Development of Pre/Post-Integrated Image Processing Hardware by High Level Synthesis for Camera Sensor Node

Sota Miura, Akira Yamawaki*

Kyushu Institute of Technology, 1-1 Sensui, Tobata, Kitakyushu 804-8550, Japan

*Corresponding Author: yama@ecs.kyutech.ac.jp

Abstract

Camera sensor network is good candidate to prevent various crimes. In order to minimize the dead angle of each camera node as much as possible, the camera node is desired to be battery-driven so that it can be installed in an arbitrary place. However, since the conventional camera node consumes a large amount of power, the battery is emptied in hours to months. We plan to develop high performance and low power consumption camera sensor nodes with batteries lasting over 2 years. To realize such a camera node, we are developing a hardware that performs simple image processing and intermittent data transfer. In order to alleviate the burden of hardware development and respond quickly and flexibly to improvement of algorithms, we use high level synthesis technology, HLS which automatically converts software to hardware. In this paper, we developed hardware which integrated the image difference processing part and image compression part. The experimental results show the effect of integrating two parts compared with the divided hardware.

Keywords: high-level synthesis, camera network, camera node, low power, battery, Pre/Post-Integrated image processing.

1. Introduction

A wide variety of crimes are increasing in Japan. In order to realize a more secure and safe social life, the importance of surveillance cameras is increasing. It is desirable that the surveillance camera be battery-driven and easy to install in an arbitrary place so that the dead angle can be reduced as much as possible. Also, the surveillance camera itself is preferably inexpensive. The existing surveillance camera transfers a large amount of moving image data at high speed.

Therefore, surveillance cameras use wired internet or WiFi. In the transfer, complex image quality adjustment and compression processing with high power consumption are performed. Therefore, in many cases, a commercial power supply is used instead of a battery. To use commercial power supply requires installation work and installation cost. Because the existing battery-powered monitoring camera has high power consumption, the time available for one charge is very short, from several hours to six months⁽²⁾. Also, the recording time has severe restrictions such as about 5 minutes a day. This is not practical as a surveillance camera.

Therefore, this research aims to realize a battery-driven wireless camera sensor node that can be monitored for a long time. Conventional camera nodes use a dedicated large-scale system on chip that operates on clocks of GHz order which is continuously transferring moving pictures using complicated processing such as MPEG 4 and MJPEG. We develop simple image processing with simple algorithmic image compression and intermittent operation. Furthermore, we realize a simple algorithm developed using small and low operating frequency hardware. They will conserve power at the camera node.

In hardware development, we use high-level-synthesis, HLS so that we can respond quickly and flexibly to improve future algorithms. We have developed the image processing unit⁽²⁾ based on the background subtraction method for intermittent data transfer and the image compression unit⁽³⁾ using a simple compression algorithm (run length compression). In this paper, for further higher performance, we develop hardware that integrates them.

Experiment evaluates the performance of the integrated image processing hardware compared with software. Also, we compare the image difference processing and image compression built on the separated individual hardware

Fig.1. Conventional Camera Sensor Node

modules and show how much integrated effect is obtained.

2. Proposed Camera Sensor Network

2.1 Conventional Camera Sensor Node

Fig.1 (a) ⁽²⁻³⁾, show the operation of the conventional network camera, and Fig.1 (b) ⁽²⁻³⁾, shows the system of the conventional network camera. A conventional camera node acquires an image frame captured by a camera, performs complicated image processing, and encodes the moving image into MP 4, MJPEG, or the like. This process is relatively large. In this way, many manufacturers develop their own custom SoC (System-on-a-Chip). This SoC operates in GHz order to realize high-definition high-quality smooth movies. Basically, the camera node is always in operation and outputs the video stream to the network. In order to provide sufficient bandwidth to transfer high definition video in stream form it is necessary to use a high

Fig.2. Proposed Camera Sensor Node

speed and high power network like LAN or Wi Fi. Therefore, the camera node needs a commercial power supply. As a result, many camera nodes need physical wiring, so they cannot be easily installed anywhere. Also, a dedicated large-scale SoC that operates with a clock of GHz order is one of the factors that increase the power consumption of the network camera ⁽⁴⁾. Therefore, the time available for one charge is very short, from several hours to six months.

