NeuFlow: Dataflow Vision Processing System-on-a-Chip

Phi-Hung Pham*, Darko Jelaca*, Clement Farabet†, Berin Martini*, Yann LeCun† and Eugenio Culurciello*
* Purdue University, West Lafayette, Indiana
† New York University, NY

Email: {phamp,euge}@purdue.edu

***Abstract*—This paper presents neuFlow system-on-a-chip - a neuromorphic vision system-on-a-chip implemented in the IBM 45 nm SOI process. The neuFlow processor was designed to accelerate neural networks and other complex vision algorithms based on large numbers of convolutions and matrix-to-matrix operations. Post-layout characterization shows that the system delivers up to 160 GOPS with an average power consumption of 570 mW. The power-efficiency and portability of this system is ideal for embedded vision-based devices, such as driver assistance, and robotic vision.**

I. Introduction

Driving a car is still a dangerous and time-consuming task. The average driver in the United States drives more than 10 thousand kilometers per year, with similar numbers for rich countries. This corresponds to 300 hours a year spent driving at speeds of 30-35 Km/h, and a monetary potential loss of productivity of 15,000$ per year. In addition driving a car is dangerous, because human drivers often get distracted while driving. The United States National Highway Traffic Safety Administration (NHTSA) counted 32 thousand severe accidents in 2010. 80% of car crashes are due to distractions, again according to the NHTSA. The design of technology that can lower these numbers is clearly a potentially transformative use of technology, and compares to the design of medical technologies in life-saving and quality-of-life potential improvements.

Electronics for car driver assistance is slowly becoming available for high-end cars [1]. These devices, generally referred as advanced driver assistance systems, can alert a driver of a dangerous condition, as an undesired lane-change, an undetected crossing pedestrian, traffic sign, etc. These devices are a great step forward to avoid distractions, but they still rely on human intervention and control of the vehicle. In this work, we discuss a neuromorphic vision system for car driving assistance. We focus here on vision-based systems because active laser system are not yet able to solve interfering problems, beside their high cost and large power requirements. We do not consider these to be viable commercial solutions to autonomous driving, even they have been successful in the recent Google autonomous car and DARPA competitions.

Being able to navigate a vehicle on the road requires very complex visual capabilities. Unfortunately, the unstructured nature of the road and the variability of its appearance in various environmental conditions and terrains makes the simple problem of detecting the road quite complex. What needed is a general-purpose vision system capable of identifying a large number of objects. This vision system needs to be able to:

- extract visual features from different objects (categorization/memory and segmentation),
- track changing object features (tracking), extract depth features (categorization of traversability of road), navigation, learning of object relationships (statistics of the visual world, object relationships, laws of the visual world),
- compress category information in a logarithmic manner, thus growing only slightly computational and architectural requirements for a linear increase in number of categories required.
- Also, the device must meet real-time requirement of frame process, flexible for vision task development, and power-efficiency for being embedded even "behind the car's rearview mirror".

These computations can only be done with a hierarchical (logarithmic) architecture with multiple stages of processing in series [2]. Simple feature-extraction stages are not enough to be able to scale to the requirements. Convolutional Neural Networks (ConvNets) implement a vision architecture that solves all these problems in a single elegant solution [3], and are the computational core of the neuFlow processor implemented in 45nm SOI process presented in this paper.

The rest of paper is organized as follows. Section II describes the architecture and operation of neuFlow system-on-a-chip (SoC). A compiler framework for quick development of vision tasks is also introduced. In Section III, a design and implementation results of neuFlow processor in IBM 45nm SOI process is reported. This section also gives a performance comparison of the chip with other conventional platforms implementing ConvNet-based visual models. Finally, section IV concludes the paper and outlines our further researches.

II. NeuFlow Architecture and Operation

ConvNets is a deep learning hierarchical artificial vision model which uses large number of convolutions to process image frames into a set of features and decisions. To implement ConvNets in real-time embedded systems, several architectural issues need to be considered:

- First, the system must be able to adapt to arbitrary *data-flow graphs*, which typically occurs in “systolic” computing systems [4] due to high-level parallelism.
- Second, the system must *quickly reconfigure* to meet the run-time change of data-flow graphs occurring during application execution under a profiling from high-level compiler. Hence, the need of a dynamically reconfiguration system [5] is mandatory.

