New Orleans JAZZED about Engineering Education

Paper ID #15811

Real-time System Implementation for Video Processing

Dr. Wagdy H Mahmoud, University of the District of Columbia

Wagdy H. Mahmoud is an Associate Professor of electrical engineering at the Electrical Engineering Department at UDC. Mahmoud is actively involved in research in the areas of reconfigurable logic, hardware/software co-design of a system on a chip using reconfigurable logic, application-specific integrated circuits (ASIC), digital logic design, image compressions, digital signal processing, computer architecture, embedded systems, system on a chip, and renewable energy.

Dr. Sasan Haghani, University of the District of Columbia

Sasan Haghani, Ph.D., is an Associate Professor of Electrical and Computer Engineering at the University of the District of Columbia. His research interests include the application of wireless sensor networks in biomedical and environmental domains and performance analysis of communication systems over fading channels.

Roussel Kamaha

Real-Time System Implementation for Video Processing

Abstract

This paper details the results of a capstone design project to develop a real-time hardware/software video processing system to implement Canny edge detection algorithm on a Zynq FPGA platform. The HSL tool, part of the Xilinx Vivado design environment, was used to develop the hardware accelerator for the algorithm. The paper will provide an overview of Canny edge detection algorithms, the details of the hardware/software system to detect edges in a real-time 1080p full HD input video stream, and the overall performance of the designed system. The paper also details the process used to help students complete such a complex project and their contributions on the professional development of students.

Keywords - High Level Synthesis, Canny edge detection, real-time video processing systems.

I. Introduction

Capstone design project are essential for preparing graduating students for a professional career. Student performances in these design projects and the quality of their produced designs are usually evaluated to access the level of students' attainment of the program educational outcomes. However, in technology-sensitive areas such as digital design, embedded processing and system-level engineering, software design packages are continually updated or replaced by more powerful, more efficient, and more complex design packages. Similarly, hardware platforms such as field programmable gate arrays (FPGAs) are continually improving in terms of their functionalities, and capabilities. Therefore, in our view, capstone design projects should not only be open-ended and realistic, but should also prepare student to work in rapidly-changing professional environment.

The ever-increasing complexity of digital systems including real-time video processing systems have compelled the Electronic System Level (ESL) community to switch over from Register Transfer Level (RTL) languages such as VHDL and Verilog to higher abstraction level languages such as C/C++ in order to reduce the overall design time and improve the productivity of system-level designers. Recent improvements in High Level Synthesis (HLS) tools has allowed for the development of highly-optimized video processing systems [1-2].

Real-time image and video processing applications employ computationally intensive algorithms that demand high computational power. High definition images or video frames contain an enormous number of pixels and large number of complex operations is usually performed on each of these pixels. It is not feasible to run video processing applications at a reasonable frame rate on general purpose computers due to their fixed memory structure, and limited processing power and input/output capabilities. A dedicated hardware/software system is needed to meet the strict real-time requirements of these applications [3].

Modern Field Programmable Gate Arrays (FPGAs) with high logic capacity, embedded multipliers and microprocessor are becoming the platform of choice for the hardware/software

co-design implementations of computational-intensive applications such as image and video-processing systems [3].

The capstone design presented in this paper aimed at designing a real-time hardware/software video processing system to implement Canny edge detection algorithm. Our students have utilized the Xilinx Integrated Synthesis environment (ISE) in some of their digital design and computer organization courses. They have also used some of the older Xilinx FPGA boards that are members of the Spartan or the Virtex 5 FPGA families. However, such the ISE design package was recently superseded by Xilinx Vivado Design suite that introduced additional features for embedded processing, system-on-a-chip development, and high-level synthesis (HLS). To make this design project challenging, students were required to use the HLS tool and to implement the design on a Zynq FPGA board.

To produce a superior system-level design that meets specifications, the designer need to have an intimate knowledge of: a) various algorithms pertaining to the design; b) the software packages needed to develop and verify the correctness of the design; c) the specific hardware platform that will be used to implement the design in the hardware; d) the coding language and its optimization techniques; and e) understand the use of Intellectual property (IP) components that can be used to speed up the application development process.

