Enhanced transplantability for smart agriculture drone by an FPGA with SD standards

Ryo Nakagawa[1,a)] Yoshiki Yamaguchi[2,b)]

Abstract: Currently, people face food shortages because of the population and demand increase. Therefore, efficient and sustainable agriculture are required. Agriculture is gradually putting the Internet of Things (IoT) technology into practical use to respond to this new demand. For example, some companies have started to introduce agricultural drones into practical use and raise product awareness. However, after the verification stage, they found out some problems under the practical use. One is the transplantability of the IoT agriculture system, and the other is the energy performance. Here, we propose an FPGA-based drone system for agriculture applications to address their solutions. The main task in agricultural drones is image acquisition and processing, so enabling onboard processing will save time to retrieve images and improve usability. However, image processing requires higher performance, and available electric capacity is limited on the drone system. Therefore, FPGA is a good candidate which implements only necessary functions to enhance power performance, compared to a general-purpose CPU or GPU. In addition, considering various farms and for multiple crops, the flexibility of the processing with FPGA is necessary. Moreover, a general-purpose digital still camera is selected as its image sensor to archive high-cost performance and transplantability. Since most consumer digital cameras don't have an interface dedicated to image transmission, the SD card interface is used. This paper describes how to emulate an SD card on FPGA to acquire image data from a digital camera. Firstly, from the SD specification, the processing required to emulate the SD card is described. Then, the implementation of the FAT file system is discussed so that the system behaves as storage. To demonstrate the proposed data acquisition mechanism, we combined a module that emulates an SD card, provides virtual FAT file system access, and a DRAM that stores the received data. These were implemented on a Xilinx ZYNQ SoC. Finally, we input the extended signal from the SD card slot of the digital camera and confirmed that the implemented system was recognized as an SD card by the camera.

Keywords: agricultural drone, FPGA, SD standards

1. Introduction

Smart farming, which utilizes IoT technology in agriculture to improve productivity, can play an essential role in addressing the issue of the future food shortage in the world. For example, as readily available technology compared to satellites or aircraft, drones are becoming an essential device because of grasping the situation of a targeted farm field [1]. There are several attempts to utilize drones in agriculture. Drones equipped with a single-view camera are used to reconstruct the 3D shape of crop field from multiple images by SfM (Structure from Motion) and estimate the crop height (e.g., [1], [2]).

Dedicated designs lead drones to success (e.g., [1] targets on the rice). However, it sometimes reduces the usability instead of enhancing the accuracy and capturing additional information. For example, sophisticated and complicated image processing requires enormous efforts and is challenging to do on a drone. The offline processing will require an additional flight when the quality of photos is insufficient for the analysis. In addition, Japanese agriculture tends to grow various crops on a small farm, compared to Europe or the U.S, so adaptability to a wide variety of produce is required.

In this paper, we discuss the utilization of drones as the means of intelligent farming in Japan. FPGA is a good candidate for direct hardware computing and circuit-level reconfigurability to enhance the adaptability to a wide range of harvests. The user flies drones to get images while watching the simplified and real-time measurement results, so real-time processing and transmission are required. After recovering the drone, a detailed post-flight analysis of the acquired images is conducted. Consequently, there is a demand for image data storage. In addition, there is a limitation on an available battery capacity due to the drone's size; fast computation with less power consumption is a better solution. Therefore we defined the requirements for our drone system as follows:

- Real-time processing (usability)
- Communication equipment to download the result data (usability)
- Image data storage for post-flight analysis (usability)
- Reconfigurable processing (adaptability)
- Low computational power consumption (drone)

This article proposes an agricultural drone equipped with FPGA and a commodity digital camera to fulfill the above requirements. As described in section 2 below, the SD standard interface is used in our proposed system. Although some micropro-

[1] Graduate School of Science and Technology, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8577 Japan
[2] Faculty of Engineering, Information and Systems, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki 305-8577 Japan
[a)] nakagawa.ryo.ni@lila.cs.tsukuba.ac.jp
[b)] yoshiki@cs.tsukuba.ac.jp

