The CAMEL Approach to Stacked Sensor Smart Cameras

Saibal Mukhopodhyay, Marilyn Wolf, Mohammed F. Amir, Evan Gebahrdt, Jong Hwan Ko, Jae Ha Kung, and Burhan A. Musassar
School of ECE, Georgia Institute of Technology, Atlanta GA USA

***Abstract*— Stacked image sensor systems combine an image sensor, memory, and processors using 3D technology. Stacking camera components that have traditionally been packaged separately provides several benefits: very high bandwidth out of the image sensor, allowing for higher frame rates; very low latency, providing opportunities for image processing and computer vision algorithms which can adapt at very high rates; and lower power consumption. This paper will review the characteristics of stacked image sensor systems and discuss novel algorithmic and systems concepts that are made possible by these stacked sensors.**

Keywords—embedded computer vision, stacked image sensor, smart camera

I. Introduction

Smart cameras are platforms for embedded, real-time computer vision [1,2]. High-performance heterogeneous multiprocessor systems-on-chips (MPSoCs) enabled the development of devices that perform computer vision operations in real time within the camera itself, rather than transmitting video back to a server. Smart cameras offer several advantages: keeping video in-camera enhances the privacy of the application; network bandwidth is substantially reduced; total deployment cost is lower; power consumption is markedly reduced. Smart cameras do require careful co-design of the algorithms and imaging/computing platform to make most efficient use of pixels and computing resources.

Traditionally, smart cameras had high-performance computing and memory placed out-side of the sensor chip. The limited off-chip bandwidth between sensor and processor has been a key bottleneck in improving overall performance and throughput of such cameras. However, advances in 3D VLSI have made possible a series of cameras on the market that provide high bandwidth [3, 4]. The 3D stacking of sensor, read-out-circuits (Analog-to-digital conversion), memory, and computation have been explored, and experimentally demonstrated [3, 4]. For example, stacked sensors have achieved very high frame rates (~1000 frame-per-seconds). Consequently, the stacked sensor creates new opportunities for designing smart cameras. The CAMEL (camera adaptation with embedded machine learning) project aims at developing a co-design methodology for stacked sensor smart cameras (SSSCs).

Stacked sensor smart cameras offer significant advantages beyond those of SoC-based smart cameras. They offer significantly higher bandwidth between the sensor, processor, and memory. That bandwidth can be used to increase both frame rates and computational performance. Increased bandwidth also provides low latency, which allows the camera to react more quickly to events. The small physical size and lower power consumption of these cameras allows them to be deployed in a wider range of environments. Jointly designing stacked sensor smart cameras and the algorithms that run on them allows us to exploit the advantages of this technology and to maximize the performance of these systems. This paper outlines our approach to the design of stacked sensor smart cameras. In particular, we discuss the advantage of stacked sensor from the perspective of computer vision, and study the feasibility and challenges of integrating advanced machine learning concepts within the camera module.

The rest of the paper is organized as follows. Section II describes the stacked sensor technology and application space; Section III outlines the design characteristics; Section IV studies the feasibility of embedding machine learning into the stacked sensor; Section V concludes the paper.

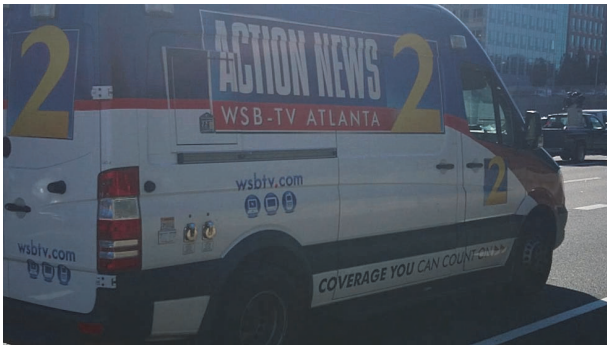
II. Stacked Sensor Smart Cameras and their Applications

High-bandwidth sensors are based on back-side illumination (BSI). The image sensor is ground down to move the back surface of the sensor closer to the photodetectors. That back side is faced toward the lens to capture the image. The front side of the sensor is then made available for interconnect. Back-side illumination combined with 3D stacking provide the basis for stacked sensor smart cameras. A typical stack would include several layers [3, 4, 5, 6]:

- An analog layer provides analog-to-digital conversion for pixels. It could also provide other forms of analog signal processing.
- A memory layer provides memory both for immediate pixel values as well as for processing elements.
- A compute layer provides processing elements in any of several styles: CPU, array processors, heterogeneous multiprocessors, *etc.*

978-3-9819263-0-9/DATE18/©2018 EDAA

30 frames/sec



960 frames/sec

Figure 1: A moving van captured at 30 frames/sec and 960 frames/sec.

