Research Article

Zynq-Based Reconfigurable System for Real-Time Edge Detection of Noisy Video Sequences

Iljung Yoon, Heewon Joung, and Jooheung Lee

Department of Electronics and Computer Engineering, Hongik University, Seoul 04066, Republic of Korea

Correspondence should be addressed to Jooheung Lee; joolee@hongik.ac.kr

Received 25 December 2015; Accepted 10 July 2016

Academic Editor: Valerio Bellandi

Copyright © 2016 Iljung Yoon et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We implement Zynq-based self-reconfigurable system to perform real-time edge detection of 1080p video sequences. While object edge detection is a fundamental tool in computer vision, noises in the video frames negatively affect edge detection results significantly. Moreover, due to the high computational complexity of 1080p video filtering operations, hardware implementation on reconfigurable hardware fabric is necessary. Here, the proposed embedded system utilizes dynamic reconfiguration capability of Zynq SoC so that partial reconfiguration of different filter bitstreams is performed during run-time according to the detected noise density level in the incoming video frames. Pratt's Figure of Merit (PFOM) to evaluate the accuracy of edge detection is analyzed for various noise density levels, and we demonstrate that adaptive run-time reconfiguration of the proposed filter bitstreams significantly increases the accuracy of edge detection results while efficiently providing computing power to support real-time processing of 1080p video frames. Performance results on configuration time, CPU usage, and hardware resource utilization are also compared.

1. Introduction

Heterogeneous embedded systems have proliferated in the Internet of Things era, and stream-based applications for multimedia services are widely used in various types of portable devices. These applications are data-intensive and need high computing capability to meet the required throughput and stringent real-time constraints. On the other hand, low power consumption is also as important as throughput for these portable embedded systems since they operate often in the energy constrained environments [1].

Recently, Zynq SoC (System on Chip) platform including ARM dual-core Cortex-A9 processor with FPGA fabric has been used in the embedded systems (e.g., Advanced Driver Assistance System) to implement computationally complex signal processing algorithms by utilizing both of SW flexibility of ARM processor and parallel processing capability of reconfigurable hardware fabric [2]. For sensor processing and tracking, object classification, and assessment, edge detection is a fundamental tool in computer vision applications. However, 1080p-resolution video processing for

object edge extraction in real-time is highly time consuming and becomes computational bottleneck for ARM processors. Therefore, algorithm migration onto the FPGA fabric has been preferred to meet the performance requirements. The previous studies [3, 4] had designed drawback such as inflexibility of the implemented hardware architecture and lack of adaptation capability for time-varying incoming video signals while partial reconfiguration technique on FPGA can be highly desirable to solve these problems. In particular, as shown in the previous studies [5], corrupted images with unwanted salt-and-pepper noises caused by faulty memory cells, sensing error in the analog-to-digital conversion, or bit error in transmission degrade the performance of edge detection filters significantly.

In this paper, we implement the adaptive partial reconfigurable system to maximize the output performance of the implemented edge detection filter. By detection of noise density levels in the incoming video, adaptive self-reconfiguration of hardware bitstream is performed in real-time in order to remove unwanted noises before edge extraction process. For efficient utilization of hardware resources,

FIGURE 1: Zynq SoC architecture.

bitstream of Median filter with the following Sobel edge detection is also mapped to the same hardware region on FPGA.

The rest of this paper is organized as follows. Section 2 introduces Zynq SoC platform, and adaptive reconfiguration approach is proposed in Section 3. In Section 4, experimental results are discussed, and we conclude this paper in Section 5.

2. Zynq SoC Platform

Zynq-7000 AP SoC platform is hybrid FPGA platform combining high-performance ARM processor with FPGA fabric. Figure 1 shows Zynq internal architecture. It consists of Processing System (PS) and Programmable Logic (PL). The PS consists of ARM dual-core Cortex-A9 MP core, Caches, DMA controller, and various built-in peripherals such as USB, UART, SPI, CAN, and I2C. The PL attached to the PS through AMBA AXI ports possesses a number of hardware resources: Configurable Logic Blocks (CLB), Digital Signal Processing (DSP) Blocks, two 12-bit analog-to-digital converters, and serial transceivers [6].

