



Preliminary draft 09:02 5 December 2024

5 December 2024
vojtech.vosahlo@cern.ch

FastIC+ readout in-depth description

Vojtech Vosahlo
CERN, CH-1211 Geneva, Switzerland

Keywords: FastIC+, Aurora

Summary

This document aims to describe the concept and working principles of the FastIC+ readout. To do so, it first introduces the FastIC+ chip itself. Afterwards, it focuses on the Aurora protocol, its packet structure and the structure of the FastIC+ packets. Then, the main ideas behind receiving the data stream with an microcontroller are presented. Last but not least, each block of the system is described in detail.

Contents

1	Introduction	4
2	FastIC+	5
2.1	Detection	5
2.2	Digitizing	5
2.3	Configuration and testing	5
3	Aurora protocol	7
3.1	Frame notation	7
3.2	Encoding	7
3.2.1	Data block	8
3.2.2	Separator-7 block	8
3.2.3	Separator block	9
3.2.4	User K-block	9
3.2.5	Idle block	10
3.3	Scrambling	10
4	FastIC+ packet structure	11
4.1	Data packet	11
4.2	Statistics packet	12
4.3	Counter Extension + Event Counter packet	13
5	Device concept	14
6	Readout board	14
6.1	Microcontroller	15
6.1.1	Receiving the Aurora stream	15
6.1.2	Omitting clock recovery with the FastIC+	15
6.2	FastIC+	16
6.2.1	I2C communication	17
6.2.2	Calibration pulse generator	17
6.2.3	Voltage monitoring	17
6.2.4	High speed outputs	18
6.2.5	Trigger input and output	18
6.3	USB	19
6.3.1	External PHY	20
6.3.2	Power Delivery	20
6.4	Clock generation	21
6.5	High Voltage	23
6.6	Power	25
6.7	Connector	26
6.7.1	Identification pins	26

7	User Board	26
7.1	Sensors	26
7.2	EEPROM	27
8	Summary	28

1 Introduction

In order to enhance the time measurement of the ToA and ToT analog pulses, FastIC+ has added a picosecond TDC for digitizing the pulses on the chip itself. This, in turn, has necessitated the need for a high-speed communication channel capable of transferring the resulting data stream. The Aurora 64B/66B protocol was chosen for its sufficient speed and easy interfacing with FPGAs.

2 FastIC+

The FastIC+ is an 8-channel front-end for photo detectors such as SiPM, PMT or MCP, capable of precisely measuring the Time-of-Arrival (ToA) and energy of photons hitting the detector from the Time-over-Threshold (ToT).

2.1 Detection

The detection part of the chip implements current mode front-ends capable of working with both positive and negative polarity sensors with dynamic range of 5 μ A to 20 mA and capacitance range of 10 pF to 1 nF. The eight channels can either operate independently or in a summation mode and can be highly configured to suffice demanding user needs. Several triggering schemes are available for discriminating the photon impacts by their energy as well as external trigger input and trigger output to allow synchronization of multiple chips. Three different measurement modes,

- ToA (Non linear ToT),
- Energy (from ToT),
- Hybrid (ToA + Energy),

are present to allow the user to optimize for measurement of ToA, ToT or both. The maximum detection rate of the impacts is approximately 2 MHz in the linear mode and 50 MHz in non-linear mode. A 40 MHz low jitter reference clock is required for the chip to operate.

2.2 Digitizing

The measured ToA and ToT are outputted separately for each channel as a digital pulse, where the beginning of the pulse marks the ToA of the photon and the length of the pulse encodes the energy (ToT). These channels are exposed to the user and can be used for photon detection.

However in order to enhance the time measurement of the ToA and ToT pulses and offload this processing from the user, a 25 ps time bin TDC for digitizing the pulses directly on the chip has been implemented. This, in turn, has necessitated the need for a high-speed communication channel capable of transferring the resulting digital data stream. The Aurora 64B/66B protocol was chosen for its sufficient speed and easy interfacing with FPGAs. The FastIC+ itself implements a simplified Aurora transmitter with one simplex lane capable of transmission at speeds from 80 Mb/s to 1.28 Gb/s.

2.3 Configuration and testing

An I2C interface has been implemented alongside the Aurora bus to allow for easy configuration of the chip parameters. This interface can run at speeds up to 1 MHz and can be shared between multiple FastIC+ chips.

For testing purposes, four debug pins are exposed to read out the state of the internal state machine. If not used for debugging, they can be utilized for configuration of the I2C address.

3 Aurora protocol

Aurora 64B/66B is a link-layer, point-to-point protocol that enables high-speed data transfers between two Aurora partners. The Aurora channel, established between the partners, can comprise one or more simplex or full-duplex lanes. Data frames are transmitted in 66-bit blocks, consisting of a two-bit synchronization preamble and 64 bits of data. The channel is shared for both data and control messages.

3.1 Frame notation

Frame notation and bit order adopted from the Aurora specification [1] will be used in this document. In this notation the leftmost bit is the one transmitted first onto the bus and is considered the most significant in the block. The rightmost bit is LSB. An example of a block notation can be seen in Figure 1.

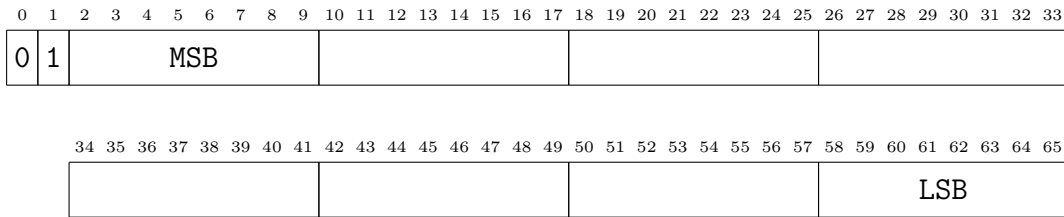


Figure 1: Example Aurora block structure

3.2 Encoding

As previously mentioned, the Aurora protocol utilizes the commonly used 64B/66B encoding. This encoding scheme converts 64 bits of data into a 66-bit block by appending a two-bit preamble. The 01 preamble signifies that the block contains 8 octets of data and is therefore called a *Data block*. If the preamble is 10, the block is recognized as *Control block* with the most significant octet in the block being a *BTF* (Block Type Field). The remaining 7 octets are data. Preambles 00 and 11 are interpreted as errors.

FastIC+ is capable of transmitting the *Data block* and four of the *Control block* types:

- *Separator-7* block indicated by the *BTF* value 0xE1,
- *Separator* block indicated by the *BTF* value 0x1E,
- *User K-Block* with *BTF* value indicating the ID of the block (see 3.2.4),
- *Idle* block indicated by the *BTF* value 0x78.

3.2.1 Data block

A *Data block*, shown in Figure 2, consists of eight octets of raw data. Data blocks are transmitted only when eight or more octets of data are available. For smaller packets, the *Separator-7* and *Separator* blocks are used.

