A novel digital pulse processing architecture for nuclear instrumentation

Yoann Moline, Mathieu Thevenin, Gwenolé Corre, M. Paindavoine

A Novel Digital Pulse Processing Architecture for Nuclear Instrumentation

Y. Moline, M. Thevenin, G. Corre
CEA, LIST, Laboratoire Capteurs et Architectures Électroniques, F-91191 Gif-sur-Yvette, France

M. Paindavoine
Université de Bourgogne, Laboratoire d'Etude de l'Apprentissage et du Développement, F-21065 Dijon, France

***Abstract*—This paper introduces a specific Multiple Program Multiple Data (MPMD) architecture designed to address the issues of nuclear instrumentation: pulse processing, real-time, multi-channel flexibility, dead-time management and programmability. The proposed architecture comprises a set of independent and programmable functional units. Their execution is driven by the pulses arrival. It is able to deal with non-deterministic events and program durations. The virtual prototype of the architecture is developed in cycle-accurate SystemC and shows promising results in terms of scalability while maintaining zero dead-time. This architecture paves the way for novel programmable embedded real-time pulse processing restricted until now to offline processing.**

Keywords—Digital Pulse Processing, Digital Architecture, Digital Signal Processing, Nuclear Instrumentation

I. Introduction

The field of nuclear instrumentation covers a wide range of applications, including counting, spectroscopy, pulse shape discrimination and multi-channel coincidence. New evolutions of such applications are constantly proposed thanks to the advances in digital signal processing. The most of them is not yet implemented in real-time instrumentation devices which have to deal with two major issues. The first is the poissonian characteristic of the signal, composed of randomly arriving pulses with variable length [1]. The second is the real-time requirement, which implies losing pulses when the pulse rate is higher than the processing capacity of the device. Indeed, dataflow architectures paralyze the acquisition of the signal during the processing of a pulse implying a dead-time. Many real-time applications such as homeland security and medical imaging have to limit the dead-time to exploit maximum information obtained from the signal. In order to overcome this limitation, recent designs are based on reconfigurable components like Field Programmable Gate Arrays (FPGAs) [2]. However, dedicated hardware algorithm implementations on reconfigurable technologies are complex and time-consuming tasks. Consequently, a Digital Pulse Processing (DPP) architecture that can be programmed in a high level language such as C or C++ is required. However, today's programmable solutions do not meet the need of performance to operate online without increasing the dead-time. This issue becomes more important with the increase of the number of acquisition channels which can exceed more than a hundred [3].

This paper proposes an asynchronous Multiple Program Multiple Data (MPMD) architecture. Its execution model relies on the non-deterministic characteristics of the signal. The paper demonstrates that this architecture is able to overcome dead-time while being programmable and is flexible in terms of number of measurement channels.

II. Related works

This section presents the state-of-the-art of electronic architectures used in nuclear instrumentation to address the need of flexibility, multi-channel scalability and dead-time management without having to design an architecture sized for the worst-case.

The first programmable system which tries to address the dead-time issue is presented in [4]. It is designed for gamma spectrometry and uses a Digital Signal Processor (DSP) associated to a two-level memory hierarchy. The first level stores the output signal directly from the Analog to Digital Converter (ADC) and transmits predefined-length slices of the signal to a second memory level used by the DSP. Additionally, a dedicated analog trigger pre-identifies the presence of pulses in the slices. Then, the DSP uses a Direct Memory Access (DMA) component to read slices of the signal that contain pulses. This architecture has been improved in [5] with the addition of several computing tiles – 2^{nd} memory level and a DSP. If enough computing tiles are present, the dead-time tends to zero. However, this approach is limited to one measurement channel. A more recent work [6] presents an FPGA-based architecture that associates a measurement channel to a processing board. Each of them works independently. Results are then correlated. This architecture introduces the concept of macro-pipeline separating the processing stages, each comprising one pulse processing algorithm. Consequently, this reduces the dead-time to the latency of the slowest pipeline stage. However, the firmware must be redesigned and updated when changing the application which requires experts in signal processing and VHDL/Verilog. This constraint is relieved by the proposed platform in [7]. It combines a FPGA board and a set of predefined firmware that perform many predefined applications' algorithms. This solution partially addresses the issues of programmability. The architecture presented in [2] is the first which is able to perform counting, gamma spectrometry, neutron-gamma discrimination and time-coincidence. It embeds four measurement channels, associated to a computing tile that operates independently and in parallel;