Advanced Microcontroller Bus Architecture (AMBA) is used for the connection of functional blocks in a System on Chip (SoC) [7]. Particularly, Advanced eXtensible Interface (AXI) high-performance slave ports with configurable 32-bit or 64-bit data width and AXI general-purpose master/slave ports with 32-bit data width are available for PS-PL interface. Therefore, it ensures that user-defined functional blocks on programmable logic region can easily exchange data with one another and data can be transferred across the system [8].

Partial Reconfiguration (PR) plays a critical role for enhancing FPGA adaptability by allowing specific region

of the FPGA to be reconfigured dynamically with a new bitstream while the remainder of the FPGA continues to run. Specific regions called Partially Reconfigurable Regions (PRRs) are used to implement the bitstreams of Partial Reconfigurable Modules (PRMs). Throughout the design approach using partial reconfiguration, many advantages over traditional full configuration can be provided, such as reduction of hardware resource utilization and reconfiguration time overhead, increased scalability, reduced system downtime, and less storage size required. Additionally, reducing the size of the FPGA can lead to reduction of cost and power consumption [9]. Therefore, partial reconfiguration techniques have been widely studied in different domains to provide benefits such as enhanced quality, performance, and reliability of the systems [10–12].

In order to access FPGAs' configuration memory for partial reconfiguration, different types of interfaces, such as JTAG, SelectMAP, and ICAP (Internal Configuration Access Port), have been offered by Xilinx FPGAs. While external reconfiguration controller outside the FPGA is needed for JTAG and SelectMAP, ICAP enables internal access within the FPGA, supporting self-reconfiguration approaches. Therefore, ICAP interface has been widely used together with soft-IP processor (Xilinx Micro Blaze) or hard-IP processor (IBM PowerPC), and many studies on new interface for ICAP have been performed to enhance reconfiguration speed [13, 14] or reduce hardware resources required [15]. For Zynq SoC, additional interface, called Processor Configuration Access Port (PCAP), is provided to enable PS to configure PL region [16].

In this paper, partial reconfiguration of the proposed filter bitstreams is performed through 32-bit PCAP interface which is clocked at 100 MHz and can support up to 400 MB/s

FIGURE 2: PR interface using PCAP.

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

(a) Vertical operator, $S_1(k, l)$

(b) Horizontal operator, $S_2(k, l)$

FIGURE 3: Sobel operator.

download throughput. PR interface using PCAP is shown in Figure 2. First Stage Boot Loader (FSBL) read from external SD card boots up PS and configures the PL with the full bitstream via the PCAP, and user application loads the partial bitstream into DDR memory later on. From this moment, software-controlled partial reconfiguration is enabled to dynamically reconfigure part of the PL with the bitstream of preimplemented IP core [17].

3. Proposed Approach

For real-time object edge extraction of 1080p-resolution video frames, Sobel filter has been implemented on PL region. As an orthogonal gradient operator, Sobel operators shown in Figure 3 are used to perform 2-dimensional convolution in every pixel of the incoming video frame. $f(x, y)$ is the pixel value at location (x, y) [18]

$$G_1(x, y) = \sum_{k=-1}^1 \sum_{l=-1}^1 S_1(k, l) f(x+k, y+l), \quad (1)$$

$$G_2(x, y) = \sum_{k=-1}^1 \sum_{l=-1}^1 S_2(k, l) f(x+k, y+l).$$

The gradient vector magnitude and direction are given by

$$G(x, y) = \sqrt{G_1^2(x, y) + G_2^2(x, y)}, \quad (2)$$

$$\theta_G(x, y) = \tan^{-1} \frac{G_2(x, y)}{G_1(x, y)}.$$

Typically, salt-and-pepper noise is caused by defective sensors, faulty memory cells, and bit error in transmission, and it degrades the performance of edge detection filter significantly. In this paper, we implement the noise detection algorithm proposed in [19] and briefly describe it as follows.

