https://doi.org/10.2352/ISSN.2470-1173.2019.9.IMSE-363

System-on-Chip Design Flow for the Image Signal Processor of a Nonlinear CMOS Imaging System

Maikon Nascimento, Dileepan Joseph; University of Alberta; Edmonton, AB, Canada

Abstract

A system-on-chip (SoC) platform having a dual-core microprocessor (µP) and a field-programmable gate array (FPGA), as well as interfaces for sensors and networking, is a promising architecture for edge computing applications in computer vision. In this paper, we consider a case study involving the low-cost Zynq-7000 SoC, which is used to implement a three-stage image signal processor (ISP), for a nonlinear CMOS image sensor (CIS), and to interface the imaging system to a network. Although the high-definition imaging system operates efficiently in hard real time, by exploiting an FPGA implementation, it sends information over the network on demand only, by exploiting a Linux-based µP implementation. In the case study, the Zynq-7000 SoC is configured in a novel way. In particular, to guarantee hard real time performance, the FPGA is always the master, communicating with the µP through interrupt service routines and direct memory access channels. Results include a validation of the overall system, using a simulated CIS, and an analysis of the system complexity. On this low-cost SoC, resources are available for significant additional complexity, to integrate a computer vision application, in future, with the nonlinear CMOS imaging system.

Introduction

Nowadays, autonomous devices require high processing data capabilities and portability, to be able to connect to any operating system (OS), but have restrictions in power, weight, and cost. Low latency, between inputs and outputs, is also crucial for high speed applications, which means the use of cloud computing is not ideal. These are technical challenges of edge computing systems, as described by Lee *et al.* [1], which feature large scales, distributed networks, cyber-physical interfaces, dynamic and adaptive environments, and heterogeneous platforms.

A case study of a multi-stage image signal processor (ISP), implemented with a system-on-chip (SoC) platform, demonstrates a capacity to process a large amount of data in hard real-time while providing web content. A field-programmable gate array (FPGA) embeds the image signal processing algorithms, while realizing true parallel processing. A microprocessor (µP) handles hardware interfaces for storage and networking, including the high-level protocols to serve web pages.

Our system corresponds well to key aspects of edge computing, where the device is capable of processing a large amount of data, namely high definition (HD) high dynamic range (HDR) video, reporting processed information in a high-level format, not overloading the network, and pushing the processing to the edge. Our method also addresses issues faced by current cloud solutions, reviewed by Shi and Dustdar in "The Promise of Edge Computing" [2], by having low latency and private data (all processing is done locally), while processing a large amount of data.

This work uses the FPGA as the main platform to embed all circuits that compose a three-stage pipelined ISP required for a nonlinear CMOS image sensor (CIS). The research delivers a design flow for hard real-time to be achieved with an SoC. Underlying software algorithms were previously published by our group. They include fixed pattern noise (FPN) correction and salt-and-pepper noise (SPN) filtering [3], as well as their digital circuit designs [4], and a tone mapping operator (TMO) [5].

Lapray *et al.* [6] developed an HDR video camera using a Virtex 6 FPGA from Xilinx. They achieved 60 frames per second (FPS) with a spatial resolution of 1280×1024 pixels in a homogeneous implementation, using only an FPGA. A lot of FPGA resources were spent in generating the HDR image by combining low dynamic range (LDR) images with different exposure times. However, the Virtex 6 is a high-end FPGA and their TMO might not be able to handle abrupt changes in illumination.

Another full HDR video camera applied for welding was developed by Mann *et al.* [7]. The system is real-time and low cost, deployed using a Spartan-6 LX45 FPGA from Xilinx. Their design employs look-up tables (LUTs), which guarantees real-time, but a lot of pre-processing has to be made to generate the LUTs. It is a good implementation for homogeneous computing, using only an FPGA that lacks connectivity. As well, the use of LUTs may be too simple to handle fast changes in luminance.

The novelty of this research is the use of a heterogeneous SoC to realize an embedded real-time ISP for an HDR imaging system that will incorporate a nonlinear CIS. Acting as the master of the system, the FPGA post-processes and interfaces the nonlinear CIS to the µP. The µP enables the system connectivity and extra processing could be realized in the µP as well.

Apparatus and Application

After describing the platform of our ISP, this section explains its functionality. As the explanation is high level, using input-output images, please consult the references for more details about the background algorithms and circuits.

System-on-Chip Platform

Fig. 1 shows the SoC development kit we used. The SoC itself is a XC7Z020 from the Zynq-7000 family of the Xilinx All Programmable SoC architecture. It is composed of a dual-core ARM Cortex-A9, called the processing system (PS), and a 7-series FPGA from Xilinx, called the programmable logic (PL). In this paper, we simply write µP for the PS and FPGA for the PL to favour generic terminology.

