An FPGA-based Hardware Accelerator for Simulating Spatiotemporal Neurons

Ghaith Tarawneh and Jenny Read
Institute of Neuroscience, Newcastle University
Framlington Place, Newcastle upon Tyne NE2 4HH, UK
Email: {ghaith.tarawneh, jenny.read}@ncl.ac.uk

***Abstract*—Simulating spatiotemporal neurons is fundamental to understanding motion detection mechanisms in the primary visual cortex and cloning these mechanisms in digital systems. We present a hardware accelerator that leverages the parallelism of a modern Field Programmable Gate Array (FPGA) to increase the speed of spatiotemporal computations by 1∼2 orders of magnitude for video framebuffer sizes up to 128×128×25 pixels. The accelerator is primarily intended for running simulations of large spatiotemporal neuron populations but can also be used in computer vision applications that require high-speed spatiotemporal processing such as realtime motion detection.**

***Index Terms*—Hardware acceleration, Gabor filter, spatiotemporal neuron, motion detection**

I. Introduction

Bio-inspired primitive visual processing units such as Reichardt detectors [1] and Gabor Filters [2] have demonstrated remarkable efficiency and robustness in a wide range of computer vision applications. In both biological and silicon systems, the outputs of these smaller units are combined in a hierarchical order to create complex filters that extract high-level visual information (e.g. global motion) irrespective of other features of the visual input such as phase or spatial scale. Such filters are realized in very dense arrays in the mammalian primary visual cortex [3] and are computationally intensive to simulate/compute by digital systems. This poses a challenge to neuroscientists trying to understand computational mechanisms in the visual cortex and computer vision system designers hoping to replicate these mechanisms in silicon.

The computational gap is particularly large in the case of spatiotemporal Gabor filters. These filters combine visual data over space and time and are biologically realized by spatiotemporal neurons whose regions of sensory inputs (receptive fields) extend in the visual x-y-t space. Simulations of these filters and their neuronal underpinnings are used by studies of local motion integration [4], complex cells in the visual cortex [5] and first and second-order motion detection systems [6]. In computer vision, spatiotemporal Gabor filters are used in motion detection [7], optical flow sensing [8], object recognition [9] and have been successfully applied in video quality assessment [10] and facial expression recognition [11].

Computing the output of a spatiotemporal filter requires cross-correlating two three-dimensional sets of the data (the filter's impulse response and input visual data in x-y-t) and is computationally intensive. This sets an upper bound on the number of filters that can be simulated in a given duration or computed in realtime. Fortunately, visual systems are inherently parallel and part of the gap between biology and silicon can be bridged by hardware acceleration. This potential is demonstrated by several VLSI implementations of Reichardt detector arrays that were modeled after insect vision systems [12] [13] [14]. FPGAs have recently become a popular medium for these implementations due to their relatively short development cycles and growing densities which are now enabling the integration of a sufficient number of primitive visual units for many real-time applications.

This paper presents an FPGA-based hardware accelerator for simulating spatiotemporal neurons. The accelerator is primarily intended for running simulations of large populations of spatiotemporal neurons but can also be used as a generic spatiotemporal processing unit in applications such as motion detection. Existing work has presented similar acceleration techniques for 2D Gabor filters with emphasis on generating filter coefficients efficiently [15], optimizing filter window sweeping across spatial dimensions [16] and exploring the algorithmic and numerical representation options for applying Gabor filters [17]. The presented architecture targets 3D Gabor filters specifically and proposes an efficient pipeline to compute the response of 3D Gabor-modeled neurons given a temporal stream of 2D images as an input. The accelerator delivers 1∼2 order of magnitude speedup compared to a software implementation running on a high-end desktop machine.

The contributions of the paper are as follows. First, we present an architectural solution with an efficient pipeline for simulating spatiotemporal neurons on modern FPGAs. Second, we explore design options to maximize the utilization of dedicated memory elements and multiplier circuits when computing 3D Gabor filter responses in hardware. Third, we evaluate the performance of the accelerator by running simulations of neuron-based motion detectors.

The remainder of the paper is organized as follows. Section II provides essential background on Gabor filters and describes how several 3D Gabor filters can be used to create a motion detector. Section III presents the accelerator and discusses its organization. Section IV presents resource utilization and speedup figures obtained by running simulations of neuron-based motion detectors on both the accelerator and a high-end desktop computer. Section V concludes with some remarks.