$p(x, y)$ is a pixel value to be processed, and 3×3 window with $p(x, y)$ as the center location is considered. $p_{\min}$ and $p_{\max}$ are minimum and maximum pixel values of 3×3 window. Then, thresholds $T_{\min}$ and $T_{\max}$ are defined as

$$T_{\min} = \begin{cases} 2p_{\min} - 255, & 2p_{\min} > 255, \\ 0, & \text{otherwise,} \end{cases} \quad (3)$$

$$T_{\max} = \begin{cases} 2p_{\max}, & 2p_{\max} < 255, \\ 255, & \text{otherwise.} \end{cases}$$

Equation (4) is used as a criterion to determine whether $p(x, y)$ is a corrupted noise pixel or not

$$\text{noise}_{\text{det}} = \begin{cases} 1, & p(x, y) \leq T_{\min} \text{ or } p(x, y) \geq T_{\max}, \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

Then, the noise density is measured as the total number of detected noise pixels divided by the total number of pixels in a given video frame.

Since the edge detection performance decreases significantly in the corrupted image by the salt-and-pepper noise, Median filter is implemented for denoising as in

$$\widehat{M}(x, y) = \text{Median} \{M_{x', y'} : (x', y') \in W(x, y)\}. \quad (5)$$

Here, Median value of neighboring pixels in window W is selected as output [20].

The proposed self-reconfiguration method replaces PRMs to the Sobel edge detection after preprocessing Median operator (hereafter referred to as the Median + Sobel filter) which is effective for noise reduction when the salt-and-pepper noise is added to the video frame.

Pratt's Figure of Merit (FOM) to evaluate the accuracy of detected edge in noisy image is used to determine corresponding threshold of noise density level. As performance of edge detection accuracy is deteriorated, partial reconfiguration process is triggered to reduce noise before edge filtering. Pratt's FOM is defined by

$$R = \frac{1}{I_N} \sum_{i=1}^{I_D} \frac{1}{1 + \alpha d^2}. \quad (6)$$

Here, $I_N = \max(I_I, I_D)$, I_I is the number of edge points in the ideal edge, I_D is the number of edge points in the detected edge, α is a calibration constant, and d is the distance between the detected and the ideal edges [21]. The distance " d " is important factor in the evaluation of edge detection using PFOM. The factor " d " is inversely proportional to the factor R . For a stained edge, the distance " d " between ideal and detected edge increases and factor R is reduced.

FIGURE 4: Video pipeline and noise detection task.

Figure 4 shows video pipeline architecture and noise detection task. The 1080p video frames from HDMI-IN are stored in DDR memory. The implemented filter mainly consists of three subfunctions that are filtering process, edge detection process, and bus interface to control input and output of video. Video DMA (VDMA) reads video frames from DDR memory and sends them to the filter engine. The AXI4-Stream interfaces are connected through VDMA and AXI interconnect block to the high-performance slave port of the PS [22]. The output frame from the filter engine is stored back into DDR memory and then stored frame is sent to the display controller for HDMI-OUT. Synthesized results show that pipelined edge detection process has 9 clock cycles of latency, and total processing time requires 2,059 clock cycles.

Noise density level detection is performed to trigger partial reconfiguration of Median + Sobel filter. Partial bitstreams for filter operations are loaded from SD card into DDR memory by the user application running on the PS. It improves the reconfiguration time and also takes advantage of caching. Then, the application can use partial bitstreams to modify the partially reconfigurable region in PL without interrupting the rest of the PL area. Partial reconfiguration of Sobel or Median + Sobel bitstreams from DDR memory to the predefined PL region is performed through the PCAP interface. If the measured noise density becomes higher than the threshold, then Median + Sobel bitstream is configured to the partially reconfigurable region (PRR) to replace Sobel bitstream.