From the available peripherals, we use: the Joint Test Action Group (JTAG) for debugging; the Universal Serial Bus (USB) universal asynchronous receiver-transmitter (UART) for serial communication; the Ethernet controller for networking; and the mem-

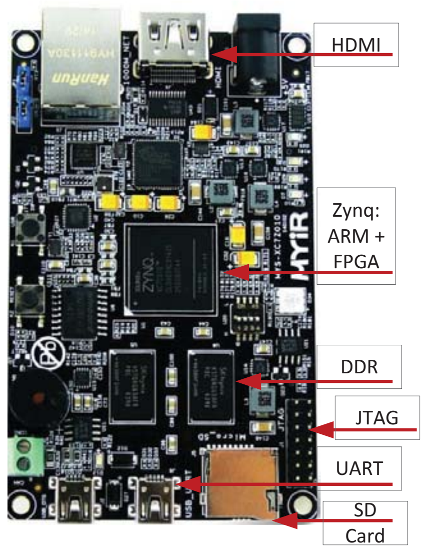


Figure 1. *Zturn development kit with Zynq-7000 SoC. This board is the platform chosen to deploy the proposed ISP, and thereby to realize a case study on edge computing. The image is adapted from the literature [8].*

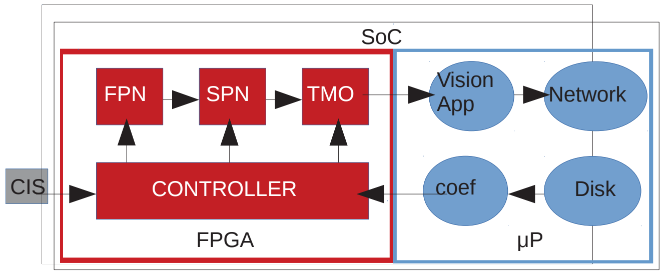


Figure 2. *Interconnection of primary SoC components. Some are in the FPGA, while others are in the µP. A key part of this work is the interfacing.*

ory controller to access 1 GB of random-access memory (RAM), specifically double data rate (DDR) synchronous dynamic RAM (SDRAM). Our ISP requires memory to store the FPN coefficients, which fit easily into the DDR memory after loading from the SD card. The OS is booted from the SD card as well.

Documentation and examples are provided by the vendor, such as C code to embed in Linux and access the board's peripherals, kernel files to configure and recompile the Linux kernel, and device tree source files, to add new hardware components.

Xilinx has a suite of tools, called Vivado, for system design with the Zynq-7000. Vivado can work with hardware description language (HDL) coding and drag-and-drop graphical user interfaces (GUIs). For bare-metal development, i.e., to program the µP with no OS, a software development kit (SDK) is included for C and C++. The Vivado High-Level Synthesis (HLS) is a third integrated development environment (IDE) that allows designers to develop their FPGA projects using C, C++, and/or System C only, without the need for HDL coding.

For standard components, such as direct memory access (DMA) and video drivers, we use intellectual property (IP) circuits from Xilinx. We also designed custom circuits to make our ISP. Drivers needed by the SoC are available, according to the FPGA design. For this project, we do not use Vivado HLS.

Image Signal Processor

The SoC allows the design of a heterogeneous system, where work is divided between an FPGA and µP, as shown in Fig. 2. The proposed system takes advantage of the highly parallel architecture provided by the FPGA and the rich set of peripheral devices and software libraries available on the µP.

On the FPGA side, a custom digital architecture can execute pixel-level algorithms and multi-channel tasks efficiently, taking advantage of the high throughput and low power of FPGAs, especially for fixed-point computation. The FPGA can also extend the peripherals of the µP, e.g., to accommodate a nonlinear CIS that requires a custom and sophisticated ISP to be functional.

The µP is composed of: a dual-core ARM processor; standard peripherals for connectivity; and GB-scale memory access with memory controllers. It is capable of running Linux, a multi-threaded OS, and performing additional computation.

As it is "low-power yet high-performance," to quote Kalb *et al.* [9], this heterogeneous system is ideal for realizing the ISP of a nonlinear CMOS imaging system. However, the design flow is challenging due to multiple development environments and the complexities of interfacing the FPGA and µP. Kalb *et al.* worked on a more unified tool chain, a customized real-time kernel for parallel computing, and a hardware system.

A nonlinear CIS, such as the logarithmic one by Mahmoodi *et al.* [10], is ideal for HDR imaging at video rates because it does not need to create the HDR image from multiple LDR images [6] [7]. While nonlinear CISs have been investigated before, the literature generally focuses on small-format cases.

In contrast, using HD HDR videos from Kronander *et al.* [11], this work simulates a large-format nonlinear CIS. Videos were pre-processed to simulate the responses of Mahmoodi *et al.*'s image sensor [10], except with the larger HD format. This resulted in compressed logarithmic responses, having 16 bits of pixel depth, with FPN and SPN incorporated.

