A Scalable Image/Video Processing Platform with Open Source Design and Verification Environment

1st Xiaokun Yang
Engineering Department
University of Houston Clear Lake
Houston, USA
yangxia@uhcl.edu

2nd Yunxiang Zhang
Engineering Department
University of Houston Clear Lake
Houston, USA

3th Lei Wu
Computer Science Department
Auburn University at Montgomery
Montgomery, USA

***Abstract*—This paper presents a scalable image/video processing platform on Field-Programmable Gate Array (FPGA), capable of capturing images via a low-cost OV7670 camera and in real time displaying the original, in-process, and final results of images on a VGA-interfaced monitor. To make the platform expandable and reusable, not only is the design with Verilog hardware description language (HDL) offered, but also the verification environment including six open verification components (OVCs) is provided. To the best of our knowledge, this proposed work costs the least FPGA resource (753 LUTs and 277 Register) compared to the existing open source implementations on the design of the Camera-FPGA-VGA data path.**

We make this platform publicly available to encourage future research projects on image/video processing, computer vision, machine learning, etc., and to serve educational studies on digital system design with Verilog HDL and FPGAs.

***Index Terms*—Field-Programmable Gate Array (FPGA), hardware description language (HDL), Verification**

I. Introduction

The rapid growth in computer vision applications such as object and facial recognition imposes many challenges of computation speed and energy savings on traditional software based frameworks [1], [2]. Due to the benefits of programmability and parallel computing on pure hardware design, recently the implementation on Field-Programmable Gate Array (FPGA) is becoming widely used in executing many image/video processing tasks [3].

Under this context, this paper proposes an open source image/video processing platform, enabling to capture frames of images by interfacing a low-cost OV7670 camera and in real time display both the original images and the results of processed images on a VGA-interfaced monitor. Specifically we presents our framework able to simultaneously display up to four 320×240 images in a 640×480 window, capable of showing the in-process images, the final results of the images, as well as the original images through the VGA interface. As a case study, we design a simple color to binary converter with two submodules - color to grayscale converter and then grayscale to binary converter. We demonstrate the validity of showing all the original color images captured from the OV7670 camera, the inter-process grayscale images, and the final binary images in different regions via the VGA master. The Design-Under-Test (DUT) was written by Verilog HDL and tested on the Nexys 4 FPGA, and the verification environment can be automatically run on ModelSim Simulator. The main contributions of this work are:

- To make the design reusable and expandable, not only was the DUT provided but also a verification environment including tcl script, filelist, and testbench with six open verification models (OVCs) like bus function models (BFMs) and scoreboards, was offered. We expect that the public release of this platform will lead to better implementations in the future, as well as to generate many application-specific designs on teaching and research in image/video processing, computer vision, machine learning, etc.
- In simulation, each design module - the I2C master, Image Capture slave, or VGA master, has been equipped with a bus function model for driving the interface, and a scoreboard for testing functions as well. The scoreboards are controllable by the tcl script to compare data between DUT and golden models, in order to automatically verify the system functions and to increase the re-usability and credibility of the publicly available design sources.
- We evaluated the design performance in terms of slice count and power consumption. As far as we know, our proposed work costs the least FPGA resource - 753 slices as Look-up-Tables (LUTs) and 277 slices as Registers, compared to all the open source implementations on the design of Camera-FPGA-Monitor data path. And the power consumption is around 220 mW for displaying and processing one frame of color image.

The organization of this paper is as follows. Section **??** briefly introduces the related works and III presents our work with design architecture and verification environment. Section IV discusses the implementation of data path and Section V depicts the details of the verification environment. In Section VI, the experimental results in terms of hardware cost, power consumption, and FPGA prototype are shown. Finally, Section VII presents the concluding remarks in our target architecture.

II. Related Works

Prior works in applications of image/video processing on FPGA mainly focused on high-level synthesis (HLS) and

block designs, and sometimes with software/GPU involved. For example, [10] proposed a design on a real-time image acquisition and pre-processing FPGA by using LabVIEW with GPU-based acceleration that is capable of sustaining the rate of data acquisition. And in [11], an integration of a camera with frame grabber on LabView, a mask creating on MatLab, and an image processing module on LabView FPGA has been presented. It is useful to demonstrate a system by directly using the black-box based designs, but to customize or improve the system is hard because the block libraries are usually not open source or changeable by users.