For Zynq SoC, an AXI-PCAP bridge consisting of “transmit” and “receive” FIFO buffers between the AXI and the PCAP interface is implemented in the Device Configuration interface (DevC) of the PS. This bridge converts 32-bit AXI formatted data to 32-bit PCAP protocol, and a DMA engine

FIGURE 5: Experimental environment.

transfers data between the FIFOs and the DDR memory for partial reconfiguration. A DevC driver function, built on top of sysfs, is called to move data across the PCAP interface through initiating the DMA transaction and then waits for an interrupt signaling that the transfer is completed. When both AXI and PCAP transfers are finished, then the function call returns. The application does not need to know about physical location of partially reconfigurable region because partial bitstream has the configuration frame addressing information. The filter in PL region is reset before transferring a partial bitstream via DevC/PCAP. When the bitstream transfer is completed, the reset is released and the configured filter is restarted with VDMA. Our measurements show that up to 5 frames of incoming video can be dropped during the partial reconfiguration process.

4. Experimental Results

Devices used in the experiment are ZC702 evaluation board with XC7Z020 AP SoC, FMC module equipped with HDMI input/output based on ADV7611/ADV7511, and 1920 × 1080 resolution monitor. ZC702 board is controlled through UART Terminal Emulator running on PC [23]. Figure 5 shows experimental setup for implementation of the proposed

FIGURE 6: The procedure of bitstream generation.

reconfigurable edge detection system. The Boot binary file booting the ZynqSoC consists of Zynq FSBL created in the SDK tool, full bit file generated in the Vivado, and U-boot called second stage boot loader. The compressed kernel image, that is, ulmage, supports linux operating system on the target board [24]. The partial bit files are initially stored in SD card and read to DDR memory for PS to perform PR through the PCAP interface. The target board utilizes uramdisk as the root file system. The Software Development Kit (SDK) tool is used to create linux application on the processor to perform the operation of the proposed method and partial reconfiguration.

In this paper, the proposed Sobel and Median + Sobel filter blocks are implemented by High-Level Synthesis (HLS) Tool [25, 26]. The HLS tool transforms C language, C++, and SystemC into a RTL implementation, and also offers the pipelining of the function through GUI interface.

Vivado Integrated Design Environment (IDE) is a development tool to provide Xilinx Integrated Synthesis Environment (ISE) and Xilinx Platform Studio (XPS). It is used to analyze and synthesize the HDL designs and perform timing analysis. Figure 6 shows the overall procedure of full and partial bitstream generation. The HDL design description of the system and the IP cores generated by the HLS tool are synthesized. Then, as shown in Figure 7, we floorplanned partially reconfigurable region so that hardware resources required for implementation of PRMs are less than 90% of the total amount of the PRR hardware resources. The hardware resource comparison of PRR and PRMs is summarized in Table 1.

As a result, one full bitstream and two partial bitstreams are generated. The PL system is initially configured with a full bitstream including static logic and Sobel filter. If detected noise density level becomes higher than threshold, partial bitstream of Median + Sobel is used to reconfigure the PRR

FIGURE 7: Static logic and PR modules.

TABLE 1: Comparison of PRR and PRMs resources.

Resources	PRR	PRMs	
	Available	Sobel	M + S
LUT	7800	2871 (36.8%)	3478 (44.6%)
SLICE	1400	1054 (75.3%)	1220 (87.1%)
RAMB18	40	3 (7.5%)	6 (15.0%)
DSPs	40	0 (0%)	0 (0%)

region. If its level becomes lower than threshold, partial bitstream of Sobel is read again from DDR memory for run-time reconfiguration.

In this paper, new reconfiguration interface called PCAP (Processor Configuration Access Port) available on Xilinx Zynq SoC is explored to perform partial reconfiguration of filter bitstreams using ARM Cortex-A9 processor. While theoretical speed of reconfiguration for 32-bit PCAP interface clocked at 100 MHz is 400 MB/s, practical reconfiguration speed is much lower due to the internal ARM interconnect architecture. Several examples using filter bitstream with JTAG, ICAP, and PCAP interfaces are shown in Table 2.