this approach reduces the dead-time. Even if this architecture is closest to a multi-application platform, it is nevertheless dependent on the FPGA firmware that comprises dedicated implementation of the algorithms. Only four measurement channels are managed, since the tiles are implemented within the same FPGA. The architecture presented in [8] separates the pulses from the rest of the signal before any other processing. Then, the pulses are distributed on two parallel processing levels to be processed independently, thus reducing the computing resources requirement and the dead-time. However, the triggering implemented to extract the pulses is not able to fit with the variability of the pulse length and rejects many of them by the use of static windowing. This solution does not offer the flexibility proposed in [2], the dead-time management given by [5] and is not programmable and multi-channel.

Previous works from the literature do not fit with the need of flexibility, multi-channel scalability and dead-time management but several elements of these architectures are used to build our proposal.

III. Architecture Proposal

This section presents the proposed execution model and compares it to the state-of-the-art work.

A. Separating Pulses First

The first step of our proposal is pulse extraction. As shown in [8] and [9] the triggering step can be achieved after the analog-to-digital conversion without deteriorating the signal with shaping and filtering stages. Pileup detection [9] and accurate pulse timestamping are done at this stage. It generates packets of data that contain individual pulses associated with their timestamp. This allows any application downstream this step to work on variable-sized arrays of samples which simplifies the implementation of pulse processing algorithms. These packets can occur at any time depending on the pulse rate.

B. Pulse-Driven Execution Model

Most pulse processing algorithms process each pulse individually (neutron-gamma discrimination [10], spectroscopy [1]) before merging the results in a final process (time correlation, histogram construction). Therefore, once pulses are extracted, it is possible to distribute them individually as soon as they arrive to Functional Units (FUs) which perform process as in [5].

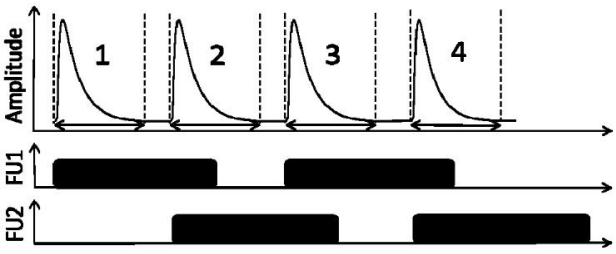

Fig. 1. Dead-time management by the pulses distribution on multiple Functional Units.

Fig. 1 illustrates the occupation of the FUs correlated to the arrival of pulses. In this example, using only one FU would not be enough because of the dead-time imposed by the processing duration which would imply losing pulses. This example also illustrates that zero dead-time is achievable provided that there is a sufficient number of FUs available to process all incoming pulses. In order to manage the pulses distribution over FUs, a scheduler (first-FU-available, first-FU-served) and an interconnection network are used. As the timestamp of each pulse is known, they can be processed individually out-of-order and without synchronization constraint. This leads to an asynchronous execution model which is driven by the arrival of the pulses.

C. Software and Hardware Macro-Pipeline

An acquisition chain is composed of a succession of algorithms. They naturally form a software macro-pipeline as exploited in [6]. As is illustrated in Fig. 2, a simple approach consists in assigning one algorithm per FU. Consequently, the global latency becomes the worst latency of the algorithms executed on an FU. Moreover, as the algorithms are executed locally, *i.e.* on an FU, this approach limits shared memory requirements. It is fully compatible with the asynchronous event-triggered execution model since each FU distribute their packets as soon as possible to the next FU.

D. Shared resources for multichannel applications

Increasing the number of channels implies that several pulse extractors must distribute their pulses to available FUs. Thus, more pulses need to be processed, resulting in an increasing number of required FUs. However, since each channel receives random events independently, the pulse extractors associated with them do not necessarily need to simultaneously distribute their pulses to FUs. Consequently, sharing FUs between channels can reduce the delay to find an available one.