Under this context, many register-transfer level (RTL) designs on the Camera-FPGA-VGA data path have been proposed. For example, the implementation in [8] has been widely used to prototype or test new research ideas related to image/video processing systems. It is written by VHDL and performed on Zedboard FPGA. Additionally, a reconfigurable platform on Virtex-4SX35 FPGA has been presented in [9], enabling to preform edge detection by interfacing an OV7610 camera as image input and VGA as the result output. However, because the FPGA resource cost and power consumption have not been presented in these two works, the design performance is not able to be compared and estimated.

In [12], [13], an open source project with digital camera on FPGA has been proposed. The project was designed by using VHDL and prototyped on Altera DE2-115 FPGA. It consumed 1,616 logic elements and 818 registers. The main concern of this open source project is the lack of a verification environment, making the design hard to be expanded and improved.

To tackle aforementioned issues, this paper presents a scalable image/video processing platform with Verilog HDL on Nexys 4 FPGA, containing both design and verification environment.

III. Proposed Work

In this section, the design structure on the image/video processing platform is discussed. Moreover the verification environment is presented to make the open source platform expandable and reusable.

A. Design Architecture of Image/Video Processing Platform

Fig. 1 shows the design architecture of the image/video processing platform, capable of capturing images in 320×240 and displaying the original images, in-process images, and the final results of processed images in a 640×480 window.

Basically the platform contains three interfaces - the I2C master, Image Capture slave, and VGA master, as well as four Frame Buffers and one PLL instantiation. The I2C master is designed to configure the functional registers released by the OV7670 camera [4]. After configuration, the camera enables to capture images and send them pixel by pixel through "VSYNC-HREF-DATA" interface. The Image Capture slave receives the images and stores them into four Frame Buffers, $320 \times 240 = 76,800$ pixels/bytes for each. In other word, each buffer is able to store one frame of 320×240 image.

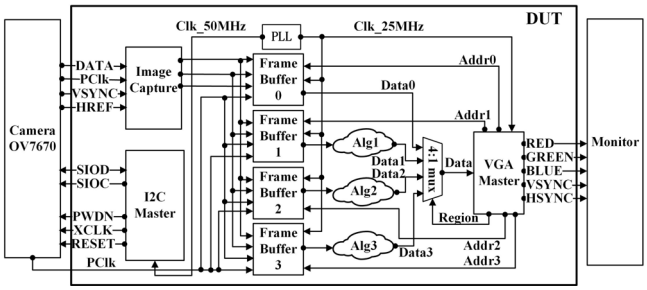


Fig. 1. Design Architecture of the Image/Video Processing Platform

Finally the VGA master creates the "HSYNC-VSYNC" timing for displaying the images on VGA-interfaced monitor. In this case we split the 640×480 window into four 320×240 regions, as shown in Fig. 2. The Region0 is used to display data read from the Frame Buffer0. And the Region1, Region2, and Region3 display the data processed by three different algorithms, Alg1, Alg2, and Alg3, respectively. Notice that the implementations of the algorithms may be interconnected or inter-operated. For performing a complex algorithm, multiple clock cycles and many buffers/FIFOs might be required.

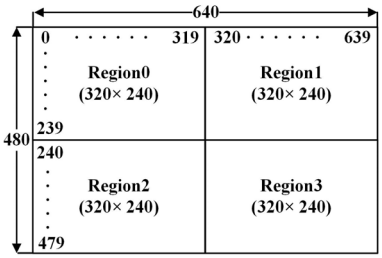


Fig. 2. 640×480 window with four 320×240 regions

B. Verification Environment

Fig. 3 depicts the entire open source packet, including not only the DUT with synthesizable Verilog HDL, but also a configurable verification environment. In particular, the green box represents the design hierarchy with three different levels, from top datapath design to submodules. The synthesis files are in the orange box, containing a constrain file, a netlist for FPGA programming, and the synthesis results in terms of slice count and power cost.