Due to the design approach using partial reconfiguration, we could achieve significant reduction of both bitstream file size and reconfiguration time through PCAP interface. As shown in Table 3, partial reconfiguration time is reduced to 12% of the full configuration time, and system downtime to replace the function of the proposed filter engine is not necessary any longer.

TABLE 2: Different configuration interfaces.

	[27]	[28]	[29]	[30]	Proposed
Partial bitstream size	242 KB	244 KB	100 KB	374 KB	460 KB
Partial reconfig. time	1 sec	460 ms	29 ms	6 ms	10 ms
Configuration interface	JTAG (Virtex-4)	ICAP (Virtex-4)	ICAP (Virtex-5)	PCAP (Zynq-7000)	PCAP (Zynq-7000)
Frequency (MHz)	6 MHz	100 MHz	100 MHz	100 MHz	100 MHz
Types of module	Median filter	Median filter	Electronic stethoscope	FPU	Median + Sobel filter

(a) Noisy video #1 (salt-and-pepper noise density: 20%)

(b) Noisy video #2 (salt-and-pepper noise density: 20%)

(c) Output of Sobel filter: original video #1

(d) Output of Sobel filter: original video #2

(e) Output of Sobel filter (noisy video #1)

(f) Output of Sobel filter (noisy video #2)

(g) Output of Median + Sobel filter (noisy video #1)

(h) Output of Median + Sobel filter (noisy video #2)

FIGURE 8: Comparison of edge detection results for the noisy video sequences.

Full HD video sequences of 1920×1080 resolution are used for experiments. Boat sequence (video #1) and Beauty sequence (video #2) are shown in Figure 8. Figures 8(a) and 8(b) show video sequences #1 and #2 corrupted by salt-and-pepper noise with 20% noise density level, respectively. Sobel

filter's outputs of original video sequences #1 and #2 are shown in Figures 8(c) and 8(d). Edges in the scene are clearly detected. In contrast, Figures 8(e) and 8(f) show that Sobel filter's performance for the noisy video sequences #1 and #2 degrades significantly, resulting in lower values of PFOMs.

FIGURE 9: PFOMs for Sobel and Median + Sobel filters.

TABLE 3: Bitstream size and configuration time.

	Full bitstream	Partial bitstream
Bitstream size	4,045,564 bytes	460,544 bytes
PCAP configuration time	83 ms	10 ms

As shown in Figures 8(g) and 8(h), Median + Sobel filter is highly effective for the noisy video sequences #1 and #2. Its edge detection results are significantly improved both subjectively and objectively. For objective evaluation of edge detection results, PFOM is used to compare the performance of Sobel and Median + Sobel filters [31, 32]. Video sequences #1 and #2 are corrupted by salt-and-pepper noise with 5% and 10% noise density level. Since Median operator removes the salt-and-pepper noise in the corrupted video frames, performance analysis of two filter engines shows that Median + Sobel filter provides about 14 to 20 times improvement of PFOM as indicated in Figure 9.

In Figure 10, frame rates supported by Sobel and Median + Sobel filters are indicated, and run-time CPU usage of hardware and software filter implementations is measured in Figure 11. While 100% of CPU power is used for S/W implementation of Sobel filter, frame rates drop significantly down to 1.5 fps, indicating software implementation is not suitable for real-time processing of 1080p video frames. Here, camera controller supports 60 input frames per second. Due to the additional computational complexity, H/W Median + Sobel filter supports up to 29 frames per second (fps), about 1 frame less than Sobel H/W implementation (30 frames per second).

After PR, the power consumption of hardware platform on Xilinx Zynq FPGA is estimated using Power Report in Vivado Design Suite [33].

As shown in Figure 12, the power consumption of Median + Sobel filter is higher than Sobel filter because Median + Sobel filter requires more hardware resources in FPGA than Sobel filter.

FIGURE 10: Comparison of frame rates.

FIGURE 11: CPU Usage.

FIGURE 12: Comparison of power consumption.

