ToF Estimation Based on Compressed Real-Time Histogram Builder for SPAD Image Sensors

I. Vornicu, A. Darie, R. Carmona-Galán, Á. Rodríguez-Vázquez
Instituto de Microelectrónica de Sevilla, IMSE, CNM (CSIC, Universidad de Sevilla)
E-mail: ivornicu@imse-cnm.csic.es

***Abstract*— This paper presents a FPGA implementation of a novel depth map estimation algorithm for direct time-of-flight CMOS image sensors (dToF-CISs) based on single-photon avalanche-diodes (SPADs). Conventional ToF computation algorithms rely on complete ToF histograms. The next generation of high speed dToF-CIS is expected to have wide dynamic range and high depth resolution. Applications such as 3D imaging based on dToF-CISs require pixel-level ToF histograms which have to be stored by huge fully-random access memory (RAM) modules. The proposed shifted inter-frame histogram (SiFH) algorithm has the same accuracy but requires a memory footprint 128 times smaller than the conventional algorithm. Thus a much larger number of pixels can be resolved using limited block RAM resources of FPGAs. Moreover the overall frame rate is also remarkably improved compared to the scanning method. The proof of concept of the SiFH algorithm on 15 bits has been implemented on Spartan-3E. An automated testbench was developed to confirm that no ambiguity errors occur along the entire dynamic range.**

Keywords—time-of-flight (ToF) computation algorithm, single photon avalanche diode (SPAD), compressed temporal histogram, FPGA implementation, 3D imaging

I. Introduction

DIRECT TIME-OF-FLIGHT CMOS Image Sensors (dToF-CIS) based on Single-Photon Avalanche-Diodes (SPADs) are constantly improving their performances for 3D imaging applications [1]. The depth map of the scene is obtained by taking a ToF measurement at pixel level. Due to the statistical nature of the photon detection and the noise along the signal path, ToF is computed across multiple measurements. The conventional method to accurately calculate the ToF is to build Complete inter-Frame Histograms (CiFH) [2], [3]. CiFH works well for streak imagers, coding the ToF on a small number of bits which involves reasonable histogram memory footprint. Nevertheless larger imagers with longer distance range and/or higher depth resolution require a huge amount of Random Access Memory (RAM). For instance, storing CiFHs of 12 bits per bin for an array of 64x64-pixels on 15 bits demand 1.5 Gbits of memory. A frame rate of 100 kfps means a transfer rate to the histogram memory of 6 Gbps which definitely calls for multiple channels. For example, 32 channels lead to a fully random bin resolve time of 75ns. Although Block RAM (BRAM) modules of high performance FPGAs match the speed requirements, the memory resources are far exceeded [4]. DDR memories are also ruled out, being optimized for burst mode rather than random access of memory locations. Moreover, multiple channels are not supported. At this point a cost effective solution is the time gate scanning approach. Thus ToF readings are coded on less number of bits, i.e. less memory resources are needed [5], [6]. However the overall frame is significantly impaired, being inversely proportional to the required number of time gates.

This work concentrates on the Spartan-3E FPGA implementation of a proof of concept of the Shifted inter-Frame Histogram (SiFH) algorithm. This innovative algorithm overcomes the memory resources limitation of CiFH algorithm, e.g. a 15 bits histogram demands 128 times less storage memory. Moreover, unlike the scanning approach, the overall frame rate is only decreased by 2 and does not depend on the dynamic range. The most challenging aspects of this implementation are discussed as well, such as memory reset strategy and automated test setup for static characterization.

