A Comparison of FPGA Architectures to Extract Gamma Arrival Times from Multiple-timestamp Digital SiPM PET Detectors

Leonardo Gasparini, *Member, IEEE*, Davide Mariz,
Roberto Passerone, *Member, IEEE* and David Stoppa, *Member, IEEE*

***Abstract*– Recently proposed Digital Silicon Photo-Multipliers integrate arrays of Time-to-Digital Converters thus providing multiple photon timestamps for each detected gamma. A considerable amount of data is generated and has to be processed to estimate the actual time of arrival of the gamma. This is typically performed on the FPGA present on the controller board. The processing stages include: (i) back conversion from TDC code to time, (ii) sorting, (iii) finding the timestamp of the first scintillation photon, removing noisy events (dark counts), (iv) and final processing to extract the Time of Arrival. In this work we discuss the implementation of two architectures specifically designed for this purpose and analyze their impact in terms of device utilization and timing. The architectures have been tested with the data generated by the SPADnet-I sensor, a 8×16-pixel d-SiPM, that generates up to 256 photon timestamps for every detected gamma.**

I. Introduction

DIGITAL Silicon Photo-Multipliers (d-SiPMs) specifically designed for PET applications provide multiple timestamps for each gamma event [1]. The large amount of generated data needs then to be processed, for example on an FPGA present on the controller. Processing must be simple and efficient; ideally, the logic must be fast, low power and require little resources to enable on-chip implementation on next generation of devices. In the present work, two FPGA implementations of an algorithm to extract the arrival time of a gamma photon are compared. The algorithm is based on the data provided by the SPADnet-I sensor, a 8×16-pixel d-SiPM which provides for each pixel the timestamp of the first photon being detected in a gamma event, coupled to an array of crystal pins.

II. Algorithm

One of the advantages of d-SiPM in CMOS technology is the possibility to implement multiple functionalities on the chip itself. State-of-the-art PET d-SiPMs [1] [2] integrate multiple photon counters and time-to-digital converters (TDCs) on the silicon, thus generating a considerable amount of data. In a typical PET system, an FPGA is used to control an array of d-SiPMs, buffer the data and process the generated information [3]. Such a system has severe memory and bandwidth requirements. To relax the system constraints, the ideal d-SiPM should include the logic to reject noisy events, perform energy windowing, locate the position of the gamma absorption and extract the time of arrival of the gamma. In this way, the system would be optimized in terms of efficiency (here defined as the number of gamma events actually used for image reconstruction over the total number of detected gammas), bandwidth and processing speed.

In this context, the processing of timing information is the most challenging and critical step. In general, it requires the following steps (not necessarily in this order):

- Back-conversion from the digital to the time domain;
- Sorting;
- Dark count filtering to find the first detected photon;
- Arithmetic processing to generate the gamma arrival time.

This work compares two different implementations of a processing unit that implements these operations. Despite being designed for a specific d-SiPM, i.e. the SPADnet-I sensor [2], the algorithm can be applied to any multiple-timestamp d-SiPM detector and the analysis is meant to provide a tool for IC designers to integrate this processing stage on the next generation of devices.

The SPADnet-I sensor has an internal triggering mechanism for gamma detection and provides two 12-bit TDC codes for every pixel. The two TDC codes belong to consecutive clock bins so that the information is preserved when the scintillation rises during the active edge of the FPGA clock. Figure 1 shows a sample event. Figure 2 shows the architecture with the processing stages used to extract the gamma arrival time from the TDC data.

The conversion of TDC codes back to the time domain requires a full-scale compensation and offset cancellation. Due to manufacturing limitations, each TDC in the array has a characteristic operating frequency, resulting in individual full scale values.

Filtering of dark events occurs at two separate stages. In the first one, the spatial information associated to each TDC is used. We take into account only the codes generated by pixels

Manuscript received April 3, 2014. (Write the date on which you submitted your paper for review.) This work was conceived within the SPADnet project (www.spadnet.eu), supported by the European Community within the Seventh Framework Programme ICT Photonics.

