Development of Image Processing Hardware by High-level Synthesis for High Performance and Low-power Camera Sensor Node on Wireless Sensor Network

Sota Miura, Ayumi Mise, Akira Yamawaki*

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

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

Abstract

To prevent many kinds of crimes, the camera sensor network is taken attention. The battery-driven camera node that can be equipped to any place is desired in order to reduce the blind spot as much as possible. However, the battery life is very short as few hours to few months because the conventional camera node requires the high power consumption. We plan to develop a high-performance and low-power camera sensor node with the battery life of two or more years, which is mandatory as the wireless sensor node driven by battery. To achieve this requirement, we are developing a simple image processing with intermittent action performed by the camera node while the conventional camera nodes tend to realize the real-time smooth movie with high resolution by the complex algorithm. We develop a simple image processing based on the background subtraction by the fully customized hardware instead of the software realization on the embedded processor or GPU. For developing the hardware, we use the high-level synthesis technology, HLS, which converts the software to the hardware automatically, in order to reduce the design burden of the hardware and quickly follow to the algorithm improvements. However, HLS tools tend to generate a poor hardware, if without deep care about the hardware organization. This paper demonstrates the software description of the background subtraction for HLS tool. The experimental result shows the effect of our reconstruction of the software program for HLS tool.

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

1. Introduction

To prevent many kinds of crimes, the camera sensor network is taken attention. The battery-driven camera node ⁽¹⁾ that can be equipped to any place is desired in order to reduce the blind spot as much as possible. However, the battery life is very short as few hours to six months ⁽¹⁾ because the conventional camera node requires the high power consumption. This is because the conventional cameras tend to perform the complex compression algorithms like mpeg4 and motion-jpeg to make the movie very smooth with high resolution on the large custom LSI with high clock frequency of GHz order ⁽²⁾. The battery powered cameras with 6 months to two years force to unrealistic constraints as surveillance camera. These constraints claim that recoding time has to be within 5 seconds to 5 minutes per day ⁽³⁻⁴⁾.

We plan to develop a high-performance and low-power camera sensor node recording a long period with the battery life of two or more years, which is mandatory as the wireless sensor node driven by battery. To achieve this requirement, we are developing a simple image processing with intermittent action performed by the camera node while the conventional camera nodes tend to realize the real-time smooth movie with high resolution by the complex algorithm.

We develop a simple image processing based on the background subtraction ⁽⁵⁾ by the fully customized hardware instead of the software realization on the embedded processor or GPU. For developing the hardware, we use the high-level synthesis technology, HLS ⁽⁶⁻⁸⁾, which converts the software to the hardware automatically, in order to reduce the design burden of the hardware and quickly follow to the

Fig. 1. Conventional Camera Sensor Network

algorithm improvements.

However, HLS tools tend to generate a poor hardware, if without deep care about the hardware organization. This paper demonstrates the software description of the background subtraction for HLS tool. The experimental result shows the effect of our reconstruction to the software program for HLS tool.

2. Camera Sensor Network

2.1 Conventional Camera Sensor Network

Fig. 1 shows a concept image of the conventional camera sensor network. Each camera consumes large power to perform the complex movie compression at run-time. The high speed and high power network like LAN and Wi-Fi must be used to provide enough bandwidth to transfer the high resolution video streamly. Thus, the camera node requires the commercial power. As a result, many camera nodes cannot be easily equipped to any places due to physical wiring is required.

Fig. 2. Proposed Camera Sensor Network

2.2 Proposed Camera Sensor Network

Fig. 2 shows a concept image of the proposed camera sensor network. The network is a wireless sensor network like Zigbee. Each sensor node is driven by the battery. Thus, the camera node can be equipped to any places since the physical wiring is not required. Since the battery life is extended to two or more years, the maintenance cost due to the battery changing will be reduced significantly.

Since wireless sensor network generally supports low-bandwidth network like 250kbps, the camera node has to perform a smart image processing based on the simple algorithm on the custom hardware in order to handle the intermittent video transferring whose quality is enough to correctly observe the place equipped.

2.3 Qualitative Difference

Let us qualitatively explain the differences between the conventional camera node and proposed camera node to clarify the novelty of our proposal more by using Fig. 3.

Fig.3. Difference of Execution Behavior between Proposal and Conventional

The conventional camera node acquires the image frame captured by the camera and performs the complex image processing to encode the movie images to MP4, MJPEG, and so on. This processing is relatively large. Thus, many makers have developed its own custom system-on-a-chip, SoC. This SoC runs at GHz order to realize high-quality smooth movie with high resolution. Basically, the camera node is constantly running and outputting the video stream into the network.