In this work we emphasize the verification environment, which is shown in the blue box. Generally it includes a tcl script, a filelist, and a testbench. The tcl script is used to automatically run the simulation, and configure the DUT and OVCs working in different modes. The filelist maintains all the design and verification components.

In the testbench, three bus function models and three scoreboards are provided as OVCs. For example, a Capture master BFM is required to send images to the DUT. The input images come from an image input file named as "rgb565_input.txt". The data in this file is in the RGB565 format - 5-bit red pixel, 6-bit green pixel, and 5-bit blue pixel. Thus each pixel transfer

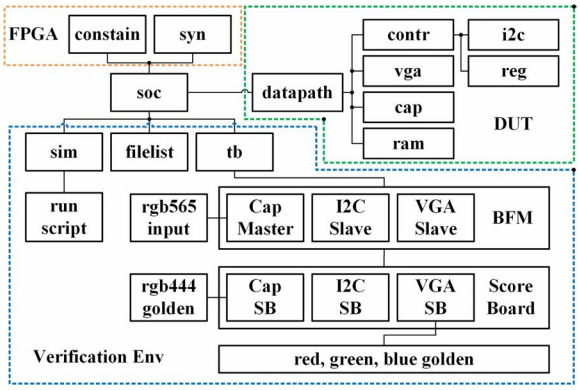


Fig. 3. Open Source Design, Verification, and Synthesis

takes two cycles, 8 bits per cycle. Likewise, an I2C slave BFM is created to receive and response to the commands from I2C master, and a VGA slave is needed to collect the images from the VGA interface.

Furthermore, three scoreboards are presented in the test-bench to check the results of each design interface. For example, the I2C scoreboard compares each command sent by the I2C master with the original register configuration. To test the images stored into the Frame Buffers and the images driven on the VGA port, two golden models or four golden files are provided. The Capture scoreboard compares the data written into Frame Buffers with the golden data in "golden_rgb444.txt" file. Similarly, the red, green, and blue pixels driven by the VGA master should be compared with the data in golden_r.txt, golden_g.txt, and golden_b.txt files, respectively. The details of the design on each BFM and OVC will be discussed in the V sections.

IV. Design Under Test (DUT)

In this section, all the design submodules and IPs are introduced, including one I2C master, one Image Capture slave, one VGA master, and two IPs - the clock PLL and Frame Buffers as well.

A. I2C Master Design

In our work, we use the low-cost OV7670 camera to provide windowed images in a wide range of format, controlled through the I2C interface. Hence, first an I2C master is needed to configure the functional registers offered by the camera. The I2C slave has an 8-bit ID that needs to be unique on the bus, "0x42" for a specified write command and "0x43" for a read.

As per the OV7670 datasheet [4], the serial clock ("SIOC") provided by the I2C master should be less than 400KHz. Using a 50MHz clock as input, we produce the "SIOC" signal around 200KHz. Additionally I2C communication typically applies transfers of 8 bits or bytes on serial data ("SIOD"). As an example of the timing shown in Fig. 4, the I2C master sequentially sends a "WRITE" command to camera by driving the first byte as the unique device ID, and the second and third bytes as the register address and value.

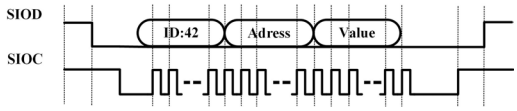


Fig. 4. I2C Controller Timing

B. Image Capture Slave Design

Using the I2C master interface, we configure the OV7670 camera to capture images in the format of 320×240 Quarter Video Graphics Array (QVGA), by writing functional register "0x12" with "0x14" [4]. Most of the other register configurations are referred from prior work [8].

After that, the camera enables to send images into the DUT through "VSYNC-HREF-DIN" interface. By receiving the valid pixels, three basic operations of the design on Image Capture slave are required: 1) collecting valid data input following the "VSYNC-HREF-DIN" timing shown in Fig. 5, two cycles for receiving each RGB565 pixel - 5-bit red, 6-bit green, and 5-bit blue; 2) converting the 16-bit RGB565 pixel into 12-bit data output - each 4 bits representing either a red, green, or blue pixel; 3) generating write data commands to store the frames of images into the Frame Buffers.

