Fast FPGA Prototyping based Real-Time Image and Video Processing with High-Level Synthesis

Refka Ghodhbani[1], Layla Horrigue[2], Taoufik Saidani[3], Mohamed Atri[4]
Laboratory of Electronics and Microelectronics, Faculty of Sciences of Monastir, University of Monastir, Monastir, Tunisia[1, 2, 3]
Department of Computer Science, Faculty of Computing and Information Technology[1, 3]
Northern Border University Rafha, Saudi Arabia[1, 3]
College of Computer Science, King Khalid University, Abha, Saudi Arabia[4]

***Abstract*—Programming in high abstraction level is known by its benefits. It can facilitate the development of digital image and video processing systems. Recently, high-level synthesis (HLS) has played a significant role in developing this field of study. Real time image and video Processing solution needing high throughput rate are often performed in a dedicated hardware such as FPGA. Previous studies relied on traditional design processes called VHDL and Verilog and to synthesize and validate the hardware. These processes are technically complex and time consuming. This paper introduces an alternative novel approach. It uses a Model-Based Design workflow based on HDL Coder (MBD), Vision HDL Toolbox, Simulink and MATLAB for the purpose of accelerating the design of image and video solution. The main purpose of the present paper is to study the complexity of the design development and minimize development time (Time to market: TM) of conventional FPGA design. In this paper, the complexity of the development™ can be reduced by 60% effectively by automatically generating the IP cores and downloading the modeled design through the Xilinx tools and give more advantages of FPGA related to the other devices like ASIC and GPU.**

Keywords*—High-level synthesis; FPGA; fast prototyping; real-time image processing; video surveillance; computer-aided design; model-based design; HDL coder; FPGA

I. Introduction

Image processing is taking place in increasingly numerous and complex fields to perform essentially control, inspection and data acquisition tasks [21]. We can cite industrial vision, video surveillance and spatial imagery, medical analysis, robotics ... The last in the list is the field of multimedia with its many recent applications. Image processing follows a well-defined process: to establish, from a raw image, a list of characteristics of the scenes viewed (or objects present in this image) to interpret the content of the image to guide or take a decision [1,2].

Advances in the integration capability of electronic circuits have opened up new perspectives for real-time image and video processing on embedded systems. On the one hand, specific processors can commonly perform billions of operations per second, and on the other hand, reprogrammable components will have billions of logical gates in the near future. These circuits make it possible to realize applications with performances in terms of speed of processing which are constantly increasing.

The past years have seen the explosion of the embedded systems market in many industrial and consumer domains such as telecommunications, satellites, and medical imaging. These increasingly important needs generate an industrial competition where factors such as cost, performance and especially the "Time To Market" become preponderant for the success of a product [25].

In this context, the Field Programmable Gate Array (FPGA) with its large integration and reconfiguration capabilities make it a key component for rapidly developing prototypes. In order to encourage the widespread diffusion of such circuits, it is necessary to improve the development environments to make them more accessible to non-experts in electronics [12,16].

Some applications of advanced computer vision algorithms include video histogram, color conversion system that can be found in modern cameras and many video surveillance [3,4]. Although it might not be necessary to have live video processing capability for many applications, some applications such as color conversion and histogram equalization used for autonomous driving system would require an input stream from cameras to be processed at real time in order to send signals back to the powertrain and steering control unit to respond properly [5,6,7]. FPGAs are a good choice platform for real-time video processing because energy efficiency and the potential to extract highly-parallelized calculations [7,8]. However, hardware development consumes typically more time and human resources than a similar software development would consume [20,22]. For a traditional development based on FPGAs, a good knowledge of digital logic circuit is necessary for Hardware Description Languages (HDLs) such as Verilog and VHDL to construct and config Register-Transfer Level (RTL) circuits in an FPGA [7,17].

Each software offers users with its model block. These tools can help users build the Simulink model with the provided block to generate HDL codes. As compared to the above three software, Simulink HDL Coder by which the generated HDL codes is characterized by its flexibility [18,19].

The goal for this paper is to conceive an automatically very high-level synthesis (VHLS) framework with the following features:

- A short time automatically creating for RTL desired rather than hours or even days.

- To examine the algorithm behavior described in very high-level languages.
- To achieve performances of the designs with the hardware constraints, including area of the target device or frequency.
- To be able to use the currently tools for hight level synthesis available.
- To boost code reuse from 0 to 60%.

The remainder of this paper is organized as follows: in the Section 2, related work on VHLS for image and video processing are presented. Section 3 present high-level synthesis proposed method for image and video prototyping, it discusses the challenge that we met when prototyping this conception, as well as the solutions. Proposed method prototyping and experiment results are given in Section 4. Finally, this paper is finished by a conclusion in Section 5.

A. Selecting a Template

First, confirm that you have the correct template for your paper size. This template has been tailored for output on the US-letter paper size. If you are using A4-sized paper, please close this file and download the file "MSW_A4_format".

B. Maintaining the Integrity of the Specifications

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

II. Related Works

To accelerate embedded real time image and video processing, in the last years, there are some given hardware-accelerated with different high-level synthesis method have been introduced in the automotive area, dedicated DSP-based systems and ASIC solutions compete against FPGA platforms and GPU applications [24].

Abdelgawad, Safar, and Wahba have developed a Canny Edge Detection Algorithm on Zynq Platform [1]. They utilized the Targeted Reference Design (TRD) for Zynq from Xilinx as the platform for the experiment of performance comparison between the CPU processor and the hardware accelerator. They declared memory accesses as the major bottleneck for a real-time video processing system. With proper buffering and directive-based optimizations, they were able to achieve a speedup of 100x on Zynq's hardware accelerator. They also provide the utilization estimation of the Canny Edge Detector hardware accelerator, but power analysis is missing. By using the TRD, they could inspect the performance improvement more directly thanks to the QT GUI interface. However, the TRD design gave rise to less control of the hardware design as well as software development.