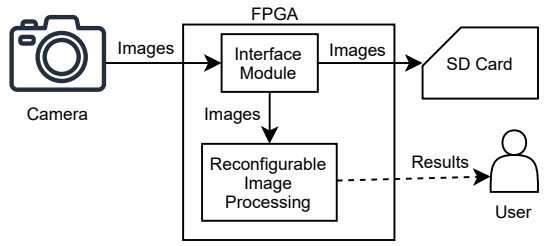

Fig. 1 Block diagram of proposed system

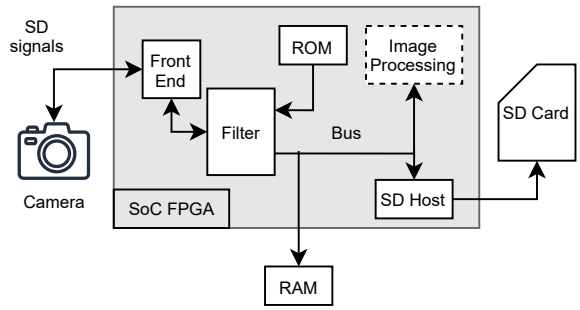

Fig. 2 The detailed block diagram of the discussing system

cessors have an SD host interface, there are no microprocessors or ASICs on the market which have an SD client (card) interface. Since the SD standard interface is a low-level serial interface, a dedicated logic circuit is required. Therefore, the FPGA is chosen to implement the SD card functionality. Moreover, we use the FPGA with new architecture because its power efficiency is superior to conventional SRAM architecture [3]. However, in this paper, off-the-shelf FPGA is used to evaluate the functionality of our design. Furthermore, the commodity camera is adopted to utilize its availability in the market and transplantability.

The remaining part of the paper proceeds as follows. The data acquisition architecture with SD interface and FAT file system is described in section 3. Section 4 presents the implementation of the system. Evaluation of the system and results are presented in section 5. Finally, conclusions are described in section 6.

2. The brief overview of a proposed system

Fig. 1 shows the relations of the main functions of the proposed system below.

- Acquire images from the commodity digital camera in real-time
- Save the acquired data to the SD card
- Real-time image processing
- Send the result to the ground by a communication equipment
- Processing is reconfigurable offline, and the camera is transplantable

In this paper, we addressed the two functions: “Acquire images from the commodity digital camera in real-time” and “Save the acquired data to the SD card”. These two are the key functions to proceed with the drone system because it is a frontend of an image processing circuit. To implement them, we focused on the SD interface, which most commodity digital cameras have. However, it is not designed specifically for image transferring, so we designed a circuit that acts as an SD card from the camera’s viewpoint.

3. Data Acquisition Architecture Design

3.1 Overview

In this section, the data acquisition system design is described. Three functions are required in this system. The first is the SD interface between the camera and FPGA. The second one is the availability to transfer acquired image data to the onboard image processing modules. The third one is the SD host function to save image data to the attached SD card.

Fig. 2 shows the block diagram of the discussing system. The front-end module in the diagram works as the SD card and interface between the camera and FPGA. Image processing is not discussed in this paper, so image data transferring is verified by writing acquired photos to the RAM. The filter, ROM, and RAM in the figure are used to realize the file system.

3.2 SD Front-end

The utilization of the SD standard interface and protocol is the key topic in this paper. **Table 1** shows the functions and direction of the SD card signals. It consists of six signals: one clock line (CLK), one command line (CMD), and four data lines (DAT0-3). The command and data lines are serial communication lines, which are synchronized with the clock. The communication direction of bi-directional signals is switched by the tri-state buffers.

Table 1 The function and direction of SD signals

Signal Name	Function	Direction (at card)
CLK	supply clock to the card	Input
CMD	command and response	Input/Output
DAT0-3	data communication	Input/Output

CRC (Cyclic Redundancy Check) is required to check the integrity of command, response and data. The length of the CRC code in command and response is 7-bit, and in data packet, it is 16-bit. When a CRC error is detected, the card sets a flag in its status register. FSM (Finite State Machine) defines card operations. Some commands are used to change the card state. The command and data shifter is a long shift register to serialize and deserialize the received packets. The received command and data are stored in the command shifter and the data shifter respectively to deserialize the data. The shifter is also used when generating a response packet or read data packet as the serializer. Parameters of the card (e.g. speed class) and the card status (e.g. current FSM state, CRC error flag) are stored in the card register. Finally, This interface is connected to the internal AXI bus of the FPGA to store the received data.