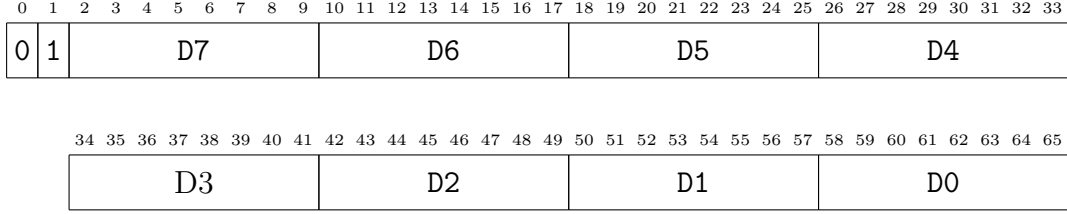


Figure 2: Data block structure

3.2.2 Separator-7 block

A *Separator-7 block* is transmitted when only seven octets of data remain to be sent over the interface. The block consists of the *BTF* indicating the *Separator-7* type and seven valid octets of data as shown in Figure 3.

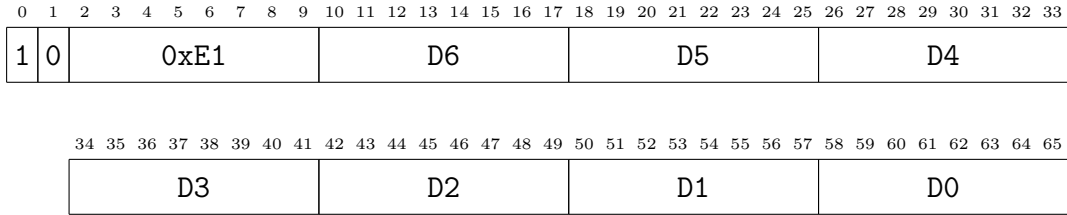


Figure 3: Separator-7 block structure

3.2.3 Separator block

A *Separator block* is transmitted when 0-6 octets of data remain to be sent. The block, shown in Figure 4, consists of the *BTF* indicating the *Separator* type and a field indicating the number of valid data octets. The order of the valid octets is from LSB to MSB.

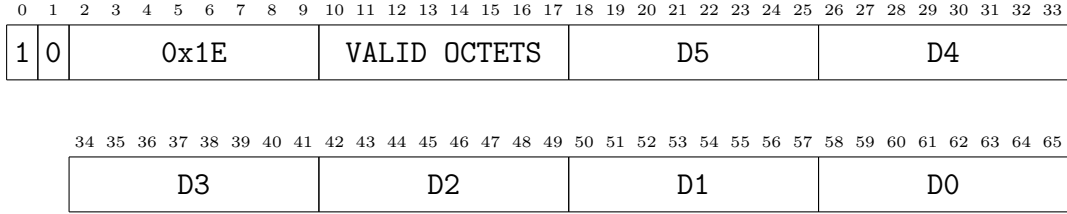


Figure 4: Separator block structure

3.2.4 User K-block

User K-Blocks are provided by the Aurora specification to allow the user application to send specific control messages separately from the data. Figure 5 shows the *K-Block* comprising the *BTF* and seven data octets. There are nine *K-Blocks* available with IDs 0 to 8, corresponding to *BTFs* 0xD2, 0x99, 0x55, 0xB4, 0xCC, 0x66, 0x33, 0x4B and 0x87, respectively.

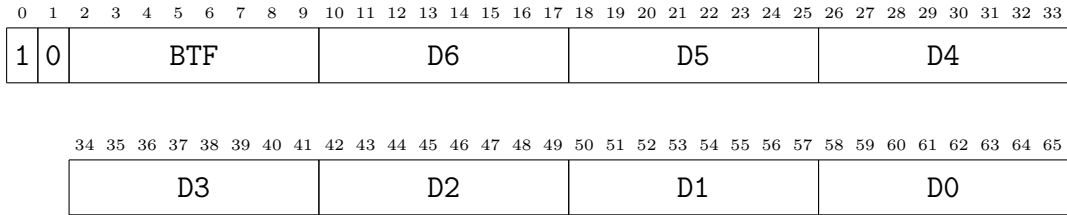


Figure 5: User K-Block structure

3.2.5 Idle block

Figure 6 shows a structure of an *Idle block*. When no data is available for transmission, *Idle* blocks are transmitted instead to maintain the receivers ability to remain in sync and recover the data clock.

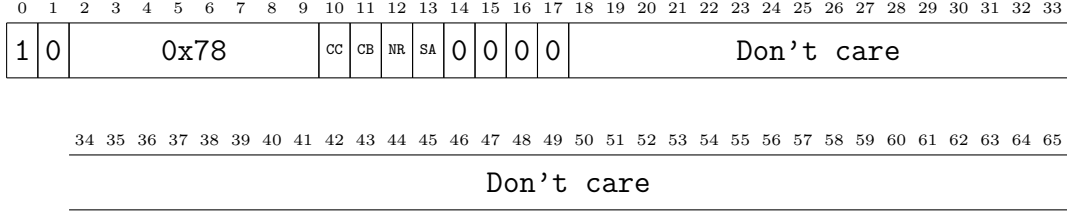


Figure 6: Idle block structure

The *Idle block* contains four flag bits determining the type of the block:

- CC bit indicating that the block is *Clock Compensation* idle,
- CB bit indicating that the block is *Channel Bonding* idle,
- NR bit indicating that the block is a *Not Ready* idle,
- SA bit indicating that the receivers obey the *Strict Alignment* rules.

As the Aurora interface in the FastIC+ is simplex and doesn't implement the functionality described above, all the flag bits in the block are set to zero thus only a *Regular Idle* block is being transmitted.

3.3 Scrambling

Whenever either *Data Block* or *Control Block* is transmitted, the eight octets in the block have to be scrambled with the polynomial shown in Equation 1. Note that the two bit preamble must never be scrambled.

$$P(x) = 1 + x^{39} + x^{58} \quad (1)$$

4 FastIC+ packet structure

4.1 Data packet

Each detection event on each channel of the FastIC+ is represented by a 48 bit data packet containing the digitized ToA and ToT values. The stream of packets is then encoded into Aurora data blocks. A structure of the packet is depicted in Figure 7.

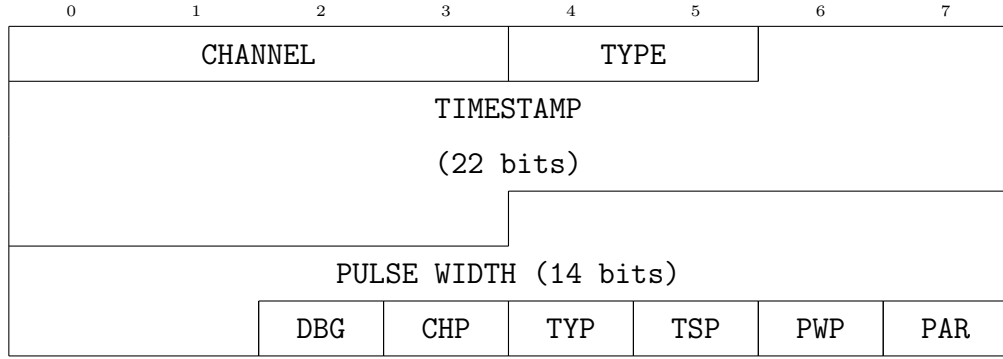


Figure 7: FastIC+ data packet structure

The **CHANNEL** field specifies the number of the channel that the event was detected on (0-7) or the trigger channel (8). The **TYPE** field indicates the packet type where the following types are recognized:

- $ToA + non-linear ToT$ type represented by the value 00,
- $ToA only$ type represented by the value 01,
- $Linear ToT only$ type represented by the value 10,
- $ToA + linear ToT$ type represented by the value 11.