The organization of this paper is as follows. Section II provides an overview the student preparation process to help them successfully implement the design project. Section III provides a brief description of various edge detection algorithms. Section IV provides the main steps of Canny edge detection algorithm. Section V provides a brief description of The Xilinx Zynq-7000 SoC Video and Imaging Kit. Section VI provides a brief description of the Vivado high level synthesis (HLS) tool. Section VII describes the implementation issues of the design project. Section VIII is the conclusion and lessons learned section. Finally Section IX is the reference section.

II. Student preparation processes

The capstone design project spans over two successive semesters. The first semester is usually used to create design teams, develop proposals for the capstone projects, conduct literature survives related to the chosen projects, and, if needed, train student teams on the use of the software packages and hardware equipment and tools needed for the impregnation of their projects.

The preparation process of our students involved in the successful implementation of the hardware/software video processing system to implement edge detection application was as follows:

1. Literature survey: Student needed to conduct a massive literature survey of various edge detection algorithms, to find the pros and cons of the FPGA implementation of each algorithm, and to choose the most suitable one for their project. Requiring students to conduct targeted literature surveys enhances their long-life learning skills. A summary of the

results of the literature survey is given in the edge detection algorithm and the Canny edge detection algorithm sections of this paper.

2. Vivado high-level synthesis and the FPGA board training: As stated earlier, students were not familiar with the newly-released Vivado software package and its High-level synthesis tool or the structure of the Zynq board that is optimized for hardware/software applications. To remedy this situation, a large set of tutorial material and application notes ware provided to them [4 -7]. These materials covered various topics including the Vivado design suite and its HLS tool, the subset of the C code that can be synthesized using the HLS tool, HLS optimization techniques, and the Zynq FPGA family architecture in general and the details of the Xilinx Zynq-7000 SoC Video and Imaging Kit used in this project. A few weeks of intense mentoring on the use of both the software and hardware tools were also provided. At the conclusion of this training, student were able to master the use of software and hardware tools as well as the various coding optimization techniques used to implement various exercises available in the used tutorial [7]. Conducting such intensive training was necessary to ensure that our graduate students are able to use modern software and hardware tools that are used by professional system-level designers. Brief descriptions of each of the hardware and software tools used are available in sections five and six of this paper.
3. Real-time video processing issues: Students were directed to research issues related to the real-time implementation of a hardware/software video processing system. These issues include: a) the proper and efficient division of the implemented algorithm between the hardware and the software parts of the developed system; b) effective techniques for removing the noise present in still images and video frames; c) the difference between the contents of still image files and video frames and how to deal with additional information present in each video frame; and d) the development of efficient memory structure needed to store and process video frames. The implementation issues section provides brief description of the techniques, and/or IP tools used to successfully deal of each of these issues.

III. Edge detection algorithms

Real time image and video processing is a requirement in many computer vision applications, e.g. intelligent video surveillance and security, image/video coding, segmentation, compression, pattern recognition, object identification and tracking, feature extraction, traffic management and medical imaging. Edge detection is the most common preprocessing step in the implementation of the algorithms of video and image processing applications. Edge detection algorithms are used to detect the contour of objects present in an image and boundaries between these objects and the background. Edge detection has many practical applications such as computer guided surgery, medical diagnosis of tumors, and finger print recognition. The application of edge detection techniques can also significantly reduce the size of an image and filters out unnecessary information [9].

A large number of edge detection algorithms have been proposed. The majority of edge detection algorithms can be classified as either gradient-based (approximation of the first order derivative)

or Laplacian-based (approximation of the second order derivative/ zero-crossing detectors) approaches. Gradient-based edge detector algorithms include Robert, Prewitt, and Sobel. Second order derivative algorithms include Laplacian of Gaussian (LOG), zero-crossing, and Hough transform. Second-order derivative algorithm are not considered in this research due to their high sensitivity to noise [10].