Embedded computer vision is a real-time application with deadlines on computations. Stacked sensors provides increased bandwidth between the pixels and computation. High pixel bandwidth allows for much higher frame rates. For example, the Sony Xperia XZ can capture video at 960 FPS; it makes use of a three-layer stack with the image sensor, DRAM, and circuit layers [7]. Figure 1 illustrates the advantages of high frame rate for target classification. One image was taken at 30 frames/sec using a Canon DSLR; the other was taken at 960 frames/sec using an Xperia XZ. The van is considerably less blurred in the high-speed images. Both the position of the van and the writing on the van are easier to read in the high-speed capture images.

Traditional smart cameras rely on packaging and printed circuit boards to connect system components. 3D stacking and through-silicon vias (TSVs) provide lower parasitic, resulting in lower power consumption. Ultimately, the performance of 3D stacked sensor is limited by the thermal challenges of 3D ICs, as studied in detail by Lie et. al [5, 6]. The heat generated by high-performance computation, memory, and analog layers raise the temperature of the entire camera stack, including the pixel arrays in the image sensor. Higher temperature can increase the temporal and spatial noise in the image sensor, potentially reducing image quality [5, 6]. The problem is specially challenging for 3D sensors, as stacking create difficulties in removing heat, requiring consideration during the design of both architectures and applications.

III. Design Characteristics

We can better understand the advantages of stacked sensor smart cameras by working through some design formulas. We will base our analysis on a 1080p image with 1920 x 1080 = 1,036,800 pixels. For simplicity, we will assume 8-bit pixels.

Figure 2 shows an illustrative architecture and physical design of a representative stacked sensor smart camera [8]. This design includes four planes: the CMOS image sensor (CIS); analog/digital converters, which could be instantiated as one per column or one per pixel; static RAM; and logic.

First, consider the bandwidth required simply to move pixels from the image sensor into the processor for a non-stacked smart camera. Moving a 30 frames/sec video requires a bandwidth of 31 Mbytes/sec. Moving a 960 frames/sec video requires 995 Mbytes/sec of bandwidth. If we assume an 8-bit-wide port for pixels, the pixel pins would need to run at 3.88 Mbits/sec for the 30 frames/sec video *vs*. 124 Mbits/sec for the

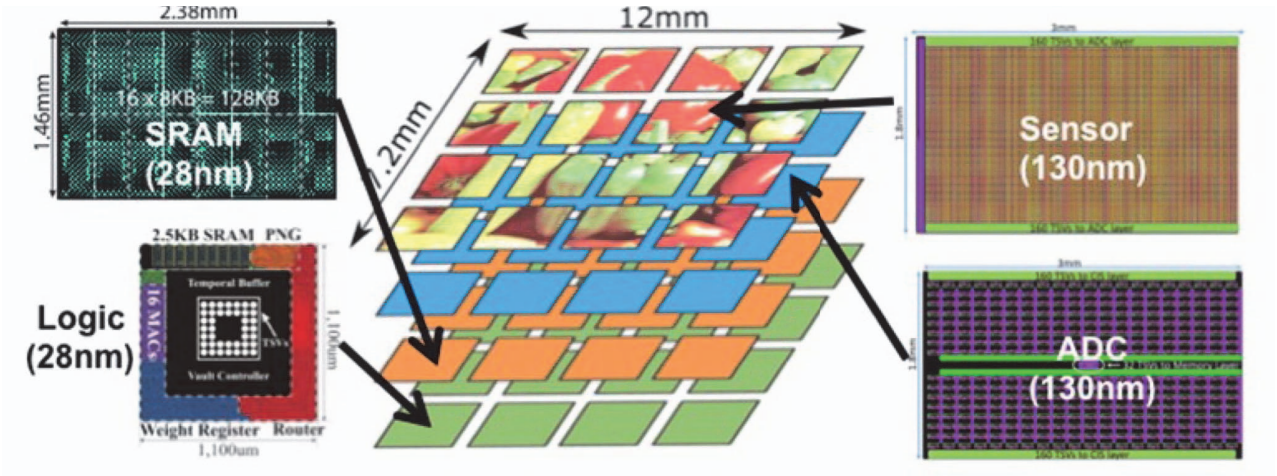


Figure 2: A sample architecture for a stacked sensor smart camera.