Fig. 3 demonstrates the importance of the three main circuits in the proposed ISP. The "All ISP" row shows the result with all circuits. The "No TMO" row shows the result with a simple TMO in place of Li *et al.*'s one [3]. The "No SPN" row shows what happens when SPN is not filtered. Finally, the "No FPN" row shows what happens when FPN is not corrected.

To make the ISP operational, a controller circuit is added to the design to provide all control and status signals. The controller implements the external communication protocol, called Advanced eXtensible Interface (AXI), between the ISP and the DMA, an AXI4-Stream. At the ISP input, the controller extracts CIS pixel values and FPN correction coefficients from data packages received via DMA. Pixel addresses are generated by the controller and provided to the SPN filter. Finally, frame synchronization signals are generated for the TMO.

Interfacing Method

As shown in Fig. 4, the interfaces between the FPGA and µP include: high performance (HP) ports, for high speed data transfer; and general purpose (GP) ports, for configuration and control. We also set up interrupts to make the FPGA the master of the sys-

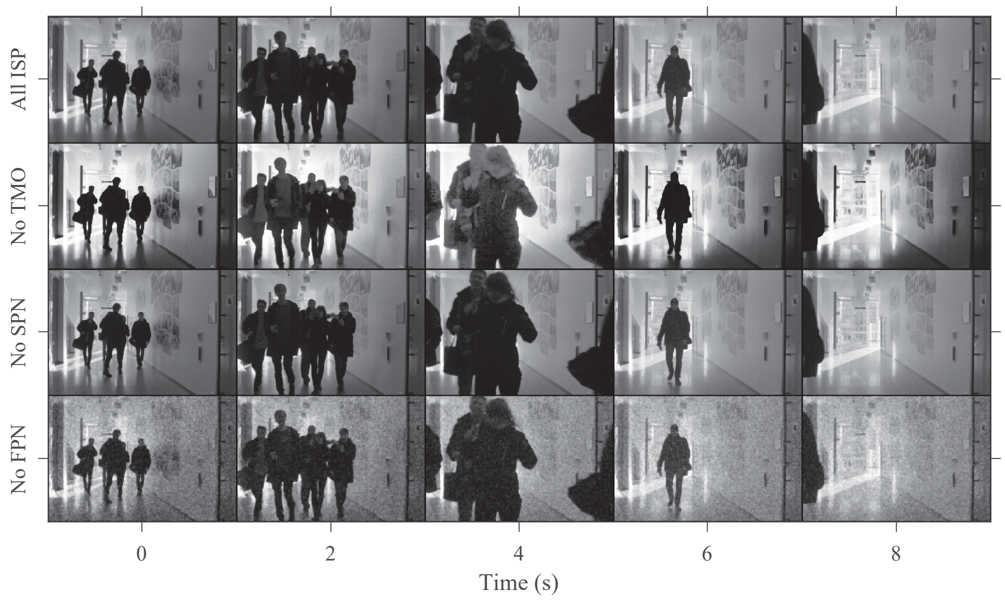


Figure 3. *Illustration of the ISP functions. From bottom to top, the image quality improves by adding circuits for FPN correction, SPN filtering, and tone mapping.*

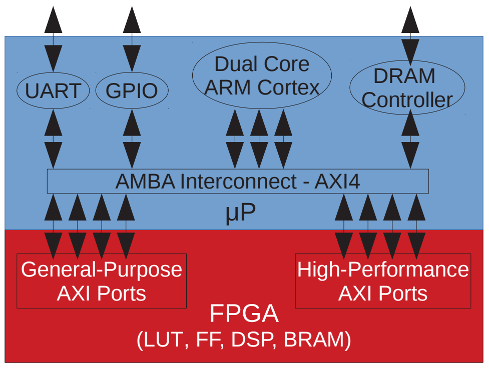


Figure 4. *Zynq-7000 SoC main blocks and interfaces. The AMBA interconnect protocol is utilized to bridge the FPGA and µP sides of the system.*

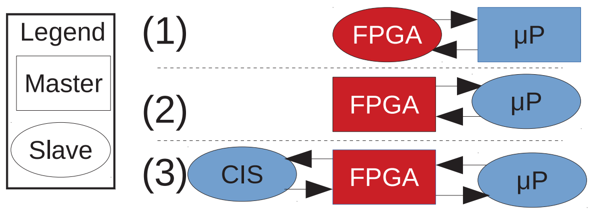


Figure 5. *Potential master-slave configurations of the system. Squares indicate the master, and ovals the slave(s). We currently use approach (2).*

tem and to avoid wasteful polling by the µP.

Our approach differs from the hardware acceleration methodology, shown in Fig. 5(1), of the literature. Because we currently simulate the nonlinear CIS, our approach is as shown in Fig. 5(2). Ideally, the architecture would be as shown in Fig. 5(3), where an actual nonlinear CIS is controlled by the FPGA.

