Sand Castle Summation For Pixel Processor Arrays

Laurie Bose[1] Piotr Dudek[2] Jianing Chen[2] Stephen J. Carey[2]
[1]University of Bristol, Bristol, United Kingdom
[2]University of Manchester, Manchester, United Kingdom

***Abstract*—Pixel Processor Arrays (PPA) present a new vision sensor/processor architecture consisting of a SIMD array of processor elements, each capable of light capture, storage, processing and local communication. Such a device allows visual data to be efficiently stored and manipulated directly upon the focal plane, but also demands the invention of new approaches and algorithms, suitable for the massively-parallel fine-grain processor arrays. In this paper we implement an image-wide population count algorithm exploiting the parallel processing of the PPA. Performing such a global count was previously unviable for vision processing tasks due to its exhaustive computation time. Our approach shows an improvement of typically two orders of magnitude reduction in computation time, thus allowing it to be incorporated as a core component of many vision tasks upon the PPA.**

I. Introduction

Recent trends in edge computing bring to the fore the concerns about power efficiency of the processing hardware. Some of the most challenging applications are in computer vision, where large amounts of raw sensory data (image pixels) need to be processed. It is well known that data movements are currently the most critical operations responsible for energy consumption as well as the overall speed of the system. Minimising external memory access has become a necessity, and one of the solutions is a distributed architecture, with memory and processing resources collocated on a single device. As many low-level image processing tasks are inherently parallel, with computations localised (results are dependent on pixels and their neighbours), and identical operations executed for all pixels in the image, they are well suited to massively parallel SIMD (Single Instruction Multiple Data) architectures. An extreme level of parallelism can be achieved by allocating a processor per pixel, in a fine-grained SIMD architecture. A very large number of processing elements, each containing local memory and arithmetic logic units, can efficiently execute pixel-parallel algorithms. Such cellular processor architectures have been considered in the past [1]–[4]. With more recent advances in silicon fabrication technologies it is now possible to integrate thousands of elementary processors on a silicon chip, in a pixel-parallel image processor. Furthermore, it is now possible to integrate image sensing elements within the compute-memory fabric of the processor array on a single "vision chip" [5]–[7]. The co-location of photosensors and processors minimises the sensor-processor communications, providing additional benefits in terms of speed and power consumption of the system. We term such a device a Pixel Processor Array (PPA), where sensing, processing, and local memory are collocated on a processor-per-pixel basis. PPA vision sensors have been demonstrated, with resolutions up to 256×256 pixels [8]. The recent technological trends of 3D silicon wafer stacking provide a vehicle for vertically integrating sensor and processor layers, promising future high-resolution vision sensor devices, where computing power can be placed behind each pixel of the image sensor [9]–[11].

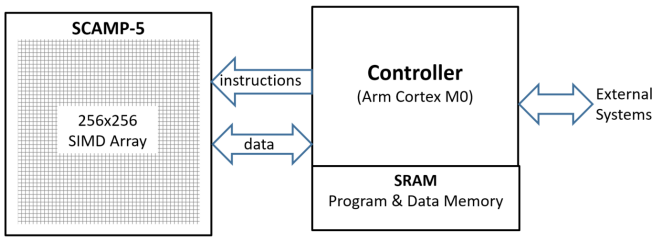

Fig. 1. Overview of the SCAMP vision system. The control program is executed on the ARM M0 core, which instructs the SCAMP-5 massively-parallel SIMD processor array to carry out operations on image arrays. SCAMP-5 has 256x256 Processing Elements.

The key advantage of PPA systems is that all low-level image processing occurs on the vision sensor integrated circuit, with no images transmitted off-chip in normal operation. Instead, only results of computations, for instance extracted features [12], classification results [13], or visual odometry information [14], are read-out directly from the device.

In this paper we illustrate how digital global summation can be efficiently implemented on a pixel-parallel device, demonstrated upon the SCAMP-5 vision sensor [8]. The architecture of the chip is briefly presented in the next section.

II. SCAMP-5 Architecture

The overall architecture of the SCAMP-5 system used in this work is illustrated in Figure 1. The SCAMP-5 chip comprises a 256×256 array of Processing Elements (PEs) which receive instructions from a single Controller (Arm Cortex-M0). The controller has its own program and data memory, and is responsible for the overall program flow, and any sequential computing required in the algorithm. It also issues microinstructions to the SCAMP-5 array. All PEs in the array execute the same microinstruction, issued by the Controller, i.e. the array operates as a SIMD processor.

