Chapter 7

The Use of FPGA in Drift Chambers for High Energy Physics Experiments

Gianluigi Chiarello, Claudio Chiri,
Giuseppe Cocciolo, Alessandro Corvaglia,
Francesco Grancagnolo, Marco Panareo,
Aurora Pepino and Giovanni Francesco Tassielli

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/66853

Abstract

In this chapter, we describe the design of a field programmable gate array (FPGA) board capable of acquiring the information coming from a fast digitization of the signals generated in a drift chambers. The digitized signals are analyzed using an *ad hoc* real-time algorithm implemented in the FPGA in order to reduce the data throughput coming from the particle detector.

Keywords: drift chamber, FPGA, CluTim, impact parameter

1. Introduction

A drift chamber (DC) is a detector used in high energy physics experiments for determining charged particle trajectories. It consists of a gas volume and of an array of thin wires at high voltages generating high electric fields. Charged particles passing through the gas ionize it creating electron/ion pairs along their path [1], which, accelerated by the electric fields, produce signal pulses on the wires. The signal pulses from all the wires are then collected and the particle trajectory is tracked assuming that the distances of closest approach (the impact parameter) between the particle trajectory and the wires coincide with the distance between the closest ion cluster and the corresponding nearest wire [1, 2]. The widespread use of light, helium-based gas mixtures, aimed at minimizing the multiple scattering contribution to the momentum measurement for low momentum particles, produces, as a consequence, a low ionization clusters density (12 cluster/cm in a 90/10 helium/isobutane mixture) [3], thus

introducing a sensible bias in the impact parameter assumption, particularly for short impact parameters and small drift cell [4]. Recently, it has been proposed an alternative track reconstruction (cluster counting/timing) technique, which consists in measuring the arrival times on the wires of each individual ionization cluster and combining these times to get a bias-free estimate of the impact parameter [5, 6]. Typical time separations between consecutive ionization acts, in a helium-based gas mixture, range from a few ns, at small impact parameters, to a few tens of ns, at large impact parameters [7, 8]. Therefore, in order to efficiently applying the cluster timing technique, it is necessary to have readout interfaces [9, 10] capable of processing high speed signals, in which one can easily isolate pulses due to different ionization cluster.

1.1. Hardware

The wire signals generated by the drift chamber, before being processed, are converted from analog to digital with the use of an analog-to-digital converter (ADC). Requirements on drift chamber performance impose the conversions at sampling frequencies of at least 1 GS/s with at least 8-bit resolution. These constraints, together with the maximum drift times, usually of the order of 1 microsecond, and with the large number of acquisition channels, typically of the order of tens of thousands, mandate some sizeable data reduction, which, however, must preserve all the relevant information. Identifying both the amplitude and the arrival time of each peak associated with each individual ionization cluster is the minimum requirement on the data transfer for storage [5, 6].

This chapter deals with the possibility of using FPGAs for the real-time analysis of the data generated by a drift chamber [11] and successively converted by an ADC (**Figure 1**).

More specifically, a fast readout algorithm (CluTim [12]) for identifying, in the digitized drift chamber signals, the individual ionization pulse peaks and recording their time and amplitude has been developed as VHDL/Verilog code implemented on a Xilinx ML605 Evaluation board [13] shown in **Figure 2**, making use of a Virtex 6 FPGA.

In particular, the used device is a Virtex-6 XC6VLX240T-1 FFG1156 [14] that allows for a maximum input/output clock switching frequency of 710 MHz. The hardware setup includes also an evaluation board AD9625-2.0EBZ [15] shown in **Figure 3** with a pipeline ADC.

The analog-to-digital converter (ADC) used is an AD9625 [16], a 12-bit monolithic sampling ADC that operates at conversion rates of up to 2.0 GSPS, with 3.48 W power dissipation.

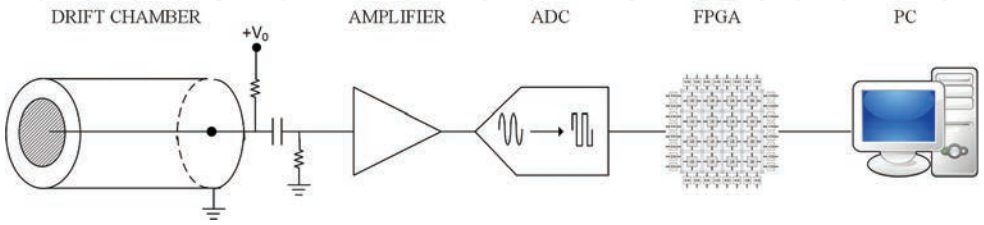

Figure 1. Channel setup.

http://dx.doi.org/10.5772/66853

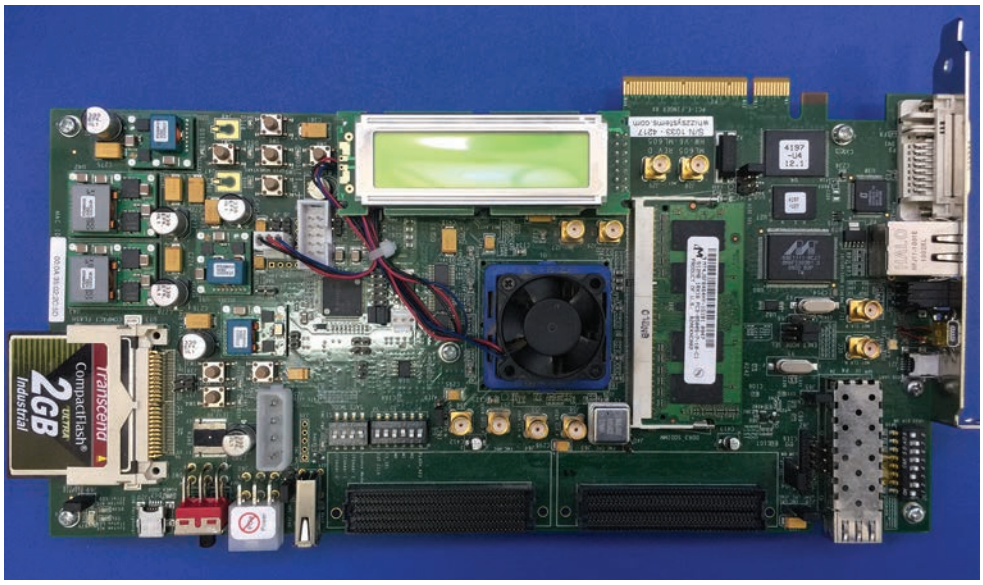

Figure 2. Xilinx ML605 Evaluation Board.