TIMESTAMP and **PULSE WIDTH** fields contain the digitized ToA and ToT values. **DBG** is a flag used in debug mode. The rest of the bits are even parity bits, namely:

- **CHP** parity bit of the **CHANNEL** field,
- **TYP** parity bit of the **TYPE** field,
- **TSP** parity bit of the **TIMESTAMP** field,
- **PWP** parity bit of the **PULSE WIDTH** field,
- and the overall parity bit **PAR** computed from all fields.

4.2 Statistics packet

In addition to the data packets, FastIC+ also periodically transmits a statistics packet that contains basic information about the events. Structure of the packet is shown in Figure 8. This packet is sent onto the Aurora bus in two *K-Blocks* due to the packet size being bigger than the *K-Block* capacity. The first 56 bits are sent in a block with the *BTF* of 0x99, the rest is sent in a second block with the *BTF* of 0x55.

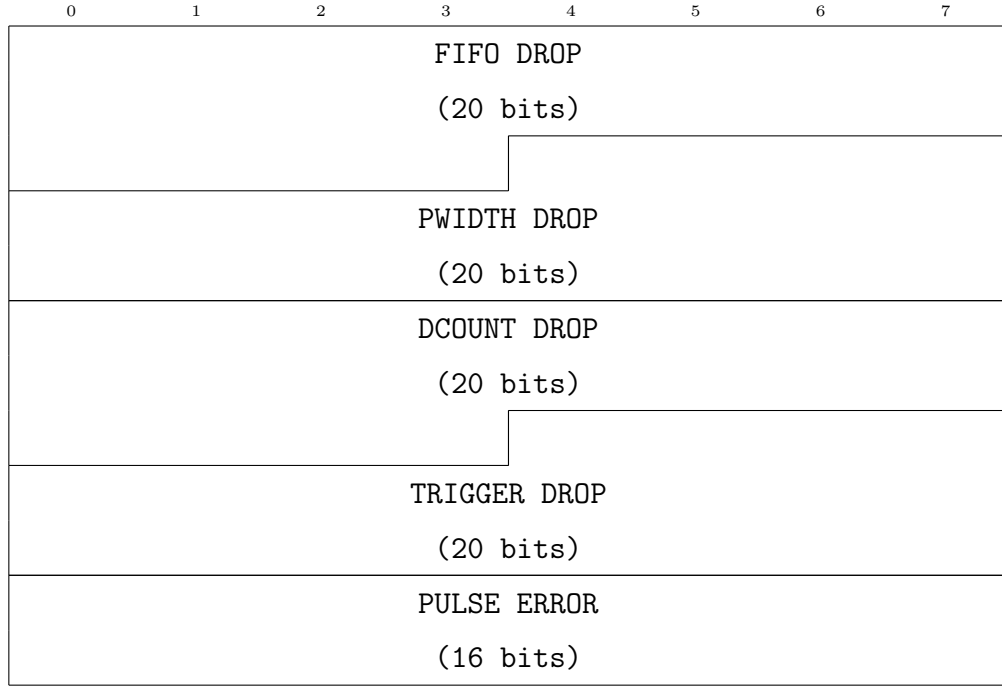


Figure 8: FastIC+ statistics packet structure

The packet consists of the following fields:

- **FIFO DROP** field containing a number of events dropped in the FIFO buffer.
- **PWIDTH DROP** field containing a number of events dropped due to the pulse width being outside of the specified range.
- **DCOUNT DROP** field holding the value of dark counts. This field is valid only if *High-Energy Resolution* mode and *Bandwidth optimization* is enabled.
- **TRIGGER DROP** field holding the number of packets dropped due to external trigger pulse not being validated.
- **PULSE ERROR** field being a malformed pulse counter. Malformed pulse is detected when two consecutive edges are read by the TDC. This may occur due to the limitation of one leading+falling edge per clock period.

4.3 Counter Extension + Event Counter packet

When no detection events are processed during the *Coarse Counter Extension* inactivity period, the timestamp counter would overflow the timestamp field. The following packet, specified in Figure 9, is transmitted periodically so the receiver can base the data timestamp on the most recent coarse counter value, thus drastically increase the dynamic range of the time detection. The packet is encoded in an *K-Block* with the *BTF* of 0xD2.

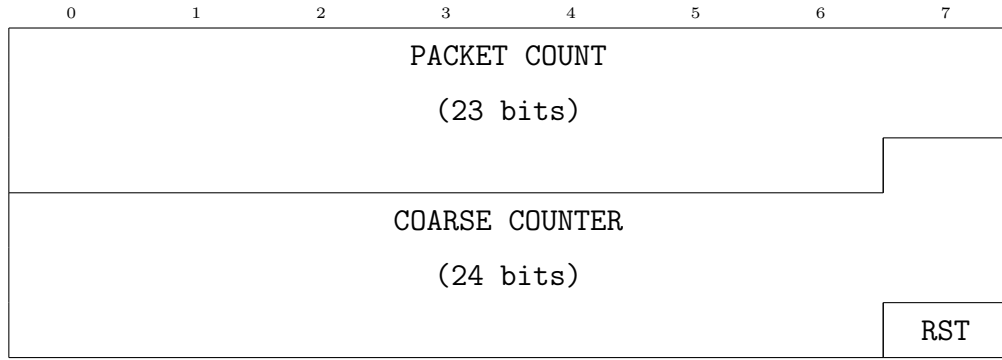


Figure 9: FastIC+ counter extension and event counter packet structure

The packets fields are:

- **PACKET COUNT** field, which holds the number of data packets transmitted since the last reset event.
- **COARSE COUNTER** field holding the timestamp from the coarse counter.
- **RST** field indicating, that the coarse counter has been reset during the coarse counter packet period.

5 Device concept

The ultimate goal was to develop a system utilizing the FastIC+ chips that could be used at CERN to quickly assess measurements, by companies to simplify prototyping with the FastIC+ chips or by schools to use them in experiments. This implied the base requirements for the system to be:

- versatile to allow for both simple and demanding tasks,
- easy to use for both novices and experts,
- standalone all-in-one device requiring no external scientific instruments,
- miniature and portable to be taken anywhere,
- accessible for schools and companies.

To allow for versatility, it was decided that two FastIC+ chips will be used, resulting in a total of 16 detection channels. The trigger inputs and outputs were exposed to the user to allow the synchronization of multiple readout boards. On the other hand, it was decided to read the data from the FastIC+ chips at the lowest possible speed of 80 Mb/s, as it was found sufficient for most of the applications and would greatly simplify the development.

To make the readout easy to use and standalone, USB interface was used to both power the device and transfer the data to a host computer for processing. This in turn required the device to be power efficient and be capable of sufficient USB bandwidth.