Figure 3: Multiple tracking targets.

960 frames/sec video.

Memory and compute are closely linked in embedded computer vision applications. Intermediate-level analysis algorithms that operate on pixels often provide embarrassing levels of parallelism. Optical flow [9], for example, requires two luminance frames in memory to perform the required matrix operations. The optical flow numerically estimates the gradient in x and y to produce a 2D array of local flow vectors. Traditional software implementations iterate over the images to generate the flow vectors. However, the flow vectors are independent of each other. Given sufficient bandwidth and processing elements, we can fully or partially parallelize the optical flow computation.

Tracking, a higher-level operation than optical flow, also provides substantial opportunities for parallelism. Figure 3 shows a sample frame from the XYZ testbench with a number of tracking targets identified. While some targets may occlude one another, many targets provide distinct regions-of-interest that can be easily parallelized.

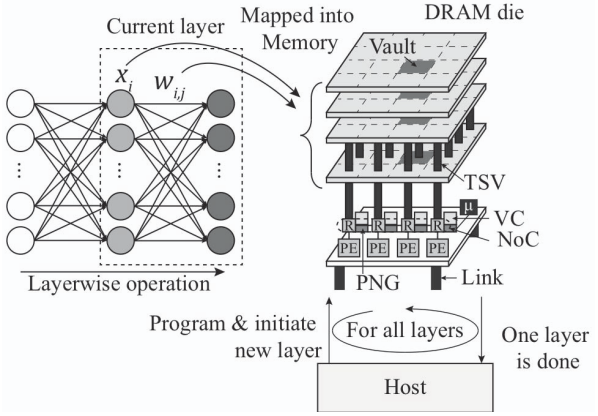


Figure 5. Architecture of the Neurocube design. [10].

IV. Machine-Learning Enabled Stacked Sensor

As discussed in the previous section, 3D integration provides opportunities to design high bandwidth and low-power CMOS image sensors (CIS) [3]. Several works have demonstrated integrating traditional vision processing and encoding within a 3D camera [3-6]. The next step is integration of machine learning based vision processing techniques within the 3D stacked sensor smart cameras. This section reviews our past work that will provide the background to achieve this goal [8, 10-13].

Amir *et al*. presented NeuroSensor [8], a smart 3D image sensor with an integrated convolutional neural network, shown in Figure 4. The rationale for this approach is that 3D integration of sensor, memory, and computing will effectively harness the inherent parallelism in neural algorithms. The design is based on Neurocube [10], a programmable and scalable digital neuromorphic architecture using 3D high density memory integrated with logic tier, shown in Figure 5a.