The neuFlow [3] processor, its architecture and development tools, were developed with these considerations.

A. NeuFlow Architecture

Figure 1 reports the neuFlow architecture [6], which is designed to accelerate data stream ConvNets computation. The architecture has several key components: *Calculator*, *Streamer* and *Flow-cpu*.

The *Calculator* consists of a 2D grid of N_{PT} Processing Tiles (PTs). Each PT contains a bank of processing operators and a multiplexer (MUX) based on-chip router. This grid-based architecture interconnected by an on-chip network is considered as the architecture of interest due to several reasons. First, the coarse-grained property of PT enables a low configuration overhead compared to FPGA approach, meanwhile it has advantage of programming flexibility of general-purpose processors. Second, the use of on-chip network can flexibly adapt the system to accelerate arbitrary dataflow graphs formed during the application execution [7]. The bank of processing operators is highly optimized for ConvNets computation. A processing operator can be a term-by-term streaming operator (MUL, DIV, ADD, SUB, MAX), a MAC-based full 1/2D parallel convolver, a configurable bank of FIFOs for stream buffering, a configurable piece-wise linear or quadratic mapper. These operators are locally connected to each other, and/or to global data wires and neighbor tiles through an on-chip network of MUX-based routers. The on-chip network, once being configured, will form the configurable paths to support streaming dataflow graphs at runtime.

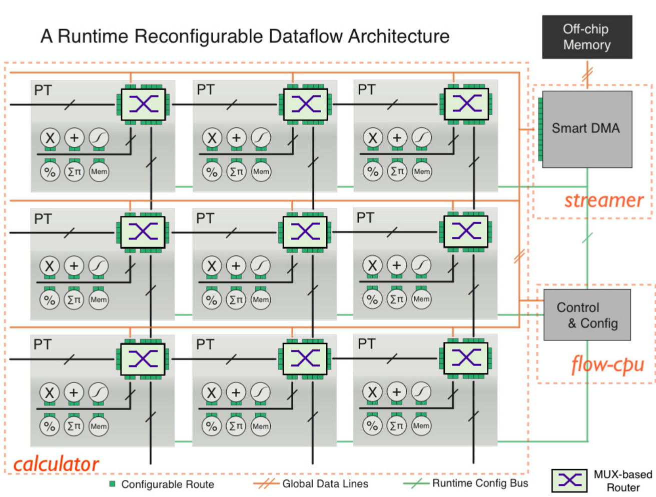

Fig. 1. The neuFlow architecture

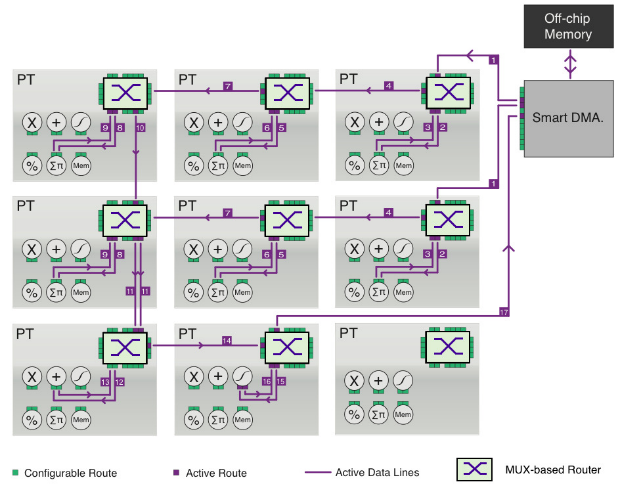

Fig. 2. An example of the grid configured for a dataflow computation: the 3 top tiles perform a 3×3 convolution, the 3 intermediate tiles another 3×3 convolution, the bottom left tile sums these two convolutions, and the bottom central tile applies a function to the result.

The *Streamer* functions as a Smart Direct Memory Access module (Smart DMA), which interfaces with off-chip memory and provides asynchronous data transfers with priority management. The *Smart DMA* module is customized to allow N_{DMA} ports to fully access the external memory. The DMA is considered to be ”smart”, because it can be configured to read or write a particular chunk of data, with an optional stride (for 2D streams), and feedbacks its status to *Flow-cpu*.