The proposed one does not take such strategy. At first, the proposed camera node makes the difference image between the captured image and the background image previously captured. If the difference between the capture and background images is large, the camera node outputs the compressed difference image into the wireless sensor network. These image processing are simple and performed by the simple hardware running at 100 MHz order. The host restores the captured image by adding the difference image sent by the camera node with the background image which is same as the camera node. Since the network transfer is intermittent and the hardware processing at a moderate clock frequency is simple and small, the power consumption may be significant reduced compared with the conventional camera node.

3. Background Subtraction for HLS

3.1 Pure Software with Pragma

We first develop a simple background subtraction shown in Fig.4. We know this simple method is not withstanding use. However, we think it is significant that we start with the simple algorithm then extend to complex algorithms incrementally to find appropriate algorithm meeting to our strategy. This strategy is to replace the conventional complex processing on the large high speed SoC to the simple moderate speed low power hardware module implementing an optimum simple algorithm. In addition, to find an efficient description method which makes the HLS tool generate a

```
MakeDiff(*capt,*back,*diff,num){
#pragma HLS INTERFACE m_axi port=capt bundle=d
#pragma HLS INTERFACE m_axi port=back bundle=d1
#pragma HLS INTERFACE m_axi port=diff bundle=d
for(i=0;i<num;i++){
#pragma HLS PIPELINE II=1
tmp=capt[i]-back[i];
if(tmp>-T && tmp<T) tmp=0;
diff[i]=tmp;
}
}
```

Fig. 4. Pure Software with Pragmas

good hardware module of the background subtraction is one of the objectives.

Since the image data is large, the image data is stored into the large memory externally attached to the hardware module. Thus, we insert the pragma inferring the AXI bus interface accessing to the external memory into the top of the function. In addition, we insert the pipeline pragma into the loop to infer the pipelining of the loop iteration. These pragmas specify the automatic hardware reconstruction to the HLS tool. That is, the responsibility of generating well-organized hardware is shifted to the HLS tool while remaining the pure software structure without considering the hardware organization.

3.2 Software Considering Hardware Organization

To make the HLS tool a good hardware module, the designer must consider the hardware organization. In the case of Fig. 4, the characteristics of AXI bus interface should be taken care to improve the performance of the hardware module.

The AXI bus provided by ARM is one of the on-chip bus standards. It becomes the industrial standard. Almost FPGA and SoC using ARM processor employs AXI bus. A master port of AXI bus has individual read and write channels which are running in parallel.

Considered this characteristic of AXI bus, the input arguments for the capture and background image of Fig. 4, **capt** and **back**, should be assigned to individual AXI bus ports as shown in Fig. 5(a). This is because the capture image

```
MakeDiff(*capt,*back,*diff,num){
#pragma HLS INTERFACE m_axi port=capt bundle=d0
#pragma HLS INTERFACE m_axi port=back bundle=d1
#pragma HLS INTERFACE m_axi port=diff bundle=d0
for(i=0;i<num;i++){
#pragma HLS PIPELINE II=1
tmp=capt[i]-back[i];
if(tmp>-T && tmp<T) tmp=0;
diff[i]=tmp;
}
}
```

(b) Software Description Corresponding to (a)

Fig. 5. Software Considering HW Organization

and background image are loaded continuously without port contention. In addition, the output argument of Fig. 4, **diff**, can be assigned to either input port. In Fig. 5, the **diff** is assigned to that of **capt**.

To realize the hardware organization shown in Fig. 5 (a), we reconstruct the software description as shown in Fig. 5 (b) from Fig. 4. The **capt** and **diff** are assigned to the same AXI bus port, **m_axi_d0** by the **INTERFACE** pragma. The **back** is assigned into another port, **m_axi_d1**. By this restructure, the hardware is expected to achieve ideal performance at 1 pixel per 1 clock. This is because two input images are fetched continuously, and then the output image is stored streamly through the pipelined data path.

4. Experiment and Discussion

4.1 Experimental Setup

To perform the comparative evaluation clarifying the effect of our restructuring to the software description, we developed a hardware platform on an 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. The software was compiled by GCC with `-O2` on the Xilinx SDK launched from Vivado 2016.4.

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 Performance Evaluation

Fig. 6 shows the experimental result. The size of the captured and background images used is 1280 [W] × 960 [H]. Each pixel is color of 32bit including 8bit R, G and B.