Gradient-based edge detection algorithms have two kernels to detect edges in an image. For example, the Sobel operator uses two convolution kernels (vertical and horizontal) to calculate the edge intensity at each pixel and the rate of change in that direction [10]. The Sobel kernels are given by Eq. (1):

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad H_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{1}$$

where H_x is the horizontal kernel and H_y is the vertical kernel. The gradient magnitudes in the horizontal and vertical directions are then computed as follows:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A, \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \tag{2}$$

where * denoted a 2-dimentional convolution operator and A is the source image. The approximated magnitude of the edge intensity is then computed using Eq. (3).

$$|G| = |G_x| + |G_y| \tag{3}$$

The direction of the gradient, relative to the pixel grid, is calculated by Eq. 4.

$$\theta = tan^{-1}\left(\frac{G_y}{G_x}\right) \tag{4}$$

As given in [11], the C code implementation of the Sobel kernels to an image is as shown in Figure 1.

```
for(i = 0; i < height, i++){
 for(j=0; j < width; j++){
  x_dir = 0;
  y_dir = 0;
  if ((i > 0 && (i < (height-1)) && (j > 0) && (j < (width-1))){
   for(rowOffset = -1; rowOffset <= 1; rowOffset++){
    for(colOffset = -1; colOffset <=1; colOffset++){
      x_dir = x_dir + input_image[i+rowOffset][j+colOffset]*   Hx[1+rowOffset][1+colOffset];
    y_dir = y_dir + input_image[i+rowOffset][j+colOffset]* Hy[1+rowOffset][1+colOffset];
  }
 }
edge_weight = ABS(x_dir) + ABS(y_dir);
output_image[i][j] = edge_weight;
```

Figure 1. C code Implementation of the Sobel kernels

IV. Canny Edge Detection Algorithm

The Canny edge detector is one of the most widely-used edge detection algorithms due to its superior performance in many real-time applications, especially those processing noisy images. The original Canny algorithm has five different steps to detect edges:

1. Smoothing: remove noise from the image using a Gaussian filter. Denote the smoothed image as I.
2. Calculate the gradient magnitude and direction. In our implementation, the Sobel kernels were used. Replace the source image A with the smoothed image in Eq. 2 and compute $|G|$ and θ.
3. Apply non-maximal suppression to remove false edges. Only local maxima should be marked as edges.
4. Compute the high T_H and low T_L thresholds. Edge pixels stronger than T_H are marked as strong; edge pixels weaker than T_L are suppressed and edge pixels between the two thresholds are marked as weak.
5. Perform hysteresis thresholding to suppress all edges that are not connected to strong edges.

V. The Xilinx Zynq-7000 SoC Video and Imaging Kit

The main hardware parts of the zynq 7000 Soc video and imaging kit is the ZC702 evaluation FPGA board, a video I/O FMC card that supports HDMI I/O and an image sensor (Camera) that can be used to produce the input video stream. The kit also contain all video libraries and intellectual property (IP) components needed to implement a set of targeted reference designs. Parts of these designs were used to implement this project [4].

The FPGA chip on ZC702 board consists of an integrating processing system (PS) and programmable logic (PL). The board has a 1GB 32-bit wide DDR3 memory interface. The PS part includes two ARM Cortex-A9 processors, Advanced Microcontroller Bus Architecture (AMBA) Open Standard Interconnect, 512 KB of L2 cache, and 256 KB on-chip RAM. The PL part contains all the structures of a standard FPGA. The PS and PL portions are connected by an AMBA AXI interface. The PS potion has 560 KB of static RAM modules, called Extensible Block RAM (BRAM). Each BRAM module has 36kb and is designed as a true dual-port RAM. The large memories available in the FPGA chip allows for the fast processing of video applications.

VI. Vivado High Level Synthesis

In the past few years, many high level synthesis tools have been developed. The Xilinx Vivado HLS is reviewed as the state off the art tool for automating the generation of optimized low-level cycle-accurate register-transfer level (RTL) designs for applications written in C/C++. To HLS tool allows the designer to embed directives that control the generation of the RTL code that can be synthesized on a specific FPGA board. The most implemented HLS directives are those

dealing with loop pipelining and loop flattening. HLS directives are inserted onto the C program as pragmas. Loop pipelining allows loop iteration to overlap in the pipeline. Loop flattening unroll instructions inside a loop thus removing the overhead associated with checking the end of the loop at the beginning of each iteration. In our implementation, the Vivado HLS tool was used to a hardware accelerator for the computationally intensive part of the Canny algorithm that deals with pixel processing and detection of edges. The hardware accelerator runs on the PL part of the system. Operations dealing with receiving video streams and displaying the processed frames are implemented in the PS.