Moreover, by the same research team, Monson, Wirthlin, and Hutchings attempted to optimize another popular image processing algorithm, Sobel filter, using Vivado HLS targeting a Zynq based FPGA [2]. Their first goal was to restructure an existing Sobel filter written in C to a C synthesizable version in Vivado HLS because the original code contains some non-synthesizable portions. Besides the restructuring, the authors discovered and applied three incremental optimizations that can be synthesized in Vivado HLS. The incremental optimization helped their design to achieve a performance of 388 FPS at a resolution of 640x480.

According to [5], the proposed approach operates on the building block level. All these devices seem to depend on hardware simulation and synthesis technology to derive performance scenarios. These figs are only available at a very late stage of the design process after a final FPGA integration.

Cai et al. utilized the capability of Vivado HLS to transform a software face recognition program to a corresponding hardware design based on Zynq platform [9,11]. Their intention was to improve the face detection performance, and the result indicates the performance was improved by up to 80% after migrating the computation onto the hardware. Their face location algorithm relies on color segmentation to detect human faces. The algorithm involves transforming from RGB color space to YCbCr color space, converting the query image to grayscale, and locating the skin color region after erosion and dilation. This algorithm results in straightforward and fast computations. Using color segmentation can be computationally efficient and it is possible to achieve real-time image processing performance. However, a relatively clean background is required for face detection using color segmentation. Also, misrecognition could occur if hands and arms are exposed in the query image.

Now, As opposed to the low-level design approach, Model Based design for FPGA are one of the methods that are based of high-level modeling for image and video processing applications on a very higher level of abstraction. Many various industrial and academic design approaches are available such as Simulink/ Xilinx System generator models, which can convert automatically into a hardware (VHDL) description [10,11].

Author in [13] provides a survey of HLS FPGA design flows for image and video processing applications. Although the given solutions focus on the composition, implementation or HDL generation of an optimal FPGA design, FPGA resources at execution time are neglecting considering reuse of.

III. Proposed Very High-Level Synthesis for Image Processing

According to [14,15], from September 2013 when MathWorks presented its hardware/software workflow for Zynq-7000 focusing Model-Based Design (MBD). Based on this new proposed workflow presented in Fig. 1, models are designed in Simulink using HDL toolbox that can show a completely dynamic system. These include a Simulink model for algorithms targeted for the Xilinx Zynq SoC platform, and Quickly create software- hardware implementations for Zynq platform directly from the algorithm and system design.

A. Rapid Prototyping Flow with HDL coder

This paper presents a video processing system rapid prototyping flow that allows engineers with little to no HDL experience to develop an FPGA based high-performance video processing system.

Fig. 2 demonstrates the rapid prototyping flow from a high-level point of view, and this paper focuses on three of the most essential steps in the flow:

- An FPGA-based SoC video processing system architecture needs to be designed. The system should allow integration of generated IPs from high-level synthesis tools to realize real-time video processing capability.
- The design enables the adoption of FPGA acceleration kernels developed by high-level synthesis tools so that engineers can quickly reconfigure the functionality of the system.
- System-level communications allow users to use software for initializing and configuring modules that are developed in the hardware system.

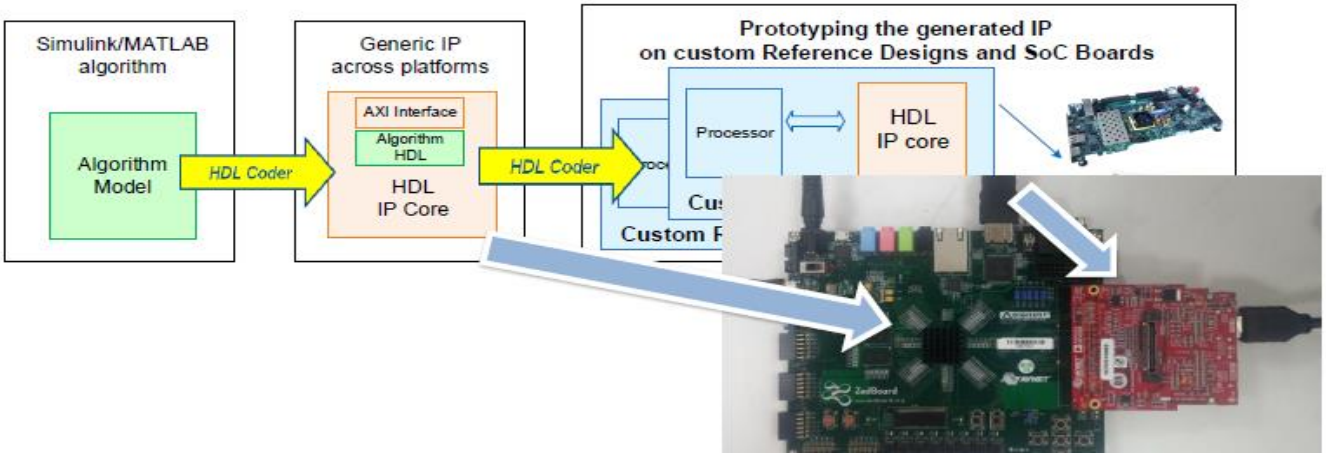

Fig. 1. HDL Coder Workflow.