Although it is possible to transfer data from the Controller to the SCAMP-5 array, the primary input to the array is optical, via photosensors in each PE. The typical operation

978-1-6654-3948-0/21/$31.00 ©2021 IEEE

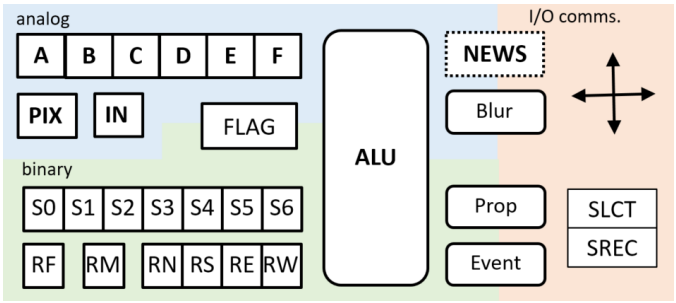

Fig. 2. The architecture of the SCAMP-5 Processing Element. A-F are analog registers, PIX is image sensor input, IN is a global input. S0-S6 are general-purpose binary registers. Rx are special-purpose registers. ALU executes transfers and arithmetic and logic operations, 'Blur' and 'Prop' are additional asynchronous hardware accelerators. FLAG is local activity register. NEWS provides 4-neighbour communications. SLCT and SREC provide array addressing and 'Event' unit enables sparse read-out.

is to acquire an image, and then process it in the SCAMP-5 array, according to the sequence of microinstructions sent by the Controller. The results of computations are read-out from the SCAMP-5 array by the Controller. Readout of entire data arrays is possible, but primarily for debugging.

The detail of the PE architecture is shown in Figure 2. Each PE contains six general-purpose "analog" registers that can store a gray-level pixel values, and thirteen binary registers. Several binary registers also have special-purpose designations. The ALU provides basic arithmetic and logic operations, for instance addition or subtraction of two analog registers, or logic AND operation between binary registers.

The analog NEWS register is used to transfer analog data between neighbouring PEs in the array. For instance, moving the content of each PEs A register, into that of its "south" neighbouring PE. Globally this results in the image held across the A registers of the array, being shifted one pixel to the South. Data transfer in binary registers is achieved using a multi-directional propagation operation. Each PE transfers content to its neighbours in the array, with the registers RN,RS,RE,RW determining which neighbours to transfer to.

The details of the SCAMP-5 implementation can be found in [8]. The datapath is implemented using mixed-signal circuits, in particular storage and arithmetic operations on registers A-F are using analog current-mode signal representation. This has some implications with respect to the precision and accuracy of arithmetic operations, and often requires special care be taken to ensure the inherent processing errors do not adversely affect the computation results.

The analog current-mode computations allow operations such as global summation (all elements of the array are effectively added in one clock cycle) but this has limited precision, and in many situations a more accurate global summation, or pixel-counting mechanism, is required as will be presented in Section III.

III. Sandcastle Summation

This section describes a method for global summation of digital binary images on SCAMP-5, outputting the number of set/white pixels in said image. Our approach is fast enough to be used in place of the SCAMP-5's global analog image summation in many SCAMP-5 applications, such as visual odometry [14] and neural network inference [13]. Doing so provides a more accurate summation free from the noise inherent in analog summation, and can thus significantly improve accuracy and performance for certain tasks.

A. Naive Approach

The SCAMP-5 has the ability to locate a set pixel within a binary image, outputting the location of the PE holding this set data. A basic approach to performing global summation of a binary image would involve iteratively locating and eliminating such set pixels from the image until none remain, counting the number of eliminations made. The computation time of such an approach scales linearly with the number of set pixels in the binary image. This becomes prohibitively expensive in most tasks where the image may contain thousands of set pixels. Our proposed approach stills make use of this functionality to locate set pixels, but is only used to readout a small set of pixels whose locations encode information about the total global summation.

Algorithm 1 Sand Castle Sum

```
S0    // Binary image to sum
RN, RS, RE, RW = False    // Clear transfer directions

// Stack set pixel data vertically using data transfers
for n = 1 to 256 do
    // Set PE transfer directions for current image
    RN = S0
    RS = NOT(S0)

    // Perform parallel data transfer:
    // DNEWS operation propagates data in the direction
    // determined by the state of RN, RS
    S0 = DNEWS(S0)
end for

RS = False    // Clear South transfer for all PEs
RN = True    // Set North transfer for all PEs
S1 = DNEWS(S0)
S0 = XOR(S1, S0)    // Eliminate all but stack tops

// Extract remaining set pixels from image
Events[256,2] = Scan_Events(R11,256)

// Compute total Sum
Total_Set_Pixels = 0
for n = 1 to 256 do
    // Add together Y locations (stack heights)
    Total_Set_Pixels += Events[n,1]
end for

return Total_Set_Pixels
```

