Preliminary draft 11:57 27 November 2024

27 November 2024
vojtech.vosahlo@cern.ch

# FastIC+ readout concept 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 Aurora protocol, it's packet structure and the structure of the FastIC+ packets. Afterwards, 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

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 Device concept

The ultimate goal was to develop a system utilizing the FastIC+ chips that could be used at CERN to quickly assess the required 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 other 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 readout channels. Also, the trigger inputs and outputs were exposed to the user to allow the synchronization of multiple readout boards. However, it was also decided to read the data from the FastIC+ chips at the lowest possible speed of 80 Mb/s, as it was found sufficent for most of the applications and would greatly simplify the development.

To make the readout easy to use and standalone, it was decided to use a USB interface to both power the device and transfer the data to a host computer for processing. This also implied other requirements for power efficency and high enough bandwith of the connection.

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,
- an easily replacable and customizable userboard containing mainly the sensors, to allow for quick sensor changes on the go or easy custom implementations for the sensor arrays.

# 3 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 implementation of the receiving logic to receive the aurora stream. 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 quiet 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 than 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 i, 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.

## 3.1 Microcontroller

The STM32H753XIH6 was chosen as a great microcontroller candidate for the system. It is based on a 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 at the 80 Mb. 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 aswell. The chip is packaged in a compact 14 mm × 14 mm TFBGA 240+25 package.

### 3.1.1 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 than used to sample the data stream. The sampled data is than shifted into a register and send over the internal microcontroller buses for processing. The only issue with using SPI is that the need to recover the Aurora clock persists.

### 3.1.2 Omitting clock recovery with the FastIC+

As noted in section XX, 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 $t_{pd} = (3.48 \pm 0.08)$ ns. If this delay is combined with an additional controlled delay, the digital stream can be correctly aligned such as that it could be sampled by a clock that is in phase with the FastIC+ 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. Considering that the period of the 80 Mb signal is 12.5 ns, the 9 ns shift aligns the waveform almost perfectly in the middle of the data pulse (even accounting for the propagation delay uncertainty). 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.

## 3.2 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 (treshold 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 treshold 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 pin without the need for voltage translation.

### 3.2.1 I2C communication

As the FastIC+ voltage domains all run at 1.2 V a voltage shifter had to be implemented for communication with the microcontroller over I2C. For this, the PCA9306DQER has been chosen for its miniature X2SON package and 400 kHz speed.

### 3.2.2 Calibration pulse generator

The FastIC+ features a `CAL` input pin, used for injecting a current test pulse into any of the eight input channels. the pin is internally connected to ground with a 70 Ω resistor to convert the current into voltage. The usual shape of such a pulse should resemble a pulse created by a real sensor, thus a decaying exponential with an amplitude of a few milliamperes and length

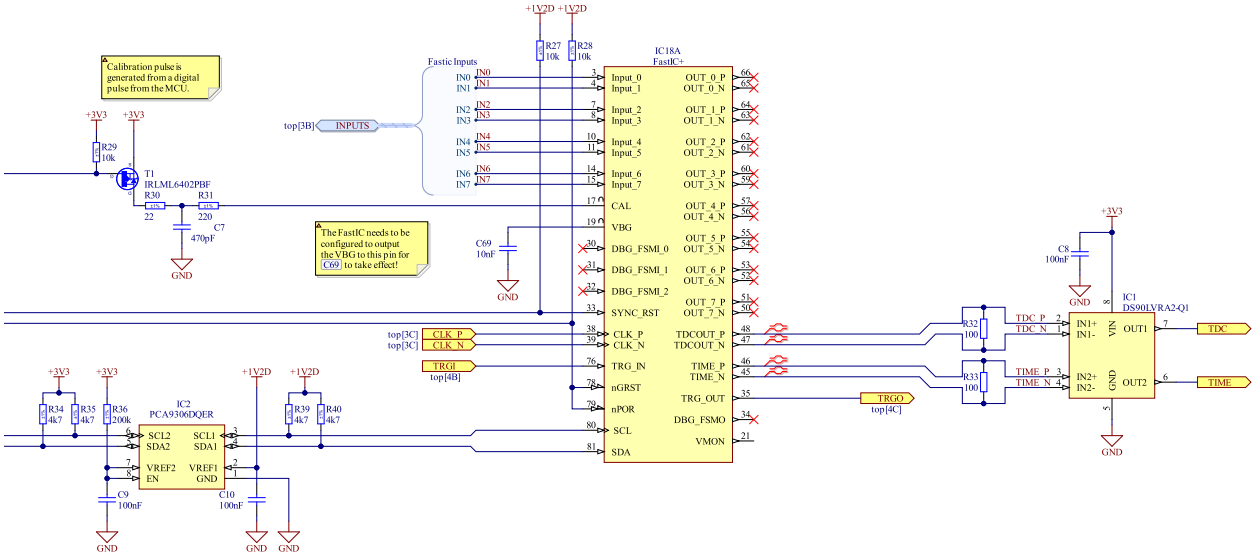


Figure 1: Schematic of the FastIC+

under a microsecond. To create this pulse, a simple high side switch has been implemented with series resistance to limit the current into the `CAL` pin and parallel capacitance to recreate the decaying exponential. The transistor gate is driven by a quick digital pulse from the microcontroller, whos length can be adjusted to adjust the current pulse width and by some degree also the amplitude.