Considering all of the above, it was decided to split the device into two parts:

- the readout board containing all the necessary electronics except the sensors,
- the easily replaceable user board containing mainly the sensors.

6 Readout board

In the usual use case, a system integrating the FastIC+ chips would be based on an FPGA with dedicated Aurora receiver, or an FPGA with a custom HDL definition of the receiver. This approach results in easier implementation, as the FPGAs integrate the necessary deserializers and are capable of synchronizing to the data stream and reconstructing the data clock. The disadvantage is that capable enough FPGAs are usually expensive and power hungry, which goes against the requirements of the device. The second thing being, even though Aurora receiver implementation is quite simple, the implementation of other interfaces like USB is complicated or requires expensive IP cores to be purchased. These downsides led to the decision of using a microcontroller for the readout system. When a proper microcontroller is selected all the interfaces as

- ADCs for monitoring voltages,
- DACs for providing voltage feedback,

- SPI and I2C for communicating with other digital devices,
- USB for connection to the host PC,

are implemented in hardware and do not have to be defined in HDL code. A simple firmware in C/C++ can then be programmed to control those interfaces, possibly speeding up and simplifying the development and allowing more users to easily customize the device functionality. The main issue with this approach is, that no microcontroller on the market implements the receivers or deserializers needed to recover the clock from the Aurora stream and synchronize to it, thus, the clock recovery has to be done externally and a suitable alternative peripheral has to be chosen to serve as the receiver.

6.1 Microcontroller

The STM32H753XIH6 was chosen as a great microcontroller candidate for the system. It is based on an Arm Cortex-M7 core running at up to 480 MHz which allows for fast computation required for processing of the two continuous 80 Mb/s streams. It integrates 2 MB of flash memory and 1 MB of RAM which is plenty for buffering the data. From the peripheral side, it supports USB High Speed with a maximum throughput of 480 Mb which is needed for transferring the large amount of data to the host. It also features multiple SPI peripherals supporting input clocks of up to 120 MHz which are an ideal choice for sampling the Aurora stream running at the 80 Mb/s. The internal data buses of the microcontroller are running at half of the core clock and support DMA which also dramatically increases the performance with such a big amount of data. Aside from these main features, the chips implementation of two ADCs, one DAC and I2C for digital communication is useful for the readout as well. The chip is packaged in a compact 14 mm \times 14 mm TFBGA 240+25 package.

6.1.1 Receiving the Aurora stream

As noted above, the most suitable peripheral that can be used for receiving the Aurora stream with this microcontroller is the SPI peripheral. This peripheral is an industry standard serial interface, implemented in all of today MCUs, meant for receiving a serial stream with a dedicated clock. In a simplex slave, receiver only mode, which is suitable for receiving of the Aurora stream, the peripheral exposes the CLK and MOSI pins. The MOSI pin is used for inputting the data stream to the peripheral. The clock, present on the CLK pin, is then used to sample the data stream. The sampled data is then shifted into a register and sent over the internal microcontroller buses for processing. The only issue with using SPI is that the need to recover the Aurora clock persists.

6.1.2 Omitting clock recovery with the FastIC+

As noted in section 2.1, the FastIC+ requires a 40 MHz reference clock to function. The chip features an internal PLL, which synchronizes to this clock and distributes it to other peripherals including the Aurora transmitter. Since the PLL phase is locked to the input clock, the Aurora transmitter, and thus the Aurora output stream, is also locked to the input clock, just delayed by a propagation delay t_{pd} . By measurement, it has been found that this

delay is very stable for temperature range of 0°C - 100°C at a value $t_{pd} = (3.48 \pm 0.08) \text{ ns}$. If this delay is combined with an additional controlled delay, the digital stream can be aligned such that the data can be sampled with an 80 MHz sampling clock that is in phase with the reference clock.

The delay can be implemented by a variable delay line. However it turns out that this delay in combination with the typical propagation delay of a common LVDS to CMOS receiver equals approximately 9 ns. The Aurora data stream is double data rate, meaning that a valid symbol is transferred on both rising and falling edge of the reference clock. This results in the symbol duration of 12.5 ns. The 9 ns delay, being almost exactly 3/4 of the symbol duration, shifts the data such that both sampling clock edges are contained in the data symbol as seen in figure 10. Thus the data symbol can be sampled by either the rising or falling edge. Since the FastIC+ outputs an SLVS stream and a MCU with CMOS inputs is used to capture the stream, this converter has to be in place anyways and if carefully chosen, it can at the same time be used as the delay line to achieve the correct sampling clock phase.

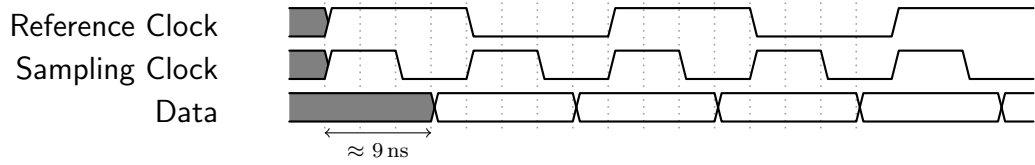


Figure 10: Alignment of the sampling clock and data stream

6.2 FastIC+

The FastIC+ requires almost no additional components aside from decoupling of the power domains. The XVDD (PLL) power domain has been isolated from the TVDD (threshold circuitry) by a ferrite bead, to reduce noise coupling between the two. However, the PLL is only expected to produce noise at startup, while the threshold circuitry is used only after the PLL has locked, so any noise coupling between the two domains should have little to no impact.

To increase the stability of the internal band gap reference, a 10 nF capacitor has been added to the VBG pin.

Both **nRST** (reset of the FastIC+) and **SRST** (reset of the synchronous counter) have been pulled up to the digital supply so that the microcontroller pins in open drain mode can be used to control these pins without the need for voltage translation.

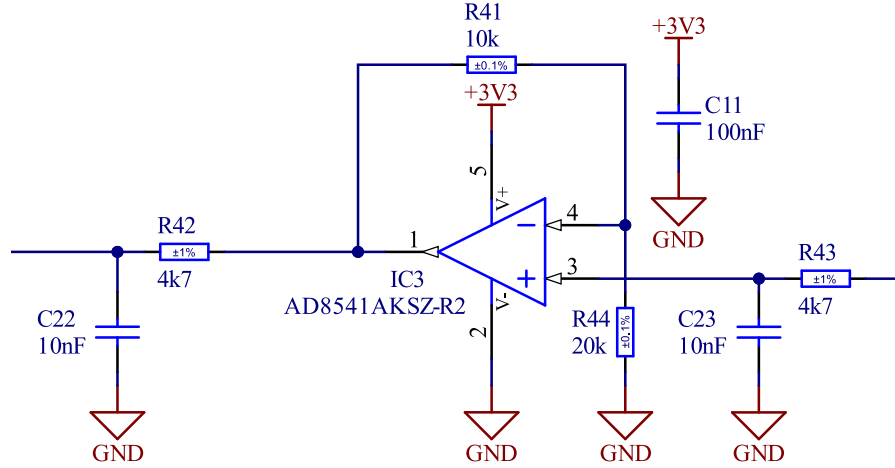


Figure 12: Schematic of the voltage monitoring amplifier

6.2.4 High speed outputs

The FastIC+ offers multiple high speed SLVS outputs. For each channel, a differential output is present capable of transmitting a pulse whose beginning timestamps the ToA of a photon and length resembles the ToT. However, as the readout uses the data digitalized by the internal TDC, these channels can be left unconnected and disabled in the chip.

