Embedded Vision Processing on System-on-Chips (SoCs)

Menna El-Shaer

Abstract—The field of machine vision has evolved rapidly during the past few years, and embedded vision applications are ubiquitous nowadays due to the availability of high quality vision sensors and cameras. Nevertheless, these applications usually have strict real-time requirements that necessitate fast communication and processing. This article presents an overview of recent hardware and software architectures and their use in real-time image processing.

Index Terms—Embedded Machine Vision, Real-time computer vision applications, multicore processing, graphical processing units, deep learning, processor architectures, system-on-chip architectures

1 Introduction

THE past two decades have seen tremendous advances in machine vision applications in consumer products from smartphones and video analytics solutions to automotive safety systems. In contrast to traditional computer vision applications in industrial settings and automated manufacturing; implementation is usually constrained by cost, size and power consumption of system components. In addition, vision processing does not start with an image in the framebuffer [1], the entire real-time image processing pipeline starting from image acquision to output needs to be considered. Figure 1 shows the components that typically constitute a real-time vision processing system. The pipeline usually starts with the photoelectric conversion of input light rays into voltage using CMOS transistors which is passed through an ADC as digital pixels followed by their packet representation to be transferred for application processing using standard interfaces e.g. PCIe, USB, GigE, ...etc. Application processing is done on multicore homogeneous or heterogeneous processing elements that are built on a single chip with their inter-network interfaces. Integrating said components with memory and advanced peripherals produces a System-on-Chip (SoC), that are predominant in the embedded systems industry nowadays and have replaced the older Digital Signal Processors (DSPs) since they offer more computational power.

Most FPGAs nowadays have a soft core that includes an SoC in addition to the reconfigurable fabric. This aids in creating powerful vision systems since they combine the flexibility of adding custom hardware to the embedded peripherals to accelerate time-critical algorithms and frees precious programmable LUTs for application acceleration.

Beyond general purpose processors, Application Specific Integrated Circuit (ASIC) designs have existed since the 1980s [2] and have evolved with Moore's Law and Very Large Scale Integration (VLSI) technology into chips with more than one million transistors, deep pipelining structures and massively connected parallel compute resources,

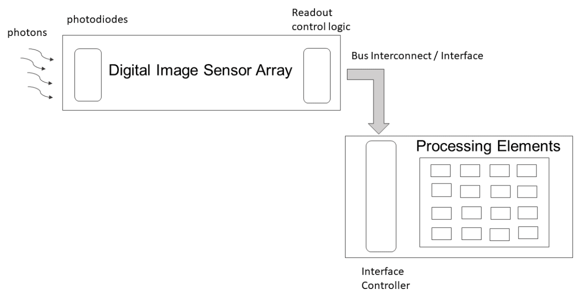

Fig. 1. A very high-level view of a typical embedded processing system used in machine vision applications

e.g. Network-on-Chip (NoC) designs that facilitate high performance computing applications such as machine vision. Low manufacturing costs and power budgets helped the SoC evolution in the embedded industry, where all system components are on the same die, thus reducing power consumption with high compute resources.

2 Hardware Architectures for Embedded Vision Applications

Traditionally, low-level vision architectures utilized 1D arrays or 2D meshes of processing elements for applications such as edge detection and image smoothing operations. The addition of DSP cores to multiprocessors on an SoC e.g. Texas Instruments' DaVinci family of processors [3] helped in implementing more complex applications like video processing. Nowadays, all systems have dedicated accelerator cores in addition to main CPUs where application specific vision algorithms are implemented. Dedicated cores could be GPUs, video processing units or even FPGA-implemented accelerator cores. Examples of different types of hardware accelerator chips used in machine vision are described next. By no means this list is exhaustive as the field is evolving rapidly and new architectures are designed everyday.

- *M. El-Shaer was with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH, 43210.*
E-mail: el-shaer.1@osu.edu

2.1 Hardware Accelerators Examples

A number of accelerators and vision and deep learning specific purpose accelerators have emerged in the past few years. Below are some of the most prominent at the time of writing.

- Google Tensor Processing Units [4]
  Their deep learning capabilities are accelerated by a Matrix Mutliply and Accumulate (MMA) 256x256 array of 8-bit multipliers.
- NVIDIA's Tesla V100 Chip [5]
  The 640 Tensor cores on this chip are also designed to accelerate MMA operations.
- Mobileye's EyeQ [6]
  This family of processors is designed specifically for automotive driving and ADAS applications.
- Intel's Knights Mill Chip (New generation Xeon Phi) [7]
  The individual cores on this many-core chip are smaller where inner loops can fit in L1-instruction caches. As a result, the performance of cores per socket is decreased but the number of cores is greater, which is good for compute intensive applications like deep networks. Its design is said to be a midway between a server CPU and a hardware accelerator.
- ThinCI's Chip [8]
  This chip incorporates small processors and a thread scheduler analogous to a CPU with execution units and an instruction scheduler. The new feature here is that the processors can stream data to each other instead of having to load/store from RAM each time a computation is needed.
- Data-Flow Engines
  One of the most sophisticated accelerator types. The main design is focused on how to map data graphs onto the data flow processing nodes to maximize computation speed and minimize Inter Process Communication overhead and synchronizations. They are basically many-core coprocessors featuring a network on a chip scratchpad memory model, suitable for a dataflow programming model, which should be suitable for many machine learning tasks.
  Examples are the Adapteva Epiphany processors [9], and Wave Computing's DPU [10].
- Movidius Vision Processing Units (VPU) [11]
  A multicore processor family with features fairly consistent with vision processing units that handle SIMD instructions and datatypes suitable for video with an on-chip DMA between scratchpad memories.
- GPU-based Accelerators
  Examples are NVIDIA's Tegra family of processors [12], and AMD's Radeon Instinct accelerators [13].
- Eyeriss [14]
  An FPGA-based accelerator for deep convolutional neural networks with low real-time energy consumption, that utilizes data reuse to avoid unnecessary reads and computations, and data compression to reduce memory bandwidth.
- Xilinx Automotive (XA) Spartan series [15]
  FPGAs designed specifically for ADAS applications.
- Synopsys' DesignWare EV5X processors [16], [17]
  A family of embedded vision processors with CNN capabilities.
- Intel's Nervana Neural Network Processor (NNP) [18]
  Another ASIC developed specifically for deep learning computations and memory operations.
- Neuromorphic-based Processors
  Integrated circuits design based on spiking neuron elements instead of traditional boolean logic gates, where a neuron fires a weighted range of values, in response to input stimuli within a certain period of time.
  Examples are the TrueNorth processors [19], and Intel's Loihi [20].