L. Gasparini and D. Stoppa are with the Center for Materials and microsystems, Fondazione Bruno Kessler, 38123 Povo (TN), Italy (telephone: +39-0461-314-182, e-mail: gasparini/stoppa@fbk.eu).

D. Mariz and R. Passerone are with the Dept. of Information Engineering and Computer Science, University of Trento, 38123 Povo (TN), Italy (telephone: +39-0461-28-3971, e-mail: roberto.passerone@unitn.it).

978-1-4673-9862-6/15/$31.00 ©2015 IEEE

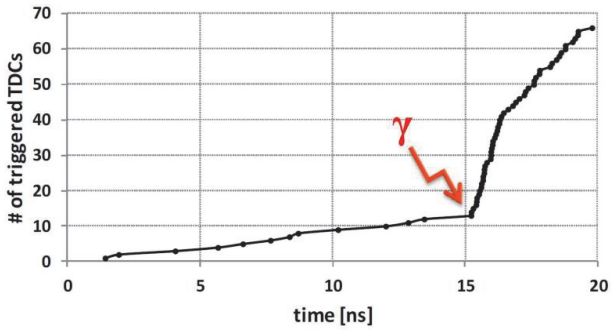

Figure 1 – Cumulative distribution of photon timestamps for a sample scintillation event. The gamma arrives at about 15ns. Previous detections represent dark counts.

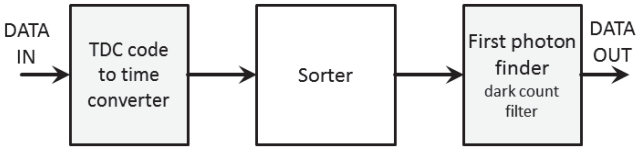

Figure 2 – Block diagram of the Time-of-Arrival estimator. The sorter is the most intensive processing stage. Two sorting architectures have been implemented and are here compared.

laying in the proximity of the crystal pin that absorbed the gamma and remove the others (a simple center-of-gravity algorithm can be used to identify the pin).

On a later stage, the system exploits the temporal correlation of the scintillation photons to remove potential dark counts occurred prior to the gamma absorption. A temporal derivative filter sorts the time values, calculates the inter-arrival time of photons and applies a threshold to them. Then it identifies the first scintillation photon when it finds three consecutive inter-arrival times below the threshold.

Further processing can then be applied. For example, the second or the third photon can obtain better performance, or a combination of them [4].

III. Sorting

In this process, sorting is the most resource hungry operation. This is a well-known problem in the literature as it is used in multiple domains, such as signal processing (e.g., median filter), computer graphics (e.g., to display objects according to their position in the 3D space), and database analysis. We have considered two FPGA implementations of the sorter, based on their suitability to the specific system (as TDC values are read out sequentially from the sensor, pipelined processing can be performed), processing speed and logic requirements, and easiness of implementation. The first implements the "insertion sort" method [5], the other one is a modified version of the sorter described in [6] (here referred to as "cyclic sorter"). Both are based on a set of identical sorting units (SU) connected in series and working in parallel.

Figure 3 shows the block diagram of the insertion sorter. Each unit is made of a multiplexer, a comparator, and a register. When a new piece of data is read

