High speed reconfigurable computation for electronic instrumentation in space applications

Dimitrios Lampridis, Sorin Dan Cotofana, and Stefan Kraft

***Abstract*—This paper presents a new reconfigurable high-performance approach for digital pulse detection of very fast transients. The design is oriented towards space applications, but is of interest to all fields where small, fast pulse detectors with low power consumption are needed. We perform on-line filtering and pulse detection with pile-up rejection, using a custom serialized high-speed digital design with zero dead-time. The proposed module is an IP core with an AMBA interface, which we can easily connect to many existing softcore processors to form a complete system-on-chip solution. Our approach combines the performance of high-speed dedicated hardware calculations with the flexibility of a general purpose processor in a single chip. We present our design, implementation and experimental results with a Xilinx XC3S1500 FPGA, the LEON3 synthesisable processor and a minimal selection of IP cores from the GRLIB library.**

***Index Terms*—pulse detector, reconfigurable hardware, system-on-chip, space applications.**

I. Introduction

Small, light structures with low power consumption are the key to success for electronic instrumentation in space applications [1]. Lighter payloads reduce the mission costs and allow us to put more instruments in the confined space of a small satellite.

However, as the number of instruments on-board the spacecraft increases, so does the amount of data acquired during the mission. It is unlikely that we will be able to transfer all this information using the unavoidably slow down-link to earth. Furthermore, apart from their scientific purpose, many of the on-board instruments have a vital role in the spacecraft's reaction to the environment and need real-time processing. The solution is to do the required calculation on-board, and System-on-Chip (SoC) approaches offer a small, light single-chip solution, fitting the requirements.

Recent developments in the defense and space industry have introduced radiation-hardened FPGAs, making one step forward towards the use of reconfigurable hardware in future space missions. We set off to investigate the benefits from such a development in future space electronic instrumentation. To this end, we design and study a digital pulse detector for fast transients. We follow a reconfigurable approach to process the digitized signals and calculate the height of the detected pulses. The module is designed as an AMBA [2] IP core that can be combined with other cores into a single-chip SoC solution.

Linear trapezoidal filtering is applied to the digitized samples. The filters can keep a high and constant throughput, independent of the number of past samples under consideration. The filtered output is further processed by a smart on-line peak detection algorithm that discards false events and pile-ups. Both the filter and peak detector parameters are fully configurable via the AMBA APB interface.

The design was implemented on a Xilinx Spartan3 XC3S1500 FPGA, using the 32-bit LEON3 Sparc V8 compliant synthesisable processor, together with a minimal set of AMBA IP cores from Gaisler Research [3]. The ADC and filter/peak detector were clocked at 100MHz, while the rest of the system was running at 40MHz. The system was configured and controlled using software written in C and cross-compiled for the Sparc architecture.

This paper starts by presenting similar work in the field (Section II). Methods like trapezoidal filtering and dual fast/slow filtering are very common in the design of digital pulse detectors. We also make use of these methods, and we briefly explain them in Section III. The system architecture is presented in Section IV, followed by a description of the experimental setup and presentation of the obtained results (Section V). Our work concludes in Section VI with a recaputilation of

This work, which was carried out in support to a project for the European Space Agency (Advanced Studies and Technology Preparation Division / SCI–PA), was sponsored by cosine Research BV (http://www.cosine.nl). The company offered scientific support and provided all the necessary equipment and facilities for our research.

D. Lampridis is a M.S. student with the Computer Engineering department, EEMCS faculty, Delft University of Technology, 2628CD, NL (Email: d.lampridis@student.tudelft.nl).

S. D. Cotofana is an associate professor with the Computer Engineering department, EEMCS faculty, Delft University of Technology, 2628CD, NL (Email: s.d.cotofana@ewi.tudelft.nl).

S. Kraft is general manager of cosine Research BV, 2333CA, NL (Email: s.kraft@cosine.nl).

our work and final remarks.

II. Similar work

Over the past 15 years, the introduction of fast ADCs in ever-increasing speeds has brought digital processing into fields that used to be dominated by analog solutions. It was not long before the research community and the industry came up with a variety of digital solutions for pulse detection and spectroscopy.

