High-performance Imaging Subsystems and their Integration in Mobile Devices

Menno Lindwer, Mark Ruvald Pedersen
IAG/MCG/VIED
Intel Corporation
Eindhoven, The Netherlands

Abstract— **Within today's SoCs, functionality such as video, audio, graphics, and imaging is increasingly integrated through IP blocks, which are subsystems in their own right. Integration of IP blocks within SoCs always brought software integration aspects with it. However, since these subsystems increasingly consist of programmable processors, many more layers of firmware and software need to be integrated. In the imaging domain, this is particularly true. Imaging subsystems typically are highly heterogeneous, with high levels of parallelism. The construction of their firmware requires target-specific optimization, yet needs to take interoperability with sensor input systems and graphics/display subsystems into account. Hard real-time scheduling within the subsystem needs to cooperate with less stringent image analytics and SoC-level (OS) scheduling. In many of today's systems, the latter often only supports soft scheduling deadlines. At HW level, IP subsystems need to be integrated such that they can efficiently exchange both short-latency control signals and high-bandwidth data-plane blocks. Solutions exist, but need to be properly configured. However, at the SW level, currently no support exists that provides (i) efficient programmability, (ii) SW abstraction of all the different HW features of these blocks, and (iii) interoperability of these blocks. Starting points could be languages such as OpenCL and OpenCV, which do provide some abstractions, but are not yet sufficiently versatile.**

Keywords—MPSoC; ASIP; imaging; IP integration; Software

The recent spectacular progress in semiconductor technologies has enabled implementation of increasingly proficient multiprocessor subsystems, and their integration in large Systems-on-Chip (SoCs). A big stimulus has been created towards the development of innovative subsystems. However, this has introduced new silicon and system complexities, resulting in a number of difficult design issues, such as:

- vastly increased validation complexity;
- interconnect scalability, while accounting for physical system characteristics (leakage, wire latency, etc.);
- pressure on development and production budgets, shorter time-to-market;
- programmability of accelerators; while making sure that processors within all of the IP subsystems are optimally utilized;
- Power consumption needs to be in the same ballpark as fixed-function solutions.

This leads to multi-ASIP systems, with a low number of processors, yet each processor being highly application-specific and inherently very parallel in nature (e.g. 100s of SIMD vector elements). The development of software such heterogeneous programmable IP subsystems involves many stages such as: application analysis and characterization, application parallelization and partitioning, application scheduling and mapping, and software compilation. In many cases, the hardware configuration of IP subsystems is being driven from such application analysis. And in such cases, the phases of application development actually also encompass system macro-architecture design, processor design, and hardware generation.

Similar aspects apply to many other IP subsystems as they are currently being integrated in SoCs. For example, audio subsystems need to support many audio coding standards and use cases [2]. Graphics engines are increasingly being used for video analytics tasks, while rendering images and performing hard real-time display post-processing. Software-defined radios in communication systems need to scan multiple bands and multiple protocols, and respond immediately to activity in any one of those protocols.

The traditional approach to programming multi-core systems, consisting of multiple IP subsystems has been to construct Kahn Process Networks (KPNs), or similar networks. We argue that these formalisms are not powerful enough to deal with the issues of programming systems consisting of a limited number of truly heterogeneous and inherently parallel processors.

OpenCL [3] is sometimes presented as a solution, offering abstractions, custom devices, and a programming language for accelerators. However, its programming language lacks generic support for many types of ASIPs. OpenCL provides a means to declare and handle buffers, yet does not support efficient buffers access by ASIPs. The compute model assumes that the CPU is synchronizing operation between other compute elements. This is not desirable.

Other approaches, such as OpenGL and OpenCV are specifically meant for certain application domains and therefore do not provide an easy-to-use generic acceleration mechanism.

We present an analysis of the challenges when aiming to achieve a generic way of utilizing and integrating IP subsystems, consisting of such programmable accelerators.

ACKNOWLEDGMENT

This work is being performed in the scope of the ASAM project [1] of the European ARTEMIS Research Program and has been partly supported by the ARTEMIS Joint Under-taking under grant no. 100265

REFERENCES

[1] L. Jóźwiak, M.M. Lindwer, R. Corvino, P. Meloni, L. Micconi, J. Madsen, E. Diken, D. Gangadharan, R. Jordans, S. Pomata, P. Pop, G. Tuveri and L. Raffo, "ASAM: Automatic Architecture Synthesis and Application Mapping", DSD 2012 - 15th Euromicro Conference on Digital System Design, pages 1-11, Cesme, Izmir, Turkey, 2012.

[2] P. van der Wolf and R. Derwig, "Modular SoC Integration with Subsystems; The Audio Subsystem Case", DATE'13

[3] A. Munshi and others, "The opencl specification," Khronos OpenCL Working Group, vol. 1, pp. 11–15, 2009.

978-3-9815370-0-0/DATE13/©2013 EDAA