II. SiFH Algorithm

Let us suppose that the pixel readings are coded on N_P bits. Thus the corresponding complete histogram would have 2^{N_P} bins. Instead, SiFH algorithm relies on 2 histograms, coarse (CH) and fine (FH), each one of 2^{N_S} bins, $N_S < N_P$. The ToF is computed in 2 steps: i) CH is built from the N_S most significant bits out of N_P. The peak position of CH, $x_{p,c}$ is detected, stored and used to compute 2 thresholds:

$$TH_+ = 2^{N_P-N_S}x_{p,c} + SB;\ TH_- = 2^{N_P-N_S}x_{p,c} - SB \quad (1)$$

$$\forall x_{p,c},\ 1 < x_{p,c} < 256;\ SB = 2^{N_S-1}$$

At this point, the ToF information of CiFH is estimated to be somewhere in between the thresholds.
ii) CH is overwritten by the FH which is built from the incoming pixel values filtered between TH_- and TH_+. Remember that the pixel values are represented on N_P bits. Therefore they have to be shifted downwards with Δ in order to be indexed into a N_S bits histogram.

$$\Delta = \left\lfloor \left(\frac{2^{N_P-N_S}x_{p,c}+SB}{2^{N_S}} \right) - 1 \right\rfloor 2^{N_S} + \mathrm{mod}\left(\frac{2^{N_P-N_S}x_{p,c}+SB}{2^{N_S}} \right) \quad (2)$$

Finally, the peak position of FH, $x_{p,f}$ is shifted upwards to compute the ToF.

$$ToF = x_{p,f} + \Delta \quad (3)$$

978-1-7281-0397-6/19/$31.00 ©2019 IEEE

It is worth to mention that the ToF has been computed without storing the CiFH. Thereby SiFH algorithm effectively locks on the ToF information represented by the pixel readings that appear most of the time. The rest of the samples are disregarded, being considered part of the CiFH noise floor.

III. SiFH Implementation

The proof of concept of the SifH algorithm was implemented on Spartan-3E for a single pixel and optimized to use minimum resources. However, the design is portable and can be easily scaled for larger arrays of pixels. N_P and N_S are set to 15 and 8 bits respectively. The block diagram of the algorithm is presented in Fig. 1.

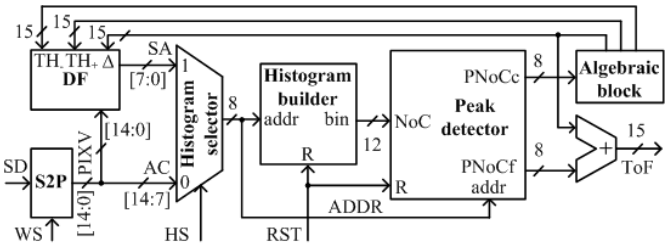

Fig. 1. SifH block diagram

It contains the following modules:

A. Serial-to-Parallel converter (S2P)

The pixel readings are serially transferred on SD input at CLK of 50 MHz. Each pixel value, PIXV is stable for 300ns. This is the time budget to place each PIXV on the proper histogram bin. The end of PIXV is signaled by WS.

B. Digital filter (DF)

It is enabled to build the FH, i.e. HS signal is high. *DF* filters the incoming pixel values such that $TH_- < \text{PIXV} < TH_+$. Further on, Δ is subtracted from the filtered PIXV to obtain a valid histogram address on 8 bits.

C. Histogram selector

It provides the proper addresses to the *Histogram builder* as follows: *i)* CH is selected when HS is set low. ADDR is connected directly to the 8 MSBs of PIXV. *ii)* FH is selected when HS is set high. ADDR is provided by the *DF* module.

D. Histogram builder

CH and FH are sequentially mapped in the BRAM. It is configured as a single port memory in read-before-write mode. Each ADDR value is pointing to a 12 bits memory location (histogram bin) whose content is incremented by 1 at a time. It involves a read-write operation which takes only $3T_{CLK}$ out of $15T_{CLK}$ available. To store the histogram for a single pixel, only 3 kbits are needed instead of 384 kbits required by the CiFH algorithm. To the best of our knowledge, this is the highest histogram memory compression rate without any accuracy loss ever reported. Further compression involves uncertainty errors and frame rate decrease [7].