This product is designed for sampling wide bandwidth analog signals up to the second Nyquist zone. The combination of wide input bandwidth, high sampling rate, and excellent linearity of the AD9625 is ideally suited for data acquisition systems, and for the purpose of the experiment.

The analog input clock signals are differential. The standard output used is JESD204B-based high speed serialized output [17] that is configurable in a variety of one-, two-, four-, six-, or eight-lane configurations.

The ADC configuration (sampling frequency, number output lines, power control, etc.) is set with three single-ended lines (clock, date, and enable) that configure the internal register of an SPI. The signals on these lines are managed from the VIRTEX, where a VHDL script generating the bits stream to configure the SPI register, the enable signal and the clock is implemented, and sent to the ADC.

The AD9625 digital output complies with the JEDEC Standard No. JESD204B, serial interface for data converters.

JESD204B is a protocol linking the AD9625 to a digital processing device over a serial interface up to link speeds in excess of 6.5 Gbps. The benefits of the JESD204B interface over LVDS include a reduction in the required board area for data interface routing and enabling smaller packages for converters and logic devices.

The JESD204B data transmit block assembles the parallel data from the ADC into frames and uses 8-bit/10-bit encoding as well as optional scrambling to form serial output data. Lane

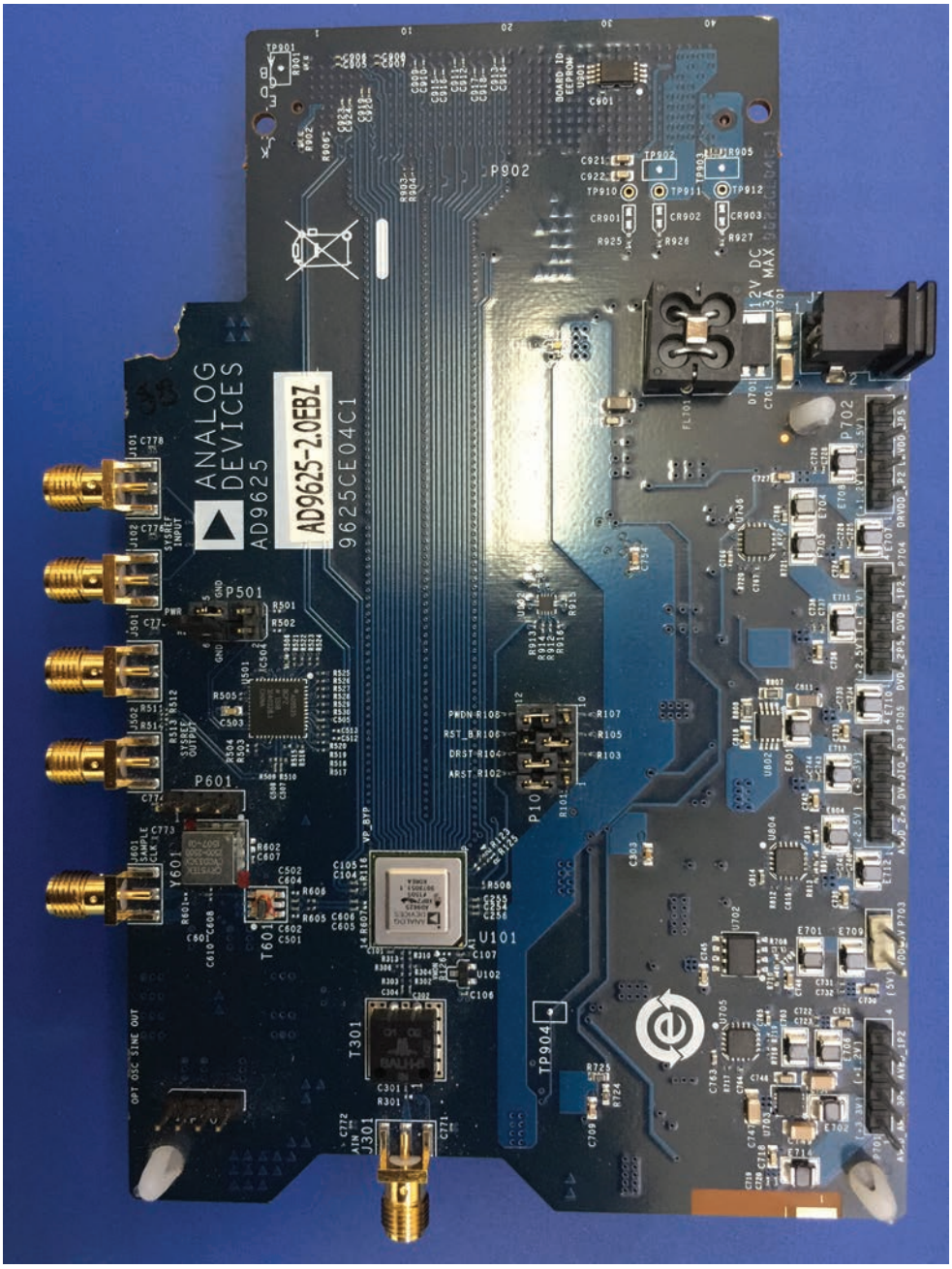

Figure 3. AD9625-2.0EBZ Evaluation Board.

http://dx.doi.org/10.5772/66853

synchronization is supported using special characters during the initial establishment of the link. Additional data that are used to maintain synchronization are embedded in the data stream thereafter. A JESD204B receiver (the FPGA) is required to complete the serial link.

The AD9625 JESD204B transmits block maps to two digital down converters for the outputs of the ADC over a link.

A link can be configured to use up to eight JESD204B lanes. The JESD204B specification refers to a number of parameters to define the link, and these parameters must match between the JESD204B transmitter and receiver.

The JESD204B protocol stack consists of seven functional blocks in the transmit path and seven functional blocks in the receive path, as shown in **Figure 4**.

Xilinx offers a complete working solution that simplifies the adoption of JESD204B [17], including best-in-class serial transceivers, IP, design tools, reference designs, and ecosystem partners.