The protocol we adopted is AXI from Advanced Microcontroller Bus Architectures (AMBAs). It has three variations: AXI4-Full, for multiple devices on HP buses; AXI4-Little, for control; and AXI4-Stream, for point-to-point communication. We use AXI4-Stream, which is the simplest and more efficient version. This protocol, which has a finite state machine with four states, is implemented in the controller circuit.

The design flow for the SoC can be classified either as horizontal or vertical. In the horizontal case, as in Fig. 6(1), the FPGA design in Vivado yields the bit file that configures the FPGA. For the µP, repositories and makefiles, configured for cross compilation, generate the boot file, kernel, device tree binary (DTB), and root file system. In the vertical case, as in Fig. 6(2), Vivado exports a hardware description file (HDF), which is a container with the FPGA bit file and other design configuration files. PetaLinux [12] then uses the HDF to generate the configuration files for the µP. Modules, packages, and applications can be added in PetaLinux before building the whole project.

We use DMA to transfer data efficiently between the FPGA and µP. The DMA is realized by a standard IP block, which we add to the FPGA design, that is configured and controlled by software running on the µP. To make the FPGA the master, we trigger the µP to run the software, implemented as a Linux interrupt service routine (ISR), in response to interrupt requests (IRQs) generated by the FPGA and detected by the µP.

The ISR may be in the kernel space or user space of the OS. We implemented it in the user space, which does not require a custom device driver. We make use of control and status registers, which are memory mapped, of the DMA circuit [13].

A kernel space implementation is more robust, but requires more time and expertise. To design a custom device driver, there are three components: memory allocation, DMA device control, and cache control. Also, the kernel has to reserve a contiguous area of memory in which the driver operates. As in our user space implementation, we favour the simple operation mode of the DMA, as opposed to the scatter-gather mode, because it simplifies both the ISR and AXI4-Stream implementations.

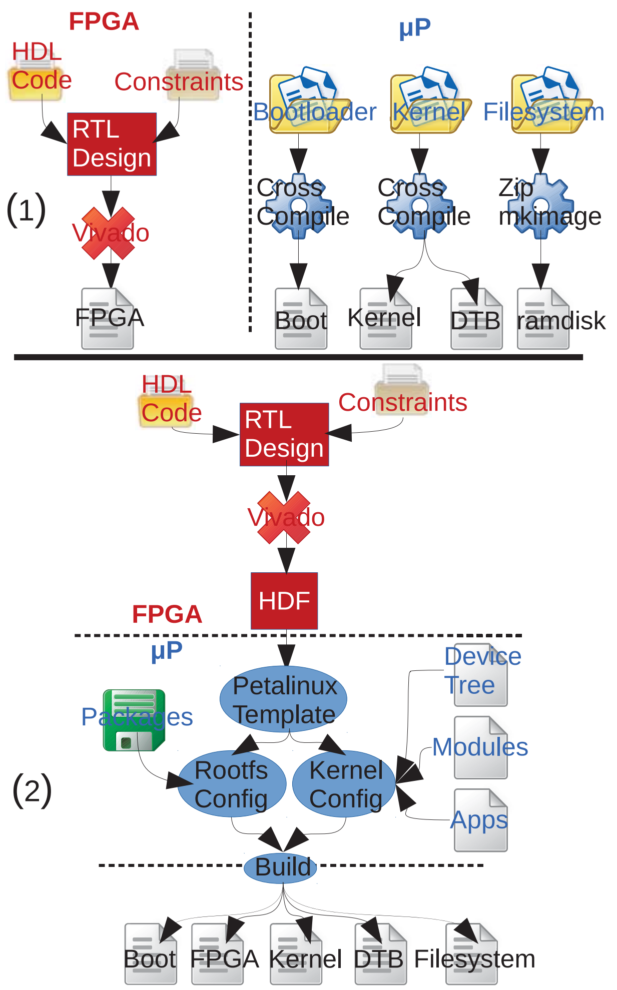


Figure 6. *Potential design flows for the SoC system. (1) The horizontal one has a more independent methodology, where compilation of the FPGA and μP projects are done separately. (2) The vertical one emphasizes integration, making it easier to load extra packages and configurations.*

Results and Discussion

Having presented the apparatus, application, and method, we now turn our attention to system validation and evaluation.

System Validation

On the FPGA side, after HDL code entry and synthesis, the design is tested with functional validation, where most bugs are caught. An initial validation is realized manually with a few small images. Thereafter, for many large images, automatic validation is performed against the same data processed in MATLAB. Binary files are used to load input data and save output data. After debugging, zero bit error is consistently achieved.