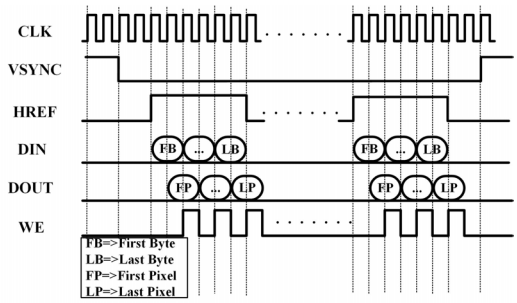


Fig. 5. Image Capture Timing

C. VGA Master Design

In what follows, a VGA master is needed to display videos on a VGA-interfaced monitor. As per the VGA timing standardized in [5], the Nexys 4 board uses 14 FPGA signals to create a VGA port with four bits-per-color and the two standard sync signals, horizontal sync ("HSYNC") and vertical sync ("VSYNC"). The "VSYNC" signal defines the "refresh" frequency of the display, or the frequency at which all information on the display is redrawn. The minimum refresh frequency is a function of the display's phosphor and electron beam intensity, with practical refresh frequencies falling in the 50HZ to 120Hz range.

The number of lines to be displayed at a given refresh frequency defines the horizontal "retrace" frequency. For a 640×480 display using a 25 MHz pixel clock and 60 +/-1Hz refresh, the signal timings shown in Fig. 6 can be derived. Timings for pre- and post-sync pulse times during which information cannot be displayed are named as front porch (FP) and back porch (BP). The porch interval between two lines or

TABLE I
VGA TIMING FOR 640 × 480 IMAGES

Symbol	VSYNC			HSYNC	
	Time (us)	Clock	Lines	Time (us)	Clock
H/VSP	16,700	416,800	521	32	800
H/VDIS	15,360	384,000	480	25.6	640
H/VPW	64	1,600	2	3.84	96
H/VFP	320	8,000	10	0.64	16
H/VBP	928	23,200	29	1.92	48

frames of images is the pulse width (PW). Together with the display (DIS) time, the total sync pulse (SP) can be computed as $SP = FP + PW + BP + DIS$.

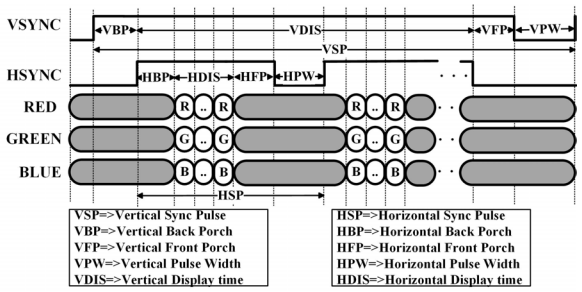


Fig. 6. VGA Timing

In particular, Table I shows the VGA system timing information of how a VGA-enabled monitor will be driven in the 640×480 mode. In this case the time interval of displaying one frame of image is $VSP = VFP + VPW + VBP + VDIS = 16.7$ ms. Hence, the refresh frequency of the display is $1/VSP = 59.88$ Hz.

D. Display Regions

Basically the window size of 640×480 is able to simultaneously display four 320×240 images. To identify different regions, two counters are created to realize the timing, "hcnt" for horizontal sync and "vcnt" for vertical sync. As shown in Fig. 7, when "hcnt" is in the cycles of $HBP \sim (HBP + 319)$, the region0 and region2 are selected. And the region1 and region3 are asserted when "hcnt" is in the cycles of $(HBP + 320) \sim (HBP + 639)$. Likewise, while the "vcnt" is in the cycles of $VBP \sim (VBP + 239)$, region0 and region1 are selected. And in $(VBP + 240) \sim (VBP + 479)$, region2 and region3 are selected.