The **TIME** output can either be used to generate a digital pulse whenever a pulse is received on any of the channels or can be internally connected to the trigger comparator and used for trigger calibration routine, the latter being used in this case. When the pin is not used for calibration, it should be disabled to reduce any EMI generated by the high speed edges. The **TDCOUT** is the output of the Aurora stream from the transmitter.

Both of the above mentioned pins are converted to a CMOS signal using the DS90LVRA2 dual channel differential line receiver. This receiver has been chosen specifically for its small size, high enough speed but also for its typical propagation delay $t_{pd} = 4.4\text{ns}$ to correctly align the data to the sampling clock.

6.2.5 Trigger input and output

The trigger inputs (used for externally triggering the ASIC) and outputs (outputting the signal from the internal comparator) have been exposed to the user on two SMA connectors. Termination has been placed on these to mitigate any reflections and ESD protection diodes have been added to protect the chip from ESD events. It's important to note that the ESD diodes add capacitance to the trigger lines, thus degrading (slowing) the trigger edge and introducing a slight delay in the trigger. This either needs to be accounted for when using the readout or the diodes need to be left unassembled at the expense of worse ESD immunity.

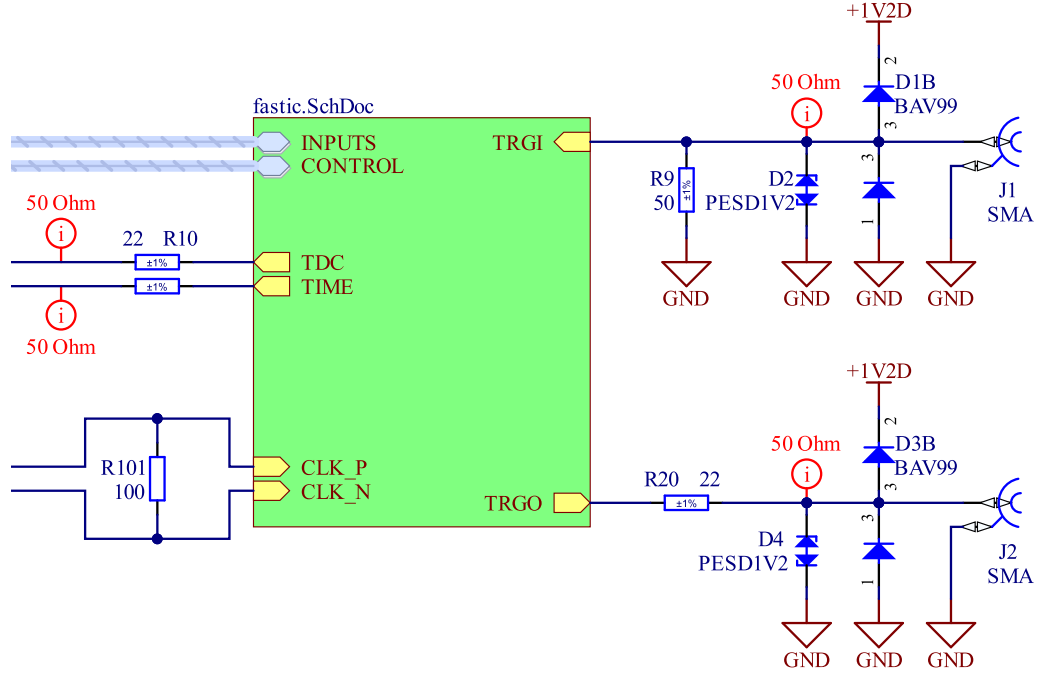


Figure 13: Schematic of the FastIC+ trigger circuitry

6.3 USB

To supply power to the device and handle communication with the host computer, a USB type C connector has been used.

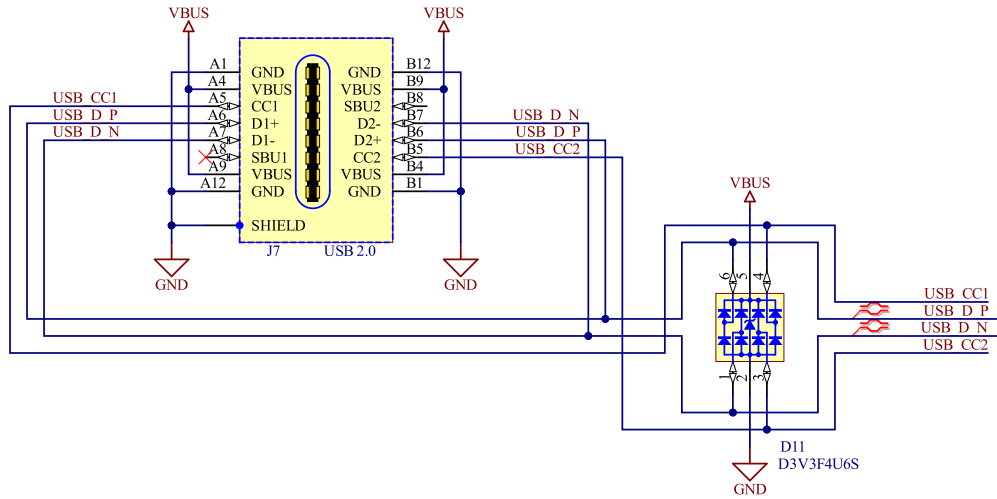


Figure 14: Schematic of the USB connector with ESD protection

6.3.1 External PHY

Because the combined data rate of the FastIC+ chips of 160 Mb/s exceeds the USB 1.1 specification, a USB 2.0 (High Speed) was implemented, supporting up to 480 Mb/s throughput. Unfortunately, the used microcontroller does not integrate the USB 2.0 PHY directly. Instead, it only integrates the necessary logic and interfaces with an external PHY via the ULPI interface.

A USB3320 interface was chosen as it is well supported by the microcontroller and widely used in countless designs. A 48 MHz crystal oscillator has been used as the external reference clock required for the device to operate. Different crystal frequency selection has been made possible with 0Ω resistor jumpers. The ULPI connections have been series terminated, to suppress any possible reflections caused by poor impedance matching.