Fig. 6. Experimental Result

The software execution on the embedded processor at 650MHz takes 51.45 ms. The HLS hardware converted from the pure software without considering the hardware organization shown in Fig. 4 takes 500.6 ms. This hardware degrades the performance significantly compared with the software execution. This fact indicates that the HLS tool certainly cannot generate a good hardware module without consideration of the hardware organization at software level.

In contrast, the execution time of the hardware generated from the restructured software shown in Fig. 5 is 12.29 ms. As a result, our restructuring improves the performance of 41 times compared with the hardware without any restructure. Finally, our hardware can achieve 4.2 times performance improvement compared with the software execution. This is because the memory loading from the captured and background images are streamly performed without stalling the data path pipeline.

4.3 Power Efficiency

Our restructuring can improve the performance of HLS hardware significantly. The embedded processor used runs at 650MHz clock frequency. In contrast, the hardware module runs at 100MHz. Thus, the embedded processor is running 6.5 times faster than the hardware module. Generally, faster clock frequency, larger power consumption. From this aspect, we attempt to roughly estimate the performance-power efficiency, **PPE**, by following expressions.

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

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 the logic circuit by the CMOS technology is proportional to the clock frequency ⁽⁹⁾. Thus, we approximate 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) using the measured values shown in Fig. 6. The result indicates that the straight-forward hardware with only pragmas cannot improve the performance-power

Fig. 7. Performance-Power Efficiency of

Hardware to Software

efficiency to the software execution. In contrast, our restructuring can achieve 27.2 times performance-power efficiency compared with the software execution.

5. Conclusions

The battery-driven camera node that can be equipped to any place is desired to equip to any places easily. However, the battery life is very short as few hours to few months because the conventional camera node requires the high power consumption. We plan to develop a high-performance and low-power camera sensor node with the battery life of two or more years. To achieve this requirement, we are developing a simple image processing based on the background subtraction with intermittent action performed by a fully customized hardware.

For developing the hardware, we use the high-level synthesis technology, HLS, which converts the software to the hardware automatically. However, HLS tools tend to generate a poor hardware, if without deep care about the hardware organization.

This paper demonstrates the software description of the background subtraction for HLS tool. The experimental result shows the effect of our reconstruction of the software program for HLS tool

As future work, we will develop hardware modules by using HLS technology to realize the proposed camera-node.

References

- (1) Emily Holmes, "Battery Powered Security Cameras Compared", <http://securitysystemscompare.com/battery-powered-security-cameras-compared/>, 2017. [Accessed: 22- Feb-2018]
- (2) Takahau Tanaka and Kousuke Yoshioka, "UniPhier

System LSIs and Their Core Technologies", Panasonic Technical Journal Vol. 58, No. 1, pp. 36-40, 2012 in Japanese.

- (3) NETGEAR.com, "How long do Arlo camera batteries last?", <https://kb.arlo.com/1202753/How-long-do-Arlo-camera-batteries-last>, 2017, [Accessed: 22- Feb-2018].
- (4) Blink, "How long do the camera batteries last?", <http://support.blinkforhome.com/customer/en/portal/articles/2104091-how-long-do-the-camera-batteries-last>, 2017, [Accessed: 22- Feb-2018].
- (5) Birgi Tamersoy, "Background Subtraction - Lecture Notes", University of Texas at Austin, https://www.cs.utexas.edu/~grauman/courses/fall2009/slides/lecture9_background.pdf, 2009, [Accessed: 22- Feb-2018].
- (6) R. Nane, V.-M. Sima, B. Olivier, R. Meeuws, Y. Yankova, and K. Bertels, "DWARV 2.0: A CoSy-based C-to-VHDL Hardware Compiler", In FPL, pp. 619-622. 2012.
- (7) C. Pilato and F. Ferrandi, "Bambu: A Modular Framework for the High Level Synthesis of Memory-intensive Applications", In FPL, pp. 1-4, 2013.
- (8) A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, J.H. Anderson, S. Brown, and T. Czajkowski, "LegUp: High-Level Synthesis for FPGA-based Processor/Accelerator Systems", In ACM FPGA, pp. 33-36. 2011.
- (9) Sung-Mo Kang, Yousef Leblebici, CMOS DIGITAL INTEGRATED CIRCUITS: Analysis and Design, pp. 462-463 (1998/9/16) William C Brown Pub; 2nd version