2.2 A 978GOPS/W Flexible Streaming Processor for Real-Time Image Processing Applications in 22nm FDSOI

Sander Smets¹, Toon Goedemé¹, Anurag Mittal², Marian Verhelst¹

¹KU Leuven, Heverlee, Belgium, ²San Jose, CA

The recent trend towards high frame rate, high resolution image processing necessitates a solution for the growing Von Neumann bottleneck. Bandwidth pressure is relieved in traditional image processors by tiling the image and storing pixel patches in a local scratchpad memory [1]. Yet, this only partially alleviates the memory bottleneck, due to repeated fetches for overlapping patches and the scratchpad access becoming the main bottleneck (Fig. 2.2.1). Streaming architectures [2] eliminate this overhead by streaming the image pixels into an array of processing elements and transferring intermediate data results to a next processing element instead of sending them back to memory. Crucial in the efficiency of such streaming image processing architectures is the line buffering strategy, required to merge intermediate results in the stream with previously computed results. Existing state-of-the-art architectures either have dedicated, application-specific line buffer instances which are inflexible towards various image processing workloads [1], or achieve flexibility by routing streams back to centralized scratchpad memories, again resulting in a memory bottleneck around the centralized buffers, and introducing the need for inefficient on-chip-networks between the processing elements [3]. This paper introduces the concept of flexible, instruction-programmable streaming processing with embedded configurable-delay FIFOs to enable line buffering for a wide range of computer vision algorithms on a single platform. Providing a deep reconfigurable pipeline, this design facilitates the mapping and execution of complex vision tasks, like dense optic flow in real-time (30fps VGA) at low power (10.7mW), marking a 5.8x improvement over the state-of-the-art.

The architecture (Fig. 2.2.2) consists of 330 single-cycle processing elements (PEs) and 90 configurable delay FIFOs, which are each flexible in the instruction they execute, and in the way they are concatenated, forming a deeply pipelined processing fabric handling the so-called pixel streams. Every PE or FIFO processes a new input every clock cycle and passes its result to the next PE or FIFO in the pipeline. The configuration of the units and the connections between them determines how pixels travel through the pipeline and how they are processed.

The PE, FIFO and crossbar configurations are provided in an instruction stream preceding the pixel stream. During the programming phase, indicated by a dedicated program signal, all PEs revert to a pass-and-store mode during which they pass the instruction stream to their successive PEs. The instruction stream has the same format as the data stream, allowing the signal interconnection fabric to be re-used, resulting in very limited configurability overhead. To ensure the instruction stream passes by all instruction registers, the program signal configures the crossbars in a default pattern that passes by every single PE. When the program signal goes low, the PEs store their current instruction in an instruction register.

Every individual PE consists of a 16b or 32b fixed-point ALU, an instruction register (IR) and a guard control unit (GC). The ALU selects its inputs from 5 input channels, which consist of 2 outputs of the previous PE, 2 inputs from the beginning of the PE unit chain (e.g. for data reuse within a PE chain), and a constant value locally stored in the PE (e.g. for a filter weight). The operation performed by the ALU and settings of internal multiplexers are determined by the PE's local instruction register, which remains frozen for the complete processing stream. This eliminates all activity overhead in instruction fetching and decoding. Finally, the guard control unit enables on-line control flow decisions by guarding operation outputs based on 4b flags travelling along with the pixels, enabling conditional instruction execution. The PEs are clustered in groups of 4 pipelined instances which connect to a hierarchical configurable interconnect network, enabling concatenation of PEs into longer pipelines and various configurations. Overhead of unused PEs is minimized through clock gating and bypassing.

The distributed memories for line buffering total 690KB of SRAM, spread over 90 configurable-delay FIFOs. Each of these buffers is clustered with one PE, hence extending the PE instruction set with the capability to specify a configurable delay. This clustering is beneficial, as several computer vision kernels only require a single operation between successive buffers. Each of the FIFOs can delay a pixel

stream up to 1024 cycles, buffering up to one line of HD frames without chaining FIFOs together. Unused SRAM blocks are placed in sleep mode for the full execution duration to minimize power consumption.

Crucial to streaming fabrics is their ability to map and concatenate a wide range of image processing compute kernels found in typical image processing pipelines, as listed in Fig. 2.2.1. Fig. 2.2.3 demonstrates how 3 of these kernels can be mapped on the presented fabric: horizontal and vertical Sobel filters and integral image construction. The extension from the first 2 kernels to larger or 2D filters is straightforward. For the third kernel, integral image construction, the first PE computes the running sum over a single line in the image and the PE with FIFO sums the current running sum with the result of the previous line. The reset of the values of the running sums at the start of a new line is implemented using a conditional operation using the guard control unit. Many of such basic image processing kernels can be concatenated in one long streaming pipeline across different interconnected PE units. We illustrate this for an optic flow application, using pyramidal Lucas-Kanade (LK) dense optical flow consisting of many concatenated image kernels (Fig. 2.2.4). The complete multi-scale algorithm is split into 7 passes over the streaming fabric (3x subsampling, 4x single-scale LK evaluation, warping performed off-chip). During subsampling, an 11-tap anti-aliasing Sobel filter is applied on both input images, occupying 52 out of 330 PEs and 20 out of 90 FIFOs. LK evaluation occupies 219 out of 330 PEs and 39 out of 90 FIFOs, accounting for 150 effective operations per clock cycle, with only 39 SRAM accesses. In between passes, the fabric is reconfigured with the next instruction stream.