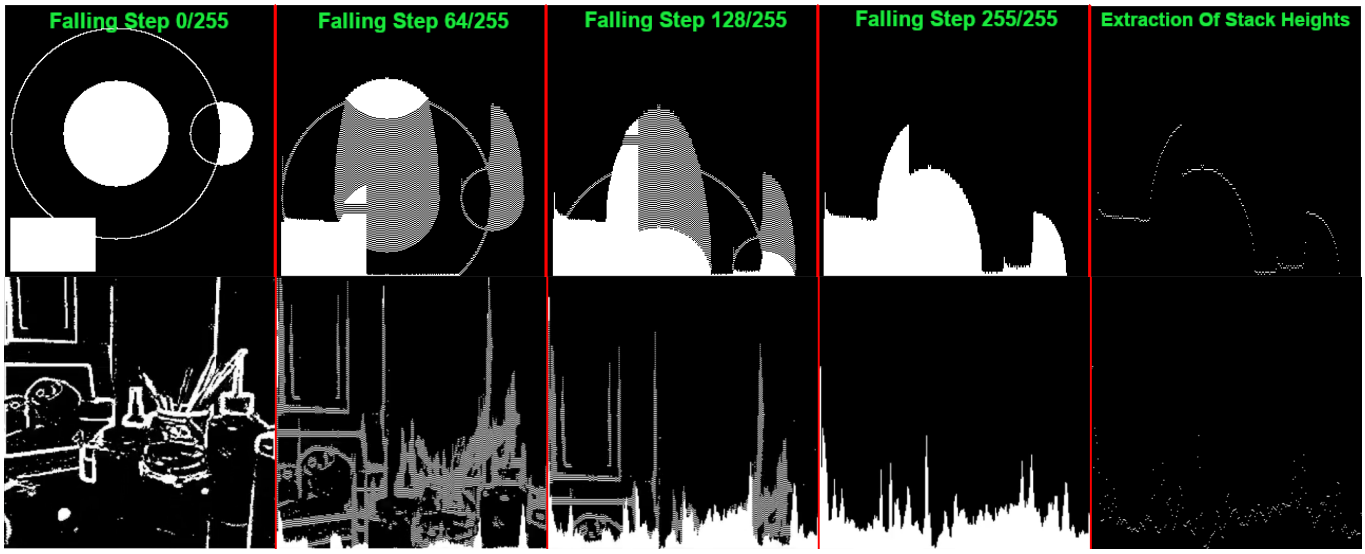

Fig. 3. Stages of global summation of a binary image using our proposed sandcastle summation approach both both a generated image (top row) and a real life scene (bottom row). Set pixels iteratively fall from their PEs into those below exploiting the parallel data transfer of SCAMP-5, forming into vertical stacks. Progress over an increasing amount of iteration is illustrated from left to right, with the rightmost image showing the elimination of set pixels but those at the top of each stack. These remaining set pixels can be extracted from the array to give the height of each stack which when summed give the original total number of set pixels.

B. Accelerated Approach

Our improved approach first transforms the binary image into a form more applicable to conducting global summation. This involves iteratively letting each set pixel ”fall” vertically from its current PE, into the PE below whenever that PE does not contain a set pixel. Similar to letting falling grains of sand form piles, this iterative process results in the ”falling” pixels stacking across the bottom of the PE array. This iterative process can be implemented efficiently using parallel data transfer operations, performing 255 iterations to ensure that all set pixels have come to rest within a vertical stack as is illustrated in Figure 3.

After these vertical stacks are formed the transformation is completed by performing an XOR operation between this stacked image and a vertically shifted copy of itself. This eliminates all set pixel but those at the top of each vertical stack as shown in Figure 3, leaving us 256 set pixels - one per column of PEs. The Y location of each of these remaining pixels then encodes the height of its associated stack. By iteratively locating and eliminating these remaining pixels (up to 256), the heights of each stack can be extracted from the array and added together, giving the total number of set pixels in the original binary image.

Essentially this approach can be viewed as conducting partial summations of the set pixels directly upon the PE array itself, thus leading to less information needing to be extracted from the array to determine the global summation. As the transfer of data off of the array is highly time consuming relative to most other operations, this new approach is easily over an order of magnitude faster than the naive approach of iteratively eliminating set pixels from the image.