5. Conclusion

In this paper, we propose adaptive partial reconfigurable system to maximize the output performance of the implemented edge detection filter. Hardware implementation of filter engine onto the FPGA fabric provides computing capability of real-time edge detection of 1080p video sequences.

According to detection of noise density levels in the incoming video, adaptive self-reconfiguration of hardware bitstream is performed during run-time and it enables significant performance improvement in both subjective and objective results. Experimental results show that partial reconfiguration time is reduced to 12% of the full configuration time, and about 14 to 20 times improvement of PFOM is achieved.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (NRF-2012R1A1A1008674) and also by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration (C03319350100437169) and 2014 Hongik University Research Fund. Additionally, the authors would like to acknowledge Xilinx and the Xilinx University Program for its generous donation of S/W design tools and H/W boards.

References

- [1] A. Majumdar, S. Cadambi, and S. T. Chakradhar, "An energy-efficient heterogeneous system for embedded learning and classification," *IEEE Embedded Systems Letters*, vol. 3, no. 1, pp. 42–45, 2011.
- [2] UG1165 (v2015.3), *Zynq-7000 All Programmable SoC: Embedded Design Tutorial*, Xilinx, November 2015.
- [3] D. Crookes, K. Benkrid, A. Bouridane, K. Aiotabi, and A. Benkrid, "Design and implementation of a high level programming environment for FPGA-based image processing," *IEE Proceedings: Vision, Image and Signal Processing*, vol. 147, no. 4, pp. 377–384, 2000.
- [4] P. Greisen, M. Runo, P. Guillet et al., "Evaluation and FPGA implementation of sparse linear solvers for video processing applications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 8, pp. 1402–1407, 2013.
- [5] A. M. Mahmood, H. H. Maraş, and E. Elbaşı, "Measurement of edge detection algorithms in clean and noisy environment," in *Proceedings of the 8th IEEE International Conference on Application of Information and Communication Technologies (AICT '14)*, pp. 1–6, Astana, Kazakhstan, October 2014.
- [6] Xilinx, *Zynq-7000 All Programmable SoC Overview*, DS190(v1.8), Xilinx, 2015.
- [7] UG585(v1.10), *Zynq-7000 All Programmable SoC Technical Reference Manual*, Xilinx, February 2015.
- [8] J. Silva, V. Sklyarov, and I. Skliarova, "Comparison of on-chip communications in Zynq-7000 all programmable systems-on-chip," *IEEE Embedded Systems Letters*, vol. 7, no. 1, pp. 31–34, 2015.
- [9] Xilinx, *Vivado Design Suite User Guide Partial Reconfiguration*, UG909(v2014.4), Xilinx, 2014.
- [10] E. Stott, P. Sedcole, and P. Y. K. Cheung, "Fault tolerant methods for reliability in FPGAs," in *Proceedings of the International*