Today, with low-power ADCs operating well in the order of several million samples per second, the range of solutions spans from purely analog solutions to almost completely digital (apart from source conditioning circuits and the ADC itself, although recent FPGAs are even equipped with embedded ADCs). Interested readers should further consult [4], [5] and [6] for a more detailed explanation of the benefits of digital processing in high-count pulse detection and spectroscopy.

On one end of existing proposals lies that of [7]. Its authors propose an analog processing circuitry that only digitizes the detected pulse peaks for storage. They use a 10MHz, 12-bit ADC and an FPGA for control and on-chip storage.

Other researchers propose fast data acquisition systems, coupled with off-line processing blocks [8]. In [9], the off-line processing is actually performed in software, to increase the flexibility of the device. Moreover, in [10], a very fast 200MHz, 14-bit ADC is used for data acquisition, and an FPGA is used to compress and store the results for later software processing.

Yet other groups of researchers have been investigating purely hardware solutions using FPGAs ([11], [12], [13], [14]). One thing all these proposals have in common is that they rely on external chips (DSPs and/or microcontrollers) to assist the FPGA in the calculations and system control.

We feel that our work shares the most with that of [15]. Its authors propose another hardware solution with an FPGA and a external DSP. However, all calculations are performed on-line within the FPGA, and the DSP is only used for spectra generation and transmission of results via a serial port. The source signal is sampled at 60MHz with an 8-bit ADC.

To the best of our knowledge, our idea is unique in that it proposes a single-chip solution, using an FPGA and no other supporting chips. We take advantage of our managed SoC approach to embed the complementary processor and interconnection bus within the FPGA, resulting in a small and lightweight implementation. We suggest a modular IP core with an AMBA bus interface that can be easily connected to many of the available synthesisable processors.

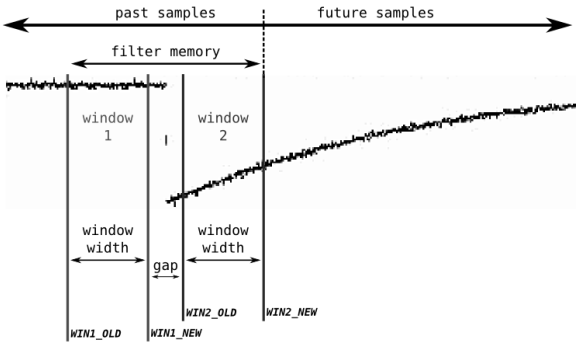


Fig. 1: Segmentation of input samples for trapezoidal filtering

III. Theory of operation

Although pulse detectors may exist in many different flavours, they often operate under a common idea: a triggering system detects the pulse, signaling a second stage that calculates the height of the pulse, or some linear function of it.

In this section we provide details on how we have chosen to implement these concepts in our design. We use well-tested and compact methods, based on their suitability for fast on-line filtering and peak detection.

A. Trapezoidal filtering

Many digital pulse detectors use triangular and/or trapezoidal functions to filter their input. The reason is that those functions are easy to understand and implement in digital logic. Triangular weighting is useful for very fast detector channels, while trapezoidal weighting is preferred for slow, good resolution channels. If the flat-top of the trapezoid shape is set to zero, the function is identical to triangular weighting. Therefore, the same trapezoidal function with different parameters can be used for both channels. When compared to Gaussian shaping, a trapezoidal function has comparable resolution but needs less processing time [16]. These facts make trapezoidal functions a good all-around choice for high-speed digital pulse detectors.

A trapezoidal function considers two data-sets (windows) of input samples at a time. Between the two data-sets exists an optional “gap”, represented by the flat-top of the trapezoid. Fig. 1 illustrates a radiation-induced pulse event and the segmentation of its input samples in data-sets. Both windows must have the same width, and the sum of window widths and possible gap must not exceed the filter’s memory.