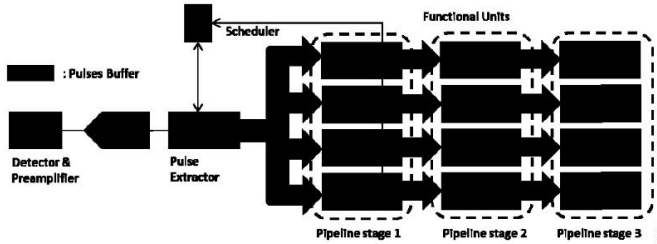

Fig. 2. Proposed pipelined and event-triggered asynchronous execution model.

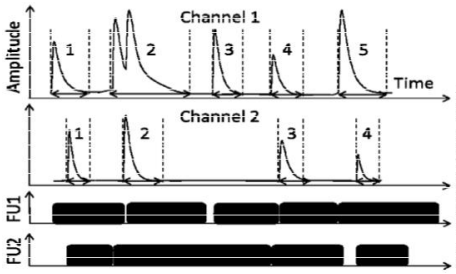

Fig. 3. Pulses distribution on a multichannel architecture with shared resources.

In the example illustrated by Fig. 3, two FUs are required to handle pulses from two acquisition channels. Without FU sharing, the channel 1 alone would require two FUs. Our multi-channel model can then benefit from the use of shared resources to reduce the dead-time issue with the increase of the number of acquisition channels. Moreover, as the processing time of each pulse depends both on the algorithm implementation (in an FU) and the pulse length itself, each pipeline stage can benefit from resources sharing. Therefore, each stage is heterogeneous in term of number of FU and processing capability. A crossbar switch is used between each FU stage. Its routing table is controlled by the scheduler restricted to only one routing command per cycle. Taking into account the granularity of an FU, the crossbar is the best choice to give the best case of the simulation. A further interconnection network that would offer a better scalability for multi-channel applications [3] can then be implemented. The final model that combines all the elements described above is shown in Fig. **4**.

E. Functional Units

As previously illustrated, the proposed event-triggered asynchronous execution model allows the processing of every incoming pulse. In this model, the process duration does not influence dead-time anymore provided that enough FUs is available to process data. FUs are based on general purpose processors (ARM, Microblaze) to execute the algorithms. Algorithm modification does not imply a costly modification of the hardware, but a simple compilation of the application. As each FU is independent with its own clock domain, a specific memory hierarchy must be designed to ensure the data input/output with the rest of the architecture. The current FU is presented in Fig. 5. Dual-clocked FIFOs are used for input and output. Their size is defined to contain the largest pulse that can be transmitted by the pulse extractor. The execution of the algorithm is started by the presence of pulse/packet in the FIFOs. A signal *busy* informs the scheduler the FU is currently processing a pulse. As data processing is completely implemented in software, different kinds of metadata can be added to the pulses (signal is a considered as a variable length table). For example, *Data Emittable* allows stream transmission of samples/data allowing pipelined algorithms (filter) while *End Packet* requires the complete memorization of the pulse/packet in the FIFO OUT before allowing its transmission (check of the validity of a pulse, FFT).

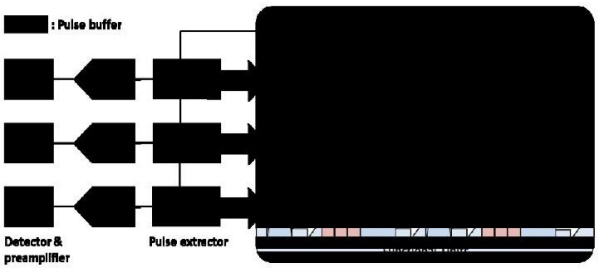

Fig. 4. Proposed DPP architecture model able to meet nuclear instrumentation requirements.