After completing the CH, the BRAM has to be cleared to acquire the FH. The problem is that BRAMs modules do not have a reset signal. A time efficient reset mechanism is proposed. The reset is issued during the accumulation phase of the FH. Thus it does not introduce any latency. Further details are provided in Section IV.

E. Peak detector

It is looking for the CH/FH peaks in real time. It constantly updates the maximum **N**umber **o**f **C**ounts (NoC) per bin every time a new PIXV is resolved. The corresponding ADDR of the bin is stored as well. The position of CH/FH peak, PNoCc/PNoCf is available right away after CH/FH is completed, i.e. after M samples were acquired.

F. Algebraic block

It calculates TH_-, TH_+ and Δ according to the eqs. (1-2). Note that $x_{p,c}$ is the value of PNoCc after CH is completed. The full adder located in Fig. 1 below the Algebraic block is employed to compute the ToF according to eq. (3).

IV. Memory clearing mechanism

CH and FH are sequentially built in the same BRAM. Therefore, after CH is completed and TH_-, TH_+, Δ are calculated, the BRAM must be reset for the FH. BRAMs modules based on 6T-SRAM cells do not have a reset signal. In the following, two different clearing approaches are considered. They have been incorporated in the *Histogram builder* module.

A. Sequentially clearing mechanism (SeqCM)

This straightforward approach is presented in Fig. 2. As long as clrMem signal is set low, the incoming ADDR is counted in the corresponding histogram bin. When clrMem is set high, the clearing mechanism is activated. The address input of the BRAM is connected to an 8 bits counter (CNT8). On the negative edge of the clock, all 256 addresses are swept one by one while the data input is set low by the AND gate.

Despite its simplicity, this clearing mechanism introduces a latency of $256T_{CLK}$, affecting the ToF computation rate.

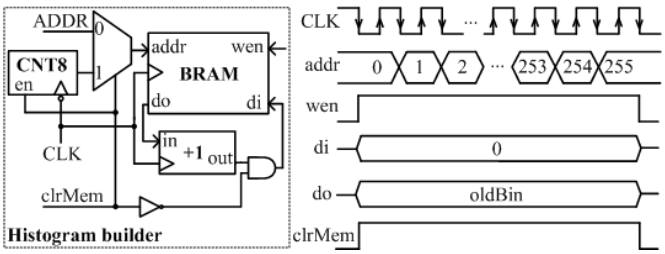

Fig. 2. Sequentially clearing mechanism

B. Signaled clearing mechanism (SigCM)

This approach does not involve latency at all because the histogram bins are initialized on the fly, when it is needed. The block diagram and the signal chronogram which explains the operation principle are presented in Fig. 3. Basically each incoming ADDR is processed by the *Histogram builder* differently as follows: *i)* suppose that ADDR appears for the first time during the accumulation of a CH/FH. In this case the SEL signal is set low. This means that the corresponding histogram bin (oldBin) belongs to a previous histogram. Therefore the old bin value is set to 1 (see Fig. 3–L side); *ii)* if the incoming ADDR has appeared before, the corresponding

histogram bin is updated by incrementing its previous value by 1. The SEL signal is set high (see Fig. 3-R side).

The module *Reset memory* decides whether the current histogram bin has to be initialized or just updated by incrementing its previous value. The decision is made upon the selection signal (SEL) of the multiplexer.