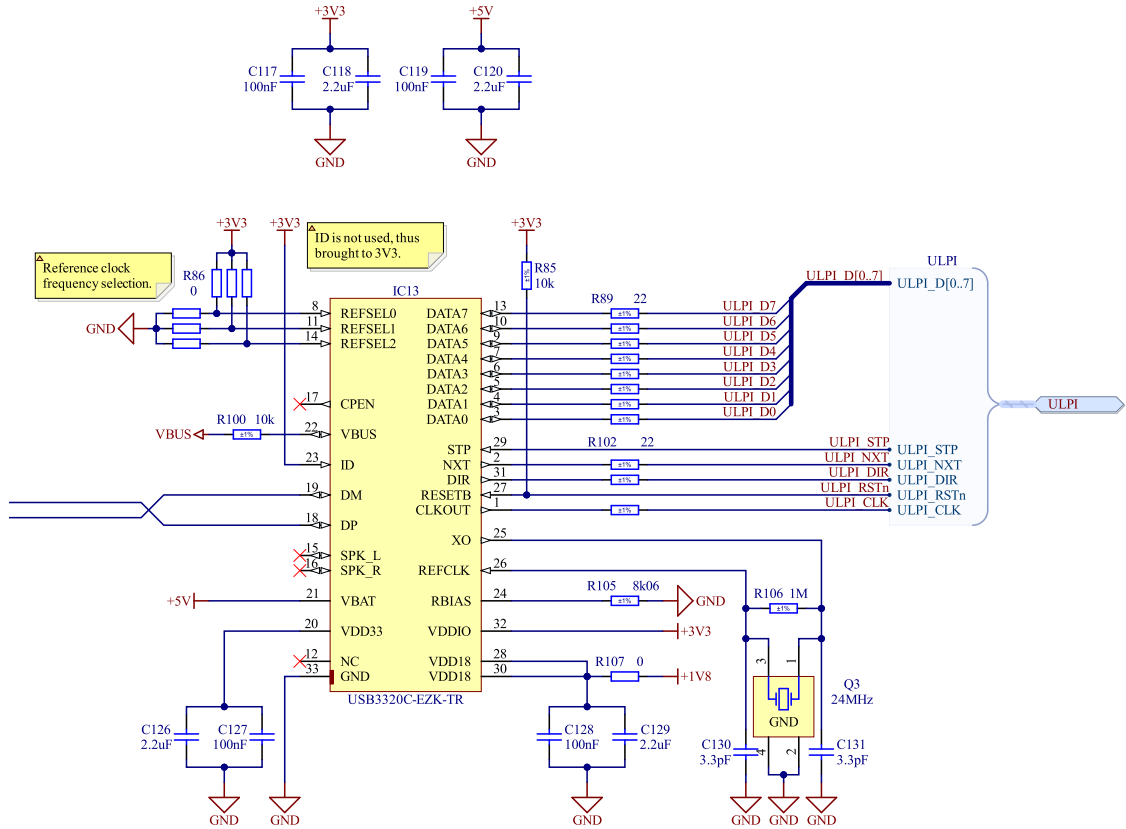


Figure 15: Schematic of the USB

6.3.2 Power Delivery

A FUSB302 chip was added to allow for power limit negotiation with a UCPD capable power source. The chip handles all the necessary signaling and communicates with the MCU via a I2C interface. Optional $5.1\text{ k}\Omega$ resistors have been added to passively negotiate the highest power limit, if it would be decided to omit the power delivery functionality and not assemble the FUSB302.

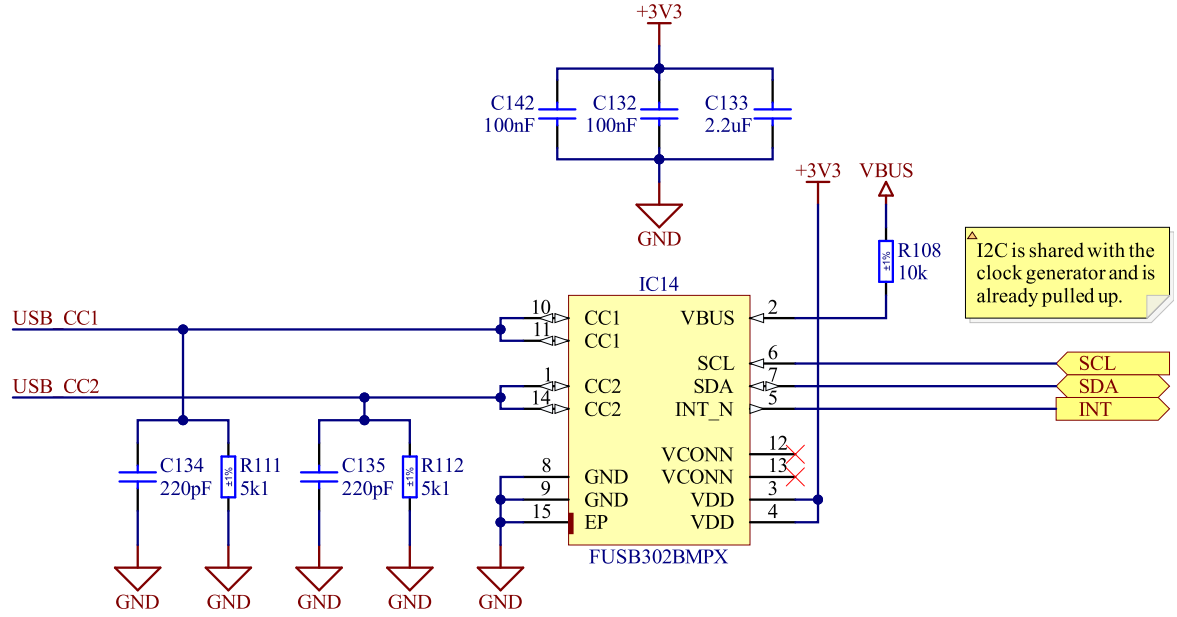


Figure 16: Schematic of the USB

6.4 Clock generation

For generating the 40 MHz reference clock for both of the FastIC+ chips and the 80 MHz sampling clock for the SPI peripherals, a SI5340 has been used. It is a four output clock generator with 90 fs RMS jitter, supporting both CMOS and differential output. Additionally, each output can be powered by an independent power supply allowing for mixing the 3.3 V CMOS signals for the microcontroller and 1.8 V LVDS signals for the FastIC+. A 48 MHz crystal oscillator has been used as the reference clock. The chip features both I2C and SPI interface for configuration. The I2C was selected to be used in this design.

Unfortunately, the FastIC+ implements SLVS instead of LVDS, which uses voltages down to 1.2 V, not supported by the generator. Because of this, a divider on the differential had to be implemented. This circuit was provided by the chip designers themselves and has been proven to work reliably, thus is not explained in detail in this document.

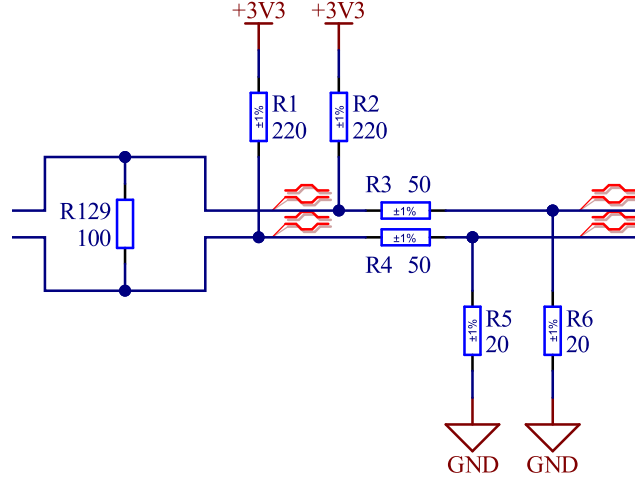


Figure 18: Divider for the LVDS to SLVS conversion

6.5 High Voltage

All the usual sensors like SiPMs, PMTs or MCPs need high voltage biasing for their function. To eliminate the need for an external HV supply, an internal one has been implemented using the LT3571. This DC/DC converter, intended for biasing of avalanche photodiodes, is capable of generating up to 70 V 2 mA output from a low voltage input. It fully integrates the power switch and regulation along with soft-start and variable switching frequency.