On the μP side, an initial test is to verify that the Linux kernel detects the new device, i.e., the interrupt-based DMA transfer. This can be done by entering “`dmesg`” at a command prompt. Other useful commands are “`cat /proc/interrupts`,” to verify the interrupt number and name, and “`cat /proc/iomem`,” to verify the DMA address. When using a PetaLinux driver, self tests of the DMA can be loaded, in the kernel configuration phase, and

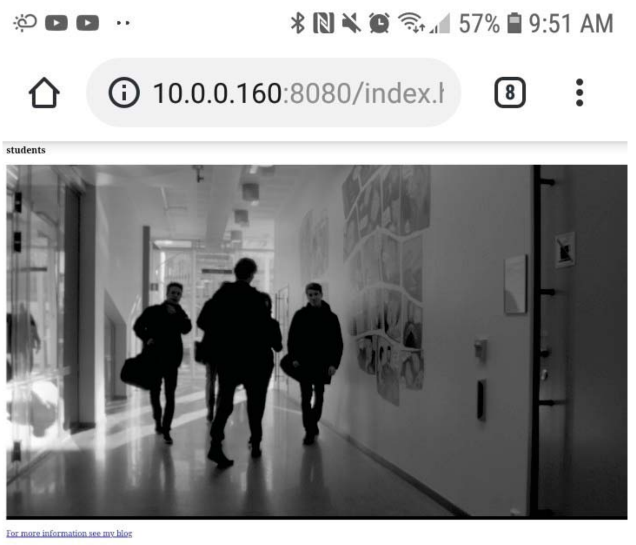


Figure 7. *Browser screenshot of a web page served by the SoC system. The μP is running a web server to serve the FPGA output over a network.*

basic DMA operation can be verified during boot.

To validate the system, a web server was implemented on the μP. It serves a simple web page, accessible from any browser on the same network, that contains a bitmap image. The ISR we wrote for Linux, coded in C, converts binary data, received from the FPGA via DMA, to a bitmap image file, which is embedded in a static hypertext markup Language (HTML) script. Fig. 7 shows a screenshot of the web page, using a cellphone browser.

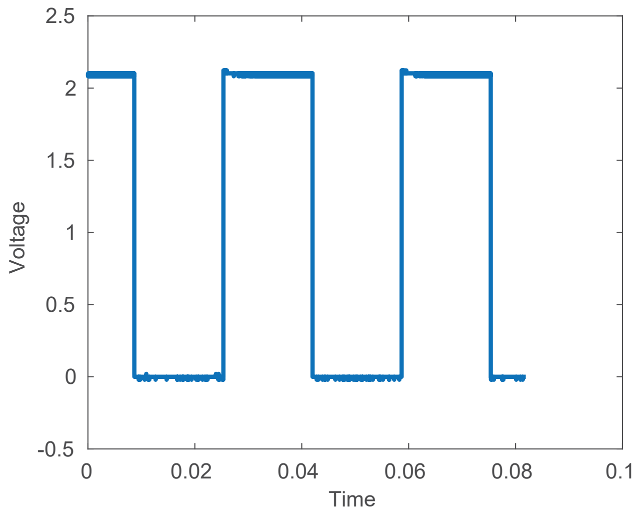
Using an oscilloscope, we verified that IRQs were generated at exactly 30 Hz. A trace is shown in Fig. 8. For this experiment, we made the μP busy from time to time, also shown in Fig. 8, while verifying that the FPGA still generated the IRQs at a constant 30 Hz, which enables hard real-time performance.

System Evaluation

To evaluate the complexity of an FPGA design, we have previously reported aggregate numbers of logic elements (LEs) and memory bits [4]. In this work, because we focus on a Xilinx device only and use the Vivado IDE, we report four parameters: logic and memory in LUTs; additional memory in blocks of RAM (BRAMs); additional memory in flip-flops (FFs); and additional logic in digital signal processing blocks (DSPs).

An overview of the occupancy is given in Fig. 9. This floorplan shows the layout of the ISP for HD resolution. Vertical narrow rectangles, for example, represent BRAMs instantiated for the TMO circuit. Placement of components may be affected by introducing additional constraints on the design. A breakdown of resource usage, as reported by Vivado, is given in Fig. 10.

In terms of the circuits being evaluated, we refer to all the auxiliary circuitry needed to support the DMA as “DMA aux,” which includes: the smart connectors to bridge HP AXI channels, between the FPGA and the μP; the inter-connectors for bridging GP AXI channels; and the reset circuits to reset and synchronize