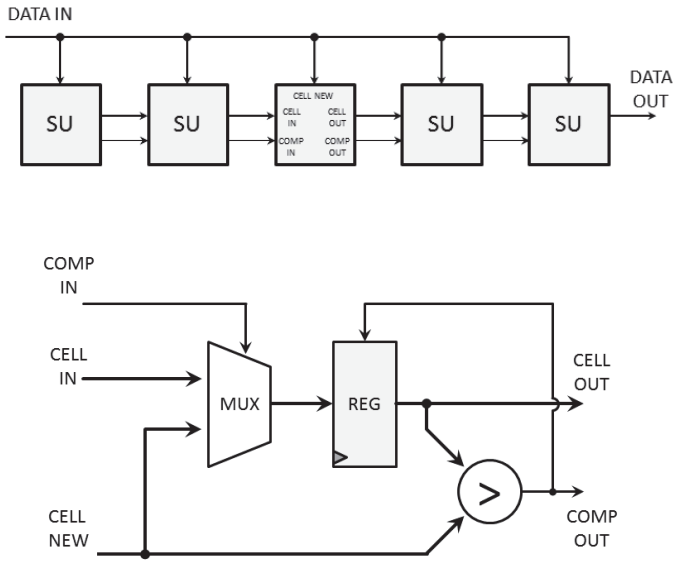

Figure 3 - Block diagram of the insertion sorter (top) and of its sorting units (bottom). The sorter is built placing a number of sorting units in series. The sorting units contain a multiplexer, a register and a comparator. When a new piece of data is provided, each sorting unit compares it with the value stored in its own register. This is done in parallel for all the sorting units. All the sorting units having a value in the register larger than the new one retain their value. Due to the sorting mechanism, these will be all located in the left part of the chain. The first element for which the new value is larger than the one stored in the register will replace it, and the old value will be transferred to the next unit. All the following sorting units will also update their register, replacing the old value with the one of the previous cell.

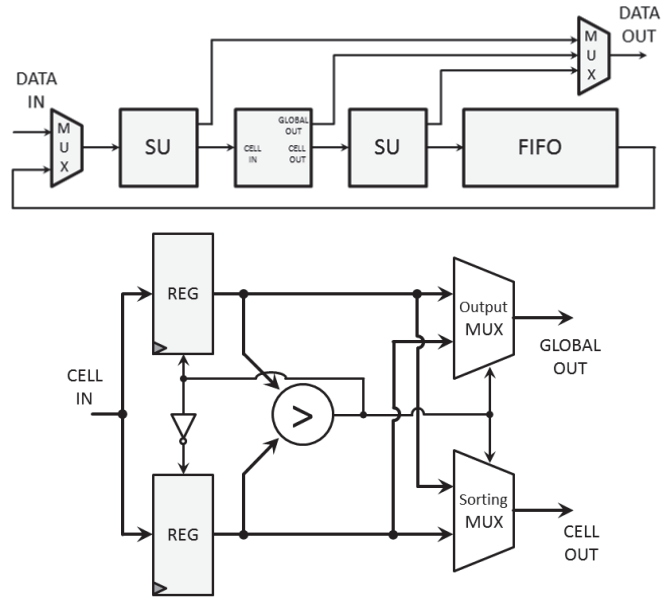

Figure 4 - Block diagram of the cyclic sorter (top) and of its sorting units (bottom). The sorter is built placing a number of sorting units in series and a FIFO memory. The sorting units contain a couple of registers, a comparator, and two multiplexers. When a new piece of data is provided, it is compared with all the previous values in a cyclic process. The new value is fed only to the first sorting unit. Each cell compares the values stored in its own registers and replaces the larger one with the input value, coming from the previous cell or from the input pins. The value being replaced is moved to the next cell. In order to fully sort the sequence, data needs to be cycled throughout until all the values are compared. The FIFO is needed to reduce the number of sorting units, saving area but requiring a longer sorting time.

out from the sensor each cell compares it with the one present in its own register. If the new value is larger, the one stored in memory is moved to the next cell, while the register is updated with either the new value or the value coming from the previous cell, according to the result of the comparison of the previous cell, otherwise it retains the present value. The circuit may suffer from poor scalability since the new piece of data is provided to all the sorting units at the same time.

The cyclic sorter avoids potential issues by employing the scheme depicted in Figure 4. The new piece of data is provided only to the first cell and then compared to all the other elements in the chain in separate instants. Each cell has two registers, a comparator and two multiplexers (here referred to as "sorting MUX" and "output MUX"). At every cycle, the input data replaces the smallest piece of data in the cell, which is in turn moved to the next cell through the sorting MUX. A flag is required to keep track of the number of comparisons performed and control the output MUX, which is connected to the sorter's global output. The cyclic sorter provides a certain degree of flexibility as we can combine a number of sorting cells with a First-In, First-Out (FIFO) memory. The larger the number of sorting cells, the shorter is the latency to generate the sorted sequence, at the expense of a larger area occupied.

