A Dynamic Partial Reconfiguration Design for Camera systems

Jalal Khalifat, Ali Ebrahim, Adewale Adetomi and Tughrul Arslan
System Level Integration Group, Institute for Integrated Micro and Nano Systems
University of Edinburgh
Edinburgh EH9 3JL, Scotland, UK
Email: {J.Khalifat, A.Ebrahim, a.adetomi, T.Arslan }@ed.ac.uk

Abstract-- **The image-processing pipeline is the core part of any camera system including digital still cameras, camcorders, camera phones and video surveillance equipments. The image-processing pipeline consists of a number of processing stages that enhance the image or remove any effects that are caused by surrounding conditions. These stages are computationally intensive and need special requirements to meet the real time processing. This paper discusses the pipeline parts and presents a high-performance and cost-effective implementation of the pipeline on Field Programmable Gate Arrays (FPGAs) using Dynamic Partial Reconfiguration (DPR) feature to exploit the FPGA resources over time and space. The paper shows that the implemented system adds much of flexibility to camera systems by using a reconfigurable region. The system can use an unlimited number of image processing pipeline stages to process the images without the need of huge number of logic resources to fit all the stages. Moreover, the stages are not fixed in this system, they can be changed upon the user's decision .The architecture is designed to process still images of size 1920x1080. Each stage could process a full frame within 7.25 ms. A fast configuration engine is designed and deployed in the system. The engine shows that it can outperform the engine provided with zynq SoC by three times. The overall throughput of the system reaches 250 Megapixel/s.**

Keywords—Image processing pipeline ; DPR; ICAPE2; Real time processing; Camera ;

I. INTRODUCTION

With the increasing number of smart devices being used in our daily life, capturing our life moments has been becoming an essential part of our life style. Therefore, most of the companies try to embed a digital camera in their smart devices. The image processing systems in general, require a high computation power; therefore, the structure of these systems should fulfill specific requirements of power consumption, performance and throughput to be a valid image processing system. Traditionally, these systems are built using Application Specific Integrated Circuit (ASIC) due its low power consumption, high-speed optimization and low unit cost. Nowadays, Field Programmable gate arrays (FPGA) appeared to be a competitive environment in this field. Modern FPGAs have number of features that could make them a good choice for such systems such as low power consumption compared to the old generations, high speed, fast time to market and the most important feature is their flexibility. The term flexibility refers to a number of aspects in FPGAs and the main aspect is the Reprogrammability [1], which enables you to adapt to changing standards and support design reuse. Moreover, FPGAs provide additional flexibility to the designers by using the Dynamic Partial Reconfiguration (DPR) feature which exists in modern FPGAs. DPR is the feature, which allows changing the functionality of part of the design while the rest of the design is working. This feature can reduce the needed resources for a specific system and the consumed power. Accessing the FPGA's configuration memory is performed internally through the Internal Configuration Access Port (ICAP).

The operations in digital cameras can be divided into 3 main parts: the input, the processing and the storage/output parts. The input part captures the image data using sensors. The processing part performs the baseline and enhanced image processing on the raw data produced by the camera sensors to make them look like what we see using our eyes. This part is usually called image-processing pipeline. The Final part is the storage and output part, which either saves the result in non-volatile memory or shows it on the screen. Obviously, the image-processing pipeline is the core part of any digital camera and decides the quality of its images. Moreover, it requires a lot of computation power as it consists of number of complex algorithms working sequentially. The critical issue in building this part is the timings demands. Therefore, it is important to be built in high performance environment such as ASIC, Digital signal processors (DSP) or FPGAs but not using General Purpose processors (GPP).

There are a lot of research discussed in this field but most of them are focused in the traditional way of building such system. With the appearance of Modern FPGAs, a number of research has discussed developing such system on FPGAs.

In [2], Texas instruments developed a full commercial imaging System on Chip (SoC), which consists of ARM processor and DSPs and application specific hardware. The processing pipeline implemented in a configurable manner to combine the performance advantage from the hardware with flexible control to fine-tuning of the algorithms parameters. Xilinx has built imaging processing pipeline [3] on 7000 All programmable SoC in a configurable manner to adjust the algorithm parameters on the fly either manually by the user or automatically based on statistical information gathered from the frames data. Moreover, number of research such as [4]-[7] focused on implementing individual parts of the image processing pipeline on reconfigurable platforms and evaluate its performance, resource utilization and efficiency in general.