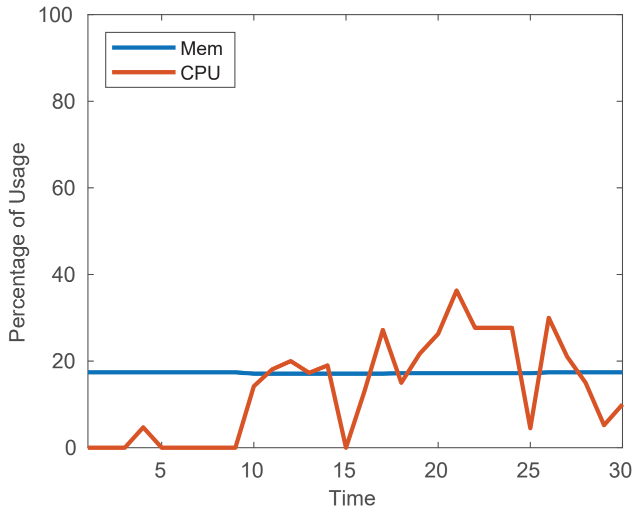


Figure 8. *IRQ voltage of the FPGA and usage of the µP. The IRQ rate, measured by an oscilloscope, is never disturbed by the µP load, measured by a shell script. Usage is varied by interleaving calculations and sleeping.*

data transfer, between the FPGA and the µP. These circuits were all inserted automatically by Vivado.

The controller circuit refers to all circuitry responsible for integrating the FPN correction, SPN filtering, and TMO circuits, as well as for interfacing the DMA using AXI4-Stream.

In Fig. 10, the number of LUTs primarily represents logic used, but some of it could represent memory. The total usage, primarily for the DMA and DMA aux circuits, is less than a quarter of the amount available. Especially considering the HD resolution of the ISP, all remaining circuits are highly efficient in terms of logic used. There are a total of 53,200 LUTs available.

The BRAMs represent most of the memory in the design. Xilinx embeds in their FPGAs configurable 36 Kb blocks of true dual-port RAM. Fig. 10 shows that our TMO consumes a significant number of BRAMs, but it is not much more than a quarter of the total available. The TMO relies on BRAMs to store histograms and mapping functions in ping-pong buffers. There are

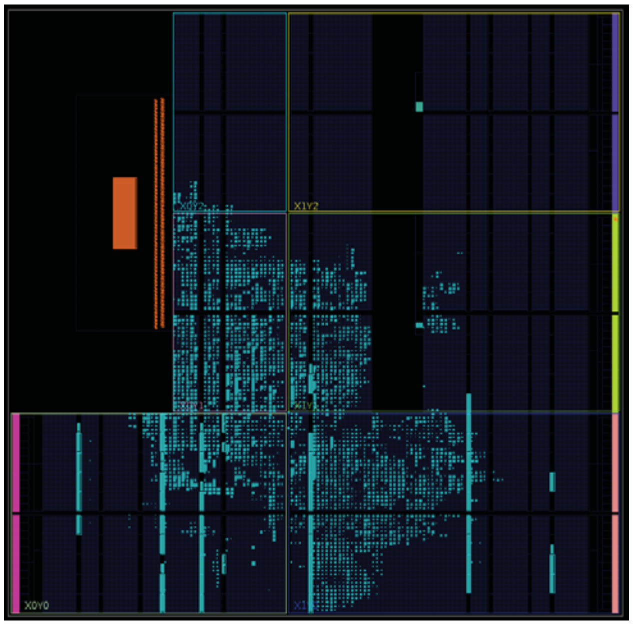


Figure 9. *Zoomed-out floorplan image of the ISP design. Using the Vivado IDE, designers can explore the complexity of their FPGA circuits. For this case study, plenty of resources are available for additional complexity.*

140 BRAM totalling 4.9 Mb available.

As with the LUTs, Fig. 10 shows that the DMA and DMA aux circuits are the ones that most use the FFs. Notwithstanding a router inside the SPN circuit, which is purely combinational logic, the FPN, SPN, and TMO circuits use highly-pipelined sequential logic, which requires FFs. However, as a fraction of the 106,400 FFs available, the amount used by these circuits is negligible.

The last resource depicted, in Fig. 10, is the number of DSPs. We can see here the influence of the FPN circuit, which is basically composed of adders and multipliers. The TMO also has arithmetic to calculate a mapping function. Only a small fraction of the 220 DSPs that are available is used.

In addition to unused resources on the FPGA side, there are unused resources on the µP side. Whereas IRQs are always generated at 30 Hz, if the ISR takes too long to execute, e.g., when saving an image unnecessarily to a disk, then some IRQs will not be served. However, given that the µP has two high-performance cores, there is plenty of scope for computation on its end.

Conclusion

This work achieved a SoC design flow for hard real-time image signal processing of a nonlinear HDR imaging system. Although algorithms and some digital circuits in the ISP are from previous work, this work integrates all of them into a single real-time platform. The design flow to implement such a system is presented and the system design is validated and evaluated.

Our ISP used 31.4% of BRAMs, 14.5% of LUTs, 9.0% of FFs, and some DSPs, available in a Xilinx Zynq-7000 SoC, to process HD video. Because the FPGA in the SoC is a low-cost device, the realized ISP design is therefore very efficient.