- **GTH transceiver architecture:** Reducing the BER in serial transceiver transmissions directly correlates to the peak bandwidth that can be achieved through the serial link. Xilinx FPGAs offer industry-leading jitter performance, achieved through a unique capability called adaptive receiver equalization. This improves the performance of the decision feedback equalizer (DFE) block, significantly reducing BER.
- **Xilinx JESD204B IP core:** Xilinx offers the industry's first fully JESD204B-compliant IP core for programmable logic. This core supports the full JESD204B bandwidth specification of 12.5 Gbps over one to eight lanes. It can be configured as both a transmitter for interfacing to a DAC and as a receiver for interfacing to an ADC. The core also includes support for scrambling and initial lane alignment.
- **JESD204B reference designs:** Complete JESD204B reference designs are provided for Xilinx development boards by a variety of third-party analog vendors, such as analog devices.

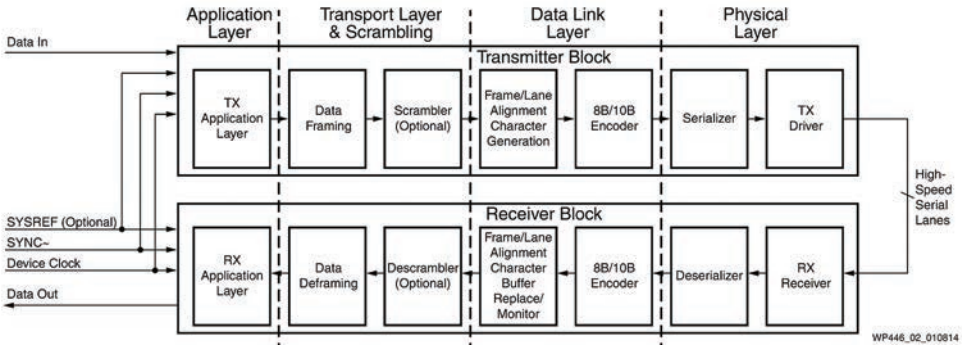

Figure 4. JESD204B protocol.

The two evaluation boards are connected by a high-speed VITA 57 Mezzanine Connector in order to limit parasitic effects due to pin couplings.

The FPGA Mezzanine Card (FMC) standard has proven to be highly popular with over 100 total FMC cards now available from a variety of partners. Over 30 of these FMCs specifically support high-speed data converters. The FMC provides a way for customers to quickly configure their standard Xilinx development boards with real-world analog interfaces.

Xilinx partners have been providing many easy-to-use (and to re-use) reference designs that save customers weeks or even months of development time. Building on this success are the first high-speed analog FMC cards supporting JESD204B from industry-leading analog providers such as analog devices, IDT, 4DSP, NXP, and others.

1.2. Software

The Xilinx ISE 14.5 software has been used to design, develop, and test the CluTim algorithm. It allows for the analysis and synthesis of source code written in a hardware description language (HDL) such as Verilog and VHDL, provides designs, and is also able:

- **To perform timing analysis:** To provide a detailed analysis of the FPGA design. This ensures that the specified timing constraints are properly passed to the implementation tools.
- **To examine RTL diagrams:** After the HDL synthesis phase of the synthesis process, it is possible to show a schematic representation of the synthesized source file. This schematic shows a representation of the preoptimized design in terms of generic symbols, allowing for control of design issues early in the design process.
- **To simulate a design:** ISE simulator (ISim) provides a complete, full-featured HDL simulator integrated within ISE, with which is possible to perform waveform tracing, waveform viewing, and HDL source debugging.
- **To configure the target device**: It is possible to program the device with a JTAG programmer.
- **To provide IP cor**e: IP (intellectual property) core is a block of logic or data that is used in an FPGA. It is part of the growing electronic design automation (EDA) industry trend toward repeated use of previously designed components. Ideally, an IP core should be entirely portable—that is, must be easy to insert into any vendor technology or design methodology.

The real data can be stored and visualized using the Chip Scope PRO software. It inserts logic analyzer, system analyzer, and virtual I/O low-profile software cores directly into design, allowing them to view any internal signal or node, including embedded hard or soft processors. Signals are captured in the system at the speed of operation and brought out through the programming interface, freeing up pins for the design. Captured signals are then displayed and analyzed using the ChipScope Pro analyzer tool. These signals can be also saved to be processed with other tools, i.e., MATLAB.

http://dx.doi.org/10.5772/66853

2. Algorithm description and FPGA implementation

The chosen hardware involves the use of an FPGA device in which 20 transceiver are implemented. This means that one can use a single FPGA to connect multiple ADC in parallel, up to three, if one adopts a configuration which uses six out of the eight lines at high speeds, configurations with fewer lines at the maximum sampling frequency are also possible, but this requires a digital-down-conversion (DDC) of the signal, implemented internally to the ADC, which cannot be applied in the described case, since it acts like a filter to the signal which, if not limited to a narrow band, rather than harnessing the entire band up to 1 GHz (as in the described case), results in loss of information.

The ability to connect multiple devices in parallel has the effect of having a very large number of data to store; if the data from the ADC were stored without preprocessing to perform a reduction and store only those useful for experimental purposes would involve several disadvantages, as follows:

- A time window of observation of the event greatly reduced, because the internal memory to the FPGA would be filled very quickly.
- A very long data transfer time from FPGA to PC for their postprocessing, due to the very large amount of data to be transferred.
- A very long time to postprocess data.

All of these issues require a real-time data preprocessing to store only useful data.

The CluTim algorithm [12], here described, is able to process the data in real time. In particular, it

- identifies, in the digitized signal, the peaks corresponding to the different ionization electrons.
- stores each peak amplitude and timing in an internal memory.
- sends the data stored to an external device when specific trigger signals occur.

The ability of an FPGA to perform multiple operations in parallel turns out to be useful having to manage a large amount of data coming from the ADC at a very high rate.

In fact, in this way, one can execute on different data, at the same time, multiple instructions performing the same function.

Before the written algorithm starts to work properly, the ADC is configured by loading the SPI internal registers with appropriate values.

The clock handling the loading of the SPI has a frequency of 12 MHz, supplied by FPGA by means of the IP core Clocking Wizard 3.6 [18], able to generate a number of clocks shifted in a phase by predetermined values and with frequencies selected from a specific range, starting from the same master clock signal frequency.