3.3 File System

When there is no file system on the SD card memory, the camera fails to initialize the card. Therefore in most cases, the file system such as FAT16, FAT32, and exFAT is implemented on the SD memory. In this paper, FAT16 was selected for its simplicity. **Fig. 3** shows the structure of the FAT16 file system. There are four major areas in that file system. These are reserved area, FAT area, root directory area, and data area. The reserved area

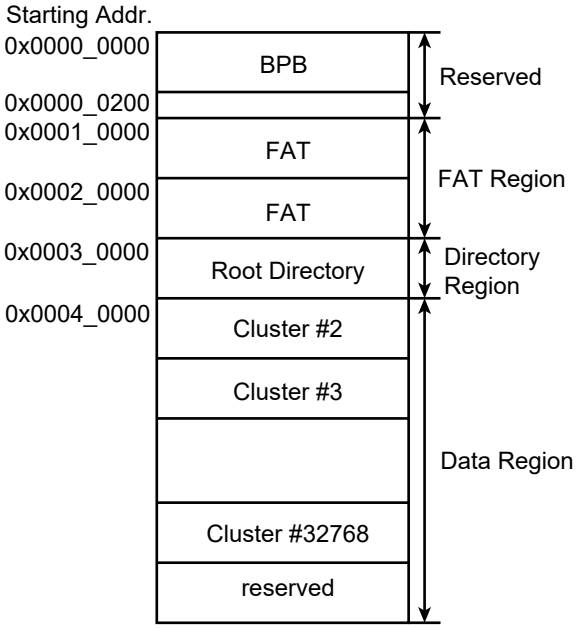

Fig. 3 Structure of FAT16 file system [4]

includes BPB (BIOS Parameter Block) and stores the basic parameter of the volume, such as size. The FAT area manages the relation between multiple clusters in the data area in the form of a linked list. The root directory area stores the metadata of files and directories on the root directory on the SD card volume. The data area consists of multiple clusters, and stores files or subdirectories itself.

However, as we are assuming the FPGA with less size RAM, we can not use the large area of memory to implement the whole file system. So we use a minimum size RAM by separating one file system into two memory areas: ROM area and RAM area, and filtering the access. The reserved, FAT, and root directory area is stored on the ROM area.

The detailed behavior of the file system depends on the platform, so it is described in the implementation section (section 4.1.2)

4. Implementation

We used Zynq-7020 SoC (System on Chip) FPGA (Xilinx) and ZYBO Z7-20 board (Digilent). This FPGA consists of a reconfigurable part, PL (Programmable Logic), and a dual-core ARM Cortex-A CPU and its peripheral, PS (Processing System). The SD front-end, the filter, and ROM are implemented on PL, and the SD host module in PS is used to write the image data to the separately connected SD card. As the FPGA board has 512MB DDR3 DRAM, the RAM on Fig. 1 is implemented on that. In addition, CPU in PS is used to control the system's behavior. Vivado 2019.2 (Xilinx) and Vitis 2019.2 (Xilinx) are used for hardware and software development respectively.

4.1 Hardware Implementation

4.1.1 SD Front-end Implementation

The SD front-end module in this paper is based on the SDHC module used in Google's open-source project named "Vault"[5].