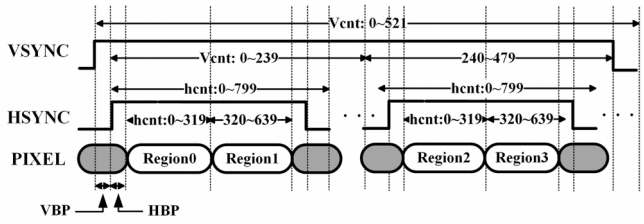


Fig. 7. Region Display Timing

As a case study, in region0 the original images are displayed. And in region1, region2, and region3, respectively, the black, grayscale, and binary images are displayed. Theoretically the grayscale images can be computed by

$$grayscale = 0.299 \times R + 0.587 \times G + 0.114 \times B \quad (1)$$

where R, G, and B represents pixels of red, green, and blue pixels, respectively. To simplify the design, Eq. 1 can be approximated as

$$grayscale = \{R, R\} >> 2 + \{G, G\} >> 1 + \{B, B\} >> 4. \quad (2)$$

Here, the operator "$\gg$" comes from the Verilog HDL Standard, meaning that performs right shift of the left operand by the number of bit positions given by the right operand. For example, "$R \gg 2$" indicates right shifting of the red pixel by 2 bits or multiplying each red pixel by 4. The operator "$\{\}$" represents the concatenation. "$\{R, R\}$" means that concatenates two 4-bit red pixel as an 8-bit red pixel, in order to increase the contrast of the grayscale pixel.

Based on the grayscale output, the binary converter can be simply designed as a 4-bit comparator. Assume that the threshold is hexidecimal "0x8". The grayscale pixel greater than "0x8" would be displayed as white ("0xf"). On the contrary, the grayscale pixel less than or equal to "0x8" would be displayed as black ("0x0").

E. IPs of PLL and Frame Buffer

In this data path, five IPs are generated using Vivado software, one PLL and four Block RAMs. As shown in Fig. 8(a), the PLL is used to distribute two clocks with 50MHz and 25MHz. The 50MHz clock is used by I2C master, and the 25MHz clock is applied by the VGA master, as summarized in Table II. The I2C serial clock produced by the I2C master is around 200KHz. Notice that the clock of Image Capture slave comes from the OV7670 Camera. It is measured as around 20MHz.

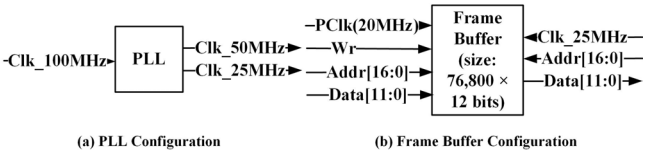


Fig. 8. IP Configuration

In Fig. 8(b), the Frame Buffer is configured with asynchronous "WRITE" and "READ" operations based on a $320 \times 240 \times 12bits = 76,800 \times 12bits$ Block RAM. The "WRITE" clock is the pixel clock ("PClk" around 20 MHz) from the Image Capture slave. And the VGA master performs the data read so the clock is "Clk_25MHz" from the PLL. For displaying four 320×240 images in parallel, totally four Frame Buffers are needed.

TABLE II
CLOCKS FOR THE DUT

Clock	Input	From	Freq. (MHZ)
Clk_100MHz	Clock Generator	FPGA	100
Clk_50MHz	I2C Master	PLL Output	50
Clk_25MHz	1) VGA Master 2) Frame Buffer (Clkb)	PLL Output	25
PClk	1) Image Capture Slave 2) Frame Buffer (Clka)	Camera OV7670	20
SIOC	Camera Register	I2C Master	0.2

V. DESIGN FOR VERIFICATION (DV)

In this section, six OVCs - including three bus function models and three scoreboards, are created for testing the functions of the data path. And then a typical simulation report generated by the scoreboards is discussed.