Every clock period, the filtering function averages the samples inside each window and subtracts the two

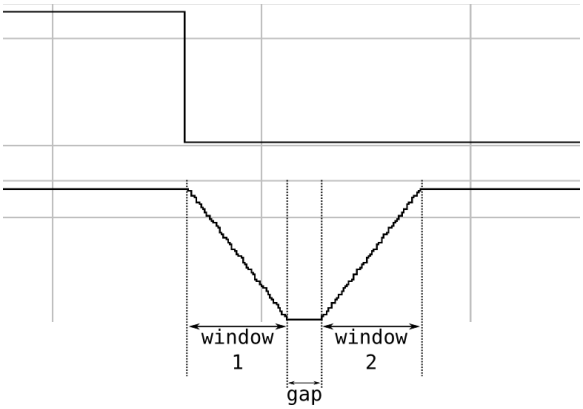


Fig. 2: Step function input (top) and example filter response (bottom)

resulting sums. That is, if $O[n]$ is the output of the filter at time unit n, $I[n]$ is the input of the filter at the same moment, w is the window width and g is the gap, then:

$$O[n] = \left(\sum_{k=n-w}^{n} \frac{I[k]}{w} \right) - \left(\sum_{k=n-2w-g}^{n-w-g} \frac{I[k]}{w} \right) \quad (1)$$

Fig 2 shows a simulated example output (bottom curve) of the trapezoidal filter, given a step function as input. The figure shows why this type of filter is called "trapezoidal", but also how the various parameters of the filter can affect the shape of the output. For optimal performance, the gap should always be greater than the event rise time of the input.

B. Dual-channel filters

Modern pulse detectors, both digital and analog, often process their input using two channels simultaneously. A "fast" channel is used to detect the event, triggering a "slow" channel that extracts information about the pulse height and provides good resolution. The fast channel is also used to detect multiple input events while the slow channel is still evaluating (pile-up inspection) [6].

Fig. 3 shows an output of such a dual-channel configuration: From top to bottom, we have a radiation-induced signal followed by the slow and fast channel responses. We can see how the input events A and B are captured by both channels, but the arrival times of events C and D are close enough to cause a pile-up on the slow channel. The fast channel on the other hand has no problem to detect all four distinct events. We can use this combined information to register the first two clean events and discard the third one as pile-up, using the enhanced resolution values of the slow filter.

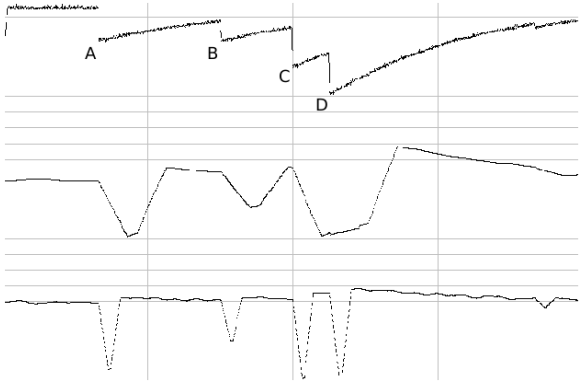


Fig. 3: Pile-up inspection using dual-channel filtering

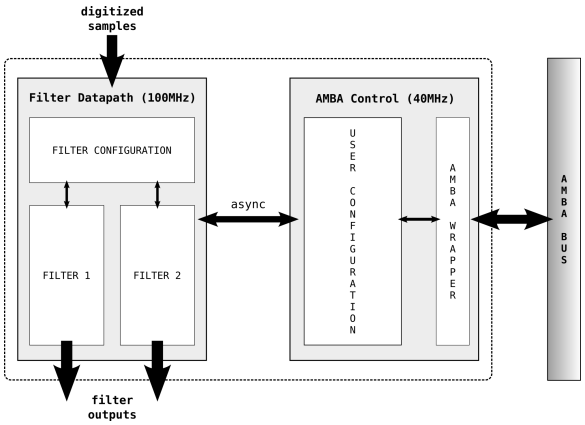


Fig. 4: IP core functional diagram

IV. System architecture

The IP core consists of two main blocks: the AMBA interface and the filter datapath.

Fig. 4 shows the functional diagram of the IP core. The AMBA block controls the communication between the filters and the AMBA bus. To the user, the device appears as a memory-mapped set of registers, accessible over the system bus. The user can alter the filter parameters and query the status of the device by accessing those registers.

Once the desired configuration is in place, the control block transfers the new parameters for both filters to the next block over an asynchronous link, and processing begins. The outputs of both filters are connected to pins of the FPGA and can be inspected for verification of the filter functions and debugging purposes.