This paper presents an implementation for an image-processing pipeline on FPGAs, which uses DPR feature to increase the possibility of using maximum number of imaging enhancement stages. Moreover, it shows how the implementation utilises the resources, and reduces the power consumption. It also presents a brief introduction of the image processing pipeline stages. Finally, it contains implementation results and comparison to other similar designs.

The rest of the paper is organized as follows. In section II, we provide some background information on the main elements of a typical image processing pipeline components based on different conventional processors. In section III, core design tools are introduced and examples of the design process are given. In section IV, we show the architecture of the system, the design components and a designed configuration engine. In section V, we show the experimental results. Finally, section VI concludes the paper.

II. IMAGE PROCESSING PIPELINE

The data captured by the camera sensors should match the scene seen by our eyes. Unfortunately, this does not happen. Therefore, a number of image enhancement and constructing stages are required to adjust the scene data to look similar to the scene seen by the eyes. In this context, different companies use its own image-processing pipeline. Fig.1 shows the image processing pipeline is used in Texas Instruments [2] based on DaVinci technology. A typical image-processing pipeline consists of a number of stages as follows:

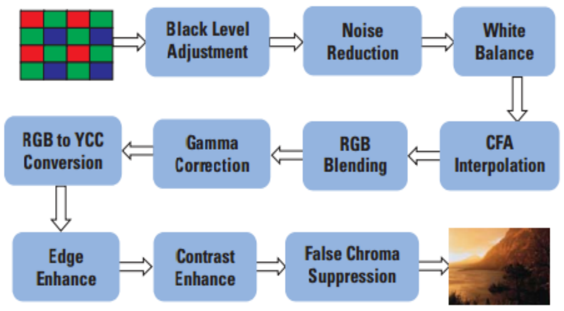


Figure 1. DaVinci imaging-processing pipeline [9].

A. *Color Interpolation*

In digital cameras, the charge-coupled device (CCD) or complementary metal oxide semiconductor (CMOS) image sensors are covered with a Color Filter Array (CFA). Each sensor is overlaid with one type of color filter. This arrangement results in an incomplete image samples with two missing colors from each pixel as shown in the case of (RGGB) arrangement of the Bayer color filter array in Fig.2. Therefore, a color interpolation algorithm is needed to construct the full image and predict the missing colors in each pixel. A lot of color interpolation algorithms are presented in the literature such as [8] and [9].

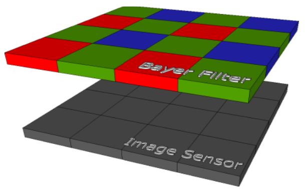


Figure. 2. Bayer Filters and Image sensors.

B. *Automatic White Balance (AWB)*

Automatic white balance is the process of keeping the color of objects constant automatically under different illumination conditions (light sources) by calculating a number of parameters from the image data [10]. These parameters are used to change the image pixel values to keep the color constant. A number of algorithms have been proposed in the literature such as Gray World Assumption (GWA) [11], Perfect Reflector Assumption (PRA) [12], Retinex theory, standard deviation-weighted gray world and Gamut mapping method [13] and others.

C. *Gamma Correction*

It is the adjustments applied during the display of a digital representation of color on a screen in order to compensate for the fact that the e-cathode ray tubes used in monitors in general produce light intensity, which is not proportional to the input voltage [2]. Therefore, if the image has to be displayed on a screen, linear RGB components should be converted to a non-linear signal. Gamma Correction controls the overall brightness of the image frames, as they usually appear too dark on screen. The parameters used in gamma correction depend on the characteristics of the display.

D. *Color Correction*

The images captured by digital cameras are affected by many environmental contents like illuminations or the object's color properties. Because of that, we need to map the captured images data to the device color space. This can be done using a color compensation chart [2]. Many algorithms have been proposed in the literature dealing with this reproduction image processing pipeline stage.