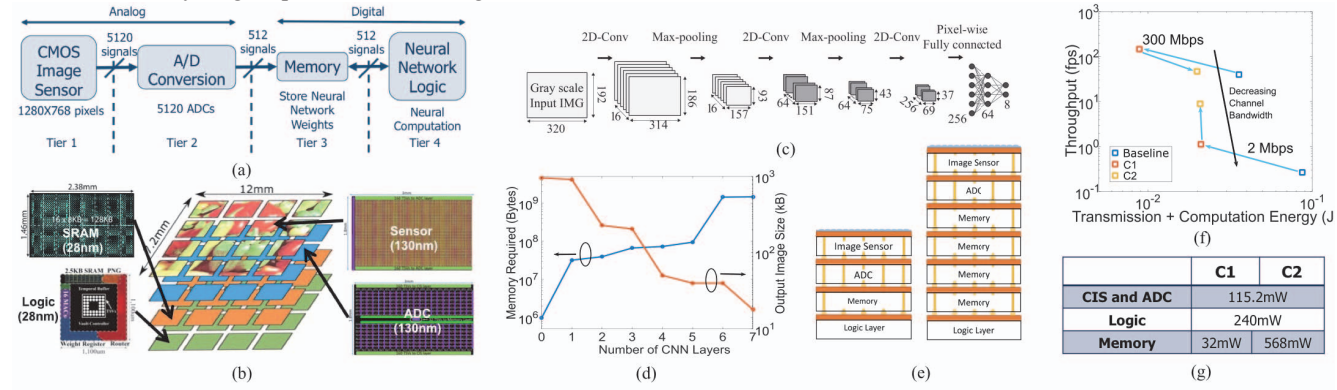


Figure 4. Architecture and simulation results of the Neurosensor design. (a) Architecture of the image sensor system with four tiers. (b) Organization of the 4X4 grid and layout of a single segment CIS, ADC, memory (SRAM), and logic tier. (c) Convolutional neural network and programming parameters for each layer. (d) Memory requirement increases and output image size decreases as more CNN layers are integrated in NeuroSensor. (e) Schematic of 3D image sensor system. (Left) Configuration C1 that performs only feature extraction. (Right) Configuration C2 having complete CNN with the extra memory tiers. (f) Variation of energy and throughput under limited channel bandwidth. (g) Power Breakdown for C1 and C2 [8].

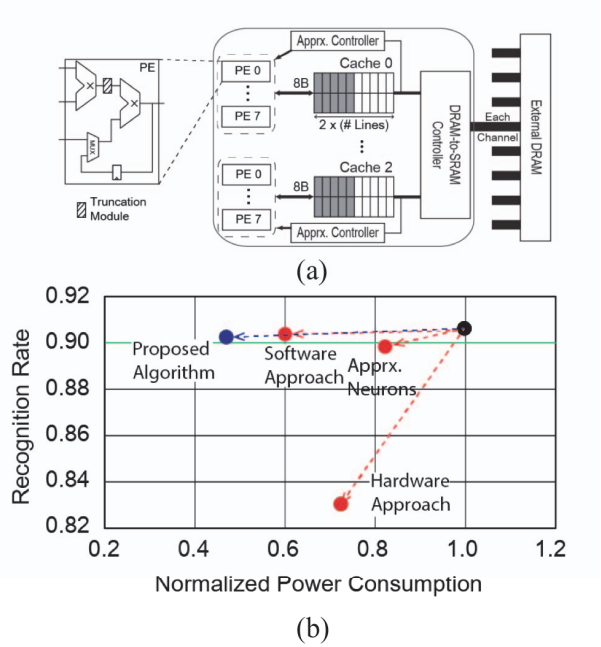


Figure 6. Application of approximate computing in a feedforward neural network. (a) A system diagram: a processing engine (PE) and the overall NN system including PEs, caches and controllers. (b) Comparison of the proposed algorithm to just doing software (reduced bit-precisions) or hardware (approximate PEs) approach [11].

The Neurocube integrates a fine grained, highly parallel compute layer within a 3D high-density memory package, the hybrid memory cube (HMC). The design exploits the data-driven nature and statically known memory access patterns of neuro-inspired algorithms to implement a programmable memory system. Host programming interface is a set of memory-based programmable state machines (neurosequence generators) that exposes abstractions for connectivity and synaptic weights as the vehicle for programming different neural nets. Neurocube synthesized with 15nm can deliver throughput up to 132GOPs/s during inference and 127GOPs/s during training within a compute power envelope of 3.4W in 15nm FinFet. The simulations project 4X improvement in computing power-efficiency (GOPs/s/W) over reported GPU based implementation while providing the programmability and scalability advantages over ASIC/FPGA platforms.