978-1-4799-4242-8/14/$31.00

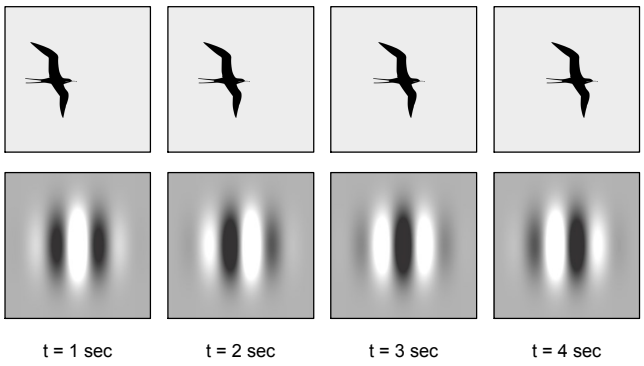

Fig. 1: Motion detection by cross-correlating a 3D spatiotemporal pattern (top) with a matching 3D filter (bottom)

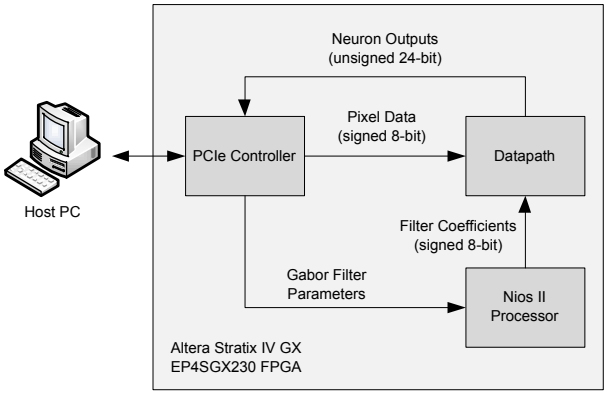

Fig. 2: System-level Diagram

II. Background

A 2D Gabor filter $g(x, y)$ is the product of a Gaussian envelope (of a standard deviation σ and an aspect ratio γ) and a sinusoidal wave (of a frequency f and phase ϕ):

$$g(x,y) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \times \cos(2\pi f x' + \phi) \quad (1)$$

where x' and y' are orthogonal coordinates specified by the filter's orientation θ:

$$\begin{aligned} x' &= x\cos(\theta) + y\sin(\theta) \\ y' &= -x\sin(\theta) + y\cos(\theta) \end{aligned}$$

Conventionally, two out-of-phase Gabor filters (a quadrature pair) are combined to create a phase-insensitive filter and Equation 1 is referred to as the real part of the filter.

Convolving a 2D Gabor filter with an image returns a measure of the degree of matching between the local features of the image and the spatial frequency and orientation of the filter. Motion detection filters can be created by generalizing this principle to three dimensions. An object moving in two dimensions (x-y) can be represented as a three-dimensional spatiotemporal pattern $h(x, y, t)$ and detected by cross-correlating this pattern with a matching three-dimensional filter $g(x, y, t)$. This is illustrated in Figure 1. The spatiotemporal filter can be created by encoding motion velocity v as a change in the phase of a conventional 2D Gabor filter as follows:

$$g(x,y,t) = \exp(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}) \times \cos(2\pi(f x' + vt) + \phi) \quad (2)$$

Convolving $g(x, y, t)$ and $h(x, y, t)$ returns a measure of the degree of matching between g and the local features of h. Specifically, correlation is highest for features of spatial frequency f, orientation ϕ moving at a speed v. Studies of the mammalian visual cortex has uncovered visual processing mechanisms that are modeled accurately by such filters [18]. Simulations of these mechanisms involve computing spatiotemporal neuron outputs w_i, each modeled as a dot product between a spatiotemporal receptive field response $g_i(x, y, t)$ and the animal's visual input $h(x, y, t)$:

$$w_i = \sum_X \sum_Y \sum_T g_i(x,y,t) \cdot h(x,y,t) \quad (3)$$

To simulate the transient output of a spatiotemporal neuron for a given input video, the neuron receptive field is first computed and stored in a 3D array. Video frames are then loaded sequentially into a pipeline of 2D arrays (a 3D framebuffer). For each loaded frame, the framebuffer content are dot multiplied with the neuron's receptive field coefficients to obtain the neuron output. Simulations of neuron populations require instantiating multiple 3D receptive field arrays to compute individual neuron responses.