A. Filter datapath

Processing of input samples takes place in a serial way, using shallow-pipelined logic (three stages). The filters produce one output each clock cycle, and the peak detection algorithm performs on-line inspection.

It is easy to spot that equation (1) has a clear relation between two consecutive outputs: each output is equal to the previous one, with the addition of one new sample and the removal of the oldest one inside each window. It is also safe to ignore for implementation purposes the division by w. All filter parameters are saved in the configuration registers and can be retrieved later by the application. With these observations in mind, and following the notation of figure 1, we can rewrite (1) in a serialized form:

$$O[n] = O[n-1] + WIN2_{NEW} - WIN2_{OLD} - WIN1_{NEW} + WIN1_{OLD} \quad (2)$$

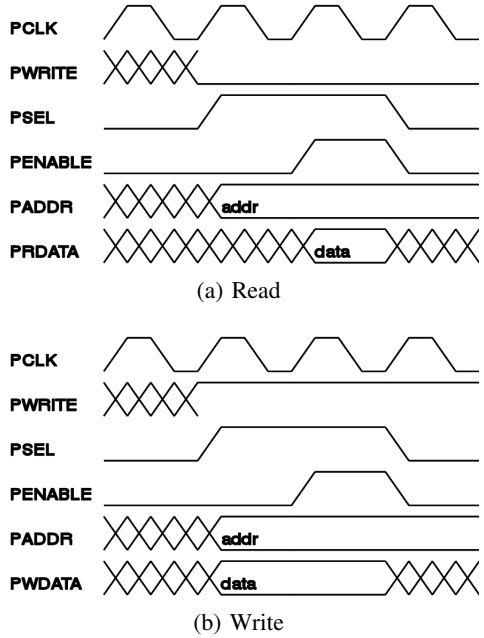
The main advantage of this approach is that we keep the hardware complexity of the calculation fixed, since (2) will always include five operands, independent of window and gap widths.

In our design, past samples are stored in a FIFO memory structure. Every clock cycle, a new sample is inserted and the oldest one is dropped. During an initial configuration step, the application sets the desired values of w and g. Depending on these values we choose four cells of the FIFO, representing the four samples at the windows boundaries. During runtime, we keep track of the previous output and add/subtract to it the sampled values as they move through these four cells.

Our idea is further illustrated in figure 5. The first stage consists of the FIFO memory and a set of multiplexers to select the operands of the fast (top) and slow filter (bottom), according to the configured values. Given the nature of the instrument, it is expected that the user will update the configuration values once every time before starting his measurement. Therefore the slow path from the selection input of these wide multiplexers to their output is actually static by the time the device starts processing and does not contribute to the delay of the stage.

The second stage performs the trapezoidal filtering. Based on (2), we design a reduction tree of Carry-Save-Adders (CSA) [17]. Every cycle, we get the previous output in sum and carry redundant form and add to it the four new operands, producing a new pair of sum and carry. This leads to a 6-to-2 reduction tree. The final result is obtained by adding every cycle the two outputs of the tree. The final addition is described in behavioral VHDL, in order to let the synthesis tool optimize it in any way possible.

The third stage constitutes the peak detection algorithm. While configuring the detector, the user provides upper and lower thresholds for both filter outputs. During runtime, each filtered output is connected to three comparators: two comparators are used to check the output against the upper and lower threshold, while a third one is used to trace the maximum value reached. The four threshold comparator outputs (two per filter) are driven into a finite state machine (FSM). Using these four signals the FSM is able to reject pile-ups and only record complete events. When a new event is recorded and its maximum value is determined, the FSM signals the AMBA interface to update the status registers with the new value.



(a) Read

(b) Write

Fig. 7: AMBA APB read (a) and write (b) cycles

B. AMBA interface

The second half of the device is the AMBA interface. Our pulse detector is designed to communicate over the AMBA APB bus. We can afford a slow bus since we perform real-time calculations to reduce the amount of output data. The advantages over faster interconnections like AHB are lower power consumption and reduced interface complexity.

