Efficient DMA transfers management on embedded Linux PSoC for Deep-Learning gestures recognition

Using Dynamic Vision Sensor and NullHop one-layer CNN accelerator to play RoShamBo

Antonio Rios-Navarro[†]
Architecture and Tech of Computers Department
University of Seville
Sevilla, Spain
arios@atc.us.es

Ricardo Tapiador-Morales
Architecture and Tech of Computers Department
University of Seville
Sevilla, Spain
ricardo@atc.us.es

Gabriel Jimenez-Moreno
Architecture and Tech of Computers Department
University of Seville
Sevilla, Spain
gaji@atc.us.es

Alejandro Linares-Barranco
Architecture and Tech of Computers Department
University of Seville
Sevilla, Spain
alinares@atc.us.es

ABSTRACT

This demonstration shows a Dynamic Vision Sensor able to capture visual motion at a speed equivalent to a high-speed
camera (20k fps). The collected visual information is presented as normalized histogram to a CNN accelerator hardware, called NullHop, that is able to process a pre-trained CNN to play Roshambo against a human. The CNN designed for this purpose consist of 5 convolutional layers and a fully connected layer. The
latency for processing one histogram is 8ms. NullHop is deployed on the FPGA fabric of a PSoC from Xilinx, the Zynq 7100, which is based on a dual-core ARM computer and a Kintex-7 with 444K logic cells, integrated in the same chip. ARM computer is running Linux and a specific C++ controller is running the whole demo. This controller runs at user space in order to extract the maximum throughput thanks to an efficient use of the AXIStream, based of
DMA transfers. This short delay needed to process one visual histogram, allows us to average several consecutive classification
outputs. Therefore, it provides the best estimation of the symbol that the user presents to the visual sensor. This output is then mapped to present the winner symbol within the 60ms latency
that the brain considers acceptable before thinking that there is a trick.

KEYWORDS

Deep-Learning, CNN, hardware accelerator, FPGA, Linux, DMA

This work has been supported by Samsung Electronics Corporation through the NPP project, and by the excellence project from the Spanish government grant (with support from the EuropeanRegional Development Fund) COFNET (TEC2016-77785-P).

[†] The work of R. Tapiador has been supported by a"Formación de Personal Investigador" Scholarship from the University of Seville.

1 Scenario

Figure 1 shows a block diagram of the scenario used for this demonstration. A DVS sensor [1] is used for visual stimuli collection. This stimulus consists on the collection of a stream of events produced by the sensor through the cAER [2]. For this demo the system is programmed to collect 2k events and integrate them in a histogram. This histogram represents the input to the CNN that is being executed in the FPGA part of the system using the NullHop CNN accelerator [ref]. The Processing System (PS) is running a C++ application under Linux.

This application is in in charge of (1) taking the histogram from cAER that represents the input to the first layer, (2) configuring the accelerator with the kernels of the corresponding layer, (3) sending the input of the next convolutional layer to be computed, and (4) taking back its output. After the 5 iterations to complete the whole CNN a (5) fully-connected-layer is executed in software to obtain the final classification output.

A laptop is connected through ETH interface and SSH for Linux interaction to the application. Once the app is started, the console output can be watched on the laptop screen to check the execution latency per frame and the winner class of the CNN: paper, rock or scissor.

The PS can be connected to the FPGA part (programmable logic – PL) of the chip through several possible interfaces: AXI-Lite, AXI4 or AXI-Stream. This last one supports direct-memoryaccess (DMA), which extracts the maximum performance and it doesn't require a continuous intervention of the computer.

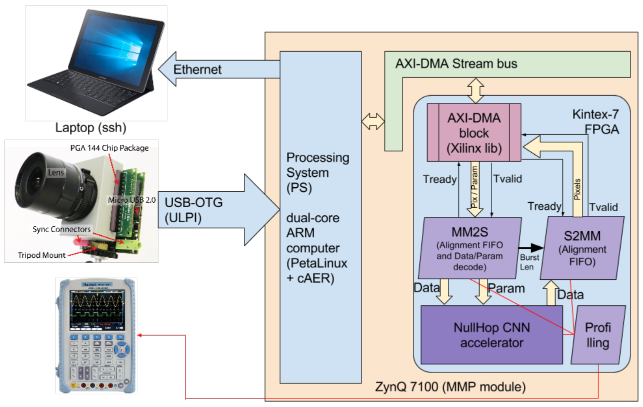

Figure 1: Block diagram of the system based on a PSoC with DMA interface.

