Bio-Inspired Vision Processor for Ultra-Fast Object Categorization

Clément Farabet[1,2], Berin Martini[2], Polina Akselrod[2], Benoit Corda[1]
Selçuk Talay[2], Yann LeCun[1] and Eugenio Culurciello[2]
[1] The Courant Institute of Mathematical Sciences and Center for Neural Science, New York University, USA
[2] Electrical Engineering Department, Yale University, New Haven, USA

***Abstract*— We present a scalable hardware architecture to implement large-scale bio-inspired synthetic vision systems. The system is a fully digital implementation of a modular vision engine that can perform real-time detection, recognition and segmentation of mega-pixel images. We present performance comparisons between software versions of the vision system executing on CPU and GPU machines, and show that our FPGA implementation can outperform these systems by a factor of four.**

I. Introduction

Micro-robots, UAVs, imaging sensor networks, wireless phones, and other embedded vision systems all require low cost and high-speed implementations of synthetic vision systems capable of recognizing and categorizing objects in a scene, from faces [1] to objects [2], [3] or obstacles for robot navigation [4].

In this paper we present a scalable hardware architecture for large-scale multi-layered synthetic vision systems based on large parallel filter banks called *NeuFlow*. This vision processor (VP) is a data-flow engine that can perform real-time detection, recognition and localization in mega-pixel images processed as pipelined streams. The system was designed with the goal of providing categorization of an arbitrary number of objects, while consuming ten times less than a bench-top or laptop computer, on the order of 10W.

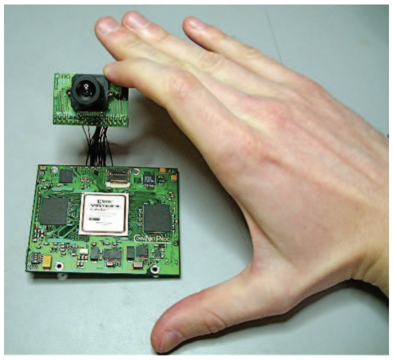

Fig. 1. A custom embedded vision system prototype composed of an input camera, an FPGA and two external QDR memory chips.

II. System Implementation

The architecture of our vision processor is designed to increase data throughput by using parallel vector processing units and allowing individual streams of data to operate seamlessly within processing blocks.

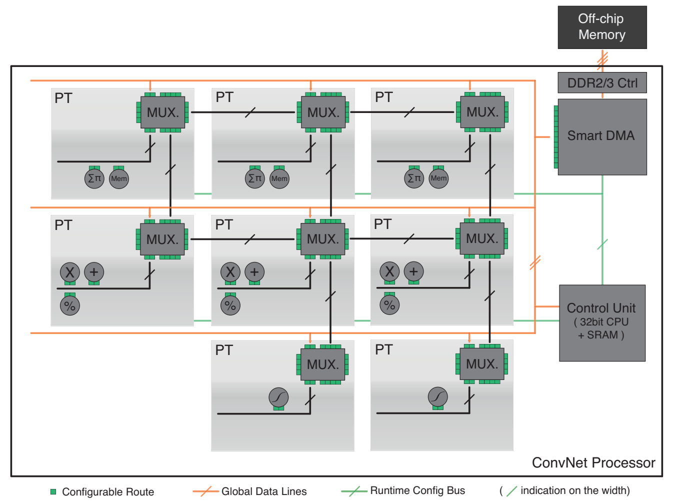

Fig. 2. Overview of the *NeuFlow* vision processor. A grid of multiple full-custom Processing Tiles tailored to massive convolutional operations, and a fast streaming memory interface (Smart DMA).

Recent FPGAs include a large number of multiply-accumulate (MAC) units, which allows fast prototyping and real-time simulation of vision system that heavily rely on these operation to process image flows. An example of our first vision system prototype using modern FPGAs is portrayed in Figure 1.

A schematic summary of the *NeuFlow* system is presented in Figure 2. The main components are: (1) a *Control Unit* (implemented on a general purpose CPU), (2) a grid of *Processing Tiles (PTs)*, and (3) a *Smart DMA* interfacing external memory via a standard controller. The Control Unit is implemented by a *general purpose 32bit CPU*, a more convenient solution than custom state machines, as it allows the use of standard C compilers. Moreover, the CPU has full access to the external memory (via global data lines), and it can use this large storage to store its program instructions.

The *PTs* are independent processing tiles laid out on a two-dimensional grid. The PTs contain a routing multiplexer (MUX) and local operators. This implementation is specialized for vision processing applications that rely heavily on two-dimensional convolutions. The components of the systems are:

- the top row PTs of Figure 2 only implement Multiply and Accumulate (MAC) arrays ($\sum\prod$ operators), which can be used as 2D convolvers (implemented in the FPGA by dedicated hardwired MACs). It can also perform on-the-fly subsampling (spatial pooling), and simple dot-products (linear classifiers) [5].

- the middle row PTs contain general purpose operators (for divisive normalization, squaring and dividing are required),
- the bottom row PTs implement non-linear mapping engines, used to compute all sorts of functions from $Tanh()$ to $Sqrt()$ or $Abs()$. Those can be used at all stages of the ConvNets, from normalization to non-linear activation units.