E. *Other stages*

There are number of other components of the image processing pipeline such as noise reduction, RGB to YCC Conversion, Edge enhancement and contrast enhancement. Each of these stages enhances the frame in some way and removes any effects that are caused by the surrounding conditions.

In addition, the compression stage is a post-processing stage, which is used to reduce the total image data for storage purpose. There are number of compression standards widely used but the main one, which is used in digital cameras is the JPEG standard.

All the aforementioned components need an intensive computation power to meet the real time processing which general-purpose processors cannot provide. Therefore, these components traditionally are implemented in hardware environments.

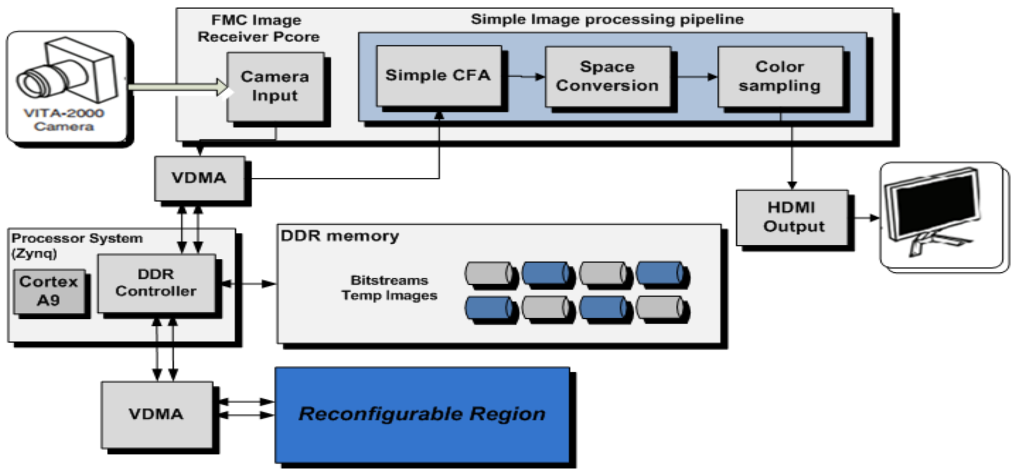


Figure 3. Camera System Implementation

III. Development Tools

The following tools have been used in the development of this system:

A. *Vivado Integrated Design Environment(IDE).*

Part of the Vivado design suite. It is the new generation of Xilinx development tool. It is replacing the Xilinx Integrated Synthesis Environment (ISE) and Xilinx platform studio (XPS). It is mainly used as a solution for designing embedded processing systems. Moreover, it is used in synthesis and analysis of HDL designs, perform timing analysis and examining RTL designs.

B. *Vivado High Level Synthsis (HLS)*

The tool accelerates the IP creation by enabling C, C++ and system C to be targeted into xilinx devices without the need to create RTL manually using Hardware Description Language (HDL). This tool has been used to develop the image processing pipeline components.

C. *Software development kit (SDK)*

The tool is used as an environment for creating embedded applications on the embedded processors such the ones exist in the Zynq SOC and on the industry leading MicroBlaze. The C developed software is used to control the design and the DPR swapping process.

The steps of developing the system based on the aforementioned tools as follow:

1) Develop the image processing pipeline components in C language individually and let the vivado HLS converts the codes to xilinx IPs that can be used directly in the system.

2) Develop the full system using Vivado IDE with only one component of the image processing pipeline.

3) Create application for the embedded design using the drivers provided by each IP in the design. The application should run on the vivado SDK.

4) Apply dynamic partial reconfiguration on the part that holds the image processing pipeline component.

IV. Camera System Implementation

The proposed implementation is illustrated in Fig. 3. The design exploits the feature of DPR to reduce the resources and power consumption. The system is divided into two parts: the static part and reconfigurable part. The static part contains a simple processing pipeline mainly to allow the user to capture a specific scene. Also it contains the FMC-IMAGEON VITA Receiver core [14][15], HDMI output interface and the IPs which are used in data communication between the modules and memory such as Video Direct Memory Access (VDMA).The reconfigurable part contains the part of the design, which can be replaced regularly to change the system functionality (image processing pipeline stages).