Fig. 4 shows the interface circuit of the SD interface around

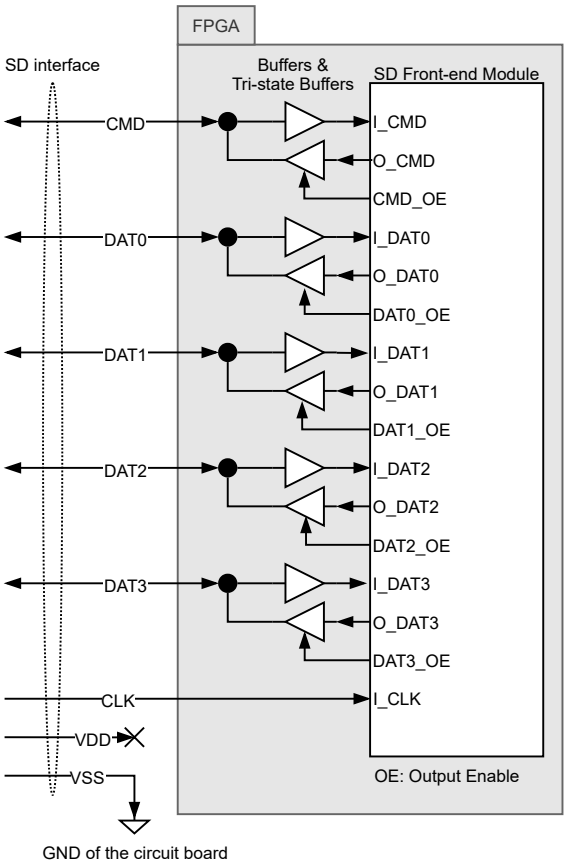

Fig. 4 The interface circuit of the SD interface

the FPGA board. As described in section 3.2, SD standard interface has bi-directional signals, so tri-state buffers on FPGA are used to toggle the signal direction. The VSS line of the SD interface is connected to the FPGA board's ground (GND), and the clock from the camera is supplied to the FPGA. However, since the camera supplies power to the card, the VDD line of the SD interface was cut to avoid interference of the power.

The detailed structure of the SD front-end module is shown in **Fig. 5**. There are two clock regions. One is driven by an onboard clock source, and the other is driven by an SD clock supplied from a host (camera). To transfer data between two regions, CDC (Clock Domain Crossing) architecture is implemented to avoid the metastable state of the flip-flop.

The SD host must initialize the card before accessing the data. The initialization sequence is described in the SD standard specification, and it consists of initialization and identification of the card. During this sequence, the camera issues some commands and gets responses from the card to read out card information, for instance interface speed, card status register, CID (Card IDentification) register and card address. **Fig. 6** shows the flowchart of the actual initialization sequence of the camera we used. This flowchart includes commands which are not defined in the SD specification, such as CMD52 and CMD5. These commands are for SDIO standards, therefore the card is not required to respond to these commands.

The read and write operations of the SD are also controlled by the commands. First, the host issues CMD18 (`READ_MULTIPLE_BLOCK`) to read, and CMD25

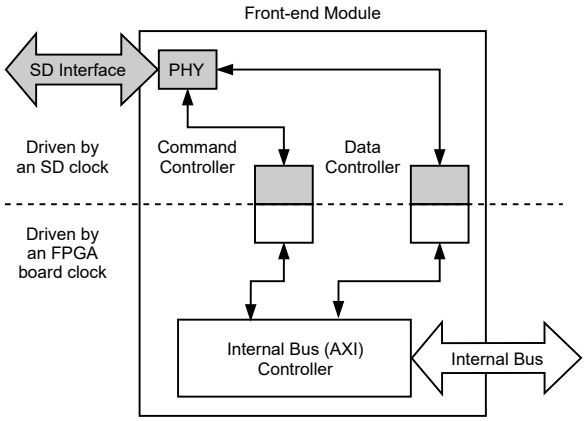

Fig. 5 The detailed structure of the SD front-end module

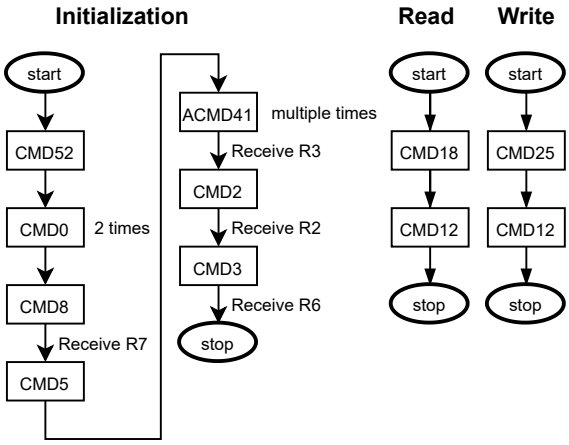

Fig. 6 Flowchart of initialization, read and write operations

(`WRITE_MULTIPLE_BLOCK`) to write the data. The memory address is specified in the command as an argument. Then, the card returns the response for the command, and simultaneously the data is transmitted using four data lines. When the card receives those multiple-block operation commands, it continuously performs read or write operations until CMD12 (`STOP_TRANSMISSION`) is issued.