2.2 Proposed Camera Sensor Node

Fig.2(a), show the operation of the proposed network camera, and Fig.2(b) shows the network camera system proposed. The network is a wireless sensor network like Zigbee. In the proposed method, a difference image is created between the captured image and the previously captured background image. If the difference between the captured image and the background image is large, the camera node outputs the compressed difference image to the wireless sensor network. These image processes are simple and are performed by simple hardware operating in the order of 100 MHz. A host

```
void Diff_top(*back, *capt, *diff_img, n){
//Allocate to individual bus master port
#pragma HLS IF_m_axi port=back bdl=d0
#pragma HLS IF_m_axi port=capt bdl=d1
#pragma HLS IF_m_axi port=diff bdl=d0
// All functions operate in parallel
#pragma HLS DATAFLOW
// Inter-function array to FIFO
#pragma HLS STREAM variable=r
:
#pragma HLS STREAM variable=tmp_r

// Data input
dat_loader ( back, bk_r, bk_g, bk_b, n );
dat_loader ( capt, ct_r, ct_g, ct_b, n );

//Background subtraction
Makediff ( ct_r, bk_r, diff_r, n );
Makediff ( ct_g, bk_g, diff_g, n );
Makediff ( ct_b, bk_b, diff_b, n );

// Data output
Diff_img( diff_r,diff_g,diff_b, diff_img );
}
```

(a) Pre-processing Part

```
void Comp_top(*diff,*R,*G,*B, n){
//Allocate to individual bus master port
#pragma HLS IF_m_axi port=diff bdl=d2
#pragma HLS IF_m_axi port=R bdl=d1
#pragma HLS IF_m_axi port=G bdl=d2
#pragma HLS IF_m_axi port=B bdl=d3
// All functions operate in parallel
#pragma HLS DATAFLOW
// Inter-function array to FIFO
#pragma HLS STREAM variable=df_r
:
#pragma HLS STREAM variable=tmp_r

// Data input
dat_loader ( diff, df_r, df_g, df_b, n );

// Runlength compression
run_len_enc(df_r, n, tmp_r );
run_len_enc(df_g, n, tmp_g );
run_len_enc(df_b, n, tmp_b );

// Data output
dat_output( tmp_r, R );
dat_output( tmp_g, G );
dat_output( tmp_b, B );
}
```

(b) Post-processing Part

Fig.3. Image Processing Software

Fig.5 Hardware Organization

such as a PC performs complicated and high power consumption processing such as restoration and reproduction using a small amount of transmitted data. Since network transfers are intermittent and hardware processing at

Synthesis) is used so that it can respond quickly and flexibly to improvements in future algorithms. However, the HLS tool tends to generate poor hardware unless careful attention is paid to the hardware configuration. In this paper, we describe a preprocessing unit that creates a difference image between a captured image and a previously captured background image and a post-processing unit that compresses the image based on the simplest compression algorithm (run length compression) Make it with software.

Fig.3(a) is a program of the pre-processing unit which creates a difference image between the captured image and the previously captured background image. The input argument capt and the output argument diff in Fig.3(a) are assigned to the same AXI bus port m_axi_d0 by the pragma directive. The input argument back is assigned to another port m_axi_d1. The hardware can achieve ideal performance with 1 pixel per clock, since the two input images are retrieved consecutively and then the output image is stored sequentially via the pipelined data path.

Fig.3(b) is a program of a post-processing unit which compresses an image based on the simplest compression algorithm (run length compression). Each argument (diff, R, G, B) is assigned to each AXI bus port, and compression processing and memory writing are performed in parallel. Specifically, each channel of RGB simultaneously executes run length compression and memory writing in parallel, and processing from image input to compression and memory writing is smoothly performed.