In the case of the FPGA used, this range is from a minimum value of 10 MHz to a maximum of 700 MHz (the maximum frequency value sustainable by the FPGA). The master clock used has a frequency of 66 MHz and is generated by a MBH2100H—66 MHz oscillator, which is mounted on the demo board.

After properly configuring the ADC, it begins the first phase of communication between the JESD204B transmitter implemented on the ADC and the receiver implemented on, called synchronization group code (CGS).

In this phase, the receiver finds the boundaries between the 10-bit characters in the data stream. During the CGS phase, the JESD204B transmit block transmits a known sequence of characters *K*. The receiver must locate this *K* characters in its input data stream using clock and data recovery (CDR) techniques. The receiver issues a synchronization request by activating the SYNCINB± pins of the ADC. The JESD204B Tx begins sending *K* characters until the next clock boundary. When the receiver has synchronized, it waits for the correct reception of at least four consecutive *K* characters. It then deactivates SYNCINB±. The ADC then transmits an initial lane alignment sequence (ILAS) on the following clock boundary.

The ILAS phase follows the CGS phase and starts on the next clock boundary. The ILAS consists of four multiframes, with *R* known characters marking the beginning and *A* known characters marking the end. The ILAS begins by sending *R* characters followed by 0–255 ramp data for one multiframe. On the second multiframe, the link configuration data are sent starting with the third character. The second character is *Q* known characters to confirm that the link configuration data follows. All undefined data slots are filled with ramp data. The 3 and 4 multiframe are the same as multiframe 1.

After the initial lane alignment sequence is completed, the user data is sent. In a usual frame, all characters are user data.

The synchronization clock signal comes from the ADC with a frequency 1\4 of the sampling frequency, 500 MHz. This external clock is used as a reference clock of the RX PLL implemented in each transceiver. The transceiver gives in output a clock at a half frequency of 250 MHz.

The schematic representation of the connection is shown in **Figure 5**.

From such a signal, using a block called advanced mixed-mode clock manager (MMCM_ ADV) [19] provided by ISE, all the clock signals necessary to the management of the transceiver modules and of the module JESD204B provided by XILINX are generated.

The ADC communicates with the FPGA through the transceivers. Each transceiver has a high-speed (up to 6.5 Gbps) serial data line as input and a 32-bit word with a frequency of 125 MHz as output, all 32-bit words (which may be from 1 to 8 according to the number of lines used) are passed simultaneously to the JESD204B block. Here, they are recombined to provide at its output the correct information consisting of 16 words of 12 bits (the ADC resolution) in output at a frequency of 125 MHz (sampling frequency = 16 * 125 MHz = 2 GHz).

The algorithm, if it were to carry out its function to each data serially, must have an execution frequency of 2 GHz in order to be able to process all the information before it is overwritten

http://dx.doi.org/10.5772/66853

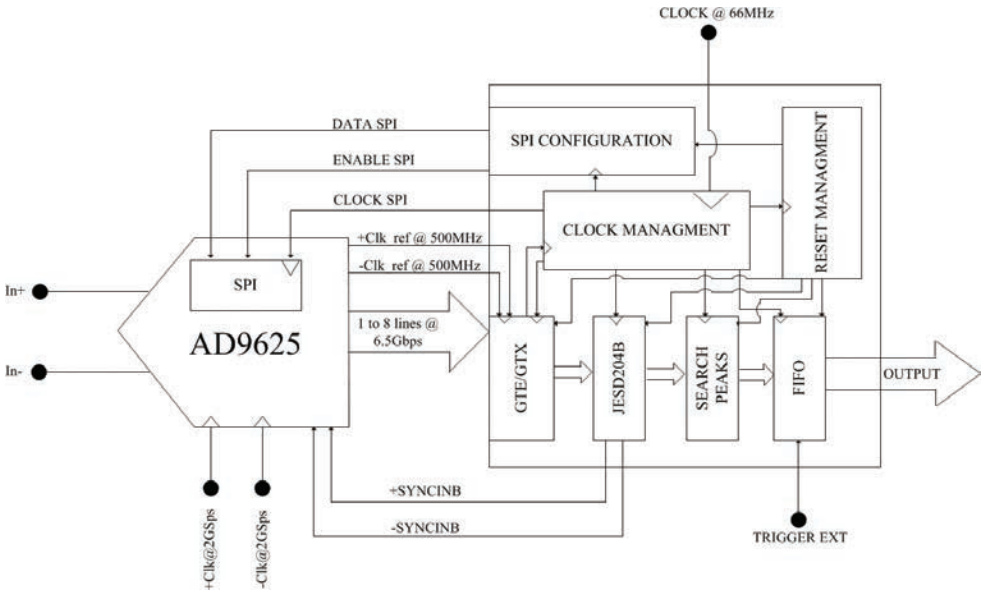

Figure 5. Schematic representation.

by new data. This is physically impossible because the maximum operating frequency of the used FPGA is 700 MHz. To overcome this problem, one has to exploit the ability to process more data in parallel, thus reducing the operating frequency of the FPGA with the advantage of relaxing the time constraints by avoiding the introduction of time delays inside the device.

At the beginning of the signal processing procedure, a counter starts to count providing the timing information related to the event under scrutiny. The determination of a peak is done by relating the *i*th sampled bin to a number *n* of preceding bins, where *n* is directly proportional to the rise times of the signal peak.

The value of *n* has been chosen to be 2. Supposing a 1 ns rise time for the signal, which is sampled at a rate of 2 GS/s, two maxima must be separated by at least three samples to be associated with two distinct peaks.

The implemented algorithm is shown schematically in **Figure 6**.