The *Flow-cpu* works as a central Control Unit that can reconfigure the computing grid and the Smart DMA at runtime. The configuration data from *Flow-cpu* placed on a Runtime Configuration Bus (re-)configures most aspects of the grid at runtime, including connections, operators and Smart DMA modes.

B. Operation

An execution on neuFlow typically has the following steps: (1) the Control Unit configures each tile to be used for the computation and each connection between the tiles and their neighbors and/or the global lines, by sending a configuration command to each of them, (2) it configures the Smart DMA to prefetch the data to be processed, and to be ready to write results back to off-chip memory, (3) when the DMA is ready, it triggers the streaming out, (4) each tile processes its respective incoming streaming data, and passes the results to another tile, or back to the Smart DMA, (5) the control unit is notified of the end of operations when the Smart DMA has completed.

The computing grid interconnected by the on-chip network can perform arbitrary computations on streams of data, from plain unary operations to complex nested operations. By a networking of MUX-based routers, operators can be easily cascaded and connected across tiles, independently managing their flow by the use of input/output FIFOs. As illustrated in

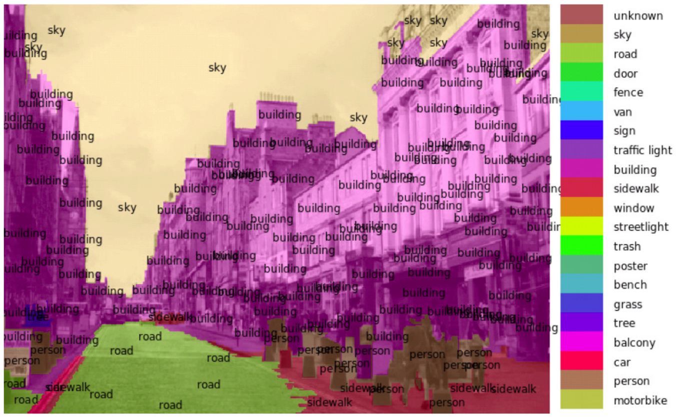

Fig. 3. A ConvNet implemented on neuFlow functionally prototyped in FPGA capable of performing real-time street scene parser.

Figure 2, the network-based grid is configured for dataflow computation of a sum of two convolutions followed by a non-linear activation function.

C. Compiler and Application Development

Prior to being executed on neuFlow, a given ConvNet algorithm has to be converted to a representation that can be interpreted by the Control Unit to generate controls/configurations for the system. For that purpose, a compiler and dataflow API—*luaFlow* [3]—are created. *LuaFlow* is a full-blown compiler that takes sequential, tree-like or flow-graph descriptions of algorithms in the *Torch* [8] environment, and parses them to extract different levels of parallelism. Pattern matching is used to map known sequences of operations to low-level, pre-optimized routines. Other unknown operators are mapped in less optimized ways. Once each high-level module has been associated with a set of low-level operations, a static sequence of grid configurations, interspersed with DMA transfers is produced, and dumped as binary code for the embedded Control Unit.

Street scene parser is an essential task for car driving assistance. This application aims at segmenting and recognizing the content of a scene: from objects to large structures. e.g., roads, sky, buildings, cars, etc (see Figure 3). In other words, the goal is to map each pixel from a given input image to a unique label. We use the ConvNet for street-scene parser presented in [6] for a 500×375 input image. Once trained, the network is passed over to *luaFlow*, and transparently mapped to neuFlow. A key advantage of convolutional networks is that they can be applied to sliding windows on a large image at very low cost by simply computing convolutions at each layer over the entire image . The output layer is replicated accordingly, producing one detection score for every 92×92 window on the input, spaced every 4 pixels. The neuFlow is functionally prototyped in FPGAs and predicted to produce one image in $83ms$ with an average error of 10^{-2} (quantization noise). For more details of street scene parsing application, please refer to [6].