A. *Static design.*

In the static design, the simple processing pipeline consists of color filter array interpolation, color space converter and Chroma resampler Xilinx IP cores. The cores are used only for showing the data received by the design in the output side to let the user capture the desired scene. Also in the static design, FMC IMAGEON VITA Receiver core controls and synchronizes the stream of data from the image sensors and the user design. The HDMI output core controls the flow of data with the HDMI interface. The VDMA is used to redirect the captured data to the DDR memory and return it back to the design. This action allows us to process any frame of the data once the user presses the capture button. At that moment, the system will keep the last frame in the buffer.

B. *Reconfigurable design.*

The reconfigurable part consists of the image processing pipeline stage. Only one stage exists in the design at a time. The stages are swapped sequentially by using the DPR feature. To configure the stage dynamically, the current configured stage should be completely isolated from the system before configuring the new stage otherwise; an unknown data will be sent to/from the reconfigurable region during the configuration process, which leads to hang the entire system. The user should reset the Reconfigurable Module (RM) itself and disconnect any data bus is connected to the RM specifically the AXI bus.

Fig.4 shows the way of isolating the RM using user signals. These signals are controlled from the user application.

The output Res[0:0] is the isolation signal which prohibits the M00_AXI bus to send or receive data from the reconfigurable IP (CFA_DPR32_0). Moreover the gpio_io[0:0] will reset the RM before any configuration attempt. These signals will be released after the configuration process.

Three stages of the image-processing pipeline have been implemented using Vivado HLS tool: Demosaicing, Automatic white balance and noise reduction. Demosaicing stage has been implemented based on Adams-Hamilton interpolation algorithm [8]. The automatic white balance stage is divided into two stages to reduce the resources utilisation and the implementation is based on gray world and retinex theory [16]. The noise reduction stage is based on 3x3median filter.

In Zynq boards, the Processor Configuration Access port (PCAP) is used to configure the full or partial bitstreams. The PCAP configuration path is shown in Fig.5. Although the configuration primitive provided in the FPGA can support up to 400 MB/s configuration throughput, the transfer rate through the PCAP is approximately 145 MB/s [17]. The overall throughput is limited by the PS AXI interconnect. Therefore, this limitation could cause some problems for complex designs, which have large reconfigurable regions. To overcome this limitation, a fast configuration engine has been designed and deployed to increase the configuration speed and to utilise the full speed of the ICAP primitive. The configuration engine consists of a Direct Memory Access (DMA) to move data from the memory to the ICAP primitive in addition to the ICAP interface. The ICAP interface is a direct interface. The valid data signal is connected to the ICAP enable signal in a write mode as the new generation of ICAP in 7-series devices has no busy signal [18]. Fig.6 shows the designed configuration engine diagram. The DMA IP is configured with max burst option to increase the overall throughput. The Configuration engine is only 847 LUT and utilises only 1.5% of the programmable logic in Zynq 7020 SoC.