The block diagram of the AMBA interface can be seen in Fig. 6. A set of 32-bit configuration/status registers (gray boxes) appear as a 32-bit aligned memory-mapped region, accessible over the APB bus. Communication with the device consists of read and write operations inside this I/O region. Fig. 7 shows the timing diagrams of the read and write bus cycles.

The registers are used to store the configuration and to control the filter (start, stop, reset, reconfigure). Configurable parameters include the window width and gap of the trapezoidal filters, and the peak detection thresholds.

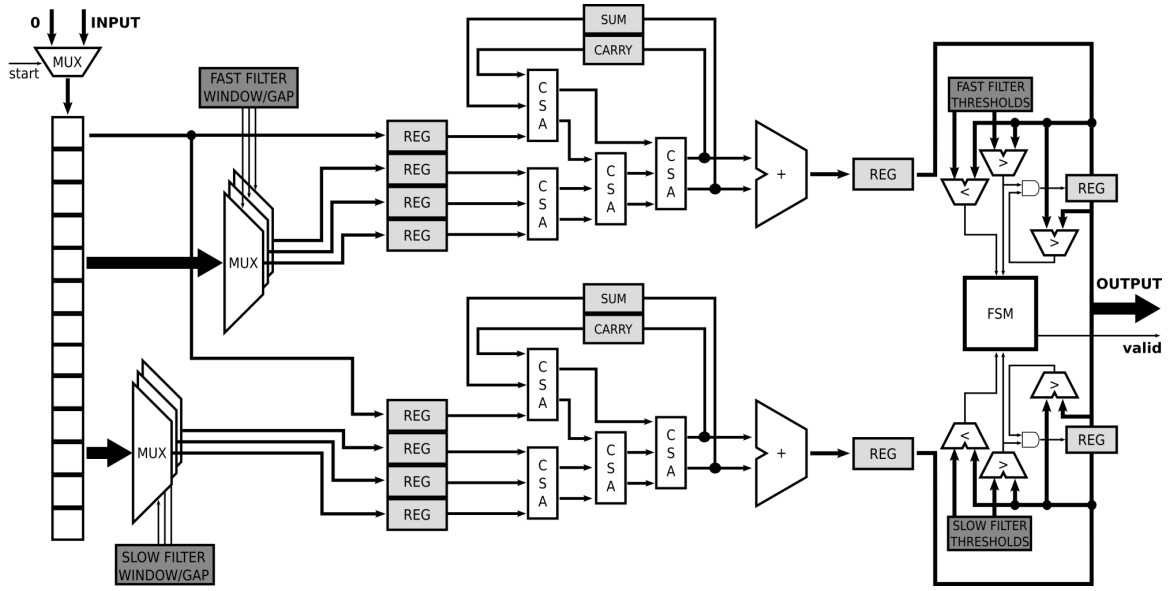


Fig. 5: Block diagram of the complete pipelined filter datapath

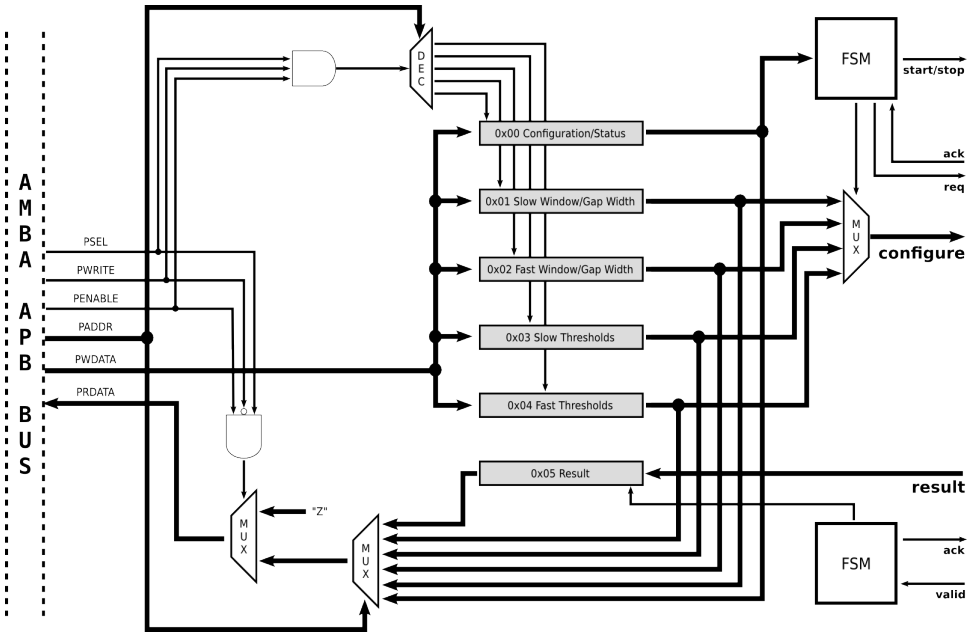


Fig. 6: Block diagram of the AMBA interface

Another register is used to hold the last detected pulse height.

Two FSMs control the asynchronous communication with the filter datapath. One FSM (top one in Fig. 6) is used to transmit the configuration values to the datapath registers (dark gray boxes in Fig. 5), while the other FSM (bottom one in Fig. 6) is in charge of storing the received results and signaling the application of the new data. Special attention was given to minimize the number of signals which cross clock domains, and to properly synchronize those that do so.

V. Experimental setup and results

We implemented the above design as a modular IP core in standard VHDL (IEEE Standard 1076–1993), without using any manufacturer-specific libraries. Combined with the LEON3 synthesisable processor, the result is a very portable design that can be programmed in the majority of existing FPGAs. The LEON3 processor was picked from the GRLIB library of IP cores, together with a debug support unit, AHB controller and APB bridge, memory controller, UART port and interrupt controller. The complete set of IP cores was programmed inside a

TABLE I: Detector setup for Cobalt-60 samples

	Fast filter	Slow filter