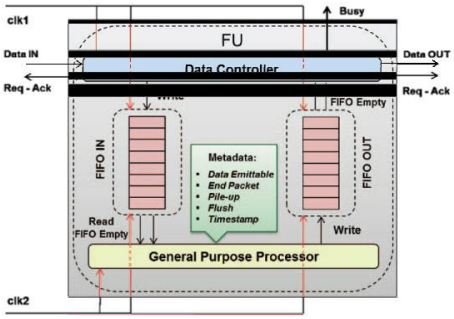

Fig. 5. Internal view of an FU comprising two dual-clocked FIFOs.

IV. EXPERIMENTS AND RESULTS

The macro-model of the proposed architecture is developed in cycle-accurate SystemC. This allows the architecture (number of channels, number of FUs) to be easily modified, different class of algorithms to be implemented and different kinds of interconnection networks, schedulers and execution models to be tested.

A. Experimental Protocol

1) Benchmark

Literature does not propose specific benchmark to compare DPP architectures. However, the works presented in [11] uses a dedicated simulator to evaluate the evolution of dead-time in accordance to a given application and the number of processing tiles used. In the same vein, we propose a benchmark based on the number of pulses lost by the lack of hardware resources (computing, memory etc.). This makes possible fair comparisons between DPP architecture provided that the same dataset and pulse extractors are used.

2) Signal

The data used for tests were obtained with an Am-241 source and a 4γπ crystal-well type NaI(Tl) detector. Analog to digital conversion is ensured by a 16-bit 125 MHz component. The signal is then recorded in a file per 2-second slices, which corresponds to 17 000 pulses when the activity of the source is taken into account. Then a data file is associated with each channel modeled in SystemC.

3) Applications

FUs are programmable, then, software modifications directly impact the program execution duration. In order to be representative of different applications, program durations are expressed in number of cycles required to process each pulse's sample. They are obtained from analysis of traditional pulse processing algorithms.

B. Results

1) Pulses distribution simulation

In order to evaluate the number of FUs required to reach the zero dead-time, we propose to execute a simulation with an instance of the architecture that comprises one acquisition channel. Multiple executions of the model are done using the same data, for different number of FUs and durations of the program execution. The number of pulses lost by the lack of computing resources is recorded and plotted in Fig. **6**. It shows

that the pulses distribution over FUs allows reaching of zero dead-time (no pulse loss), confirming the theory. The gain obtained by adding FUs exponentially decreases as presented in [11]. In our example, no pulse is lost with at least six FUs for each tested configuration of the simulation. It shows that a compromise in term of number of FUs without over-sizing the architecture. This is confirmed by the analysis of the occupation time (FU busy state) of the FUs during the acquisition which is presented in Tab. I. For example, the sixth FU is busy only 0.34 % of the time, it usefulness can be questioned for some application that do not require strict zero-dead time.

2) FU sharing simulation

In order to evaluate the gain, in term of FUs, of FUs sharing between acquisition channels, we propose to execute a simulation with an instance of the architecture that comprises two acquisition channels simultaneously acquiring a signal. The number of pulses lost is recorded for channel 1 only (channel 2 disabled), for channel 2 only (channel 1 disabled), for both channels without FUs sharing and for both channels with FUs sharing. Results are presented in Tab. II and show a reduction of the required number of FUs when FUs sharing is enabled. The number of pulses lost tends to an exponential decay as in the previous results. Using the randomness of the pulses arrival to share resources between channels allows a better scalability for multi-channel applications.

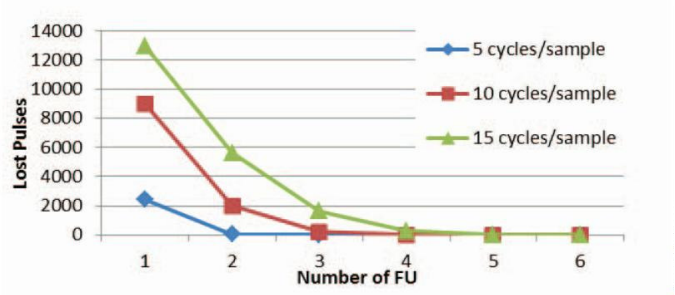

Fig. 6. Simulation results for one acquisition channel compared to the number of FUs required to reach the zero dead-time.