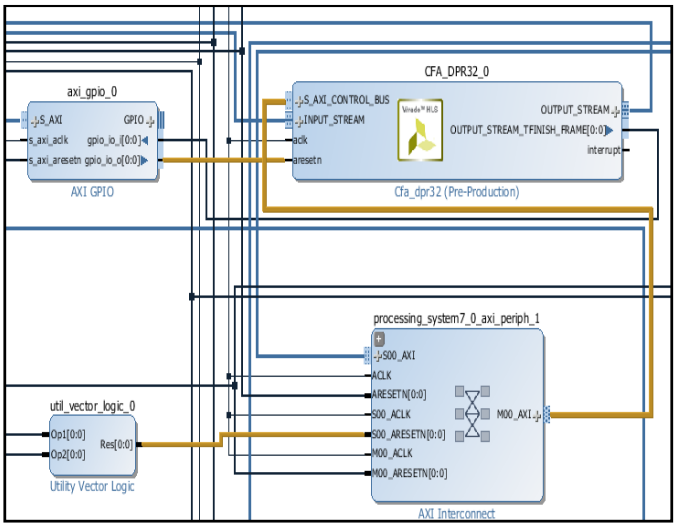


Figure 4. Isolation process of the Reconfigurable module

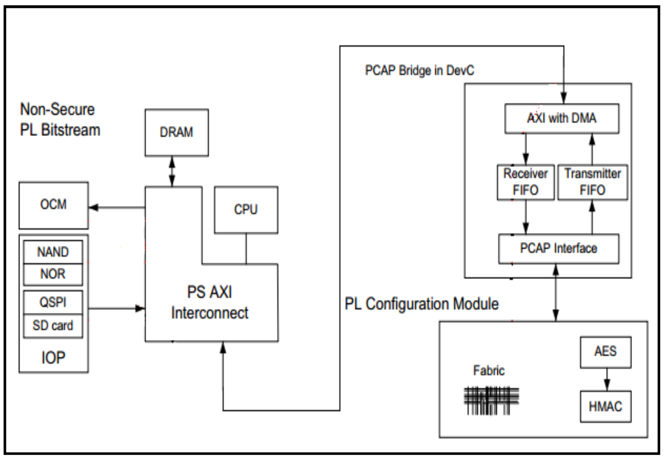


Figure 5. PCAP configuration path [17].

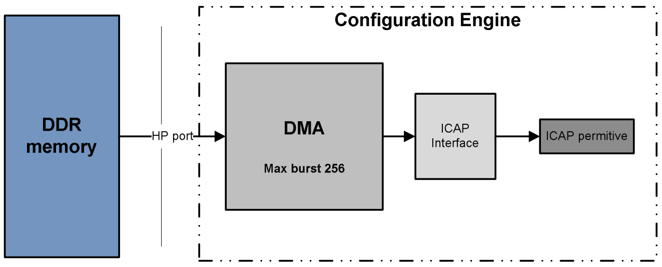


Figure 6. Configuration Engine.

Fig.6 shows the designed configuration engine diagram. The DMA IP is configured with max burst option to increase the overall throughput. The Configuration engine is only 847 LUT and utilises only 1.5% of the programmable logic in Zynq 7020 SoC.

V. Experimental Results

The system has been implemented on Zynq-7020 SoC evaluation board. The system has three different frequencies, 200 MHz for the IMAGEON VITA Receiver core, 150 MHz for the image-processing pipeline and 100 Mhz for the configuration engine as this is the maximum operational frequency for the ICAP primitive. Moreover, the 100 MHz is used to synchronise the AXI control buses with Processor. Two different sets of reconfigurable modules have been implemented; the first set processes one pixel per clock cycle to get a throughput around 125MPixels/s. On other hand, the second set of modules processes two pixels per clock cycles, which leads to achieve a double throughput compared to the first set.

The reconfigurable region size depends on the resources utilized by the largest stage among the image-processing pipeline stages. In this design, the reconfigurable region is approximately 15% of the total chip area if the first set of modules are used. Fig.7 shows the floor planning of the implemented system and Table 1 shows the resources utilization for different IPP stages for the two set of modules. The system has been tested with only three different stages. The reconfigurable region should have 20% more resources compared to the largest stage for routing purposes. All the stages have been implemented using Vivado HLS tool.

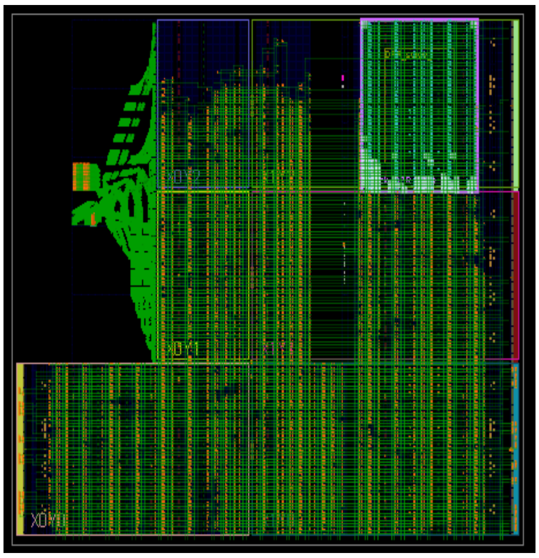


Figure 7. Implemented system floorplanning.