To study the trade-offs between complexity, performance, and power, two design cases are considered; the configuration C1 includes the initial 2D convolutional layer and max pooling layer, and C2 includes the complete Convolution Neural Network (CNN) platform [Figure 4]. As more layers of CNN are integrated on-chip the overall memory requirement increases. Conversely, more neural computation reduces the output image size, and hence, less bandwidth is required. System level power analysis, illustrated in Figure 4g, shows that logic is the most power consuming block in C1, while memory power dominates the system power in C2. C2 also shows better energy efficiency for low bandwidth scenarios as

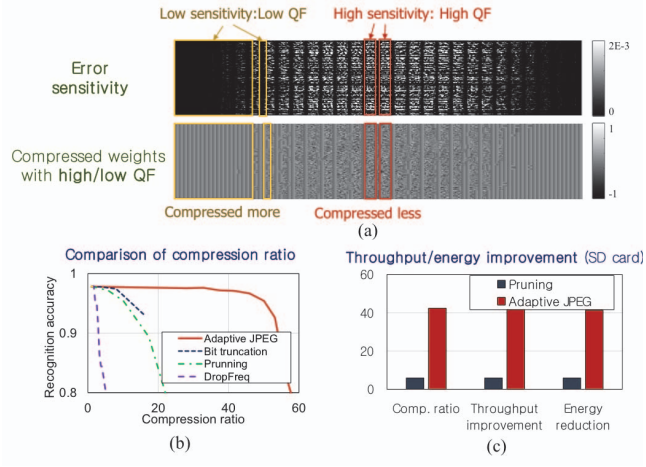


Figure 7.. Adaptive weight compression. (a) Concept of the adaptive weight compression scheme using two quality factors (QFs) of JPEG. (b) Comparison of compression ratio with other compression methods. (c) Throughput and energy improvement through the adaptive compression [13].

the low transmission energy (due to reduced size output) compensates for the high computation energy as shown in Figure 4f.

One of the key challenge to embed machine learning in 3D stack is reducing power dissipation of logic and memory system. Approximate computing has been applied to reduce computation demand of neural networks, by reducing the bit precision [14] or utilizing approximate multipliers [15]. Recently, Kung et al. proposed a power-aware digital neural network platform that integrates bit-precision control and approximate hardware [10], shown in Figure 6. First, the proposed approach determines a set of 'approximate synapses' in NN that have least impact on output quality using a greedy algorithm. The algorithm identifies them from the error sensitivity computed during the training phase. Next, the selected approximate synapses are processed with reduced bit-precision and/or using approximate PEs to reduce energy dissipation. A feedforward NN architecture is designed in 130nm CMOS utilizing the preceding approach. The analysis with the handwritten digit recognition problem (MNIST) shows that, compared to baseline design with all accurate PEs and synapses, the proposed approach reduces the PE power of feedforward NN by ~57% and the system power (including SRAM and controller) by ~38% with 0.4% degradation in quality as shown in Figure 6b. The concept of approximate computation has been further extended to recurrent neural networks. For example, Kung et. al. has presented feedback-controlled dynamic approximation that enables energy-accuracy trade-off in digital RNN for activity recognition [12]. The underlying principle is to reduce precision for frames where a specific activity dominates and can be recognized with high confidence. On other hand, precision is increased for sequences where detection confidence is lower. Authors have show feasibility of ~36% energy saving at the expense of ~4% degradation in recognition accuracy in 28nm CMOS design.

As neural networks require significant memory demand due to large synaptic weights, the second challenge is to reduce memory size and power dissipation. For example, the analysis in Figure 4 shows that the configuration 'C2' where the entire network is computed at the sensor is dominated by the power dissipation of the the memory system. Hence, there have been various approaches to compress the weights. Simple approaches are reducing the bit-precision of the weights and pruning connections [16, 17]. More recently, higher compression was achieved by the combination of pruning, quantization, and encoding [18]. As an alternative, Ko et al. proposed image-based approach that exploits the spatial locality and smoothness of the weight matrix [13]. To minimize the loss of accuracy due to JPEG encoding, this approach adaptively controls the quantization factor of the JPEG algorithm depending on the error-sensitivity (gradient) of each weight [Figure 7(a)]. With the adaptive compression technique, the weight blocks with higher sensitivity are compressed less for higher accuracy. The compression performance of the proposed approach is demonstrated with a multilayer perceptron (MLP) based neural network for three benchmark datasets. As Figure 7(b) shows, the presented method achieves 42X compression at the expense of ~1% degradation in recognition accuracy for MLP with the MNIST dataset. For system-level performance and energy analysis, a digital neural network inference engine with an JPEG decoder embedded in the memory controller is synthesized in 28nm CMOS technology. The proposed approach shows 3X higher effective memory bandwidth and ~19X lower system energy for inference [Figure 7(c)].