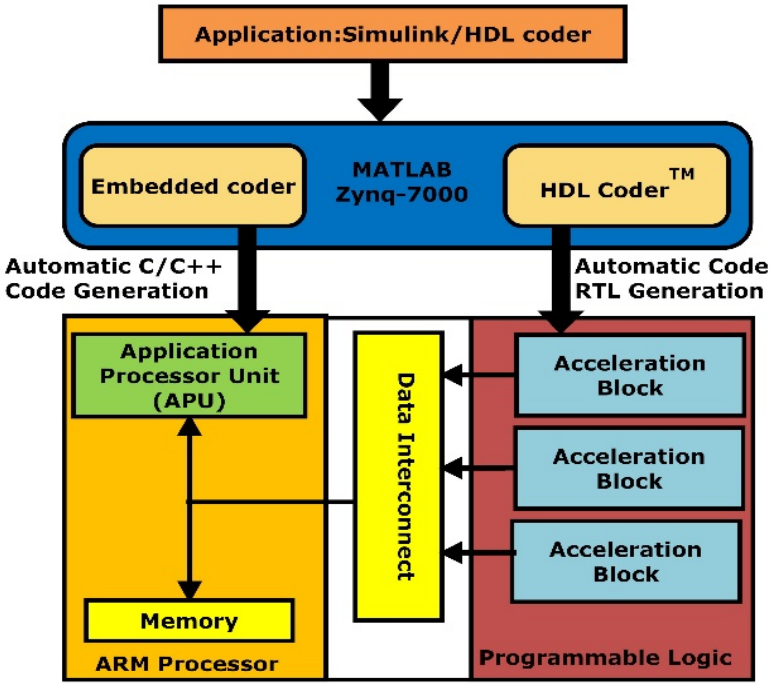

Fig. 2. Model based Design Prototyping with MATLAB/HDL Coder.

The development of the proposed real-time video processing system is divided into two parts: 1) Video processing system architecture design, and 2) Video processing algorithms design. The first part discusses the major components contributed to the video processing system on the Zynq platform including the AXI4 Interfaces used for high throughput data transmissions, while the second part discusses the approaches and optimizations, we have taken for building video processing algorithms using Vivado HLS [25]. The proposed approach is based on the following key:

- Simulation in Simulink used by system designers and algorithm developers is utilized for two reasons. The designer involves creating models for a complete system – communications, image and video processing components. The second reason is to facilitate partition model between hardware and software component and make a good compromise for high-level synthesis.
- High-speed I/O cores for the Xilinx Zynq 7000 platform and IP cores creating can be easily generated by using HDL code generation from HDL coder TM.
- The Zynq Cortex-A9 cores programming, by using of embedded coder from Simulink support rapid embedded software iteration [23].
- Relating to the ARM processing system and programmable logic with support for Xilinx Zynq 7000, automatic AXI4 interfaces cores can be generated.
- Integration with downstream tasks, including software compilation, the executable for the ARM and bit stream generation using Xilinx implementation tools like Vivado and downloading directly to Zynq 7000 platform boards permits a rapid prototyping workflow.

IV. Experiments Applications

The experimental of the proposed approach are investigated by utilizing real-time applications for image and video applications (Fig. 3).

As clarified in, the design is structured and verified in MATLAB and Simulink. Then, it targeted to the Zynq-7000 on the Xilinx Zed board development kit [23]. The real-life application algorithm is achieved on the FPGA fabric through HDL Coder for system acceleration, and it is executed on the ARM Cortex-A9 processor, as shown in Fig. 4.

A. Color Histogram Equalization

A histogram can be defined as a diagram that describes how many pixels of an image or a video frame have a particular intensity. It includes different applications in image and video processing [1]. This is due to the simplicity of extracting histogram features. Its characteristics are invariant to image rotation. Moreover, it has low storage demands as compared to the size of the image.

Fig. 5 presents histogram equalization module flowchart operations. The flowchart is composed by two states of operation. When executing, the ready input signal is approved, using a lookup table the input value is transformed, and the histogram is generated. The module enters the second mode of operations once the complete image has been streamed via the module, if the input is not ready so that the new lookup table can be calculated. For the new transformation lookup table generation, the size of the input image approves the accumulating and normalizing histogram module. Lookup table is updated by normalized values, and running mode of operation is done once all 255 values have been updated.

1) Simulink HDL coder Model

a) Video Partition: The video partition component in this design divides a big input frame to 4 small images. For each small frame histogram is generated. The big input image is divided into 160 by 120 small images. There is a connection between each small partition, Frame, pixel and block. This video partition module generates pixel stream and corresponding control signals.

b) HDL Histogram: this module is a part of hardware acceleration. It is designed with HDL coder toolbox and Simulink library. Using the vision HDL toolbox Histogram, the pixel stream of histogram is calculated. The grayscale input pixels are classified into 256 bins.

The model presented by Fig. 6 reads the calculated histogram bins sequentially once the block asserts the read Rdy signal. The bin values are sent for cumulative histogram calculation. After all 256 bin values are read, the model asserts binReset to reset all bins to zero. The collected histogram of each small image is then added together to compute the accumulated histogram of the big image (Fig. 7).

Equalization module: The calculated and accumulated histogram for the current frame generated by a histogram module is processed by equalization module to store the input video. This last input video is delayed by one frame. The uniform equalization is performed to the original video. Finally, a comparison between the original video and the equalized video is done.

2) Synthesis and FPGA implementation: Once the Histogram process is completed and Simulink code of design is successfully converted into hardware design, generated VHDL code of histogram equalization is verified through co-simulation using ModelSim 10.3d software. A further design is processed in Vivado 17.4 Design Suite for synthesis and implementation on Xilinx Zynq xc7z020clg484-2 FPGA device. The logic resources utilized by design with timing performance are presented in Table I. Table I represents the total number of slices and look-up tables used in this design, which indicates entire area occupied in the target device. From the Table II, it is found that the proposed design is working with an estimated speed of 170 MHz by utilizing only 3350 slices. Proposed model is using 2770 lookup tables.