Window width	500 ns	1000 ns
Gap width	150 ns	150 ns
Upper threshold	512	1000
Lower threshold	256	1000

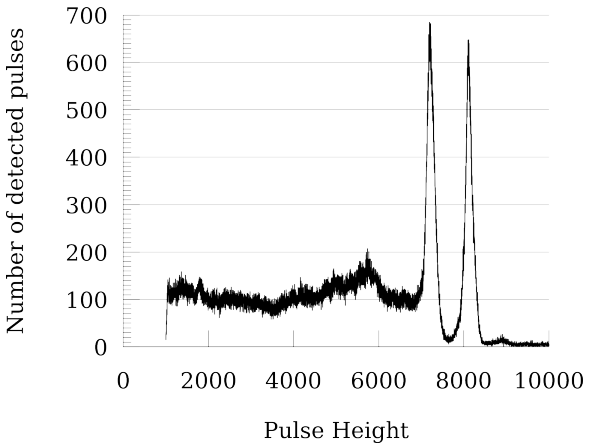


Fig. 8: Pulse height distribution of one million pulses from Cobalt-60 samples

Xilinx Spartan3 XC3S1500 FPGA, occupying 77% of the available slices.

The necessary source events were generated from the particles emitted by Cobalt-60 isotope samples, and captured by a scintillator cube coupled to a Hamamatsu S3204-08 photodiode. The pulses from the photodiode were sampled and digitized at the speed of the filter datapath, 100MHz. The LEON3 processor and AMBA bus were clocked at 40MHz. A single 50MHz clock signal was provided to the system. To create the other clock frequencies needed by the processor, the filters and the external ADC, we made use of the clock managers available on the Spartan3 FPGA.

We developed software to control the peak detector and classify the pulses according to their height. The software was developed on an x86 Linux platform in standard C and was cross-compiled for the 32-bit Sparc processor. The binary executable was uploaded to the PROM of the development board that hosted the FPGA.

The resulting histograms were transferred through the serial port to a laptop and visualized with the help of plotting software. Fig. 8 shows the pulse height distribution for one million detected pulses obtained from the Cobalt-60 samples. Table I summarizes the configuration that was used during that measurement.

VI. Conclusion

In this paper we described the principles, design and implementation of a fully-configurable digital pulse detector. We proposed a single-chip SoC solution and created a small, lightweight device, keeping in mind the needs of current and future space applications. All processing is performed in real-time with zero dead-time. The ability to run additional software on the general purpose on-chip processor adds to the flexibility of the design.