The data input (dat_loder), the difference processing unit

```
void Diff_comp_top(*back, *capt, *R,*G,*B, n){
//Allocate to individual bus master port
#pragma HLS IF m_axi port=back bdl=d0
#pragma HLS IF m_axi port=capt bdl=d1
#pragma HLS IF m_axi port=R bdl=d0
#pragma HLS IF m_axi port=G bdl=d1
#pragma HLS IF m_axi port=B bdl=d2
// All functions operate in parallel
#pragma HLS DATAFLOW
// Inter-function array to FIFO
#pragma HLS STREAM variable=r
#pragma HLS STREAM variable=tmp_r

// Data input
dat_loader ( back, bk_r, bk_g, bk_b, n );
dat_loader ( capt, ct_r, ct_g, ct_b, n );

//Background subtraction
Makediff ( ct_r, bk_r, diff_r, n );
Makediff ( ct_g, bk_g, diff_g, n );
Makediff ( ct_b, bk_b, diff_b, n );

// Runlength compression
run_len_enc(diff_r, n, tmp_r );
run_len_enc(diff_g, n, tmp_g );
run_len_enc(diff_b, n, tmp_b );

// Data output
dat_output( tmp_r, R );
dat_output( tmp_g, G );
dat_output( tmp_b, B );
}
```

Fig.4. Pre/Post Integrated Image Processing Software

moderate clock frequencies is simple and small, power consumption can be greatly reduced as compared with conventional camera nodes.

3. Pre/Post Integrated Image Processing Hardware

In hardware development, HLS (High-Level-

(Makediff), the difference image creation unit (Diff_img), the run length compression (run_len_enc) and the data output (dat_Output) shown in Fig.3(a)(b) are pipelined.

Fig.4 shows a program in which the pre-processing unit and post-processing unit are integrated. Each argument (back, capt, R, G, B) is assigned to each AXI bus master port, and differential processing, compression processing, and memory writing are simultaneously executed in parallel. That is, we wrote the software so that the hardware configuration will be as shown in Fig.5. Specifically, each channel of RGB simultaneously executes difference processing, run length compression and memory writing in parallel, and processing is performed smoothly from image input to memory writing for each of RGB. With this reconfiguration, hardware is expected to achieve ideal performance with 1 pixel per clock. This is because two input images are taken out consecutively, difference processing and compression processing are concurrently performed concurrently for each of the RGB channels, and the output images are successively transferred via a pipelined data path. It can be seen that there is no process for creating a difference image as compared with the case where the pre-processing unit and the post-processing unit are not integrated.

4. Experiment

4.1 Experimental Setup

In order to compare and evaluate the hardware impact of the integrated software description and the divided software description, we developed a hardware platform on the FPGA. The used FPGA is Xilinx ZYNQ FPGA on Digilent ZYBO. The hardware modules run at 100MHz. The used HLS tool is Vivado HLS 2016.4. The FPGA implementation tool is Vivado 2016.4. In addition, we compare with the execution

time of the software on the embedded processor, ARM Cortex-A9 at 650MHz.

To measure the execution time of the software and hardware, we have equipped the performance counter running at 100MHz as a memory mapped register on the FPGA.

4.2 Experimental result

The experimental results are shown in Fig.6 The size of the photographed image and background image used is 1280 [W] × 960 [H]. Each pixel is color of 32bit including 8bit R, G and B.

The software execution of the embedded processor at 650 MHz takes 81.01 ms.

The HLS hardware converted from the divided software shown in Fig.4 takes 24.58 ms. This hardware improves the performance by 3.3 times compared to the execution of software.