Hardware consumption in any design determines its cost. Therefore, the cost of proposed design is decreased due to lesser hardware utilization. Hence, the suggested design methodology improves efficiency in area and provides good choice in terms of low-cost hardware. The resource usage and maximum frequency for this module are shown in Table II.

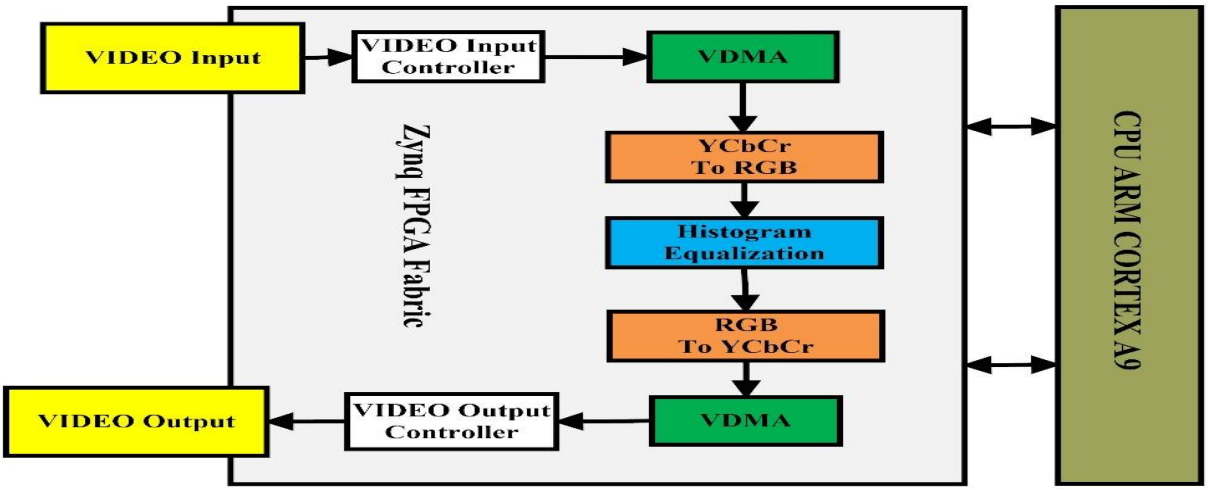

Fig. 3. Low-Light Video Processing Architecture Implemented on the FPGA.

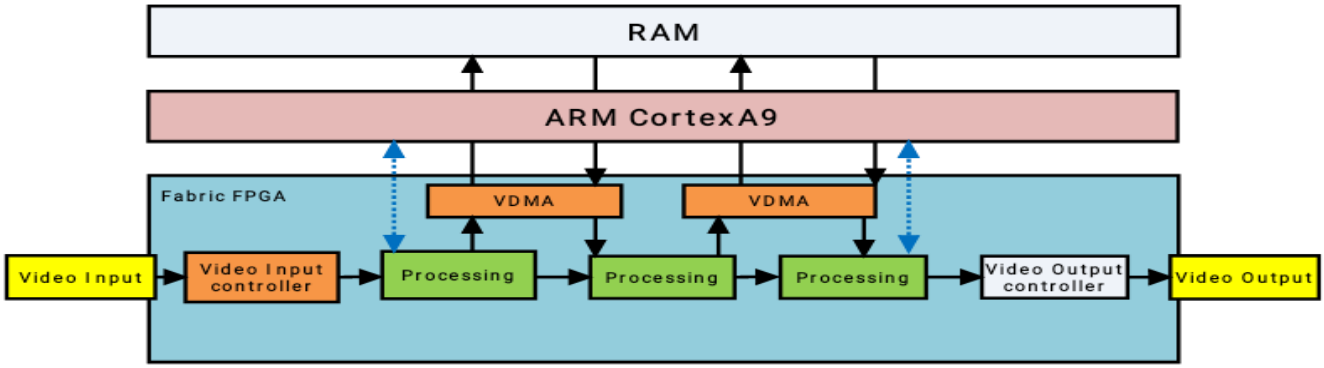

Fig. 4. Real-Time Video Processing Architecture based on Zynq 7000 FPGA and ARM Processor.