IV. FPGA Implementation

The two architectures have been implemented on a low-cost, low-power Xilinx Sparan-3 xc3s400-4 FPGA. The architecture based on the insertion sorter takes 530 clock cycles to extract the gamma arrival time. In order to compare the two architectures, the cyclic sorter has been implemented using a number of sorting cells leading to the same number of clock cycles to sort 256 elements, i.e., 113 sorting cells.

The architecture based on the insertion sorter requires fewer logic resources and supports a higher operating frequency. The cyclic sorter suffers from the presence of a large multiplexer at the output that limits the speed of the circuit.

Note that the SPADnet-I sensor generates 256 TDC codes. So far, we have considered a worst case scenario in which all pixels detect a photon in the first clock bin of the scintillation process. But when a pinned crystal scintillator is used, the number of pixels (and therefore of informative TDC codes) involved in the acquisition process is typically limited to 25-36 pixels, according to the pin size. In this configuration, the system requires fewer sorting units, and the overall occupancy scales down accordingly.

Table 1 compares the implementations in terms of logic resources and timing performance. Results suggest that the cyclic sorter can provide a good compromise between area and speed for an on-chip implementation only if the number of sorting cells is limited.

TABLE I. COMPARISON OF THE THREE IMPLEMENTATIONS

Architecture #	**1**	**2**	**3**
Sorter type	insertion	Cyclic	Cyclic
Num. of sorting units	256	113	36
Slice Flip-flops	2,437 (34%)	2,581 (36%)	(12%)
4-input LUT	5,519 (77%)	5,687 (79%)	(29%)
Total Slices	2939 (82%)	3,582 (99%)	(39%)
16K Block RAM	2/16	3/16	3/16
Multipliers	1	1	1
Maximum clock freq.	63MHz	39MHz	70MHz
Sorting time for 256 valid TDC codes	8.40µs	13.59 µs	25.71 µs (estimated)
Sorting time for 36 valid TDC codes	-	-	3.70 µs (estimated)

References

[1] A. Carimatto, S. Mandai, E. Venialgo, Ting Gong, G. Borghi, D.R. Schaart, E. Charbon, "11.4 A 67,392-SPAD PVTB-compensated multi-channel digital SiPM with 432 column-parallel 48ps 17b TDCs for endoscopic time-of-flight PET," *Solid- State Circuits Conference - (ISSCC), 2015 IEEE International* , pp.1,3, 22-26 Feb. 2015.

[2] L. H. C. Braga, L. Gasparini, L. Grant, R. K. Henderson, N. Massari, M. Perenzoni, D. Stoppa, and R. Walker, "A Fully Digital 8x16 SiPM Array for PET Applications With Per-Pixel TDCs and Real-Time Energy Output," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 301–314, Jan. 2014.

[3] C. Degenhardt, B. Zwaans, T. Frach, R. de Gruyter, "Arrays of digital Silicon Photomultipliers — Intrinsic performance and application to scintillator readout," *Nuclear Science Symposium Conference Record (NSS/MIC), 2010 IEEE* , pp.1954,1956, Oct. 30 2010-Nov. 6 2010

[4] L. Gasparini, L.H.C. Braga, M. Perenzoni, and D. Stoppa, "Characterizing single- and multiple-timestamp time of arrival estimators with digital SiPM PET detectors," *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2013 IEEE*, vol., no., pp.1,4, Oct. 27 2013-Nov. 2 2013

[5] R. Marcelino, H. Neto, and J.M.P. Cardoso, "Sorting units for FPGA-based embedded systems." In *Distributed Embedded Systems: Design, Middleware and Resources*, pp. 11-22. Springer US, 2008.

[6] E. Mumolo, G. Capello and M. Nolich, "VHDL design of a scalable VLSI sorting device based on pipelined computation." *CIT. Journal of computing and information technology* 12, no. 1 (2004): 1-14