V. Conclusions and Future Work

Stacked sensor smart cameras promise to provide significant application-level improvements over traditional SoC-based smart cameras. The high bandwidth provided by 3D VLSI improves both performance and power management. The high levels of parallelism exposed by computer vision applications is well-suited to 3D computing systems. Integration of machine learning and neuromorphic processing is an important enhancement to stacked sensor smart cameras.

The future progress of the CAMEL project seeks to build on the preceding advancements to enable real-time feedback between embedded machine learning platform and the camera front-end to improve information content of the collected images. At a high level the goal is to integrated a machine learning algorithms that operate on the collected frames to determine how to modify the camera settings such that we collect most useful information with highest quality. The camera settings will include parameters such as exposure, spatial resolution, temporal sampling rates, to name a few. Achieving this goal will require innovations in ML algorithms and feedback control, and coupling those innovations with hardware architecture and power management.

Acknowledgment

This material is based on work was supported in parts by Office of Naval Research Young Investigation Program, National Science Foundation CAREER Award, and Defense Advanced Research Project Agency (DARPA). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, ONR, NSF, or the U.S. Government.

References

[1] B. Rinner and W. Wolf, "An introduction to distributed smart cameras," *Proceedings of the IEEE*, 96(10), October 2008, pp. 1565-1575.

[2] W. Wolf, B. Ozer, and T. Lv, "Smart cameras as embedded systems," *IEEE Computer*, 35(9) September 2002, pp. 48-53.

[3] [11] T. Kondo et al., "3-D-Stacked 16-Mpixel Global Shutter CMOS Image Sensor Using Reliable In-Pixel Four Million Microbump Interconnections with 7.6-µm Pitch," IEEE TED, 2016.

[4] T. Haruta *et al*., "4.6 A 1/2.3inch 20Mpixel 3-layer stacked CMOS Image Sensor with DRAM," *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, 2017, pp. 76-77.

[5] D. Lie, K. Chae and S. Mukhopadhyay, "Analysis of the Performance, Power, and Noise Characteristics of a CMOS Image Sensor With 3-D Integrated Image Compression Unit," in *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 4, no. 2, pp. 198-208, Feb. 2014.

[6] D. Lie, A. R. Trivedi and S. Mukhopadhyay, "Impact of Heterogeneous Technology Integration on the Power, Performance, and Quality of a 3D Image Sensor," in *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 1, pp. 61-67, Jan.-March 2016.

[7] Lars Rehm, "Sony Xperia SZ Premium features 960 fps slow-motion and 4K display," Digital Photography Review, February 27, 2017, https://www.dpreview.com/news/7636712678/sony-xperia-xz-premium-features-960-fps-slow-motion-and-4k-display.

[8] M. F. Amir et al., "NeuroSensor : A 3D Image Sensor with Integrated Neural Accelerator," IEEE S3S, 2016.

[9] C. Tomasi and T. Kanade, *Detection and Tracking of Point Features*, Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991.

[10] D. Kim et al., "Neurocube : A Programmable Digital Neuromorphic Architecture with High-Density 3D Memory," ISCA, 2016.

[11] J. Kung et al., "A power-aware digital feedforward neural network platform with backpropagation driven approximate synapses," ISLPED, 2015.

[12] J. Kung et al., "Dynamic Approximation with Feedback Control for Energy-Efficient Recurrent Neural Network Hardware," ISLPED, 2016.

[13] J. H. Ko et al., "Adaptive Weight Compression for Memory-Efficient Neural Networks," DATE, 2017.

[14] S. Venkataramani et al., "AxNN: Energy-Efficient Neuromorphic Systems using Approximate Computing," ISLPED, 2014.

[15] U. Lotrič et al., "Applicability of approximate multipliers in hardware neural networks," Neurocomputing, 2012.

[16] M. Courbariaux et al., "Low Precision Storage for Deep Learning," ICLR, 2015.

[17] J. A. Hertz et al., Introduction to the Theory of Neural Computation. 1991.

[18] S. Han et al., "Deep Compression - Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," ICLR, 2016.