Among the 16 samples $S_{K,X}$ where *K* is the sample number among those available and *X* is the time at which they are present, the functions $D1_{K,X}$ and $D2_{K,X}$ are calculated according to the relations of Eqs. (1) and (2), respectively (step 1).

$$D1_{K,X} = \left(\frac{2 * S_{K,X} - S_{K-1,X} - S_{K-2,X}}{16} * 3 \right) \tag{1}$$

$$D2_{K,X} = \left(\frac{2 * S_{K,X} - S_{K-2,X} - S_{K-3,X}}{16} * 5 \right) \tag{2}$$

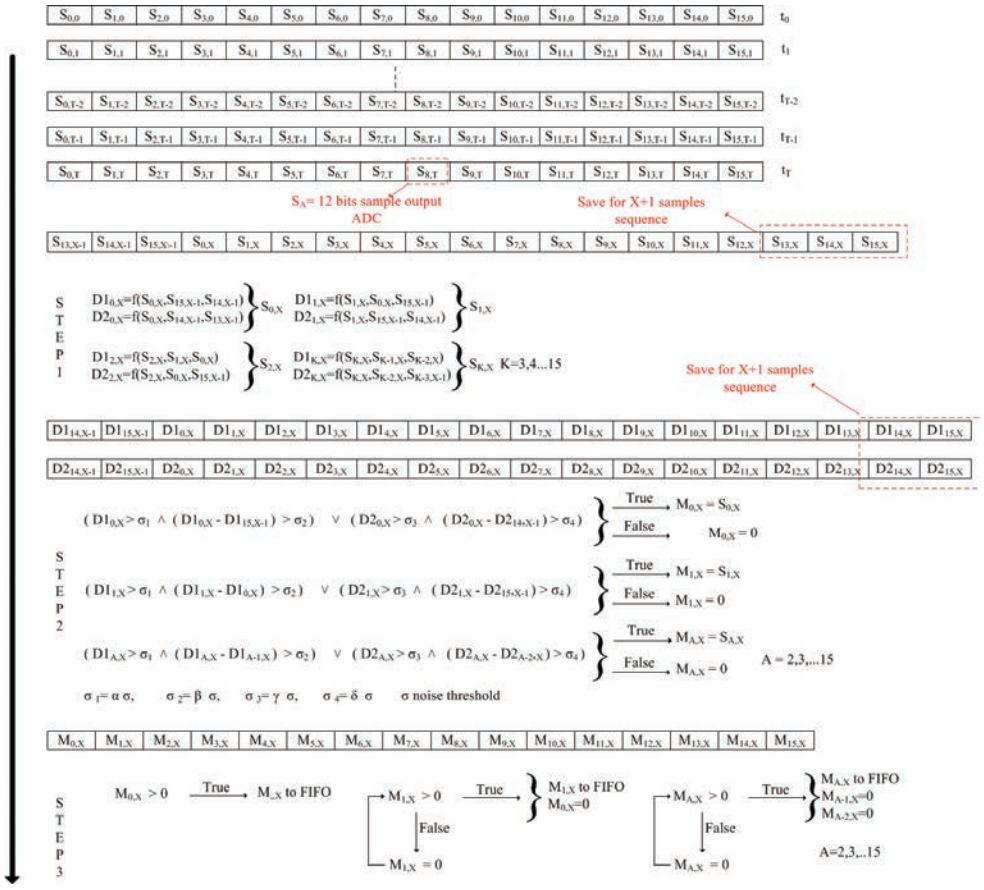

Figure 6. Algorithm implemented.

The value of $D1_{K,X}$ function provides an estimate of the variation of the amplitude of the *i*th sample compared to the (*i*-1)-th and (*i*-2)-th samples. Likewise, the $D2_{K,X}$ function as far as the (*i*-2)-th and (*i*-3)-th samples are concerned.

Figure 7 shows the input signal to the ADC, the peaks found, and the values of the functions and of their differences. As can be noticed, the functions assume their maximum values in correspondence with the signal peaks.

The values of the $D1_{K,X}$ and $D2_{K,X}$ functions are stored in a 16-element vector. For the first three samples in the series of 16 input words, they make use of the last three corresponding samples of the previous 16 input words. A temporary storage is, therefore, used to this purpose. Likewise, in order to be able to calculate the differences $D1_{K,X}$ and $D2_{K,X}$ in the samples head, the values of the functions $D1_{K,X-1}$ and $D2_{K,X-1}$, calculated at the previous time and relating to the samples tail must be temporarily stored.

http://dx.doi.org/10.5772/66853

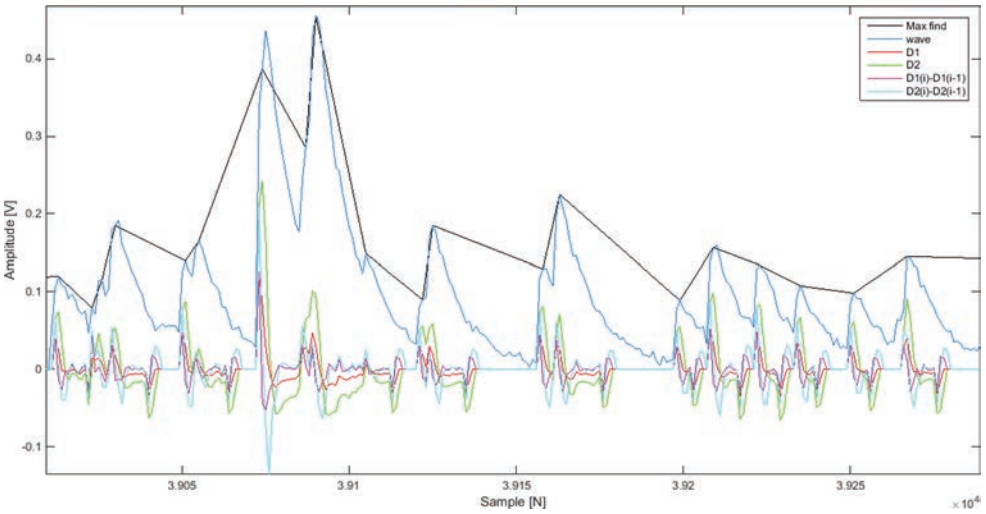

Figure 7. Input signal, found peaks, and discrimination functions.

The values of $D1_{K,X}$ and $D2_{K,X}$ and the differences between $D1_{K,X}$ and $D1_{K\text{-}1,X}$ and between $D2_{K,X}$ and $D2_{K\text{-}1,X}$ are compared with respect to thresholds proportional to the level of noise present in the input signal (step 2).

If the imposed conditions are met, the test sample is identified as a possible maximum and its value is temporarily stored in $M_{K,X}$.

In order to transfer the data to the memory, the last step consists in checking that, according to the conditions imposed on the signal rise time, there are no adjacent peaks (step 3).