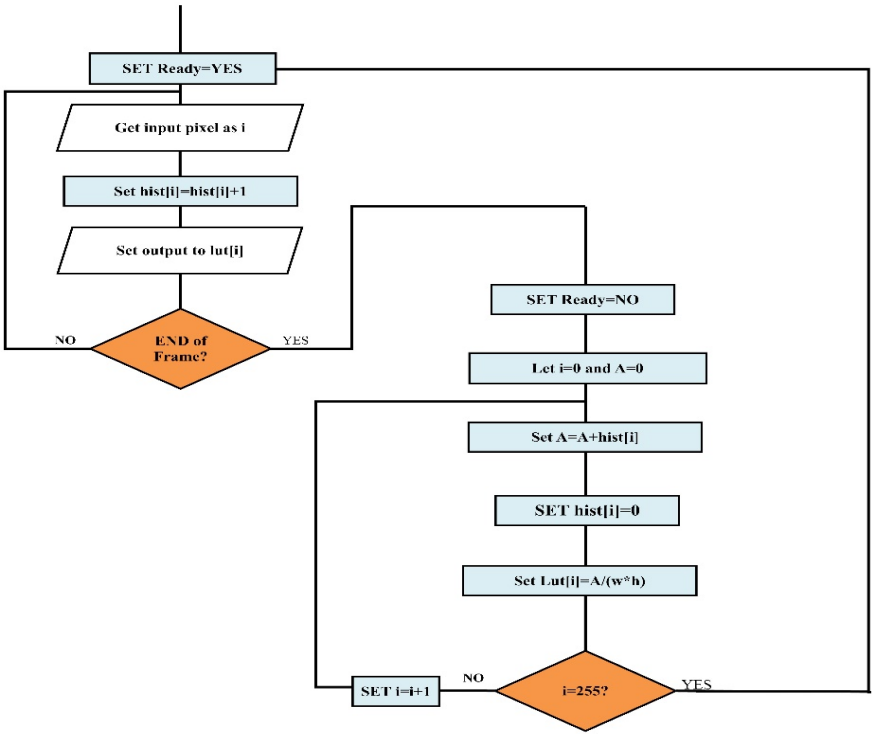

Fig. 5. Histogram Equalization Module Flowchart.

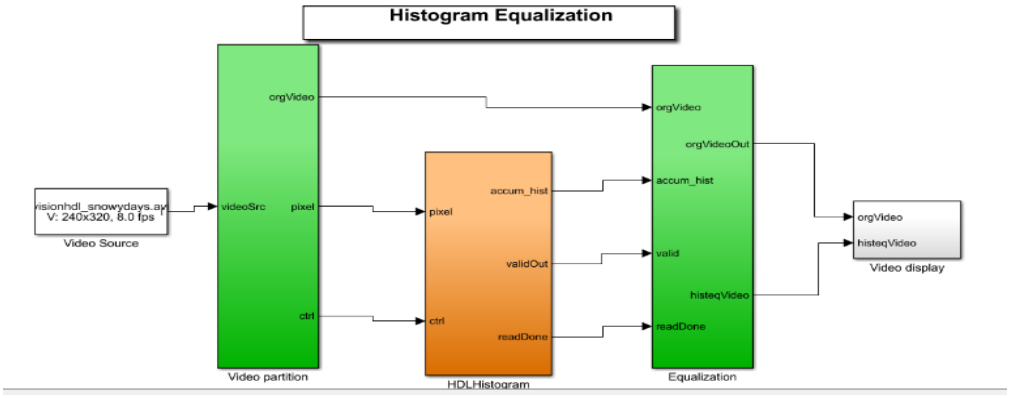

Fig. 6. Model based Design of Histogram Equalization using HDL Coder.

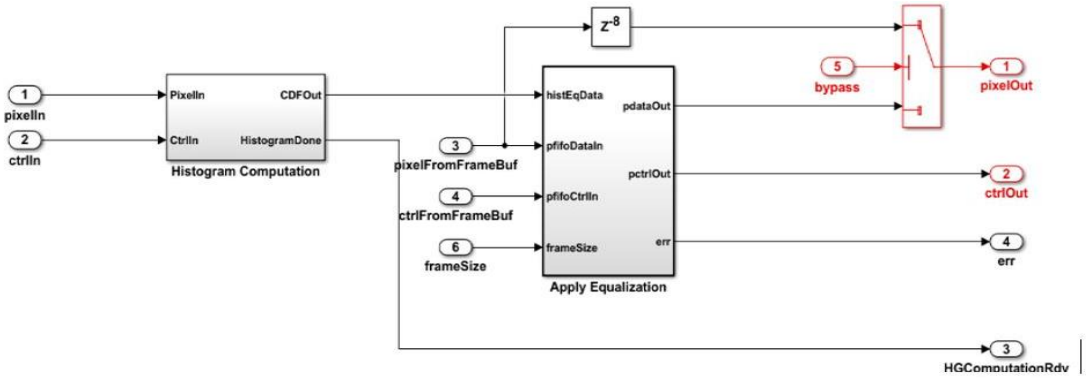

Fig. 7. HDL Histogram Equalization Subsystem.

TABLE. I. UTILIZATION OF THE AVAILABLE RESOURCES IN THE ZYNQ XC7Z020CLG484-2 PART

Bit Depth	8	
Channels	1	
LUT-FF Pairs	3740	7%
LUTs as Logic	2770	5%
LUTs as Memory	289	1.66%
Slice Registers	3350	3%
RAM 36/18	0.5	0.36%
DSP48	0	0.00%

TABLE. II. UTILIZATION AND MAXIMUM FREQUENCY FOR THE HISTOGRAM EQUALIZATION MODULE

Max frequency: 170 MHz		
Resolution	***Pixel Per Frame***	***Maximum Frame Rate***
1920x1080	2073600	82.1FPS
1440x900	1296000	129.6 FPS
1024x1024	1048576	159.7 FPS
1280x720	921600	181.0 FPS
1024x768	786432	211.5 FPS
640x480	307200	536.9 FPS
512x512	262144	628.6 FPS

B. *Color Conversion System*

Color conversion converts the raw image having colors belonging to the color space of the sensor into values in a standard color space independent of the sensor. The RGB color space is the standard widely adopted by the image and video processing system. Hence the interest of making the conversion directly to this color space. The conversion is performed using a standard method which is the use of a 3x3 conversion matrix.