In the CSD register, which contains information on the performance of the card, the speed class is set to 4, and the TAAC, a parameter related to the data access time, is set to 2.0 ms. As a result, the camera recognizes the card as a low-speed card. This is because it was found that when the card was set as a high-speed card, a CRC error occurred in the command and response when the clock frequency was switched to 50 MHz during the initialization phase. When a CRC error occurs, the camera recognizes the card as abnormal, and subsequent communication becomes impossible.

4.1.2 File System Implementation

The filter module has two states: “Standby” and “Image Writing”. The standby state is used to access the BPB, FAT, and root directory area of the FAT16 file system. In this state, read access gets values stored in ROM, and all write access is discarded. The image writing state is used to write the acquired image data to RAM. In this state, read access always gets zero and written

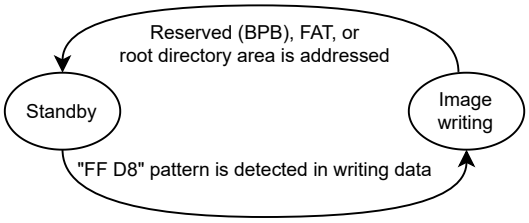

Fig. 7 State machine of filter module

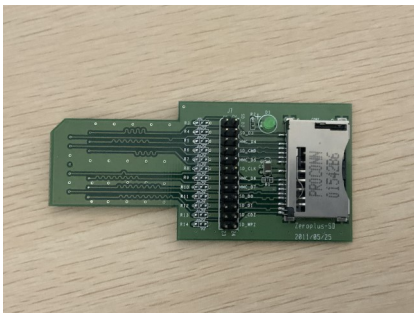

Fig. 8 SD card adapter

data is transferred to DRAM. **Fig. 7** shows the state machine of the filter module. The transition condition is based on the actual behavior of the SD card attached to the camera. To understand this behavior, we used an SD card adapter (**Fig. 8**) and a logic analyzer (Zeroplus LogicCube Pro) and got the following image to write access sequence of the camera:

(1) Read the directory entry and identify the file name to be saved
(2) Find the starting cluster for the directory ”101___01”
(3) Based on the information in the root directory area, write the data in the appropriate data area cluster
(4) Write to the two FAT areas and label the cluster where the image was written as in use
(5) Write the file name and other information to the directory entry of the newly created image file to complete the file creation

The filter module also includes the word size counter. This counts the number of words written during the image writing state and notifies its value to the CPU when the state changes to the standby state by activating the interrupt line.

The resource utilization of our system is shown in **Table 2**.

Table 2 Resource utilization of FPGA

Resources	Used	Available	(%)
LUT	6019	53200	11.31
LUTRAM	705	17400	4.05
FF	9204	106400	8.65
BRAM	32	140	22.86
IO	11	125	8.80
BUFG	4	32	12.50

4.2 Software Implementation

The software running on the CPU has three basic functions. Firstly, it initializes the hardware at start-up time. Secondly, it handles interrupt signals from the filter module. Thirdly, it calculates the image size. Fourthly, it writes the image file to the SD

Fig. 9 Illustration of scanning to detect the end of image file

card.

Here, we describe the calculation of the image size in detail. As mentioned above, the filter module counts the number of words written. However, as the written data is aligned with the cluster boundaries of the FAT file system, an additional process is required to get the precise size of the image file. Therefore, DRAM is manipulated one byte at a time in the direction of decreasing memory address as shown in the **Fig. 9** to find the "FF D9" pattern, which indicates the end of JPEG and RAW images.

