Chapter 35
Efficient Low-Power Compact Hardware Units for Real-Time Image Processing

Khaldoon M. Mhaidat
Jordan University of Science and Technology, Jordan

Mohammad I. Alali
Jordan University of Science and Technology, Jordan

Inad A. Aljarrah
Jordan University of Science and Technology, Jordan

ABSTRACT

This paper presents efficient low-power compact hardware designs for common image processing functions including the median filter, smoothing filter, motion blurring, emboss filter, sharpening, Sobel, Roberts, and Canny edge detection. The designs were described in Verilog HDL. Xilinx ISE design suite was used for code simulation, synthesis, implementation, and chip programming. The designs were all evaluated in terms of speed, area (number of LUTs and registers), and power consumption. Post placement and routing (Post-PAR) results show that they need very small area and consume very little power while achieving good frame per second rate even for HDTV high resolution frames. This makes them suitable for real-time applications with stringent area and power budgets.

INTRODUCTION

Real-time image and video processing will always be essential in many applications (Chillet & Hübner, 2014; Sahlbach, Thiele, & Ernst, 2014; Baxes, 1994; Gonzalez & Woods, 2007; Nixon & Aguado, 2012). Examples include real-time medical imaging (Birk, Zapf, Balzer, Ruiter, & Becker, 2014), multimedia wireless sensor networks (Taysi, Yavuz, Guvensan, & Karsligil, 2014), automated surveillance (Ratnayake & Amer, 2014), pattern recognition (Brost, Yang, & Meunier, 2014), automated industrial visual inspection (Vega-Rodríguez, Sánchez-Pérez, & Gómez-Pulido, 2002), and high-definition video streaming (Hoffmann, Itagaki, Wood, & Bock,

DOI: 10.4018/978-1-4666-8789-9.ch035

2006; Elhamzi, Dubois, Miteran, & Atri, 2014). Therefore, there will always be a great need for efficient ways to perform image and video processing at real-time speed while satisfying sometimes stringent quality, area, and power requirements as in the case of embedded systems (Chillet & Hübner, 2014). For example, H.264 is one of the most popular video compression formats and requires a large amount of computations. Therefore, different architectures and techniques can be explored to come up with efficient implementations as in the H.264-compatible motion estimation accelerator presented in (Elhamzi, Dubois, Miteran, & Atri, 2014).

In general, there are four ways to implement image and video processing algorithms: Software running on general-purpose microprocessors/ CPUs or microcontrollers, software running on specialized graphics processing units (GPUs) or digital signal processors (DSPs), hardware implemented in field-programmable gate arrays (FPGAs), or hardware implemented in application-specific integrated circuits (ASICs; Nelson, 2000; Johnston, Gribbon, & Bailey, 2004; da Silva, et al., 2013; Ratnayake & Amer, 2014; Sahlbach, Thiele, & Ernst, 2014).

Image and video processing software will be slow when run on general-purpose processors and controllers because these devices are designed to handle many other functions or tasks and not designed specifically or solely for image and video processing. Moreover, they are generally serial (or sequential) in the way they run the code. Even a multi-core multi-threaded processor will not fully utilize or exploit all the parallelism inherent in image and video processing algorithms due to various reasons including memory and I/O sharing and threads communication overhead not to forget the serial nature of a the individual threads. GPUs will be faster than CPUs because they have been designed mainly for graphics processing but they will still be slower than a dedicated image and video processing hardware since they need to support a large set of general graphics processing functions.

Dedicated hardware that is designed specifically or solely for image and video processing can be implemented either in an ASIC chip or in an FPGA chip. Designers can use all kinds of optimization techniques before fabricating the ASIC chip to maximize speed, minimize area, and power consumption. However, this makes ASIC design and verification much more complex and expensive than FPGA. The advantages of the FPGA are coming from the fact of being reprogrammable. This means design can be easily fixed (debugged) or upgraded to handle more or different functionality. Current FPGAs are improving amazingly in speed, power consumption, functionality, and capacity, making them suitable for large and heavy or complex applications. For example, Xilinx Virtex UltraScale FPGA chips (Xilinx, 2014) have up to 4.4M logic cells at 20nm 3D IC technology. They also have a roadmap for the 16nm FinFET technology. FPGAs also lend themselves very well to real-time high-definition image and video processing because of their massively parallel structure (Johnston, Gribbon, & Bailey, 2004). In fact, they are great candidates for many applications these days including cryptography, information security, networking, communication, industrial control, signal processing, reconfigurable computing, and pattern recognition (Al-Khaleel, Idris, Mhaidat, & Aljarrah, 2013; Al-Khaleel, et al., 2011; Al-Khaleel, Tulic, & Mhaidat, 2012; Mhaidat, Altahat, & Al-Khaleel, 2013; Akoushideh, Shahbahrami, & Maybodi, 2014).

One good example of how FPGAs are much faster than CPUs can be found in (Al-Khaleel, Idris, Mhaidat, & Aljarrah, 2013). The character features extraction unit presented in (Al-Khaleel, Idris, Mhaidat, & Aljarrah, 2013) for OCR using Spartan-6 FPGA was more than 17000 times faster than the MATLAB software implementation presented in (Aljarrah, et al., 2012) and run on Pentium-4 processor with 3 GHz clock frequency and 1 GB RAM. Another good example is the texture features extraction in (Akoushideh, Shahbahrami,