TABLE I. RESOURCES UTILIZATION OF THE RECONFIGURABLE MODULES

IPP stage	Resources (set 1/set 2)			
	LUT (1/2)	*RAMB18 (1/2)*	*DSP48 (1/2)*	*Slices (1/2)*
Color Filter array Interpolation + Median Filter	5379 / 7420	23 / 23	0 / 0	1390 / 1802
Automatoc White Balance part1	4230 / 5123	0 / 0	13 / 15	1105 / 1312
Automatoc White Balance part2	5802 / 7211	0 / 0	6 / 12	1450 / 1723
Reconfigurable Rrgion	6400 / 8000	40 / 40	20 / 40	1600 / 2000

Based on Table I different sizes of reconfigurable regions provide different Bitstream sizes. Therefore, the needed configuration time will vary from size to size. Table II shows the size of the partial bitstreams for each set of the reconfigurable modules and the configuration time for each bitstream using the two different methods. Bigger Bitsreams need more time to be configured. The designed configuration engine will reduce the gap band, which will increase the system performance, as the total needed time for configuring a stage will be reduced. The configuration engine is three times faster than the one, which is provided with Zynq SoC.

TABLE II. SIZE OF THE BITSTREAMS OF RECONFIGURABLE MODULES AND CONFIGURATION TIME

Design	Size of bitstream (KB)	Configuration time (ms)	
		PCAP	*Configuration Engine*
RMs Set 1 (1ppc)	436 KB	3.54	1.11
RMs Set 2 (2ppc)	581 KB	4.72	1.47

In term of the total time, Fig.8 shows the timing for the system execution path. Obviously, it shows that the overhead in the system is the time for configuring a new stage. Table III. shows the total execution time for the system and how the designed configuration engine reduces the overhead in the configuration. This overhead is proportional with the number of IPP stages. As shown in Table III, a good improvement has been achieved when the stages are bigger and consume more logic resources.

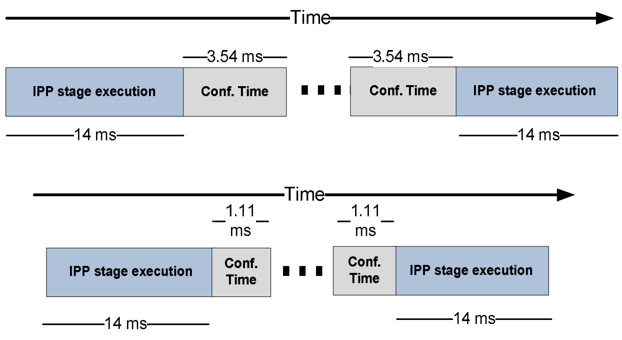


Figure 8. The System total execution time and configuration time improvement.

As mentioned previously, the system is designed in a DPR manner as only one of the stages exists on the FPGA logic at a time. This method allows unlimited number of stages to be used dynamically. On the other hand, the static design will allow using a few number of stages as the FPGA logic is not enough to fit all the stages at once. As a result, the total FPGA resources which would have been used to design the system with DPR is around 80% of the total resources. Therefore, it would not have been possible to implement the system as a static design with all the possible stages. Table IV shows the resources utilization for the DPR implementation. Not all the resources are fully utilized except the LUTs.