- Conference on Field Programmable Logic and Applications (FPL '08)*, pp. 415–420, Heidelberg, Germany, September 2008.
- [11] C. Insaurralde, “Reconfigurable computer architectures for dynamically adaptable avionics systems,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 30, no. 9, pp. 46–53, 2015.
 - [12] M. G. Parris, C. A. Sharma, and R. F. Demara, “Progress in autonomous fault recovery of Field Programmable Gate Arrays,” *ACM Computing Surveys*, vol. 43, no. 4, article 31, 2011.
 - [13] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch, “Run—time partial reconfiguration speed investigation and architectural design space exploration,” in *Proceedings of the International Conference on Field Programmable Logic and Application*, pp. 498–502, Prague, Czech Republic, September 2009.
 - [14] R. Bonamy, H.-M. Pham, S. Pillement, and D. Chillet, “UPaRC—Ultra-fast power-aware reconfiguration controller,” in *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE '12)*, pp. 1373–1378, IEEE, Dresden, Germany, March 2012.
 - [15] M. Hübner, D. Göhringer, J. Noguera, and J. Becker, “Fast dynamic and partial reconfiguration data path with low hardware overhead on Xilinx FPGAs,” in *Proceedings of the IEEE International Symposium on Parallel & Distributed Processing*, pp. 1–8, April 2010.
 - [16] K. Vipin and S. A. Fahmy, “Zycap: efficient partial reconfiguration management on the xilinx zynq,” *IEEE Embedded Systems Letters*, vol. 6, no. 3, pp. 41–44, 2014.
 - [17] XAPP1159(v1.0) and C. Kohn, *Partial Reconfiguration of a Hardware Accelerator on Zynq-7000 All Programmable SoC Devices*, Xilinx, January 2013.
 - [18] S. Jin, W. Kim, and J. Jeong, “Fine directional de-interlacing algorithm using modified Sobel operation,” *IEEE Transactions on Consumer Electronics*, vol. 54, no. 2, pp. 857–862, 2008.
 - [19] P.-Y. Chen, C.-Y. Lien, and Y.-M. Lin, “A real-time image denoising chip,” in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '08)*, pp. 3390–3393, Seattle, Wash, USA, May 2008.
 - [20] C. Chen, J. Ni, and J. Huang, “Blind detection of median filtering in digital images: a difference domain based approach,” *IEEE Transactions on Image Processing*, vol. 22, no. 12, pp. 4699–4710, 2013.
 - [21] W. K. Pratt, *Digital Image Processing*, PIKS Inside, 3rd edition, 2000.
 - [22] UG1037 (v3.0), *Vivado Design Suite : AXI Reference Guide*, Xilinx, June 2015.
 - [23] C. Kohn, *Partial Reconfiguration of a Hardware Accelerator with Vivado Design Suite for Zynq-7000 ApSoC Processor*, XAPP1231 (v1.1), Xilinx, 2015.
 - [24] UG821 (v12.0), *Zynq-7000 All Programmable SoC Software Developers Guide*, Xilinx, San Jose, Calif, USA, 2015.
 - [25] F. M. Vallina, C. Kohn, and P. Joshi, *Zynq All Programmable SoC Sobel Filter Implementation Using the Vivado HLS Tool*, XAPP890 (v1.0), Xilinx, 2012.
 - [26] Xilinx, *Vivado Design Suite User Guide: High-Level Synthesis*, UG902 (v2015.4), Xilinx, 2015.
 - [27] S. U. Bhandari, S. Subbaraman, S. S. Pujari, and R. Mahajan, “Real time video processing on FPGA using on the fly partial reconfiguration,” in *Proceedings of the International Conference on Signal Processing Systems (ICSPS '09)*, pp. 244–247, Singapore, May 2009.
 - [28] S. U. Bhandari, S. Subbaraman, S. Pujari, and R. Mahajan, “Internal dynamic partial reconfiguration for real time signal processing on FPGA,” *Indian Journal of Science and Technology*, vol. 3, no. 4, pp. 365–368, 2010.
 - [29] A. A. Prince and S. Mishra, “Multi-mode electronic stethoscope implementation and evaluation using Dynamic Reconfigurable Design,” in *Proceedings of the 5th IEEE International Advance Computing Conference (IACC '15)*, pp. 228–232, Bangalore, India, June 2015.
 - [30] M. Al Kadi, P. Rudolph, D. Göhringer, and M. Hübner, “Dynamic and partial reconfiguration of Zynq 7000 under Linux,” in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig '13)*, pp. 1–5, IEEE, Cancun, Mexico, December 2013.
 - [31] I. E. Abdou and W. K. Pratt, “Quantitative design and evaluation of enhancement/thresholding edge detectors,” *Proceedings of the IEEE*, vol. 67, no. 5, pp. 753–763, 1979.
 - [32] J.-A. Jiang, C.-L. Chuang, Y.-L. Lu, and C.-S. Fahn, “Mathematical-morphology-based edge detectors for detection of thin edges in low-contrast regions,” *IET Image Processing*, vol. 1, no. 3, pp. 269–277, 2007.
 - [33] Xilinx, *Vivado Design Suite User Guide: Power Analysis and Optimization*, UG907(v2015.4), Xilinx, 2015.