A. Open Verification Components (OVCs)

In order to drive or trace interconnected signals and IOs, three bus function models - the Capture master BFM, I2C slave BFM, and VGA slave BFM, are offered. As per the timing requirements discussed in Fig. 5, the Capture master BFM generates the control signals "VSYNC-HREF" and feeds in the images with an 8-bit "DIN" bus. Likewise, I2C slave BFM collects the serial data ("SIOD") at each rising edge of the clock ("SIOC") as shown in Fig. 4. And the VGA slave BFM captures red, green, and blue pixels by identifying the "VSYNC-HSYNC" valid signals as per the timing diagram shown in Fig. 6.

Additionally three scoreboards - the Image Capture scoreboard, I2C scoreboard, and the VGA scoreboard, for testing the functions and timings of the three design interfaces are provided. For example, each 24-bit data frame collected by the I2C slave BFM should be compared with the register configuration command, which is composed of the 8-bit ID, 8-bit register address, and 8-bit resister data. The Capture scoreboard is used to compare the Frame Buffer input with the data in golden_rgb444.txt file, and the VGA scoreboard is applied to check the VGA output with the data in golden_r.txt, golden_g.txt, and the golden_b.txt files.

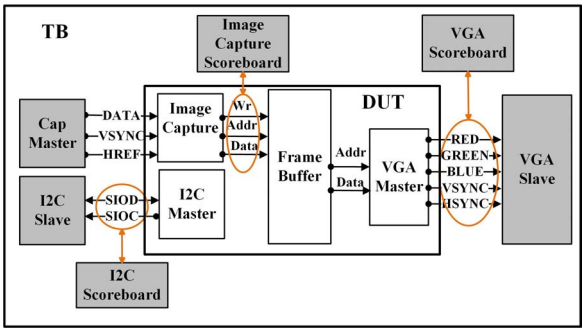


Fig. 9. Testbench with OVCs

B. Simulation Report

The DUT can be configured in simulation mode or synthesis mode by a control script shown in Fig 10. Notice that the five IPs discussed in IV-E will be only used in the synthesis mode. When the open source platform is configured as in the simulation mode, two equivalent models will be employed instead. Additionally the verification environment is also configurable through the tcl script. For example, the three scoreboards are able to be turned on and off by three definitions: "DEBUG_CAP", "DEBUG_I2C", and "DEBUG_VGA".

```
echo "+-------------------------------------"
echo "| Complile RTL Code of soc            "
echo "+-------------------------------------"
vlog +define+SIMULATION\
     +define+DEBUG_I2C\
     +define+DEBUG_CAP\
     +define+DEBUG_VGA\
     -f ../filelist/filelist.f
```

Fig. 10. DUT and Verification Environment Configuration

Fig. 11 shows the simulation report by turning on all the three scoreboards. It can be observed that the register configuration completes around 9ms for writing 56 registers. After that, the Capture master BFM enables to send pixels in RGB656 format, 8 bits per clock cycle. Then the DUT collects and converts the data from RGB565 mode into RGB444 type, and writes the 12-bit pixels into Frame Buffers. The Frame Buffer interface is monitored by Image Capture scoreboard. Meantime the 12-bit pixels are read out from the Frame Buffers, computed by image processing algorithms, and finally displayed as 4-bit red, 4-bit green, 4-bit blue in parallel via the VGA interface. The timing is checked by the VGA scoreboard.

```
# ----------------------------------------
# Start checking I2C Reg Configuration!
# ----------------------------------------
# Received command at 149885 ns is corect! Exp: 421280; Rcv: 421280
# Received command at 313745 ns is corect! Exp: 421280; Rcv: 421280
# Received command at 477605 ns is corect! Exp: 421214; Rcv: 421214
                    ......
# Received command at 8834465 ns is corect! Exp: 42b20e; Rcv: 42b20e
# Received command at 8998325 ns is corect! Exp: 42b382; Rcv: 42b382
# Received command at 9162185 ns is corect! Exp: 42b80a; Rcv: 42b80a
# ----------------------------------------
# Finish checking I2C Reg Configuration!
# ----------------------------------------
# ----------------------------------------
# Start checking CAP & VGA RGB output!
# ----------------------------------------
# Capture pixel at 16705100 ns is corect! Exp: rgb - ddc; Rcv: rgb - ddc
# Capture pixel at 16705200 ns is corect! Exp: rgb - ddc; Rcv: rgb - ddc
# Capture pixel at 16705300 ns is corect! Exp: rgb - ddc; Rcv: rgb - ddc
                    ......
# VGA pixel at 16704885 ns is corect! Rcv: red - e, green - e, blue - e
# VGA pixel at 16704925 ns is corect! Rcv: red - e, green - e, blue - e
# VGA pixel at 16704965 ns is corect! Rcv: red - e, green - e, blue - e
                    ......
# Capture pixel at 24397900 ns is corect! Exp: rgb - ddc; Rcv: rgb - ddc
# Capture pixel at 24398000 ns is corect! Exp: rgb - ddc; Rcv: rgb - ddc
# Capture pixel at 24398100 ns is corect! Exp: rgb - ddc; Rcv: rgb - ddc
# VGA pixel at 24332765 ns is corect! Rcv: red - d, green - a, blue - 6
# VGA pixel at 24332805 ns is corect! Rcv: red - e, green - b, blue - 7
# VGA pixel at 24332845 ns is corect! Rcv: red - f, green - b, blue - 8
#
# Finish checking CAP & VGA RGB output!
# ----------------------------------------
```