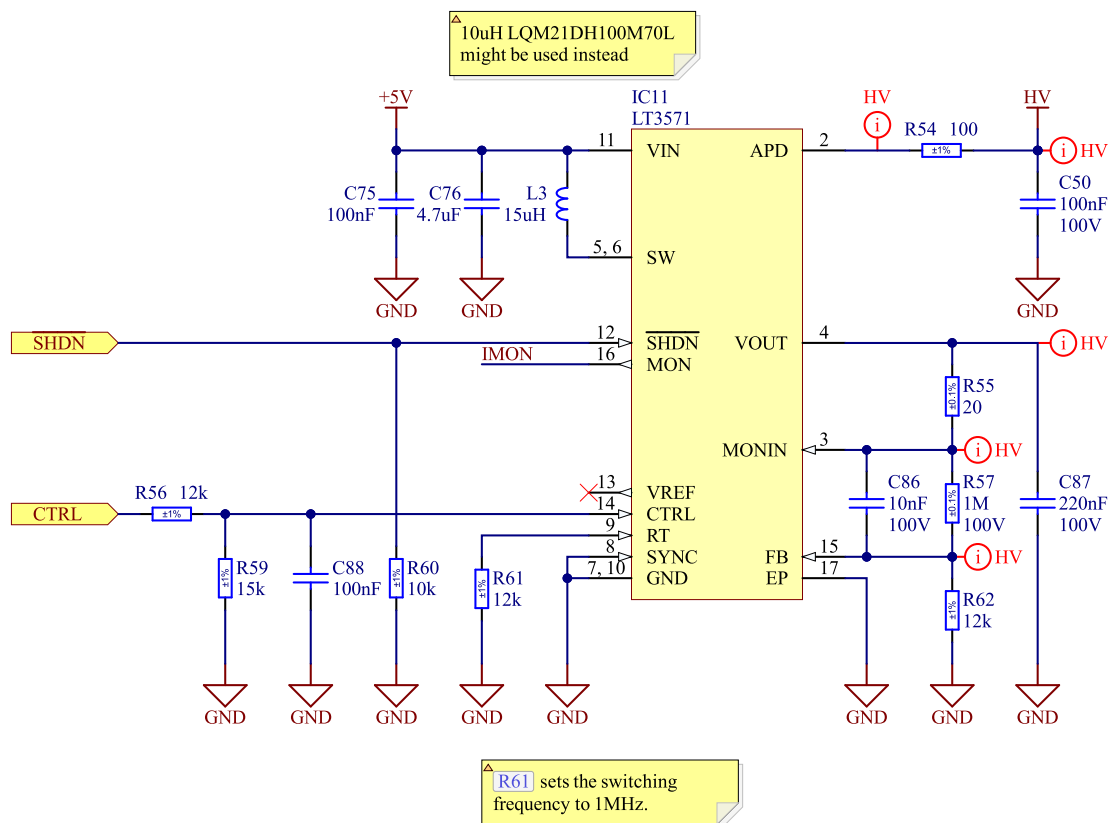


Figure 19: High voltage power supply schematic

A CTRL input is available for adjusting the output with a control voltage. This reference is generated by the MCU DAC and divided by a voltage divider to a suitable 0 V - 1 V range.

To allow for closed loop control with the microcontroller, the output voltage on the APD pin is monitored via a suitable voltage divider with an operational amplifier buffer. For current monitoring, the LT3571 offers the IMON pin outputting a current proportional to the one out of the APD pin. This current is then converted to voltage with a 1 k Ω shunt and buffered with an operational amplifier.

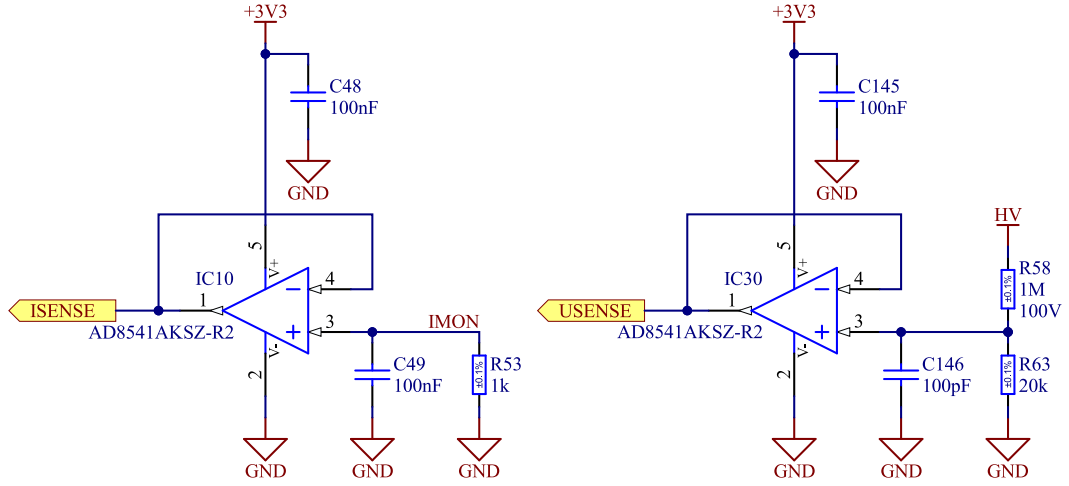


Figure 20: Sensing of the high voltage and current

6.6 Power

A DC/DC converter has been implemented to efficiently lower the 5 V input voltage to the 3.3 V used by the microcontroller. The 3.3 V for the analog domain has been generated with a low ripple LDO. The 1.8 V domain for the USB interface has been derived from the 3.3 V using an LDO as well, as the low power consumption will not cause too much of a power loss.

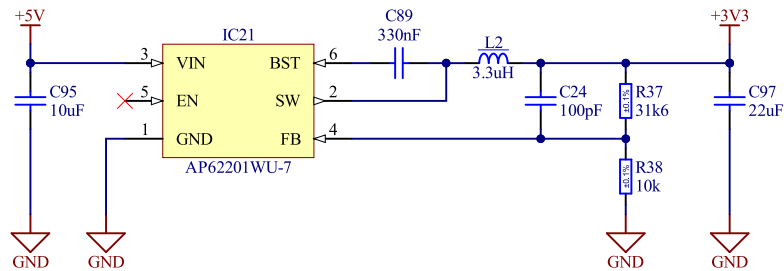


Figure 21: Step down regulator schematic

A power sequencing has been implemented for all the three FastIC+ domains. First, the digital domain supply is activated, followed by the supply for the threshold circuitry and PLL and lastly, the analog domain is supplied. All of these domains are derived from the 3.3 V using a 1.2 V LDOs which are sufficient for the low power consumption of the FastIC+ chips.