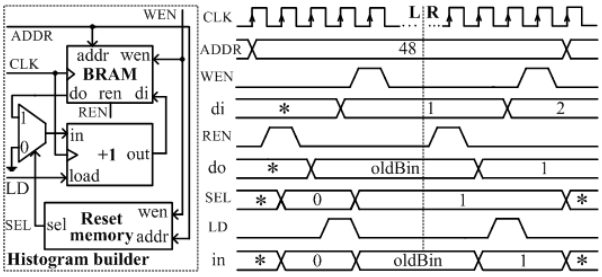

Fig. 3. Signaled clearing mechanism

The block diagram of the *Reset Memory* is depicted in Fig. 4. The array of latches is reset before every CH/FH. Each SR latch is used to keep track of a histogram bin such that Q = 0 if the corresponding ADDR appears for the first time, otherwise Q = 1. After CH/FH is completed, the state of the latches can be used for histogram readout. If $Q_n = 0$, $\forall$ n = [0:255] then the address of the nth bin has been never hit. Therefore this bin has to be readout as 0.

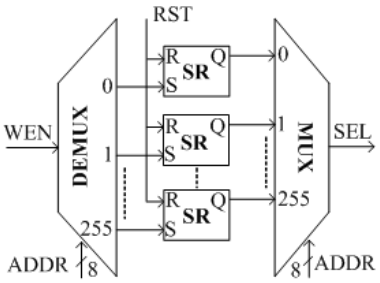

Fig. 4. *Reset Memory* block diagram

V. Measurement Results

The SiFH algorithm was implemented as a proof of concept on a Spartan-3E FPGA, incorporating both clearing mechanisms. The hardware design was optimized in terms of resources (see Table I) in order to be easily scaled up for large arrays of pixels.

The experimental setup is presented in Fig. 5. The data set loaded into the **P**attern **G**enerator (PG) was extrapolated based on raw captures taken by a SPAD camera prototype which has been designed in-house [8]. Further on, the data samples are processed by the SiFH hardware (DUT) to compute the ToF. The control signals required by the SiFH circuit design are generated inside the DUT module. Finally the experimental results are downloaded by a **L**ogic **A**nalyzer (LA) to be compared to the software implementation results of the algorithm.

To have a better understanding of how the experimental setup works, let us consider one test file containing 32240 pixel values on 15 bits. PG and DUT are synchronized on the same CLK signal.

PG issues a Start pulse to send the pixel values to DUT as follows: *i)* DUT sends a wait0 signal, forcing the PG to stop the execution of the program sequence.

TABLE I. FPGA Resource Utilization

Resource	Available	Utilization SeqCM/ SigCM
Slices	9312	391 (4%)/ 304 (3%)
4-input LUTs	9312	410 (4%)/ 1096 (11%)
RAMB16s	20	1 (5%)

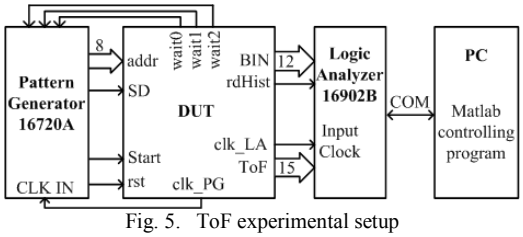

Fig. 5. ToF experimental setup

After 80 ns dead-time, PG stars to send the pixel values on the serial input SD. Each pixel value takes 320 ns to be resolved by the DUT. Consequently CH is completed in 11ms. *ii)* Next, PG is stopped by the signal wait1. After another 80 ns, CH is readout by LA through the BIN port. The 8 bits addresses of CH are generated by PG. The readout takes 5.12µs. CH is presented in Fig. 6. *iii)* Next, the first two steps are repeated with the same test file in order to obtain the FH which is depicted in Fig. 7. Finally, PG is stopped by the signal wait2. After 80 ns delay, the computed ToF is readout by LA through the ToF port.

The ToF is computed with the same accuracy as CiFH, although the complete histogram is never built. This is explained by the fact that the FH is actually the CiFH by considering only the histogram bins located between the thresholds. As shown in Fig. 6, the thresholds are computed based on the position of the CH peak. CH is used to estimate straight away the location of the ToF information in the CiFH. The ToF computation is performed in the second acquisition based on the FH. Therefore the ToF **C**omputation **R**ate (ToF-CR) of SiFH only decreases to one half compared to the CiFH approach which requires only one acquisition.