Register Configuration

Image Capture

VGA Display

Fig. 11. Simulation Report by Scoreboards

TABLE III
RESOURCE COST OF OUR WORK

Name	Slice LUTs	Slice Registers	RAM	IO
blk_mem0	134	11	26.5	0
blk_mem1	134	11	26.5	0
blk_mem2	134	11	26.5	0
blk_mem3	134	11	26.5	0
image_caputure	2	43	0	0
ov7670_controller	87	90	0	0
vga	134	100	0	0
Top	753	277	106	34

TABLE IV
RESOURCE COST COMPARISON

Resource Cost	[12,13]	[11]	Our work
Devices	Altera DE-115	Xilinx Virtex 5	Xilinx Nexys 4
Slice LUTs	1,616	10,283	753
Slice Registers	818	9,974	277
Block RAMs	-	52	106
IOs	94	-	34

VI. EXPERIMENTAL RESULTS

In this section, we employ Mentor Graphic ModelSim 10.4d as the simulator and Xilinx Vivado as the synthesis and implementation tool with the target device Nexys 4 FPGA.

A. FPGA Resource Cost

In Table III, the resource cost is summarized in terms of slice count, RAM utilization, and the number of IOs. As shown in the second and third columns, the number of slice LUTs and slice Registers are 753 and 277, respectively. In the fourth and fifth columns, it can be observed that 106 RAMs and 34 IOs are used by this platform.

Furthermore, we compare our design with the existing works in Table IV. The third column shows the resource cost on a design of image acquisition and processing using LabView FPGA [11]. It is obvious that the design consumes much more hardware resource compared to the RTL designs in the second and fourth columns.

[12], [13] implemented color to grayscale conversion and edge detection with VHDL on Altera DE-115. Compared with the resource cost shown in the second column, our proposed design reduces the slice of LUTs by 53.4% and the slice of registers by 66.1%, and consumes less than a half of the IOs. The reduced number of logics and IOs has a great potential to lower the switching activities of signals, resulting in less power consumption on FPGA development.

B. Power Consumption

Table V demonstrates the abbreviated breakdown of power estimation report generated by Xpower Analyzer [6], [7]. It can be observed that the static power consumption (SP) is 102 mW as shown in the eighth column and dynamic power cost (DP) is 118 mW with the total power consumption (TP) of 220 mW.

Due to the reduced-complexity design, the power dissipation decided by the toggle rate of clock, signals, logics, and IOs

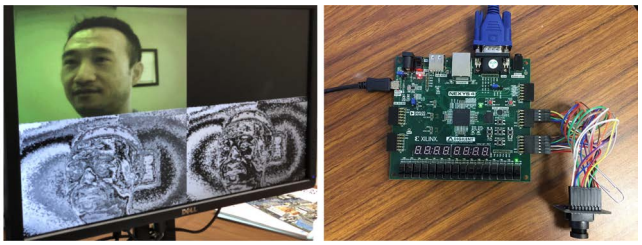
TABLE V
POWER CONSUMPTION ON NEXYS 4 FPGA

TP (mW)	DP (mW)						SP (mW)
	Clock	Signal	Logic	BRAM	PLL	I/O	
220	2	4	1	10	97	4	102 mW

is only 11 mW or 9.3% of the dynamic power cost, as shown in the second, third, and fourth columns. And the remaining power consumption is mainly come from the BRAM and PLL, totally 107mW or 90.7% of the dynamic power cost. The power cost has not been estimated in [11]–[13], it is thus not able to compare the power dissipation with the prior work.