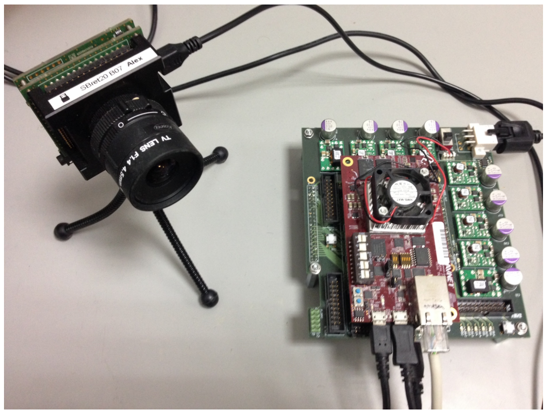

Figure 2: Picture of the system composed of a DVS retina and a SoC-Dock platform with a MMP module from AvNet (Zynq 7100).

Figure 2 shows the hardware used for this demonstration. There are two PCB connected. The one in the bottom is called SoC-Dock. It provides all the needed power supplies for the MMP commercial platform (based on Zynq 7100) on top and the USB-based JTAG interface.

2 AXI-DMA transfers management

First version of this co-design platform barely could process 30 histograms per second. Thanks to the use of optimized DMA interfaces (hardware FIFOs and state-machines) and software controllers with support of interruptions, simple/double buffered techniques and configurable transfer lengths, up to 120 fps can be processed in the current version when running the demo.

Figure 3 shows the data flow from the DDR connected to the PS, to the accelerator. Data managed by user app is stored in the virtual-space. This data needs to be moved to the physical space to be accessible by the DMA controller. These movements are done by the software API. Then the kernel driver is in charge of sequencing, configuring, starting and managing the interruptions for DMA transfers.

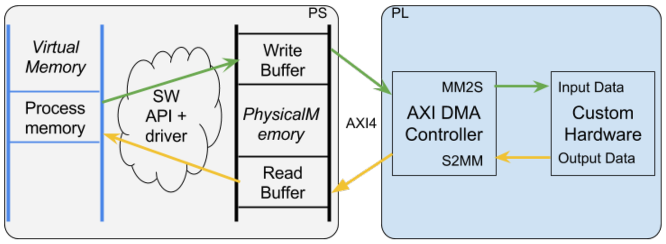

Figure 3: Picture of the system composed of a DVS retina and a SoC-Dock platform with a MMP module from AvNet (Zynq 7100).

NullHop RoShamBo	Unique mode, single-buffer		
	TX (us/byte)	RX (us/byte)	Frame (ms)
user-level polling	0.0054	0.197	6.31
user-level drv scheduled	0.0072	0.335	6.57
kernel-level drv	0.011	0.294	7.39

Table 1: TX/RX transfers latencies and frame computation times for Roshambo CNN for different tested techniques

Table 1 shows results over best configuration for three tests that have been performed using 5KB blocks. The testing scenario to obtain these latencies consists in a simple logic where any data coming from the processor through the AXI is sent back to the DDR through the same channel. The technique uses: (1) single-buffer, which establishes only one channel for data transfers between virtual and physical memory; (2) unique-mode sends all the data at once to the buffer, without any kind of partitioning. Two user-level versions have been compared: one completely based on polling, and a second one, closer to the kernel-level, where a scheduler is managing the different DMA requests, to avoid dead-lock waits. User-level polling obtains the best results.

ACKNOWLEDGMENTS

We thank the INI NPP team, coordinated by Tobi Delbruck, from UZH for the NullHop development and the collaboration and hard work done together in the success of this demonstration. This NPP project has been supported by Samsung Electronics Corporation.

REFERENCES

[1] P. Lichtsteiner, C. Posch, and T. Delbrück, "A 128 x 128 120 dB 15 usLatency Asynchronous Temporal Contrast Vision Sensor," IEEE Journalof Solid-State Circuits, vol. 43, pp. 566–576, feb 2008.
[2] Event-based processing framework for Neuromorphic devices, written in C/C++, targeting embedded and desktop systems. https://github.com/inivation/caer (accessed on 15-april-2019)
[3] A. Aimar, H. Mostafa, E. Calabrese, A. Ríos-Navarro, R. Tapiador-Morales, I.-A. Lungu, M. B. Milde, F. Corradi, A. Linares-Barranco, S.-C.Liu, and T. Delbrück, "Nullhop: a flexible convolutional neural networkaccelerator based on sparse representations of feature maps," CoRR,vol. abs/1706.0, 2017.