For the forward conversion module, the conversion module uses the following matrix conversion:

$$Y = \frac{54.4256}{256} \cdot R + \frac{183.0912}{256} \cdot G + \frac{18.4832}{256} \cdot BCb$$

$$= -\frac{29.3305}{256} \cdot R - \frac{98.6695}{256} \cdot G + \frac{128}{256} \cdot B + 128Cr$$

$$= \frac{128}{256} \cdot R - \frac{116.2631}{256} \cdot G - \frac{11.7369}{256} \cdot B + 128$$

The following equations present the backward conversion from YCbCr space to RGB space:

$$R = Y + \frac{403.1488}{256} \cdot Cr - 201.5744G$$

$$= Y - \frac{47.9550}{256} \cdot Cb - \frac{119.8398}{256} \cdot Cr + 83.8974B$$

$$= Y + \frac{475.0336}{256} \cdot Cb - 237.5168$$

1) Simulink HDL coder model: Color image processing is a logical extension to the processing of grayscale images. The essential difference involves the fact that each pixel is composed of a vector of components rather than a scalar. Usually, a pixel from an image has three parts: red, green and blue. These are defined by the human visual system. A three-dimensional vector and user mainly present color can determine how many bits each component have.

The pre-defined video reference design, which contains other IPs to handle the HDMI input and output interfaces. The HDL DUT IP processes a video stream coming from the HDMI input IP, generates an output video stream and sends it to the HDMI output IP. All of these video streams are transferred to AXI4-Stream Video interface. The HDL DUT IP can also include an AXI4-Lite interface for parameter tuning. Compared to the AXI4-Lite interface, the AXI4-Stream Video interface transfers data much faster, making it more suitable for the data path of the video algorithm.

The color conversion system IP core was established in Matlab Simulink environment with HDL coder. This tool allows the user to drive the Zynq programmable logic (PL) part at a high level without having to deal with low-level hardware details. The proposed architecture is shown in Fig. 8.

Fig. I gives an overview of the entire color system conversion Simulink system. The other blocks are identical with those in color correction model except the RGB2YCbCr kernel block. The input bus signal is represented in Xilinx video data format, which is 32'hFFRRBBGG. So, we first separate the bus signal to three color components RGB. Red component is bit 23 to 16; Green is bit 7 to 0; Blue is bit 15 to 8. The gain blocks implement multiplication. The sum blocks calculate add and subtract result, which is defined in block parameter. Matrix multiplication is performed using these gain and sum blocks. Finally, again to a bus signal. The delay blocks inserted in between will be transferred to registers in hardware, which break down the critical path to achieving higher clock frequency.

2) Synthesis and FPGA implementation: A block design for color space conversion that implements an entire image processing system can be created.

The vivado project for the video processing system is generated by a proposed worksflow based on model-based design (MBD). This project can be opened with Xilinx Vivado version 2017.4.

Fig. 10 presents a full diagram for color conversion system. This diagram is composed by many blocks such as RGBtoYCbCr conversion IP, YCbCrtoRGB conversion IP. Also, the video processing HDMI input output, the Zynq's CPU cores, the Video DMA engines, and all the supporting blocks.

We validate the entire color conversion system design on a Zynq FPGA platform using generated HDL code. The logic resources utilized by design with timing performance are presented in Table III. The reported maximum frequency is 302 MHz for RGB to YcbCr and 260 MHz for YCbCr to RGB. The resources utilization of the YCbCr to RGB system on Zynq xc7z020clg484-2 FPGA is as follows: 1850 slice registers and 2280 slice LUTs.

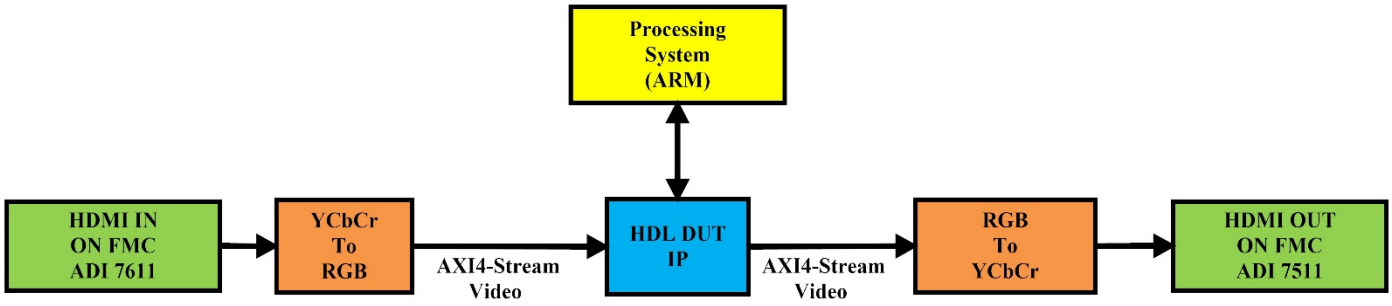

Fig. 8. AXI4-Stream Video Interface in Zynq 7000.

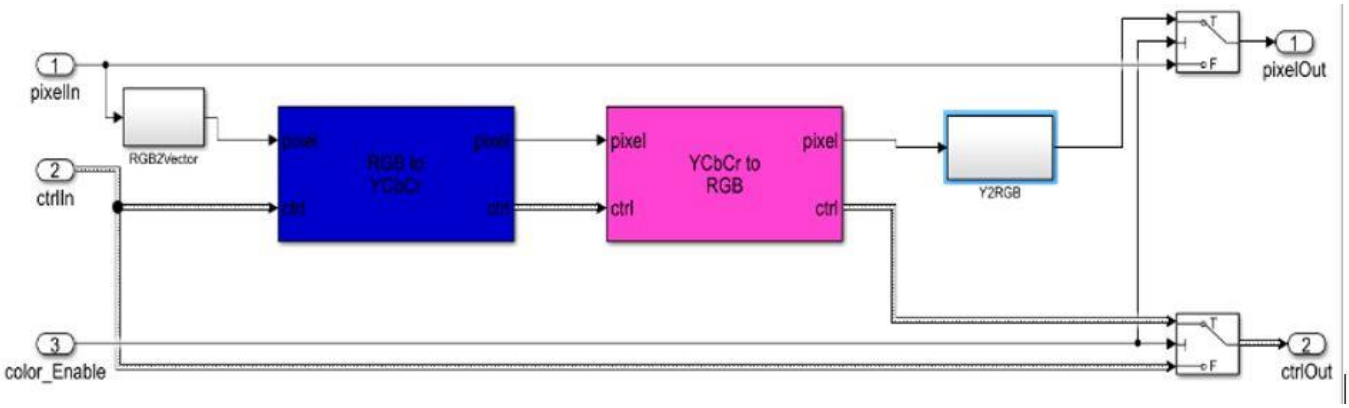

Fig. 9. Hardware Prototype for Color System Conversion.

