Software Hardware Co-processing in RISC-V using OpenVX™ for Embedded Vision Applications

Balaji Chegu, Prakash Reddy Battu, Yogesh Agrawal and Arun Naik
Microchip Technology India Pvt. Ltd.

Abstract

The complexity of computation logic in embedded vision applications is on the rise due to increase in video resolutions and image quality requirements. There are many frameworks that support acceleration of embedded vision applications, such as OpenVX™ [1]. It is a royalty-free cross platform framework using connected graph representation of operations In OpenVX™, image operations are expressed as graph of nodes, where nodes can be run on hardware or software. Typically, the run functions of the computationally complex nodes can be optimized in specialized hardware logic like FPGA to achieve acceleration and thus better FPS (frame per second). The graph is represented in C/C++ and is hosted on OS like Linux. This paper explains the acceleration of 2D convolution as OpenVX™ run functions using a Generic Matrix Multiplier (GMM) implemented in FPGA logic and summarizes the benefits of the proposed architecture by comparing the execution times and frame rates with and without FPGA acceleration.

Introduction

Embedded vision solutions in general can be developed on various platforms like CPU, ASIC or FPGA. CPU based software solutions are highly configurable as the function modules can be re-written for any change in requirement and can be developed in a less amount of time. But the acceleration in CPU is limited by its instruction set architecture (ISA) and number of cores. FPGA logic can give good acceleration by increasing the number of logic gates but can be reconfigured only by reprogramming. This is a time-consuming development cycle. Recently FPGA SoC (System on Chip) s architectures are gaining traction where some computing can be offloaded to the processor and some to the fabric to get the best of both worlds.

The concept of software-hardware coprocessing has been used for many years and the architectures have been evolving. Real time system with hardware-software co-design using Xilinx's ZedBoard are demonstrated in [2]. A convolution coprocessor developed in FPGA handles the image processing algorithms like 2D filtering. The role of the software in their system is confined to image acquisition via V4L2 and controlling the modules.

A System C based approach for software-hardware architecture based codesign methodology is demonstrated in [3]. System-C provides well defined set of C++ classes for flexible description of hardware. It also supports partitioning of image processing between hardware and software. However, this is useful for new designs where the developer must use System-C and the image processing platforms such as OpenVX™ are not directly compatible with this approach. An interesting mixed system is experimented in [4] where FPGAs and DSPs are connected via fast serial link. The DSPs take the floating-point computation load while the FPGAs provide fast I/O.

In this paper we demonstrate a novel method of wrapping a Register Transfer Logic (RTL) IP in C code that could be used in a software framework like OpenVX™ making it to work like a software library with other software modules. As a proof-of-concept Sobel edge detector on a live camera resolution of 720p where time consuming gradient calculation nodes are handled in Fabric, while L1 norm on the RISC-V processor is implemented.

Implementation

Hardware and Software Details

PolarFire® SoC FPGA SEV Kit [5] features a full pledged multimedia development kit with MPFS250T PolarFire® SoC and interfaces like CSI-2, HDMI etc. The SoC combines 4 RISC-V U54 application cores for a Microprocessor Subsystem (MSS) that can run Linux® and the PolarFire® fabric in a single device.

The MSS contains 4 Fabric Interconnects (FIC) that are used to communicate with fabric. The FICs support protocols like AXI and APB. It can host an Embedded Linux® built on Yocto Project® built system with feature rich libraries like libpthread, libgpiod and other necessary drivers. A new recipe for OpenVX™ is written for Yocto® and thus making it available in the user libraries.

Hardware-Software Interactions

An RTL IP called Generic Matrix Multiplier (GMM) that can do four 2D convolutions in parallel is designed. The GMM is used to calculate horizontal and vertical gradients for Sobel edge detection. The FPGA resource count for the

IP is – 4K 4LUTs, 3K DFFs, 20 LSRAM 18K and 20 Math 18x18 blocks on MPFS250T FPGA.

To mimic a software function call to the GMM, it is equipped with the following control logic.