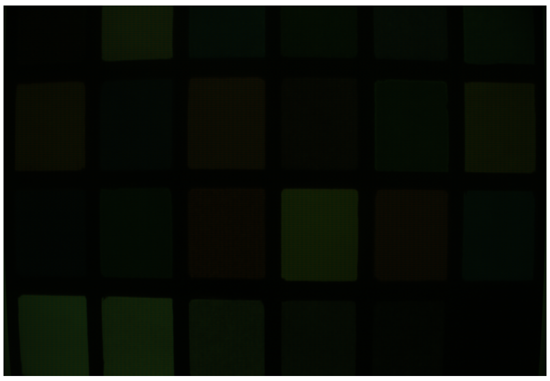
TABLE III. EXECUTION TIME OF THE CAMERA SYSTEM FOR NUMBER OF IPP STAGES

IPP stage	Time (ms)			
	Set 1 (1 ppc)		*Set 2 (2 ppc)*	
	PCAP	*Conf. Engine*	*PCAP*	*Conf. Engine*
CFA Interpolation + Median Filter	14.5	14.5	7.25	7.25
Configure next stage	3.54	1.11	4.72	1.47
Automatoc White Balance part1	14.5	14.5	7.25	7.25
Configure next stage	3.54	1.11	4.72	1.47
Automatoc White Balance part2	14.5	14.5	7.25	7.25
Total Time	50.58	45.72	31.19	24.69
Improvement	9.6%		20.8%	

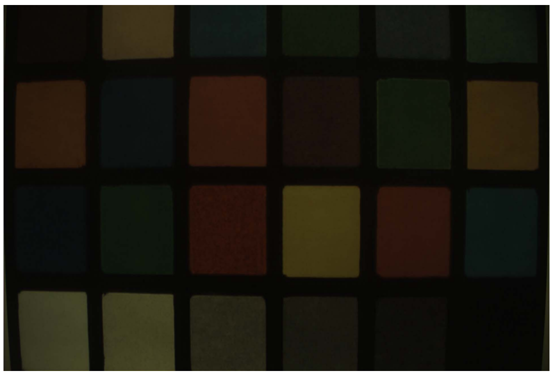
TABLE IV. RESORCES UTILIZATION OF THE IMPLEMENTED DESIGN

Design	Resources / Utilization			
	LUT	*Slices*	*RAMB*	*DSP*
DPR- (1 ppc)	39526 / 78%	11431 / 85.9%	43 / 30.7%	56 / 26%
DPR- (2 ppc)	41632 / 80%	11835 / 88.9%	48 / 34.3%	56/ 26%

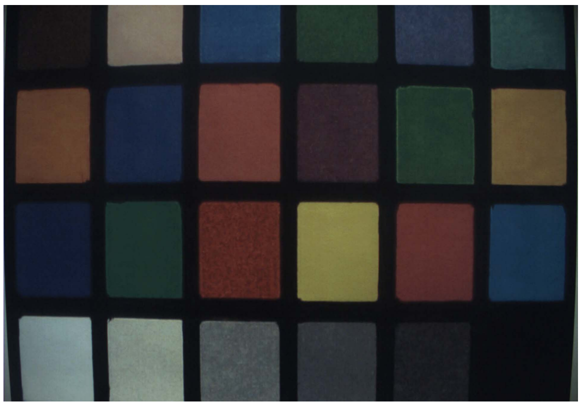
The implemented system processes images with HD resolution (1920x1080). The image sensors generate the image data and pass it directly to the system. Fig.9 shows the produced images after different image processing pipeline stages. Fig.9a shows the initial raw image from the sensor, Fig.9b shows the image after the color filter array interpolation stage and Fig.9c shows the image after the AWB stage. The AWB stage is divided into two parts, the first part is dedicated for reading the parameters and the second part uses the extracted parameters to modify the image pixel contents. Therefore, the image will be produced after executing two consecutive stages. The quality of the image depends on the quality of the algorithm is used in each stage. To have a system with good quality images, top algorithms in the state of art should be used. These algorithms will need more computation power compared to most of algorithms in the literature. The quality of the images is not provided in this work, as the algorithms can be changed dynamically.



(a)



(b)



(c)

Figure 9. Output images. (a) raw image (b) Color filter array interpolation output (c) AWB output.