C. FPGA Prototype

After programming the netlist on the Xilinx Nexys 4 FPGA, Fig. 12 demonstrates the application of displaying original video in Region0, and the processed images in grayscale and binary in Region2 and Region3 in parallel. In Region1, the displayed images are all black pixels.



(a) FPGA Results with Original Color Image in Region0, Black in Region1, Grayscale in Region2, and Binary in Region3

(b) FPGA with OV7670 Camera and VGA

Fig. 12. FPGA Prototype

VII. CONCLUSION

This paper presents a scalable image/video processing platform on FPGAs containing both open source design and verification environment. Compared to the prior open source projects, our work reduces the slice utilization and offers a potential to improve the power efficiency. More important, this platform is reusable and expandable to a diverse range of applications in image processing and computer vision on FPGAs. We expect that our public release of the entire implementation will lead to multiple designs in the future, and serve as a framework to projects of research and education.

REFERENCES

[1] H. He, et al., "Dual Long Short-Term Memory Networks for Sub-Character Representation Learning," The 15th Intl. Conf. on Information Technology - New Generations (ITNG-2018), Jan. 2018.

[2] L. Nwosu, et al., "Deep Convolutional Neural Network for Facial Expression Recognition Using Facial Parts," 15th IEEE Intl Conf. on Dependable, Autonomic and Secure Computing, Feb. 2018.

[3] A. Gajjar, et al, "An FPGA Synthesis of Face Detection Algorithm using HAAR Classifiers," Intl. Conf. on Algorithms, Computing and Systems (ICACS2018), PP.133-137, July 27-29, Beijing China, 2018.

[4] OV7670 Datasheet, Version 1.01, OmmiVision Technologies, Sunnyvale, CA, USA, 2005.

[5] Nexys 4 FPGA Board Reference Manual, Rev. B, Digilent, Sunnyvale, CA, USA, April 2016.
[6] X. Yang, et al., "A Novel Bus Transfer Mode: Block Transfer and A Performance Evaluation Methodology," Elsevier, Integration, the VLSI Journal, Vol. 52, PP. 23-33, Jan. 2016.
[7] X Yang, et al., "A low-cost and high-performance embedded system architecture and an evaluation methodology," 2014 IEEE Computer Society Annual Symposium on VLSI (ISVLSI), PP. 240-243, July 2014.
[8] M. Field (2013, Jan.). "http://www.hamsterworks.co.nz/mediawiki/index.php /Zedboard_OV7670."
[9] M. Birla, "FPGA Based Reconfigurable Platform for Complex Image Processing," 2006 IEEE Intl. Conf. on Electro/Information Technology, PP. 204-209, May. 2006.
[10] K. Jin, et al., "High-speed FPGA-GPU processing for 3D-OCT imaging," 3rd IEEE Intl. Conf. on Computer and Communications (ICCC), PP. 2085-2088, March 2018.
[11] S. Rahangdale, et al., "MBSEM image acquisition and image processing in LabView FPGA," 2016 Intl. Conf. on Systems, Signals and Image Processing (IWSSIP), PP. 1-4, July 2016.
[12] C. Ababei, et al., "Open source digital camera on field programmable gate arrays," Intl. Journal of Handheld Computing Research (IJHCR), Vol. 7, No. 4, PP. 30-40, 2016.
[13] C. Ababei, et al., "Open source digital camera on field programmable gate arrays," IEEE Intl. Conf. on Electro Information Technology (EIT), Grand Forks, ND, May 2016.