5. Evaluation

To confirm our system's functionality, an evaluation of the image acquisition system is required. The evaluation is performed by following two steps: Firstly, we confirm the SD standard initialization sequence. Secondly, we confirm the transport of shot images to the SD card. Thirdly, we confirm image data integrity.

The FPGA board we used has an external expansion connector called Pmod, and in this study, we used it with an SD card slot board attached. We used a flat cable called "extension cable" which has SD card-shaped terminals on both ends as shown in **Fig. 10** and inserted one end into the SD card slot of the FPGA board and the other end into the SD card slot of the digital camera.

First, we programmed the design into the FPGA and the camera started its initialization sequence when it was powered on. The system was then confirmed to be recognized as a 1.9GB SD card.

Next, after the initialization sequence was completed, we took a picture with a digital camera set to JPEG shooting mode. The LED on the FPGA board lit up to confirm that the DRAM was being written. In addition, the system was recognized as a normal card by the camera even after taking a picture.

The acquired image was confirmed its integrity using the image viewer software and binary editor.

6. Conclusions

This article proposed a transplantable agricultural drone system with onboard image processing. In this system, image processing is reconfigurable with FPGA, and the camera is replaceable using a commodity digital camera and SD standard interface. Besides, we designed a mechanism to acquire images from the camera by emulating an SD card, and it worked well as an image acquisition system. The new ASICs for the proposed system are in production, and our design is embedded as a bridging module between the physical interface and reconfigurable circuit area. The power consumption of the system we implemented in this paper will be evaluated in future works. The proposed drone system will also be implemented on an SoC and work in an actual field.

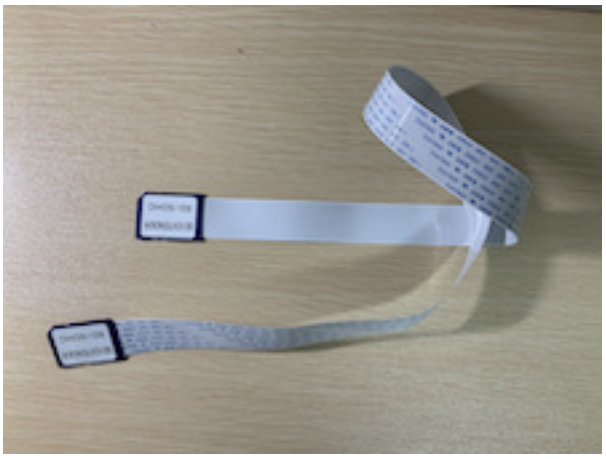

Fig. 10 A picture of SD extention cable

Acknowledgement

This work was supported in part by TIA collaborative research program "KAKEHASHI" in FY2020 and FY2021. We also thank the Xilinx University Program for the kind donation of software tools.

References

[1] TANAKA, K. and KONDOH, A.: Mapping of Rice Growth using Low Altitude Remote Sensing by Multicopter, *The Journal of the Remote Sensing Society of Japan*, Vol. 39, No. Journal Article, pp. S1–S17 (online), DOI: 10.11440/rssj.39.S1 (2019).

[2] Holman, F. H., Riche, A. B., Michalski, A., Castle, M., Wooster, M. J. and Hawkesford, M. J.: High Throughput Field Phenotyping of Wheat Plant Height and Growth Rate in Field Plot Trials Using UAV Based Remote Sensing, *Remote Sensing*, Vol. 8, No. 12, p. 1031 (online), DOI: 10.3390/rs8121031 (2016).

[3] Nebashi, R., Banno, N., Miyamura, M., Bai, X., Funahashi, K., Okamoto, K., Iguchi, N., Numata, H., Sugibayashi, T., Sakamoto, T. and Tada, M.: A 171k-LUT Nonvolatile FPGA using Cu Atom-Switch Technology in 28nm CMOS, *2020 30th International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 323–327 (online), DOI: 10.1109/FPL50879.2020.00060 (2020).

[4] ELM: FAT Filesystem, (online), available from ⟨`http://elm-chan.org/docs/fat_e.html`⟩ (accessed 2021-10-15).

[5] ProjectVault: ORP, (online), available from ⟨`https://github.com/ProjectVault/orp`⟩ (accessed 2021-10-15).