VI. CONCLUSIONS

In this paper, we have presented an implementation of a reconfigurable camera system. The image-processing pipeline parts in the camera system have been implemented using the DPR technique to show that it is possible to fit all the possible stages or as much as needed in a compact design. Moreover, it shows that this technique adds a lot of flexibility to the system by providing the possibility of changing the functionality and choosing between different algorithms for the same function. The system has been implemented on Zynq 7000 All programmable SoC (z-7020) and it has been tested using 2.3 Megapixel VITA 2000 CMOS image sensor and display screen. The image processing pipeline parts have been developed using vivado HLS tool, which reduces the total development time. Although the parts are consuming much resources as they are computationally intensive, the implementation is highly cost effective and is in real time. The system is dedicated to process images of size 1920x1080 based on the input images sensors. A fast configuration engine has been designed and deployed to increase the configuration speed and increase the performance of the system. The system architecture achieved the needed clock speed by real time processing applications.

As a future work, we will focus on enhancing the performance of the system and add more flexibility in term of the size of RM on the FPGA. Moreover, we will examine the power consumption of this system based on different scenarios.

REFERENCES

[1] J. Chiang, S. Zammattio, WP: "Five ways to build flexibility into industrial applications with FPGAs," Altera Corporation, September 2014.

[2] J. Zhou, WP: "Getting the Most Out of Your Image-Processing Pipeline," Texas Instruments, Octorber 2007.

[3] M. Bergeron, S. Elzinga, G. Szedo, G. Jewett, T. Hill, XAPP794: LogiCORE 1080p60 Camera Image processing Reference Design (v1.3), Xilinx Inc., December 2013.

[4] W. C. Kao, S. H. Wang, L. Y. Chen and S. Y. Lin, "Design considerations of color image processing pipeline for digital

cameras", IEEE Transactions on Consumer ElectroniCS, vo1.54, no.4, pp. 1144- 1152, November 2006.

[5] Xin Zhao; Ying Yi; Erdogan, A.T.; Arslan, T., "Dual-core reconfigurable demosaicing engine for next generation of portable camera systems," Design and Architectures for Signal and Image Processing Conference (DASIP), pp.289, 294, 2010.

[6] C. Chen, S. Tan, W. Huang, "A novel hardware-software co-design for automatic white balance, " *Proceedings of the 7th Conference on 7th WSEAS International Conference on Multimedia, Internet & Video Technologies - Volume 7* (MIV'07), Lang Congyan (Ed.), Vol. 7. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 203-212.

[7] X. Tan, S. Lai, B. Wang, M. Zhang, Z. Xiong, "A simple gray-edge automatic white balance method with FPGA implementation," Journal of Real-Time Image Processing, January 2013, 1-11.

[8] J. F. Hamilton and J. E. Adams, "Adaptive Color Plane Interpolation in Single Sensor Color Electronic Camera," U. S. Patent, No. 5629734, 1997.

[9] W. T. Freeman, "Median filter for reconstruction missing color samples" U.S. Patent No. 4,724,395(1988).

[10] G. Zapryanov, D. Ivanova, I. Nikolova, "Automatic white balance algorithms for digital still camera- a comarative study," Information Technologies and Control 01/2012; 1:16-22.

[11] M. Fedor, "Approaches to color balancing," PSYCH221/EE362course project, Department of Psychology, Stanford University, U.S.A., 1998.

[12] J. Chiang and F. Al-Turkait, "Color balancing experiments with the HP-photo smart-C30 digital camera," PSYCH221/EE362 course project, Department of Psychology, Stanford University, U.S.A., 1999.

[13] D. A. Forsyth, "A novel algorithm for color constancy", International Journal of Computer Vision, vol. 5 (1), 1990, pp. 5 – 36.

[14] Avnet EM. , "HDMI Input/Output FMC Module." Internet: https://www.em.avnet.com/en-us/design/drc/pages/supportanddownloads.aspx?RelatedId=442, [Jan. 30, 2015].

[15] ON semiconductor. VITA 2000 2.3 Megapixel 92 FPS Global Shutter CMOS Image Sensor.

[16] E. Y. Lam, "Combining Gray World and Retinex Theory for Automatic White Balance in Digital Photography," *Proceedings of the Ninth International Symposium on Consumer Electronics*, June 2005, pp. 134-139.

[17] UG585: Zynq-7000 All Programmable SoC Technical Reference Manual v1.9.1, Xilinx Inc., November 2014.

[18] UG585: "7 Series FPGAConfiguration User Guide." v1.9, Xilinx Inc., November 2014.