III. Proposed Architecture

Multi-dimensional dot products can be serialized and performed efficiently by a pipelined array of multiplier circuits. We have implemented a hardware accelerator for spatiotemporal computations on a Terasic DE4 development board (with an Altera Stratix IV GX EP4SGX230 FPGA device) to exploit this property. The accelerator receives video frames and computes the response of several neurons in parallel. Neuron parameters are specified by a host computer prior to the start of each simulation and are used by an embedded soft processor to generate receptive field coefficients. Data is exchanged between the accelerator and host computer using the PCIe bus. The system-level schematic of the accelerator is shown in Figure 2. Below we describe its organization.

A. Generating Receptive Field Coefficients

Although a hardware implementation of Equation 2 can be used to generate receptive field coefficients, we have opted for a software implementation that is executed by an embedded soft processor (Altera Nios II). Computing the coefficients in software is slow but is performed only once at the beginning of

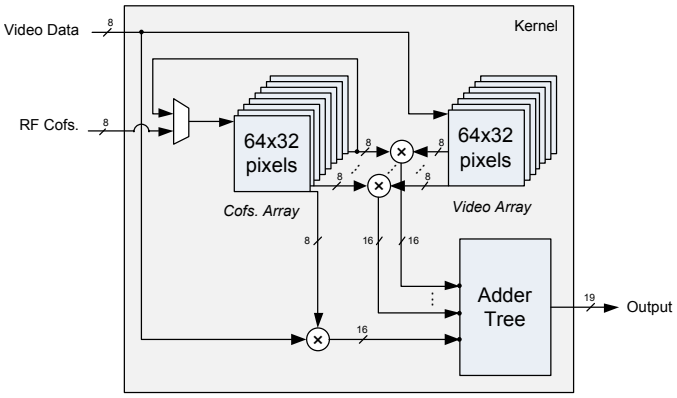

Fig. 3: Datapath Kernel

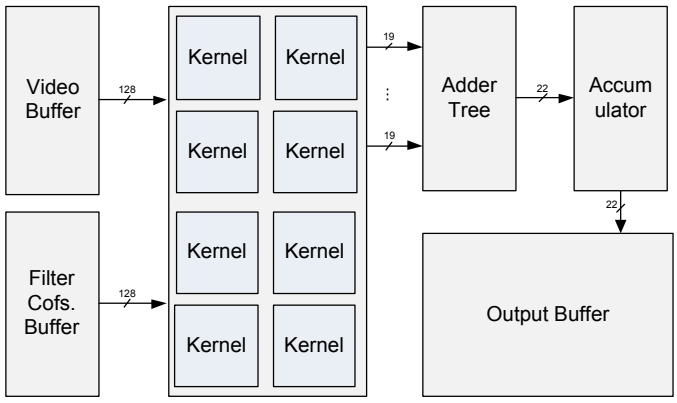

Fig. 4: Datapath Block Diagram

each simulation so its impact on simulation time is negligible (assuming a sufficiently-large number of frames per simulation). It also frees dedicated arithmetic resources for use in the accelerator datapath and offers a greater degree of filter design flexibility that is needed for simulating models with Gabor receptive field variants [3].

B. Datapath

The core accelerator datapath consists of a number of kernel units, each computing the dot product of 64×32×8 signed 8-bit pixel data and a corresponding coefficient array per neuron. Arrays are serialized and stored in dedicated memory blocks which have been configured as shift registers[1] (the mapping between x-y and linear coordinates is performed in a consistent fashion to preserve the position correspondence between video pixels and receptive field coefficients). Element-wise multiplication is performed in parallel at 8 taps in the shift registers, placed at 2048 element intervals, corresponding to the last position of each x-y slice. The 16-bit products are summed by a pipelined adder tree per neuron.

After pushing coefficients into the coefficient shift register, the latter is reconfigured to operate as 8 independent circular shift registers (each representing a 64×32 pixel x-y slice). As pixel data are pushed into the video framebuffer and multiplied with their respective coefficients, the coefficients are "recycled" and eventually restored to their initial positions following the insertion of each complete frame (2048 pixels).

