Published in final edited form as:

Proc IEEE Annu Northeast Bioeng Conf. 2015 April ; 2015: .

A FPGA Implementation of JPEG Baseline Encoder for Wearable Devices

Yuecheng Li¹, Wenyan Jia¹, Bo Luan², Zhi-hong Mao², Hong Zhang⁴, and Mingui Sun^{1,2,3}

¹Department of Neurological Surgery, University of Pittsburgh, Pittsburgh, PA 15213, USA

²Electrical & Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15213, USA

³Bioengineering, University of Pittsburgh, Pittsburgh, PA 15213, USA

⁴Image Processing Center, Beihang University, Beijing, 100191, China

Abstract

In this paper, an efficient field-programmable gate array (FPGA) implementation of the JPEG baseline image compression encoder is presented for wearable devices in health and wellness applications. In order to gain flexibility in developing FPGA-specific software and balance between real-time performance and resources utilization, A High Level Synthesis (HLS) tool is utilized in our system design. An optimized dataflow configuration with a padding scheme simplifies the timing control for data transfer. Our experiments with a system-on-chip multi-sensor system have verified our FPGA implementation with respect to real-time performance, computational efficiency, and FPGA resource utilization.

Keywords

JPEG; high level synthesis; health and wellness; efficient implementation; parallel computing; wearable devices

I. Introduction

In recent years, there have been rapid advances in wearable devices. Smart wristwatches and various physical activity monitors have entered people's daily life. Currently, most these devices do not have a camera. However, for numerous health and wellness applications, such as monitoring diet, evaluating sedentary events, and assisting the blind, a camera is required in wearable devices, such as the chest-worn eButton. This device is able to acquire images along with other data automatically [1]. In order to facilitate storage and wireless transmission, image compression is essential. The JPEG baseline codec, which is based on the 8×8 discrete cosine transform (DCT) and sequential encoding, has been the most implemented codec suitable for wearable devices [2].

In our work, a Zynq System-on-Chip (SoC) based wearable device with multiple cameras has been designed for health and wellness applications [3]. This device requires compressing

color images of 1280×960 pixels at up to 1 frame per second (fps) using the minimum FPGA resources. In this work, we describe our efficient implementation of the JPEG Baseline encoder with the use of the Xilinx HLS tool. Although the procedures in the JPEG Baseline encoder are still in the sequential form as in other forms of implementations, target real-time performance can be achieved with an optimized dataflow design. In order to minimize resource utilization, some data processing flexibility, such as adjustment of arbitrary image quality (only 4 quality levels are available) and reconfiguration of interleave modes, may be sacrificed.

II. Efficient Implementation

With the Vivado HLS tool, algorithms can be verified easily in C-code with software/hardware co-simulation and then converted to the Register Transfer Language (RTL) code instantly. Though certain coding styles is restricted, such as no run-time dynamic memory allocations and iterations with unpredictable cycles are allowed, styled coding on loops can be optimized automatically and data types with an arbitrary precision are supported. However, FPGA implementation with a specific performance criterion is still a difficult, non-intuitive task. The algorithm structure must be dissected thoroughly so that dataflow can be optimized according to the inherent parallelism which efficiently balances between real-time performance and resources utilization.

A. Dataflow Design

In JPEG codec, the Minimum Coded Unit (MCU) is defined as the smallest group of coded unit [2]. There are multiple interleave modes defined using the MCU to deal with color images and accommodate formats with different sampling rates for color image components. Therefore, the number of data units in the MCU is determined by the image sampling factors of components.

In order to obtain high computational efficiency, the source image in the YUV 4:2:2 format, commonly adopted by camera modules, is converted to the YUV 4:2:0 format. The MCU types were chosen to be H2V2 for the Y component and H1V1 for both the U and V components according to the different sampling factors. As a result, the data units in each MCU is 6 including 4 Y units, one U unit, and one V unit. The overall dataflow of this implementation is described in Fig. 1. For each MCU, six 8×8 data units are processed in a sequence of three steps, including the DCT, quantization, and Huffman coding.

Specifically, the unsigned integer image data are shifted, which is equivalent to subtract the value of 128, to produce signed integer data. Each 8×8 unit is fed to the Forward DCT (FDCT) to generate a single DC coefficient and 63 AC coefficients. Then, each DCT coefficient is quantized using a pre-specified quantization table. After quantization, the Huffman code is applied to the integer data, again using a predefined table (Huffman Table). Due to the strong correlation between DC coefficients of adjacent 8×8 units, the difference of the quantized DC coefficients in adjacent units is encoded separately from AC coefficients. The 63 quantized AC coefficients in each unit are encoded in a Zig-Zag order. This order ensures the low-frequency coefficients, which generally have larger values, are organized before high-frequency coefficients, which generally have smaller values. This

special ordering increases encoding efficiency. The pipeline optimization directives provided by the Vivado HLS tool are utilized here to implement the data flow.