3 High-Level Software Frameworks

Despite the ease of directly using general parallel computing frameworks such *OpenMP* for shared-memory architectures, and *MPI* for distributed memory systems, on different multicore architectures to accelerate applications, that abstraction is usually not the best approach taken when designing embedded software. To get the most out of multicore systems, using programming models mapped to the system architecture that can well expose all types of parallelism on that system is recommended. When designing parallel programs, one should start with *partitioning* and breaking up computation among the different Processing Elements (PEs), either according to different functional steps in the algorithm i.e. functional decomposition, or according to data to be processed i.e. data decomposition [21]. After partitioning, *mapping* task groups to the available PEs is done. When mapping, data dependencies between tasks should be considered as well as reducing communication needs between PEs i.e. type of memory used: shared memory is usually fastest. Acceleration is thus achieved with increasing the total amount of work done per unit time i.e. application *throughput*, or decreasing turnaround times and overheads i.e. *latency*. The availabilty of profiling tools that can provide accurate space (memory) and time measurements can aid in the optimization process by providing feedback on the performance and adjusting accordingly.

4 Conclusion

New architectures are being developed everyday to speed up the processing of the huge number of pixel data collected from image sensors/cameras. Even though one might initially achieve fast processing times using the appropriate hardware or software; we usually optimize for acceleration by finding the best software implementation for a specific hardware implementation. Studying the interactions between hardware and software collectively at that optimization stage is suggested to achieve real-time performance requirements for many of the big data vision applications. Despite the availability of different accelerator types, a thorough study of an application's algorithm implementation on that accelerator instead of a generic solution can improve the algorithm's real-time performance moving forward.

References

[1] K. Branislav, S. S. Bhattacharya, and S. Chai, *Embedded computer vision*. Springer, 2010.

[2] S. K. Tewksbury, *Application Specific Integrated Circuits (ASICs)*, 1996.

[3] "Digital video processors products." [Online]. Available: http://www.ti.com/processors/dsp/media-processors/digital-video/products.html

[4] [Online]. Available: https://cloud.google.com/tpu/docs/system-architecture(7/12/2018)

[5] [Online]. Available: www.nvidia.com/v100(7/12/2018)

[6] [Online]. Available: https://www.mobileye.com/our-technology/evolution-eyeq-chip/(7/12/2018)

[7] [Online]. Available: https://ark.intel.com/products/series/132784/Intel-Xeon-Phi-72x5-Processor-Family(7/12/2018)

[8] [Online]. Available: https://thinci.com/about_us.html(7/12/2018)

[9] [Online]. Available: http://www.adapteva.com/epiphanyiv/(7/12/2018)

[10] [Online]. Available: https://wavecomp.ai/technology(7/12/2018)

[11] [Online]. Available: https://www.movidius.com/vision-processing-units(7/12/2018)

[12] [Online]. Available: http://www.nvidia.com/object/tegra.html(7/12/2018)

[13] [Online]. Available: https://www.amd.com/en/graphics/servers-radeon-instinct-mi(7/12/2018)

[14] Y. H. Chen, T. Krishna, J. Emer, and V. Sze, "14.5 eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, Jan 2016, pp. 262–263.

[15] *XA Spartan-6 Automotive FPGA*, Xilinx, 12 2012, v1.3.

[16] *DesignWare EV52 and EV54 Processors*, Synopsis, 2015.

[17] J. Campbell and V. Kazantsev, *Using an Embedded Vision Processor to Build an Efficient Object Recognition System*, May 2015.

[18] [Online]. Available: https://ai.intel.com/intel-nervana-neural-network-processor/(7/12/2018)

[19] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proceedings of the National Academy of Sciences*, vol. 113, no. 41, pp. 11 441–11 446, 2016. [Online]. Available: http://www.pnas.org/content/113/41/11441

[20] [Online]. Available: https://newsroom.intel.com/editorials/intels-new-self-learning-chip-promises-accelerate-artificial-intelligence/(7/12/2018)

[21] G. Barlas, "Chapter 2 - multicore and parallel program design," in *Multicore and GPU Programming*, G. Barlas, Ed. Boston: Morgan Kaufmann, 2015, pp. 27 – 54. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780124171374000022

Menna El-Shaer is an electrical and computer engineering Ph.D. student at the Ohio State University. Menna got her B.S. in computer engineering from Ain Shams University, Cairo, Egypt, in 2009. She has extensive computer programming experience and worked on multiple computer vision and image processing projects sponsored by the Office of Naval Research, the National Institute of Health, and the National Science Foundation. Menna has also taught multiple computer programming and hardware design at Wright State and Ohio State Universities.