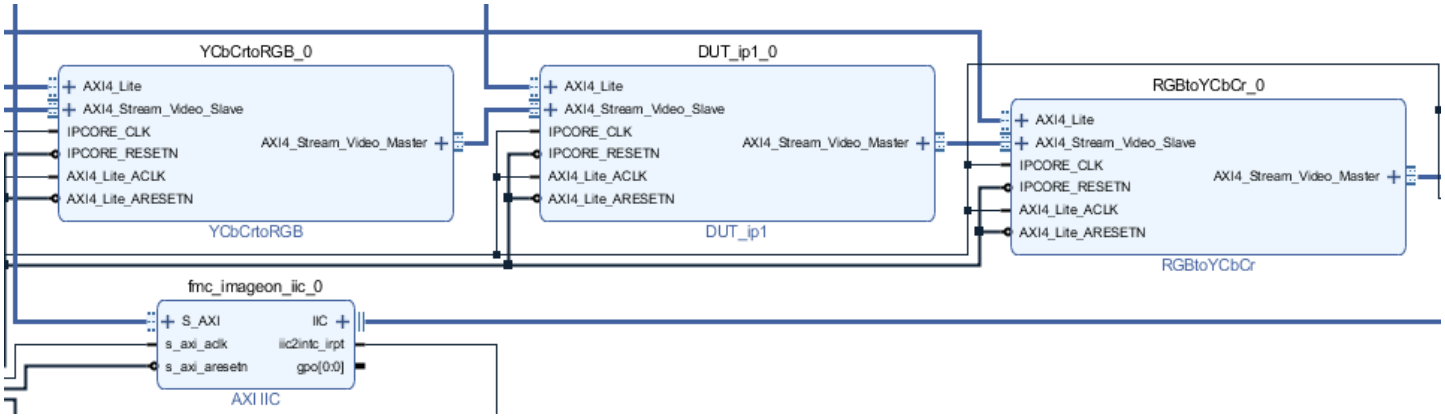

Fig. 10. Vivado IP Integrator with Several IP-Blocks for Color System Conversion.

TABLE. III. UTILIZATION OF AVAILABLE RESOURCES IN THE ZYNQ XC7Z020CLG484-2 PART

	YCbCr to RGB		**RGB to YCbCr**	
Bit Depth	8		8	
Channels	3		3	
LUT-FF Pairs	2280	4.20 %	4400	8.1 %
LUTs as Logic	1938	3.64%	3864	7.26%
LUTs as Memory	6	0.03%	4	0.02%
Slice Registers	1850	1.50%	3580	1.4%
RAM 36/18	0	0.00%	0	0.00%
DSP48	0	0.00%	0	0.00%

Table IV illustrates the result of our YCbCr to RGB system on hardware for color system conversion. Table IV present the resource usage and maximum frequency.

The resources utilization of the RGB to YCbCr system on Zynq xc7z020clg484-2 FPGA is as follows: 3602 slice registers and 4400 slice LUTs.

TABLE. IV. UTILIZATION AND MAXIMUM FREQUENCY COLOR SPACE CONVERSION

YCbCr to RGB			**RGB to YCbCr**	
Max frequency: 302 MHz			***Max frequency: 260 MHz***	
Resolution	***Pixel Per Frame***	***Maximum Frame Rate FPS***	***Pixel Per Frame***	***Maximum Frame Rate FPS***
1920x1080	2073600	143.1	2073600	125.5
1440x900	1296000	228.2	1296000	197.6
1024x1024	1048576	282.5	1048576	242.5
1280x720	921600	321.3	921600	376.2
1024x768	786432	373.4	786432	324.1
640x480	307200	955.5	307200	823.1
512x512	262144	1.125.6	262144	963.2

V. DISCUSSION AND CONCLUSION

The current paper suggests a VHLS method for image processing designs. This method gives a high abstraction level environment to the users, which can improve the development productivity by automating the MATLAB/Simulink-to-RTL synthesis process. We prototyped the suggested method by utilizing recently available Model-Based Design tools based on Simulink /HDL coder modeling. This method is then verified within two real-life applications. Experiments show that it can lead to the benefits of FPGA related to the tools of other kinds in the same abstraction level. It is worth noting that the findings of the study show that it will decrease the complexity of the algorithm behaviors depicted using MATLAB in routine level. This study also demonstrates the usefulness of heterogeneous Zynq SoC to establish an embedded vision system for a smart camera dedicated to traffic surveillance.

Resolving the following issues can facilitate reaching this goal; one of the needs of hardware-software architecture is to monitor the mastery of the HDMI video signal to the system the AXI bus-based communication between the FPGA and ARM processor. It shows that the suggested flow reduces FPGA prototyping time by up to 60% with MATLAB and HDL Coder. MATLAB and HDL Coder are used to eliminate the step of translating the initial algorithm to HDL by hand. HDL coder facilitates the improvements completed in hours, not weeks.