The HLS video library provide videos interfaces that was used to input and process videos frames. In addition to image pixels, each Video frame contain two other regions, namely vertical blanking and horizontal blanking. The vertical and horizontal blanking regions, respectively, contain the HS and the VS synchronization signals that detect and display the image pixels. The video library contains a soft IP core named AXI VDMA that is used remove the synchronization signals and provides high bandwidth direct memory access (DMA) between the FPGA on-chip memory and the AXI4 video stream. In our implementation, the edge detection system was designed to handle 1080p60 (1920 x 1080 pixel/frame at 60 frames/second). In this work, other cores from the video library such as the AXIvideo2Mat and Mat2AXIvideo were also used. The AXIvideo2Mat function was used to transform the image area of the frame into a matrix. The Mat2AXIvideo was used to display the processed matrix. Each pixel of the high quality video stream is stored in 4 pixels.

VII. Implementation Issues

The designers of this project have to face many implementation issues. T A brief discussion of each of these issues follows.

a) Greyscale conversion: To reduce the memory required to store each frame, the colored stream of frames is converted from RGB color space onto the Y'UV color space. In our implementation, this step is done using the video function HLS::cvtcolor. This function allows the user to define the format of the input and output image format. The function cvtcolor was used to convert the RGB color space into greyscale format.

b) Noise removal: As stated earlier, the first step in the Canny algorithm is to remove the noise from the image or video frame. A normalized 2-D Gaussian blur library function hls::GaussianBlur was initially used to remove the noise. The Gaussian convolution kernel used is 3 x 3 kernel given in Eq. (5)

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix} \tag{5}$$

c) Efficient memory structure: The code shown in Figure 1 shows that each pixel will be accessed multiple times. The convolution kernels given in equations 3-5 operates on local neighboring pixels. Therefore, memory structure that can retain data for multiple access must be created. The developed memory structure uses a combination of line buffers and window

buffers. A line buffer is used to store a complete line of the image and window buffers are 3 x 3 shift registers used to store the 9 neighboring pixels that are used in a specific operation. The window buffers move to the right and their contents are shifted to the left and are updated by the line buffers.

d) Optimization techniques: directives such as the loop and pipeline directives described earlier are embedded in the C code to unroll the loops and to pipeline the execution of the code. In our implementation, optimization codes were impeded in loops implementing the Gaussian blur, the Sobel kernels and the non. For example, the code given in Figure 1 is optimized as shown in Figure 2. Similar optimization directives were used in noise removal part of the code.

```
for(i = 0; i < height, i++)
  for(j=0; j < width; j++){
  #pragma HLS loop_flatten off
  #pragma HLS dependence variable=&buff_A false
  #pragma HLS PIPELINE II = 1
     . . .
```

Figure 2: Code optimization

The HLS automatically flatten perfect and semi-perfect loops. The loop-flatten off directive was used to turn off automatic flattening of the code. The HLS dependence variable=&buff_A false directive is used to remove dependency between read and write operations. The PIPELINE II = 1 directive is used to force the pipelined function to process a new input every one clock cycle [11].