To this purpose, a check is performed on the last maximum. If its corresponding location contains a nonzero value, this is transferred into the memory and the two preceding locations are assigned a zero value. The procedure continues by scrolling all locations and sending to memory all effective maxima. The time information is provided with a counter, the value of which is stored in an FIFO every time a peak is found. The counter is clocked at 125 MHz clock frequency and, therefore, when a peak is found, the time to be stored is multiplied by 16 to take into account the correct ADC sampling rate and added to the sample number corresponding to the maximum found, which is illustrated in **Figure 8**.

The memories are continuously filled as new peaks are found. When a trigger signal occurs at time $t0$, indicating with T the observed time window, which coincides with the maximum drift time, the reading procedure is enabled and only the data related to the found peaks in the $[t0, t0 + T]$ time interval are transferred to an external device. This results in data reduction factors of more than one order of magnitude.

For storing data, ISE provides several types of IP cores. The one used is the FIFO generator 9.3 [8]. Within this IP, one can choose between various options, each one supporting different

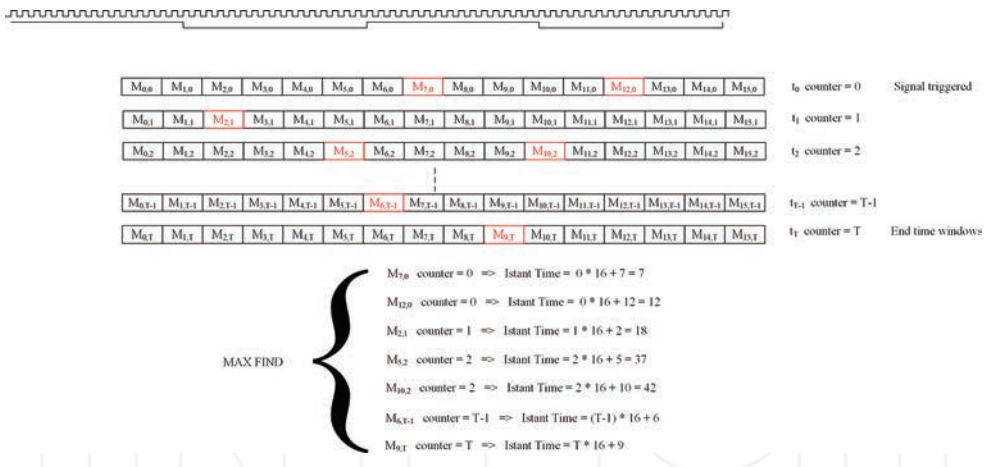

Figure 8. Time information algorithm.

features. For these purposes, an FIFO common clock has been chosen, the depth of which depends on the observed time window T of the event, while the width has been chosen of 8 bits. This is linked to the way in which the data is sent to the external device.

ISE provides several methods of data exchange, such as communication through UART and Ethernet.

A UART-type interface has been used. It requires 1 start bit, 8 data bits, and 1 stop bit to communicate.

The dates are sent with a baud rate of 115,200 bps. The clock UART (16 times baud rate) is obtained from a clock of 50 MHz obtained from the Clock wizard IP core.

However, since its transmission speed is very low, the time required to transfer data stored in the FIFO to the external device is very long. To considerably reduce this time, an Ethernet module, having significantly faster transfer times, can be implemented.

The implemented algorithm has been executed both in MATLAB and VHDL, using a test signal with a defined noise of 0.5 mV rms, and a number of peaks. A comparison of the obtained results can be used to derive an index of the algorithm performance. By comparing the results obtained from the number of real and fake peaks found, one can assess if any error is due to the algorithm itself or due to approximations in the VHDL implementation, when the function $D1_{K,X}$ and $D2_{K,X}$ are rounded to integer.

Figures 9 (MATLAB) and **10 (VHDL)** show the algorithm efficiency, calculated using Eq. (3), and the percentage of fake peaks, calculated using Eq. (4)

$$\text{Eff}[\%] = 100^* \frac{Pf - \text{Pfake}}{Pr} \tag{3}$$

$$P\text{fake}[\%] = 100^* \frac{\text{Pfake}}{Pf} \tag{4}$$

http://dx.doi.org/10.5772/66853

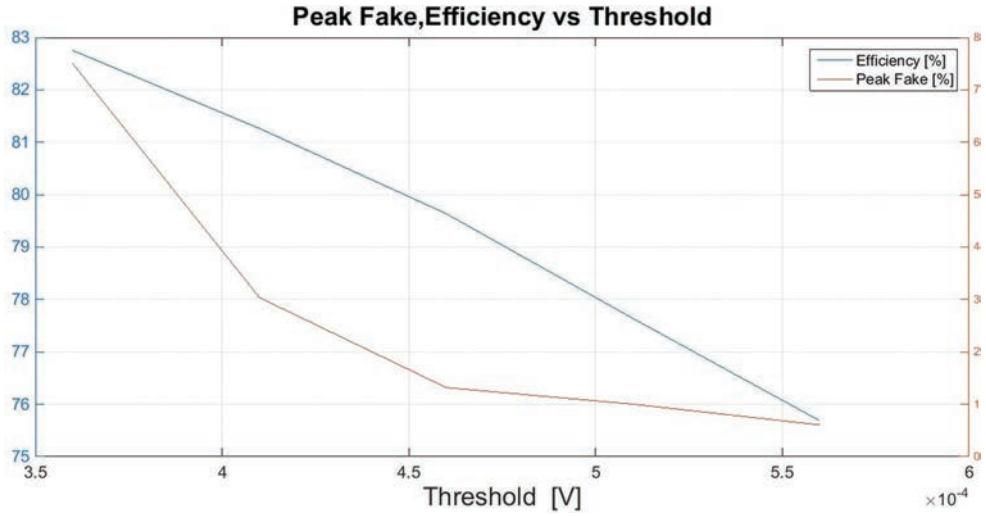

Figure 9. Algorithm results with MATLAB.

