley, C A Kilbourne, S J Lee, S H Moseley, J P Porst, F S Porter, J E Sadleir, and S J Smith. Progress Towards Improved Analysis of TES X-ray Data Using Principal Component Analysis. *Journal of Low Temperature Physics*, 184(1-2):382–388, November 2015.

[Bac] Dave Bacon. Quantum Entanglement and Bell's Theorem.

[Bel64] John Stewart Bell. On the Einstein Podolsky Rosen paradox. *Physics*, 1(3):195–200, 1964.

[BMG$^+$07] Agata M Branczyk, Paulo E M F Mendonca, Alexei Gilchrist, Andrew C Doherty, and Stephen D Bartlett. Quantum control of a single qubit. *Physical Review A (Atomic, Molecular, and Optical Physics)*, 75(1):012329, 2007.

[CHSH69] John F Clauser, Michael A Horne, Abner Shimony, and Richard A Holt. Proposed Experiment to Test Local Hidden-Variable Theories. *Physical Review Letters*, 23(1):880–884, October 1969.

[CLFN11] Brice Calkins, Adriana E Lita, Anna E Fox, and Sae Woo Nam. Faster recovery time of a hot-electron transition-edge sensor by use of normal metal heat-sinks. *Applied Physics Letters*, 99(24):241114, 2011.

[DAB$^+$07] D Drung, C Assmann, J Beyer, A Kirste, M Peters, F Ruede, and T Schurig. Highly Sensitive and Easy-to-Use SQUID Sensors. *IEEE Transactions on Applied Superconductivity*, 17(2):699–704, June 2007.

[DLR77] A P Dempster, N M Laird, and D B Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, January 1977.

[dW00] A T A M de Waele. Pulse-tube refrigerators: principle, recent developments, and prospects. *Physica B: Physics of Condensed Matter*, 280(1):479–482, May 2000.

[EM] A Einstien and Born M. *The Born-Einstien Letters, 1916-1955.* Friendship, Politics and Physics in Uncertain Times. Times (Macmillan Science).

[Ens05] Christian Enss. *Cryogenic Particle Detection.* Topics in applied physics. Springer, Berlin, 2005.

[EPR35] A Einstein, B Podolsky, and N Rosen. Can Quantum-Mechanical Description of Physical Reality Be Considered Complete? *Phys. Rev.*, 47(10):777–780, May 1935.

[FFCM$^+$00] E Figueroa-Feliciano, B Cabrera, A J Miller, S F Powell, T Saab, and A B C Walker. Optimal filter analysis of energy-dependent pulse shapes and its application to TES detectors. *Nuclear Instruments and Methods in Physics Research Section A*, 444(1):453–456, April 2000.

[Fin] Arthur Fine. The Einstein-Podolsky-Rosen Argument in Quantum Theory.

[FP12] Alessandro Ferraro and Matteo G A Paris. Nonclassicality Criteria from Phase-Space Representations and Information-Theoretical Constraints Are Maximally Inequivalent. *Physical Review Letters*, 108(2):260403, June 2012.

[GDL$^+$10] G G Gillett, R B Dalton, B P Lanyon, M P Almeida, M Barbieri, G J Pryde, J L O'Brien, K J Resch, S D Bartlett, and A G White. Experimental Feedback Control of Quantum Systems Using Weak Measurements. *Phys. Rev. Lett.*, 104(8):080503, February 2010.

[GHSZ90] Daniel M Greenberger, Michael A Horne, Abner Shimony, and Anton Zeilinger. Bell's theorem without inequalities. *American Journal of Physics*, 58(12):1131–1143, 1990.

[GVW$^+$15] Marissa Giustina, Marijn A M Versteegh, Sören Wengerowsky, Johannes Handsteiner, Armin Hochrainer, Kevin Phelan, Fabian Steinlechner, Johannes Kofler, Jan-Åke Larsson, Carlos Abellán, Waldimar Amaya, Valerio Pruneri, Morgan W Mitchell, Jörn Beyer, Thomas Gerrits, Adriana E Lita, Lynden K Shalm, Sae Woo

Nam, Thomas Scheidl, Rupert Ursin, Bernhard Wittmann, and Anton Zeilinger. Significant-Loophole-Free Test of Bell's Theorem with Entangled Photons. *Physical Review Letters*, 115(2):250401, December 2015.

[Hol08] Pavel Holoborodko. Smooth Noise Robust Differentiators, 2008.

[IH05] K D Irwin and G C Hilton. Transition-Edge Sensors. In Christian Enss, editor, *Cryogenic Particle Detection*, pages 63–150. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[IHWM98] K D Irwin, G C Hilton, D A Wollman, and John M Martinis. Thermal-response time of superconducting transition-edge microcalorimeters. *Journal of Applied Physics*, 83(8):3978–3985, April 1998.

[Irw95] K D Irwin. An application of electrothermal feedback for high resolution cryogenic particle detection. *Applied Physics Letters*, 66(15):1998–2000, April 1995.

[Jar84] Jon P Jarrett. On the Physical Significance of the Locality Conditions in the Bell Arguments. *Noûs*, 18(4):569, November 1984.

[JNN13] J R Johansson, P D Nation, and Franco Nori. QuTiP 2: A Python framework for the dynamics of open quantum systems. *Computer Physics Communications*, 184(4):1234–1240, April 2013.

[KLM01] E Knill, R Laflamme, and G J Milburn. A scheme for efficient quantum computation with linear optics. *Nature*, 409(6816):46–52, 2001.

[Lar98] Jan-Åke Larsson. Bell's inequality and detector inefficiency. *Physical Review A (Atomic*, 57(5):3304–3308, May 1998.