We believe our work is important because it highlights the potential benefits in the use of reconfigurable hardware in future space missions; by compacting multiple functions inside a complete SoC design, higher levels of integration can be achieved, while increasing the reusability of the design and decreasing development time, effectively reducing the overall cost of the mission.

Acknowledgment

D. Lampridis would like to thank in particular Alejandro Palacios-Robles and Erik Maddox for their unlimited support during the course of the research, as well as all the personnel of cosine Research BV for their contribution to this project.

References

[1] S. Kraft *et al.*, "On the study of highly integrated payload architectures for future planetary missions," *Proceedings of 11th SPIE International Symposium on Remote Sensing*, 2005.

[2] *AMBA specification, IHI 0011A*, 2nd ed., ARM limited, 1999.

[3] "LEON3 synthesizable processor and GRLIB portable IP library," [Online]. Available: http://www.gaisler.com.

[4] W. Warburton and P. Grudberg, "Current trends in developing digital signal processing electronics for semiconductor detectors," *Nuclear Inst. and Methods in Physics Research, A*, vol. 568, no. 1, pp. 350–358, 2006.

[5] G. Ripamonti, A. Pullia, and A. Geraci, "Digital vs. analogue spectroscopy: a comparative analysis," *Instrumentation and Measurement Technology Conference IMTC/98. Conference Proceedings. IEEE*, vol. 1, 1998.

[6] W. Warburton, M. Momayezi, B. Hubbard-Nelson, and W. Skulski, "Digital pulse processing: new possibilities in nuclear spectroscopy," *Applied Radiation and Isotopes*, vol. 53, no. 4-5, pp. 913–920, 2000.

[7] S. Buzzetti, M. Capou, C. Guazzoni, A. Longoni, R. Mariani, and S. Moser, "High-Speed FPGA-Based Pulse-Height Analyzer for High Resolution X-Ray Spectroscopy," *IEEE Transactions on Nuclear Science*, vol. 52, no. 4, pp. 854–860, 2005.

[8] C. Loureiro and C. Correia, "Innovative modular high-speed data-acquisition architecture," *IEEE Transactions on Nuclear Science*, vol. 49, no. 3 Part 2, pp. 858–860, 2002.

[9] M. Bautista-Palacios *et al.*, "Configurable hardware/software architecture for data acquisition implementation on FPGA," *International Conference on Field Programmable Logic and Applications*, pp. 241–246, 2005.

[10] B. Esposito, M. Riva, D. Marocco, and Y. Kaschuck, "A digital acquisition and elaboration system for nuclear fast pulse detection," *Nuclear Inst. and Methods in Physics Research, A*, vol. 572, no. 1, pp. 355–357, 2007.

[11] D. Odell, B. Bushart, L. Harpring, F. Moore, and T. Riley, "Zero deadtime spectroscopy without full charge collection," *9. symposium on radiation measurements and applications, Ann Arbor, MI (United States)*, 1998.

[12] V. Jordanov, D. Hall, M. Kastner, G. Woodward, and R. Zakrzewski, "Portable radiation spectrometer using low power digital pulseprocessing," *Nuclear Science Symposium, 1999. Conference Record. 1999 IEEE*, vol. 1, 1999.

[13] J. Cardoso *et al.*, "A high performance reconfigurable hardware platform for digital pulse processing," *IEEE Transactions on Nuclear Science*, vol. 51, no. 3 Part 3, pp. 921–925, 2004.

[14] R. Abbiati, A. Geraci, and G. Ripamonti, "Self-configuring digital processor for on-line pulse analysis," *IEEE Transactions on Nuclear Science*, vol. 51, no. 3 Part 3, pp. 826–830, 2004.

[15] M. Bolic and V. Drndarevic, "Digital gamma-ray spectroscopy based on FPGA technology," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 482, no. 3, pp. 761–766, 2002.

[16] CANBERRA, "Performance of digital signal processors for gamma spectrometry," [Online]. Available: http://www.canberra.com/pdf/Literature/a0338.pdf.

[17] B. Parhami, "Computer arithmetic," *Oxford University Press*, pp. 128–131, 2000.