1. Parameter configuration: Input and output addresses, convolution coefficients are passed through APB interface. The APB is a memory mapped region which can be accessed through devmem from Linux user space.
2. Control logic: The MSS triggers a GPIO whenever the data needs to be processed and gets an acknowledgement in other GPIO. A userspace library called libgpiod is used for this purpose.
3. Data transfers: The data transfers between cached (DDR-C) and non-cached (DDR-NC) regions are done using PDMA (platform direct memory access). Fabric computations are efficient in DDR-NC region and the MSS i.e., RISC-V processor's computations are efficient in DDR-C. So, the data needs to be moved appropriately and PDMA helps in these memory transfers.

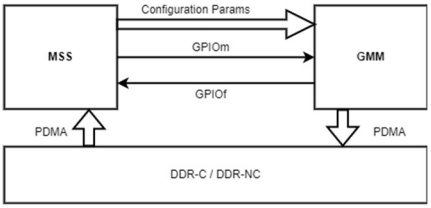

Fig 1: MSS and GMM interface

GMM as OpenVX™ Node

Any function with well-defined C interface can be a run function to a user defined node in OpenVX™ framework. Conv2D user kernel with appropriate input/output validators and GMM as run function is coded in C. This is built on Yocto and the user node from this kernel is registered for the OpenVX™ graph. The following figure shows the nodes in Sobel graph. The Conv2D node executes in the fabric and L1 Norm in RISC-V demonstrating a hardware-software co-processing.

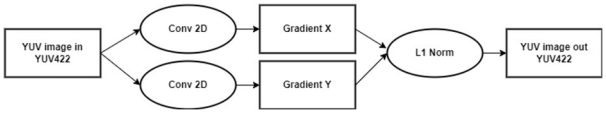

Fig 2: Sobel edge detector graph running in RISC-V with fabric coprocessing.

1. YUV Image in, YUV Image out: Memory mapped region in DDR-NC. It can be accessed by Fabric thru the physical address. OpenVX™ can manage this thru APIs like *vxCreateImageFromHandle*.
2. Conv2D – A user defined node with GMM running in fabric as run function. Computes gradient and transfers the output to DDR-C region for the MSS to consume.
3. L1 Norm – Another user defined node that completely runs on RISC-V.
4. Gradient X, Y: Memory mapped regions in DDR-C which are accessed by RISC-V.

Results

For HD (1280x720) resolution image the GMM takes about 9 msec to perform 2D convolution whereas the same in software i.e., in C it takes about 30 msec using 4 threads. The L1 norm which is run in C takes about 15 msec. The node execution is in sequence i.e., first the Gradient X is calculated, then Gradient Y and at last L1 Norm. Hence the overall Sobel magnitude takes 33 msec (9+9+15) when the 2D convolution is performed using GMM and takes 75 msec (30+30+15) when the 2D convolution is performed in C.

Configuration	Time taken in msec	FPS
Without SW-HW coprocessing	75	13
With SW-HW coprocessing	33	30

Conclusion

A novel method of prototyping an algorithm in C using OpenVX™ and later accelerating the needed portions in FPGA using a GMM is presented in this paper. An unconventional approach of giving a C interface to fabric IP (GMM) and integrating it into OpenVX™ framework is realized. This facilitates the reuse of FPGA IPs very identical to user space C libraries.

Complex use cases like stereo vision are better suitable for this approach. For instance, the disparity estimation which involves lot of bitwise operations can be done in fabric and other computations on the processor.

References

[1] https://www.khronos.org/openvx/.

[2] M. Ali Altuncu, Taner Guven, Yasar Becerikli, Suhap Sahin, "Real-time system implementation for image processing with hardware/software co-design on the Xilinx Zynq platform", IJIEE, November 2015, 473-477.

[3] Wang Chong, et.al., "Hardware/software co-design of embedded image processing system using systemc modeling platform," 2011 International Conference on Image Analysis and Signal Processing, 2011, pp. 524-528

[4] Adler, Felix et.al., "A new versatile hardware platform for digital real-time simulation: Verification and evaluation", 2012 IEEE 13th Workshop on COMPEL.

[5] https://www.microchip.com/en-us/development-tool/MPFS250-VIDEO-KIT