To interface the FPGA and the µP, IRQs and DMA were used. With a horizontal design flow, having ISR software in user space, we achieved 25 FPS at HD resolution. With a vertical de-

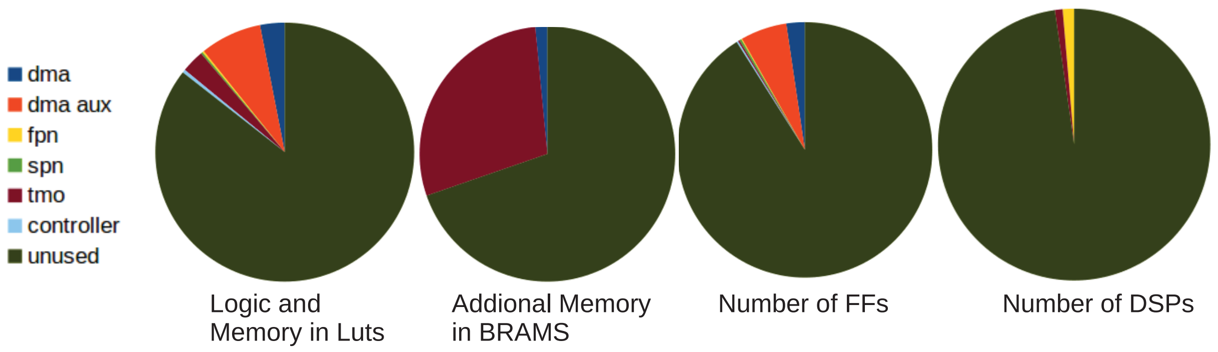


Figure 10. *Utilization of FPGA resources, at HD resolution, by the ISP. Relative to the DMA and TMO, few resources are used by the FPN, SPN, and controller circuits. Even considering BRAM memory, mainly used by buffers in the TMO, plenty of unused resources are available for additional processing.*

sign flow, using PetaLinux, we compiled and inserted a new driver in kernel space, updated missing packages, and implemented a DMA self test. We leave completion of the PetaLinux approach, which would support higher bandwidths, to future work.

One novelty of our design flow is that the FPGA is the master of the SoC platform, which includes a µP running Linux and will, in future, include a nonlinear CIS. Using this approach, our design can achieve hard real-time operation. At present, we simulated the nonlinear CIS using binary data loaded from a file, which consumed some of the bandwidth between the FPGA and µP. By freeing up this bandwidth, in future, we would be able to achieve the full 30 FPS required for HD video.

We implemented a new circuit to integrate the ISP to the µP via DMA. The main interface uses an AXI4-Stream bus and not video direct memory access (VDMA), because our goal was not a video processing pipeline with the end point being a display. Instead, our goal was to send the data to the µP for communication over a network, i.e., an edge computing case study.

Even taking simple modes of operation for the DMA, and AXI4-Stream for the protocol, a heterogeneous system requires mastery of multiple disciplines of knowledge. Working with embedded Linux is an advantage for its networking, open source tools, and standard OS functions. However, any new interface between the FPGA and µP has to be recognized by the Linux kernel. This means editing the Linux device tree, configuring the Linux kernel to recompile, and possibly, for better performance, creating a device driver in the kernel space. All of these tasks were investigated here in order to fully support HD video.

The proposed design flow is especially suited for future edge computing applications involving HDR imaging and computer vision. In addition to substantial unused logic and memory available on the FPGA, the µP was used here, beyond the ISR software, only to connect to a network and serve simple web content. Its dual-core ARM processor may be used for additional processing, by leveraging open-source frameworks for Linux.

References

[1] E. A. Lee, B. Hartmann, J. Kubiatowicz, T. S. Rosing, J. Wawrzynek, D. Wessel, J. Rabaey, K. Pister, A. Sangiovanni-Vincentelli, S. A. Seshia, D. Blaauw, P. Dutta, K. Fu, C. Guestrin, B. Taskar, R. Jafari, D. Jones, V. Kumar, R. Mangharam, G. J. Pappas, R. M. Murray, and A. Rowe, "The swarm at the edge of the cloud," *IEEE Design Test* **31**, pp. 8–20, June 2014.

[2] W. Shi and S. Dustdar, "The promise of edge computing," *Computer* **49**, pp. 78–81, May 2016.

[3] J. Li, A. Mahmoodi, and D. Joseph, "Using Polynomials to Simplify Fixed Pattern Noise and Photometric Correction of Logarithmic CMOS Image Sensors," *Sensors* **15**, pp. 26331–52, Oct. 2015.

[4] M. Nascimento, J. Li, and D. Joseph, "Digital Circuit Methods to Correct and Filter Noise of Nonlinear CMOS Image Sensors," *Journal of Imaging Science and Technology* **62**(6), pp. 60404–1–60404–14, 2018.