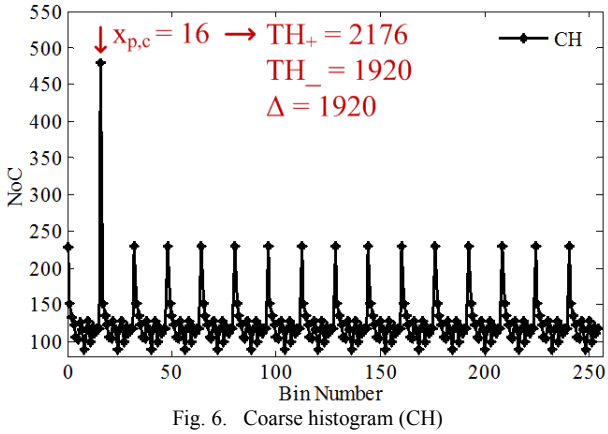

Fig. 6. Coarse histogram (CH)

This drawback can be overcome by doubling the CLK frequency. However, the obtained ToF-CR is far better than the one of the scanning techniques which would require 128 acquisitions, i.e. ToF-CR decreases by 128 times.

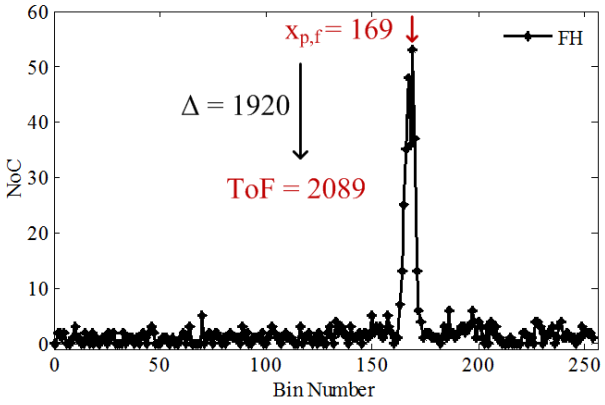

Fig. 7. Fine histogram (FH)

In order to prove that the hardware implementation of the SiFH algorithm is free of uncertainty errors, the histogram peak has to be swept across the entire bin range. Thus, 2^{15} ToFs have to be computed as explained before. This means that PG has to load 2^{15} test files, each one containing 64480 pixel values on 15 bits. But PG has only 16M vectors of memory, loading only 16 test files at a time. Moreover, data loss is circumvented by gating the LA which has a limited buffer of 4M vectors.

An automated testbench has been developed to run the huge amount of test files. The Matlab interface using COM server is able to run on a PC terminal. Thus, large test files can be imported into the PG without any size restriction, regardless of the PG limited internal memory. It also allows transfer the measurement results from the LA to a PC terminal for further processing. The static characteristic of the SiFH algorithm has been automatically extracted (see Fig. 8). All ToFs are computed without any ambiguity error along the entire bin range. ToF_{min} and ToF_{max} are the minimum and maximum ToF values that can be accurately computed. These values have to take into account the maximum deviation of the ToF information. Consequently, this limitation appears also in the CiFH.

It is worth to mention a particular case when $x_{p,c}$ is either 1 or 256. In this case eq. (1) does not apply for TH_{-} and TH_{+} which instead are set to the appropriate predefined values.

Thanks to the compressed implementation of per-pixels ToF histograms, the proposed algorithm requires a much smaller histogram memory. Thus the hardware required by the implementation of the SiFH algorithm for a typical 32×32-pixels array fits into a Spartan-6 FPGA with less than 5 Mbits of block RAM.

VI. Conclusions

This paper concentrates of the FPGA implementation of a proof of concept of the SiFH algorithm. It is a novel inter-frame histogram-based ToF computation algorithm for high performance dToF-CISs.