REFERENCES

[1] H. M. Abdelgawas, M. Safar, A. M. Wahba, “High Level Synthesis of Canny Edge Detection Algorithm on Zynq Platform,” International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:9, No:1, 2015, pp. 148-152.

[2] J. Monson, M. Wirthlin and B. L. Hutchings, "Implementing high-performance, lowpower FPGA-based optical flow accelerators in C," Application-Specific Systems, Architectures and Processors (ASAP), 2013 IEEE 24th International Conference on, Washington, DC, 2013, pp. 363-369.

[3] Ben Hamida, A., Koubaa, M., Nicolas, H., Amar, C.B.: Video surveillance system based on a scalable application-oriented architecture. Multimedia. Tools Appl. pp. 1–27 (2015). doi: 10.1007/s11042-015-2987-5.

[4] Fleck, S., Strasser, W.: Smart camera-based monitoring system and its application to assisted living. Proc. IEEE 96(10), 1698–1714 (2008). doi: 10.1109/JPROC.2008.928765.

[5] Huang, D.Y., Chen, C.H., Chen, T.Y., Hu, W.C., Chen, B.C.: Rapid detection of camera tampering and abnormal disturbance for video surveillance system. J. Vis. Commun. Image R. 25(2), 1865–1877 (2014).

[6] Chao Li, Yanjing Bi., Benezeth, Franck Marzani, Fan Yang: Fast FPGA prototyping for real-time image processing with very high-level synthesis. J. Real-Time Image Process. (2017). doi: 10.1007/s11554-017-0688-1.

[7] Henning Sahlbach, Daniel Thiele, Rolf Ernst: A system-level FPGA design methodology for video applications with weakly-programmable hardware components. J. Real-Time Image Process. (2017). doi: 10.1007/s11554-014-0403-4.

[8] Baklouti, M., Aydi, Y., Marquet, P., Dekeyser, J., Abid, M.: Scalablempnoc for massively parallel systems—design and implementation on FPGA. J. Syst. Archit. 56(7), 278 – 292 (2010). doi:10.1016/j.sysarc.2010.04.001. Special Issue on HW/ SW Co-Design: Systems and Networks on Chip.

[9] T. Han, G. W. Liu, H. Cai and B. Wang, "The face detection and location system based on Zynq," Fuzzy Systems and Knowledge Discovery (FSKD), 2014 11th International Conference on, Xiamen, 2014, pp. 835-839.

[10] P. K. Dash, S. S. Pujari and S. Nayak, "Implementation of edge detection using FPGA and Model-based approach", Proceedings of 2014 International Information Communication and Embedded Systems (ICICES), IEEE, (2014), pp. 1- 6.

[11] Sukhwani, B., Thoennes, M., Min, H., Dube, P., Brezzo, B., Asaad, S., Dillenberger, D.: A hardware/software approach for data base query acceleration with FPGAs. Int. J. Parallel Prog. 43(6), 1129–1159 (2015). doi:10.1007/s10766-014-0327-4.

[12] Jiang, J., Liu, C., Ling, S.: An FPGA implementation for real time edge detection. J. Real-Time Image Process. (2015). doi:10. 1007/s11554-015-0521-7.

[13] S. Sanchez-Solano, M. BroxJimenez, E. delToro, P. BroxJimenez and I. Baturone, "Model-based design methodology for rapid development of fuzzy controllers on FPGAs", IEEE Trans. Ind. Informat, vol. 9, no. 3, (2013), pp. 1361-1370.

[14] The MathWorksInc, (2014, May 17), "Optimizing HDL Code" [Online], Available:http:// http://www.mathworks.se/products/hdl-coder/description3 .html.

[15] The MathWorksInc, (2014, May 17), "Automating FPGA Design" [Online], Available:http:// www.mathworks.se/products/hdl-coder/description4.html.

[16] Bailey, D.G.: Design for Embedded Image Processing on FPGAs. Wiley (Asia) Pte Ltd, Singapore (2011).

[17] The MathWorksInc, (2014, May 18), "GenerarateVerilog and VHDL code for FPGA and ASIC designs." [Online], Available: http:// www.mathworks.se/products/hdlcoder/.

[18] The MathWorksInc, (2014, May 17). "HDL Coding standards" [Online], Available:http:// www.mathworks.se/products/hdl-coder/description7. html.

[19] The MathWorksInc, (2014, May 17), "Generating HDL Code" [Online], Available:http:// www.mathworks.se/products/hdl-coder/description2. Html.

[20] Kyo, S., Okazaki, S.: IMAPCAR: a 100 GOPS in-vehicle vision processor based on 128 ring connected four-way VLIW processing elements. J. Signal Process. Syst. 62, 5–16 (2011).

[21] Leu, A., Aiteanu, D., Graser, A.: A novel stereo camera-based collision warning system for automotive applications. In: IEEE International Symposium on AppliedComputational Intelligence and Informatics (SACI), pp. 409–414 (2011).

[22] Stein, G.P., Rushinek, E., Hayun, G., Shashua, A.: A computer vision system on a chip: a case study from the automotive domain. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPRW) (2005).

[23] Crockett, L.H.; Elliot, R.A.; Enderwitz, M.A.; Stewart, R.W. The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc; Strathclyde Academic Media: Glasgow, UK, 2014.

[24] Xilinx: Introduction to FPGA design with vivado high-level synthesis. Tech. Rep. UG998 (v1.0), Xilinx (2013).

[25] Coussy, P., and Morawiec, A.: 'High-Level Synthesis: from Algorithm to Digital Circuits', Berlin: Springer Science + Business Media, chapters 1, 4, 2008.