[5] J. Li, O. Skorka, K. Ranaweera, and D. Joseph, "Novel Real-Time Tone Mapping Operator for Noisy Logarithmic CMOS Image Sensors," *Journal of Imaging Science and Technology* **60**, pp. 020404–1–13, Mar. 2016.

[6] P. J. Lapray, B. Heyrman, M. Ross, and D. Ginhac, "A 1.3 megapixel fpga-based smart camera for high dynamic range real time video," in *Distributed Smart Cameras (ICDSC), 2013 Seventh International Conference on*, pp. 1–6, Oct 2013.

[7] S. Mann, R. C. H. Lo, K. Ovtcharov, S. Gu, D. Dai, C. Ngan, and T. Ai, "Realtime hdr (high dynamic range) video for eyetap wearable computers, fpga-based seeing aids, and glasseyes (eyetaps)," in *2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–6, April 2012.

[8] "www.myirtech.com." `http://www.myirtech.com/list.asp?id=502`. Accessed: 2019-02-04.

[9] T. Kalb, L. Kalms, D. Ghringer, C. Pons, F. Marty, A. Muddukrishna, M. Jahre, P. G. Kjeldsberg, B. Ruf, T. Schuchert, I. Tchouchenkov, C. Ehrenstrahle, F. Christensen, A. Paolillo, C. Lemer, G. Bernard, F. Duhem, and P. Millet, "Tulipp: Towards ubiquitous low-power image processing platforms," in *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pp. 306–311, July 2016.

[10] A. Mahmoodi, J. Li, and D. Joseph, "Digital pixel sensor array with logarithmic delta-sigma architecture," *Sensors* **13**(8), pp. 10765–10782, 2013.

[11] J. Kronander, S. Gustavson, G. Bonnet, and J. Unger, "Unified hdr reconstruction from raw cfa data," in *IEEE International Conference on Computational Photography (ICCP)*, pp. 1–9, April 2013.

[12] "Embedded Design Hub - PetaLinux Tools." `https://www.xilinx.com/support/documentation-navigation/design-hubs/dh0016-petalinux-tools-hub.html`. Accessed: 2019-02-28.

[13] Xilinx, *AXI DMA v7.1*, 04 2018. Vivado Design Suite.

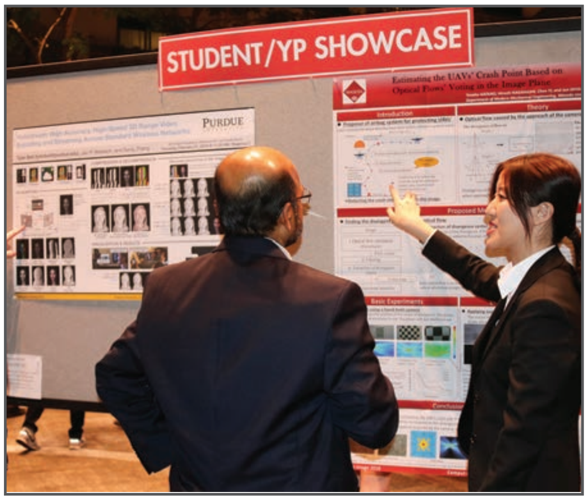
JOIN US AT THE NEXT EI!

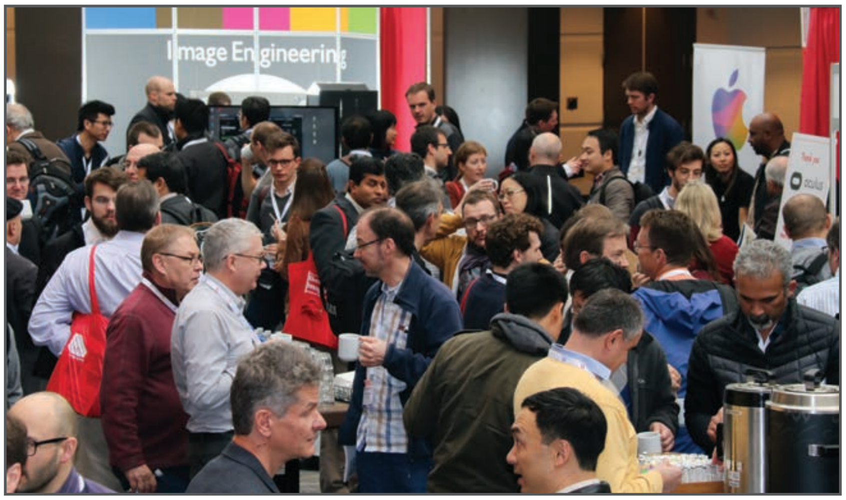
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





AR is a fundamental and transformative interface and platform.
20+ years from my initial working demo to today's AR displays
Optimistic that it won't take another 20 to reach ubiquitous consumer AR!
My first working AR prototype
Today's platforms
Ubiquitous consumer AR

• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org