The HLS hardware converted from the integrated software shown in Fig.5 takes 12.29 ms. This hardware improves performance by 6.6 times compared to software execution. As a result, it improves twice the performance compared to the HLS hardware converted from the split software. This is because processing from the input of the photographed image and the background image to the difference processing, the compression processing, and the memory writing are continuously processed without stopping the data path pipeline. In addition, in the case of divided software, it is thought that the difference image is made after the difference image processing of the input image and written once into the external memory, which is twice as slow as compared with the case of integration.

4.3 Power Efficiency

When the pre-processing unit and the post-processing unit are integrated, the performance of the HLS hardware can be greatly improved. The embedded processor used operates at a clock frequency of 650 MHz. In contrast, the hardware module operates at 100 MHz. Therefore, the embedded processor operates 6.5 times faster than the hardware module. Generally, the faster the clock frequency, the higher the

Fig.6. Experimental Result

$$\begin{aligned}
 PPE &= \frac{\text{Software exec. time} \times \text{Ave. power of processor}}{\text{Hardware exec. time} \times \text{Ave. power of hardware}} \\
 &= \frac{\text{Software exec. time} \times \text{Clock freq. of processor}}{\text{Hardware exec. time} \times \text{Clock freq. of hardware}} \\
 &= \frac{\text{Software exec. time} \times 650\text{MHz}}{\text{Hardware exec. time} \times 100\text{MHz}} \\
 &= \frac{\text{Software exec. time}}{\text{Hardware exec. time}} \times 6.5 \dots\dots\dots (1)
 \end{aligned}$$

power consumption. From this point of view, we attempt to

estimate the performance-power-efficiency, PPE by the following equation.

The first expression means the improvement of the power consumption of the hardware execution to the software execution during the processor and hardware are executing their own processing. It is well known that the average power of logic circuits based on CMOS technology is proportional to the clock frequency ⁽⁴⁾. Thus, we approximate

Fig.7. Performance-Power Efficiency of Hardware to Software

the power consumption during execution by using only clock frequency shown as the second expression. Finally, we can get Eq. (1).

Fig.7 shows the performance-power efficiency calculated by Eq. (1). This result shows that both HLS hardware created from divided software and HLS hardware created from integrated software improve performance efficiency for software execution. Furthermore, in the case of integration, it is possible to achieve twice as much as the case of division and 42.8 times for software execution.

5. Conclusions

We are aiming to realize a battery-driven camera sensor node that can operate for a long time. In this paper, we have created a difference processing unit which is a preprocessing unit of image processing in such a camera node and a compression processing unit based on an algorithm of run length compression which is a post processing unit. Then, hardware that integrated the preprocessing part and post-processing part was developed by HLS. As a result of describing the software considering the hardware configuration in which the RGB compressed channel operates in parallel from the image input, the integrated hardware can be expected to save power by 42.8 times compared with software which simply describes the algorithm. Also, it can be expected to save power twice as

much as the divided hardware.

Future works include introducing wireless communication, searching for optimal algorithms, and power evaluation using real cameras.

References

- (1) Emily Holmes, "Battery Powered Security Cameras Compared", <http://securitysystemscompare.com/battery-powered-security-cameras-compared/>, 2017. [Accessed: 22- Feb-2018]
- (2) Sota Miura, Ayumi Mise and Akira Yamawaki: "Development of Image Processing Hardware by High-level Synthesis for High Performance and Low-power Camera Sensor Node on Wireless Sensor Network", Proceedings of the 6th IIAE International Conference on Industrial Application Engineering 2018, pp.210-214 (2018.3)
- (3) Sota Miura and Akira Yamawaki: "Development of Image Compression Hardware by High level Synthesis for Wireless Camera Sensor Node ", The 33rd Int'l Technical Conf. on Circuits/Systems, Computers and Communication (ITC-CSCC2018), pp.290-293(2018.7)
- (4) Sung-Mo Kang, Yousef Leblebici, "CMOS DIGITAL INTEGRATED CIRCUITS: Analysis and Design", pp.462-463(1998/9/16) William C Brown Pub; 2nd version