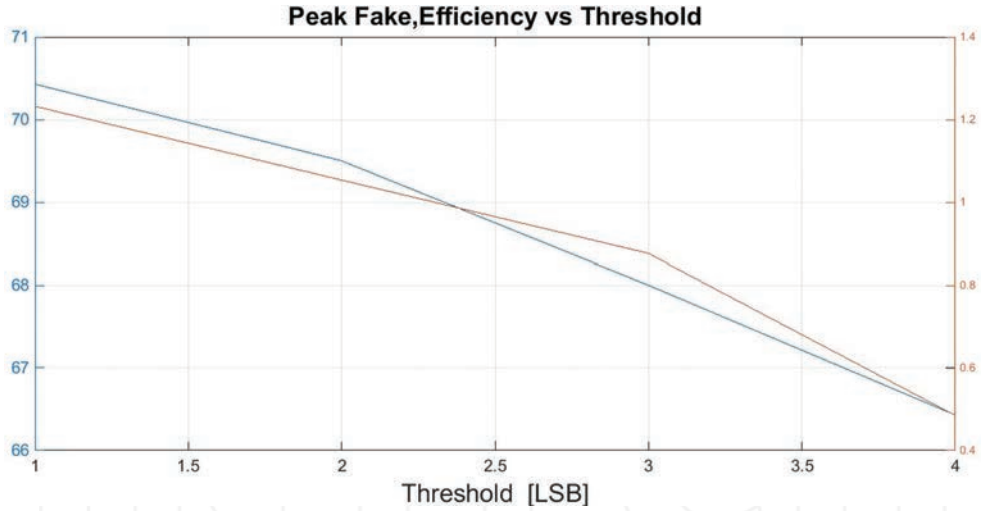

Figure 10. Algorithm results with VHDL.

where *Pf* is the number of peaks found in the signal, *Pr* is the number of real peaks, and *P*fake is the number of fake (equal to *Pf* – *Pr*).

The results were obtained by varying the proportionality factors used to calculate the thresholds to which the $D1_{K,X}$ and $D2_{K,X}$ functions and their differences are compared.

As expected, in both cases, increasing the thresholds not only decreases the efficiency but also reduces the number of false peaks.

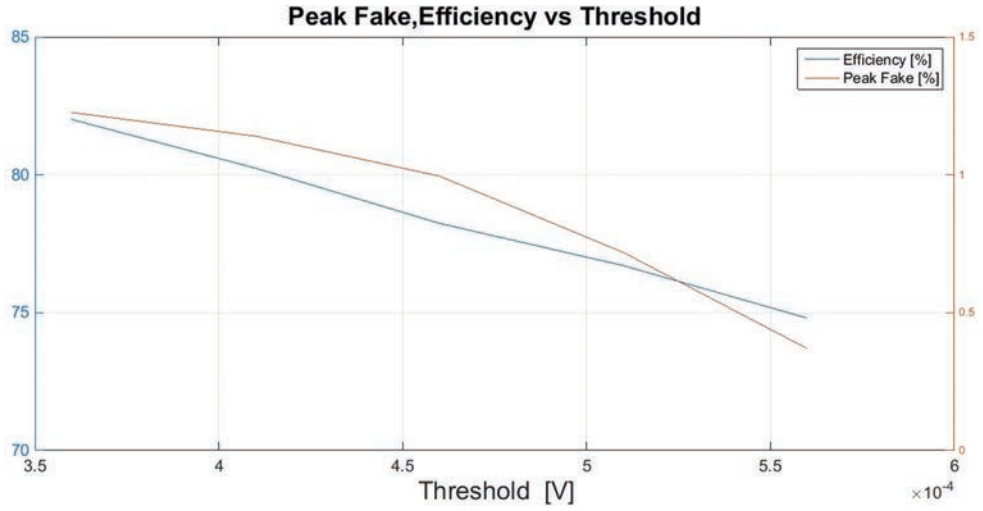

Figure 11. Algorithm results with MATLAB without noise.

The algorithm executed with MATLAB performs slightly better than VHDL, for example, for a rate of fake peaks of 1%, one has an efficiency of 76% in the case of MATLAB and an efficiency of 69% for VHDL.

A further test was performed by implementing the algorithm in MATLAB on the signals previously analyzed without the noise contribution. The results obtained are shown in **Figure 11**.

By comparing **Figures 9** and **11**, it can be seen how the presence of noise induces a reduction in efficiency and an increase of false peaks, indicating that a part of the errors of the algorithm may actually be due to the characteristics of the signal at input.

This problem can be mitigated somehow by trying to increase the signal-to noise-ratio by filtering the input signal to the ADC.

The use of the algorithm described results in data reduction factors of more than one order of magnitude. It is executed for each readout channel and, considering the high number of I/O FPGA pins, it is possible to process multiple channels corresponding to different drift chamber signals with a single device.

3. Conclusion

A key role in the algorithm is represented by the parallel data processing combined with the use of a pipeline structure, which allows managing more input data at the same time, thus relaxing the speed specifications.

http://dx.doi.org/10.5772/66853

The choice of an FPGA device turns out to be the best among those available, since it allows many advantages as follows:

- Adaptability: The possibility of being programmed to perform "*ad hoc*" functions, optimizing its performance in relation to the task to be performed.
- Parallelization: The possibility of executing the same instructions on multiple data simultaneously. In the written HDL code, one can declare multiple processes, which are executed in parallel sheets, while the internal instructions are executed serially.
- Diffusion: The FPGA devices are widely used in various application areas, this has led several manufacturers of electronics to develop scripts in HDL that can be implemented in FPGA to provide the user devices a starting code to be modified according to the needs to create a device-FPGA communication link.
- Management of the links: The user during the programming phase can create appropriate constraints to manage internal links to the FPGA, thus being able to optimize the time delays of the lines, the most critical connections, with a rooting favoring closer.
- IP core: The FPGA programming tool (in this case ISE) makes available the intellectual property, a block of logic data having multiple features, such as the clock management, the management of interfacing with various external devices in the various possible standards, the management of the storage of the information with various types of implemented memories, and many others.

Examining how the performance of FPGA devices has evolved over the years, by taking into consideration factors such as the data storage capacity and maximum operating frequency, one can note that these are strongly linked to the progressive miniaturization of the technology with which they are made.

They follow the technological scaling, which led to the advent of FinFET technology to reduce the size of the channel length to less than 16 nm. The effect of technological scaling leads to different types of advantages as follows:

- A reduction of the length of the connections, thus allowing for a lower propagation delay between the various cells and for a higher operating frequency.
- Being able to integrate, for equal occupied area, more logic cells, thus increasing the computational capacity and the ability to store data.
- A reduction in size, and hence a greater integration of multiple devices in a smaller area.
- A reduction in the power consumption.