SiFH has been demonstrated to accurately compute the ToF, free of ambiguity errors. For large ToF depths, such as 15 bits, SiFH requires 128 times less memory than the conventional CiFH approach.

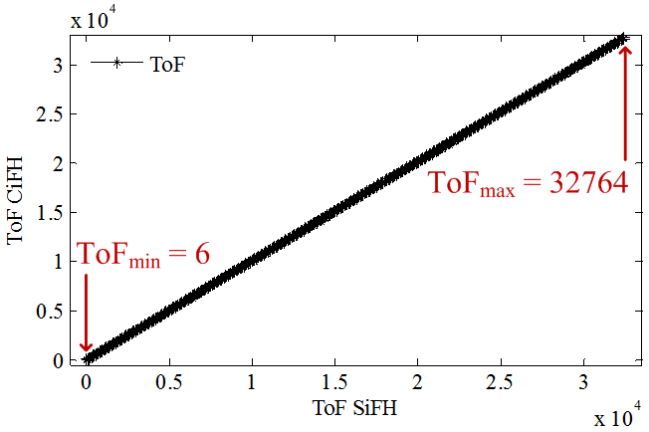

Fig. 8. Static characteristic

Instead, ToF-CR only decreases by half. However compared to the scanning techniques, ToF-CR is improved at least by 64 times. The FPGA block RAM memory has been used to store as fast as possible the compressed histograms, CH/FH. The memory modules do not have a reset signal. Therefore, two clearing mechanisms have been contemplated. The hardware implementation can be easily scaled up for larger pixel arrays.

Acknowledgment

This work has been mainly funded by the Office of Naval Research (USA) ONR, grant No. N00014-19-1-2156, the Spanish MINECO and the European Region Development Fund (ERDF/FEDER) through project 'iCaveats' (Ref. TEC2015-66878-C3-1-R), and partially supported by Junta de Andalucía through project 'SmartCIS3D' (Ref. TIC 2338-2013).

References

[1] M. Perenzoni, D. Perenzoni, D. Stoppa, "A 64×64-Pixels Digital Silicon Photomultiplier Direct TOF Sensor With 100-MPhotons/s/pixel Background Rejection and Imaging/Altimeter Mode With 0.14% Precision Up To 6 km for Spacecraft Navigation and Landing", J. of Solid-State Circ., Vol. 52, No. 1, Jan. 2017

[2] N. Dutton, S. Gnecchi, L. Parmesan, A. J. Holmes, et al, "A time-correlated single-photon-counting sensor with 14GS/s histogramming time-to-digital converter", Int. Solid-State Circ. Conf., Sens. and Imag. for Life Sci., Session 11, pp. 204-206, 2015

[3] N. Dutton, J. Vergote, S. Gnecchi, L. Graant, et al, "Multiple-event direct histogram TDC in 65nm FPGA technology", Ph.D. Research in Microelectronics and Electronics, 2014

[4] Xilinx, 7 series FPGAs memory resources. Available from: https://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf

[5] SPAD array with gated histogram construction, by A. K. Sharma, A. Laflaquiere, et al. (2017, Feb. 23). Patent US2017/0052065A1.

[6] Time to digital converter and applications thereof, by N. Dutton, K. Henderson, S. Gnecchi. (2015, Feb. 12). Patent US2015/0041625A1.

[7] S. Lindner, Chao Zhang, M. Wolf, E. Charbon, "A 252 x 144 SPAD pixel FLASH LiDAR with 1728 Dual-clock 48.8 ps TDCs, Integrated Histogramming and 14.9-to-1 Compression in 180nm CMOS Technology", Symposium on VLSI Circ. Digest of Tech. Papers, Honolulu, June 2018.

[8] I. Vornicu, R. Carmona-Galan, A. Rodriguez-Vazquez, "Live demonstration: Photon counting and direct TOF camera prototype based on CMOS SPADs," Int. Symp. on Circ. and Syst., Baltimore, MD, USA, 2017