TABLE I. SIMULATION OF THE OCCUPATION TIME OF THE FUS

Cycles /sample	Busy state of FUs during the measurement (in %)					
	FU 1	FU 2	FU 3	FU 4	FU 5	FU 6
15	97.71	N.A	N.A	N.A	N.A	N.A
15	88.43	82.79	N.A	N.A	N.A	N.A
15	81.44	72.78	57.84	N.A	N.A	N.A
15	75.42	68.73	52.28	29.11	N.A	N.A
15	75.20	67.09	49.90	27.19	7.63	N.A
15	75.20	67.09	49.96	26.79	7.63	0.34

TABLE II. SIMULATION OF THE SCALABILITY PROVIDED BY SHARING FUS BETWEEN ACQUISITION CHANNELS.

Acquisition Protocol	Number of FU required			
	15 cycles/s	10 cycles/s	5 cycles/s	2 cycles/s
Channel 1	6	4	3	2
Channel 2	6	4	3	2
Channel 1&2	12	8	6	4
Channel 1&2 sharing FU	9	6	4	2
Gain (%)	25	25	33	50

V. CONCLUSION & FURTHER WORK

Traditional DPP architectures are limited by dead-time, scalability and programmability. To address these issues, an innovative model of event-driven asynchronous DPP MPMD architecture is proposed. This architecture is programmable and is particularly suited to achieve multi-channel digital pulse processing especially for nuclear instrumentation. The pulse separation and pulse distribution to different and pipelined programmable FUs solves the problem of the dead-time and programmability. The issue of scalability in multi-channel applications is addressed by FU sharing. The SystemC model shows promising results in terms of scalability while maintaining zero dead-time on a real experimental dataset. Further work will focus on the interconnection network associated to an optimized scheduler that can exploit the variable-length of pulses allowed by [9] regarding to their scalability for a very large number of FUs.

REFERENCES

[1] G. F. Knoll, *Radiation Detection and Measurement*, 4th ed. Wiley, John & Sons, 2010.

[2] R. T. Schiffer, M. Flaska, S. A. Pozzi, S. Carney, and D. D. Wentzloff, "A Scalable FPGA-based Digitizing Platform for Radiation Data Acquisition," *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 652, no. 1, pp. 491–493, Oct. 2011.

[3] D. Bazzacco, "The Advanced Gamma Ray Tracking Array AGATA," *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 746, pp. 248–254, Dec. 2004.

[4] J. Basilio Simoes and C. M. B. . Correia, "Pulse Processing Architectures," *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 422, no. 1–3, pp. 405–410, Feb. 1999.

[5] J. M. . Cardoso, J. Basilio Simoes, and C. M. B. . Correia, "A High Performance Reconfigurable Hardware Platform for Digital Pulse Processing," *Nucl. Sci. IEEE Trans.*, vol. 51, no. 3, pp. 921–925, 2004.

[6] S. Normand, V. Kondrasov, G. Corre, and C. Passard, "PING : A New Approach For Nuclear Fuel Cycle Instrumentation," *1st Int. Conf. Adv. Nucl. Instrumentation, Meas. Methods their Appl.*, pp. 1–4, Jun. 2009.

[7] CAEN, "WP2081 Digital Pulse Processing in Nuclear Physics," no. August. 2011.

[8] P. Lee, C. Lee, and J. Lee, "Development of FPGA-based Digital Signal Processing System for Radiation Spectroscopy," *Radiat. Meas.*, vol. 48, pp. 12–17, 2012.

[9] Y. Moline, M. Thevenin, G. Corre, and T. Peyret, "Procédé et système d'extraction dynamique d'impulsions dans un signal temporel bruité," *Pat. number FR14 50568*, 2014.

[10]M. Moszynski, G. Bizard, and G. Costa, "Study of n-γ Discrimination by Digital Charge Comparison Method for a Large Volume Liquid Scintillator," *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 317, no. 1–2, pp. 262–272, 1992.

[11]J. M. . Cardoso, J. Basilio Simoes, and C. M. B. a. Correia, "Dead-time Analysis of Digital Spectrometers," *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 522, no. 3, pp. 487–494, Apr. 2004.