The dimensions of kernel 3D arrays are chosen to maximize utilization of the limited number of dedicated multipliers and memory blocks available to the datapath. In our implementation, the datapath is allocated 512 multipliers and 1 MB of memory giving a 1/2048 multiplier-to-byte ratio so 3D arrays were split into segments of 2048 elements which have been arbitrarily arranged as 64×32 slices.

The datapath contains an array of 8 kernels, each computing the dot product of an x-y-t sub-region of the video input for all simulated neurons. After preloading filter coefficients, the input video is streamed from the host computer in 8-byte chunks and distributed across the kernels. Kernel outputs are summed by an adder tree, accumulated and then (following the insertion of each complete frame) stored in a memory buffer and reset. The content of the output buffer is periodically flushed by the PCIe controller and sent to the host computer.

[1]Dedicated memory blocks are featured in many modern FPGA architectures (M9K and M144K blocks in Altera Stratix IV devices) and offer a compact way to implement shift registers without exhausting the FPGA's interconnection fabric.

IV. Benchmark Results

To evaluate the performance of the accelerator we ran simulations of various-size neuron populations on both the accelerator and a desktop computer (Intel Core i7-3770 3.4 GHz) running Matlab. The simulated neurons had Gabor-modeled receptive fields with different parameters and function collectively as a motion detector. We have synthesized datapath variants with different arrangements to explore resource utilization and speedup figures. All synthesized datapaths were optimized to run at 200 MHz.

A. Resource Utilization

Receptive field dimensions and the number N of neurons that can be simulated in parallel are primarily constrained by the number of dedicated memory elements available to the datapath. In the used FPGA device, we were able to fit datapaths ranging from 1 neuron with a 128×128×25 element receptive field up to 256 neurons with 64×64×16 element receptive fields. Simulations of larger populations were split into batches of N-neuron simulations which required streaming video input to the accelerator several times.

B. Speedup

The performance of the accelerator (expressed in frames/second) is compared against that of the reference desktop computer in Table I. The speedup figures range from 30x to 219x depending on video frame dimensions (higher speedup figures obtained for smaller dimensions). The throughput drop at large video frame dimensions is due to the transfer time of pixel data from the host computer to the accelerator. In our implementation, pixel data was transferred at 200 MBps using the PCIe bus but transfer times nonetheless consumed about 88% of the roundabout processing time for

TABLE I: Accelerator Resource Utilization and Performance

Video Dimensions	Neurons	Kernels	M9K Blocks	Multipliers	PC Throughput (fps)	Accelerator Throughput (fps)	Speedup
128×128×25	1	8	784	200	378	11316	**29.9x**
128×128×16	2	8	736	256	295	11315	**38.3x**
128×128×8	8	8	1024	512	148	11312	**76.6x**
128×128×1	64	8	16	512	148	11278	**76.4x**
96×64×25	6	3	1014	450	168	25275	**150.5x**
96×64×16	10	3	996	480	157	25262	**160.4x**
96×64×8	21	3	930	504	150	25229	**168.2x**
94×64×1	170	3	6	510	148	24785	**167.2x**
64×64×25	9	2	964	450	168	33541	**199.7x**
64×64×16	16	2	1024	512	148	33504	**226.9x**
64×64×8	32	2	928	512	148	33418	**226.3x**
64×64×1	256	2	4	512	148	32266	**218.5x**

frames sizes of 128×128 pixels. Using a faster communication link between the accelerator and host computer can decrease simulation time significantly but this is constrained by the availability of high speed transceivers on the used FPGA device. The number of dedicated M9K memory blocks and multiplier circuits that could be allocated in each datapath variant is close to the design limits (1024 M9K blocks and 512 multipliers) which demonstrates the efficiency of internal kernel organization.

V. Conclusion

Spatiotemporal processing using 3D Gabor filters is a key building block of motion detection systems in both biology and silicon. We presented a hardware accelerator that increases the speed of spatiotemporal neuron simulations up to 2 orders of magnitude compared to a software implementation. The accelerator datapath contains a number of kernels which process individual x-y-t subregions of the video input and have been optimized to maximize the utilization of dedicated memory blocks and multiplier circuits. It is intended to run simulations of large neuron populations but can also be used in computationally intensive computer vision applications such as realtime motion detection.