At the output of the image compression stream, data padding, also designed in C-code, is adopted to simplify timing control of high speed AXI bus data transfer. At each activation of the AXI bus, 128 bytes of compressed data are transferred. A hexadecimal value of 0x00 is added to the end of the compression stream (after JPEG delimiter 0xFFD9) so that the total number of bytes is dividable by 128. In this way, the data transfer module can run without a complex timing control.

B. Optimizations and Tradeoffs

A shift operation is often required to determine the valid bits of a quantized coefficient. This operation is one the most time consuming procedures in JPEG coding. For example, a maximum of 16 clock cycles are required to examine a 16-bit coefficient. We refined this part by sembling parallel HDL code so that the significance of each bit is obtained in parallel and the number of valid bits for a coefficient is calculated by only single addition in the pipeline. This approach decreases the computational delay to only a single clock cycle, which accelerates the JPEG implementation significantly.

Allowing arbitrary precision in FPGA implementation helps save resources and is supported by the Vivado HSL tool. However, it can be applied only after co-simulation. Thus, for verification with co-simulation, we keep using byte-aligned precision in the original C code, and then simply switch to arbitrary precision through macro definitions. Assertion in C-code is also used to restrict iteration times when it can be derived explicitly, especially in Huffman coding.

For most images, the DCT coefficients in the first 8 frequency bands represent over 99.9% of the image energy [4]. Therefore, AC coefficients in higher frequency bands may be discarded without significant information loss. However, for images of the VGA size, we found that all 63 AC coefficients should be preserved to avoid visiable degradation on edges.

In addition, data bus width also affects the computation time. Because each 16 lines of source image data must be loaded into the FIFO buffer to initiate encoding, we adopt a 64-bit data bus width in implementation to decrease the computational delay caused by data loading in spite of the 32-bit width co-simulation, limited by the HLS tool.

III. Experiments

The real-time performance of our FPGA implementation was firstly assessed by a co-simulation with the 32-bit data format. For image of 1280×960, 707,255 clock cycles were used for compressing 16 lines of data. 40,435,300 clock cycles were used for the entire image which correspond to a rate of 2.35 fps at the 100MHz clock rate. As shown in Table I, only a minor portion of the FPGA resources were used.

Finally, we test our FPGA implementation on our Zyqn based multi-sensor system with four camera modules. A 64-bit bus was used which improved the frame rate further. A 0.5 fps

was achieved for the 1280×3840 image (which appends the four 1280×960 images). The frame rate increased to 11fps when the VGA (640×480) image was used as the input.

IV. Conclusion

In this paper, we have investigated the FPGA implementation of JPEG baseline encoder using the HLS tool for applications with wearable health monitoring systems. This implementation is highly efficient and the FPGA design are simplified significantly and become more flexible than the design using pure HDL programming. We believe that the same approach can be used to realize other system functions, such as denoising and adaptive compression rate control.

ACKNOWLEDGMENT

This work is supported by National Institutes of Health grants R01CA165255, and R21CA172864.

REFERENCES

- [1]. Mingui, Sun; Lora, E.; Burke; Zhi-Hong, Mao; Yiran, Chen; Hsin-Chen, Chen; Yicheng, Bai; Yuecheng, Li; Chengliu, Li; Wenyan, Jia. eButton: A Wearable Computer for Health Monitoring and Personal Assistance; Proceedings of the 51st Annual Design Automation Conference (DAC '14); ACM, New York, NY, USA. 2014. Article 16, 6 pages
- [2]. Wallace GK. The JPEG still picture compression standard. Consumer Electronics, IEEE Transactions on. Feb; 1992 38(1):xviii–xxxiv.
- [3]. Yuecheng, Li; Wenyan, Jia; Zhi, hong Mao; Hong, Zhang; Mingui, Sun. A Low Power, Parallel Wearable Multi-Sensor System for Human Activity Evaluation. 2015 submitted to NEBEC.
- [4]. Sangkeun, Lee. An Efficient Content-Based Image Enhancement in the Compressed Domain Using Retinex Theory. Circuits and Systems for Video Technology, IEEE Transactions on. Feb; 2007 17(2):199–213.

Fig.1.
Dataflow of JPEG Encoder Implementation

TABLE I

Resources utilization of FPGA implementation

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	1	0	337
FIFO	-	-	-	-
Instance	11	14	4739	6538
Memory	0	-	48	94
Multiplexer	-	-	-	230
Register	-	-	379	-
Total	11	15	5166	7199
Available	280	220	106400	53200
Utilization (%)	3	6	4	13