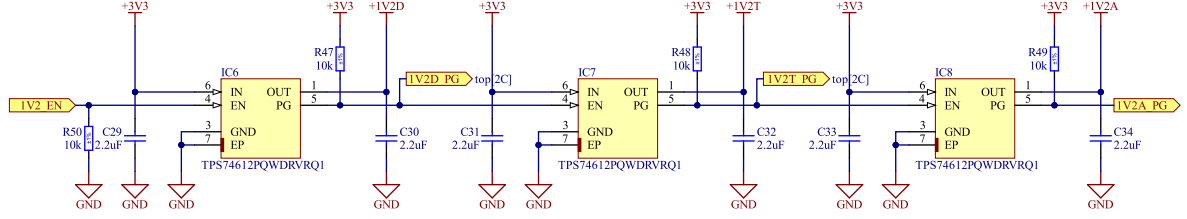


Figure 22: FastIC+ power domain sequencing

6.7 Connector

For interfacing with the user board, the 9 mm high version of ERM8-020 has been used for its durability and suitable pin count. This connector carries all the sixteen input channels alongside the high voltage biasing supply. The 3.3 V supply is also exposed and four pins are dedicated to the user board identification.

6.7.1 Identification pins

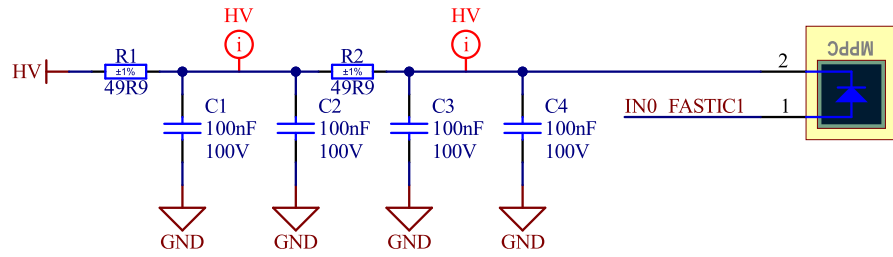
The identification pins serve as an easy way to assign a four bit ID to a specific user board. This ID can then be used by the software to load a configuration preset defined for the user board. For advanced cases, the pins ID0 and ID1 are used as I2C communication lines. A compatible EEPROM can then be assembled on the user board, saving the full configuration on the user board itself, as well as additional data like name. If the I2C interface is to be used, all the ID pins need to be pulled up high by a suitable resistor. At least one of the ID pins has to be high at all times, this means that an ID of 0b0000 is not allowed and in this case, the readout will not recognize a valid user board.

7 User Board

The user board has been developed as a template for the users to get easily started with the readout system. It contains a matrix of 4×4 SiPM sensors as well as an EEPROM for storing the board configuration. The 9 mm high ERF8-020 connector has been chosen for the user board to match the counterpart present on the readout and allow enough clearance between the assembled readout board and user board.

7.1 Sensors

The footprint for a typical generic THT SiMPs has been implemented on the board. Each SiPM is powered from the HV plane and the input is filtered with a dual RC low pass to eliminate a possibility of cross triggering of the sensors.



7.2 EEPROM

The 24AA04T-I/OT 4 kb EEPROM has been used on the user board for the configuration storage. It has been selected for the low price and sufficient capacity.

8 Summary

The two concepts described above were designed and manufactured keeping in mind the required performance. The final PCBs can be seen on the images bellow.

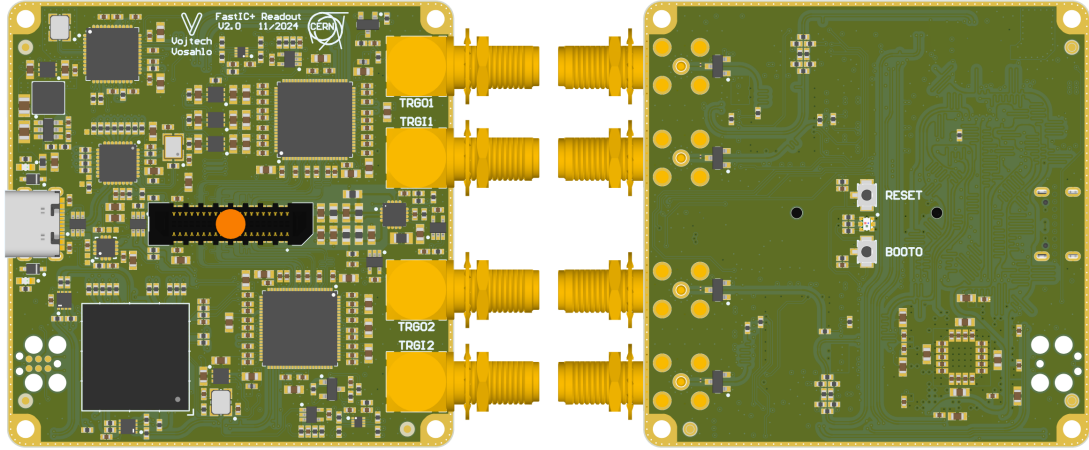


Figure 23: The readout PCB

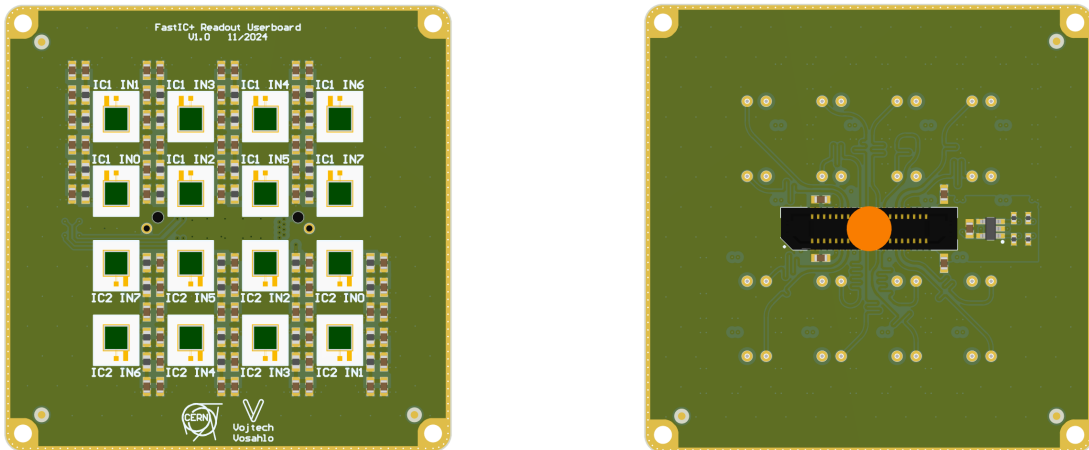


Figure 24: The user board PCB

References

- [1] Aurora 64b/66b protocol specification. https://docs.xilinx.com/v/u/en-US/aurora_64b66b_protocol_spec_sp011, 2014.