III. Design and Implementation

A. Testchip Design

The NeuFlow SoC consists of one calculator configured with 4 convolvers with up to 10 x 10 kernels. A smart DMA is used for off-chip memory communication to QDR. The chip includes a programmable PLL, one Ethernet interface for *Flow-cpu*'s off-chip communication, GPIO and UART interfaces for debug/testing purposes. The neuFlow hardware is coded in Verilog HDL and prototyped in FPGA for functionality validation [3]. The design is synthesized and implemented in IBM 45 nm SOI STD-cell technology using a digital design flow based on Synopsys tools. The design flow features a Multi-Corner/Multi-Mode implementation with Multi-V_t STD-cells for a better design closure and power optimization. The chip uses area-array pads for packaging with flip-chip technology. The choice of flip-chip packaging has advantages over conventional wire-bond packages due to its small size, high performance, high pin-count and low cost per connections.

B. Implementation Results

The average power consumption of the implemented chip is estimated to be 570 mW under a system clock of 400 MHz and 1 V V_{core}.

Figure 4 reports the area utilization of the IBM neuFLow. The *Flow-cpu* occupies only around 9% of chip area. Meanwhile the *Streamer* and *Calculator* occupy 23% and 31% of the chip area, respectively. The rest 37% chip area is used for flip-chip I/O drives, decoupling capacitances and PLL.

Figure 5 shows the power breakdown estimated from post-layout tools with parasitic back annotation. The *Flow-cpu* and *Streamer* consume 7% and 27% of total chip power, respectively. Meanwhile, *Calculator*, which is the most computation-dominant component in the processor, consumes up to 46% of total chip power. The clock distribution network is estimated to consume around 21% of the chip power.

Table I summarizes the chip specifications. The neuFlow die area is 12.5 mm^2, and is shown in Figure 6.

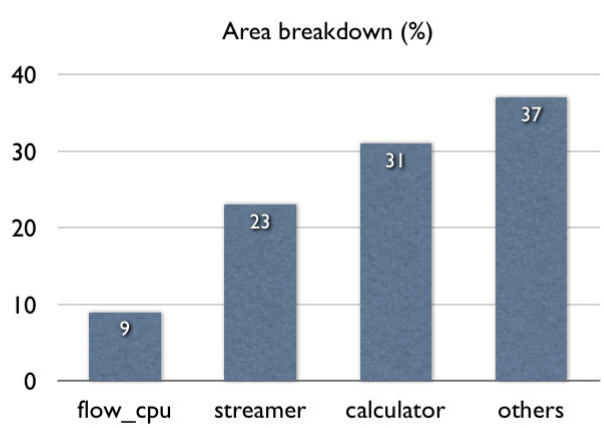

Fig. 4. Chip area breakdown

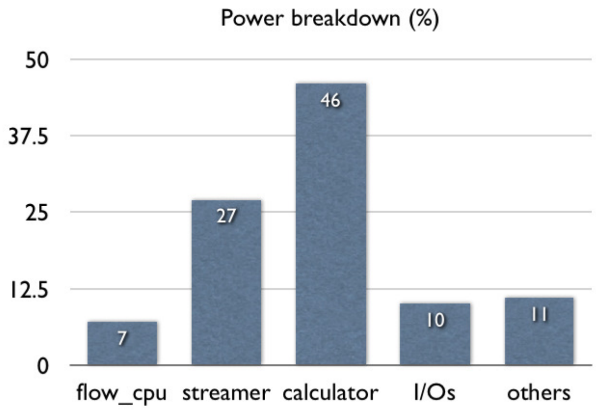

Fig. 5. Chip power breakdown estimation

TABLE I
POST-LAYOUT CHIP SUMMARY

Process	IBM SOI 45nm
Chip area	2.5 x 5 mm^2
Supply Voltages	1V V_{core}, 1.8V and 3.3V $V_{I/O}$
Target frequency	400MHz
Estimated average power	570mW
Peak performance	160 GOPS
GOPs per Watt	$\sim$ 254
Number of transistors, memories, etc.	23.6 million transistors and 75KB 2-port RAM
Pin count	317 (299 I/Os and 18 P/Gs)
Packaging	Flip-chip

C. Performance Comparison

Table II reports the performance comparison for a typical ConvNets computation implemented in various platforms. The CPU data is measured from compiled C code (GNU C compiler and Blas libraries) on a Core 2 Duo 2.66 GHz Apple Macbook PRO laptop operating at 90 W (30 W for the CPU). The mGPU and GPU data are obtained from a CUDA-based implementation running on a laptop/mobile nVidia GT335m operating at 1 GHz and 30 W and on a nVidia GTX480 operating at 1 GHz and 220 W. The FPGA performance was measured on a Xilinx Virtex-6 VLX240T operating at 200 MHz and 10 W [3]. The SoC characteristics are estimated from post-layout chip implemented in IBM 45 nm SOI process at a target frequency of 400 MHz.