The adoption of an FPGA, as the main block of data management and communication between various devices, appears to be winning, not only relying on the existing technology and on the various application tools created by the different manufacturers, but also thinking about the future, that is, how much they can be improved in terms of performance with improved technologies and how fast they can be improved.

Author details

Gianluigi Chiarello[1, 2], Claudio Chiri[2], Giuseppe Cocciolo[1, 2], Alessandro Corvaglia[2], Francesco Grancagnolo[2]*, Marco Panareo[1, 2], Aurora Pepino[1, 2] and Giovanni Francesco Tassielli[1, 2]

*Address all correspondence to: franco.grancagnolo@le.infn.it

1Department of Mathematics and Physics "Ennio De Giorgi" – Salento University, Lecce, Italy

2 INFN (Istituto Nazionale Fisica Nucleare), Lecce, Italy

References

[1] Blum W., Riegler W., Rolandi Li. Particle Detection with Drift Chambers. 2nd ed. Springer-Verlag, Berlin, Heidelberg; 2008. pp. 448. DOI: 10.1007/978-3-540-76684-1

[2] Sauli F., Principles of operation of multiwire proportional and drift chambers. In: Experimental Techniques in High Energy Physics. 2nd ed. Addison-Wesley; 1987. pp. 79–188. DOI: 10.5170/CERN-1977-009

[3] Cataldi G., Grancagnolo F., Spagnolo S. Cluster counting in helium based gas mixtures. Nuclear Instruments and Methods in Physics Research Section A: Accelerators Spectrometers Detectors and Associated Equipment. 1997;**386**:485–469. DOI: 10.1016/S0168-9002(96)01164-3

[4] Tassielli G.F., Grancagnolo F., Spagnolo S. Improving spatial resolution and particle identification. Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. 2007;**572**(1 SPEC. ISS.):198–200. DOI: 10.1016/j.nima.2006.10.300

[5] Cascella M., Grancagnolo F., Tassielli G. Cluster Counting/Timing Techniques for Drift Chambers. In: Francesco Grancagnolo and Marco Panareo, editors. 1st Conference on Charged Lepton Flavor Violation; May 2013; Lecce (Italy). Elsevier; 2014. pp. 127–130. DOI: 10.1016/j.nuclphysbps.2014.02.025

[6] Signorelli G., Donofrio A., Venturini M. A novel method to estimate the impact parameter on a drift cell by using the information of single ionization clusters. Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. 2016;**824**:581–583. DOI: 10.1016/j.nima.2015.11.028

[7] Grancagnolo F. The ultimate resolution drift chamber. Nuclear Physics B—Proceedings Supplements. 2007;**172**:25–27. DOI: 10.1016/j.nuclphysbps.2007.07.033

[8] Baldini A.M., Baracchini E., Cavoto G., Cascella M., Cei F., Chiappini M., Chiarello G., Chiri C., Dussoni S., Galli L., Grancagnolo F., Grassi M., Martinelli V., Nicol D., Panareo M., Pepino A., Piredda G., Renga F., Ripiccini E., Signorelli G., Tassielli G.F., Tenchini F., Venturinia M., C. Voena. Single-hit resolution measurement with MEG II drift chamber prototypes. Journal of Instrumentation. 2016;**11**:18. DOI: 10.1088/1748-0221/11/07/P07011

[9] Chiarello G., Corvaglia A., Grancagnolo F., Panareo M., Pepino A., Primiceri P., Tassielli G. A Full Front End Chain for Drift Chambers. In: 1st Conference on Charged Lepton Flavor Violation; May 2013; Lecce. Elsevier; 2014. pp. 140–142. DOI: 10.1016/j.nuclphysbps.2014.02.029

[10] Chiarello G., Corvaglia A., Grancagnolo F., Panareo M., Pepino A., Pinto C., Tassielli G. A high performance front end for MEG II tracker. In: 6th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI), 18–19 June 2015; Gallipoli. IEEE; 2015. DOI: 10.1109/IWASI.2015.7184937

[11] Krzysztof T. P. FPGA-based, specialized trigger and data acquisition systems for high-energy physics experiments. Measurement Science and Technology. 2010;**21**(6):17. DOI: 10.1088/0957-0233/21/6/062002

[12] Cappelli L., Creti P., Grancagnolo F., Pepino A., Tassielli G. A fast readout algorithm for cluster counting/timing drift chambers on a FPGA board. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment. 2013;**718**:226–228. DOI: http://dx.doi.org/10.1016/j.nima.2012.10.087

[13] Xilinx. ML605 Hardware User Guide [Internet]. 2012 . Available from: http://www.xilinx.com/support/documentation/boards_and_kits/ug534.pdf [Accessed: 2016-08-10]

[14] Xilinx. Virtex-6 Family Overview [Internet]. 2015. Available from: http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf [Accessed: 2016-08-10]

[15] Analog-Devices. Evaluating the AD9625 Analog-to-Digital Converter [Internet]. [Updated: 2015]. Available from: https://wiki.analog.com/eval/ad9625 [Accessed: 2016-08-11]

[16] Analog-Devices. 12-Bit, 2.6 GSPS/2.5 GSPS/2.0 GSPS, 1.3 V/2.5 V Analog-to-Digital Converter [Internet]. 2014. Available from: http://www.analog.com/media/en/technical-documentation/data-sheets/AD9625.pdf [Accessed: 2016-08-14]

[17] Anolog-Device. JESD204B Survival Guide [Internet]. 2014. Available from: http://www.analog.com/media/en/technical-documentation/technical-articles/JESD204B-Survival-Guide.pdf

[18] Xilinx. LogiCORE IP Clocking Wizard 3.6 (ISE) / 4.2 (Vivado) [Internet]. 2012. Available from: http://www.xilinx.com/support/documentation/ip_documentation/clk_wiz/v4_2/pg065-clk-wiz.pdf [Accessed: 2016-08-11]

[19] Xilinx. Mixed-Mode Clock Manager (MMCM) Module (v1.00a) [Internet]. 2009. Available from: https://www.xilinx.com/support/documentation/ip_documentation/mmcm_module.pdf [Accessed: 2016-08-11]

INTECH