The operators in the PTs are fully pipelined to produce one result per clock cycle. Image pixels are stored in off-chip memory as Q8.8 (16bit, fixed-point), but scaled to 32bit integers within operators, to keep full precision between successive operations.

III. Results

Figures 3 and 4 report a performance comparison between a laptop CPU (Apple Macbook Pro with 2.4GHz Intel Core 2 Duo), two GPU systems (NVIDIA 9400M from a Macbook Pro laptop, and a NVIDIA Tesla machine C1060), and our system implemented in 2 different FPGAs: a Xilinx Virtex4 (custom board SX35 with two external QDR memory chipset, as illustrated in Figure 1) and a Xilinx Virtex6 (ML605)

To compare these systems, we executed a state-of-the-art convolutional neural network (ConvNet) synthetic vision system trained for obstacle avoidance, scene classification [4]. The network is composed of a non-linear normalization layer, 2 convolutional layers, 1 pooling layer, and a linear classifier. The convolutional layers and pooling layers are followed by non-linear activation units (hyperbolic tangent). Overall, it possesses 920 KxK learned kernels, 40 4x4 learned subsampling kernels, and N 200 dimension classification vectors. For a 500x500 input image and $K = 7$, the network has 435 Million linear connections (multiply and accumulate operations).

Figure 3 shows the frame per second versus input image size with a fixed 9x9 convolution filter ($K = 9$) for the whole ConvNet, and 5 output classes ($N = 5$). When the input image size varies, the network adapts the sizes of all its internal maps accordingly, producing an output map with a size linearly related to the input size.

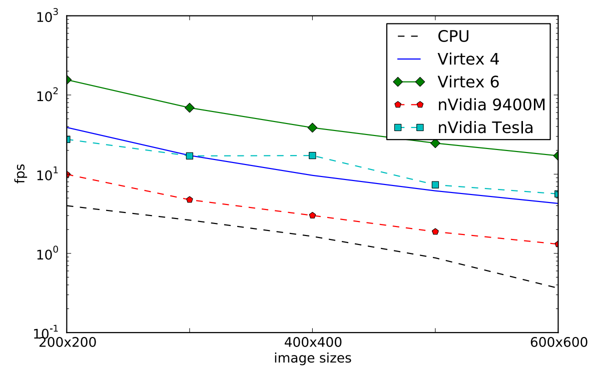

Fig. 3. Frames per second vs the size of input images and using convolutional filters of 9x9.

Figure 4 reports the frames per second vs convolution filter sizes, assuming the ConvNet uses the same filter size in all three layers, an input image of 500x500 pixels, and other parameters as mentioned above. When the kernel sizes vary, the internal maps sizes vary accordingly.

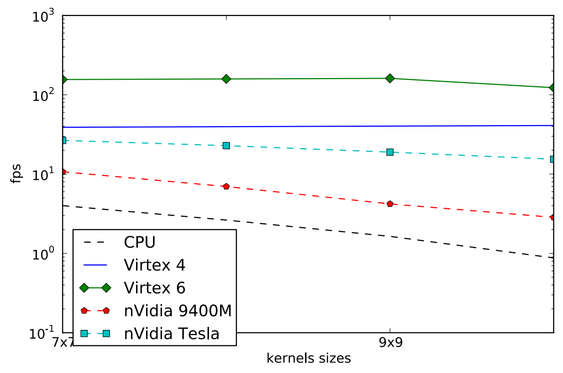

Fig. 4. Frames per second vs the size of convolution filters for a fixed image size of 500x500.

Figure 3 attests that our newest Virtex 6 FPGA system can run the ConvNet system in real time ($> 30fps$) with image sizes of 512x512 pixels, surpassing one of the most performing GPU in the market by 4 times. Figure 4 shows that performance is independent of the convolution kernel size, up to the number of MACs unit used in the system (100 in this case).

IV. Conclusion

We report the design of a hardware accelerated ConvNet system that is capable of running in real time with low power consumptions, while providing performance that is better than conventional laptop computers and advanced GPUs. Future work will include implementation in a high-performance ASIC system capable of delivering real-time operation on 1 mega-pixel images with 1W of power.

Acknowledgment

This work was partially supported by NSF grant ECCS-0901742, by ONR MURI BAA 09-019, DARPA NeoVision2 program BA 09-58.

References

[1] R. Osadchy, M. Miller, and Y. LeCun, “Synergistic face detection and pose estimation with energy-based model,” in *Advances in Neural Information Processing Systems (NIPS 2004)*. MIT Press, 2005.

[2] Y. LeCun, F.-J. Huang, and L. Bottou, “Learning methods for generic object recognition with invariance to pose and lighting,” in *Proceedings of CVPR'04*. IEEE Press, 2004.

[3] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun, “Learning invariant features through topographic filter maps,” in *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'09)*. IEEE, 2009.

[4] R. Hadsell, P. Sermanet, A. Erkan, J. Ben, J. Han, B. Flepp, U. Muller, and Y. LeCun, “On-line learning for offroad robots: Using spatial label propagation to learn long-range traversability,” in *Proc. Robotics Science and Systems 07*, 2007.

[5] C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun, “Cnp: An fpga-based processor for convolutional networks,” in *International Conference on Field Programmable Logic and Applications (FPL'09)*. Prague: IEEE, September 2009.