### 3.2.3 Voltage monitoring

The `VMON` pin on the ASIC serves for monitoring of the internal analog tresholds and DAC outputs. Since the microcontrollers internal voltage reference has been selected to run at 1.8 V, a simple non inverting amplifier has been implemented to amplify the 1.2 V output to the microcontroller full-scale range and thus improve the performance. Because the `VMON` output is used only for treshold monitoring, thus only DC voltages, a slow RC filter has been used to reduce the noise coupled to the analog signal.

### 3.2.4 High speed outputs

The FastIC+ offers multiple high speed SLVS outputs. For each channel, a differential output is present capable of transmitting a pulse whos 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.

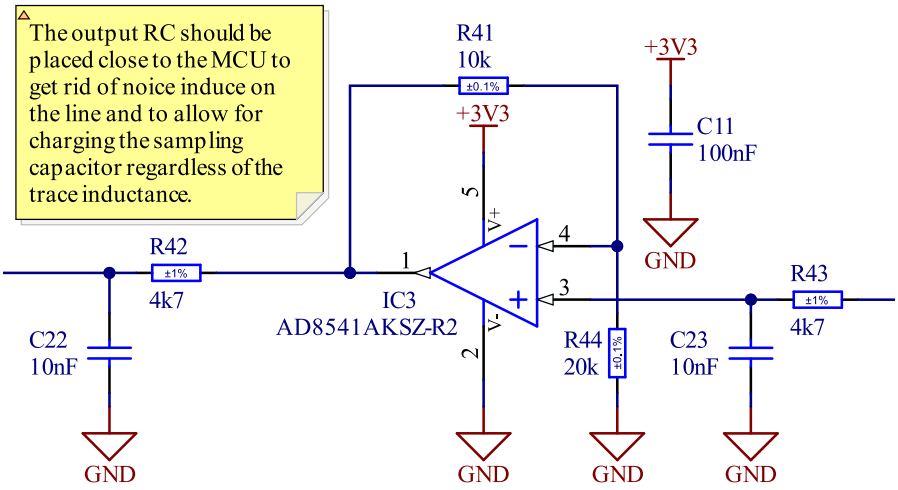


Figure 2: Schematic of the voltage monitoring amplifier

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 most importantly for its typical propagation delay $t_{pd} = 4.4\,\text{ns}$ which is very close to the measured FastIC+ propagation delay and thus the receiver can itself be used as a delay line for the signal.

### 3.2.5 Trigger input and output

The trigger input (used for externally triggering the ASIC) and output (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.

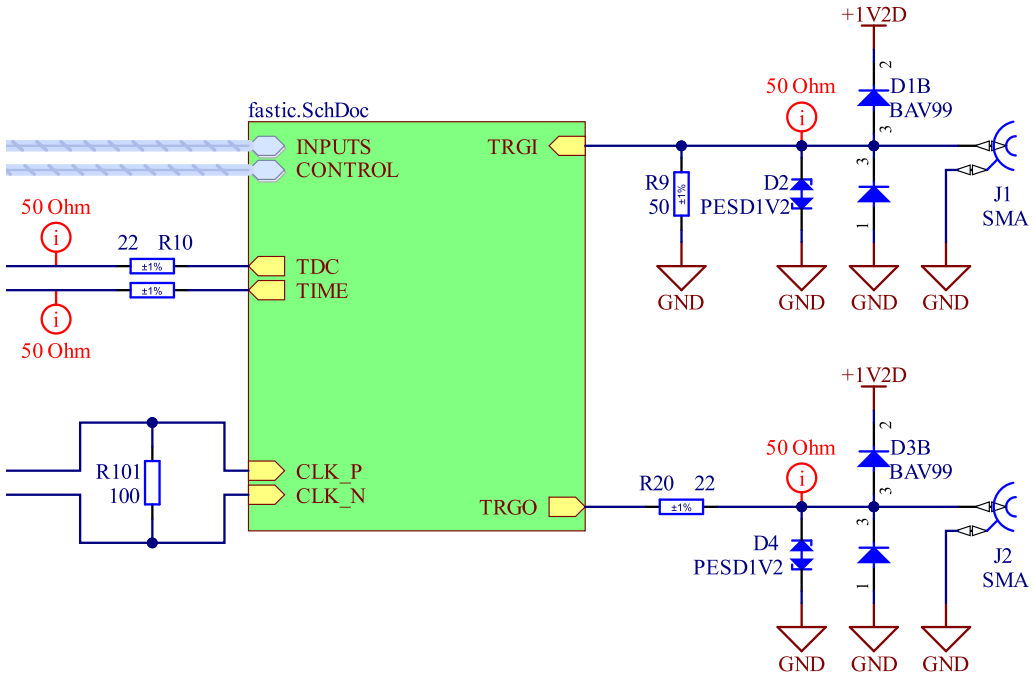


Figure 3: Schematic of the FastIC+ trigger circuitry

## 3.3 USB

### 3.3.1 USB HS Interface

### 3.3.2 power Delivery

## 3.4 Clock generation

## 3.5 High Voltage

## 3.6 Power

## 3.7 Connector

### 3.7.1 Identification pins

# 4 User Board

## 4.1 Sensors

## 4.2 Connector

### 4.2.1 Identification pins

# References