References

[1] W. Reichardt, "Autocorrelation, a principle for the evaluation of sensory information by the central nervous system," *Sensory communication*, pp. 303–317, 1961.

[2] D. Gabor, "Theory of communication. part 1: The analysis of information," *Electrical Engineers-Part III: Radio and Communication Engineering, Journal of the Institution of*, vol. 93, no. 26, pp. 429–441, 1946.

[3] N. Petkov and E. Subramanian, "Motion detection, noise reduction, texture suppression, and contour enhancement by spatiotemporal gabor filters with surround inhibition," *Biological Cybernetics*, vol. 97, no. 5-6, pp. 423–439, 2007.

[4] N. J. Majaj, M. Carandini, and J. A. Movshon, "Motion integration by neurons in macaque mt is local, not global," *The Journal of neuroscience*, vol. 27, no. 2, pp. 366–370, 2007.

[5] S. Nishimoto and J. L. Gallant, "A three-dimensional spatiotemporal receptive field model explains responses of area mt neurons to naturalistic movies," *The Journal of Neuroscience*, vol. 31, no. 41, pp. 14 551–14 564, 2011.

[6] T. Nagano, M. Hirahara, and W. Urushihara, "A general model for visual motion detection," *Biological cybernetics*, vol. 91, no. 2, pp. 99–103, 2004.

[7] A. Spinéi, D. Pellerin, D. Fernandes, and J. Hérault, "Fast hardware implementation of gabor filter based motion estimation," *Integrated Computer-Aided Engineering*, vol. 7, no. 1, pp. 67–77, 2000.

[8] D. J. Heeger, "Optical flow using spatiotemporal filters," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 279–302, 1988.

[9] K. Takahashi, Y. Kuriya, and T. Morie, "Bicycle detection using pedaling movement by spatiotemporal gabor filtering," in *TENCON 2010-2010 IEEE Region 10 Conference*. IEEE, 2010, pp. 918–922.

[10] K. Seshadrinathan and A. C. Bovik, "Motion tuned spatio-temporal quality assessment of natural videos," *Image Processing, IEEE Transactions on*, vol. 19, no. 2, pp. 335–350, 2010.

[11] T. Wu, M. S. Bartlett, and J. R. Movellan, "Facial expression recognition using gabor motion energy filters," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE, 2010, pp. 42–47.

[12] F. Aubépart, M. El Farji, and N. Franceschini, "Fpga implementation of elementary motion detectors for the visual guidance of micro-air-vehicles," in *Industrial Electronics, 2004 IEEE International Symposium on*, vol. 1. IEEE, 2004, pp. 71–76.

[13] A. Borst, K. Kuhnlenz, and M. Buss, "An FPGA implementation of insect-inspired motion detector for high-speed vision systems," *2008 IEEE International Conference on Robotics and Automation*, pp. 335–340, May 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4543230

[14] T. Köhler, F. Röchter, J. P. Lindemann, and R. Möller, "Bio-inspired motion detection in an FPGA-based smart camera module." *Bioinspiration & biomimetics*, vol. 4, no. 1, p. 015008, Mar. 2009. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/19258686

[15] O. Y. Cheung, P. H. W. Leong, E. K. Tsang, and B. E. Shi, "A scalable fpga implementation of cellular neural networks for gabor-type filtering," in *Neural Networks, 2006. IJCNN'06. International Joint Conference on*. IEEE, 2006, pp. 15–20.

[16] Y. C. P. Cho, S. Bae, Y. Jin, K. M. Irick, and V. Narayanan, "Exploring gabor filter implementations for visual cortex modeling on fpga," in *Field Programmable Logic and Applications (FPL), 2011 International Conference on*. IEEE, 2011, pp. 311–316.

[17] N. Voß and B. Mertsching, "Design and implementation of an accelerated gabor filter bank using parallel hardware," in *Field-Programmable Logic and Applications*. Springer, 2001, pp. 451–460.

[18] D. L. Ringach, "Mapping receptive fields in primary visual cortex," *The Journal of Physiology*, vol. 558, no. 3, pp. 717–728, 2004.