The design was implemented in a 22nm FDSOI technology with an active area of 4.9mm² and runs at 200MHz at 0.72V and room temperature. Fig. 2.2.7 provides chip details and a die micrograph. Fig. 2.2.4 shows measurement results of the chip while running the LK dense optic flow at a 30MHz clock, which suffices for real-time VGA processing. Note that reconfiguring the architecture and clearing the pipeline only blocks execution around 3% of the time and only accounts for 2% of the overall energy consumption.

Figure 2.2.5 shows how throughput and energy efficiency can be traded off by modulating the supply and body bias voltages for full utilization (all PEs chained in a 330-tap filter) and during LK evaluation. The minimal energy point, at 36MHz, a supply of 480mV and a bias voltage of -150mV, achieves a peak energy efficiency of 978GOPS/W. Towards higher resolutions and/or frame rates, the body bias allows tuning the dynamic vs. leakage power ratio. At 220MHz (30fps, Full HD LK flow) a 450mV bias voltage enables a 7% power reduction with respect to zero-bias.

Figure 2.2.6 compares the presented design to recent streaming architectures and optic flow implementations. The demonstrated peak efficiency of 978GOPS/W at 36MHz remains within a factor 3 compared to non-programmable solutions. As a result of the configurable line buffering scheme, the architecture scales to high resolutions at a low energy cost. In terms of energy efficiency, this work marks a 5.8x improvement for optic flow, enabling its usage for autonomous navigation on energy-constrained platforms.

Acknowledgements:

The authors would like to thank Research Foundation Flanders (FWO) for supporting the work of S. Smets by a Doctoral Fellowship, and GlobalFoundries for chip fabrication support.

References:

- [1] Y. Sasagawa and A. Mori, "High-level Video Analytics PC Subsystem Using SoC With Heterogeneous Multicore Architecture," *IEEE JSSC*, vol. 51, no. 4, pp. 1051-1059, 2016.
- [2] F. Barranco, et al., "Parallel Architecture for Hierarchical Optical Flow Estimation Based on FPGA," *IEEE TVLSI*, vol. 20, no. 6, pp. 1058-1067, 2012.
- [3] X. Fan, et al., "Stream Processing Dual-Track CGRA for Object Inference," *IEEE TVLSI*, vol. 26, no. 6, pp. 1098-1111, 2018.
- [4] Z. Li, et al., "A 1920 x 1080 25fps, 2.4TOPS/W Unified Optical Flow and Depth 6D Vision Processor for Energy-Efficient, Low Power Autonomous Navigation," *IEEE Symp. VLSI Circuits*, pp. 135-136, 2018.

Figure 2.2.1: Overview of patch-based computer vision kernels and their applications.

Figure 2.2.2: Top-level architecture.

Figure 2.2.3: Mapping of typical computer vision kernels on the streaming fabric.

Figure 2.2.4: Measured power consumption for Lucas-Kanade dense optical flow.

Figure 2.2.5: Body bias tuning and measured efficiency.

	TVLSI '12 [2]	JSSC '16 [1]	TVLSI '18 [3]	VLSI '18 [4]	This work
Technology	90nm FPGA	28nm CMOS	55nm CMOS	28nm CMOS	22nm FDSOI
Application	LK flow	Sparse LK flow	AlexNet	NG-ISGM flow	LK flow
Processing approach	Streaming (distributed mem, fixed buffer sizes)	Patch-based	Streaming (central memory)	Patch-based	Streaming (distributed mem, config buffer sizes)
Programmability	Fixed datapath	SoC with fixed datapath for flow accelerator	Programmable CGRA	Fixed datapath	Programmable CGRA
Image size and throughput	640x480, 32 fps	1920x1080, 30 fps	227x227, 35 fps	640x480, 30 fps 1920x1080, 25 fps	640x480, 30 fps 1920x1080, 30 fps
On-chip memory	252 KB	2 MB	108 kB	196 KB	690 KB
Chip area	N/A	143.95 mm ²	5.19mm ²	9.3mm ²	4.9mm ²
Peak performance	N/A	3.5 TOPS	77.4 GOPS	1.82 TOPS	145 GOPS
Efficiency	N/A	1871 GOPS/W	50.78 GOPS/W	2.4 TOPS/W	978 GOPS/W
Power consumption	N/A	1878 mW	841 mW	62 mW @ 30 fps VGA 760 mW @ 25 fps FHD	10.7 mW @ 30 fps VGA 101.4 mW @ 30 fps FHD
Normalized energy	N/A	56.8 nJ/pix	N/A	6.73 nJ/pix @ 30 fps VGA 14.66 nJ/pix @ 25 fps FHD	1.16 nJ/pix @ 30 fps VGA 1.63 nJ/pix @ 30 fps FHD

Figure 2.2.6: Performance comparison.

Figure 2.2.7: Annotated chip photograph and chip summary.