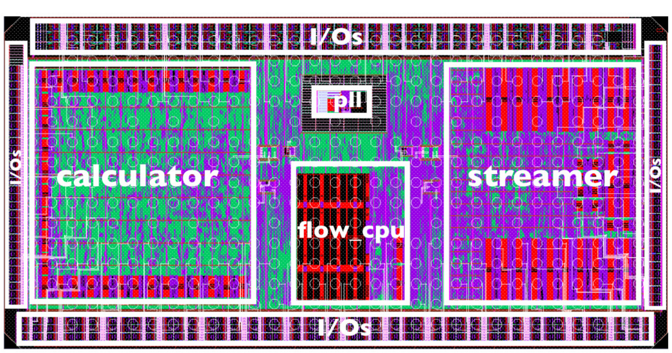

Fig. 6. Chip layout in a 2.5 x 5mm^2 die area

TABLE II
PERFORMANCE COMPARISON

	CPU[1]	**mGPU**[2]	**GPU**[3]	**neuV6**[4]	**neuIBM**[5]
Peak GOPs	10	182	1350	160	**160**
Real GOPs	1.1	54	294	147	**147**
Power (W)	30	30	220	10	**0.579**
GOPs/W	0.04	1.8	1.34	14.7	**254**

[1] CPU: Intel DuoCore, 2.7GHz, optimized C code
[2-3] mGPU, GPU: a mobile Nvidia GT335m and a high-end GTX480
[4] neuV6: neuFlow prototyped Xilinx Virtex 6 FPGA
[5] neuIBM: 45nm IBM SOI process neuFlow (*this work*)

As denoted in Table II, the neuFlow ASIC chip offers a peak performance of 160 GOPs, which satisfies the real-time computation requirement of many driving assistance vision tasks typically ranged from 60 to 120 GOPs [6]. Particularly, the chip power efficiency of 254 GOPS/W enables development of vision tasks in embedded systems.

IV. CONCLUSION

This paper presented the neuFlow system-on-a-chip architecture and its IBM 45 nm implementation. The neuFlow SoC is highly optimized for vision tasks in car navigation. Implementation result in IBM 45nm SOI process shows a high power efficiency of the chip, which can be easily to develop embedded applications for car driving assistance. Future works will extend the computation capacity of the systems and develop turnkey vision applications for driving assistance in a post-silicon prototype.

ACKNOWLEDGMENT

This work was partially supported by NSF award 0901742 and ONR award N000141110287. We would like to thank TAPO (www.tapoffice.org) for the chip tape-out.

REFERENCES

[1] Mobileye. [Online]. Available: http://www.mobileye.com
[2] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. International Conference on Computer Vision (ICCV'09)*. IEEE, 2009.
[3] Neuflow. [Online]. Available: http://www.neuflow.org
[4] M. H. Cho, C.-C. Cheng, M. Kinsy, G. E. Suh, and S. Devadas, "Diastolic arrays: throughput-driven reconfigurable computing," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2008, pp. 457–464.
[5] M. Platzner, J. Teich, and N. Wehn, *Dynamically Reconfigurable Systems: Architectures, Design Methods and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2010.
[6] C. Farabet, B. Martini, B. Corda, P. Akselrod, E. Culurciello, and Y. LeCun, "Neuflow: A runtime reconfigurable dataflow processor for vision," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, june 2011, pp. 109 –116.
[7] P.-H. Pham, P. Mau, J. Kim, and C. Kim, "An on-chip network fabric supporting coarse-grained processor array," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10.1109/TVLSI.2011.2181546 (in press).
[8] R. Collobert, "Torch," presented at the Workshop on Machine Learning Open Source Software, NIPS, 2008.