[LCP$^+$10] A E Lita, B Calkins, L A Pellouchoud, A J Miller, and S Nam. Superconducting transition-edge sensors optimized for high-efficiency photon-number resolving detectors. In Mark A Itzler and Joe C Campbell, editors, *SPIE Defense, Security, and Sensing*, pages 76810D–76810D–10. SPIE, April 2010.

[Lev44] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[LGM$^+$12] Zachary H Levine, Thomas Gerrits, Alan L Migdall, Daniel V Samarov, Brice Calkins, Adriana E Lita, and Sae Woo Nam. Algorithm for finding clusters with a known distribution and its application to photon-number resolution using a superconducting transition-edge sensor. *Journal of the Optical Society of America B*, 29(8):2066–, August 2012.

[LGM$^+$14] Zachary H Levine, Boris L Glebov, Alan L Migdall, Thomas Gerrits, Brice Calkins, Adriana E Lita, and Sae Woo Nam. Photon-number uncertainty in a superconducting transition edge sensor beyond resolved-photon-number determination. *Journal of the Optical Society of America B*, 31(10):20–B24, October 2014.

[LGPM15] Zachary H Levine, Boris L Glebov, Adam L Pintar, and Alan L Migdall. Absolute calibration of a variable attenuator using few-photon pulses. *Optics Express*, 23(12):16372–, June 2015.

[Lin00] Mark Anton Lindeman. Microcalorimetry and the transition-edge sensor. *ProQuest Dissertations And Theses; Thesis (Ph.D.)–University of California*, October 2000.

[LLCT$^+$13] Antia Lamas-Linares, Brice Calkins, Nathan A Tomlin, Thomas Gerrits, Adriana E Lita, Jörn Beyer, Richard P Mirin, and Sae Woo Nam. Nanosecond-scale timing jitter for single photon detection in transition edge sensors. *Applied Physics Letters*, 102(2):231117, June 2013.

[LMN08] Adriana E Lita, Aaron J Miller, and Sae Woo Nam. Counting near-infrared single-photons with 95 *Opt. Express*, 16(5):3032–3040, March 2008.

[LRN$^+$05] A E Lita, D Rosenberg, S Nam, A J Miller, D Balzar, L M Kaatz, and R E Schwall. Tuning of Tungsten Thin Film Superconducting Transition Temperature for Fabrication of Photon Number Resolving Detectors. *Applied Superconductivity, IEEE Transactions on*, 15(2):3528–3531, June 2005.

[McC05] D McCammon. Thermal Equilibrium Calorimeters - An Introduction. *Cryogenic Particle Detection*, 9:1–, 2005.

[Mer85] N David Mermin. Is the moon there when nobody looks? Reality and the quantum theory. *Physics Today*, 38(4):38–47, April 1985.

[Mig08] Alan Migdall. Correlated-Photon Metrology Without Absolute Standards. *Physics Today*, 52(1):41–46, January 2008.

[MLC$^+$11] Aaron J Miller, Adriana E Lita, Brice Calkins, Igor Vayshenker, Steven M Gruber, and Sae Woo Nam. Compact cryogenic self-aligning fiber-to-detector coupling with losses below one percent. *Opt. Express*, 19(10):9102–9110, May 2011.

[NCC$^+$99] Sae Woo Nam, B Cabrera, P Colling, R M Clarke, E Figueroa-Feficiano, A J Miller, and R W Romani. A new biasing technique for transition edge sensors with electrothermal feedback. *IEEE Transactions on Appiled Superconductivity*, 9(2):4209–4212, June 1999.

[OZ01] Harold Ollivier and Wojciech H Zurek. Quantum Discord: A Measure of the Quantumness of Correlations. *Phys. Rev. Lett.*, 88(1):281, December 2001.

[Pai79] A Pais. Einstein and the quantum theory. *Review of Modern Physics*, 51(4):863–914, October 1979.

[SGdA$^+$12] Devin H Smith, Geoff Gillett, Marcelo P de Almeida, Cyril Branciard, Alessandro Fedrizzi, Till J Weinhold, Adriana Lita, Brice Calkins, Thomas Gerrits, Howard M Wiseman, Sae Woo Nam, and Andrew G White. Conclusive quantum steering with superconducting transition-edge sensors. *Nat Commun*, 3, January 2012.

[SMSC$^+$15] Lynden K Shalm, Evan Meyer-Scott, Bradley G Christensen, Peter Bierhorst, Michael A Wayne, Martin J Stevens, Thomas Gerrits, Scott Glancy, Deny R Hamel, Michael S Allman, Kevin J Coakley, Shellee D Dyer, Carson Hodge, Adriana E Lita, Varun B Verma, Camilla Lambrocco, Edward Tortorici, Alan L Migdall, Yanbao Zhang, Daniel R Kumor, William H Farr, Francesco Marsili, Matthew D Shaw, Jeffrey A Stern, Carlos Abellán, Waldimar Amaya, Valerio Pruneri, Thomas Jennewein, Morgan W Mitchell, Paul G Kwiat, Joshua C Bienfang, Richard P Mirin, Emanuel Knill, and Sae Woo Nam. Strong Loophole-Free Test of Local Realism*. *Physical Review Letters*, 115(2):250402, December 2015.

[WGR$^+$08] Till J Weinhold, Alexei Gilchrist, Kevin J Resch, Andrew C Doherty, Jeremy L O'Brien, Geoffrey J Pryde, and Andrew G White. Understanding photonic quantum-logic gates: The road to fault tolerance. *arXiv.org*, August 2008.

[WJD07] H M Wiseman, S J Jones, and A C Doherty. Steering, Entanglement, Nonlocality, and the Einstein-Podolsky-Rosen Paradox. *Physical Review Letters*, 98(1):140402, April 2007.