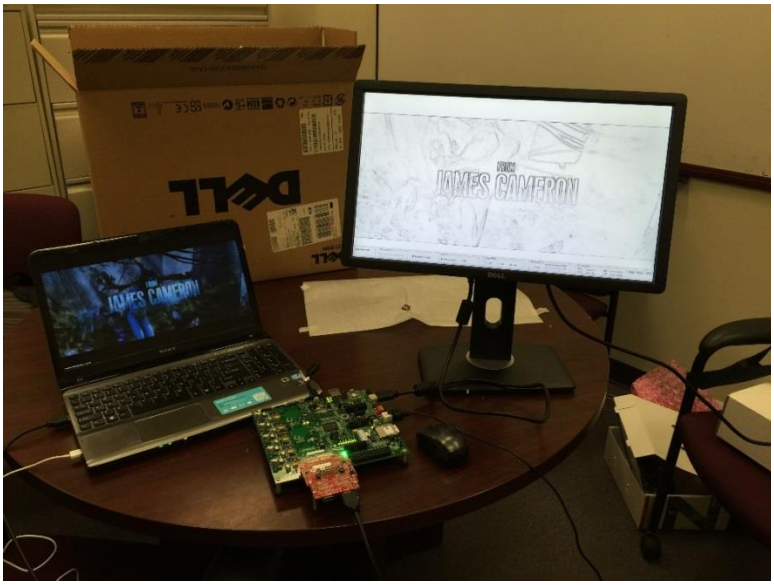

Figure 3. The designed edge detection system

VIII. Conclusion and lessons learned

This project provided valuable experience for both the capstone design team and the faculty mentor. Students were able conduct a targeted literature survey that taught helped them understand the project algorithm and master the use of many modern professional hardware and software, and IP tools. After countless hours of work, student were able to successfully develop the real-time edge detection system. The system was able to process 1080p60 (1920 x 1080 pixel/frame at 60 frames/second. A snap shot showing the working system is shown in Figure 3.The project mimicked a realistic work environment where designer are always have to adopt new design software and hardware tools. By changing the C code implemented, the project hardware and software can easily be used to develop a wide variety of computationally-intensive image processing applications.

IX. Acknowledgement

This work was supported in part by the U.S. National Science Foundation under Grant HRD-1435947

References

[1] Josh Monson, Mike Wirthlin, Brad L Hutchings: "Optimization Techniques for a High Level Synthesis Implementation of the Sobel Filter" 2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig), December 2013.

[2] J. Cong, S. Neuendorffer, J. Noguera, and K. Vissers, "High-level synthesis for FPGAs: From prototyping to deployment," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 30, no. 4, pp. 473–491, Apr. 2011.

[3] Yahia Said, Taoufik Saidani, Fethi Smach and Mohamed Atri," Real Time Hardware Co-simulation of Edge Detection for Video Processing System," 16th IEEE Mediterranean Electrotechnical Conference (MELECON), pages 852-855 ,2012.

[4] Xilinx UG925 (v7.0) Zynq-7000 All Programmable SoC ZC702 Base Targeted Reference Design (Vivado Design Suite 2014.2) User Guide, August 27, 2014.

[5] Xilinx UG926, Zynq-7000 All Programmable SoC: ZC702 Evaluation Kit and Video and Imaging Kit (ISE Design Suite 14.2) Getting Started Guide.

[6] Xilinx XAPP793, Implementing Memory Structures for Video Processing in the Vivado HLS Tool.

[7] Xilinx UG871, Vivado Design Suite Tutorial: High-Level Synthesis.

[8] Hong Nguyen. T. K., Cecile. Belleudy, and Tuan.V. Pham, "Performance and Evaluation Sobel Edge Detection on Various Methodologies, "International Journal of Electronics and Electrical Engineering Vol. 2, No. 1, March, 2014.

[9] M. Dinesh Chandra, Muni Praveena Rela, and M.Gurunadha Babu, "Edge Detection Algorithm for Video Processing System using FPGA," *International Journal of Emerging Trends in Electrical and Electronics (IJETEE), Vol. 10, Issue. 8, Sep. 2014.*

[10] Rashmi, Mukesh Kumar, and Rohini Saxena," ALGORITHM AND TECHNIQUE ON VARIOUS EDGE DETECTION: A SURVEY," Signal & Image Processing: An International Journal (SIPIJ) Vol.4, No.3, June 2013 .

[11] M. V. Fernando, C. Kohn, and P. Joshi, "Zynq all programmable SoC Sobel filter implementation using the Vivado HLS tool," vol. 890, pp. 1–16, 2012.