Note that direction of digital data transfer operations upon SCAMP-5 can be chosen on a per PE basis, with each PE containing 4 digital registers, determining which of its 4 neighbouring PE to connect to during transfer. This control over the direction of data transfer allows us to efficiently implement a scheme whereby PEs containing set pixels copy the data from the PE below, and PEs with empty pixels copy from the PE above. This directional transfer setup causes set pixels to ”fall” into any empty PE below their current location whenever a digital data transfer operation is performed, and when performed repeatedly forms the vertical stacks of set pixels required for our approach. This approach listed Algorithm 1.

The computation time to perform this proposed method of global summation is constant, coming in at $331\mu s$, whereas the computation time of the naive approach described in Section III-A increases with the number of set pixels in the image. For a typical binary edge image our proposed approach can be well over two orders of magnitude faster. As an example for the image shown in 3 such a naive approach performing sequential pixel eliminations takes over $43000\mu s$.

IV. Conclusions

This paper presented a novel algorithm for global summation of a binary image upon PPA devices. Our approach is around two orders of magnitude faster in practice than a naive approach, fast enough to be used as a standard function for many real-time vision application. While exploiting the parallel processing of the SCAMP-5, this work should applicable to PPA architectures in general and should aid others in building their own applications for pixel-parallel architectures.

REFERENCES

[1] M. J. Duff *et al.*, "Review of the CLIP image processing system," in *Proc. National Computer Conference*. AFIPS Press Arlington, Va, 1978, pp. 1055–1060.

[2] J. C. Gealow, F. P. Herrmann, L. T. Hsu, and C. G. Sodini, "System design for pixel-parallel image processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 4, no. 1, pp. 32–41, 1996.

[3] M. Ishikawa, K. Ogawa, T. Komuro, and I. Ishii, "A cmos vision chip with simd processing element array for 1ms image processing, 1999 dig. tech. papers of 1999 ieee int," in *Solid-State Circuits Conf.(ISSCC99)(San Francisco, 1999.2. 16)/Abst*, pp. 206–207.

[4] P. Dudek and P. J. Hicks, "A general-purpose cmos vision chip with a processor-per-pixel simd array," in *Proceedings of the 27th European Solid-State Circuits Conference*. IEEE, 2001, pp. 213–216.

[5] J. Poikonen, M. Laiho, and A. Paasio, "MIPA4k: A 64× 64 cell mixed-mode image processor array," in *2009 IEEE International Symposium on Circuits and Systems*. IEEE, 2009, pp. 1927–1930.

[6] A. Lopich and P. Dudek, "A general-purpose vision processor with 160x80 pixel-parallel SIMD processor array," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 2017.

[7] A. Rodriguez-Vazquez, J. Fernández-Berni, J. A. Leñero-Bardallo, I. Vornicu, and R. Carmona-Galán, "CMOS vision sensors: embedding computer vision at imaging front-ends," *IEEE Circuits and Systems Magazine*, vol. 18, no. 2, pp. 90–107, 2018.

[8] S. J. Carey, A. Lopich, D. R. Barr, B. Wang, and P. Dudek, "A 100,000 fps vision sensor with embedded 535GOPS/W 256× 256 SIMD processor array," in *2013 Symposium on VLSI Circuits*. IEEE, 2013, pp. C182–C183.

[9] T. Yamazaki, H. Katayama, S. Uehara, A. Nose, M. Kobayashi, S. Shida, M. Odahara, K. Takamiya, Y. Hisamatsu, S. Matsumoto *et al.*, "A 1ms high-speed vision chip with 3d-stacked 140 GOPS column-parallel PEs for spatio-temporal image processing," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2017, pp. 82–83.

[10] L. Millet, S. Chevobbe, C. Andriamisaina, L. Benaissa, E. Deschaseaux, E. Beigne, K. B. Chehida, M. Lepecq, M. Darouich, F. Guellec *et al.*, "A 5500-frames/s 85-GOPS/W 3-d stacked BSI vision chip based on parallel in-focal-plane acquisition and processing," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1096–1105, 2019.

[11] T. Finateu, A. Niwa, D. Matolin, K. Tsuchimoto, A. Mascheroni, E. Reynaud, P. Mostafalu, F. Brady, L. Chotard, F. LeGoff *et al.*, "A 1280× 720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86 μm pixels, 1.066 GEPS readout, programmable event-rate controller and compressive data-formatting pipeline," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 112–114.

[12] J. Chen, S. J. Carey, and P. Dudek, "Feature extraction using a portable vision system," 2017.

[13] L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, "A camera that cnns: Towards embedded neural networks on pixel processor arrays," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[14] ——, "Visual odometry for pixel processor arrays," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4604–4612.